Simulation of the Inner Solar System, Satellite Trajectories and Energies of the System

Declan Mathews
s1610357

Introduction

The aim of this experiment was to produce a working code that modelled the inner solar system in 2D using Object Orientated Python. The inner bodies included the Sun out to Jupiter, with circular orbit starting conditions using a three-step Beeman integration scheme that updates the velocity and position of the bodies. The orbital periods and the total energy of the system were checked, along with the initial conditions for satellite launches from Earth to Mars for a fly pass and from Earth to slingshot around Jupiter and how they affect the trajectories.

The solar system was set up to have the data of the bodies read in from files, with each planet beginning on the positive x axis with an initial velocity determined by:

$$v_{inital} = \sqrt{\frac{GM}{r}} \qquad \textbf{Eqn. 1}$$

with G being the gravitational constant, M being the mass of the Sun and r the radius of the planet.
The forces on each planet will be governed by:

$$\vec{F} = \frac{Gm_1m_2}{r^3}\vec{r} \qquad \textbf{Eqn. 2}$$

where $\vec{r}$ is the position vector of the body.
Using these base principles and the three-step integration scheme:

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{6}[4\vec{a}(t) - \vec{a}(t+\Delta t)]\Delta t^2 \qquad \textbf{Eqn. 3}$$

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{1}{6}[2\vec{a}(t+\Delta t) + 5\vec{a}(t) - \vec{a}(t-\Delta t)]\Delta t \qquad \textbf{Eqn. 4}$$

with Δt being a small time-step, $\vec{v}$ being the velocity vector of the body and $\vec{a}$ the acceleration vector of the body.

Methods

Two classes were used to construct the underlying simulation of the solar system. These were the `Body` and `OrbitalMotion` classes. There was also a programme to begin the simulation and one to create graphs of the energy data. There were several files for the data of the bodies and energy of the system.

The *Body* class' function was to read in the data from the files and create the mass, position vector and velocity vector arrays using numpy, which are then used to initialise the `OrbitalMotion` class. This allowed the addition of extra bodies to be separated and easily adjusted and meant that `OrbitalMotion` was only concerned with the actual motion of the bodies and the animation.

`OrbitalMotion` construction began from a simple two-body system with a basic integration scheme to determine the proper iterations required over the arrays of mass, position vector and velocity vector and was updated to take all bodies before finally implementing the final integration scheme which required more calculations. This resulted in a slower pace but caught errors more easily and took small steps to reach a far goal and made the coding much simpler. This was especially important as an 'animation on the fly' technique was used which can be more complex to implement but allowed indefinite running of the simulation allowing a better study and more data to be obtained. This class was initialised with the lists from the `Body` class and stored as self-lists which could then be easily updated.

The required patches were added in the animation methods (`animate` and `run`), with the satellites being set far out of the screen so their patch was not seen, their mass, position and velocity not yet appended. They were added 2 and 3 Earth years after initiation for the Mars and Jupiter probes respectively as per required by the question.

In each animation iteration, the `step` method was completed. In this the iterations for calculations were governed, while the calculations were carried out by respective methods. This allowed the calculations to easily be altered, especially when implementing the final integration scheme, and then when bugs were encountered in a calculation or iteration the debugging was much more manageable.

For the integration, the previous, current and next step accelerations were needed. I decided to store the previous as a self-value and then calculate the current and return using the same function, `finda`. This meant when the position was updated first using the current and previous (which was set to 0 initially) the previous could be stored and the current returned and stored in an array. The next acceleration was then calculated and returned using the new position and stored in a different array. Both could then be passed for the new velocity calculation and the previous acceleration updated in this method. This allowed simple calculations with a list value corresponding to each body and iteration required over each other body and summing the total to be stored.

The kinetic energy and gravitational energy could be calculated over the same iteration scheme using methods for the calculation and storing the summed results in arrays which could then be passed to a separate method, `Ewrite`. This was called if a random number generated was below a certain value, ensuring it was not called for huge quantities of data. This calculated the total and wrote the time passed, potential energy total, kinetic energy total and total energy out to a file. This time was increased every iteration by the time-step and stored as a self-value in Earth years, to avoid huge numbers. A separate programme was written to print a graph of this energy against time, to check its conservation. Multiple time arrays were needed to print a graph for the 3 energy totals.

