

2-Dimensional Simulation of Inner Planets of our Solar System using the Beeman Integration Scheme

Anant K. Gupta | Computer Simulation Project

Introduction

Firstly, I created a 2-D simulation of the Sun and the inner planets of our solar system, namely Venus, Mercury, Earth, and Mars. I did so in python using the 3 step Beeman integration scheme (elaborated upon later) to calculate the movement of the celestial bodies. I then carried out the following experiments:

1. Orbital Periods

The aim of this experiment was to check the accuracy of the solar system. As we are simulating a system, we must compare the orbital periods of the planets in the simulation to the actual figures to see how accurate the simulation is.

2. Conservation of Energy

The aim of this experiment was to see if the simulation was properly implementing the Beeman algorithm and if the simulation itself was stable. As the Beeman algorithm is a symplectic integrator, the total energy of the simulated system should remain conserved, though some variation is expected as we treat time as discrete rather than continuous in the Beeman Algorithm.

3. Satellite to Mars

The aim of this experiment was to simulate the launching of a satellite from Earth and perform a fly-past of Mars. I launched the satellite with various velocities and at various angles to find the most optimal way to perform a fly-past of Mars, i.e., how close we can get to Mars and what is the least amount of velocity required to get there. I also checked if the satellite ever returns to Earth.

Methods

BODY CLASS

The purpose of this class is to create celestial bodies for the simulation. The class implements the various methods that may be required for the celestial bodies such as updating their acceleration, calculating their kinetic energy etc. and holds all the data related to the celestial body such as their mass, velocity, position etc. The methods calculate the body's next position and velocity using the Beeman Algorithm:

$$\vec{r}(t + \Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + 16[4\vec{a}(t) - \vec{a}(t - \Delta t)]\Delta t^2$$

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \frac{1}{6}[2\vec{a}(t + \Delta t) + 5\vec{a}(t) - \vec{a}(t - \Delta t)]\Delta t$$

Where Δt is the small time-step, t is the current time and \vec{r} , \vec{v} and \vec{a} represent the position, velocity, and acceleration vectors for the motion.

SYSTEM CLASS

This class contains and manages the entire simulation. It requires the argument filename – a CSV file containing the data for all the celestial bodies in the simulation and abiding by the format given in the readme file (README.md). It also values for time-step and number of frames, but these are optional. The simulation runs endlessly but can be limited to a certain number of frames.

The animation is achieved on-the-fly, i.e. each timestep execution is immediately displayed rather than compiling all the data beforehand. Even though it is possible to run the experiments with animation, it is processor-intensive and time-consuming, thus its better to simply collect and observe the data. This was accomplished using a method ‘iterate’ that simulates 1 time-step and we can call this method for the number of frames we desire to run the simulation without the animation. The ‘display’ method uses Funcanimation from matplotlib to display the animation of the simulation. I also added methods to add satellites to the simulation as I felt this would lead to better code structure.

For the displaying of planets, I chose not to display their size realistically as this makes it extremely difficult to observe them. Instead, I created a logarithmic function that scales their radius in an appropriate manner, reducing the drastic difference in their size. I also used images of the celestial bodies and background to make the simulation more pleasing. Nonetheless, I have kept the code for using circles instead of the images if the user so desires and this can be done by following the instruction in the accompanying readme file (README.md).

Experiments

ORBITAL PERIODS

This experiment was to record the orbital periods of the various celestial bodies and compare them to real-life values to see how accurate the simulation was. At every timestep, I checked if the y-coordinate for a planet changes signs from negative to positive. I did so as all planets start the simulation with their y-coordinate set to zero and would complete an orbit whenever they crossed the y-axis again near the same point they started. All the calculated orbital periods are written to a separate file for each planet. The orbital periods are stored in a list after every orbital period for each planet. The average of all the recorded orbital periods is then written to the respective CSV file of each planet. This is done to reduce error and achieve greater accuracy.

