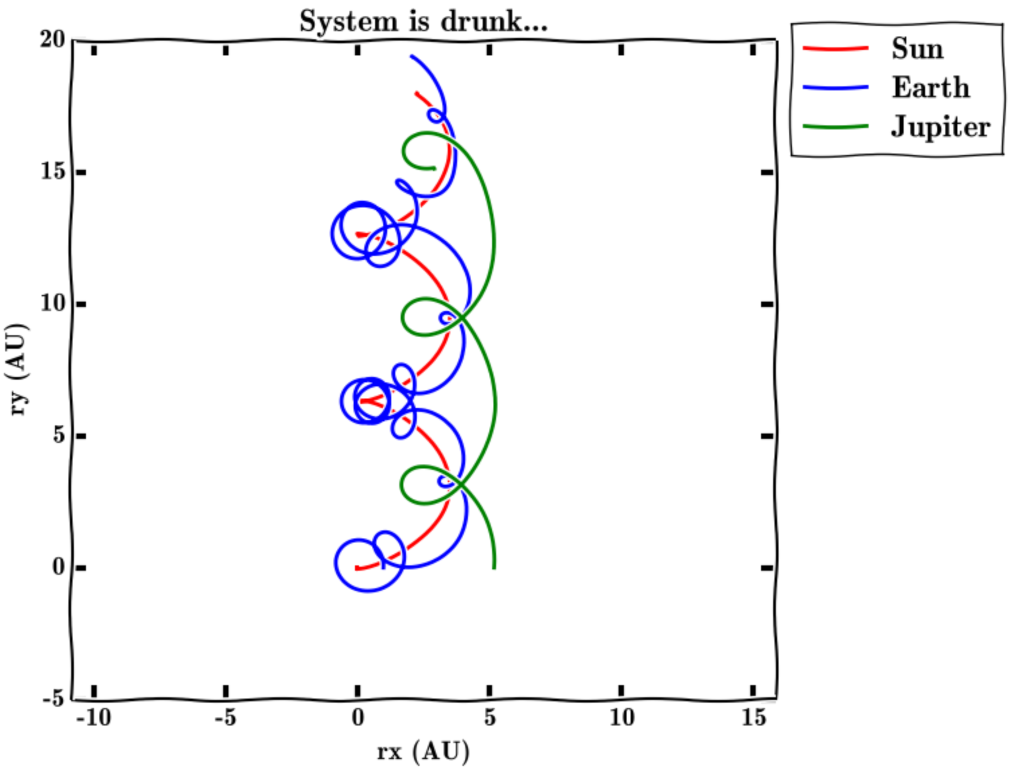
FYS3150 - Project 2

Alfred Alocias Mariadason

20.10.2014



Introduction

In this project we were tasked with building a model for the solar system, that is, develope a code for simulating the solar system. The physical law used was mainly newton's law for gravitation and the second law of motion(newtons second law). This was then solved as a second order differential equation which was then rewritten as a set of coupled first order differential equations. This set was then solved by discretizising the equations and setting up an algorithm.

The algorithm used for solving the differential equations was the fourth order Runge-Kutta method(also known as RK4) and Verlet method. Note that the Verlet method is specialized by the velocity verlet method, which assumes that the acceleration is only dependant on the position.

The programming language used for creating the celestial bodies and the solarsystem and implementing the above algorithms was C++, while the visualization(plots) is in python.

Link to programs used: `https://github.com/Oo1Insane1oO/project3`.

Theory

Setting up the equations

For setting up the equation we will here limit ourselves to a hypothetical solarsystem with just the Sun and Earth orbiting around it. The only force acting on the two bodies is given by Newton's gravitation law:

$$F_G = G\frac{M_{sun}M_{earth}}{r^2}$$

Here G is the graviational constant, the M's are respectivly the mass of the sun and earth and r is the distance between the two objects. Note that r is the distance between the center of the two bodies and not the distance from the surfaces. If we now use Newton's second law of motion we have that $\Sigma F = F_G = ma \Rightarrow a = \frac{F_G}{m}$, where m is the mass of the object the force is acting on(in our problem this is earth). With $a = \frac{d^2\vec{r}}{dt^2}$ we have the following second order differential equation:

$$\frac{d^2\vec{r}}{dt^2} = \frac{\vec{F}_G}{M_{earth}}$$

In order to solve this equation we can rewrite it as a set of coupled first order differential equation by using the fact that $\vec{v} = \frac{d\vec{r}}{dt}$. The set would then look like this:

$$\frac{d\vec{r}(t)}{dt} = \vec{v}(t)$$
$$\frac{d\vec{v}(t)}{dt} = \vec{a}(t,\vec{r}(t))$$

