Sandeep Singh

Computer Simulation Report

Introduction:

This project is about simulating the motion of planets in the solar system orbiting around the Sun and we needed to find out the orbital period of each of the planets and find out the total energy of the system. I tackled Experiment 1 all the way to Experiment 3, and I made the code from scratch.

The aim of Experiment 1 was just to check how the orbital periods of my simulation of the planets orbiting around the Sun match with the actual orbital periods of the planets orbiting around the Sun.

Experiment 2 was changing the Beeman scheme which showed the position and velocity of the planets revolving around the Sun at a certain timestep, I would now use the Forward Euler method which showed the position and velocity of the planets around the Sun at a certain timestep and compare the plot of total energy of system with time of Direct Euler method and Beeman scheme.

Experiment 3 which I did was about launching a satellite from Earth to land on Mars, even though this was a random satellite I did call the satellite as “Perseverance” to make things easier, and we were asked to find out the minimum distance of the satellite and Mars, the time taken to obtain the minimum distance of the satellite and Mars and whether the satellite ever reaches Earth again.

Methods:

Firstly, for the Simulation:

Since we had to regularly read the simulation parameters from the file, I had a file called “bodies.csv” which basically has all the data for all the planets from Mercury to Mars, the Sun, and the satellite for Experiment 3. The file data is separated by a semicolon, and it basically has the name of the body, the color of the body, the mass (in kg), the Perihelion distance (in meters) and its max orbital velocity (in meters per second) respectively. I collected the data for each of the bodies using a website shown in the reference.[1] That website shown in the reference, is the website for mercury and you can change the part which says “mercuryfact” to “xfact” where x is for another planet and in that website I used the mass of the body, the max orbital velocity which I converted to

meters per second and the perihelion distance(which I converted to meters). Perihelion is the point of the planet when its closest to the Sun, and I had a method called read_input() which is a simple program that splits the data every time it sees a semicolon and the split data would then be received in the Body class and the data will be organized properly based on its order.

Showing the orbit of the planets in a graphical display was tough to make. To do this, I made a couple of methods called animate() and show_animation(). The animate() function sets the center of every patch in the animation to the position of the body, and it gets called after every timestep from the function FuncAnimation in the method show_animation(). The tough part of simulating the animation was in the show_animation() method. I had to figure out the size of each body in the animation which I set to 5e9 but there is also a problem with this value as in some parts of the animation the circles look like they are colliding (Mars and the satellite) because they are extremely big, I explained more about this problem in the Discussion section.

I also have a graph showing the total energy of the system against time, and as you can see, I also have a csv file with a list of numbers, those values are a list of the total energy of the system for the graph to be plotted.

For Experiment 1:

To check for the orbital period of each body I just had a function called orbital_period() which basically moves the planets around one loop around the solar system and returns the time it took to complete the full cycle and I compared that value to Earth years and printed out the values. In the method orbital_period() I did it by first only making the planet move below the y-axis (which is half an orbit) and I saved the amount of time that took, and after that I made sure the planet only moved above the y-axis (which is the second half of the orbit) and I found the amount of time that took, and I printed out the total time for that full orbit. I did it this way as that is the only way to make sure the planet does a full orbit, as it takes the sum of the time it took when the planet is below the Sun and the time it took for the planet to be above the Sun.

For Experiment 2:

For the direct Euler method, I basically had 2 functions which returned the Euler position of a planet and its Euler velocity of the planet after each timestep as shown in the update_position_euler() method and update_velocity_euler() method in the Body

class. Implementing these methods wasn't hard as the Direct Euler formulas were already given, and to call these functions, in the move() method in the class solar(), change the update_position_beeman() and update_velocity_beeman() to update_position_euler() and update_velocity_euler().

For Experiment 3:

I had to find some values for the satellite and as you can see in the bodies.csv file I made the value of the mass of the satellite to be 2*10^15 kg due to this website [4]. (At the bottom of that link you can see that a satellite called Deimos has a mass of 2.4*10^15kg). I just launched the satellite right above earth thus the position vector of the satellite is [147.095e9,1] which means its launched 1m above earth. But finding the velocity vector was tough and it required a lot of trial and error. Following this website [2], it shows that to find the orbital velocity of a satellite I should first use the formula. $v = \sqrt{\left(G * \frac{M}{R}\right)}$ and it shows that for a 400km orbit, the speed of the satellite would then be 3361 meters per second, thus I just rounded it up to 3500 meters per second as the website got that value from a 400km orbit but I wanted a closer orbit thus I used a higher value for the x vector of the satellite speed but for the y velocity vector was just a lot of trial and error to figure out the correct value for the motion of the satellite to look as normal as possible.

Results:

For the results I'll first be showing an image of the simulation of the planets revolving around the solar system as this is needed for the Simulation.

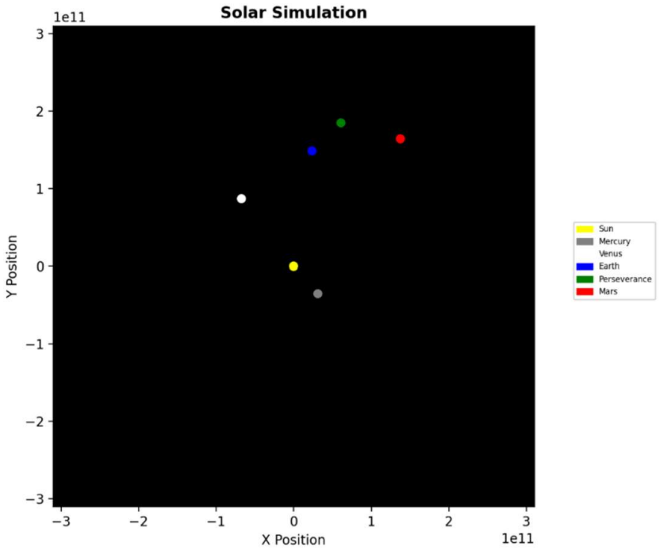

Figure 1

Figure 1 shows a scene of my solar system animation, which shows all the planets in the solar system as well as the satellite flying to Mars.