This time was also used to determine the orbital period of the planets. The initial position was also passed to `OrbitalMotion` as a separate array to avoid updating it, and then the magnitude of the distance between the planet and its initial position was calculated each iteration. When this was less than a certain value the time was printed. There was also a check for the time being greater than 0.23 Earth years, which prevented printing at the very beginning, but not for the first orbit of Mercury. The value used was 4.8e9 which was small enough to be assumed at the same point but big enough to not be skipped by the calculation as a large time-step is needed for the simulation to occur over a short time. A `self.stop[]` value for each planet was included which increased out of range to prevent multiple prints.

In `step`, there was also conditional statements which, when the 2 and 3 years had passed, the mass of the satellite and its velocity and position vectors were appended for the Mars and Jupiter satellite respectively. The position was taken as Earth's position plus 1 to the x coordinate. This is slightly unrealistic as treating the Earth as a point mass and being that close produces a huge force required to overcome. It was thought that this would give a similar process to escape velocity and drag as well as other prohibitive forces. Reasonable times and velocities were achieved and so the value was kept as this. Given more time the experiments would repeated with a position at Earth's surface.

A `self.satellitecheck[]` array was implemented to increase out of range when this happened to prevent continuous appending. For the Mars satellite, there was a check if the probe was close enough to Mars, by the magnitude of the vector between the two being less than a set distance away and another check value (to prevent printing twice), which, if both passed, printed the travel time to compare to the Viking probe. The trajectories were then studied of each to see the effects of the initial conditions, some brief comments were included in the code of these results.

Results

Animation

The base animation ran as planned, with the planets executing circular orbit based on Eqn. 1 through 4. Reasonable time step and interval values are used but can be altered depending on requirement, as can the number of iterations affecting the length of the animation.

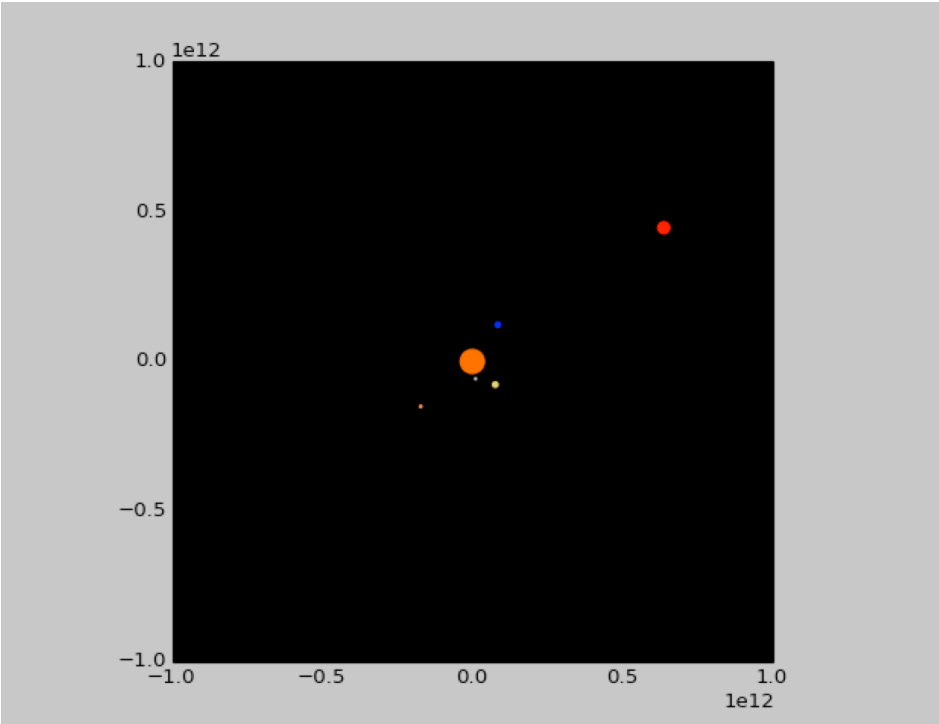

Fig. 1 An image of the programme simulation, showing the planets; Mercury, Venus, Earth, Mars and Jupiter (in order of increasing distance from the Sun) orbiting the Sun

Orbital Time Periods

The orbital time periods recorded at the set time step, with the required distance for printing are:

