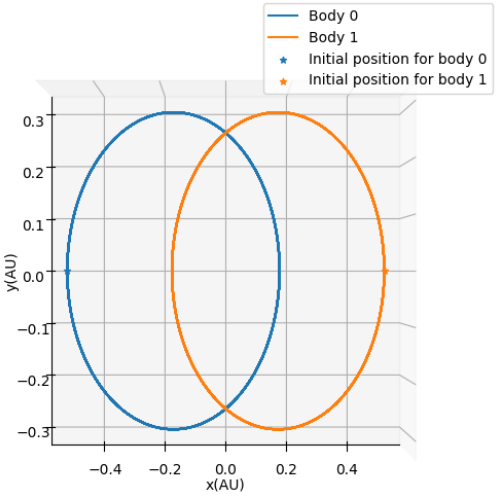
N-body problem. Celestial stability and chaos.

Alejandro Sanz Ramirez, for the University of Nottingham

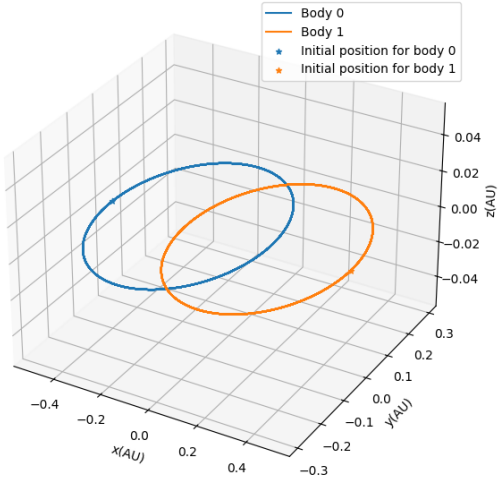
November 2023

Instructions on how to execute the code are provided in Section 4, at the end of the document.

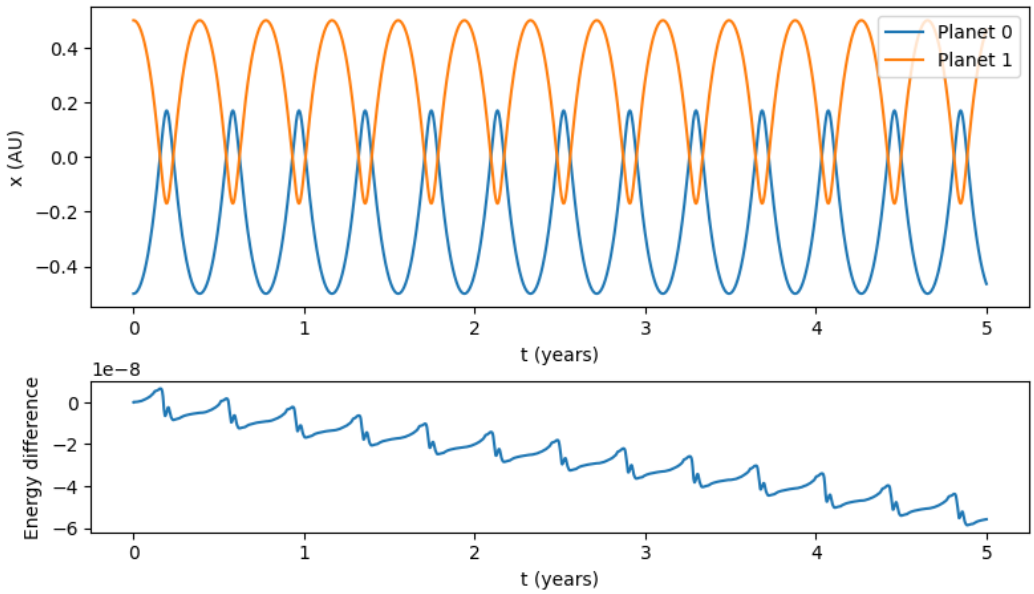
1 Validation.



(a) Top-down view of position



(b) 3D view of position



(c) X position and ΔE against time

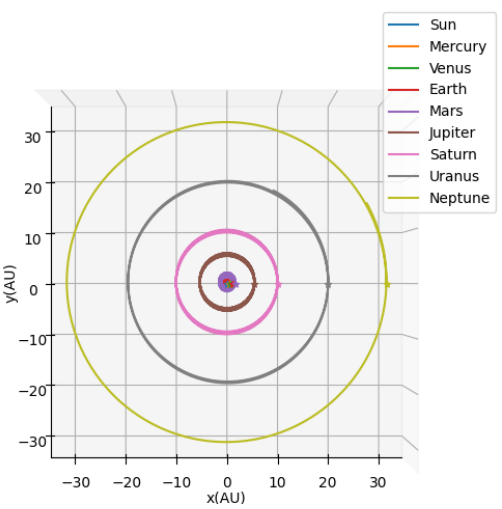
Validation test case: Two planets with the same mass orbit each other for 5 years. While close approaches result in a sudden drop in total energy, the relative energy difference (ΔE) is minuscule. The code implementation yields the same result for both available methods. The 3D view illustrates that each coordinate is managed independently and appropriately.

