N-Body simulations of star-star encounters

150193993

Abstract

N-Body simulations are carried out on the Solar System by developing a complex fourth order predictor-corrector method. The main code is constructed in a block manner with different tasks like the Bootstrapping, Predictor-Corrector, Adaptive timestep and Downshifting. The second order code created in the first semester is included in bootstrapping. In comparison to the second order code, the fourth order method allows small timesteps but not resulting in long computational time. With a higher accuracy obtained, the simulations are carried out for different times of 1000 yr, 100,000 yr and 1 Myr. The stability of the system is checked using the fractional energy and the orbital distance calculations. Simulations show the system being unstable due to the rise in the fractional energy. The source for the errors in the code is checked via a brute force method. These code checks reveal problems in the adaptive timestep block. The aim for producing Myr simulations was to observe the periodic glacial cycles occurring from the variations of eccentricity of Earth's orbit.

1. Introduction

Stars like our Sun are born in groups/clusters with other stars. Gravitational interactions in the clusters cause changes in the dynamical properties of the members. The evolution of this group of stars can be produced in N-Body simulations. In this project, N-Body simulations are carried out on the Solar System. Across the two semesters, two methods were used which provided accuracies.

Star clusters are important targets in astronomical observations. An observation of the cluster represents a snapshot in time. Detailed spectroscopic observations can help in producing an H-R diagram for the cluster. This plot provides important information about the cluster like the different populations of stars, the age of the cluster etc. However, to look at how the cluster will vary with time under the influence of gravity, theoretical simulations represent the perfect choice. Here, by knowing the initial values of the of the dynamical properties of the members of the Solar system, we perform N-Body simulations.

Two methods were used in this project over the two semesters. In the first semester, a second order Euler's method was employed. At the base this method lies two first order differential equations for acceleration and velocity. The Euler's method obtains the future position/velocity as an approximate solution of the differential equations. The errors occurring from this method go as dt^2. Higher accuracy is obtained with a smaller timestep, with a consequence of long computational time. Therefore, a higher order method is used to give a higher accuracy. A fourth order predictor-corrector method is applied in this semester for the N-Body simulations.

2. Method

The system we consider here is our Solar System with all the planets except Mercury. N-Body simulations are done on this system with an eye on monitoring its evolution due to the gravitational interactions. The code is developed in a block structure. This made the job for testing the code easier and efficient. The first part of the code involves calculating the initial acceleration of the all the members of the system. In addition to that, the initial total energy of the Solar System is also determined which will help in doing the energy conservation check in the later part. The motivation for developing this complex fourth order code lies in producing the periodic features called the Milankovitch cycles. These cycles describe the effects of the changes in the eccentricity and inclination of the Earth's orbit on the climate. Periodic variations in the eccentricity, axial tilt and the precession represent the three dominant cycles. With a periodicity of 100,000 yrs (Muller 1995), the periodic glacial cycle originates from the varying solar insolation which is result of the variability in the eccentricity.

The important cog in the code is the Bootstrapping. The fourth order predictor-corrector method employed here requires past information of the system in order to determine the future points. The second order code used in the previous semester is drafted here. This section is followed by the predictor-corrector block. A time loop is created with the predictor-corrector, adaptive timestep and the energy conservation check included. The method used was the Adams-Bashforth-Moulton predictor-corrector (Conte and De Boor 1972). This predictor-corrector represents a multi-step method which requires four starting values for the predictor step. Having the predicted position and velocity, the predicted acceleration is determined. The corrector uses the predicted value along with three past points to provide the corrected position and velocity. This is followed by calculation of the fractional energy and the orbital distances for the system.

Another important feature of this code comes in the adaptive timestep section. Using the predicted and corrected values, error calculation is done. This error is compared against a relative error and the timestep either is doubled or halved. A minor difference between the two cases is that new interpolated steps are introduced when the timestep is halved. Finally, the end of the code involves the part of refreshing the arrays of the position, velocity and acceleration. The last element in the array with the index, $(l-8)$ is overwritten by the one at $(l-7)$. Also, the future value at $(l+1)$ slots in at $(l=0)$ step which makes it the current element.

2.1. Bootstrapping

The code for the fourth order predictor corrector method is divided into blocks for ease in error checking. Newton's law of gravity is the main theory behind these simulations of the Solar System. The second order code is used as a part of the fourth order predictor-corrector method. Below are the equations for the second order code block.

$$\mathbf{a} = \sum_{i=1, i\neq j}^{N} \frac{Gm_j}{\left|\mathbf{r}_{ij}\right|^2}\widehat{\mathbf{r}_{ij}} \quad \text{-1}$$

$$\mathbf{r_1} = \mathbf{r_0} + \mathbf{v_0}dt + \frac{\mathbf{a_0}}{2}dt^2 \quad \text{-2}$$

$$\mathbf{v_1} = \mathbf{v_0} + \frac{\mathbf{a_0} + \mathbf{a_1}}{2}dt \quad \text{-3}$$

where,

N - Number of bodies in the system

m_j - mass of the second object (in Kg)

$\widehat{\mathbf{r}_{ij}}$ - unit vector of the distance vector between two bodies

$\mathbf{r_1}, \mathbf{r_0}$ - current and future positions of the body (in m)

$\mathbf{v_1}, \mathbf{v_0}$ - current and future velocities of the body (in m s^{-1})

$\mathbf{a_0}, \mathbf{a_1}$ - current and future accelerations of the body (in m s^{-2})

For the next part of the project, this second order code will provide the past information. Eight past steps are generated which are then bootstrapped into the predictor-corrector code.

2.2. Predictor-Corrector

In comparison to the second order code used in the first semester, a fourth order predictor-corrector scheme allows for usage of small timesteps but not compensating in long computational time. In addition, higher accuracy is obtained from the fourth order predictor-corrector method.

A basic predictor-corrector method works on the basis of two steps:

a) Predictor- This uses current value of the variable, y_l in determining a predicted value, y_{l+1}^P approximately.
b) Corrector- With the predicted value, y_{l+1}^P and y_l, the corrected value at the $(l+1)^{\text{th}}$ step, y_{l+1}^C can be calculated.

In this project, the Adams-Bashforth-Moulton predictor-corrector (Conte and De Boor 1972) is applied. Being a multistep method, four starting points are required in the calculation of the future value. The predicted position, $\mathbf{r}_{l+1}^{P}$ and velocity, $\mathbf{v}_{l+1}^{P}$ is determined using the formulae below.

$$\mathbf{r}_{l+1}^{P} = \mathbf{r}_{l} + \frac{dt}{24}(-9\mathbf{v}_{l-3} + 37\mathbf{v}_{l-2} - 59\mathbf{v}_{l-1} + 55\mathbf{v}_{l}) \quad \text{-4}$$

$$\mathbf{v}_{l+1}^{P} = \mathbf{v}_{l} + \frac{dt}{24}(-9\mathbf{a}_{l-3} + 37\mathbf{a}_{l-2} - 59\mathbf{a}_{l-1} + 55\mathbf{a}_{l}) \quad \text{-5}$$

where,
$\mathbf{v}_{l-3}$, $\mathbf{v}_{l-2}$, $\mathbf{v}_{l-1}$, $\mathbf{v}_{l}$ and $\mathbf{a}_{l-3}$, $\mathbf{a}_{l-2}$, $\mathbf{a}_{l-1}$, $\mathbf{a}_{l}$ – positions and accelerations which resemble the past information

Now, the predicted acceleration, $\mathbf{a}_{l+1}^{P}$ is determined by using $\mathbf{r}_{l+1}^{P}$ and $\mathbf{v}_{l+1}^{P}$ in an acceleration loop which uses equation (1).

With the ingredients obtained, the corrected position, $\mathbf{r}_{l+1}^{C}$ and velocity, $\mathbf{v}_{l+1}^{C}$ is evaluated from the formulae given below.

$$\mathbf{r}_{l+1}^{C} = \mathbf{r}_{l} + \frac{dt}{24}(\mathbf{v}_{l-2} - 5\mathbf{v}_{l-1} + 19\mathbf{v}_{l} + 9\mathbf{v}_{l+1}^{P}) \quad \text{-6}$$

$$\mathbf{v}_{l+1}^{C} = \mathbf{v}_{l} + \frac{dt}{24}(\mathbf{a}_{l-2} - 5\mathbf{a}_{l-1} + 19\mathbf{a}_{l} + 9\mathbf{a}_{l+1}^{P}) \quad \text{-7}$$

where,
$\mathbf{v}_{l-2}$, $\mathbf{v}_{l-1}$, $\mathbf{v}_{l}$, $\mathbf{v}_{l+1}^{P}$ and $\mathbf{a}_{l-2}$, $\mathbf{a}_{l-1}$, $\mathbf{a}_{l}$, $\mathbf{a}_{l+1}^{P}$ become the past information given to the corrector.

As done after the predictor, the corrected acceleration, $\mathbf{a}_{l+1}^{C}$ is found from the corrected position and velocity. This predictor-corrector block is included in an infinite time loop which runs for a simulation time, t_{out}.