We want to simulate this on a computer, which means one can easily obtain errors in the floating point operations being conducted throughout the algorithm. Writing the equations in terms of dimensionless variables can greatly reduce these errors by fixing the constants to better suitable numbers. In our problem we can see that we have the gravitational constant(which is a small number) multiplied with the mass of the sun(a big number). As mentioned one can use dimensionless variables, but because the magnitude of the distances in the solarsystem is so large calculating the distance in AU(astronomical units) and time in years is a much better approach. Since the mass of the sun is so large compared to the planets orbiting it, scaling all masses with the mass of the sun is also a good idea.

In order to fix the gravitational constant and the sun mass we can use the fact that Earth's orbit around the sun is close to circular. Using the law of force for circular motion we have the following relation:

$$F_G = \frac{M_{Earth} v^2}{r} = G\frac{M_{Sun} M_{Earth}}{r^2}$$

We know that for a circular motion the velocity is given by $\vec{v} = \frac{2\pi r}{T}$, with the above equation we then have that:

$$v^2 r = \frac{4\pi^2 r^2}{T^2} r$$

If we measure r in AU and time T in years the constants can be fixed as:

$$v^2 r = G M_{Sun} = 4\pi^2 AU^3/yr^2$$

The final equation for force is then:

$$F_G = G\frac{M_{Sun} M_{Earth}}{r^2} = \frac{4\pi^2 M_{Earth}}{r^2}$$

Runge-Kutta-4

The Runge-Kutta methods are basically a collection of different methods for solving differential equations numberically. The order indicates the precision in respect to the number of iterations needed. The higher the orden, the better the relative precision is. The first order Runge-Kutta is just Eulers method, while second order is Euler midpoint. For the fourth order methods one first calulates the value on the tangent of the slope of our function(Eulers methods), we then use the midpoint method twice to calculate the values in the middle of the slope and then we add these to a fourth value by using Eulers again. Note that after fourth order one has to jump to sixth order to gain a higher level of precision and this behaviour repeats itself for every orden higher than fourth, this is why fourth order is usually preferred over fifth.

For our problem the function we want is given by a set of coupled first order differential equations(as shown above). We need to calculate both the position and the velocity. If we set $dt = h$ and $\frac{dv}{dt} = g(t, v)$ the algorithm according to the RK4 methods is:

$$
\begin{aligned}
K_1 &= hv_k \\
L_1 &= hg(t_k, r_k) \\
K_2 &= h(v_k + L_1/2) \\
L_2 &= hg(t_k + h/2, r_k + K_1/2) \\
K_3 &= h(v_k + L_2/2) \\
L_3 &= hg(t_k + h/2, r_k + K_2/2) \\
K_4 &= h(v_k + L_3) \\
L_4 &= hg(t_k + h, r_k + K_3) \\
v_{k+1} &= (1/6)(L_1 + 2L_2 + 2L_3 + L_4) \\
r_{k+1} &= (1/6)(K_1 + 2K_2 + 2K_3 + K_4)
\end{aligned}
$$

Verlet method

Verlet algorithm is a numerical method spesifically derived to solve Newtons's law for motion. It is also part of a large collection of numerical methods known as predictor-corrector methods.

The basic approach is to compute the slope at t_k(again Eulers method) and use this to compute the predicted position and use this to compute the next slope. Finally we correct the predicted slope by taking the average between the two slopes giving the final value. The algorithm would then look like this:

$$
\begin{aligned}
K_1 &= f(t_k, r_k) \\
r_{k+1} &= r_k + hK_1 \\
K_2 &= f(t_{k+1}, r_{k+1}) \\
r_{k+1} &= r_k + (h/2)(K_1 + K_2)
\end{aligned}
$$

Now this is the general procedure of Verlet method, however for our problem we can use the Velocity Verlet method. This method incorporates the velocity as the name suggests. One note of precaution though, Velocity Verlet assumes that the function for second derivative(the acceleration) isn't dependant on the predicted velocity and only the position. The final algorithm is basically as above, but with the added velocity:

$$\begin{aligned} K_1 &= \frac{h}{2} g(t_k, r_k) \\ r_{k+1} &= r_k + h(v_k + K_1) \\ K_2 &= \frac{h}{2} g(t_{k+1}, r_{k+1}) \\ v_{k+1} &= v_k + K_1 + K_2 \end{aligned}$$

For our problem the function g is just the function for acceleration and h is the same time-step defined in the RK4-method.