2 Generalisation - Implementation on known systems.

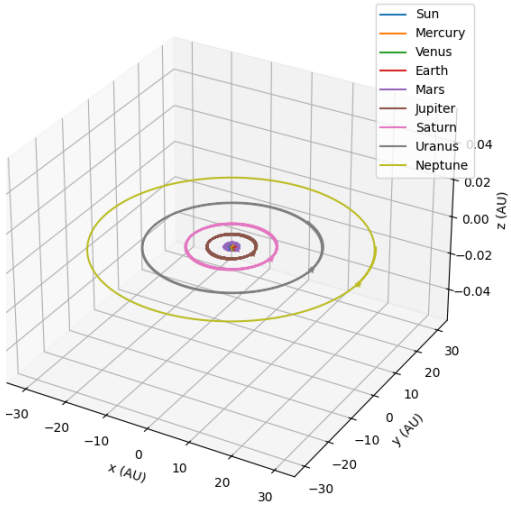
2.1 The Solar System.

Figure 2 illustrates the computation spanning 180 years for the eight planets in the solar system. Their initial positions are aligned on the y = 0, z = 0 plane, with their x coordinates equal to their semi-major rotation axes. Initial velocities are set along the y-axis and correspond to the mean velocities in each of their orbits. The data was obtained from NASA, and 3D viewpoints are provided to ensure Ecliptic confinement.

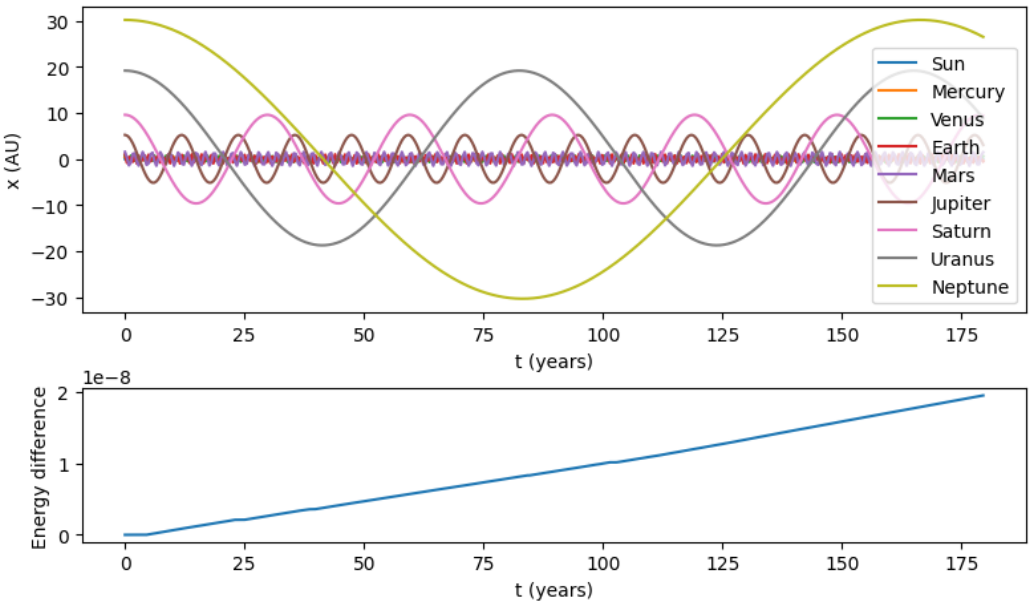
Figure 3 on the next page illustrates the computation exclusively for the four inner planets and the Sun. It's evident that these models of the solar system are not accurate or stable over the long term. When the planets align, their velocities may not match the inputted values, and they need not be precisely contained in the same plane. However, this representation serves the purpose of demonstrating that they approximately maintain stable orbits, due to the great distances between them and the Sun's immense mass.



