# <u>Simulating the motion of planets in our Solar System in two-dimensions</u>

## 1. Introduction

This simulation models the movements of planetary bodies within our Solar System, on a 2-Dimensional plane. To do this I modelled the sun and the first 4 planets surrounding it: Mercury, Venus, Earth, Mars. When initiating the model, it is assumed that  the sun is at the centre of the system, with the planets all starting from the point in their orbit where they're closest to the sun (perihelion). Starting at this point in their orbit also means that their initial speed is equal to their maximum orbital velocity[1]. To find the initial values of the simulation I used the Euler-Cromer Method[2]. Thereafter, the simulation used the three-step Beeman scheme[3] to predict the next time-step's position, then the new acceleration and finally the velocity. The first experiment aimed to compare the simulated orbital periods to their actual orbital periods, and how the accuracy of the simulation was affected by altering the time-step. The second experiment aimed to investigate whether the total energy of the system is conserved during the simulation over a large timescale. The final experiment aimed to simulate launching a satellite from Earth and explore the launch condition(s) that allowed it to do a fly-by of Mars.

## 2. Methods

The main simulation consists of two classes: the SolarSystemBody class, which stores the position, velocity, and acceleration of each body, and the SolarSystem class, which oversees running the simulation itself (i.e., setting up simulation using data from input file, running the simulation, printing output data, and displaying the graphics). To manage the vector quantities, I created my own Vector2D class that performs the basic vector arithmetic functions and can calculate the vector dot products, magnitudes and normalize vectors. I created this class as it notably simplifies the handling of the vector calculations and increases the code reusability of my program. For simulation I decided to implement a 2-step system, where I would first calculate and store all the positions, and only once that was done would I run the methods to display the data. I decided to do this because it was easier to implement, allowed for a better separation between algorithmic code and visualization code, and the increase in computational time was relatively small.

To calculate the orbital periods for the first experiment, I measured how long it takes a body to complete half an orbit (time elapsed until y-coordinates turned negative) and doubled it. While this was by far the easiest way to implement this method, I did notice a couple problems that arose from this implementation. Firstly, this method is evidently less accurate than if I were to measure the full period. However, the more significant issue was the case where, using certain combinations of time-step size and number of iterations, the simulated body would not reach halfway through its orbit and thus the simulation would not calculate an orbital period for the body, returning the default value of 0.

[1] Williams, D., 2021. *Planetary Fact Sheet*. [online] Nssdc.gsfc.nasa.gov. Available at: https://nssdc.gsfc.nasa.gov/planetary/factsheet/

[2] Nikolic, B., n.d. *Computational Methods of Physics*. [online] Physics.udel.edu. Available at: https://www.physics.udel.edu/~bnikolic/teaching/phys660/numerical_ode/node2.html.

[3] Hardy, J. and Ord, R., n.d. *Computer Simulation Course Book*. [online] Www2.ph.ed.ac.uk. Available at: https://www2.ph.ed.ac.uk/AardvarkDeployments/Protected/93310/views/files/Python/Deployments/CompSimLearn2122/inner.node/__Projects/CompSim/Solar/Notes/Solar/web.html.

For the second experiment, when initially contemplating how to obtain the data for the energy-time graph, I considered reading the data off the output file. However, I quickly realised that it was too inefficient to re-read and parse the data I had just written to the output file, and decided to keep the data stored within the program in two arrays, one for each axis. Then you can just call the method that displays the graph after the simulation data has been calculated.

In the final experiment, to represent the satellite object, I created a child class to the SolarSystemBody class called Satellite. This inherits the methods from its superclass but takes different parameters when being instantiated. To instantiate a satellite, you must pass it two SolarSystemBody instances, one which is designated as the origin where the satellite is launched from and the other as the intended destination of the satellite. On top of that, a satellite requires a launch velocity in the form of a Vector2D. A satellite's starting position is 6000 km (radius of earth) to the right of the origin body and has a starting velocity equal to the sum of the origin bodies velocity and the launch velocity. I modelled the satellites this way as it streamlined the integration process into the model, as it manages the satellites together with the other bodies. However, it is important to note that when calculating the gravitational forces on bodies, the program does not calculate the gravitational force the satellite induces on other object, but only the force other bodies induce on the satellite. This is because the satellite has such a small mass compared to the much larger planetary bodies that the force is negligible, and I decided not worth the computing power to calculate it.

