Exercise No. 2

David Bubeck, Pascal Becht, Patrick Nisblè

May 3, 2017

2 - Error analysis of the Euler scheme

a)

At first we write the code for the 2 - body problem by using the forward Euler algorithm. We integrate the problem for one orbit and plot it on a double - logarithmic scale. The error will be plotted as a function of Δt. Therefore we will be using three different eccentricities and various different time steps.

Function that integrates the 2 - body problem using the forward Euler algorithm.

Listing 1: Exercise02a.py

```python
import numpy as np
import matplotlib.pyplot as plt


#function that integrates the 2 - body - problem using the forward
#euler algorithm. The initial velocity needs to be two dimensional
#vector, the step size h can be choosen. The program breaks when the
#orbit is complete.
def euler(v_0, h, s_0 = np.array([0, 1])):
    S = [s_0]    #list for spatial coordinate
    V = [v_0]    #list for relative velocity
    E = []       #list for the Energy
    test = 0     #variable to check if the orbit is complete
    step = 0
    maxstep = 1000000

    while((test < 2) and (step < maxstep)):
        S.append(S[-1] + h * V[-1])
        V.append(V[-1] - h * S[-1] / (np.sqrt(S[-2][0]**2 + S[-2][1]**2)**3))
        E.append((0.5 * (V[-1]**2)[0] + (V[-1]**2)[1]) - (1/((S[-1]**2)[0] + (S[-1]**2)[1])**0.5))

        if((S[-1])[0] < 0):
            test = 1
        if(test == 1 and (S[-1])[0] > 0):
            test = 2

    return np.array(S), np.array(V), np.array(E)
```

Now we plot a circular orbit with the error in energy in a additional plot. The values that were used are given in the code.

Listing 2: Exercise02a.py

```
#for the given initial values the eccentricity is zero,
#therefore the orbit is circular
S, V, E = euler(np.array([1, 0]), 0.0001)

plt.figure(figsize = (6.5, 10))
plt.plot(S[:, 0], S[:, 1])
plt.xlabel('x')
plt.ylabel('y')
plt.show()

energyerror = []
H = [0.01, 0.001, 0.0001, 0.0001, 0.00001]
for h in H:
    S, V, E = euler(np.array([1, 0]), h)
    energyerror.append((abs(E[-1] - E[0])) / (abs(E[0])))

plt.plot(H, energyerror)
plt.xscale('log')
plt.yscale('log')
plt.xlabel('Stepsize h')
plt.ylabel('error in energy')
plt.show()
```