(a) Top-down view of position

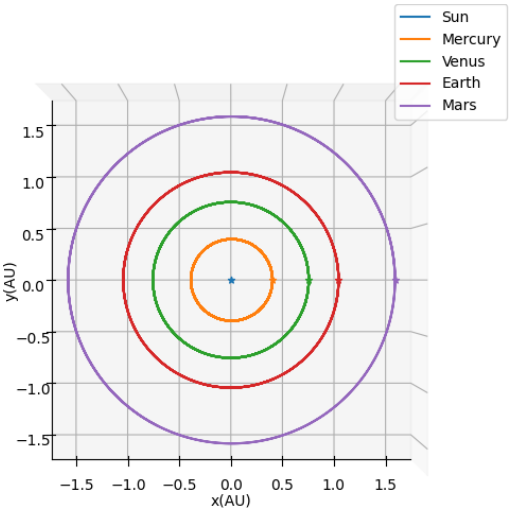


(b) 3D view of position

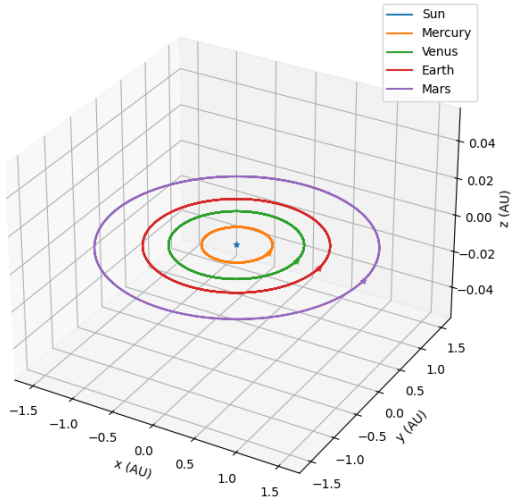


(c) X position and ΔE against time

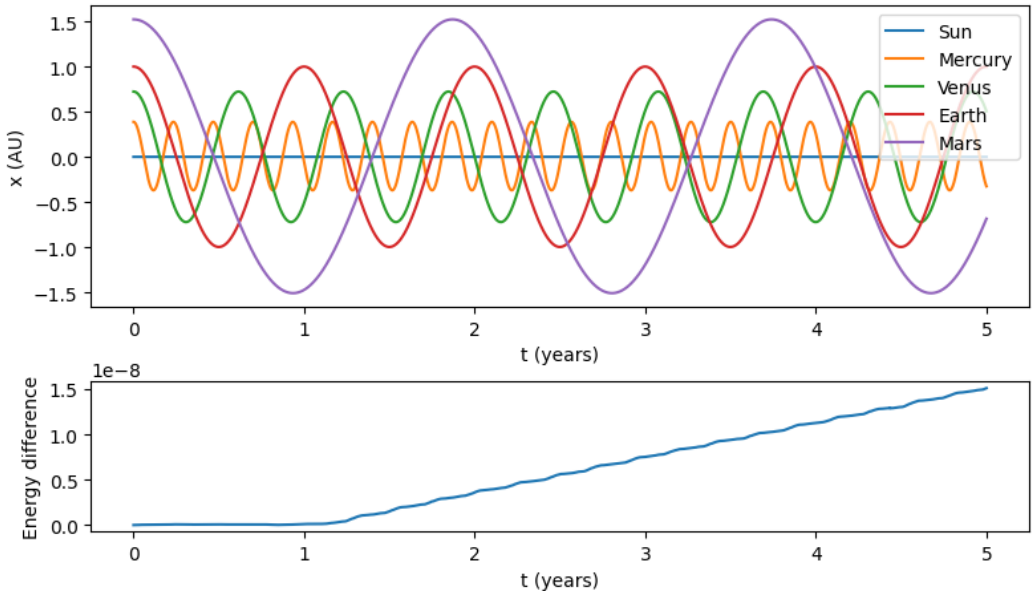
Figure 2: The 8 solar system planets orbiting the Sun over a span of 180 years.



(a) Top-down view of position



(b) 3D view of position

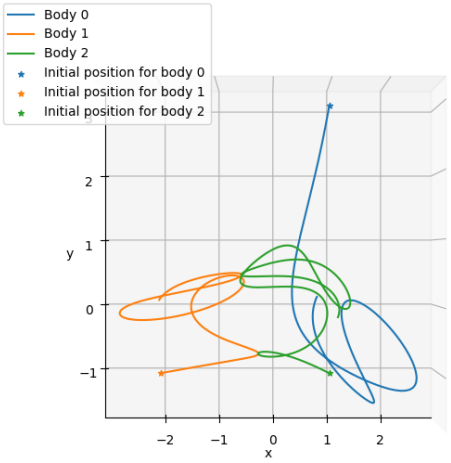


(c) X position and ΔE against time

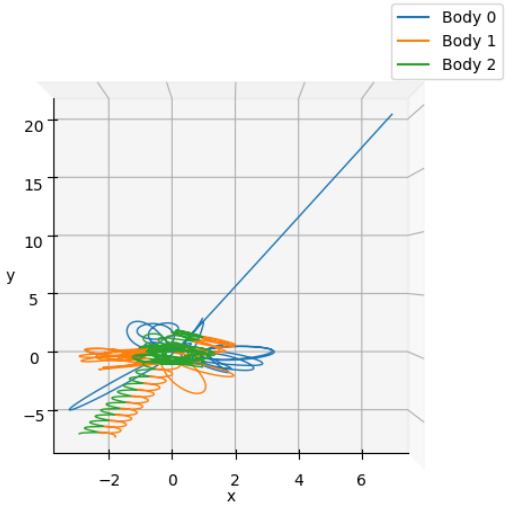
Figure 3: The 4 inner solar system planets orbiting the Sun over a span of 5 years.

2.2 Burrau's problem

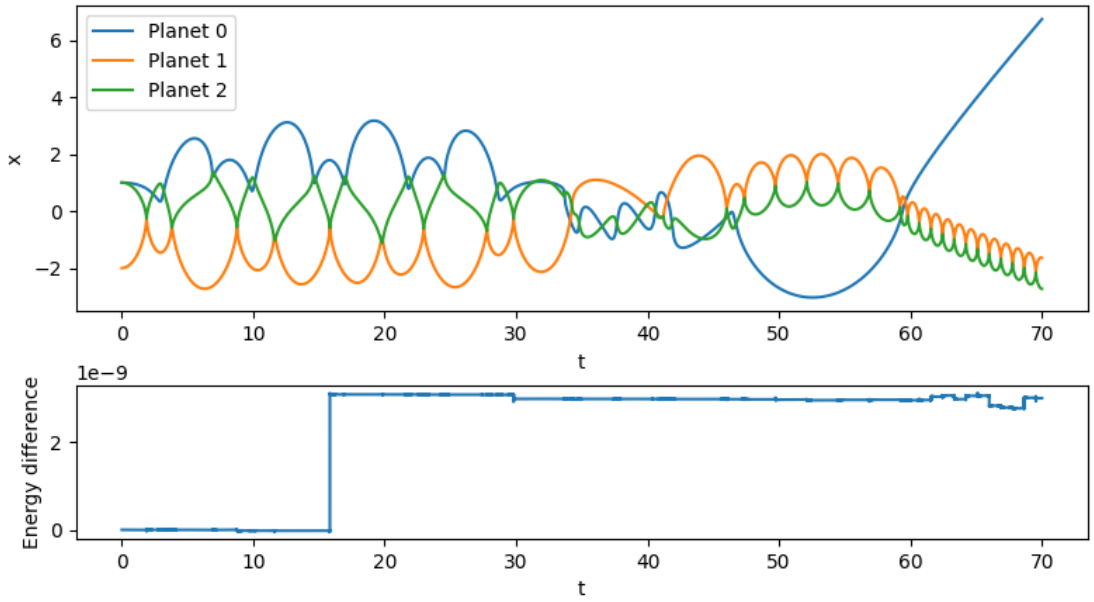
The provided paper gives instructions on how to set up this particular system with 'special' units, for $G = 1$. This greatly simplifies the calculations, which is paramount when calculating close encounters. For this part, significantly more time steps and tolerance had to be set. The bodies are arranged by ascending mass in the legend. **An animation, much more illustrative than the following figure, is included "Figure4.mp4"**.