Earth-Sun system

For the Earth-Sun system we can assume cirular orbit, the initial velocity is then given by:

$$v_0 = \frac{2\pi r}{T}$$

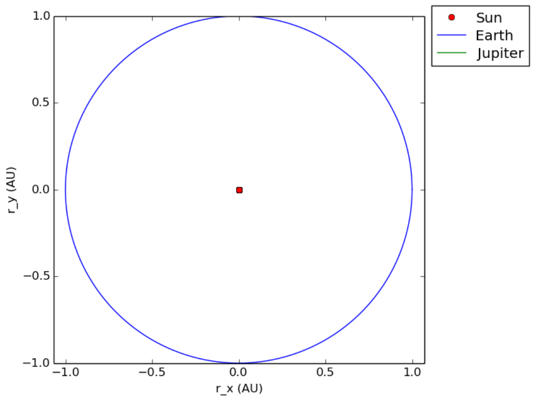
Since the distance is measured in AU(and 1 AU is defined as the distance between the Earth and Sun) and time is measured in years, we can set $r = 1$ and $T = 1$, which gives an initial velocity(for cirular orbit) of:

$$v_0 = 2\pi$$

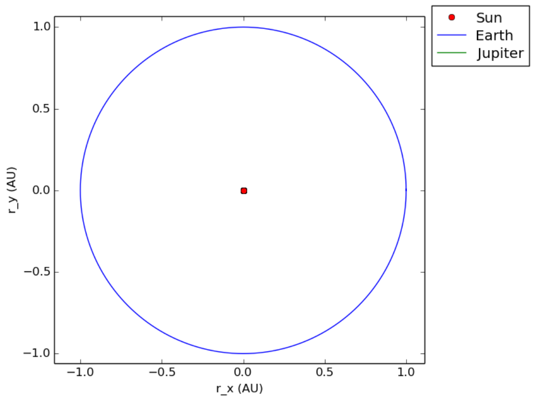
Another quantity to mention is energy, that is, potential and kinetic energy. These two are both conserved in a circular motion because the radius, or the distance to the sun for the earth, is the same for the whole orbit. This means that r is constant. Because the potential energy is defined as $-G\frac{M_{sun}M_{earth}}{r}$ one can easily see that it must be conserved for cirular motion. As for the kinetic energy, defined as $\frac{1}{2}mv^2$, it is conserved because the velocity of the planet doesn't change.

The last value, that is conserved, is the angular momentum. This is also due to arguments above. Both r and v are constant values in a circular motion and the angular momentum is defined as $\vec{L} = \vec{r} \times m\vec{v}$. It is easy to see that every quantity involved are constant.

Using the program mentioned above a plot of the results for Earth-Sun system with the initial velocity above looks like this:

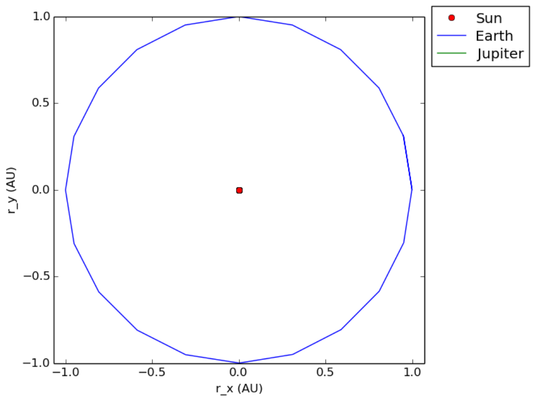


Earth-Sun system with RK4

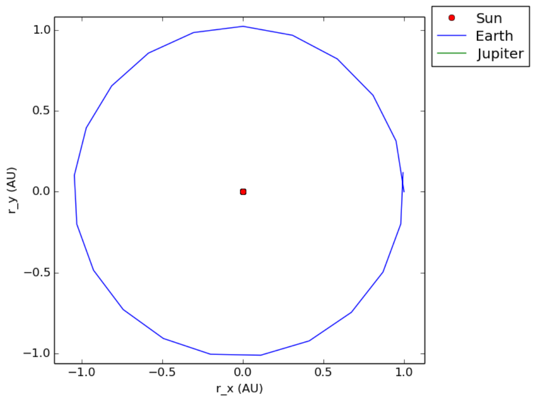


Earth-Sun system with Verlet

The time-step used here is $dt = 0.001$. With such a low dt both methods behave correctly and are very much stable. With some experimentation it was found that RK4 seemingly acted a bit more stable for high values of dt. With a step-value of 0.05 we get these simulations:



