FYS3150 - Computational Physics

Project 3

The Solar system

Filip Henrik Larsen
filiphenriklarsen@gmail.com

Dato: 19. oktober 2014

Introduction

In this project we made a simple model of the solar system. We assumed the solar system to be co-planner, and considered every planet/object as point particles. We also only considered the effect of gravitational forces. This of course simplifies the problem, but it wouldn't be difficult to implement an additional dimension in space. The aim of this project was to learn how we can solve second-order ordinary differential equations, such as Newtons second law, as a set of coupled first order differential equations. Newtons second law of motion states that:

$$m\frac{\partial^2 x}{\partial t^2} = \Sigma F$$

If the only force doing work is the gravitational force between two objects, this is:

$$\cancel{m}\frac{\partial^2 x}{\partial t^2} = -\frac{GM\cancel{m}}{x^2} \tag{1}$$

Stating that

$$v(t) = \frac{\partial x}{\partial t}$$

We can express (1) as two coupled first order differential equations.

$$\frac{\partial x}{\partial t} = v(t) \qquad \text{and} \qquad \frac{\partial v}{\partial t} = -\frac{GMm}{x^2}$$

We were to solve these using the *Runge-Kutta 4 (RK4)* and the *Verlet* methods. Through this project we have also become better acquainted with the use of classes in C++.

Algorithms

The main goal of the algorithms is to use the current value of position and velocity to predict the next. Both the methods we use in this project are derived using Taylor series, however I will not derive the RK4.

Runge-Kutta 4

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

$$\begin{aligned}
k1 &= f\left(x_i,\ y_i\right) \\
k2 &= f\left(x_i + \frac{h}{2},\ y_i + k_1\frac{h}{2}\right) \\
k3 &= f\left(x_i + \frac{h}{2},\ y_i + k_2\frac{h}{2}\right) \\
k4 &= f\left(x_i + h,\ y_i + k_3 h\right)
\end{aligned}$$

The main thought is, as in the Euler method, the first derivative timed the step length, takes us to the next position. However the RK4 does an attempt at making a better approximation of the first derivative throughout the interval. It does this by taking a weighted mean value of it at four positions. To explain it simple:
k_1 is the slope at the beginning of the interval (current position). We follow this slope one half-step from the initial position and make a predicament of the slope in this position, k_2. We then follow k_2 from the initial position one half-step, calculate k_3. From the initial position we follow k_3 one entire step and calculate the slope at this point k_4. Finally we take the average of them, with the two slopes halfway through weighted double. This generally makes a better approximation, more precise calculations, but of course it is also more time-consuming as it takes 5 calculations to predict the next position. (I write position, but it can just as well be velocity if $f(x, y)$ is the acceleration. The point is that $f(x, y)$ is the derivative of y with respect to x.)

However in this project we had two coupled first order differential equations. I find it hard to explain, so I will basically explain my algorithm.

$$\begin{aligned}
k_{1p} &= V * dt \\
k_{1v} &= Acceleration(P) * dt \\
k_{2p} &= (V + 0.5 * k_{1v}) * dt \\
k_{2v} &= Acceleration(P + 0.5 * k_{1p}) * dt \\
k_{3p} &= (V + 0.5 * k_{2v}) * dt \\
k_{3v} &= Acceleration(P + 0.5 * k_{2p}) * dt \\
k_{4p} &= (V + k_{3v}) * dt \\
k_{4v} &= Acceleration(P + k_{3p}) * dt \\
P+ &= (1/6.) * (k_{1p} + 2 * (k_{2p} + k_{3p}) + k_{4p}) \\
V+ &= (1/6.) * (k_{1v} + 2 * (k_{2v} + k_{3v}) + k_{4v})
\end{aligned}$$

k_{1p} calculates the change in position while k_{1v} calculates change in velocity. From this step on the k_p's uses the sum of the initial velocity and the change in velocity in its arguments, while the k_v's uses the sum of the initial position and the change in position.