Energy conservation check is then carried out to keep a track on the accuracy of this system. A counter was placed at this point, which forced the fractional energy to be calculated every six months. The corrected positions and velocities were used in this calculation as per the equations given below.

$$KE = \frac{1}{2}\sum_{i=1}^{N} m_i |\mathbf{v}_i|^2 \quad \text{-8}$$

$$PE = -Gm_i \sum_{j=0, j\neq i}^{N} \frac{m_j}{|\mathbf{r}_{ij}|}$$

-9

where,

$$|\mathbf{v}_i|^2 = v_{ix}^2 + v_{iy}^2 + v_{iz}^2$$

$\mathbf{v}_i$ - corrected velocity of the object, $\mathbf{v}_{l+1}^C$

$$\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$$

$\mathbf{r}_i$, $\mathbf{r}_j$ - corrected positions of the objects.

The fractional energy is then given by,

$$\Delta E = \frac{E_{curr} - E_{init}}{E_{init}} \qquad \text{-10}$$

where,

E_{curr} – current total energy of the system at the particular iteration

$$E_{curr} = KE + PE$$

E_{init} - initial total energy of the system

Along with the determination of the fractional energy, the orbital distance of the planets was calculated. This gives an additional idea on the stability of the system.

$$d_{sun} = \left(r_x^2 + r_y^2\right)^{\frac{1}{2}}$$

where,

r_x, r_y - magnitude of the x, y component of the corrected position, $\mathbf{r}_{l+1}^C$

2.3. Adaptive timestep and error check

This method has an advantage which allows for the timestep to vary to result in better accuracy. At each iteration, the predicted and corrected values are used in determining an error estimate. The estimated error is compared against a relative error, RelErr. A value of $5.0 * 10^{-6}$ is taken in the beginning. The formula below explains the calculation of this error, called as error factor, 'Errfact'

$$\text{Errfact} = \frac{19}{270}\left(\frac{|y_{l+1} - p_{l+1}|}{|y_{l+1}| + \text{Small}}\right) \qquad \text{-11}$$

where,
y_{l+1} and p_{l+1} are the corrected and predicted values for the x/y/z component of the position/velocity of a body respectively.
Small - an offset used in the calculation with a fixed value of 10^{-5}.

The values for the position and velocity are considered in the error calculation. For each direction component of the position and velocity, the error factor is determined. The maximum value among the three directions is taken for each body. With these values obtained for all the bodies, a maximum of those is taken for the position and velocity. This ends with two final contributions to the error from the position and velocity. Finally, the larger among the two errors is compared against the 'RelErr'.

Two cases are considered in this comparison with the relative error.

Case 1: $$\text{Errfact} > \text{RelErr}$$
$$dt = 0.5 * dt$$

On halving the timestep, new interpolated steps, $\mathbf{v}_{l-1/2}$ and $\mathbf{v}_{l-3/2}$ are created as shown below,

$$\mathbf{v}_{l-1/2} = \frac{-5\mathbf{v}_{l-4} + 28\mathbf{v}_{l-3} - 70\mathbf{v}_{l-2} + 140\mathbf{v}_{l-1} + 35\mathbf{v}_{l}}{128} \quad \text{-12}$$

$$\mathbf{v}_{l-3/2} = \frac{3\mathbf{v}_{l-4} - 20\mathbf{v}_{l-3} + 90\mathbf{v}_{l-2} + 60\mathbf{v}_{l-1} - 5\mathbf{v}_{l}}{128} \quad \text{-13}$$

These formulae are similar for getting the interpolated steps for acceleration ($\mathbf{a}_{l-1/2}$ and $\mathbf{a}_{l-3/2}$). The four past steps used by the predictor are $\mathbf{v}_l$, $\mathbf{v}_{l-1/2}$, $\mathbf{v}_{l-1}$, $\mathbf{v}_{l-3/2}$.

Case 2: $$\text{Errfact} < \frac{\text{RelErr}}{100}$$
$$dt = 2 * dt$$

Doubling the timestep represents the easier task as interpolated past steps are not required. So, the first four past steps which will be used by the predictor are, $\mathbf{v}_l$, $\mathbf{v}_{l-2}$, $\mathbf{v}_{l-4}$, $\mathbf{v}_{l-6}$

3. Results

The different blocks of the code are tested individually. At first the adaptive timestep is commented out. The timestep is set at 500 s with the t_{out} for the simulations being 1000yrs. The fractional energy of the system calculated was plotted against time.

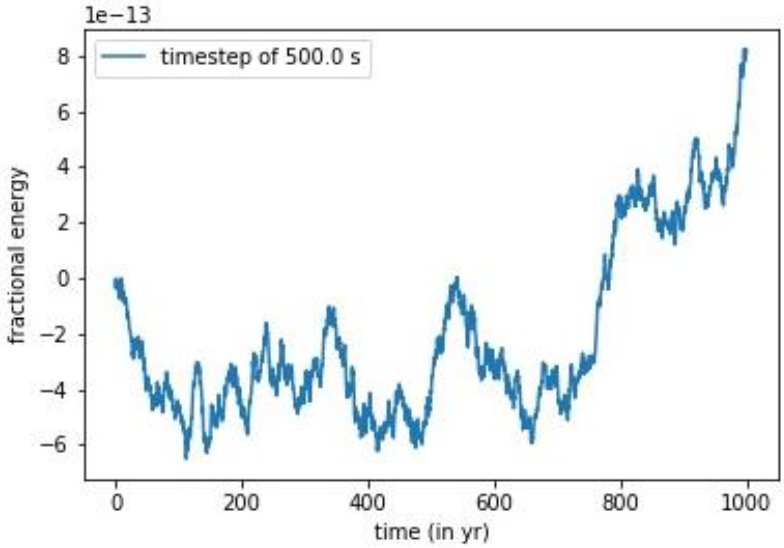


Figure 1: Fractional energy of the system for 1000 yr simulations with $dt = 100$ s

Figure 1 shows how the fractional energy varies from $-6 * 10^{-13}$ to $8 * 10^{-13}$. The system is found to be stable from this plot, thus indicating that the predictor-corrector is working. However, prior to this result the system showed instability with the fractional energy rising rapidly to a value close to 2.0. This led to an error check in the code. For this simulation time, the Kinetic and Potential energies were compared against each other. The PE started to rise in contrast to the KE remaining roughly constant. By backtracking in the goal of finding the source of the error, the predicted positions and velocities were found to be increasing rapidly. On further checks, this error was arising due to a problem in the downshifting of the arrays.

Now, the adaptive timestep is included in the main code. Results showed that the orbits of the outer planets remain stable in contrast to those of the inner planets. Figure 2 shows the evidence of these variations in the orbits of the inner planets.

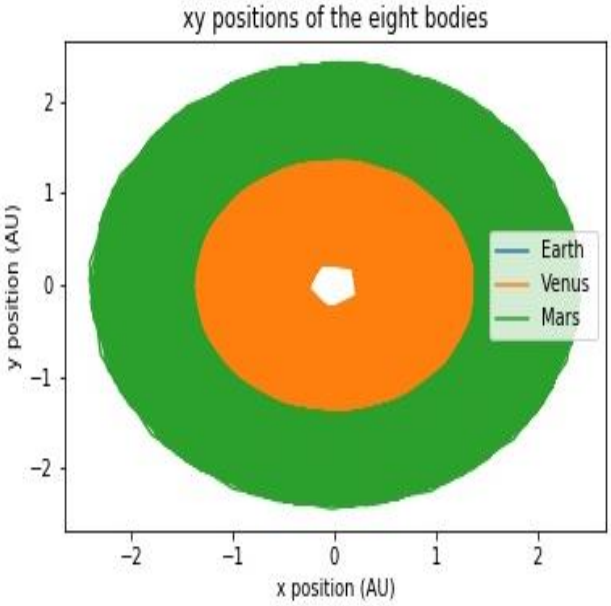


Figure 2: Orbits of the inner planets produced from 1000 yr simulations with initial $dt = 500$ s

In a try to correct the error, the timestep is reduced to 10 s and the simulations are carried out. However, similar problems emerge for the orbits of the inner planets. The energy conservation of the system is also checked at this point. The fractional energy was found to have values in the range of 10^{-4} to 10^{-2} which is unexpected for a stable system. In order to find the source for this logical error, each block is checked through brute force. The search was narrowed down to two sections: Downshifting and Adaptive timestep. The downshifting worked along with the case of halving the timestep. However, the error was found to occur in the case of doubling the timestep. On doubling the timestep, there will be four past steps at indexes $l-8, l-7, l-6, l-5$ which cannot be used for the next iteration. This meant that the timestep cannot be doubled consecutively. So, to refresh the past steps, a counter is included in the code which allows the timestep to be doubled after every four iterations.

Now, the initial timestep is set at 1.0 s and the fractional energy of the system is checked for the planets. However, the fractional energy shows a rise to a value of 10^{-4} which indicates a bad accuracy. The timestep is raised to 1000 s and the orbits of the planets are also checked. Below are the plots of the fractional energy and planetary orbits. The value for the relative error and the simulation time is kept same.