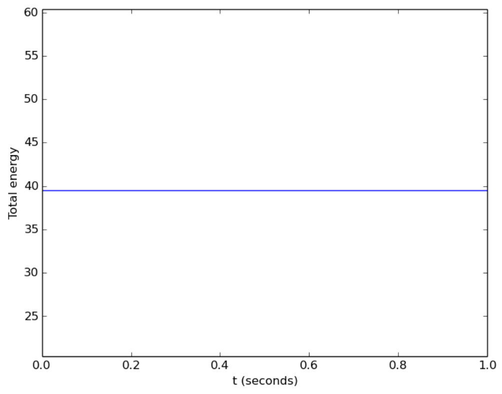
Earth-Sun system with RK4, $dt = 0.05$



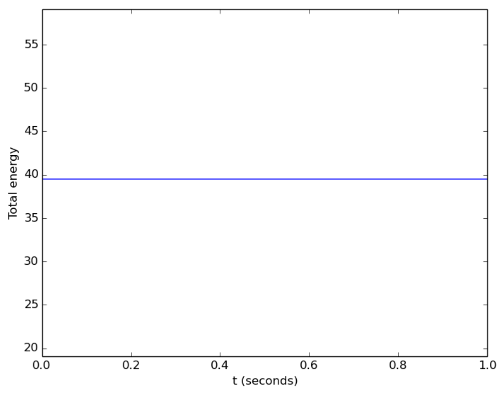
Earth-Sun system with Verlet, $dt = 0.05$

As we can see, RK4 makes the orbit slightly more closed than RK4.

To find out which method is the most stable we can look at the conservation of energy, with other words, check if the potential and kinetic energy are constants. We can plot the total energy of the system as a function of time:

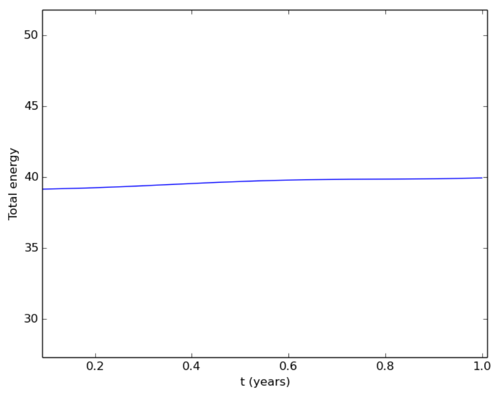


Total energy with RK4, $dt = 0.0001$

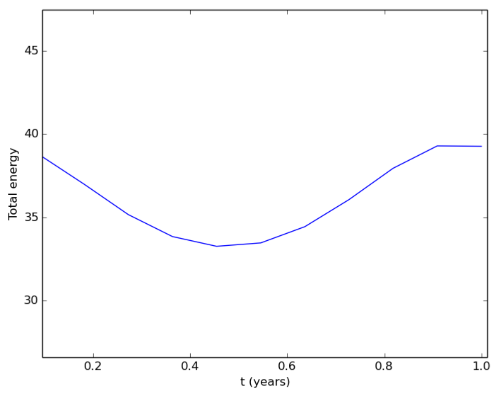


Total energy with verlet, $dt = 0.0001$

As we can see the total energy is very much constant with both methods. However, with a higher time-step we get:



Total energy with RK4, $dt = 0.08$



Total energy with verlet, $dt = 0.08$

As we can see here verlet does behave rather poorly for high step-values.

Escape velocity

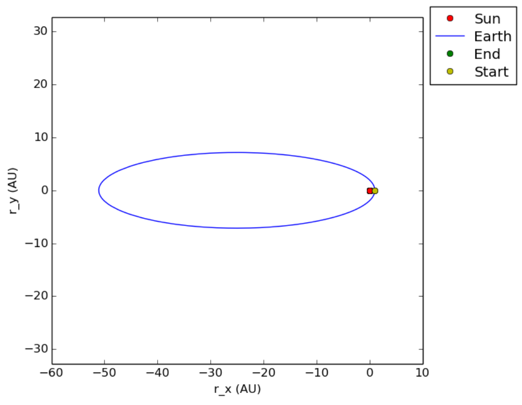
Considering a planet orbiting the sun we can easily find an initial velocity that makes the planet escape from the sun, rather than orbiting it. For both an analytical and numerical approach we can use the fact that in order for one body to escape from another, the total energy of the escaping body must be less than zero, with other words, the kinetic energy must be greater than the potential energy. With this at hand we get the following equation to solve:

$$\frac{1}{2}M_{planet}v_0^2 = G\frac{M_{sun}M_{Planet}}{r}$$
$$v_0 = \sqrt{\frac{2GM_{sun}}{r}}$$