Verlet

The verlet algorithm is simple to derive. If we taylor expand $x(t + h)$ we get:

$$x(t+h) \simeq x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{6}x'''(t) + O(h^4)$$

Taylor expanding $x(t - h)$ we get:

$$x(t-h) = x(t) - hx'(t) + \frac{\hbar^2}{2}x''(t) - \frac{h^3}{6}x'''(t) + O(h^4)$$

Adding these we get:

$$x(t+h) + x(t-h) = 2x(t) + h^2x''(t) + O(h^4)$$

Using the discretization

$$x(t \pm h) = x_{i\pm1} \qquad \text{and} \qquad x(t) = x_i$$

We have derived the Verlet algorithm

$$x_{i+1} = 2x_i - x_{i-1} + h^2 x_i'' + O(h^4)$$

This algorithm is really great if we do not have to worry about the first derivative. It is simple, quick and the error goes as $O(H^4)$. The problem, however, is that it is not self starting. As we need to know the two first initial values of x in order to calculate the next. In my script I solved this issue simply by setting

$$x_1 = x_0 + v_0 dt + \frac{1}{2}\frac{d^2 x_0}{dt^2}$$

For the velocities I used the formula

$$v_i = \frac{x_{i+1} - x_{i-1}}{2h} + O(h^2)$$

As we can see the truncation error goes as $O(h^2)$ which is not that great if the precision of velocity matters.

The classes

I chose to create two classes in this project. I'm not sure if I overkilled it a bit with so many functions in *System*, I doubt I will need all of them if I should ever reuse this code. However for this project it is awesome.

Planet

Stores the name, mass, initial position and velocity of every object in a list *planets*. The data is entered through the function *Addplanet* taking them as input. The units I have set to:

Length: AU Mass: $S_\odot$ Time: Years

The *Addplanet*-function takes masses in *kg* as input, but converts it inside the class. I did it this way because in the problem text the masses were given in *kg*.

System

This class contains all the functions needed to calculate the motion of the entire system of planets (and the Sun). One function calculates the acceleration of every planet both due to the Suns gravitational force, but also the gravitational force caused by every other planet. One thing worth noticing is the *if*-test. It reassures that Sun is the point of reference in the system. I did it this way because the problem text said to fixthe sun in the origin. The reasoning for this is easy to grasp. The acceleration of a planet seen from the Sun, must be the sum of the planets own acceleration toward the Sun and the Suns acceleration toward the planet. Thus the acceleration of a planet as seen from the Sun is:

$$a_p' = \frac{F}{m_p} + \frac{F}{M_\odot} = F\left(\frac{1}{m_p} + \frac{1}{M_\odot}\right)$$

This configuration was especially convenient when we were to set the mass of Jupiter to be 1000 larger than the real value. Had the Sun then been fixed without considering this interaction it would not return a realistic result at all. And had the Sun been free and we just chose some stationary point of reference, the entire system would move with the center of mass. Another good alternative would of course be to have the center of mass as point of reference, but since I was told to fix the Sun, I did it this way.
Energy is a function which calculates the kinetic, potential and total energy of the system several times throughout the time period.
Setup inserts the initial positions and velocities from the data stored in the list *planets*. *Solve* is the function actually calculating the the evolution of the system. Here there are three possible methods to use: Euler, RK4 and Verlet. Each of them are set up to carry us through one iteration. So we call upon them continuously. For every such iteration we receive the evolvement of one timestep. The time, positions and energies are then written to individual files.

Main

is only used to add planets, call the *setup* function and start the *solve* function. Lastly we tell the system to run a python-script which plots the trajectories and energies, and then delete the data files.

Results

We started out trying only to make a model consisting of the Sun and the Earth. As we were given the distance between them and we chose the sun to be in the origin, the initial position of Earth was set to 1AU. We did not get a specific value for the velocity though, so I set this value to that of a circular motion. For circular motion we have the relation:

$$ma = \frac{mv^2}{r}$$
$$\frac{\partial^2 r}{\partial t^2} = v^2$$

Using expression (1) we have:

$$v = \frac{\sqrt{GM}}{r} \tag{2}$$

In as the mass M is set to 1, I do not bother includeing it in the script. Running the program with these settings, It returned the plots bellow.

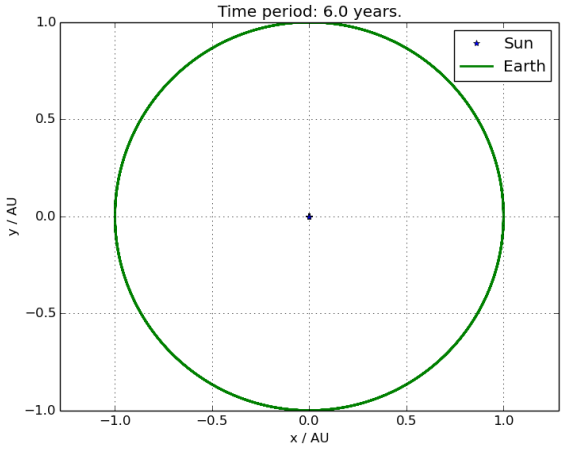

Figure 1: Trajectory of the Earth througout six years.