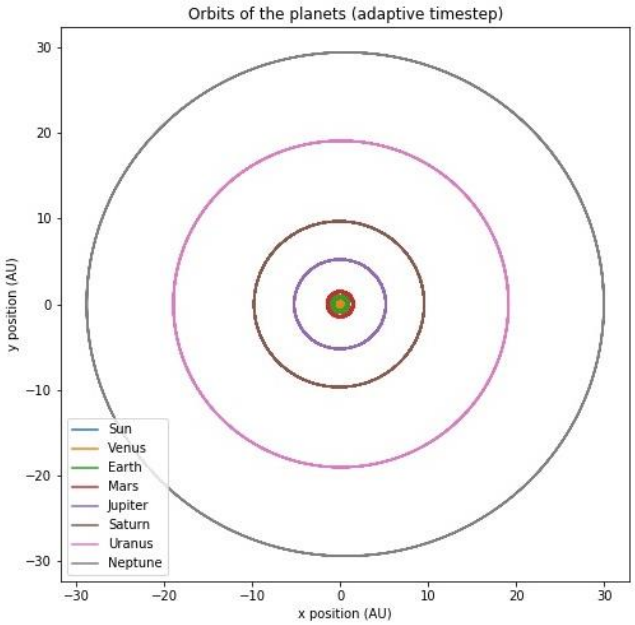


Figure 3: Orbits of all planets in 1000 yr simulations with initial $dt = 1000$ s

Figure 3 shows the orbits of all the planets of the Solar System from 1000 yr simulations. Along with the energy loss of the system, the plot of orbital distances of the planets is produced. This increase in the fractional energy of the system can be found here. Figure 4 provides the evidence of the increase in the orbital distances for the inner planets.

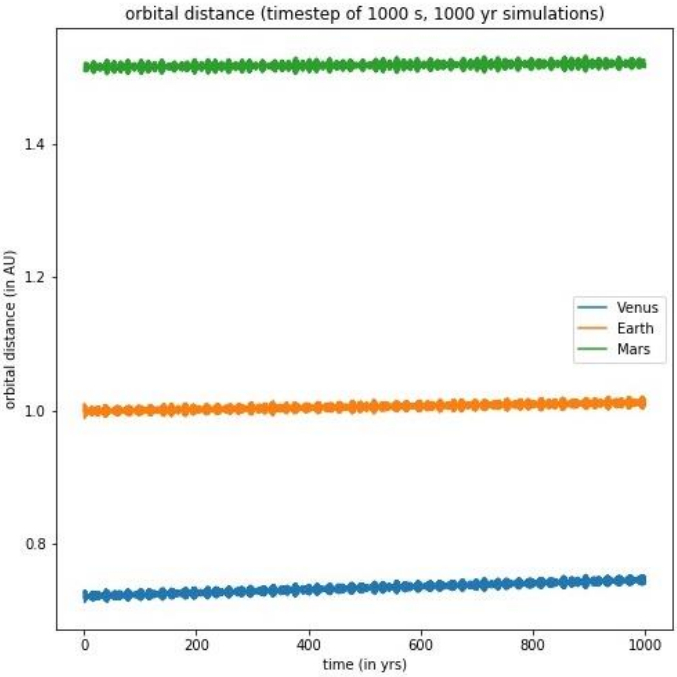


Figure 4: Orbital distances from 1000 yr simulations of the Solar System

In order to observe Milankovitch cycles in the orbital distances, the simulation time is increased to 1 Myr and the timestep is dropped to 1 s. Milankovitch cycles represent a periodic activity happening every 100,000 yrs. To observe these features, the orbital distances of all the planets are produced. Since the code had an error in the adaptive timestep part, the periodic cycles could not be identified from the curves in the figure.

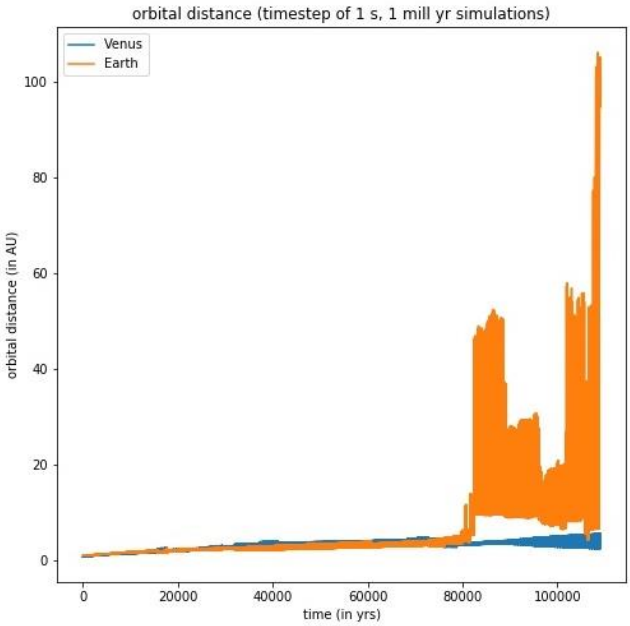


Figure 5: Orbital distances for the Earth and Venus against time. The simulations were stopped few years above 100,000 yrs. This was done since unreal results were obtained for the orbital distance.

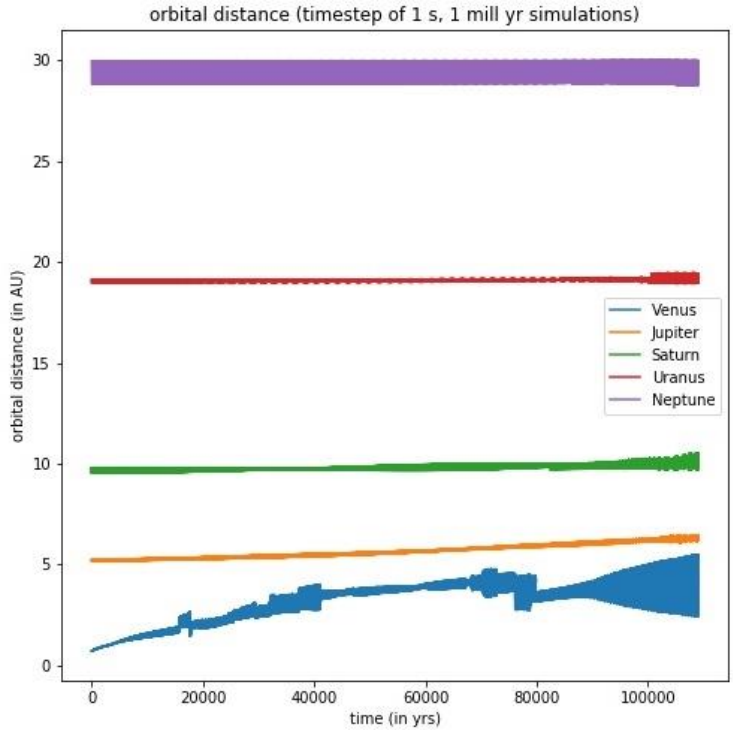


Figure 6: Orbital distances for the planets (except Earth) against time

Figure 5 includes the orbital distances of the Earth and Venus varying with time. The curve for Earth shows huge deviations close to 100,000 yrs. The Milankovitch cycles could not be notified. On looking at figure 6, the curve for Venus shows abnormal during the simulation. Both these figures prove that the code does not completely work in order to produce Milankovitch cycles in the orbital distance curves.

The results produced from the tests proved the code had minor faults. From carrying out brute force checks throughout the sections of the code, the error was found to occur from the scenario of doubling the timestep.

4. Conclusion

N-Body simulations were carried out the Solar System with the fourth order predictor-corrector method applied. Different simulation times were considered to look at the energy conservation for the system. The second order code used in the first semester was included in the bootstrapping for the main code. Each section of the code, like the predictor-corrector, adaptive timestep, downshifting etc. were checked for any faults. To consider the stability of the system, energy conservation check was done using the fractional energy calculations. In addition to that, the orbital distances provided a second stability check on the system.

The motivation for constructing this complex code rested in understanding how a group of bodies (in this case, the planets) evolved over time while interacting only gravitationally. The consequence of these N-Body simulations on the Solar System was in observing the periodic features called the Milankovitch cycles every 100,000 yrs approximately. The final code had faults in the adaptive timestep block. From the simulations, the system was found to be unstable due to the rise in the fractional energy to value of 10^{-4} over time. The source of the error was identified through brute force checks on the different parts of the code. This error could not be corrected even after several tries. On trying to compare with the future plan developed in the first semester, certain sections took longer time than expected. The testing of the main code took longer than the expected amount of time. This caused delays in developing the final report for the project.

5. References

Conte, S. D. and De Boor, C., (1972). *Elementary numerical analysis: an algorithmic approach*. 2nd ed. New York: McGraw-Hill

Goldstein, M.E. and Braun, W.H., (1973). *Advanced Methods for the Solution of Differential Equations*. Washington, D.C.

Muller, R.A. and MacDonald, G.J., (1995). Glacial cycles and orbital inclination. *Nature*, **377** (6545), pp.107-108.