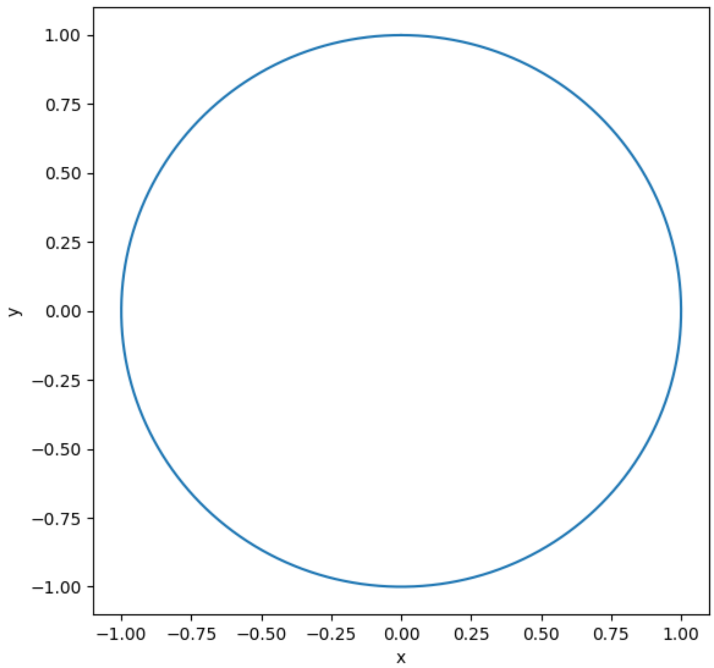

Figure 1: First orbit with forward Euler algorithm

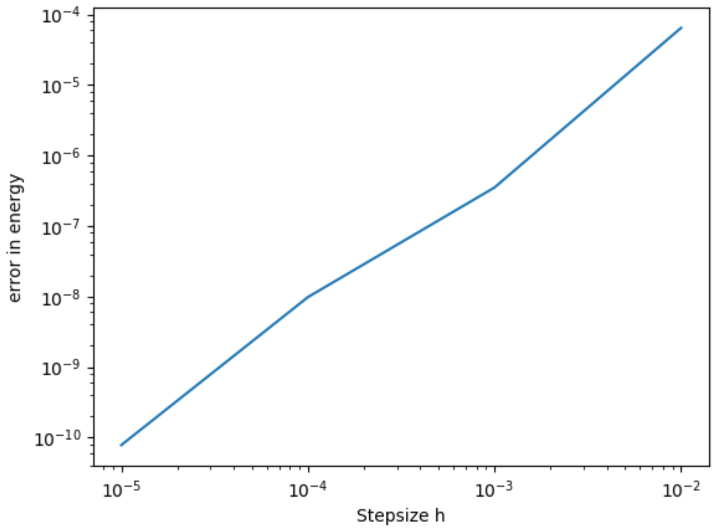

Figure 2: First orbit energy error with forward Euler algorithm

We repeat the calculation for two more different initial velocities.
For the second orbit we chose:

Listing 3: Exercise02a.py

```
#use different initial velocity to achieve another orbit
S, V, E = euler(np.array([1.3, 0]), 0.0001)

plt.figure(figsize = (6.5, 10))
plt.plot(S[:, 0], S[:, 1])
plt.xlabel('x')
plt.ylabel('y')
plt.show()

energyerror = []
H = [0.01, 0.001, 0.0001, 0.0001, 0.00001]
for h in H:
    S, V, E = euler(np.array([1.3, 0]), h)
    energyerror.append((abs(E[-1] - E[0])) / (abs(E[0])))

plt.plot(H, energyerror)
plt.xscale('log')
plt.yscale('log')
plt.xlabel('stepsize h')
plt.ylabel('error in energy')
plt.show()
```