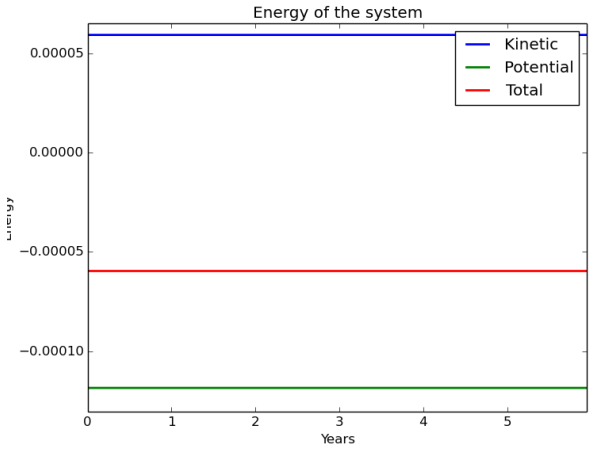

Figure 2: Energy levels compaired throughout six years.

As Figure 1 shows, Earth has a perfectly circular trajectory about the Sun, as we would hope. As we gave the Earth initial conditions for a circular motion, the absolute velocity and the distance from the Sun should be constant (=1AU). Though it will seem as though the motion is about the Sun, it really is about the center of mass. In fact the Sun moves about the center of mass as well, but scince the Sun is so massive compaired to all the planets, the center of mass is very close to the Sun (In reality inside the Sun, most of the time). We therfore don't notice it compaired to the other planets motion. Figure 2 shows that the the kinetic and potential energy are constant throught the timeinterval, or conserved as we like to say. We notice also that the magnitude of the potential energy is twice that of the kinetic. This is also expected since for circular motion we have velecity as shown in (2), and hence:

$$E_k = \frac{1}{2}mv^2 = \frac{1}{2}\frac{GMm}{r^2} = \frac{1}{2}E_p$$

To check if the program gave accurate results I simmulated the Eart-Sun system for 10 years, and checked what posission the Earth had at the end of the simulation. Obviously the analytical solution is $x = 1, y = 1$, but there is bound to bee some error when we do this numerically. How good approximation the simulation gives for the Earths position depends of course on the time step length. I used $\Delta t = 1$hour in the simulation giving the plots above. I ran the script with both the RK4 and the Verlet algorithms. The result can be seen below.

Time step	X	Y
10 days	0.9824267694	0.185216948800
5 days	0.9962168736	0.086806629540
1 day	0.9998452714	0.017590538360
12 hours	0.9999596430	0.008983982664
1 hour	0.9999999289	0.000376988563

Table 1: Position of Earth after 10 years using RK4.

Time step	X	Y
10 days	0.9044665532	-0.4265455881
5 days	0.9976792657	-0.06811910745
1 day	0.9999351843	0.01138619723
12 hours	0.9999723784	0.00743259780
1 hour	0.9999999330	0.00036621065

Table 2: Position of Earth after 10 years using Verlet.

As we see even with a time step length of 5 days, the approximation is quite good for both of them and it seems to only get better with decreasing time step length. Taking this into consideration I would say the program is quite stable as a function of Δt.

Next we were to place a planet at a distance of 1AU from the Sun, and find out what initial velocity it should have in order to get a circular trajectory. We were supposed to find it by trail and error, which I did't do, because I gave the correct value the first time. I knew beforehand that the velocity should be as in (2), and thus I got a plot just as in Figure(1). I tried then to reduce the velocity to see what I would have seen If I had given some random value.