Appendix

A: Final code used for the project was built by the fortran 90 programming language.

```
program nbodyfour
implicit none
!fourth order predictor-corrector method used here

!declaration
double precision r(1:3,1:8,-8:1),v(1:3,1:8,-8:1),a(1:3,1:8,-8:1),m(1:8),dt,tcurr,tout,G
double precision r_newpast(1:3,1:8,-3:-1), v_newpast(1:3,1:8,-3:-1),
a_newpast(1:3,1:8,-3:-1)  !these arrays store the new position, velocity and
acceleration produced when the timestep is halved. They can be considered as t=-
1/2 step
double precision r_c(1:3,1:8,1), v_c(1:3,1:8,1),a_c(1:3,1:8,1) !r_c, v_c, a_c store
the future corrected values
double precision dr(1:3), rad2, AU, yr, com(1:3), cov(1:3), mtot, KE, PE, Ei, Ecurr,
delE, rad, tcount, dsun(1:8)
double precision day, relerr, small, err_fact_r(1:3,1:8), err_fact_v(1:3,1:8),
err_max_r(1:8), err_max_v(1:8)
double precision err_r, err_v
integer i,j,k,n,p,counter

!---------------------------------------
!number of objects
n = 8
!a day
day = 24.*3600.
!Year (in s) and a astronomical unit (in m)
yr = 365.*24.*3600.
AU = 1.5e11

!Orbital distances for Sun-Jupiter-Earth-Venus-Mars-Saturn-Uranus-Neptune (in m)
r(1,1:n,0) = (/0., 778.6e9, 149.6e9, 108.2e9, 227.9e9, 1433.5e9, 2872.5e9,
4495.1e9/)
r(2:3,1:n,0) = 0.

!Mass of the bodies (in Kg)
m(1:n) = (/1.989e30, 1.898e27, 5.97e24, 4.87e24, 0.642e24, 568.e24, 86.8e24,
102.e24/)
!Total mass of the system (in Kg)
mtot = 0.

!Orbital velocities of the bodies in the system (in m/s)
v(2,1:n,0) = (/0., 13.1e3, 29.8e3, 35.0e3, 24.1e3, 9.7e3, 6.8e3, 5.4e3/)
v(1,1:n,0) = 0. !forcing the objects to orbit in the XY plane
v(3,1:n,0) = 0.
```

```
!Gravitational constant (in N m^2 Kg^-2)
G = 6.67e-11

!Timestep (in s)
dt = 1.
!Current time (in s)
tcurr = 0.
!counter for doubling the timestep
counter = 0
!Time for all simulations to run (in s)- setting 10. s so can get enough future steps
which then will be shifted to past steps
tout = 1.e6*yr
!Counter for the simulations (in s)
tcount = 0.

!Current distance from the sun (in m)
dsun(1:n) = 0.

!relative error criterion for predictor-corrector (from the Adams-Bashforth-Moulton
paper). This is determines whether to half or double the step-size, dt
relerr = 5.e-9
!'Small', an offset used in the error calculation from the predicted and corrected
values
small = 1.0e-5
!err_fact- will be used in the error calculation using the predicted and corrected
values
err_fact_r(1:3,1:n) = 0.     !position values are used here
err_fact_v(1:3,1:n) = 0.     !velocity values are used here
err_max_r(1:n) = 0.        !This stores the maximum error for the three objects among
the errors for each x,y,z directions in position
err_max_v(1:n) = 0.        !This stores the maximum error for the three objects among
the errors for each x,y,z directions in velocity
err_r = 0.
err_v = 0.
!-------------------------------------
!Initial conditions
dr(1:3) = 0.
com(1:3) = 0.
cov(1:3) = 0.
!Initial energy (in J)
Ei = 0.
!Kinetic and Potential energy (in J)
KE = 0.
PE = 0.
!Distance between each object (in m)
rad = 0.
!Distance squared between each object (in m2)
rad2 = 0.
!Current energy (in J)
Ecurr = 0.
```

```
!Fractional energy (in J)
delE = 0.

!Predicted position,velocity and acceleration stored in the arrays below
r(1:3,1:n,1) = 0.
v(1:3,1:n,1) = 0.
a(1:3,1:n,1) = 0.
!Position,velocity and acceleration arrays which store the corrected position,velocity
and acceleration
r_c(1:3,1:n,1) = 0.
v_c(1:3,1:n,1) = 0.
a_c(1:3,1:n,1) = 0.
!Below store the new positions,velocities,accelerations produced when the timestep
is halved.
r_newpast(1:3,1:n,-3:-1) = 0.
v_newpast(1:3,1:n,-3:-1) = 0.
a_newpast(1:3,1:n,-3:-1) = 0.

!----------------------------------
!Centre of mass and velocity correction
!For the objects to be in the CoM frame
do i=1,n
   com(1:3) = com(1:3) + m(i)*r(1:3,i,0)
   cov(1:3) = cov(1:3) + m(i)*v(1:3,i,0)
   mtot = mtot + m(i)
end do

com = com/mtot
cov = cov/mtot

do i=1,n
   r(1:3,i,0) = r(1:3,i,0) - com(1:3)
   v(1:3,i,0) = v(1:3,i,0) - cov(1:3)
end do

!---------------------------------
!Initial energy determination
do i=1,n
   KE = KE + 0.5*m(i)*(v(1,i,0)*v(1,i,0) + v(2,i,0)*v(2,i,0) + v(3,i,0)*v(3,i,0))
end do

do i=1,n-1
   do j=i+1,n
      dr(1:3) = r(1:3,i,0) - r(1:3,j,0)  !separation between each body in x,y,z directions
      rad = sqrt(dr(1)*dr(1) + dr(2)*dr(2) + dr(3)*dr(3))
      PE = PE - (G*m(i)*m(j)/rad)
   end do
end do

Ei = PE + KE
```

```
!---------------------------------
!Initial acceleration calculation at time t=0 s
dr(1:3) = 0.
do i=1,n
   a(1:3,i,0) = 0.              !Initial acceleration at t=0
   do j=1,n
      if(i==j) cycle            !to avoid the case of finding separation between the
object and itself
      do k=1,3
         dr(k) = r(k,i,0) - r(k,j,0)
      end do
      rad2 = dr(1)*dr(1) + dr(2)*dr(2) + dr(3)*dr(3)          !r^2 = x^2 + y^2 + z^2

      do k=1,3                                                !runs over the all directions
         a(k,i,0) = a(k,i,0) - (G*m(j)*dr(k)/(sqrt(rad2)*rad2))    !dr(k) is the unit vector
indicating the direction of the force from
                                                              !objects 'j' to 'i'
      end do
   end do
end do

!shifting the t=0 step to t=-8 step
r(1:3,1:n,-8) = r(1:3,1:n,0)
v(1:3,1:n,-8) = v(1:3,1:n,0)
a(1:3,1:n,-8) = a(1:3,1:n,0)

!---------------------------------
!Bootstrapping the second order code- to get the past steps required for predictor-
corrector method
!loop created which iterates over 'p'.
do p=-7,0
   !Future position obtained from the initial position,velocity and acceleration
   do i=1,n
      r(1:3,i,p) = r(1:3,i,p-1) + v(1:3,i,p-1)*dt + 0.5*a(1:3,i,p-1)*dt*dt
   end do

   !acceleration loop
   do i=1,n
      a(1:3,i,p) = 0.
      do j=1,n
         if(i==j) cycle
         do k=1,3
            dr(k) = r(k,i,p) - r(k,j,p)
         end do
         rad2 = dr(1)*dr(1) + dr(2)*dr(2) + dr(3)*dr(3)

         do k=1,3
            a(k,i,p) = a(k,i,p) - (G*m(j)*dr(k)/(sqrt(rad2)*rad2))
         end do
```

```
      end do
   end do

   !Future velocity
   do i=1,n
      v(1:3,i,p) = v(1:3,i,p-1) + 0.5*(a(1:3,i,p-1) + a(1:3,i,p))*dt
   end do
end do


!----------------------Fourth order Predictor-Corrector method-----------------------
!Uses four past steps, for example, for t+1 position, t=0,-1,-2,-3 past positions are
required
!---------Adams-Bashforth-Moulton Method---------

do

   !x--------PREDICTOR---------x
   do i=1,n
   !Predicted position, r(t+1)
      r(1:3,i,1) = r(1:3,i,0) + dt*(-9.*v(1:3,i,-3) + 37.*v(1:3,i,-2) - 59.*v(1:3,i,-1) +
55.*v(1:3,i,0))/24. !coefficients are taken from the reference of the ADAMS-
BASHFORTH-MOULTON
METHOD GIVEN IN THE REPORT.
   !Predicted velocity, v(t+1)
      v(1:3,i,1) = v(1:3,i,0) + dt*(-9.*a(1:3,i,-3) + 37.*a(1:3,i,-2) - 59.*a(1:3,i,-1) +
55.*a(1:3,i,0))/24.
   end do

   !Predicted acceleration
   dr(1:3) = 0.
   rad2 = 0.
   do i=1,n
      a(1:3,i,1) = 0.
      do j=1,n
         if(i==j) cycle
         do k=1,3
            dr(k) = r(k,i,1) - r(k,j,1)
         end do
         rad2 = dr(1)*dr(1) + dr(2)*dr(2) + dr(3)*dr(3)              !r^2 = x^2 + y^2 + z^2

         do k=1,3                                                !runs over the all directions
            a(k,i,1) = a(k,i,1) - (G*m(j)*dr(k)/(sqrt(rad2)*rad2))    !dr(k) is the unit vector
indicating the direction of the force from
                                                                 !objects 'j' to 'i'
         end do
      end do
   end do

   !x--------CORRECTOR---------x
```

```
    !Now, the predicted value at t+1 is used alongwith the t=0,-1,-2 positions for
determining the corrected value at t+1.
    do i=1,n
    !Corrected position, r_c(t+1)
        r_c(1:3,i,1) = r(1:3,i,0) + dt*(v(1:3,i,-2) - 5.*v(1:3,i,-1) + 19.*v(1:3,i,0) +
9.*v(1:3,i,1))/24.

    !Corrected velocity, v_c(t+1)
        v_c(1:3,i,1) = v(1:3,i,0) + dt*(a(1:3,i,-2) - 5.*a(1:3,i,-1) + 19.*a(1:3,i,0) +
9.*a(1:3,i,1))/24.
    end do

    !Corrected acceleration, a_c(t+1)
    dr(1:3) = 0.
    rad2 = 0.
    do i=1,n
        a_c(1:3,i,1) = 0.
        do j=1,n
            if(i==j) cycle
            do k=1,3
                dr(k) = r_c(k,i,1) - r_c(k,j,1)
            end do
            rad2 = dr(1)*dr(1) + dr(2)*dr(2) + dr(3)*dr(3)

            do k=1,3
                a_c(k,i,1) = a_c(k,i,1) - (G*m(j)*dr(k)/(sqrt(rad2)*rad2))
            end do
        end do
    end do

    !Updating the time counter
    tcurr = tcurr + dt
    tcount = tcount + dt

    !--------Energy check and orbital distance-----------
    if(tcount>(0.5*yr)) Then

    !x-----Fractional energy calculation-----x
        !getting the current total energy
        KE = 0.
        do i=1,n
            KE = KE + 0.5*m(i)*(v_c(1,i,1)**2 + v_c(2,i,1)**2 + v_c(3,i,1)**2)
        end do

        PE = 0.
        rad = 0.
        dr(1:3) = 0.
        do i=1,n-1
            do j=i+1,n
                dr(1:3) = r_c(1:3,i,1) - r_c(1:3,j,1)
```

```
            rad = sqrt(dr(1)*dr(1) + dr(2)*dr(2) + dr(3)*dr(3))
            PE = PE - G*m(i)*m(j)/rad
        end do
    end do

    Ecurr = KE + PE

    !fractional energy loss,delE
    delE = (Ei - Ecurr)/Ei

  !x-----Measuring the distance from the sun-----x
    do i=1,n
        dsun(i) = sqrt(r_c(1,i,1)*r_c(1,i,1) + r_c(2,i,1)*r_c(2,i,1))
    end do
    tcount = 0.

  end if

  !--------Adaptive timestep check------------
  !Changing the timestep based on the error found between the predicted and
corrected values
  do i=1,n
    err_fact_r(1:3,i) = (19./270.)*(ABS(r_c(1:3,i,1) - r(1:3,i,1))/(ABS(r_c(1:3,i,1)) +
small))
    err_fact_v(1:3,i) = (19./270.)*(ABS(v_c(1:3,i,1) - v(1:3,i,1))/(ABS(v_c(1:3,i,1)) +
small))

    err_max_r(i) = maxval(err_fact_r(1:3,i)) !this finds the maximum error among
the x,y,z components of the position and velocity for each object.
    err_max_v(i) = maxval(err_fact_v(1:3,i))
  end do

  !Maximum error found for position and velocity values among all the objects.
  err_r = maxval(err_max_r(1:n))
  err_v = maxval(err_max_v(1:n))

  !HALVING THE TIMESTEP
  !The maximum of the error contribution from the position and velocity is compared
against the relative error.
  if(max(err_r,err_v)>relerr) Then
    dt = dt*0.5

    do i=1,n
    !Interpolated steps at -1/2 and -3/2 produced using the past steps
      v_newpast(1:3,i,-1) = (-5.*v(1:3,i,-4) + 28.*v(1:3,i,-3) - 70.*v(1:3,i,-2) +
140.*v(1:3,i,-1) + 35.*v(1:3,i,0))/128.
      a_newpast(1:3,i,-1) = (-5.*a(1:3,i,-4) + 28.*a(1:3,i,-3) - 70.*a(1:3,i,-2) +
140.*a(1:3,i,-1) + 35.*a(1:3,i,0))/128.
```

```
        v_newpast(1:3,i,-3) = (3.*v(1:3,i,-4) - 20.*v(1:3,i,-3) + 90.*v(1:3,i,-2) +
60.*v(1:3,i,-1) - 5.*v(1:3,i,0))/128.
        a_newpast(1:3,i,-3) = (3.*a(1:3,i,-4) - 20.*a(1:3,i,-3) + 90.*a(1:3,i,-2) +
60.*a(1:3,i,-1) - 5.*a(1:3,i,0))/128.
      end do

      do i=1,n
        v(1:3,i,-4) = v(1:3,i,-2)
        a(1:3,i,-4) = a(1:3,i,-2)

        v(1:3,i,-2) = v(1:3,i,-1)
        a(1:3,i,-2) = a(1:3,i,-1)

        ! -1/2 interpolated step
        v(1:3,i,-1) = v_newpast(1:3,i,-1)
        a(1:3,i,-1) = a_newpast(1:3,i,-1)

        ! -3/2 interpolated step
        v(1:3,i,-3) = v_newpast(1:3,i,-3)
        a(1:3,i,-3) = a_newpast(1:3,i,-3)

      end do
      err_max_r(1:n) = 0.
      err_max_v(1:n) = 0.
      err_r = 0.
      err_v = 0.

    !DOUBLING THE TIMESTEP
    else if(max(err_r,err_v)<(1.e-3*relerr)) then
      if(counter==4) then
        dt = 2.*dt

        do i=1,n
          v(1:3,i,-1) = v(1:3,i,-2)
          a(1:3,i,-1) = a(1:3,i,-2)

          v(1:3,i,-2) = v(1:3,i,-4)
          a(1:3,i,-2) = a(1:3,i,-4)

          v(1:3,i,-3) = v(1:3,i,-6)
          a(1:3,i,-3) = a(1:3,i,-6)

          v(1:3,i,-4) = v(1:3,i,-8)
          a(1:3,i,-4) = a(1:3,i,-8)
        end do

        counter = 0
        err_max_r(1:n) = 0.
        err_max_v(1:n) = 0.
        err_r = 0.
```

```
            err_v = 0.
        end if
        counter = counter + 1
    end if

    !-------------------------------------------------------
    !DOWNSHIFTING arrays of position,velocity and acceleration
    !The final past step (t=-8) is shifted out of the array, and the corrected value at
t=+1 is shifted to t=0.
    !Done for position, velocity and acceleration
    !All the past steps are downshifted along the arrays
    do p=-8,-1
        do i=1,n
            r(1:3,i,p) = r(1:3,i,p+1)
            v(1:3,i,p) = v(1:3,i,p+1)
            a(1:3,i,p) = a(1:3,i,p+1)
        end do
    end do

    !The corrected value at t+1 is downshifted to t=0 step.
    do i=1,n
        r(1:3,i,0) = r_c(1:3,i,1)
        v(1:3,i,0) = v_c(1:3,i,1)
        a(1:3,i,0) = a_c(1:3,i,1)
    end do

    if(tcurr>tout) exit
end do
end program nbodyfour
```

