Final project of FYS4150
N-body simulation of an open galactic cluster

Candidate 71

Fall 2014

Abstract

In this project, a code for simulating the solar system was developed, integrating Newton's equations of motion using two different methods: Verlet and 4th order Runge-Kutta. The two solvers were used to simulate a two-body system consisting of a stationary Sun and the Earth orbiting around it, a three-body system where the planet Jupiter was added to see its influence on Earth's orbit, and a simulation including all eight planets and Pluto was performed. An algorithm with an adaptive step size was made using the Runge-Kutta-Fehlberg method, and this algorithm was used to simulate a cold collapse of a galactic cluster.

1 Introduction

The goal in this project is to develop a code that can perform simulations of an open cluster using Newtonian gravity. The velocity Verlet method and the 4th order Runge-Kutta method will be tested for stability by simulating our solar system, and then a method for using adaptive step sizes will be developed.

In this project, the stability of the methods is important because in large systems, the statistical properties of the system are more interesting than the individual motion of each particle.

By making an algorithm with an adaptive step size, it is possible to decrease the computation time of the simulation without reducing its accuracy, by letting the algorithm increase the step size when the forces between the particles are low and decreasing the step size when large step sizes would lead to errors.

2 Theory

The physical part of this problem is tackled best by first considering a hypothetical solar system composed of only the Sun, with a fixed position at the origin, and Earth orbiting it. By assuming that Earth's orbit around the Sun is circular [1] , this special case becomes a two-body problem with a simple analytical solution.

2.1 Gravity

The only force with non-negligible influence in the solar system is gravity. Newton's law of gravitation is given by a force F_G between two objects with masses M_1 and M_2:

$$F_G = \frac{GM_1M_2}{r^2}, \tag{1}$$

where the gravitational constant is G and r is the distance between the two objects.

In the two-body case, it is assumed that the Sun has a mass which is much larger than that of Earth, and the motion of the Sun can therefore safely be neglected. It is also assumed that the orbit of Earth around the Sun is co-planar, and this plane is defined to be the xy-plane. Using Newton's second law of motion we get the following equations

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_{\text{Earth}}},$$

$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_{\text{Earth}}},$$

and

$$\frac{d^2z}{dt^2} = \frac{F_{G,z}}{M_{\text{Earth}}},$$

where $F_{G,x}$, $F_{G,y}$ and $F_{G,z}$ are the x, y and z components of the gravitational force.

The above second-order ordinary differential equations can be written as a set of coupled first order differential equations by realizing that

$$\frac{d^2x}{dt^2} = a_x = \frac{dv_x}{dt},$$

which leads to

$$\frac{dx}{dt} = v_x \tag{2}$$

and

$$\frac{dv_x}{dt} = \frac{F_{G,x}}{M_{\text{Earth}}}, \tag{3}$$

with similar expressions for the y and z components.

In the rest of this section, the absolute values for force, acceleration and distance are used unless specified otherwise. The component-wise direction of these is acquired by multiplying the particular equation by $\frac{r_x}{r}$, where r_x is the distance along the x-axis between the bodies. This is done similarly for y and z.

[1] This assumption is quite accurate, as the orbital eccentricity (the amount of which the orbit deviates from a perfect circle) is only about 0,017 radians. [1]

2.1.1 Simplifying the acceleration calculations

Setting the mass of each body as the mass relative to the mass of the Sun, $M_\odot$, i.e. $M_i = m_i M_\odot$ allows for equation (1) to be written as

$$F_G = \frac{GM_\odot m_1 M_\odot m_2}{r^2}$$

which gives the following expression for the acceleration on body 1 caused by body 2:

$$a_1 = \frac{F_G}{M_\odot m_1} = GM_\odot \frac{m_2}{r^2} \tag{4}$$

where m_2 is the relative mass of body(2) and r is the distance between the bodies. Equation 4 is easily expanded to multiple bodies, as each body causes a separate force on body 1. However, when solving this for more than two bodies, it becomes necessary to include direction. For N bodies, this can be done by using vectors,

$$\vec{a}_1 = \frac{\Sigma \vec{F}_G}{M_\odot m_1} = GM_\odot \left(\frac{m_2}{|\vec{r}_2|^3}\vec{r}_2 + \frac{m_3}{|\vec{r}_3|^3}\vec{r}_3 + \ldots + \frac{m_N}{|\vec{r}_N|^3}\vec{r}_N \right) \tag{5}$$

where m_i is the mass of body i in solar masses, and $\vec{r_i}$ is the vector pointing from body 1 to body i, with length in AU. Alternatively, it can be done component-wise,

$$a_{1,x} = \frac{\Sigma F_{G,x}}{M_\odot m_1} = GM_\odot \left(\frac{m_2 r_{2,x}}{r_2^3} + \frac{m_3 r_{3,x}}{r_3^3} + \ldots + \frac{m_N r_{N,x}}{r_N^3} \right) \tag{6}$$

where $a_{1,x}$ is the x-component of the acceleration on body 1, r_i is the absolute distance between body 1 and body i, and $r_{i,x}$ is the distance along the x-axis between body 1 and body i, defined as positive when the x-component of the position vector for body i is larger (more positive) than the x-component for body 1. The component-wise calculations will have to be done for the y- and z-components as well.