CONSERVATION OF ENERGY

This experiment was to record and observe the total energy of the system. This helps us gain insight into whether the simulation is running properly. A stable value for the total energy indicates a stable system and is given by the sum of the total potential energy and the sum of the kinetic energy of all planets.

SATELLITE TO MARS

This experiment was to simulate the journey of a satellite from Earth to Mars to perform a fly-past and check if the satellite ever returns back to earth. This would help us know the angle and the speed at which to fire the satellite. For this experiment, I used two nested for loops to generate data files for 651 satellite launches, each at angles between 30 and 60 degrees and velocities between 25,000 m/s and 45,000 m/s. I assumed the satellite's weight to be 1000 kg (NASA's Perseverance Mission weighed approximately 1025 kg so this seemed to be a fair estimate). This experiment helped in finding a range of ideal velocities and angles at which to launch the satellite.

Results

ORBITAL PERIODS

The simulation yielded the following results for the orbital periods:

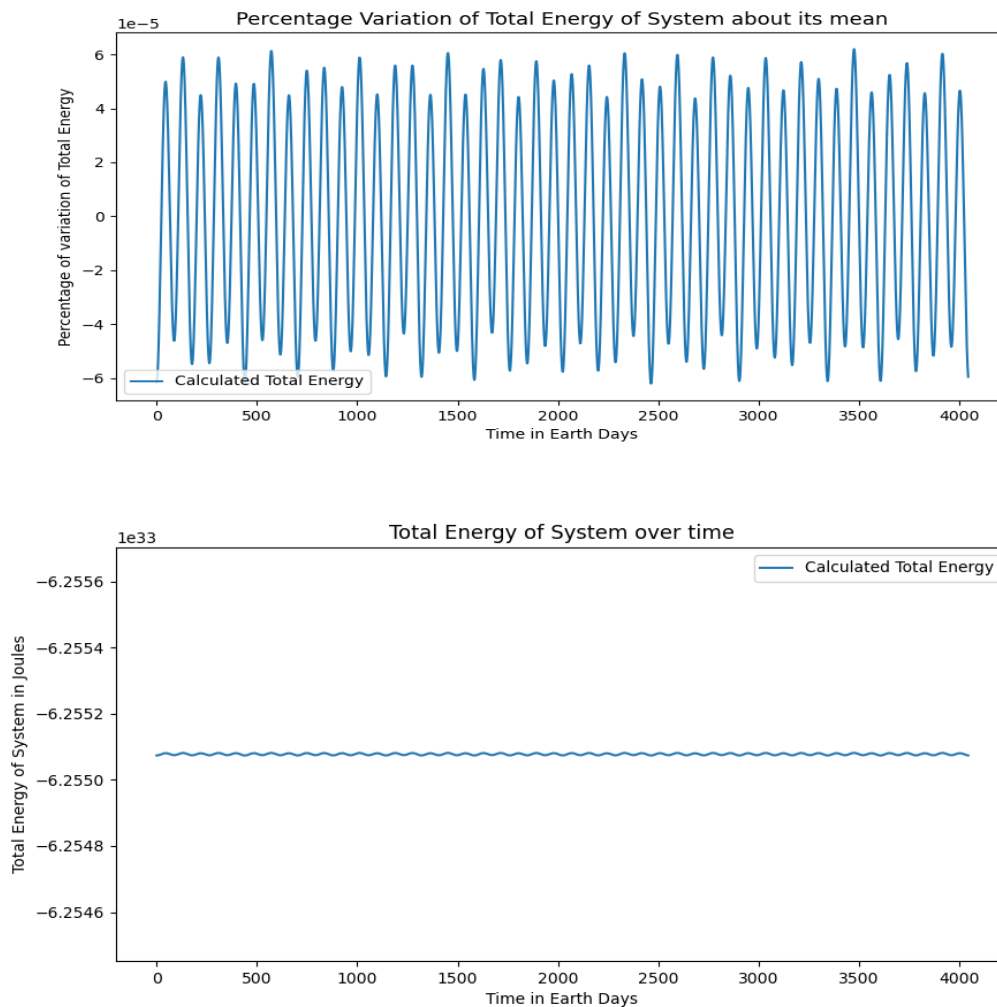
Planet	Simulated Orbital Period (Earth Years)	Actual Orbital Period (Earth Years)
Mercury	0.24130136986301362	0.24
Venus	0.6138812785388128	0.615