## 3. Results and Discussion

### 3.1. Experiment 1

The first experiment aimed to investigate how the simulated orbital periods of each planet was affected by altering the time-step, and how the simulated orbital period compared to the actual orbital period. The data for their actual orbital period is retrieved from the NASA planetary fact sheet[4].

*Table 1. Simulated orbital periods of inner planets.*

| Time-step (s) | Iterations | Mercury orbital period (Earth days) | Venus orbital period (Earth days) | Earth orbital period (Earth days) | Mars orbital period (Earth days) |
|---|---|---|---|---|---|
| **3600** | 10,000 | 88.83 | 225.42 | 366.00 | 687.75 |
| **1800** | 20,000 | 88.38 | 225.04 | 365.67 | 687.38 |
| **900** | 50,000 | 88.19 | 224.90 | 365.50 | 687.21 |
| **60** | 1,000,000 | 88.00 | 224.76 | 365.36 | 687.04 |
| **Real orbit** | **n/a** | **88.0** | **224.7** | **365.2** | **687.0** |

[4] Williams, D., 2021. *Planetary Fact Sheet*. [online] Nssdc.gsfc.nasa.gov. Available at: https://nssdc.gsfc.nasa.gov/planetary/factsheet/.

I had to use varying iteration numbers for each time-step as, due to the way I designed my model, I had to ensure each planet completed at least half an orbit. This however did not affect the accuracy of the experiment. The experiment shows that as the size of the time-step decreases, so does the inaccuracy of the simulation. It is safe to assume that as the time-step tends to 0, the simulated orbital periods tend towards their actual orbital periods.

## 3.2. Experiment 2

The second experiment aimed to investigate whether the total energy of the system was conserved over time. Initially I ran the experiment with a time-step of 43,200 seconds (12 hours) and 10,000 iterations, thus having a resultant timescale of 5000 earth days. Results of this experiment can be seen on Fig 1 below.

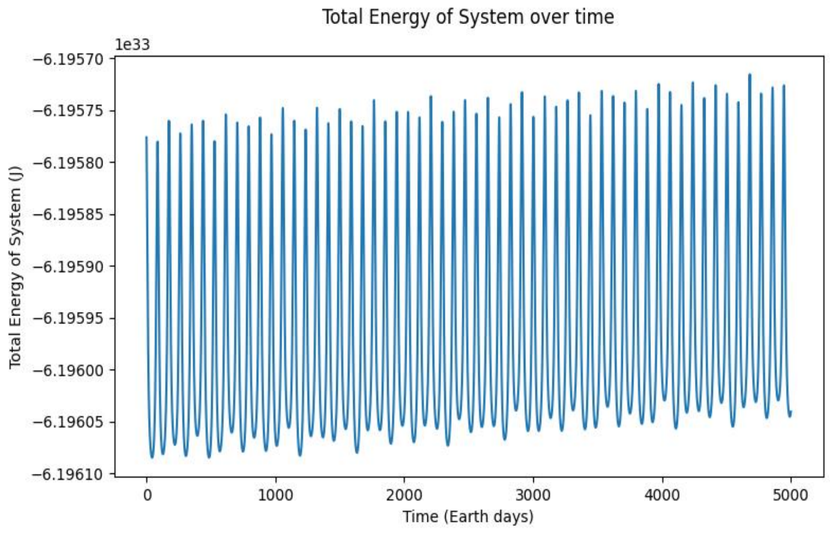

*Figure 1. Total energy of system over time - w/ time-step = 43200 s*

As can be seen on the graph, the total energy of the system appears to hold a consistent oscillation, which is what was expected due to the discretised nature of time in a numerical integration simulation. However, what was unexpected, and slightly worrying, was that there appeared to be a gradual increase in the average total energy of the system. To investigate this further I repeated the experiment with a time-step of 3600 seconds (1 hour) at 20,000 iterations (Fig. 2), a time-step of 1800 seconds (30 minutes) at 100,000 iterations (Fig. 3) and a time-step of 300 seconds (5 minutes) at 1,000,000 iterations (Fig. 4).