2.1.2 The gravitational constant

In order to avoid calculations comparing very large and very small numbers, the various variables (length, mass and time) is presented in other units than their respective SI units. As the solar system and the galactic cluster uses different units for these variables, the gravitational constant, G, will have different values for each system.

Solar system

For a solar system, Equation (1) can be vastly simplified by applying the formula for the centripetal force of a body in a circular orbit

$$F_c = ma_c = \frac{mv^2}{r}.$$

For Earth, this becomes

$$F_G = \frac{M_{\text{Earth}} v^2}{r} = \frac{GM_\odot M_{\text{Earth}}}{r^2},$$

where v is the speed of the Earth in a circular orbit with radius r around the Sun and $M_\odot$ is the mass of the sun. It's easy to see from this that

$$v^2 r = GM_\odot. \tag{7}$$

By using years instead of seconds and distance in astronomical units[2] (AU), we get $v = 2\pi$ AU/yr (one full revolution of the Earth around the Sun per year) which in turn gives

$$GM_\odot = 4\pi^2 \text{AU}^3/\text{yr}^2. \tag{8}$$

[2] 1 AU = 1.5×10^{11} m, the average distance between the Sun and Earth

Galactic cluster

For the galactic cluster, the gravitational constant is defined differently. For an near-infinite number of particles (where each particle is one or several solar systems), $N \to \infty$, with a constant mass density ρ_0, we get a continuous fluid. In this case, the system collapses into a singularity at a finite time given by

$$\tau_{crunch} = \sqrt{\frac{3\pi}{32G\rho_0}}. \tag{9}$$

Here, ρ_0 is the mass density defined by

$$\rho_0 = \frac{m}{V} = \frac{N\mu}{\frac{4}{3}\pi R_0^3},$$

where N is the number of particles, μ is the average mass of each particle and R_0 is the radius of a sphere in which the particles are uniformly (randomly) distributed. By substituting the density in Equation 9 and solving for G, we get

$$G = \frac{4\pi^2 R_0^3}{32{\tau_{crunch}}^2 N\mu},$$

which in our case, with $\mu = 10M_\odot$ and $R_0 = 20ly$ (light years), gives

$$G = \frac{100\pi^2}{N}\frac{ly^3}{{\tau_{crunch}}^2 M_\odot},$$

or

$$GM_\odot = \frac{100\pi^2}{N}\frac{ly^3}{{\tau_{crunch}}^2}.$$

Here, the units are light years for length and τ_{crunch} for time.

3 Approach

A class was made for the celestial bodies, containing mass stored as a double and position, velocity and acceleration stored as three-dimensional vectors using a vector class called $vec3$ made by Anders Hafreager. During the simulation of the solar system, the planets and moons were assumed to be moving in the xy-plane, and the z-component of these vectors were set to zero.

Two different solvers for the differential equations were written, one using the velocity Verlet method and one using the RK4 method. Both of these methods used the std::vector function to make a vector of vectors, and so I had to implement operators for this class.

3.1 Solar system

In order to test the effectiveness of my developed solvers, the simulation was run for a hypothetical solar system composed of only the Sun, with a fixed position at the origin, and Earth orbiting it. Masses, starting distance from the Sun, and mean radius for each body used in calculations is listed in Table 1. For different values of δt, the sum of kinetic and potential energy was calculated for each time step.

After testing the two solvers for the two-body system, and stabilizing them with regards to kinetic and potential energy, the escape velocity for a planet beginning at a distance of 1 AU from the sun was found using trial and error, and compared to an analytically calculated escape velocity.

Then, the system was expanded to three bodies, adding Jupiter. After seeing Jupiter's influence on Earth's orbit, Jupiter's mass was increased by a factor of 10 and 1000 to see its influence increase.

A real multi-body calculation with all bodies in motion was performed, using both the velocity Verlet and the RK4 solver, simulating all eight planets, the Moon and Pluto.

Table 1: Absolute mass, mass relative to the Sun's mass and average distance from the Sun of the celestial bodies of the solar system [2, 3]

Body	Mass (kg)	Relative mass	Avg. distance (AU)
Sun	2×10^{30}	1	0
Mercury	3.3×10^{23}	1.65×10^{-7}	0.39
Venus	4.9×10^{24}	2.45×10^{-6}	0.72
Earth	6.0×10^{24}	3.00×10^{-6}	1.00
Mars	6.4×10^{23}	3.2×10^{-7}	1.52
Jupiter	1.9×10^{27}	9.5×10^{-4}	5.20
Saturn	5.7×10^{26}	2.85×10^{-4}	9.54
Uranus	8.7×10^{25}	4.35×10^{-5}	19.19
Neptune	1.02×10^{26}	5.1×10^{-5}	30.07
Pluto	1.31×10^{22}	6.55×10^{-9}	39.48

3.2 Adaptive time step algorithm

For the adaptive time step algorithm, I first attempted to implement different time steps for different objects using the RK4 solver. However, as this solver depends on contributions from every object and using these in every halfstep computation, it quickly became clear that attempting this would be nearly impossible, it would vastly increase code complexity, and would almost certainly be slower than just calculating the entire system at the smallest time step.

As it is often used in simulation of celestial objects, I used the Runge-Kutta-Fehlberg method[4] (RKF45 hereafter), which compares a 4th order and a 5th order Runge-Kutta calculation, and adjusts the time step of the entire system according to the difference between the results of the calculations. However, I did a small adjustment to the method to decrease time used: Instead of using the weighting system used in the RKF45 method to calculate both the 4th order and the 5th order approximation, I used the weighting system used for the regular RK4 method to calculate the 4th order approximation.