(a) Positions for time from 0 to 10.



(b) Positions for time from 0 to 70.



(c) X position and ΔE against time

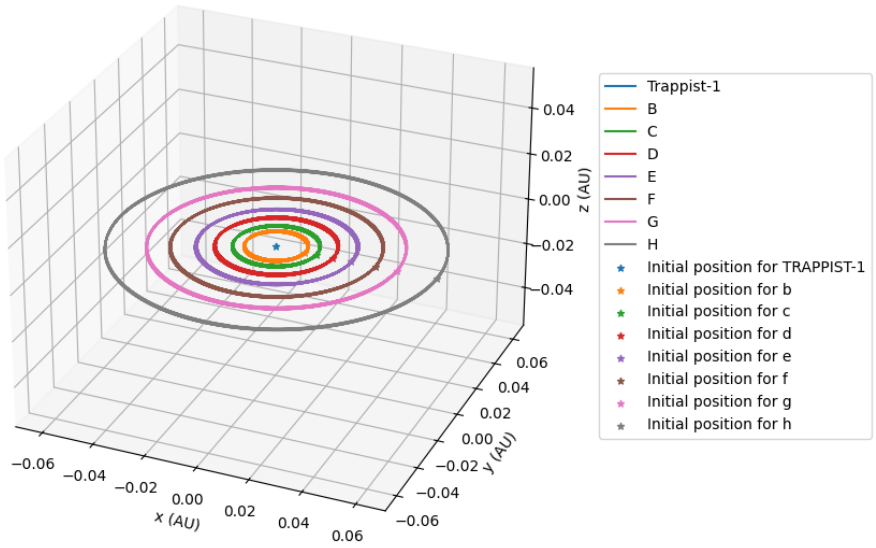
Figure 4: Numeric solution to Burrau's problem.

The change in tolerance and time steps decreased the energy difference and thus uncertainty. In part b) the trajectories are indistinguishable, yet they effectively illustrate the separation of the two heavy bodies as an independent binary system, distinct from the lighter rogue star, positioned away from the positions. Every movement in our simulation aligns with the reference paper, as demonstrated in the provided video.

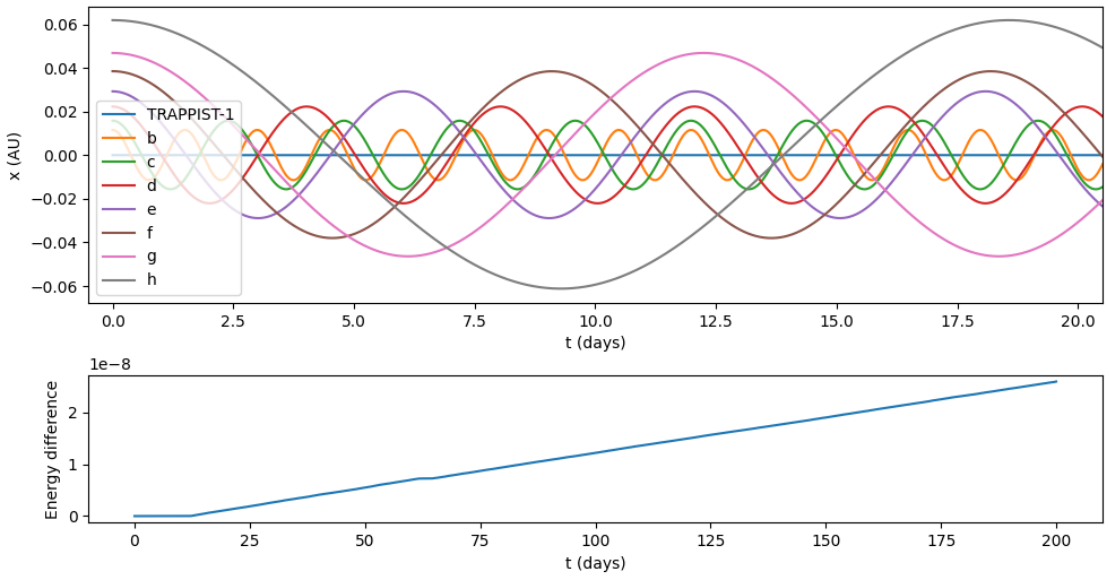
3 Novel implementations

3.1 TRAPPIST-1 planetary system

This planetary system, located approximately 40 light-years away from Earth, consists of one star and seven planets, all situated within the habitable zone. The star's luminosity is $5.53 \cdot 10^{-4} L\odot$, indicating that these planets are much closer to their star than Earth is to the Sun (Eric Agol et al 2021 Planet. Sci. J. 2 1). Due to their extremely low eccentricity, their orbital velocity can be derived as $\frac{2\pi major\ axis}{orbital\ period}$, and calculated from the data in the referenced paper.