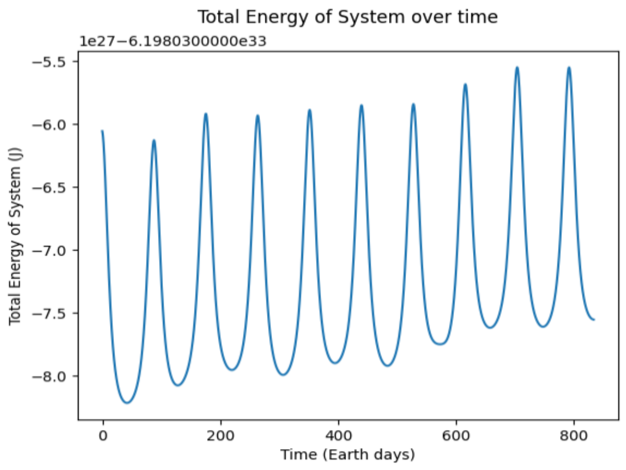

*Figure 2. Total energy of system over time - w/ time-step = 3600 s*

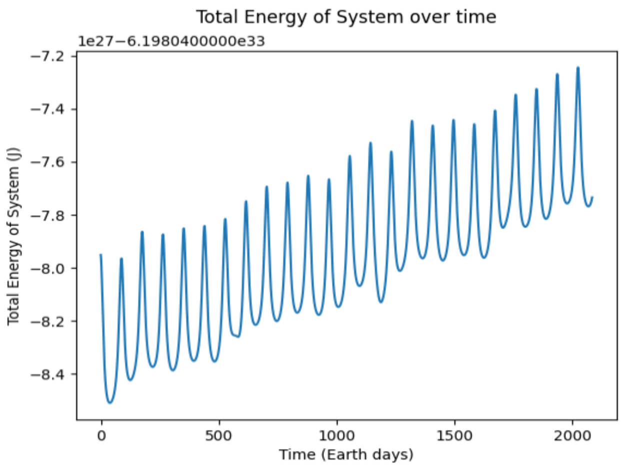

*Figure 2. Total energy of system over time - w/ time-step = 1800 s*

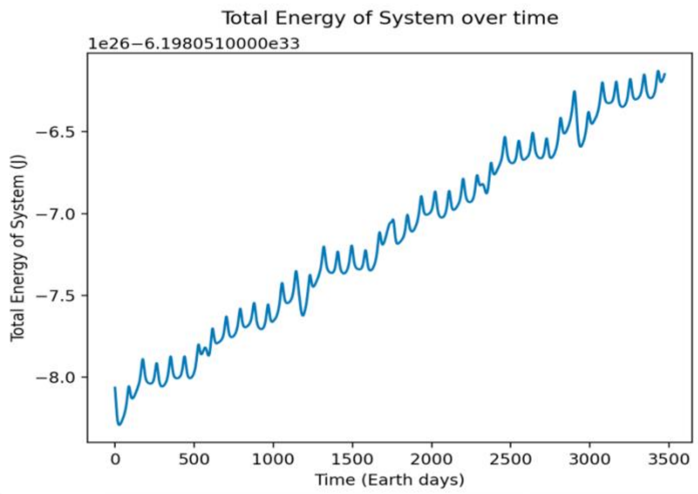

*Figure 3. Total energy of system over time - w/ time-step = 300 s*

It is evident by looking at the data on the graphs, that there is a clear long-term positive energy drift. This goes against the law of conservation of energy, that states that a closed system always has the same amount of total energy, unless it's added from the outside[5]. In a quick investigation into energy-drift, I found that energy drifts may occur in simulations due to numerical integration artifacts arising from using a discrete time-step[6]. However, due to Beeman being a second-order symplectic integrator[7], it may exhibit increases in energy over very long time scales, though the error should remain roughly constant. Looking at the graphs, it appears the energy error is not constant but rather increasing over time, meaning the error lies not due to the Beeman algorithm itself but rather my implementation of it. After considerable research and investigating, the best reason I could attribute this drift to was roundoff error within my implementation of the algorithm [8] .

## 3.3. Experiment 3

The third experiment aimed to investigate the launch velocity required to send a satellite on a fly-by of mars. I gave my satellite ("Argo") the assumed mass of 1000kg as this is roughly equal to the Perseverance's mass and is roughly what your average medium-large satellite would weigh[9]. Also, since the satellite is to be launched at the start of the simulation and all planets within my simulation initiate with only vertical velocity, I decided to launch the satellite perpendicular to the motion of the earth i.e., with no vertical component of velocity. With Earth having an escape velocity ≈ 11.2 km/s [10] I thought that would be a good place to start and would decide consequent values base on trial and error and analysis of the graphical display of the trajectory. All experiments utilised time-step = 3600 seconds (1 hour).

[5] J.M.K.C Donev et al., 2021. *Law of conservation of energy - Energy Education*. [online] Energyeducation.ca. Available at: https://energyeducation.ca/encyclopedia/Law_of_conservation_of_energy.