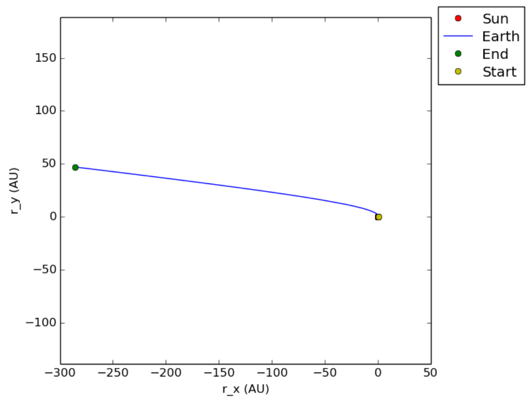
G is still the gravitational constant and r is the distance between the two objects. Note that this velocity doesn't depend on the mass of the planet and only the distance too the sun and the suns mass. Using our sun as reference and putting a planet of distance 1AU apart we get that the initial velocity must be:

$$v_0 \approx 8.89 \text{ au/yr} \approx 52.6 \text{ km/s}$$

If we want to find the velocity numerically one can always go by trial and error. We find that the initial velocity needs to indeed be 8.9 au/yr in order for the planet to escape.



$v_0 = 8.8$ au/yr



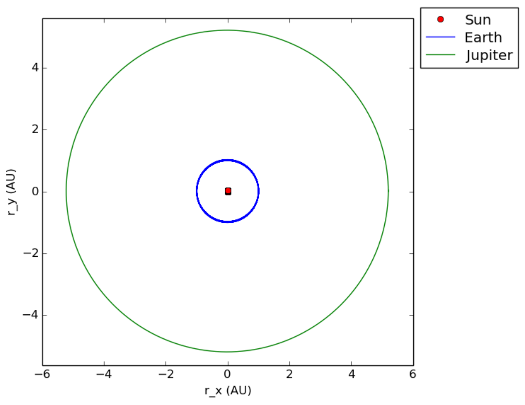
$v_0 = 8.9$ au/yr

Note that the the total time are respectively 133 years and 300 years so it takes a long time for the orbit to complete the larger the initial velocity is. The orbit itself will also be considerably larger.

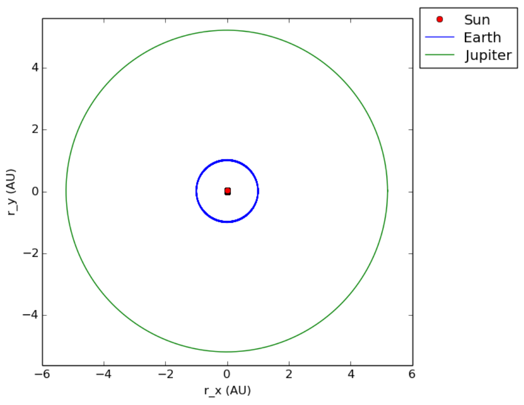
Three-body problem

Till now we have only studied a hypothetical solar system. In reality the other planets in our solar system also affect eachother with gravitational forces. The planet which affects the Earth the most is the most massive planet in our solar system, Jupiter. With Jupiter in consideration our problem is now a three-body problem.

To accommodate for this change we can easily modify the expression for force deduced in the theory section above by just adding the gravitational force between Jupiter and Earth into the loop. This can(should and is) generalized for a number of bodies in the code. We can make a simple simulation with the sun still fixed at the center. The plot looks like this:

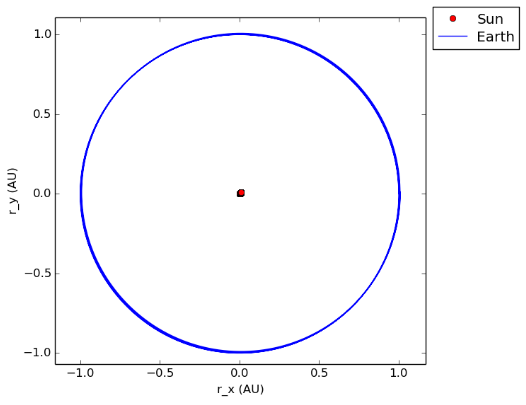


Three-body problem rk4

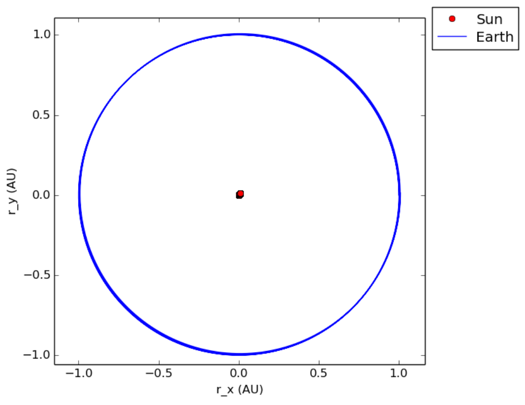


Three-body problem verlet

Plot of just the Earths orbit:



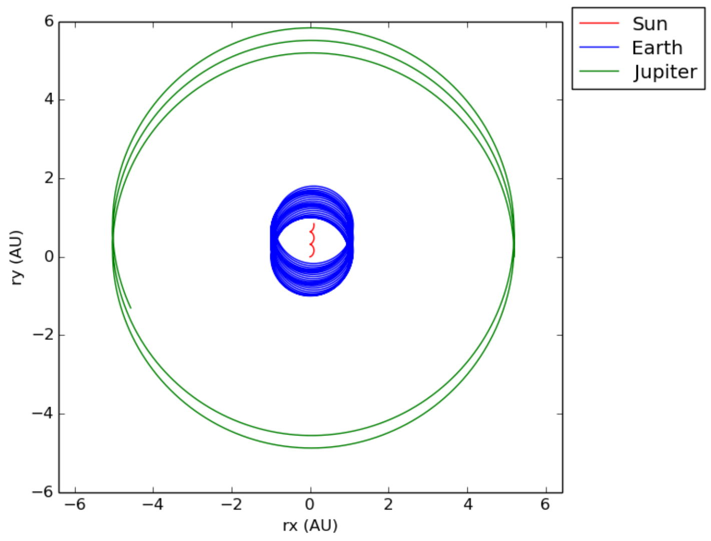
Three-body earth rk4



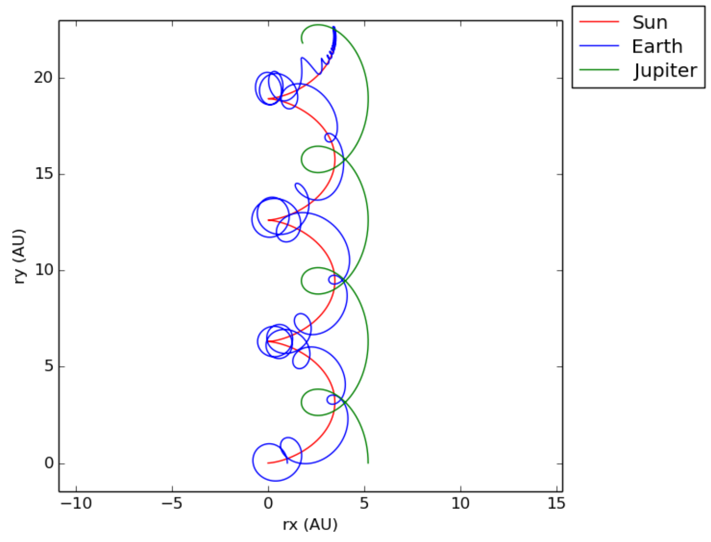
Three-body earth verlet

The total period is here 12 years, which is about the same as the orbital time of Jupiters orbit around the sun. As presumed Earths orbit is affected by Jupiter. It is not a circular orbit anymore, but gets a bit more "wobbly". Note that the last two figures are of earths orbit over a period of 5 years. Also note that both methods are still very mush stable with a low dt(dt is here 0.0001).

To see how much another planet(or any celestial body) affects Earth we can increase Jupiters mass and see how the orbit of changes.



Jupiter mass increased by factor 10



Jupiter mass increased by factor 1000

Earths orbit becomes even more wobbly and increasing Jupiters mass by a factor 1000(putting Jupiters mass equal to the suns mass) Gives a rather unstable orbital path(if we can still call it an orbit...). The total time for both figures is 15 years.