Planet	Measured Orbital Time-Period /Earth Years	Literature Value of Orbital Time-Period/ Earth Years
Mercury	0.24083	0.241
Venus	0.61158	0.615
Earth	0.99817	1.00
Mars	1.8759	1.88
Jupiter	11.832	11.9

Fig. 2 A table of the experimentally determined and literature values of the orbital time period of the 5 inner planets for a time-step of 100000ms^{-1}, within 4.8x10^{9}m of the original position and interval = 1

Energy Conservation

The total energy of the system over the time for a sample simulation was recorded and graphed producing:

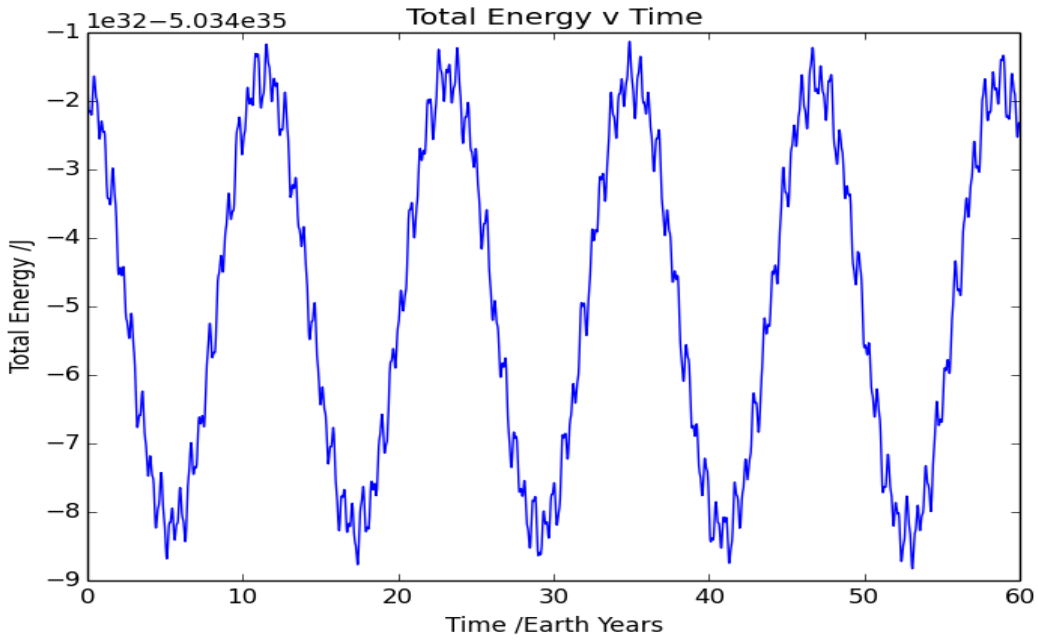

Fig. 3 A graph showing the total energy of the system in Joules against the time passed in Earth years

A graph of the kinetic energy total, potential energy total, and total energy against time was also produced.