The figure below is going to show a plot of total energy against time of all the planets in the solar system as it revolves around the Sun using the Beeman method.

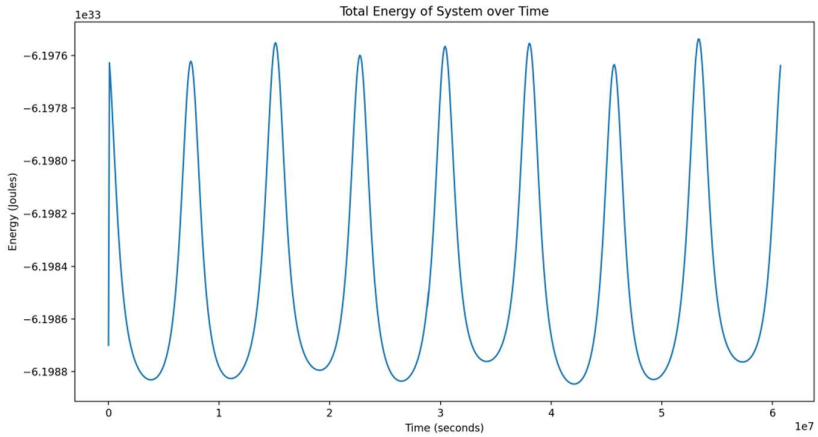

Figure 2

Thus, as shown in Figure 2, using the Beeman Method, total energy of the system is conserved as it moves around the Sun.

However, using the Direct Euler method, the total energy of the system would vary a lot as shown in the image below.

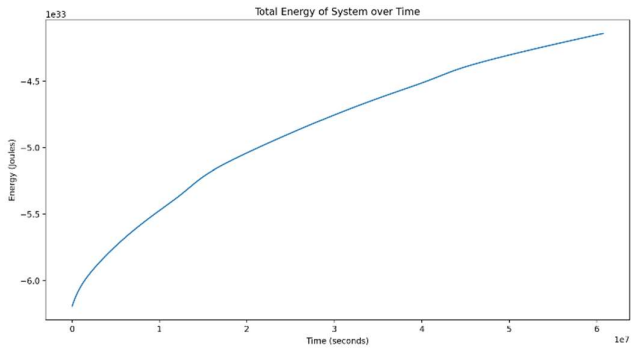

Figure 3

Figure 3 happens when I stopped using the Beeman scheme to update the position and velocity of the planets, but instead used the Direct Euler method for it.

And finally, this shows the time taken for each planet to complete a full cycle around the planet.

```
Mercury orbital period = 7689600 seconds which is 0.24 Earth years
Venus orbital period = 19440000 seconds which is 0.62 Earth years
Earth orbital period = 31622400 seconds which is 1.0 Earth years
Perseverance orbital period = 50371200 seconds which is 1.6 Earth years
Mars orbital period = 59356800 seconds which is 1.88 Earth years
Closest distance of satellite and Mars = 3150898.38 kilometres
Time to reach closest distance to Mars = 31104000 seconds
```

Figure 4

The figure above uses the Beeman Scheme to update the position and velocity of the planet at a given timestep.

Discussion:

Using my results, I shall first tell you what it means, Figure 4 shows the time taken for the planet to complete a full cycle around the solar system. I have printed out the Earth years as well for each of the planets and comparing it with the actual results, for the time taken for planets in the solar system to complete a full cycle around the Sun which

I got the values from a website[3]. (The values are shown in the section called “new” age chart)

the time taken for my simulation planets to complete a full cycle is closely related to its actual value except maybe for some planets it differs by 1, this is because I’m using the planets Perihelion distance which is the distance where the planet is closest to the sun throughout the experiment and I’m also always starting by using the planet’s max orbital velocity throughout the simulation. The general trends I see is how the energy is conserved in the system as the planets revolve around the Sun when using the Beeman method as shown in Figure 2 but for the Euler method energy is not conserved with time.

The main unphysical thing in my simulation is how the Satellite looks like its colliding with Mars as it tries to fly from Earth to Mars in the simulation. This is because the radius of the Circle in the Simulation is pretty big as the size in the simulation is 5e9 but when you change the radius of the circle to be a smaller value for the simulation, you can see that the satellite wouldn’t actually collide with Mars, in fact as shown in Figure 4, the closest distance the satellite would ever get to Mars is actually 3150898.48 km.

Conclusion:

In the final implementation of the code, you should see the whole simulation of the planets revolving around the Sun, a graph displaying total energy of the system with time, and in the terminal, you should see the time taken for the planets to complete a whole rotation around the Sun. I could have made my animation of the satellite flying to Mars a bit more realistic and not make it seem like they are colliding with each other, however I didn’t have enough time for that.

Reference:

1. https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html
2. https://www.quora.com/What-is-the-velocity-of-a-spacecraft-traveling-in-a-relatively-circular-low-Mars-orbit
3. https://www.higp.hawaii.edu/spacegrant/old/class_acts/HowOld.html
4. https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html

1707 words