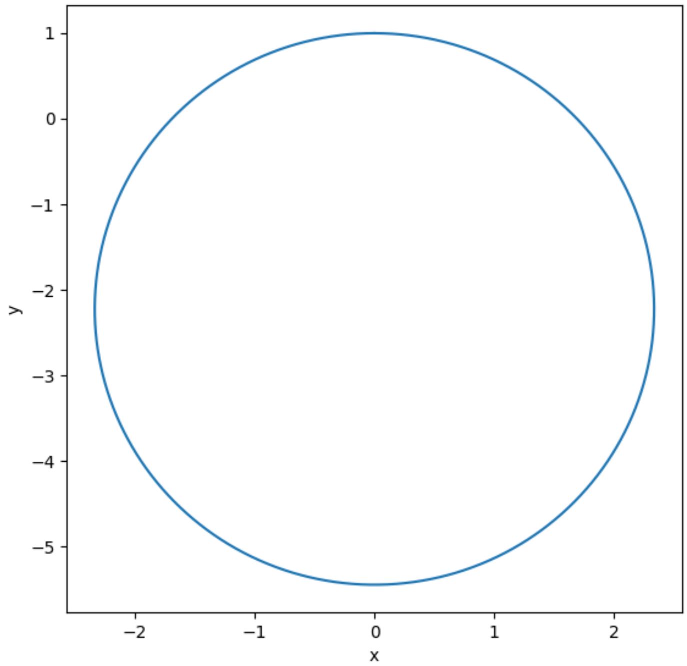

Figure 3: Second orbit with forward Euler algorithm

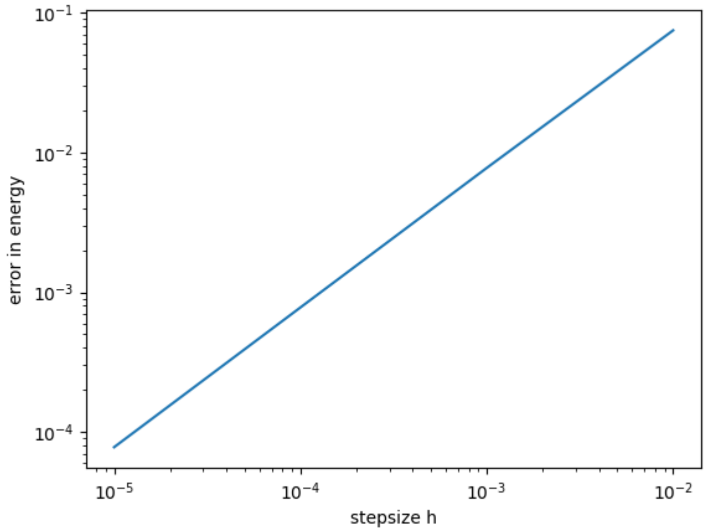

Figure 4: Second orbit energy error with forward Euler algorithm

For the third orbit we chose:

Listing 4: Exercise02a.py

```
#do the same for the third orbit with again another initial velocity
S, V, E = euler(np.array([0.8, 0]), 0.001)

plt.figure(figsize = (6.5, 10))
plt.plot(S[:, 0], S[:, 1])
plt.xlabel('x')
plt.ylabel('y')
plt.show()

energyerror = []
H = [0.01, 0.001, 0.0001, 0.0001, 0.00001]
for h in H:
    S, V, E = euler(np.array([0.8, 0]), h)
    energyerror.append((abs(E[-1] - E[0])) / (abs(E[0])))

plt.plot(H, energyerror)
plt.xscale('log')
plt.yscale('log')
plt.xlabel('stepsize h')
plt.ylabel('error in energy')
plt.show()
```

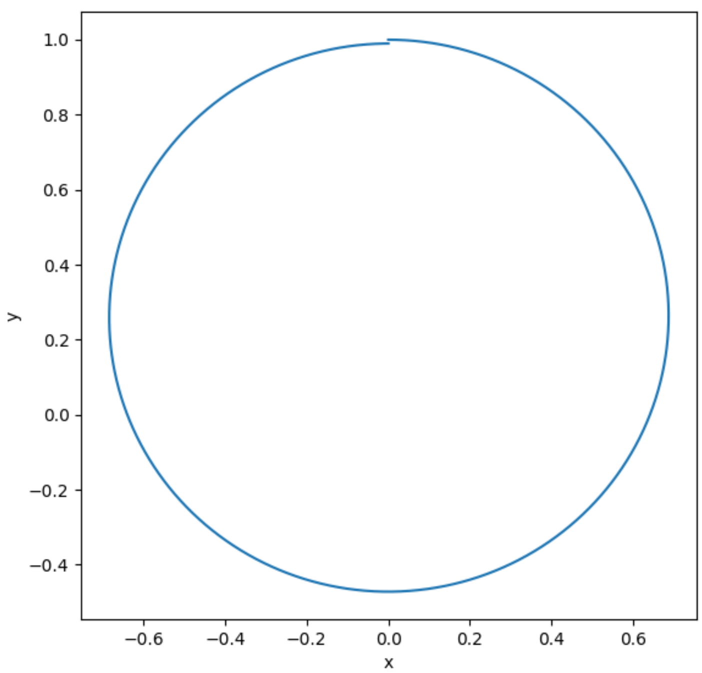

Figure 5: Third orbit with forward Euler algorithm

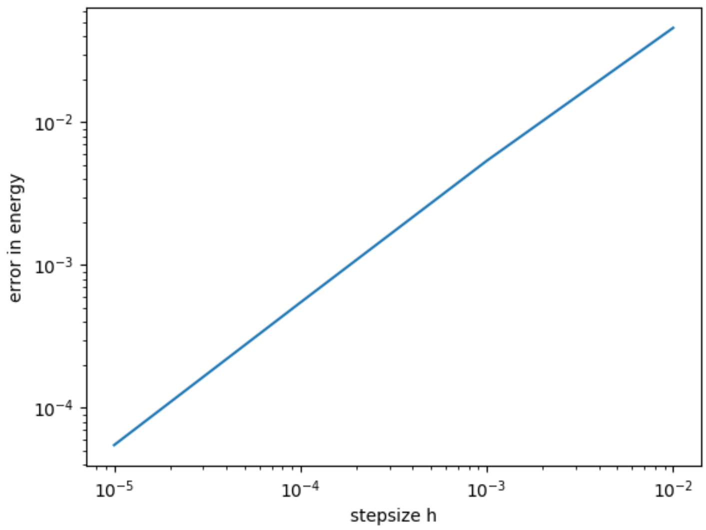

Figure 6: Third orbit energy error with forward Euler algorithm

b)