As the two approximations were vectors of vectors instead of simple values, I could not compare the two results the usual way. Instead, I used this formula:

$$\Delta_{4th-5th} = \Sigma \frac{(|RK4_{t+dt,i} - RK5_{t+dt,i}|)}{(|RK5_{t+dt,i}|)}$$

which gives the total relative difference between the two approximations. In my simulations, the maximum allowed difference was set to 10^{-6}, while the minimum allowed difference was set four orders of magnitude lower. In addition, a maximum and a minimum absolute value for the time step were set, to avoid having the step size growing or shrinking uncontrollably.

3.3 Galactic cluster

A simple model of an open cluster was built, using a random distribution of particles within a sphere, with masses randomly distributed by a Gaussian distribution around ten solar masses with a standard deviation of one solar mass. For this part of the project, light years (ly) and solar masses (M_{odot}) were used as units of length and mass, and time was measured in τ_{crunch} (See section 2.1.2). As this was a "cold collapse" simulation, all particles started at rest.

The simulation was run for a few τ_{crunch}, with and the system was animated using the scatter3 function in MATLAB. The kinetic and potential energies of the system was tracked and evaluated as well.

A smoothing function was introduced to reduce numerical instability from particles coming too close to each other. The Newtonian force was modified to become finite at short ranges

$$F_{mod} = -\frac{GM_1 M_2}{r^2 + \epsilon^2}.$$

The system was then simulated again to see the influence of this correction.

4 Results & Conclusions

4.1 Solar system

Circular orbit is given for an initial velocity $v_0 = \frac{2\pi}{\sqrt{r}}$, as found when solving equation(7) for v. This also proved correct in the calculations.

Analytically, the escape velocity is found for a planet when the potential energy of the gravitational field is equal to the kinetic energy of the planet, i.e. when

$$\frac{1}{2} m_{\text{planet}} v_e^2 = \frac{G M_\odot m_{\text{planet}}}{r},$$

where v_e is the escape velocity. This gives

$$v_e = \sqrt{\frac{2 G M_\odot}{r}}.$$

The experimental results for the escape velocity of a planet at a distance of 1 AU from the sun are shown in Figure 1. This shows that the escape velocity is about $v_e = 8.9\,\mathrm{AU/yr}$, which is consistent with the analytical result, $v_e = \sqrt{2 \cdot 4\pi^2} \approx 8.886\,\mathrm{AU/yr}$.

When adding Jupiter at its normal mass, there was nearly no difference to Earth's orbit (see Figure 2). When Jupiter's mass was increased tenfold, a small difference could be seen as a thicker line for Earth's orbit in Figure 3. When Jupiter's mass was increased to approximately the mass of the Sun, Earth was pulled almost into the Sun, and then catapulted into outer space, as seen in Figure 4.

The simulation of all celestial bodies was completed successfully for both solvers, as seen in Figures 5, 6, 7 and 8. Here, all the celestial bodies moved smoothly around the center of mass, causing a slight broadening of their orbits.

4.2 Adaptive time step algorithm

The adaptive time step algorithm used was a modified RKF45. The modification was using the regular RK4 method as the 4th order approximation instead of the approximation usually used (see [4] for weighting system), as the approximation given by the RKF45 method had much larger errors, causing the method to reduce the step size by up to several orders of magnitude compared to when the regular RK4 method was used inside the RKF45 algorithm. This led to a moderate increase in calculations per time step (loosely calculated about $40\% - 50\%$), but as the step size could be much higher, the total calculation time was decreased considerably.

The RKF45 method was less effective than the locked step size methods for simulating the solar system, or any circular, non-changing orbits for that matter. The reason for this is that the adaptive step size method finds an acceptable step size for the system, and then keeps this step size for almost the whole simulation, only adjusting it once or twice if the initial adjustments were too large. Other than that, it keeps the same step size, but uses more than twice as many calculations per time step than the locked step size methods. In these cases, it's better to use the locked step size methods, and merely set the time step a bit lower. However, for an open cluster, where the distance (and thereby the acceleration) between the particles varies, the RKF45 method increases the effectiveness of the calculations.

4.3 Galactic cluster

The two solvers, 4th order Runge-Kutta and velocity Verlet, were tested for the solar system for a period of 1000 years. As seen in Figures 9 and 10, both methods are stable over very long times

A cluster with $N = 100$ particles was simulated for $t = 3\tau_{crunch}$, shown for $t = 0.1\tau_{crunch}$ in Figure 11. The system does not collapse into a singularity in this model simply because the program sees every object as a point particle instead of a physical object with a real size. The singularity collapse could have been implemented by an extra function inside the time loop, written here as pseudocode:

```
// Collision function
inside time loop:

Integrating system and advancing positions and time
for (every particle){
  measure distance to other particles
  if (distance between two particles < set value){
    mass1 = mass1 + mass2
    momentum1 = momentum1 +momentum2
    delete particle 2
  }
}
```

As seen in Figure 12, the energy in the cluster is not conserved. The reason for this is that the particles get too close to each other, and thus gain an incredibly high velocity in opposite directions. This can be observed in the separate plots for potential and kinetic energy (Figures 13 and 14). Every time two particles come too close to each other and are ejected from the cluster, it is indicated by a downward spike in the potential energy as the distance between the particles becomes very small, and a permanent increase in kinetic energy as the two particles greatly increase their velocity and quickly gain enough distance from all other particles to effectively reduce the forces from gravity to an insignificant amount. However, after about 2 τ_{crunch}, the system reaches a semi-equilibrium, where a small number of the particles are gathered in a cluster around the origo, as shown in Figure 15. A few particles are still ejected after the equilibrium is reached, but the amount is much lower than before equilibrium.