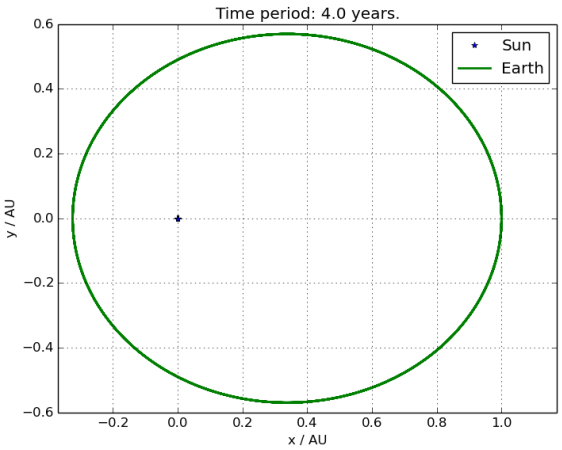

Figure 3: Trajectory of a planet starting at position 1AU with initial velocity $0.7\sqrt{GM}/r$.

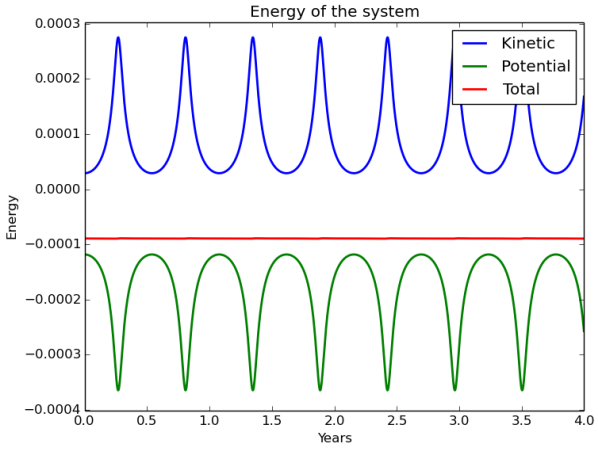

Figure 4: Energy levels compaired throughout 4 years, for the system of a planet starting at position 1AU with initial velocity $0.7\sqrt{GM/r}$.

Figure 3 shows the tjajectory of the planet. It is lebled «Earth», but it could be any planet really. Since the equation for the velocity of circular motion does not depend on the mass of the planet itself. The mass of the planet should however be much smaller than that of the Sun, since the r in equation (2) really is the distance to the center of mass of the system.
In this situation, where we do not get, nor expected, a circular motion, the periodicity in the exchange between kinetic and potential energy is physical correct. The planet will have the largst velocity/kinetic energy when closest to the sun, at the same time as it has a maximal potential energy. The fact that it has a relatively large velocity here can be seen in the plot as it the planet obviously is in this position/potential energy level for a short time. And it has the slowest velocity at the position farthest from the Sun. The total energy is conserved.
Adding Jupiter, also with a «circular velocity», the program returned the following:

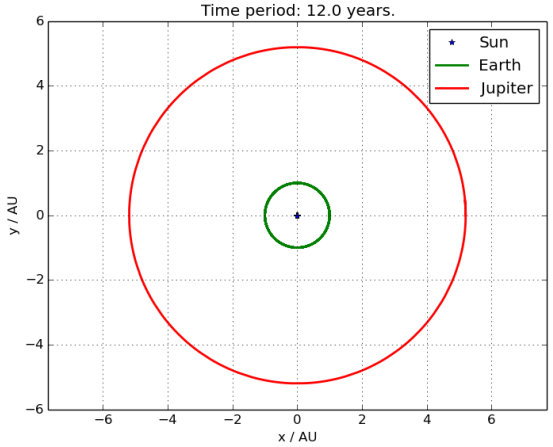

Figure 5: Trajectory of the Earth and Jupiter, both with velocities of circular motion.

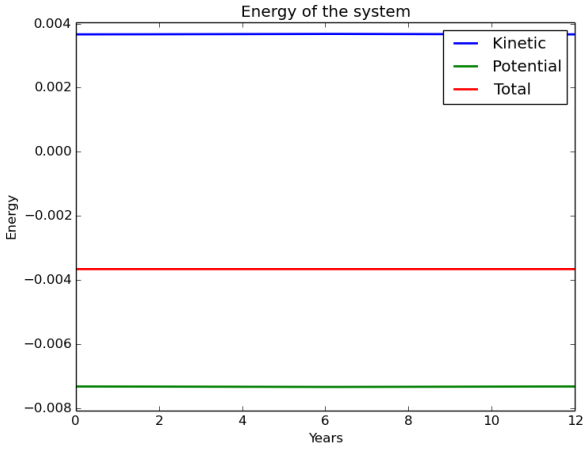

Figure 6: Energy levels of the system consisting of the Sun, Earth and Jupiter in an interval of 12 years.