Now we write the code for the 2 - body problem by using the Leapfrog algorithm. We integrate the problem for one orbit and plot it on a double - logarithmic scale. The error will be plotted as a function of Δt. Therefore we will be using three different eccentricities and various different time steps.

Function that integrates the 2 - body problem using the Leapfrog algorithm.

Listing 5: Exercise02b.py

```python
import numpy as np
import matplotlib.pyplot as plt


#function that integrates the 2 - body - problem using the leapfrog
#algorithm. The initial velocity needs to be two dimensional
#vector, the step size h can be choosen. The program breaks when the
#orbit is complete.
def leapfrog(v_0, h, s_0 = np.array([0, 1])):
    S = [s_0]    #list for spatial coordinate
    V = [v_0]    #list for relative velocity
    E = []       #list for the Energy
    #first value for the acceleration
    a = [-1*(np.array([(S[-1])[0], (S[-1])[1]]))/(((S[-1]**2)[0] + (S[-1]**2)[1])**1.5)]
    test = 0    #variable to check if the orbit is complete
    step = 0
    maxstep = 1000000

    while((test < 2) and (step < maxstep)):
        vhalf = V[-1] + 0.5*h*a[-1]
        S.append(S[-1] + h*vhalf)
        a.append(-1*(np.array([(S[-1])[0], (S[-1])[1]]))/(((S[-1]**2)[0] + (S[-1]**2)[1])**1.5))
        V.append(vhalf + 0.5*h*a[-1])
        E.append((0.5*(V[-1]**2)[0] + (V[-1]**2)[1]) - (1/((S[-1]**2)[0] + (S[-1]**2)[1])**0.5))

        if((S[-1])[0] < 0):
            test = 1
        if(test == 1 and (S[-1])[0] > 0):
            test = 2

    return np.array(S), np.array(V), np.array(E)
```

Now we plot a circular orbit with the error in energy in a additional plot. The values that were used are given in the code. To be precise we going to use the same values as before.

Listing 6: Exercise02b.py

```
#for the given initial values the eccentricity is zero,
#therefore the orbit is circular
S, V, E = leapfrog(np.array([1, 0]), 0.0001)

plt.figure(figsize = (6.5, 10))
plt.plot(S[:, 0], S[:, 1])
plt.xlabel('x')
plt.ylabel('y')
plt.show()

energyerror = []
H = [0.01, 0.001, 0.0001, 0.0001, 0.00001]
for h in H:
    S, V, E = leapfrog(np.array([1, 0]), h)
    energyerror.append((abs(E[-1] - E[0])) / (abs(E[0])))

plt.plot(H, energyerror)
plt.xscale('log')
plt.yscale('log')
plt.xlabel('Stepsize h')
plt.ylabel('error in energy')
plt.show()
```