Earth	1.0045662100456623	1
Mars	1.8827397260273973	1.88

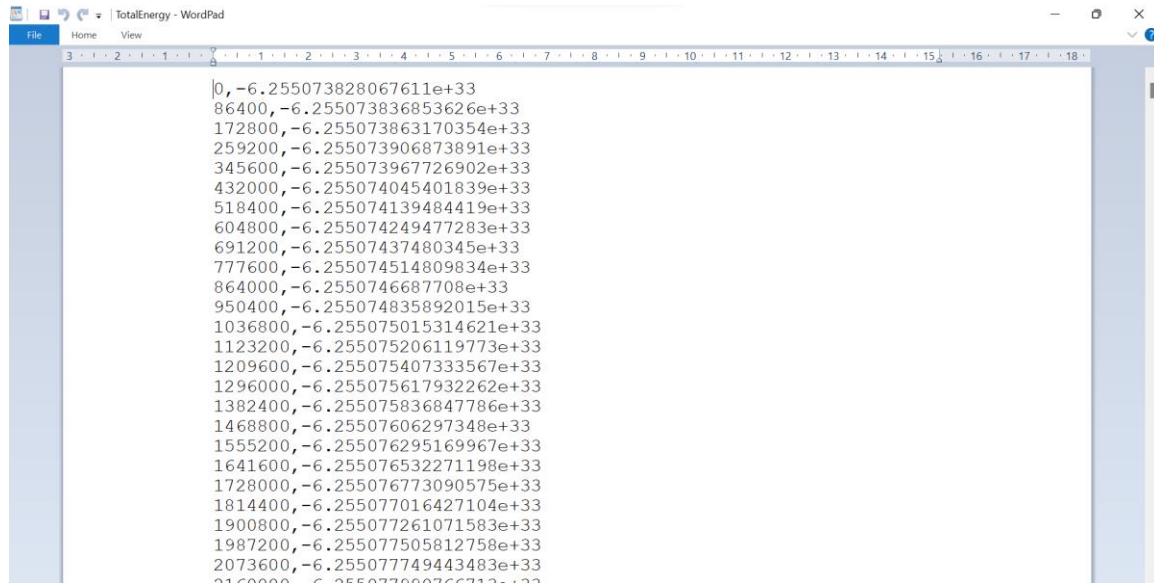
The figures obtained are fairly accurate and are expected as all initial parameters for planets are real-life figures.

CONSERVATION OF ENERGY

As seen from the logs there is variance in the total energy but is extremely small. To effectively visualize this, I plotted two graphs. For the first graph, I plotted the difference between the recorded value upon the mean of all recorded values whole times a hundred against time. This gives us a graph of the percentage variation in the total energy of the system about its mean. As we can see from the graph below, the percentage of variation is always under 6×10^{-5} . For the second graph, I plotted the total energy of the system over time in Joules. On setting the y-limits of the graph to be very close to the mean value, we are finally able to see a 'wobble' graphically.



As the total energy keeps fluctuating about a mean, rather than drifting away over time, the simulation is stable, and the total energy of the system is conserved. The fluctuation is accounted for by the discrete nature of time in the Beeman Algorithm. We can reduce the fluctuation by reducing the time step further, but it is not possible to get rid of it completely. I have also attached a snapshot of the CSV file used to store the total energy of the system along with the time at which the energy was recorded.