B: The first semester report is also included.

N-body Simulations of Star-Star Encounters

150193993

13/12/2018

Abstract

The basic N-body problem and the applications of it to problems in astrophysics are explained in the literature review. The progress section gives a brief description of the second-order code written for the system along with a mention of the values selected for the time-step and the run-time. Results produced from the energy conservation and stability checks also included. A project plan is added at the end with all the tasks for semester two in a gannt chart. Along with it, a short explanation of the all the tasks is included. A conclusion summarises all the sections mentioned in the report.

1. Introduction

The basic N-body problem is described for a system where only gravitational interactions take place. These interactions cause the positions and velocities of the particles to change with time. For a stellar cluster where Newtonian gravity dominates, these changes can be simulated. The aim of this project lies in developing a working N-body code that tracks the evolution of a star formation region.

The motivation of writing this N-body code lies in the problems of astrophysics. Star cluster evolution present as the perfect area for applying the N-body problem. An important phase of this evolution comes during mass segregation. Consider a star cluster with two different stellar mass groups. Gravitational interactions between the stars from the heavy and light groups cause a net movement of the heavy component to the centre. This causes a thermal energy outflow as energy is transferred from one component to the other. This leads to a process of mass segregation in the star cluster where the high mass stars start drifting towards the centre.

Another application comes from the planet migration theory explained in the Nice model. The current shape and structure of the Kuiper belt, existence of Kuiper belt objects, the highly eccentric orbit of Pluto and the location of the orbits of the four giant planets is explained by this theory. After the dissipation of the initial planetesimal disk, gravitational interactions occur between the giant planets and the remnant planetesimals. Exchange of angular momentum caused the orbits to drift outwards causing the giant planets to migrate. This situation of orbital migration can be recreated using N-body simulations. Similarly, other problems like the interactions between binary systems and individual stars, merger between two galaxies causing a stream of gas and other particles to appear etc. can be solved using a similar N-body code.

For the initial part of the project, a simple second order code is created for the solar system. This allows to look at the orbital evolution of the planets due to the planet-planet interactions. The blocks of the code are explained later. Energy conservation and stability checks are done on the system. Results from the tests give an idea on how good is the chosen time-step.