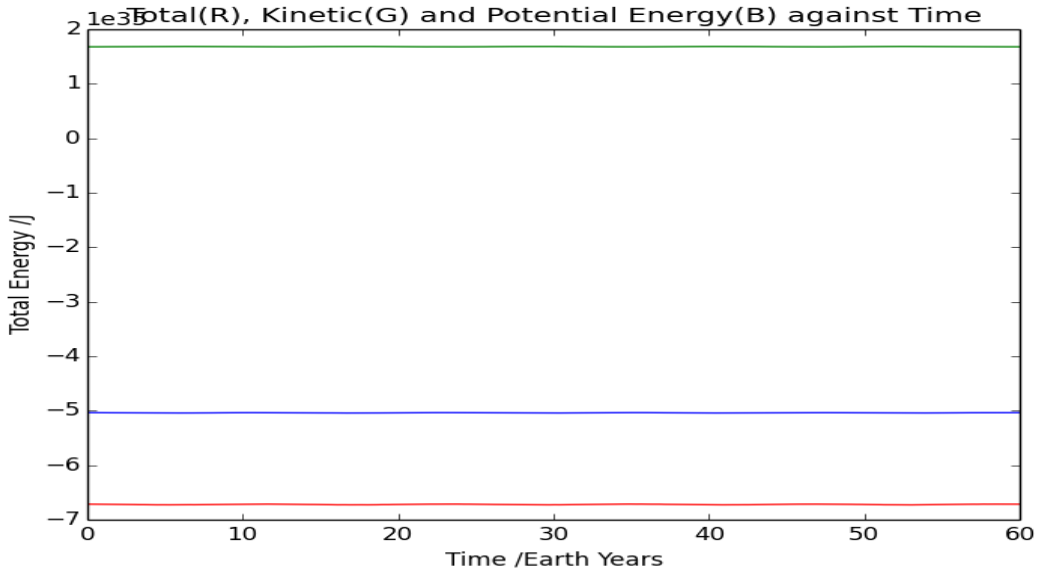

Fig. 4 A graph showing the total energy(blue), potential energy(red) and kinetic energy(green) against the time passed in Earth years

Satellite to Mars

The initial velocity conditions for the launch 2 Earth years after the beginning of the simulation were varied and the effect on the time to reach Mars and the trajectory were studied. Due to the time limit on the project, the data of the position and speed of the trajectory was not studied, instead only the time for the satellite to reach Mars and a comment about its trajectory was made, and related to other aspects of the simulation. The results were then compared with the Viking probes launched by NASA.

Trial and error was used, beginning with an estimation, for the velocities to fly past Mars. All the satellites launched to Mars remained in orbit on the grid in slightly elliptical orbits about the Sun. The characterisation was then the estimated distance between the Earth and the probe for its return, and the time to reach Mars. The results for the conditions which reached Mars are shown in fig. 5.

Velocity X Component /ms^{-1}	**Velocity Y Component /ms^{-1}**	**Time to Reach Mars /Earth Years**	**Comment on Trajectory; Return to Earth**
4500	32300	0.58258	Does not return closely
2000	33000	0.87094	Returns quite closely; dots overlap
2100	33000	0.88678	Returns quite closely; dots overlap
2500	32900	0.82341	Does not return
2200	32950	0.84876	Uncertain; Closer than [4500, 32300] but further than [2000,33000] and [2100, 33000]

Fig. 5 A table showing the successful satellite launches to pass by Mars, the time to reach Mars and their trajectory after this; based on its return to Earth for the same conditions as before

Satellite to Jupiter

This launch was after 3 Earth years and its trajectory based on the initial conditions was also studied, more precisely, the slingshot and its sensitivity to the initial conditions. Again, the time limit restricted the trajectory study and so only comments were made about it. Only some of the data for the launches reaching Jupiter are included, with the slingshot being characterised by whether it slingshot, the exit speed by rough estimation and relative to other planets, the direction of exit and whether it returned to orbit around the Sun. Due to the sensitivity, trial and error was used to find rough values for possible slingshots, then a suitable x component and y component were found. One was held constant and the other varied for both x and y. By noting y was much larger, more results were taken and with smaller differences, as it had the larger effect. The results are seen in fig. 6 and fig. 7.

Velocity X Component /ms^{-1}	Velocity T Component /ms^{-1}	Slingshot Y/N	Returns to Orbit Y/N	Trajectory and Speed Comment
2500	38450	N	Y	Dragged into wide elliptical orbit, slower wide speed that Jupiter
2500	38485	Y	N	Fired at an incredibly high speed (faster than any planet) downwards slightly to the left off grid
2500	38490	Y	N	Propelled at similar speed to Earth or Mars, but diagonally to the upper left out of the grid
2500	38495	Y	Y	Slingshots into elliptical orbit in the opposite direction to all the other bodies. Speed is quite slow at furthest point, roughly Jupiter's orbit radius
2500	38500	Y	N	Fires moderately quickly to right and back around sun (opposite direction to other bodies) and then exits grid at the bottom
2500	38505	Y	N	Propelled quite quickly to the right and slightly down out of the screen
2500	38510	Y	Y	Has a very wide elliptical orbit (off grid at one point) at a speed not too far from Jupiter's
2500	38525	Y	Y	Has an elliptical orbit at a slower speed than Jupiter with a smaller widest point than before
2500	38650	Y	Y	Has a very slow slingshot and speed and orbits on a less elliptical path again

Fig. 6 A table showing the characteristics of satellite launches to Jupiter from Earth with varying Y components of velocity; characterisations based around a slingshot event