Again, the energy levels are constant as they should be, as ther is (almost) perfect circular motion. The reason its not perfectly circular is the gravitaional interaction between the Earth and Jupiter. These forces are very small compaired to the force exerted from the Sun, and therefore we cant really notice it in the plot.
We now multiply the mass of Jupiter by a factor of 10.

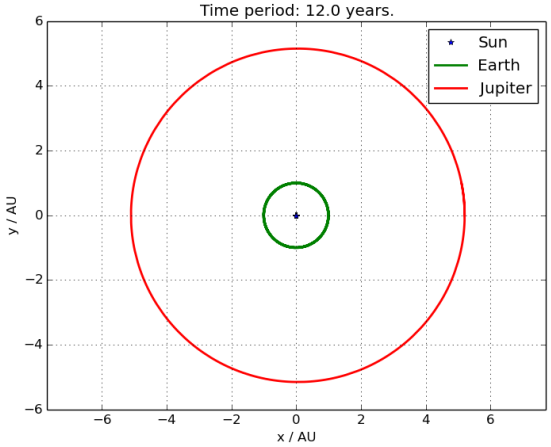

Figure 7: Trajectory of the Earth and Jupiter, with the mass of Jupiter multiplied by a factor of 10.

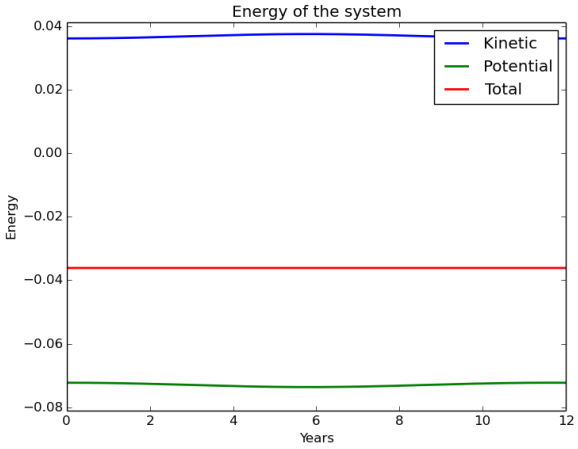

Figure 8: Energy levels of the system consisting of the Sun, Earth and Jupiter in an interval of 12 years, with the mass of Jupiter multiplied by a factor of 10.

It may be hard to read, but the trajectories in Figure 7 are eliptical. This can easier be seen from Figure 11, as we se the slight bend of the curve.
Icreasing the mass of Jupiter with a factor of 1000 gives more drastic results. By doing this, we give Jupiter almost the same mass as the Sun, meaning that the grvitational pull on Earth from Jupiter will be stronger then the Suns, if the Earth is closer to Jupiter than the Sun.

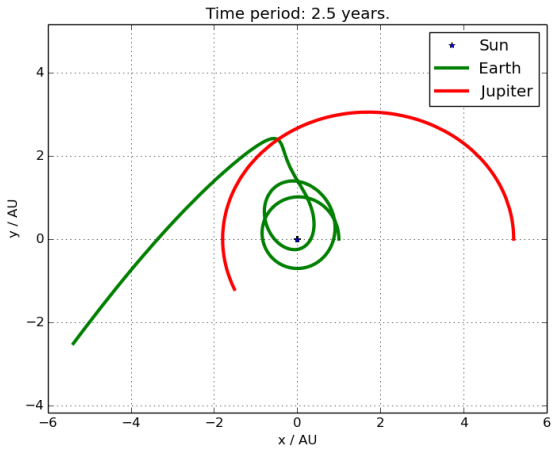

Figure 9: Trajectory of Earth and Jupiter relative to the Sun, with the mass of Jupiter multiplied by 1000.

As we se in Figure 9 the gravitational pull from Jupiter alters the Earth orbit of the Sun in such a way that it comes to a state where the gravitational force from Jupiter is larger than the gravitational force from the Sun, and its velocity is straight towards Jupiter. The Earth then accelerates toward Jupiter gaining increadible speed. So much in fact that its kinetic energy is much larger than it potential. The result is that the Earth gets flung away. The trajectory of Jupiter will then be an elipse about the center of mass. As will the Suns, but I chose the Sun to be the point of referance. This is shown in Figure 10.