In my project, the smoothing parameter was set to $\epsilon^2 = 10^{-6}$, as this corresponds to $\epsilon \approx 60$AU, a bit larger than the size of our solar system. If two solar systems were to come this close, there is a high probability that they would collide, so this size of the parameter should be reasonable.

When the smoothing function was introduced, the energy of the system became much more stable for the first few τ_{crunch}, as Figure 16 shows. However, over time the cluster becomes more compact, and the particles come closer to each other, increasing the potential energy drastically. This causes the large downward spikes and noise in the plot at $t > 3.5\tau_{crunch}$, as the particles start to overlap and lose their velocity.

5 Summary and possibilities for further work

The velocity Verlet method and the 4th order Runge-Kutta method were here used to solve the Newtonian gravity equations and simulate both our solar system an a cold collapse of an open galactic cluster. The stability of both methods were evaluated, and the Runge-Kutta-Fehlberg method for adaptive time steps was implemented to increase effectivity of the simulation.

The most interesting avenue for further work would be to implement the collision function, which probably would cause most of the system to merge into one particle at $t \approx \tau_{crunch}$. Another possibility would be to increase the size of the system to a number of particles $N = 1000 or 10000$.

References

[1] Standish, E. Myles; Williams, James C., *Orbital Ephemerides of the Sun, Moon, and Planets*

[2] NASA, *Planets and Pluto: Physical Characteristics*

[3] Standish, E, *Keplerian Elements for Approximate Positions of the Major Planets*

[4] Mathews, John H; Fink, Kurtis K., *Numerical Methods Using Matlab*, 4th Edition, 2004: Sec. 9.5, *Runge-Kutta Methods*

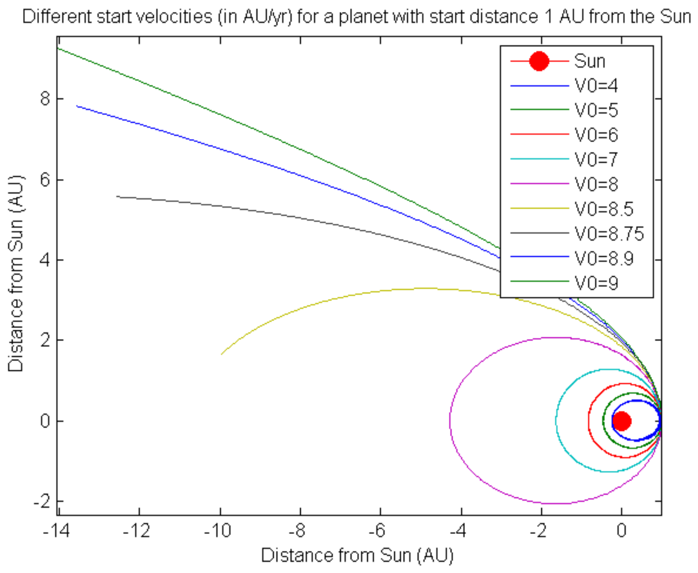

Figure 1: Different initial velocities for a planet starting at a distance of 1 AU from the sun. Each path was calculated for a time period of 5 years.

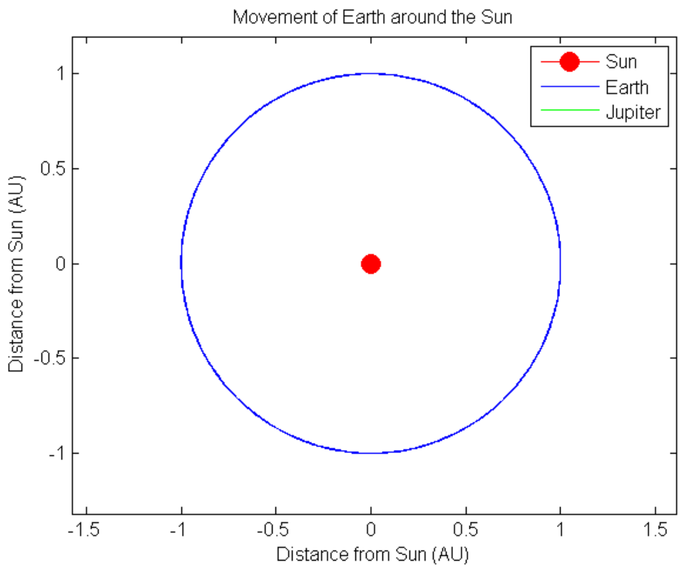

Figure 2: Movement of Earth with Jupiter added. Jupiter's orbit has been cropped from the picture to allow an increased view of Earth's orbit.