2. Literature review

2.1 N-body problem

Newton's law of gravitation describes how a group of stars interact in a star cluster. These gravitational interactions cause the dynamical properties (velocity, position, acceleration) of the stars to change. Due to this, the dynamical properties of the region inhabiting the stars is affected. Therefore, N-body simulations are useful to observe the evolution of region. The basic N-body problem is ingrained in the prediction of the future values of the positions and velocities of the particles in the system. The initial properties of the particles in the system are known. In a system of N particles, the acceleration of a particle can be defined as,

$$\boldsymbol{a_i} = \sum_{i=1, i\neq j}^{N} \frac{Gm_j}{\left|r_{ij}\right|^2}\widehat{\boldsymbol{r_{ij}}} \qquad \text{-1}$$

Where, $m_j -$ mass of the test particle

$\widehat{\boldsymbol{r_{ij}}} -$unit vector giving a direction of the force exerted by body j on i

$\left|r_{ij}\right|^2 -$ modulus square of the distance between the bodies

Integrating equation (1) provides the position and velocity of a particle at any time *t*. For *N*=2 the above equation is analytically solvable. Since we are considering stellar clusters as our system, where N≥2, numerical methods become reliable.

Numerical integration of equation 1 provides the solutions for the position and velocity.

$$\boldsymbol{r_1} = \boldsymbol{r_0} + \boldsymbol{v_0}dt + \frac{\boldsymbol{a_0}}{2}(dt)^2 + \frac{\dot{\boldsymbol{a}}_{\boldsymbol{0}}}{3!}(dt)^3 + \frac{\ddot{\boldsymbol{a}}_{\boldsymbol{0}}}{4!}(dt)^4 + \cdots \qquad \text{-2}$$

$$\boldsymbol{v_1} = \boldsymbol{v_0} + \boldsymbol{a_0}dt + \frac{\dot{\boldsymbol{a}}_{\boldsymbol{0}}}{2}(dt)^2 + \frac{\ddot{\boldsymbol{a}}_{\boldsymbol{0}}}{3!}(dt)^3 + \frac{\dddot{\boldsymbol{a}}_{\boldsymbol{0}}}{4!}(dt)^4 + \cdots \qquad \text{-3}$$

Where, $\boldsymbol{r_1}, \boldsymbol{v_1} -$ new position and velocity of the particles.

$\boldsymbol{r_0}, \boldsymbol{v_0} -$ initial position and velocity of the particles

$\boldsymbol{a_0}, \dot{\boldsymbol{a}}_{\boldsymbol{0}}, \ddot{\boldsymbol{a}}_{\boldsymbol{0}}, \dddot{\boldsymbol{a}}_{\boldsymbol{0}} -$ initial acceleration of the particles with the latter three being the 1st, 2nd, 3rd time derivatives respectively

$dt -$ increment in time, t for the simulation

The time-step *dt* determines the accuracy of the values of the future position and velocity of the particles. It goes inversely with the computational time. As *dt* drops there is a rise in the number of calculations done in a single simulation.

Accuracy in predicting the motion of stars in the cluster is the main task. The error in the solution is proportional to the n^{th} power of the time-step (where n- order number). As we go to higher orders, a small drop in *dt* will imply a large reduction in the error. The timescales for evolution of a star cluster lies are million years. The second order code would take a huge amount of computational time to observe these evolutionary features. Hence, the fourth-order method is the preferred option when simulating a star cluster evolution as it offers higher accuracy than other ones. The fourth order predictor-corrector method will be implemented in the second semester.

For our project, we consider the second order method, with the equations used given below.

$$\boldsymbol{r_1} = \boldsymbol{r_0} + \boldsymbol{v_0}dt + \frac{\boldsymbol{a_0}}{2}(dt)^2 \qquad \text{-4}$$

$$\boldsymbol{v_1} = \boldsymbol{v_0} + \boldsymbol{a_0}dt + \frac{\dot{\boldsymbol{a}}_{\boldsymbol{0}}}{2}(dt)^2 \qquad \text{-5}$$

2.2 Astrophysics applications of N-body method

The motivation of writing this N-body code come from its application to certain astrophysics problems. The problems like mass segregation in star clusters, orbital migration of the giant

planets in the planetesimal disk (Nice model), presence of boxy and peanut-shaped bars in the bar evolution phase etcetera. In these problems, the system considered (stellar cluster or an ensemble of planets and planetesimals) is self-gravitating.

Star cluster evolution is a difficult scenario to build in a numerical simulation. The interaction between the stars determine the evolution of the cluster. Khalisi, Amaro-Seoane and Spurzem (2006) observed dynamical evolution of a star cluster consisting of two different stellar mass groups (heavy and light components). A similar analysis of globular systems was done by Spitzer (1969). In dynamical equilibrium, the cluster achieving a thermal velocity distribution determines its evolution. Reaching a maxwellian velocity distribution through small changes in velocity of the stars is termed as relaxation. Relaxation forms the major part in shaping the structure of the cluster. The important phase in the evolution of the cluster comes during the central core collapse. Three processes can reach this stage: equipartition, evaporation and gravothermal instability. Star-star encounters allow mass segregation in the central regions. These encounters cause a transfer of kinetic energy from the heavy to light component. This leads to movement in the heavy and low mass components to and away from the centre of the star cluster. There is a thermal energy outflow from the center to the outer regions of the cluster. Significant evidence for mass segregation in star clusters was found in Infra-red observations of the trapezium cluster in Orion (Khalisi, Amaro-Seoane and Spurzem, 2006)

A simplest approximation of two-mass component simulations is considered. To carry out the simulations, they considered a plummer sphere model. This model describes the density distribution in the cluster. On using this model, the particles can be considered point-like, and large scattering events between the particles at small distances are avoided. In addition, other stellar evolution scenarios, like primordial binaries and tidal fields are neglected. Khalisi, Amaro-Seoane, and Spurzem, (2006) used two parameters namely, fraction of heavy mass component, $q = M_h/M_{clu}$ and the mass fraction of each particle, $\mu = m_h/m_l$ to describe a set of models. A number of runs were assigned to each model, with each of them having a different setup of initial positions and velocities for all the particles. A strong approach was done to produce accurate N-body simulations with a large number of particles, keeping in mind about a low computing power. Similar to mass segregation in star clusters, simulations are possible for the orbital migration of the giant planets in the remnant planetesimal disk.

The Nice model explains the orbital migration of the giant planets and the current structure of the outer Solar System. Specifically the orbital migration of Neptune caused the majority of Kuiper Belt Objects (KBOs) to be at Neptune's mean motion resonance of 5:2. The large eccentricity and inclination in Pluto's orbit, along with the occurrence of the Oort cloud at distances>2000 AU are result from this migration. The Kuiper belt inhabits a vast number of residual planetesimals, which did not coalesce to in the giant planet formation stage. These Kuiper Belt Objects can give the history of the outer solar system. The interaction between the planetesimal disk and the planets led to exchange of angular momentum thereby affecting their orbits.

N-body simulations of the interactions between giant planets and the planetesimal disk was carried out by Hahn, J.M. and Malhotra, R. (2005, p. 2392). The simulations included 10^4 massless particles surrounding the migrating giant planets. With a timescale of the age of the solar system, the orbital evolution was tracked. Two scenarios were considered for the initial disk: when it was dynamically cold, and when it was destabilized before migration. They used a MERCURY6 N-body integrator for the particular evolution process. The migration was demonstrated by applying an external torque to the semi-major axis of the orbit. Below is the time-dependent form of the semi-major axis,

$$a_j(t) = a_{f,j} - \Delta_j e^{-\frac{t}{\tau}}$$

Where, $a_{f,j}$ −the final semi-major axis of the planet

Δ_j −planet's net radial displacement

τ −e-fold timescale for planet migration

For the initial conditions, the current masses and orbits of the planets was considered. In addition to that, the initial semi-major axis was modified by $-\Delta_j$. Due to this, the applied torque allowed the orbits to return to the present configurations. Various other constraints were used for Δ_j for each planet's orbit depending on the different resonances. A time-step of 0.5 years was used, with the massless particles randomly distributed over a range of 20-80 AU. A more extensive approach for the N-body simulations was done by Hahn, J.M. and Malhotra, R. (1999, p. 3041) to look at the orbital migration of the giant planets in the remnant planetesimal disk. Moreover, different cases for migration were considered based on the mass of the planetesimal disk.

The advent of oligarchic growth in planet formation is another scenario for an N-body problem. The core accretion theory for planet formation gives a good description of the runaway and oligarchic growth stages. Kokubo, E. and Ida, S. (1998) explored the orbital evolution of the proto-planets embedded in the planetesimal disk using three-dimensional N-body simulations. They considered was a planet-planetesimal system. They observed the growth of the protoplanets to be slower among themselves. However, it was faster in comparison to that of the planetesimals. From orbital migration, to the formation of magellanic stream, they represent good scenarios for N-body simulations.

Galactic cannibalism of the Magellanic Clouds by the tidal field created from the merger with the Milky Way led to stream of gas called the magellanic stream. This scenario was reproduced in numerical simulations done by Maddison et. al (2002). They compared two types of numerical simulations: one, which looked at N-body only merger and the other type, hydrodynamic. For the latter type, the factors of star formation, supernova feedback and metal enrichment were included into the simulations along with the gravitational interactions. The results from the N-body only simulations showed the Small Magellanic Cloud (SMC) being tidally stripped, thereby showing the presence of the magellanic stream. However, from the second simulation the SMC was strongly disrupted, with the stream devoid of stars. Figure 1 shows these results. The astronomical observations of the magellanic stream show the stream consisting of only gas.