Velocity X Component /ms^{-1}	Velocity Y Component /ms^{-1}	Slingshot Y/N	Returns to Orbit Y/N	Trajectory and Speed Comment
2000	38525	Y	Y	Slow speed and loops round Sun moderately closely
2400	38525	Y	Y	Slow slingshot but loops round Sun very quickly with large exit path
2550	38525	Y	Y	Slightly faster slingshot than previous with wider loop at a slower speed
3000	38525	Y	N	Propelled very quickly diagonally down to the right and off the grid
3500	38525	N	N	Accelerated on past Jupiter and off grid to left

Fig. 7 A table showing the characteristics of satellite launches to Jupiter from Earth with varying X components of velocity; characterisations based around a slingshot event

Discussion

Orbital Periods

The measured periods were all very close to the literature values, to two or three significant figures. This validates the competency of the code and simulation. The small error is due to the large time-step required to run the simulation and notice movement, also in the integration scheme itself which is not perfect. Other integration schemes could be used but were not asked for in this project. Similarly, a smaller time step could be used but would require much more time to run the animation and so is impractical. The method used to check whether the planet has reached its original position is also imperfect, as it is only for within a certain distance. However, this was required due to the time-step and possibility of skipping the original position.

Energy Conservation

Fig. 4 clearly shows a constant kinetic energy, constant potential energy and constant total energy (a sum of kinetic and potential). However, fig. 3 shows the total energy performing simple harmonic motion between roughly -5.43×10^{35} J and -5.34×10^{35} J. This is a relatively small fluctuation and is due to the error created by the large time-step and imperfect integration scheme. The results can then be said to show a conservation of energy which is to be expected from the model, and so further validates the simulation.

Satellite to Mars

Satellites were launched that reached close enough to Mars to be considered a fly past. The results show various trajectories based on initial conditions. For launching from 1 m from the point mass of Earth the initial velocity had to be larger than the speed of the Earth. The first satellite to reach Mars did it in 0.58 Earth years, which was months faster than Viking probes which were the comparative real life launches. Further tests found launch conditions

that created a trajectory and time close to Viking 1, being 0.833 Earth years. However, while there were times closer to this, trajectories with slightly longer times were found which allowed the satellites to almost return directly to Earth, close enough to be considered returned by this simulation. This is most likely due to the previous imperfections. The initial velocities are also not unrealistic and so due to the similarities with the Viking probe this experiment is considered a success.

Satellite to Jupiter

The results show that due to the much larger y component, changes in it have a larger effect on the trajectory and thus slingshot effect than the x component. This is due to the position of Jupiter relative to Earth. The results also show that as little as a $5ms^{-1}$ change in the y component can cause a large change in the slingshot, going from returning to orbit with a moderate speed elliptical orbit to being propelled off the grid at a very high speed. This is due to the bodies being treated as point masses and when coming very close produce large forces. This is not completely consistent with the physical world as they are not point masses. However, this is a simplified model and works well for the simulation and timescale of the project at this ability level. The trajectory is still relatively realistic and runs well in the simulation as expected with slingshot and returning to orbit in achievable initial conditions. The sensitivity to initial conditions is likely much higher in a more realistic model and so NASA would require many experiments before launches. Likely with a much lower time-step, more accurate integration scheme, 3D modelling, not treating as point masses, elliptical conditions and other factors like other debris in space. Super computers would be essential in running such complex simulations, and many members of staff to delegate the work. This emphasises the use of object orientated code to design, but likely a faster language to run. Also, the development of new computers, like the field of quantum computing, would escalate the level of work capable for extremely complex simulations.

Conclusion

The animation runs smoothly and produces the expected result for the simulation, orbital periods correct to 2 or 3 significant figures and energy conservation is maintained. The satellites follow the expected path, with the Mars launch like the Viking probe in time and trajectory and the Jupiter launch produces a slingshot as expected.

Bibliography

https://en.wikipedia.org/wiki/Viking_program - Masses and data on probes

https://nssdc.gsfc.nasa.gov/planetary/factsheet/ - Data for bodies

https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html - Data for Sun

https://www.nasa.gov/mission_pages/viking - Data for Viking mission

https://www.universetoday.com/128259/long-take-get-jupiter/ - Data for Jupiter launch

https://nssdc.gsfc.nasa.gov/planetary/factsheet/planet_table_ratio.html - Data in Earth years