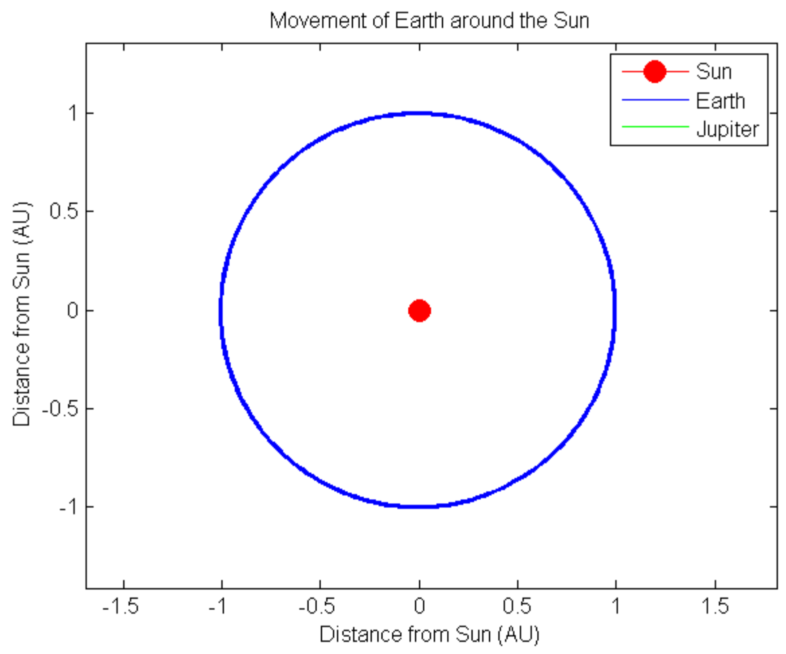

Figure 3: Earth's orbit with Jupiter added, and Jupiter's mass increased by a factor of 10. Jupiter's orbit has been cropped from the picture to allow an increased view of Earth's orbit.

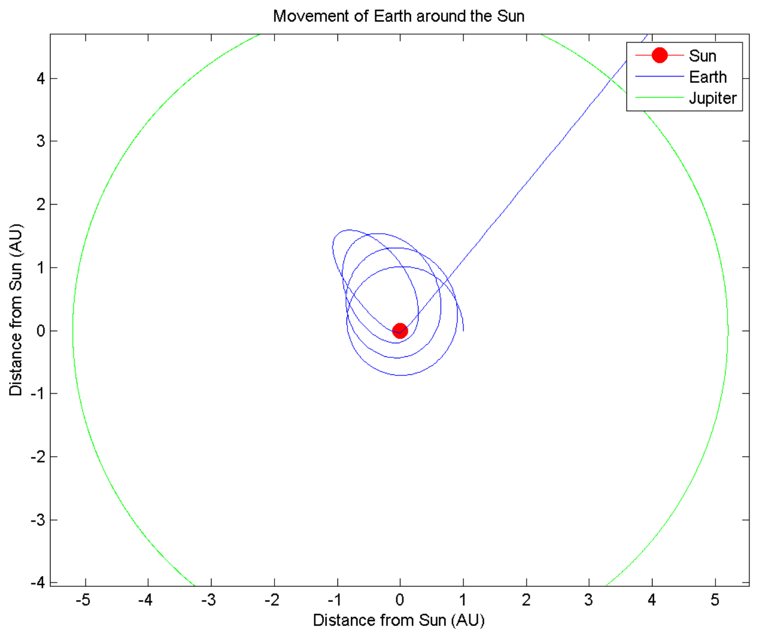

Figure 4: Earth's orbit with Jupiter added, and Jupiter's mass increased by a factor of 1000. The Earth is now pulled completely out of its orbit, and wanders off into the sun.

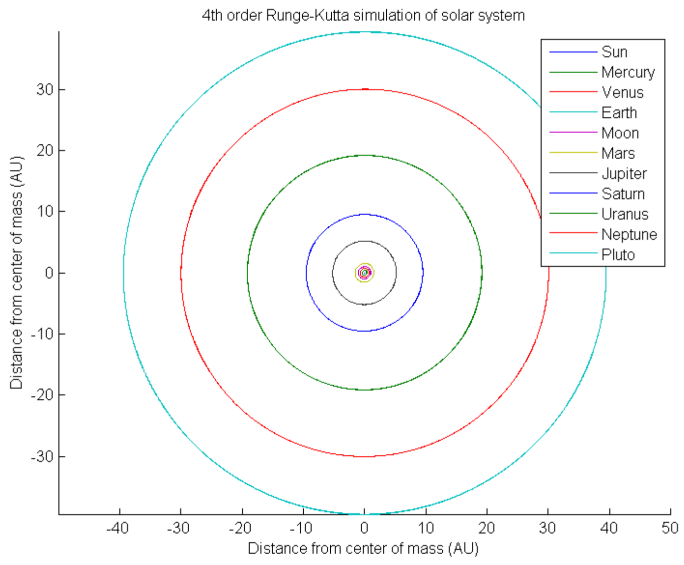

Figure 5: Plot of the simulation of all planets and Pluto using the RK4 solver, at a period of 1000 years.