Real three-body simulation

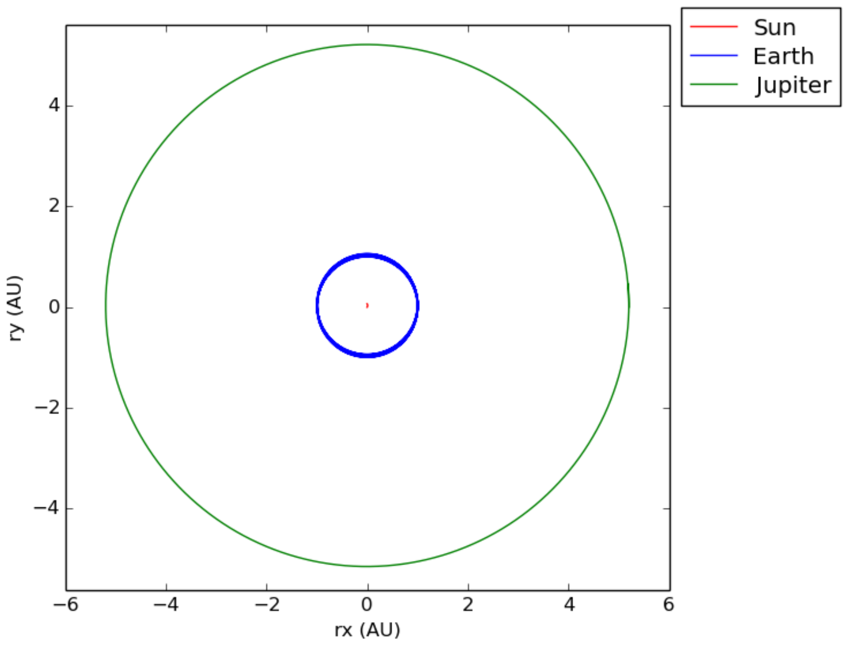
For all the previous calculations we have fixed the sun at the origin, assuming the center of mass is approximately in the sun's position. In reality this is not true. It is much more realistic to set the center of mass at the origin and rather give the sun an initial velocity which makes the angular momentum zero. We can do this by calculating:

$$\vec{L}_{\text{tot}} = \sum_i \vec{r}_i \times m_i \vec{v}_i = 0$$

$$-M_{\text{sun}} \vec{v}_{\text{sun}} + \sum_{i=1} m_i \vec{v}_i = 0$$

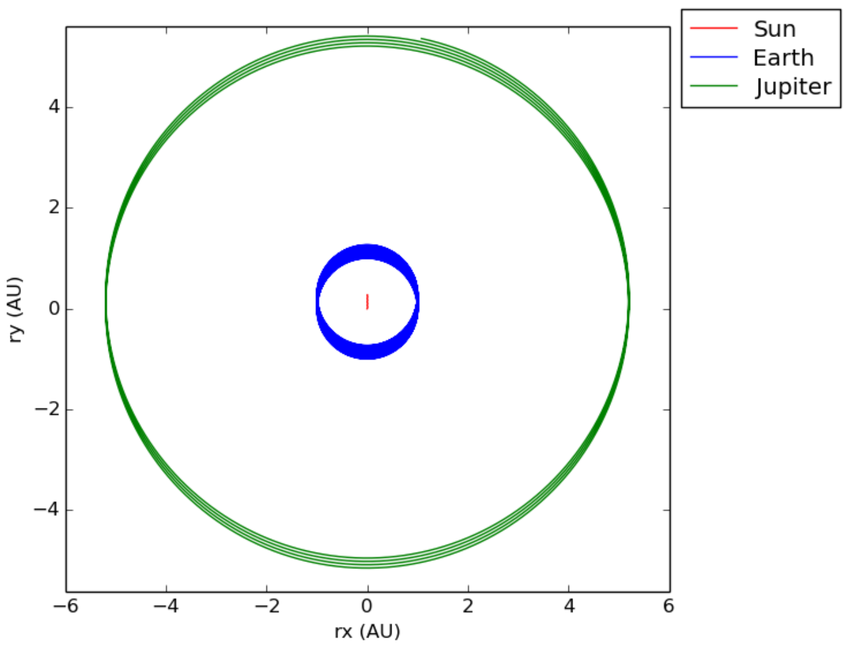
$$\vec{v}_{\text{sun}} = \frac{\sum_{i=1} m_i \vec{v}_i}{M_{\text{sun}}}$$