[6] En.wikipedia.org. n.d. Energy drift - Wikipedia. [online] Available at: https://en.wikipedia.org/wiki/Energy_drift.

[7] En.wikipedia.org. n.d. *Beeman's algorithm - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Beeman%27s_algorithm.

[8] Ima.umn.edu. 2022. *Energy Drift in Molecular Dynamics Simulations: What causes it, can we quantify its affects, and should we even try? | Institute for Mathematics and its Applications*. [online] Available at: https://www.ima.umn.edu/2015-2016/SW2.22-24.16/24629#:~:text=Often%20this%20drift%20is%20attributed,associated%20with%20the%20MD%20forcefield..

[9] Miller, A., 2022. *How big is that satellite? A primer on satellite categories*. [online] Viasat.com. Available at: https://www.viasat.com/about/newsroom/blog/how-big-is-that-satellite--a-primer-on-satellite-categories0/#:~:text=Large%20satellites%20(more%20than%201%2C000,the%20two%20current%20LANDSAT%20satellites..

[10] Wiliams, D., 2021. *Earth Fact Sheet*. [online] Nssdc.gsfc.nasa.gov. Available at: https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html.

*Table 2. Table representing the closest distance achieved to Mars by a satellite with varying launch velocities.*

| **Launch velocity (m/s, m/s) (xVelocity,yVelocity)** | (11200, 0) | (11000, 0) | (11100, 0) | (11090, 0) | (11080, 0) | (11075, 0) | (11076, 0) | (11074, 0) |
|---|---|---|---|---|---|---|---|---|
| **Closest distance to mars (km)** | 2,142,947 | 1,432,154 | 449,439 | 269,400 | 87,880 | 9,559 | 15,381 | 20753 |

Due to velocity vectors being represented by integers, a launch velocity of (11076 m/s, 0 m/s) was the highest degree of accuracy that I could achieve. The satellite got within 9,559 km from the centre of Mars, which is quite close when you consider that Mars' radius is ≈ 3400 km[11]. To reach this point, the satellite took 148.54 Earth days. Comparing this to the NASA Perseverance mission which took 203 days[12], my satellite evidently took less time but is not out of the realms of possibility. In 2006, an Atlas V rocket (the same type of rocket that launched the Perseverance) launched the New Horizons mission into an escape trajectory with a speed of about 16.26 km/s[13], a launch speed much greater than my satellite's.

Within my simulation, the Argo is launched from earth with a velocity much greater than Mars' (due to Earth's greater orbital velocity). Thus, it is launched outwards and intersects with Mars' orbit as it is at the apex of its trajectory, allowing Mars to catch up with it as it scrubs speed. Once Mars passes, the Argo falls back towards the sun and remains in a seemingly stable elliptical orbit with the perihelion roughly in line with Venus' orbit and the Aphelion in line with Mars' orbit. The orbit of the satellite doesn't get back to earth as they stay out of sync.

While these results are valid within the simulation, it does not accurately portray how satellites are launched, as in the simulation the satellite is catapulted into its trajectory using only the launch velocity, 100km from Earth's surface. In reality, rocket launches are more nuanced, with multiple acceleration and deceleration phases and several trajectory correction manoeuvres for necessary adjustments[14]. Furthermore, the simulation does not model air resistance originating from the rocket passing through the atmosphere.

## 4. Conclusion

The final implementation of the code created a model that can very accurately represent the real movement of bodies within our Solar System. The main issue that I would address if I had more time, would be to fix the long-term energy drift within my simulation, as it is an issue of implementation rather than lack of features. Another feature that I would try to implement, is to get the program to calculate the whole set of launch velocities that will allow a satellite to reach Mars, instead of doing it manually. Nevertheless, I found the project extremely fun and allowed me to research fascinating topics which I never would have :)

Word Count: 1927

[11] Wiliams, D., 2021. *Mars Fact Sheet*. [online] Nssdc.gsfc.nasa.gov. Available at: https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html.

[12] Mars.nasa.gov. n.d. *Mars 2020 Perseverance Rover*. [online] Available at: https://mars.nasa.gov/mars2020/.

[13] En.wikipedia.org. 2022. *New Horizons - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/New_Horizons.

[14] NASA/JPL. n.d. *Mars 2020 Perseverance Launch Press Kit | Mission Overview*. [online] Available at: https://www.jpl.nasa.gov/news/press_kits/mars_2020/launch/mission/.