(a) 3D plot



(b) X position and ΔE against time

Figure 5: TRAPPIST-1 Planetary system evolution.

Despite the proximity of the planets to each other and their similarities, we can observe a stable evolution over large time frames. This stability becomes evident when considering time scales on the order of the orbital periods of the bodies, a characteristic observed in most real-world systems that have had millennia to stabilize.

3.2 Barnes–Hut simulation

3.2.1 Oct and quad-trees

To address the N-body problem for a galactic number of bodies, we can explore alternative approximations that reduce computations in each time step. One such approach is proposed by Barnes and Hut (Barnes, J., Hut, P.(1986)). This algorithm divides space into an oct-tree (3D quad tree), creating consecutive octants, each capable of accommodating only one planet. If two planets happen to occupy the same octant, it is further subdivided. The algorithm then stores the total mass and center of mass for each subdivision, from the entire space down to smaller ones. This allows us to approximate gravity from distant objects using their center of mass, provided they are within a far enough node. **Refer to the animation 'Figure6.mp4'** for a visual representation.

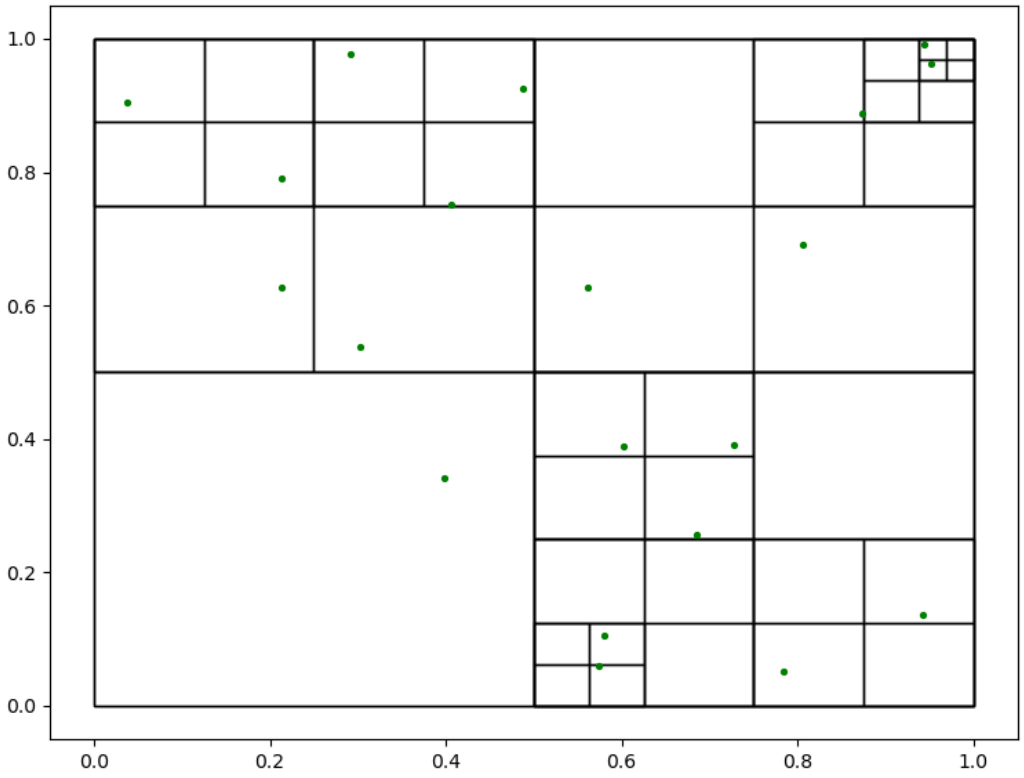


Figure 6: Quad tree visualisation for 20 bodies placed randomly from 0 to 1 in both axis.