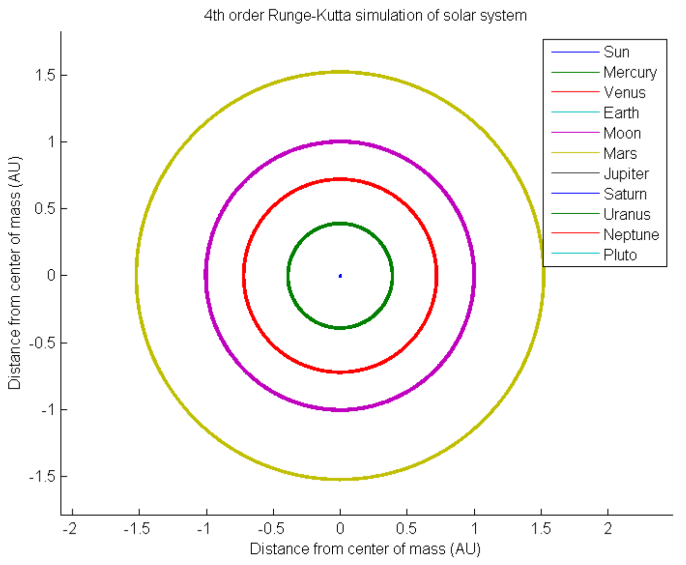

Figure 6: Cut-out of the inner 4 planets from figure 5.

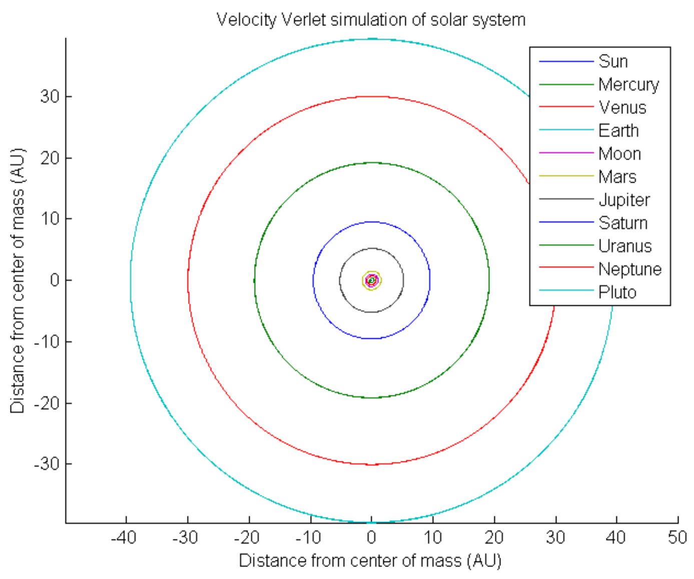

Figure 7: Plot of the simulation of all planets and Pluto using the velocity Verlet solver, at a period of 1000 years.

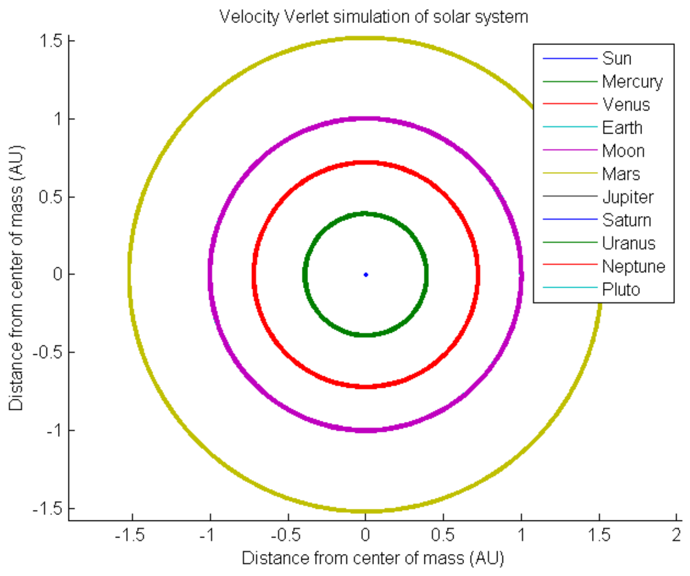

Figure 8: Cut-out of the inner 4 planets from figure 7.

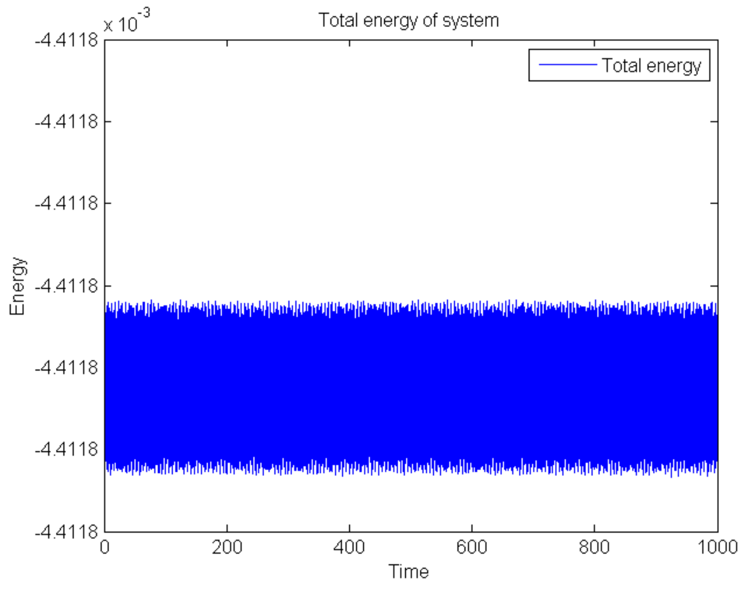

Figure 9: Sum of kinetic and potential energy over a 1000-year period for a simulation of the solar system using the velocity Verlet method.