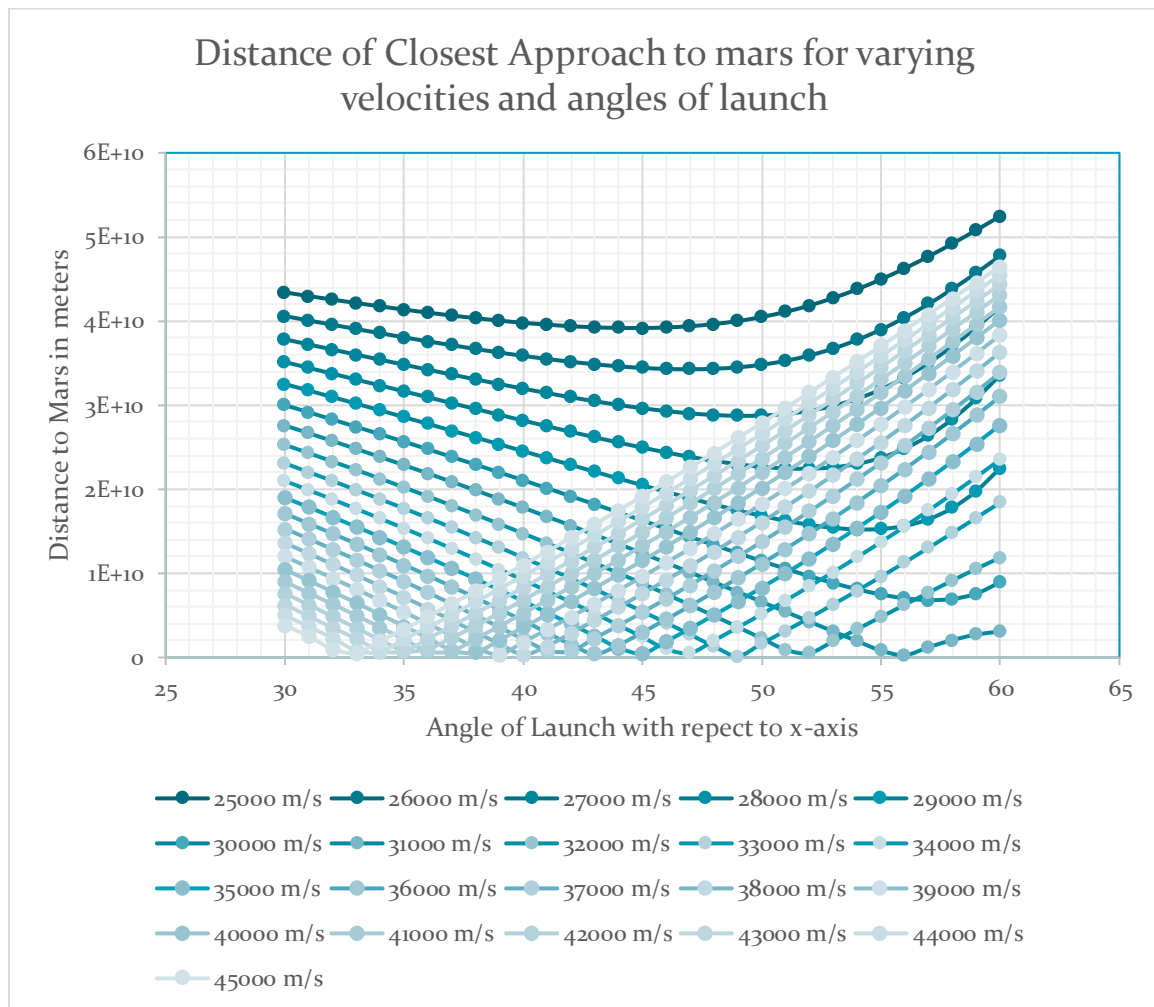
```

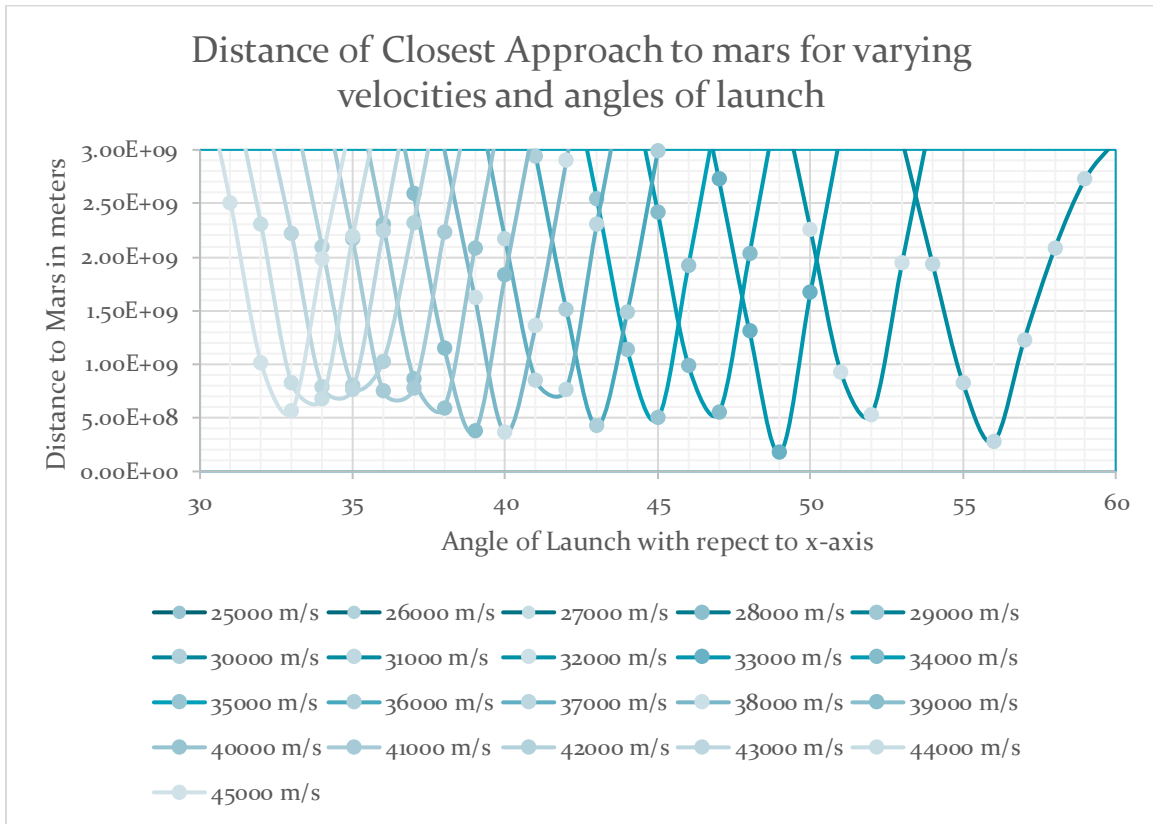
0,-6.255073828067611e+33
86400,-6.255073836853626e+33
172800,-6.255073863170354e+33
259200,-6.255073906873891e+33
345600,-6.255073967726902e+33
432000,-6.255074045401839e+33
518400,-6.255074139484419e+33
604800,-6.255074249477283e+33
691200,-6.25507437480345e+33
777600,-6.255074514809834e+33
864000,-6.2550746687708e+33
950400,-6.255074835892015e+33
1036800,-6.255075015314621e+33
1123200,-6.255075206119773e+33
1209600,-6.255075407333567e+33
1296000,-6.255075617932262e+33
1382400,-6.255075836847786e+33
1468800,-6.25507606297348e+33
1555200,-6.255076295169967e+33
1641600,-6.255076532271198e+33
1728000,-6.255076773090575e+33
1814400,-6.255077016427104e+33
1900800,-6.255077261071583e+33
1987200,-6.255077505812758e+33
2073600,-6.255077749443483e+33