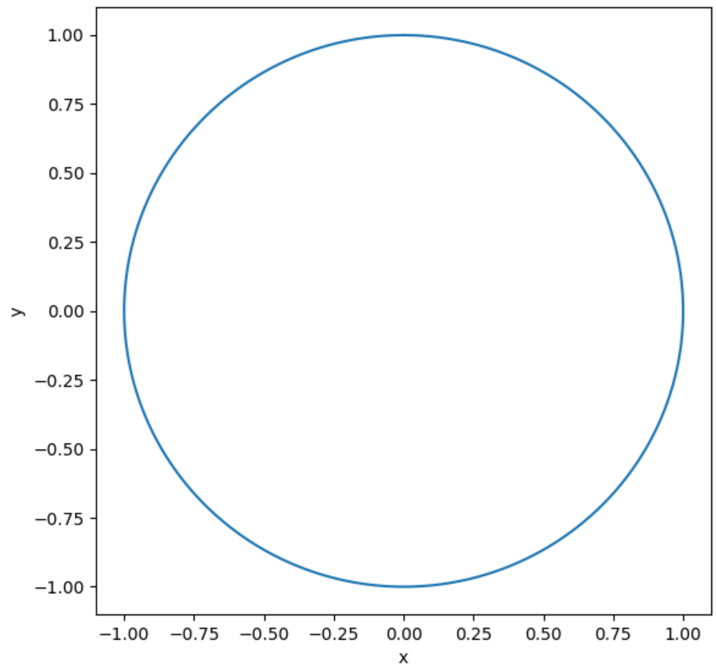

Figure 7: First orbit with leapfrog algorithm

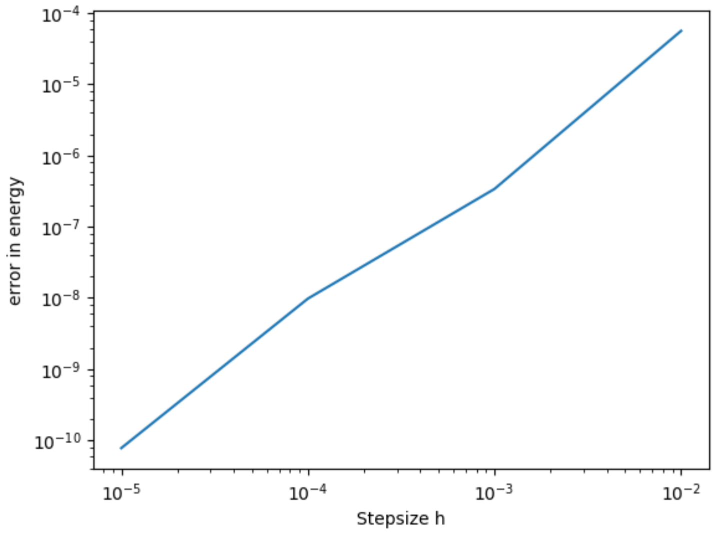

Figure 8: First orbit energy error with leapfrog algorithm

We repeat the calculation for two more different initial velocities.
For the second orbit we chose:

Listing 7: Exercise02b.py

```
#use different initial velocity to achieve another orbit
S, V, E = leapfrog(np.array([1.3, 0]), 0.0001)

plt.figure(figsize = (6.5, 10))
plt.plot(S[:, 0], S[:, 1])
plt.xlabel('x')
plt.ylabel('y')
plt.show()

energyerror = []
H = [0.01, 0.001, 0.0001, 0.0001, 0.00001]
for h in H:
    S, V, E = leapfrog(np.array([1.3, 0]), h)
    energyerror.append((abs(E[-1] - E[0])) / (abs(E[0])))

plt.plot(H, energyerror)
plt.xscale('log')
plt.yscale('log')
plt.xlabel('stepsize h')
plt.ylabel('error in energy')
plt.show()
```