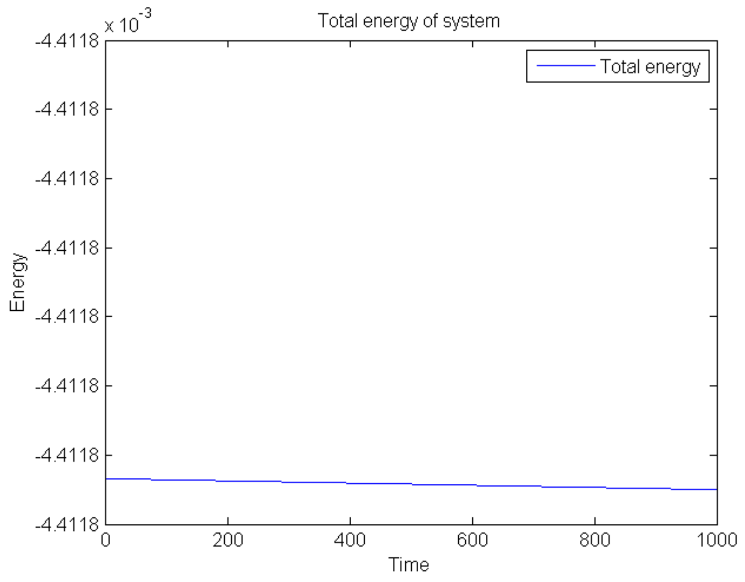

Figure 10: Sum of kinetic and potential energy over a 1000-year period for a simulation of the solar system using the 4th order Runge-Kutta method.

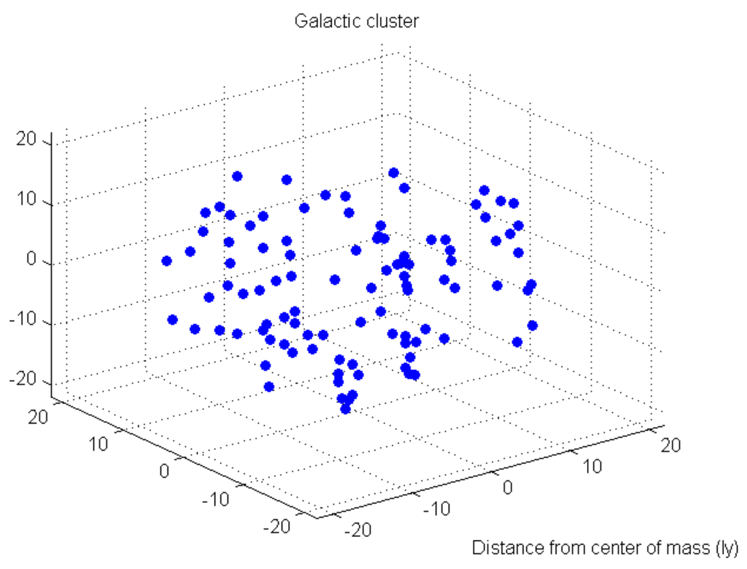

Figure 11: The positions of the particles in a cluster with $N = 100$ at $t = 0.1\,\tau_{crunch}$

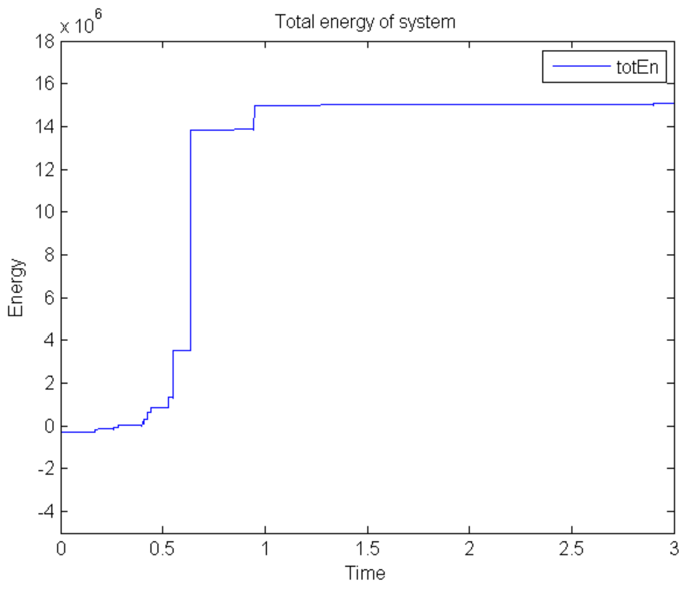

Figure 12: Plot of the sum of kinetic and potential energy for the galactic cluster with N=100