3.2.2 Test against solar system

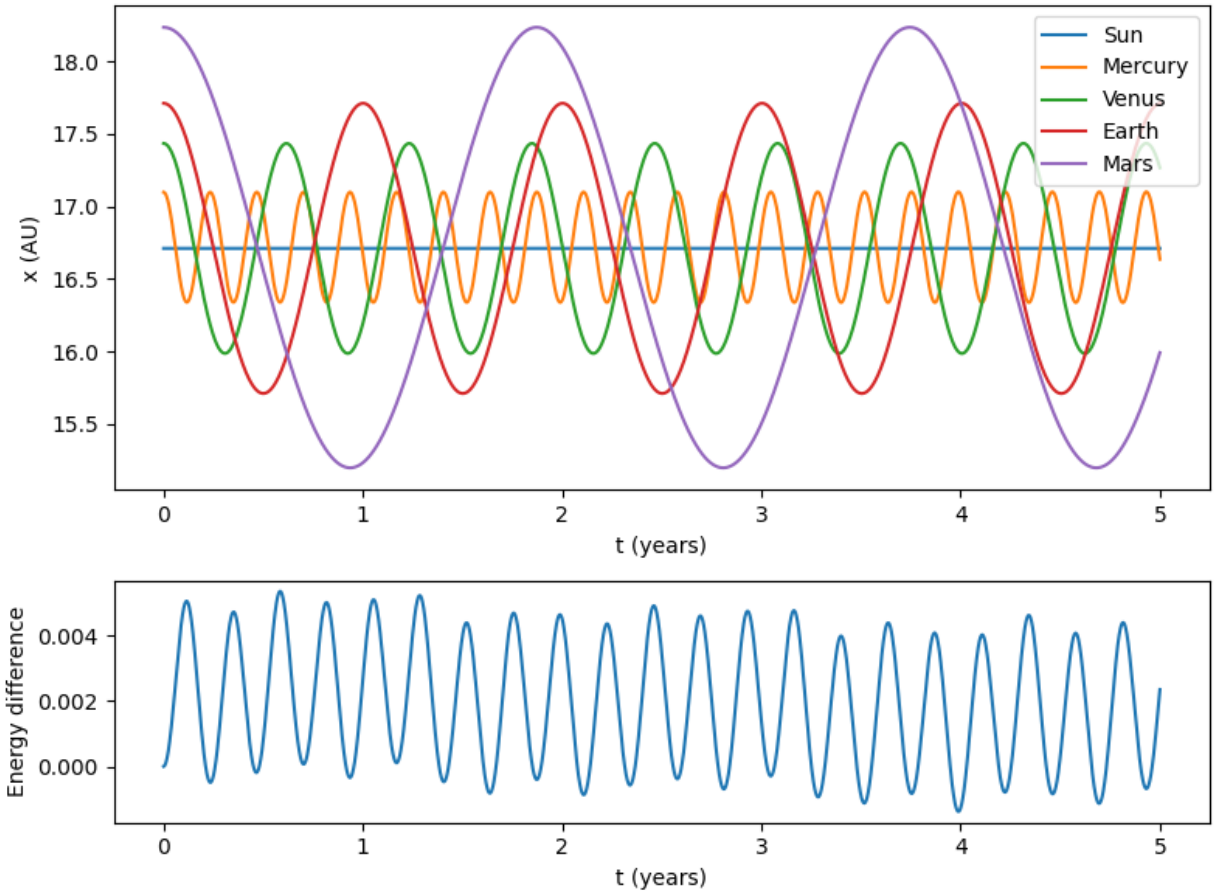


Figure 7: The 4 inner solar system planets orbiting the Sun over a span of 5 years with Barnes-Hut method.

We obtain positions that are extremely similar to those in the same situation as Figure 3. However, the order of magnitude of the energy difference is much greater. While previously it was around 10^{-6}, now it is on the order of 10^{-3}. Despite this difference, the trade-off, as explained earlier, is a much faster computation speed, especially as we increase the number of bodies.

3.3 Barnes–Hut simulation in 3D.

Now, we can develop programs to generate real oct-trees in 3D instead of 2D quad-trees. The next figure demonstrates the formation of an oct-tree for N=100. You can also refer to the animation **'Figure8.mp4'** for a visual representation.