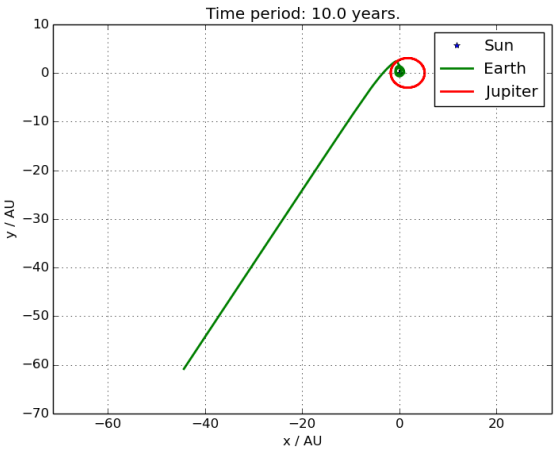

Figure 10: Trajectory of the Earth and Jupiter, with the mass of Jupiter multiplied by a factor of 1000.

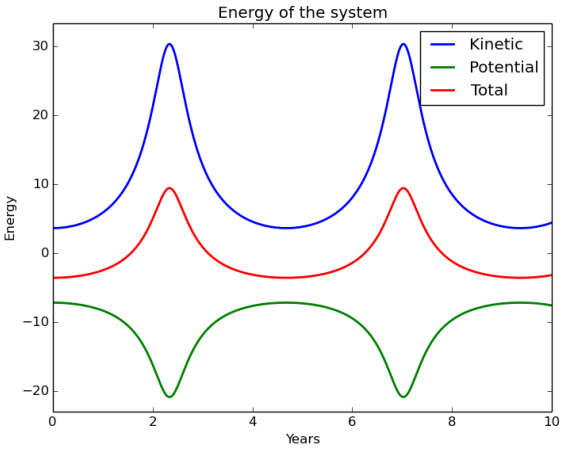

Figure 11: Energy levels of the system consisting of the Sun, Earth and Jupiter in an interval of 12 years, with the mass of Jupiter multiplied by a factor of 1000.

The Energy diagram is more difficult to discribe. The total energy varies! The total energy is obviously not conserved. My conclution is that there are relativistic effects which we do not take into consideration.
I then ran the program using all the planets, giving them circular velocity.

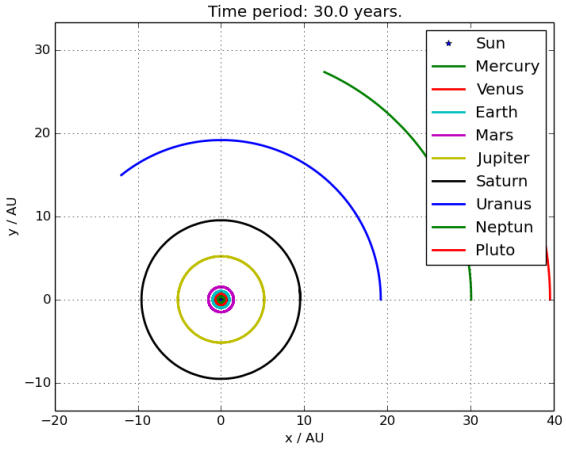

Figure 12: Trajectory of the Earth and Jupiter, with the mass of Jupiter multiplied by a factor of 10.

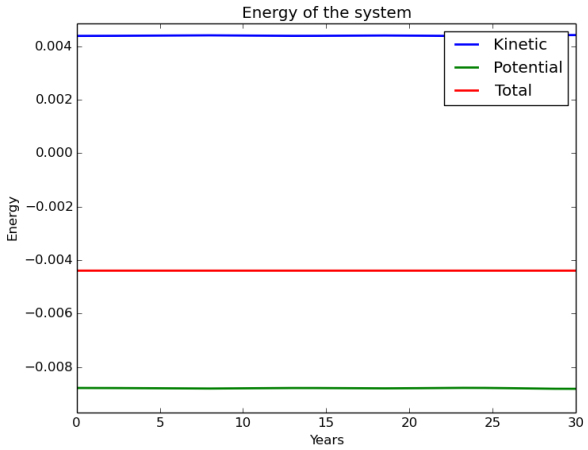

Figure 13: Energy levels of the system consisting of the entire Solar system in an interval of 30 years.

Githug repository:

`https://github.com/filiphl/FYS3150.git`