```

SATELLITE TO MARS

The satellite was launched a total of 651 times each time with a different velocity and at a different angle. This was to find what the optimum initial parameters would be so we could save fuel and costs. The results from these experiments have been plotted below.

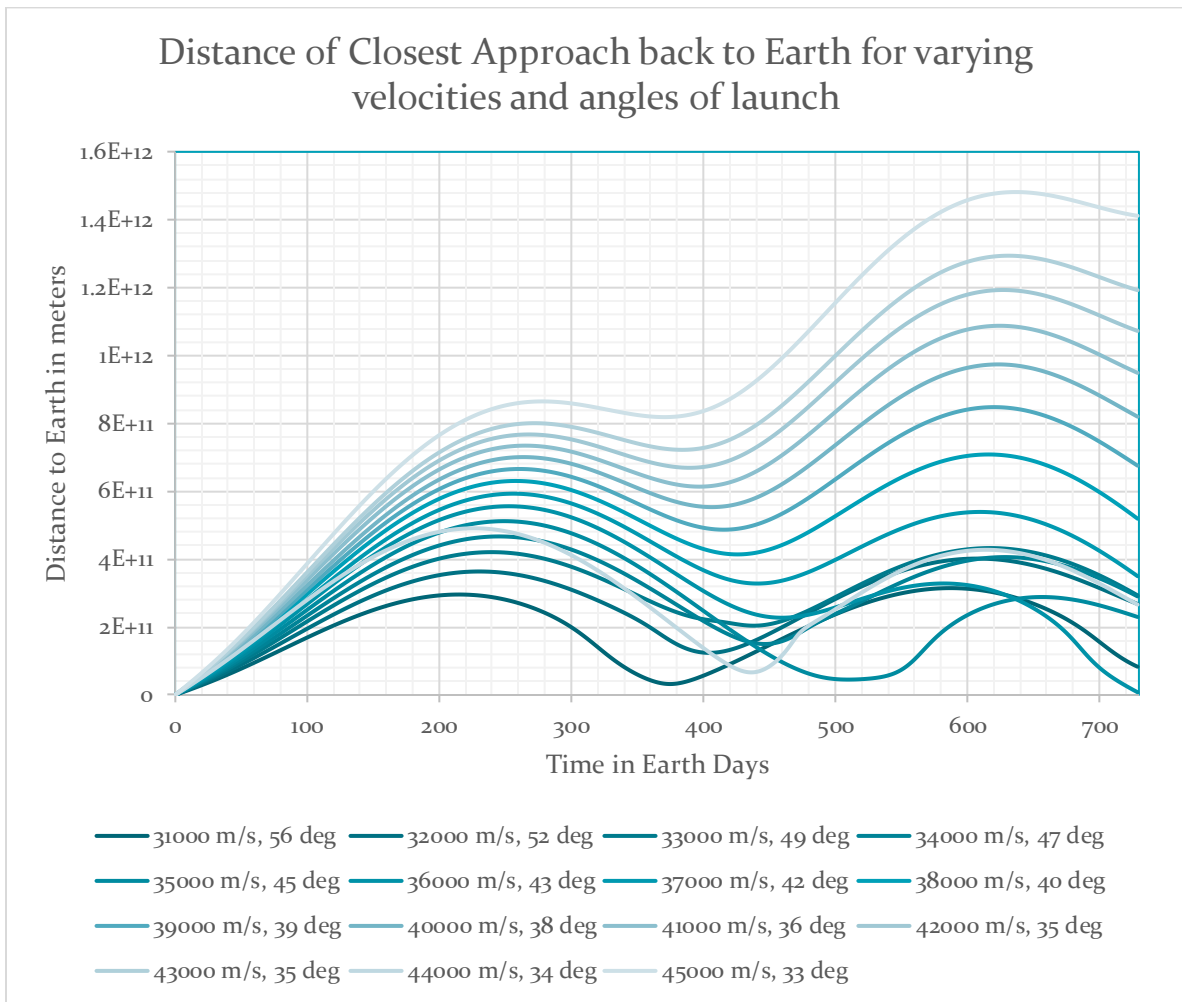




As we can see from the graphs, we can achieve a fly-past of Mars by giving the satellite a velocity of anywhere between 25,000 m/s and 45,000 m/s. These values are relative to a stable point in the solar system rather than being relative to earth. We must keep in mind that the orbital velocity of the earth is around 29,800 m/s and greatly contributes to the satellite's total velocity. However initially when the satellite is launched, the earth's velocity is parallel to the y-axis, but the satellite is being launched at an angle. The simulation achieves this angle by splitting the total velocity supplied to the satellite into a horizontal and vertical component using trigonometry. Specifically, the formula:

$$\sin^2 \theta + \cos^2 \theta = 1$$

Using this property, I split the velocity into two, magnitude times $\cos \theta$ for the x-component and magnitude times $\sin \theta$ for the y-component where θ was the angle of launch with respect to the positive x-axis.



The above graph depicts the distance to earth of the various satellites that were able to successfully perform a fly-past of mars. As we can see some of them naturally are able to come close to the earth. We can also significantly reduce this distance by doing mid-flight corrections, as even a very small change made early on can result in a huge difference. This could even help us retrieve the satellite.

Discussion

ORBITAL PERIODS

Even though my simulation has resulted in fairly accurate orbital periods, there is scope for improvement. By running the simulation with smaller time periods, the values would limit to the actual orbital periods. Also, as of now, we have only accounted for the inner planets of our solar system. We could also try making the simulation more and more accurate, but this comes at the cost of computation.

ENERGY CONSERVATION

The fluctuation in the total energy of the system occurs due to the discrete nature of time in the Beeman algorithm. We can reduce this fluctuation by once again reducing the time-step, however, the good thing is that the total energy does not drift but rather oscillates about a mean. This indicates a stable orbit for all planets.