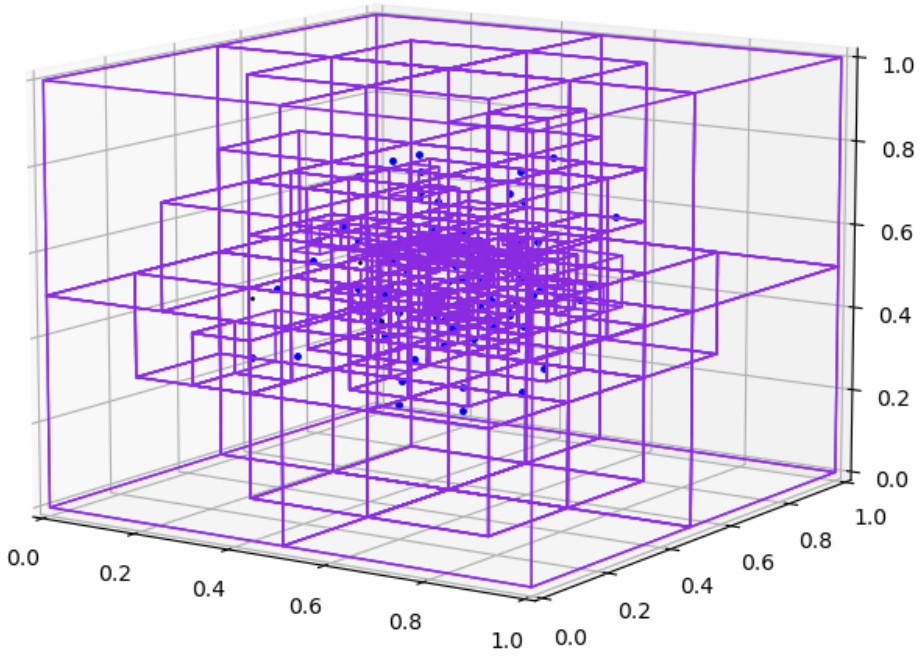


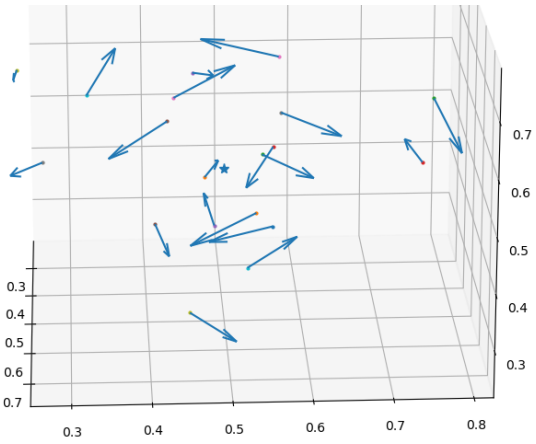
Figure 8: Oct tree visualisation for 100 bodies placed randomly from 0 to 1 in 3D.

For this large number of bodies, it's challenging to appreciate this accurately without referring to the animation. However, this image effectively illustrates the achieved density of bodies towards the center, with sparser distribution as we approach the limits of the simulation space

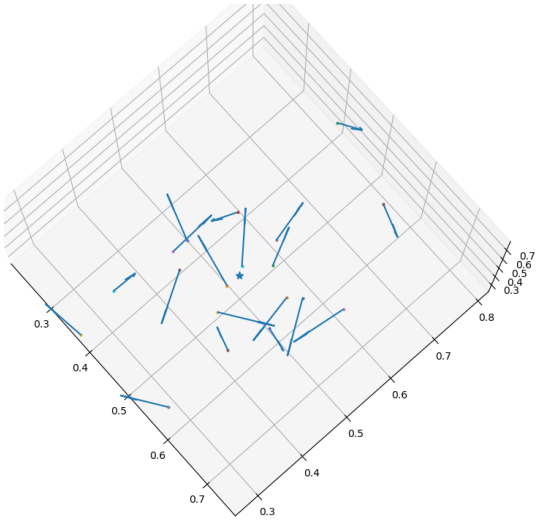
3.4 Large N simulation constraints.

To generalize this method, we introduce a threshold distance where two bodies in an extremely close encounter, compared to their expected radii, 'fuse' into one, conserving mass and momentum. We must also establish limits for the simulated volume and discard ejected bodies, even though this may introduce unexpected changes to energy. The physical simulation of the forces' effects on body movement should remain accurate, as demonstrated in previous examples, despite these deviations and uncertainties.

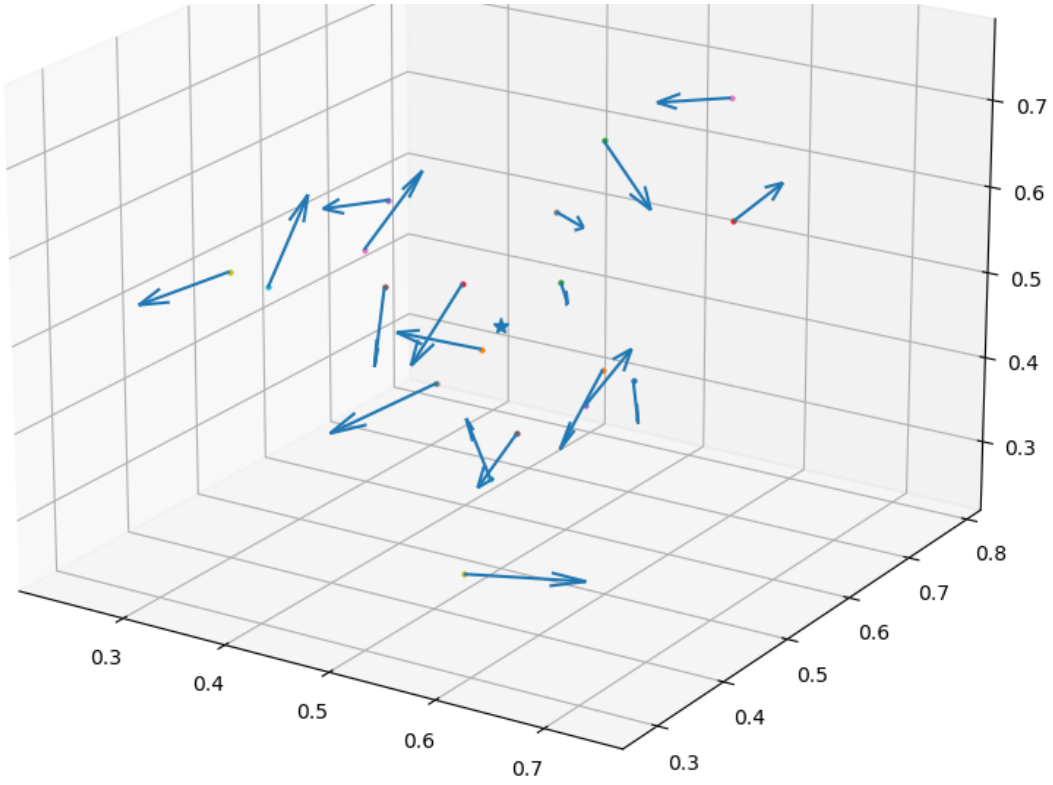
To place random objects, we create a distribution around the center, which could represent a galactic system. A massive static object serves as the main attractor for the rest of the bodies. The distribution of bodies follows the sech function. All these bodies start with velocities that have no component towards or away from the center, aiming to avoid loss through shoot-offs. The demonstration below illustrates the starting positions and velocities as described for a limited universe.