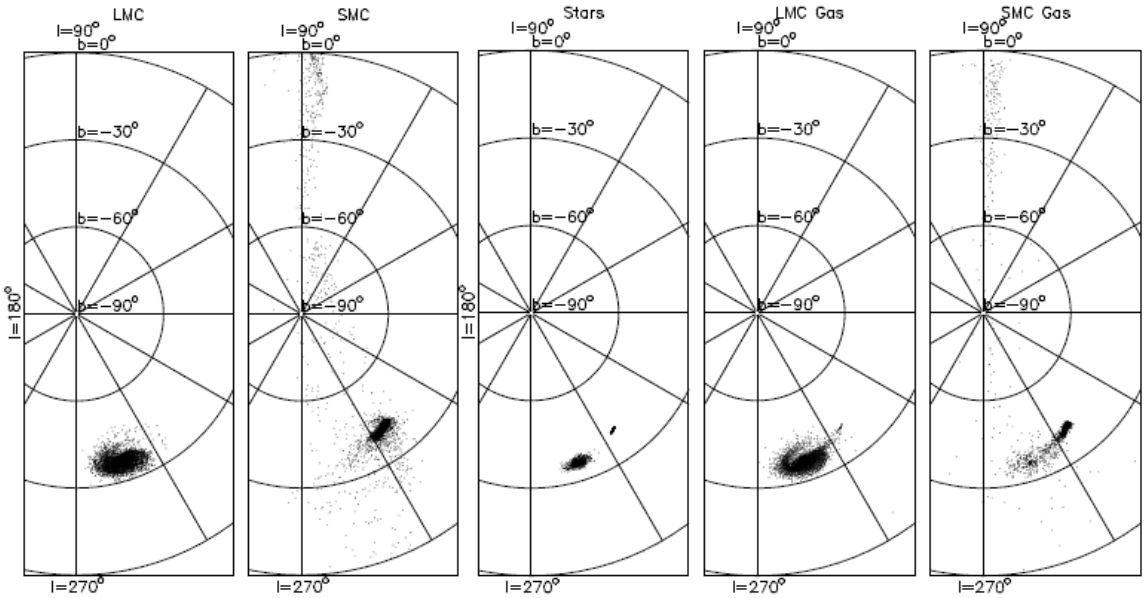


Figure 1: Different panels show the results from the N-body only and hydrodynamical simulations of the three-galaxy merger. Left two panels- N-body-only results showing the

magellanic stream consisting of stars. Three right panels- hydrodynamical simulation results show the presence of mostly gas in the magellanic stream.

3. Progress on project

The initial work on the project was done by constructing a simple second order code. Equation (5) can be corrected by considering an assumption given below.

$$\dot{\boldsymbol{a_0}} \sim \frac{\boldsymbol{a_1} - \boldsymbol{a_0}}{dt}$$

$$\boldsymbol{v_1} = \boldsymbol{v_0} + \frac{(\boldsymbol{a_0} + \boldsymbol{a_1})}{2} dt \qquad -6$$

Initially a three-body system of Sun-Earth-Jupiter was considered for the problem, with the rest of the planets added later.

For the first block of the code, the declaration of all the variables was done. Known variables like the gravitational constant *G*, the mass, initial velocity and initial position of the planets were initialised with the values taken from the planetary fact sheet on the NASA website. Certain initial conditions for the position and velocity are mentioned below. These conditions forced the planets to orbit in the XY plane. This reduced the complexity in the simulations.

$$r_y, r_z = 0$$

$$v_x, v_z = 0$$

Initially for the three-body system of the Sun-Jupiter-Earth, their orbits were produced. At this point the centre-of-mass of the system was not considered. The time-step was chosen to be 100 seconds and the run-time for the simulations was 30 years. Without including the centre-of-mass, the system would seem to drift away from the origin. Figure 2 shows the minor drift of the system.

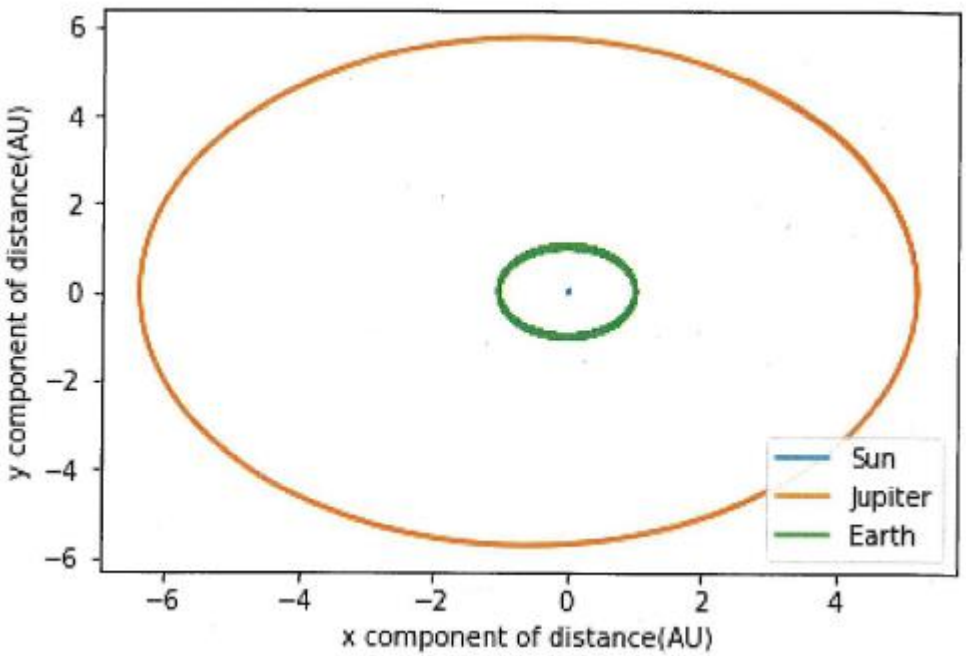


Figure 2: Orbits of the Sun-Jupiter-Earth with the simulations running for 30 yrs. The orbits are not corrected for the centre-of-mass and velocity.

To correct this, the objects' positions and velocities must be corrected for the centre-of-mass. Therefore, at this point the code which doing the corrections is added. The centre-of-mass and centre-of-velocity in the x direction is calculated as below,

$$COM_x = \frac{\sum_{i=1}^{N} m_i x_i}{m_{tot}} \qquad -7$$

$$COV_x = \frac{\sum_{i=1}^{N} m_i v_i}{m_{tot}} \qquad -8$$

Where, $m_{tot} = \sum_{i=1}^{N} m_i$, is the total mass of the system

An infinite loop was created which would run the simulations to a time t_{out}, and with a time-step *dt.* Future positions and velocities determined using equation (4) and (6). Modified form of equation (1) was used to calculate the future accelerations.

$$\boldsymbol{a_i} = \sum_{i=1.i\neq j}^{N} \frac{Gm_j.\boldsymbol{dr_{ij}}}{|r_{ij}|^3} \qquad -9$$

Where, $\boldsymbol{dr_{ij}} = \boldsymbol{r_i} - \boldsymbol{r_j}$

At certain points, small tests are given to the code to check for any logical errors. With all the positions, velocities and accelerations determined, the orbits of the planets were produced. The run-time for the simulations was upgraded to 1000 years with the time-step of 1000 sec. Due to the large runtime of 1000 years, the compilation took around 4-5 minutes. Figure 3 shows the orbits of all the planets in the Solar System.

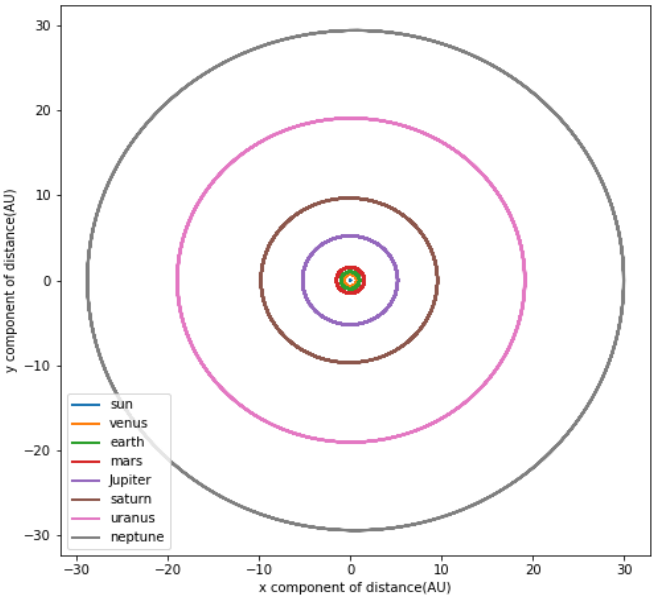


Figure 3: Orbits of the planets in the solar system. The time-step was chosen to be 1000 seconds with a run-time of 1000 years.

To check the stability of this system, an energy conservation check was done. This required the fractional energies to be determined using the equations given below. A counter, t_{count} was set so that after every 6 months this calculation was possible. This helped in speeding up the compilation.

$$PE = -\sum_{i=1,i\neq j}^{N} \frac{Gm_i m_j}{r} \qquad -10$$

$$KE = \sum_{i=1}^{N} \frac{1}{2} m_i \, v^2 \qquad -11$$

$$\Delta E = \frac{E_t - E_0}{E_0} \qquad -12$$

Where, $r = (x^2 + y^2 + z^2)^{1/2}$

$v^2 = (v_x^2 + v_y^2 + v_z^2)$

$E_t -$ total energy of the system at time t

ΔE and $E_0 -$the fractional energy and the initial energy of the system respectively

Figure 4 shows the fractional energy of the system varying with time was produced for a time step, $dt = 1000$ seconds. The time-step was then reduced to 500 seconds to look at the effects on the fractional energy. This caused a lot more computational time due to the large number of data points produced.