SATELLITE TO MARS

The satellite does successfully do a fly-past of Mars however as the code does not perform mid-air corrections, we are not able to get as close to Mars on this simulation as NASA's perseverance mission. Also, it is not possible in real-life to achieve such exact velocities and even though there are velocities and angles at which I can send the satellite directly to Mars, it will be unrealistic to achieve this without mid-air corrections. Also, NASA took a completely different approach to send the perseverance mission satellite to Mars. They did this to conserve fuel but it also meant taking longer i.e. almost 7 months. Whereas the approach taken in this experiment takes only 3-4 months depending upon the initial speed. On a side note, the reason behind the velocity values being so high is to account for the orbital velocity of the Earth. Even though before launch the satellite is stationary relative to Earth, it is moving at the same velocity as Earth with respect to a stable point in the solar system. Thus, this velocity is representative to the velocity of the satellite with respect to a stable point in the system.

Conclusion

In conclusion the simulation of the Solar System proved to be realistic, as all the results from the experiments match with real-world data. Due to the object-oriented nature of the code, such a simulation could be worked upon further to make it even more realistic by adding the asteroid belts, and various other planets. I have taken a step in this direction by including a file that contains the data for a simulation of all 8 planets. The orbital periods were extremely close to the actual figures and even the total energy of the system remained conserved showing small fluctuations about a mean. Lastly, the satellite launch was able to reach as close as around 170000 kilometres from the centre of Mars. Even though this figure is not ideal, we can further bring this down by fine-tuning the velocities and angles at which to launch the satellite. Also, as mentioned earlier, mid-flight corrections would help us get significantly closer.

Word Count – 1997

References

nineplanets.org

universertoday.com

nasa.gov

wikipedia.org