(a) Side view of space.



(b) Top view of space.



(c) X position and ΔE against time

Figure 9: Front view of space.

3.5 Large N simulation.

While the computing power of the systems available to me is insufficient for N orders of tens, even with the optimized Barnes-Hut algorithm, I present a small simulation featuring seven 'randomly' initialized bodies, adhering to the rules explained above.

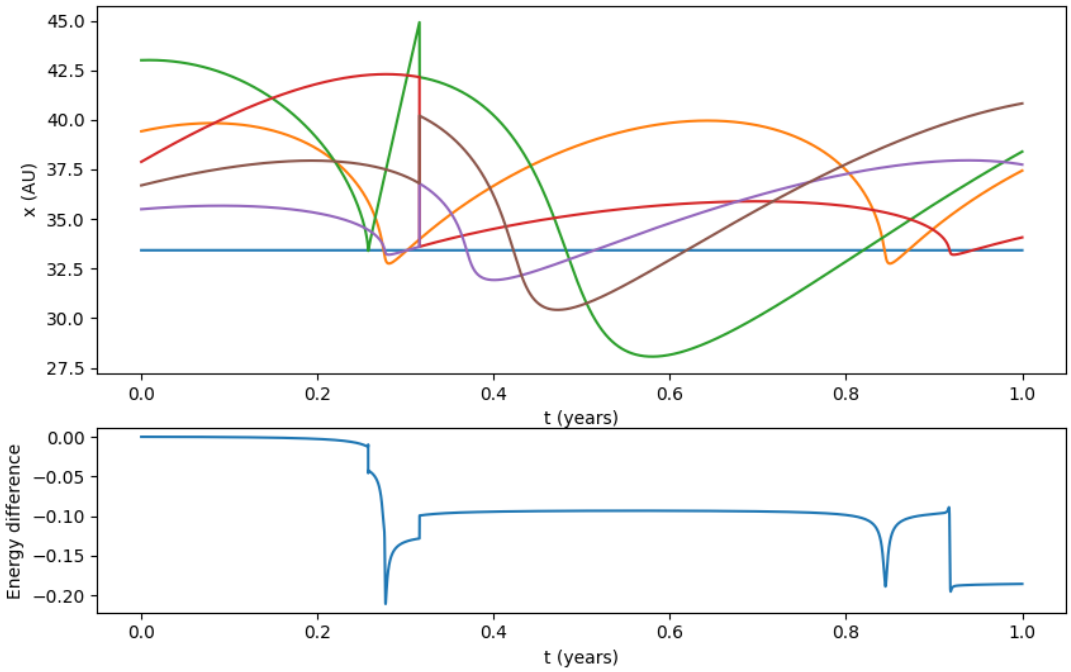


Figure 10: 7 randomly placed bodies with Barnes-Hut. Central body mass around sun mass ($1.989 \cdot 10^{33}$), other around solar system planet's masses ($10^{23} - 10^{27}$)

As observed, the energy loss is more significant, but it remains within acceptable limits. This occurs because, while this method is faster for large N, the compromise in accuracy is not justified for a small sample. Additionally, the ODE method employed is a second-order Runge-Kutta, less accurate than built-in methods. However, it appears to be the only viable option in Python for this algorithm without introducing more extensive mathematical and programming features or resorting to stricter programming languages, which would exceed the scope of the project.

The program effectively handles bodies escaping the model universe. In a specific simulation, a body exited the universe around $t = 0.61 years$ by surpassing the limits on the x-axis (set at 10^{12}). This information is printed by the program, and the data for this body is removed from the plot (leaving only six visible lines). Notably, the blue line corresponds to the star at the center of the model universe.

While this algorithm may take longer to run for smaller N values compared to the 'purest' one presented in the initial figures, it can still produce reasonable results. Its scalability, with a time complexity of $O(N * log(N))$ as opposed to the traditional $O(N^2)$, makes it remarkably faster for galactic-sized samples.

4 How to run the code and file structure

4.1 Code

In the included files, the code is separated by folders for each figure or groups of them. These are typed in the names of the folders. For figures 1-5, the code is run from the implementation files denoted by 'impl'. On the other hand, the Barnes-Hut files are all in the same folder but have similar naming scheme.

4.2 Animations and images

These are intuitively included in properly-named folders.

5 Not mentioned technical features

It would be relevant to mention some features included in code but that cannot be seen in the figures and were not mentioned.

- Distances are only calculated once for every pair of bodies. This is very useful given square roots are time consuming operations.
- The user has the option to have the program create a .csv file with the results of a simulation, and then the load of this files to skip the ODE and import the result from them.
- While all time steps are taken into account and solved, only some are used to calculate energies, plotted and saved to files, thus saving an enormous amount of memory, and speeding up plots drastically, with no compromise because the resolution of the plots is still as good as that of the screens.
- Given solving the ODEs and writing to/reading from files takes a long time, the progress of this processes is displayed in a progress bar.
- The addition of dynamic time steps (increment of them in close encounters) was considered, but the chosen ODE solvers implement it.
- The implementation of multiprocessing was considered, tried, and can still be used in the Barnes-Hut programs, but it is considerably slower because the instance where it can be used is short and managing their creation and destruction takes more time than they save.