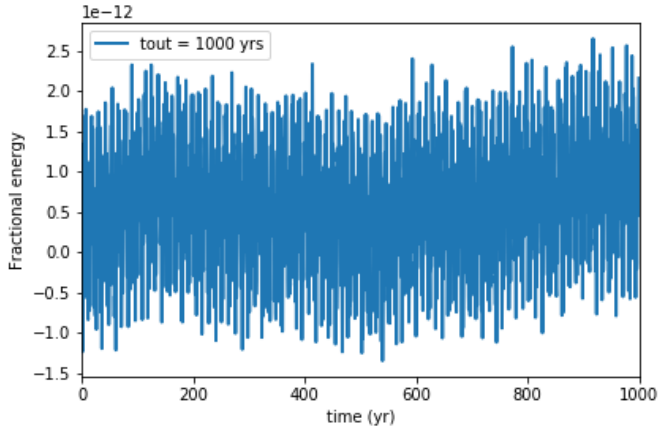


Figure 4: Fractional energy of the system against time (in years). The simulation was run for 1000 yrs with a time-step of 1000 seconds.

The fractional energy of the system depends on the time-step, *dt*. For good accuracy, the value of the time-step should be very low. This significantly increases the computational time, making the second order code slow. For a million year timescale the second order method is ineffective.

Finally, another test on the code was done by looking at how the distances from the sun varied with time for the planets. This gave an idea of the stability of the system. The run-time and the time-step were left unchanged. But to avoid a huge amount of data points, the distance from the Sun was calculated for every thousandth iteration (i.e. after every million seconds). Figure 5 shows the variations in the separation from the Sun for the planets.

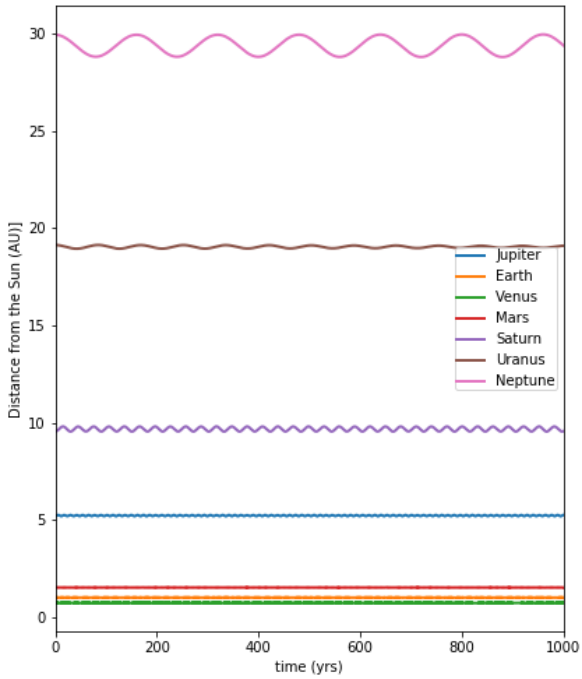
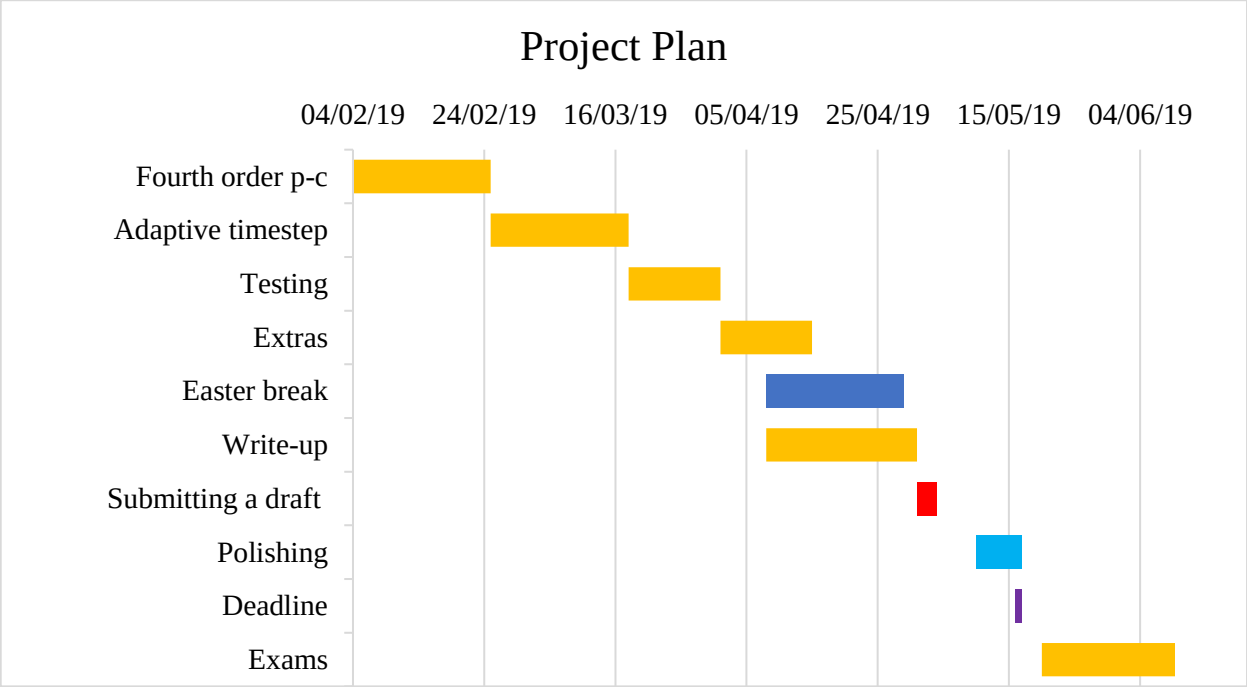


Figure 5: The separation from the sun (in AU) against time (in years). The periodic variations present in the curves is due to the interactions between the planets.

Therefore, the self-gravitating system in the code is the current the Solar System. The gravitational interactions between the bodies caused their dynamical properties to vary with time. N-body simulations showed how the dynamical properties of the objects in the Solar System evolve with time.

4. Project plan



At the start of week 1, the core part of the work in semester two begins with task-1, i.e., building a basic fourth order predictor-corrector code. A time of two-three weeks is given for this task based on its difficulty. After forming the base, we require an adaptive time-step for the code (task-2). Depending on the errors obtained from the energy checks, the code either doubles or halves the time-step, *dt*. A similar amount of time of three weeks is assigned to this task. Now, the two main components of the code are ready. This leads to task-3 of testing the code. The results obtained from these tests determine whether the code is working correctly. For example, milankovic cycles should be present in the time-varying plot of eccentricity of the orbits of the planets in our Solar System. A time of two weeks is set for this work.
Task-4 provides the motivation of writing this piece of code. This code is applied to any astrophysics problem mentioned in the previous sections. In this task, certain extra bits are added to the code. Moreover, certain tweaks are applied at this point to increase the speed of the code and to make it more efficient. With a time of two weeks, it overlaps with the Easter break. This leads to the most important task-5, the write-up of the report. All the figures and results produced in tasks 3 and 4 are included in the report. Tasks 4 and 5 overlap at the beginning of Easter break. A time of three-four weeks is assigned so that a draft of the final report can be submitted to the supervisor approximately two weeks before the deadline (17/05/2019). The final task involves in refining the report. Within a week the report is checked for any mistakes before the final submission.

5. Conclusion

A literature review was done on the basic N-body problem and the applications of the code to certain astrophysics problems. A simple second order code for the N-body simulations was constructed to apply on the current Solar System. With the necessary initial conditions given to the system and applying the centre-of-mass corrections, the orbits of the planets were produced. A check on the energy conservation and stability were applied on the system.

Finally, a project plan for semester two was added which gave an overview of all the tasks to be done for the project.

References

Khalisi, E., Amaro-Seoane, P. and Spurzem, R., (2006). A comprehensive N-BODY study of mass segregation in star clusters: energy equipartition and escape. *Monthly Notices of the Royal Astronomical Society*, 374(2), pp.703-720.

Hahn, J.M. and Malhotra, R. (2005). Neptune's migration into a stirred-up Kuiper belt: A detailed comparison of simulations to observations. *The Astronomical Journal*, 130(5), p.2392.

Hahn, J.M. and Malhotra, R., (1999). Orbital evolution of planets embedded in a planetesimal disk. *The Astronomical Journal*, 117(6), p.3041.

Kokubo, E. and Ida, S. (1998). Oligarchic growth of protoplanets. *Icarus*, 131(1), p.171-178.

Maddison, S.T., Kawata, D., Gibson, B.K. (2002). Galactic Cannibalism: The Origin of the Magellanic Stream. *Astrophysics and Space Science*, 281(1), p.421-422

Spitzer Jr, L., 1969. Equipartition and the formation of compact nuclei in spherical stellar systems. The Astrophysical Journal, 158, p.L139.

https://nssdc.gsfc.nasa.gov/planetary/factsheet/