Now we only need to give the other planets an initial velocity and the suns initial velocity will fall straight out. Now the figures are slightly different:



More realistic three-body simulation

We don't see much difference from the previous results, however if the period is increased one can easily see that the sun also moves in kind of a line. Note that the initial velocities are all only in y-direction. This is why the sun only movies in y. In reality it would actually move in a more circular motion around the center. A plot with period increased to 50 years:

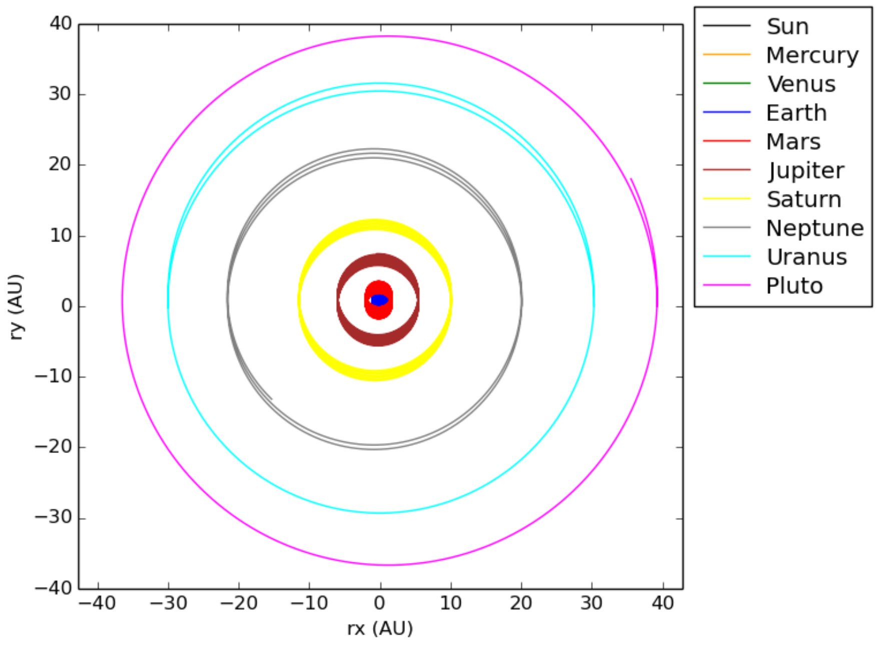


Three-body simulation 50 years

As we can see the sun is also moving. Note that both of the above simulations are done with RK4.

Complete simulation of the solar system

With the results we have at hand and information gained from simulating a two- and three-body problem, we can step up and simulate the whole solar system. Adding all the planets, giving them an initial velocity from known results and giving the sun an initial velocity which gives zero total angular momentum gives a plot like this.



The whole solarsystem(ish)

Note that for this simulation we haven't taken into account the perihelion precession of Mercury. The several moons and the rocks in the asteroid belt are also ignored. The simulation itself is run for 250 years, which is about the orbital time of pluto.

Sources

Theory behind RK4 and Verlet

Lecture notes section 8:
`http://www.uio.no/studier/emner/matnat/fys/FYS3150/h14/undervisningsmateriale/Lecture%20Notes/lecture2014.pdf` Project text:
`http://www.uio.no/studier/emner/matnat/fys/FYS3150/h14/undervisningsmateriale/projects/project-3--deadline-october-20/project3_2014.pdf`

Information on planets

Project Text:
`http://www.uio.no/studier/emner/matnat/fys/FYS3150/h14/undervisningsmateriale/projects/project-3--deadline-october-20/project3_2014.pdf`
Wikipedia:
-Sun: `http://en.wikipedia.org/wiki/Sun`
-Mercury: `http://en.wikipedia.org/wiki/Mercury_(planet)`
-Venus: `http://en.wikipedia.org/wiki/Venus`
-Earth: `http://en.wikipedia.org/wiki/Earth`

-Mars: `http://en.wikipedia.org/wiki/Mars`
-Jupiter: `http://en.wikipedia.org/wiki/Jupiter`
-Saturn: `http://en.wikipedia.org/wiki/Saturn`
-Uranus: `http://en.wikipedia.org/wiki/Uranus`
-Neptune: `http://en.wikipedia.org/wiki/Neptune`
-Pluto: `http://en.wikipedia.org/wiki/Pluto`