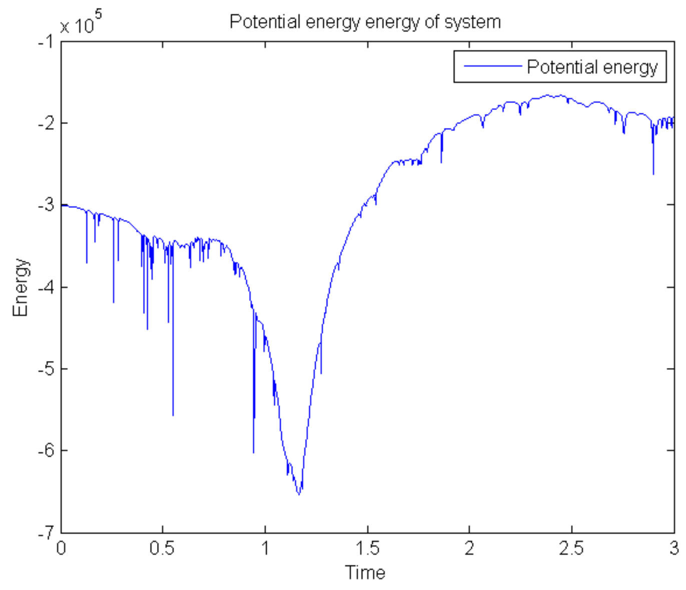

Figure 13: Plot of the potential energy for the galactic cluster with N=100

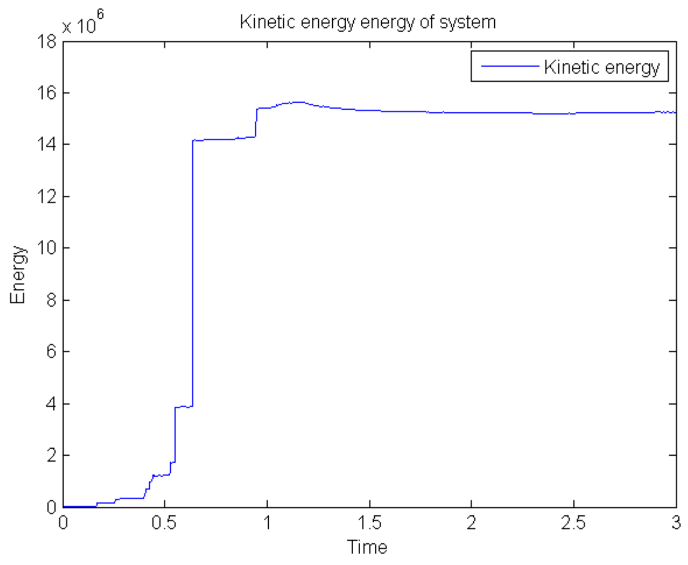

Figure 14: Plot of the kinetic energy for the galactic cluster with N=100

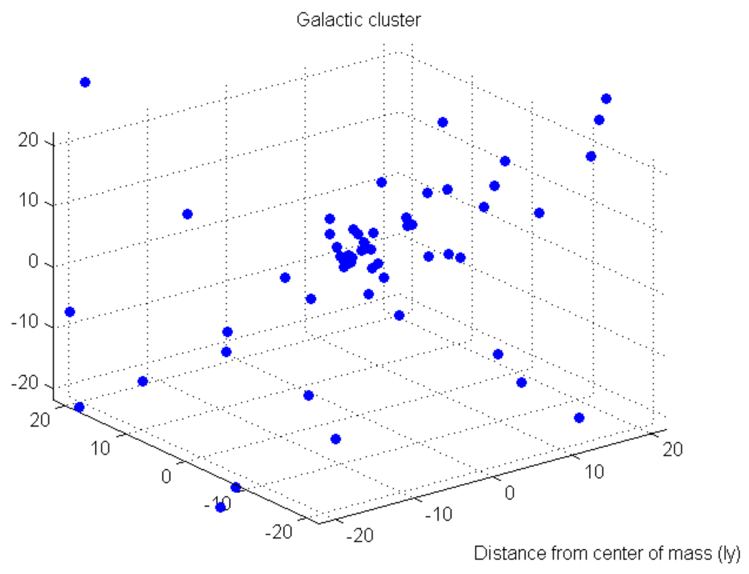

Figure 15: The positions of the particles in a cluster with $N = 100$ at $t = 2.2\,\tau_{crunch}$

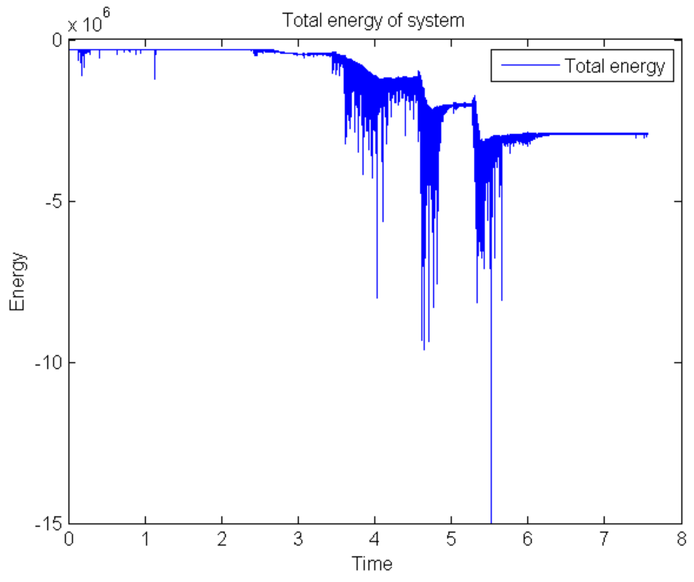

Figure 16: Plot of the sum of kinetic and potential energy for the galactic cluster with N=100 and a smoothing parameter $\epsilon^2 = 10^{-6}$