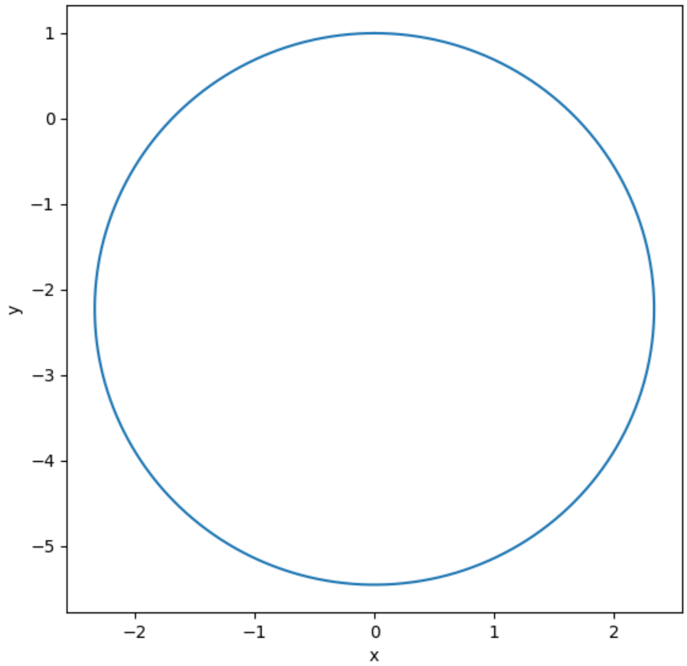

Figure 9: Second orbit with leapfrog algorithm

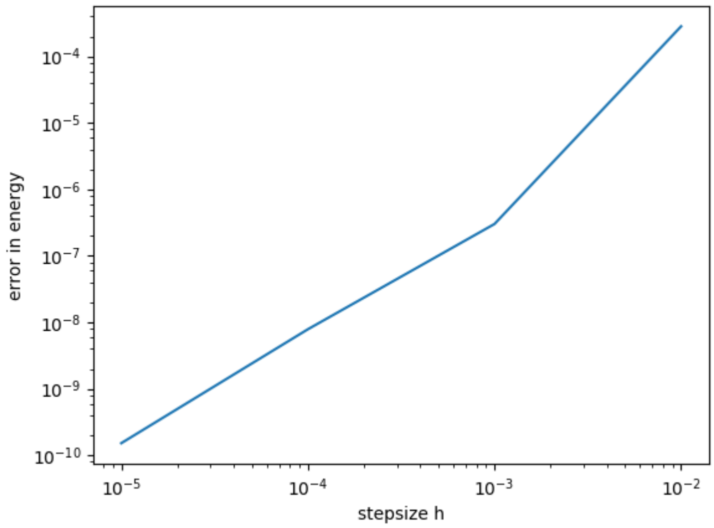

Figure 10: Second orbit energy error with leapfrog algorithm

For the third orbit we chose:

Listing 8: Exercise02b.py

```
#do the same for the third orbit with again another initial velocity
S, V, E = leapfrog(np.array([0.8, 0]), 0.001)

plt.figure(figsize = (6.5, 10))
plt.plot(S[:, 0], S[:, 1])
plt.xlabel('x')
plt.ylabel('y')
plt.show()

energyerror = []
H = [0.01, 0.001, 0.0001, 0.0001, 0.00001]
for h in H:
    S, V, E = leapfrog(np.array([0.8, 0]), h)
    energyerror.append((abs(E[-1] - E[0])) / (abs(E[0])))

plt.plot(H, energyerror)
plt.xscale('log')
plt.yscale('log')
plt.xlabel('stepsize h')
plt.ylabel('error in energy')
plt.show()
```

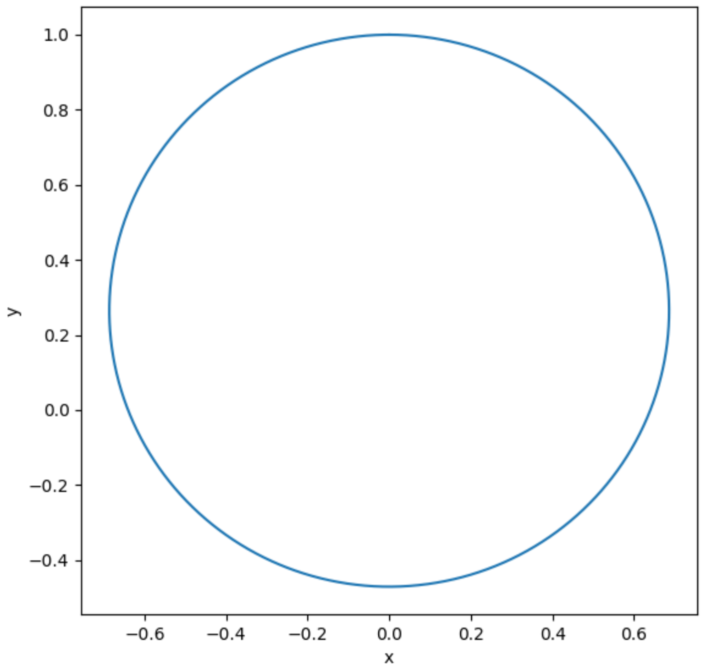

Figure 11: Third orbit with leapfrog algorithm

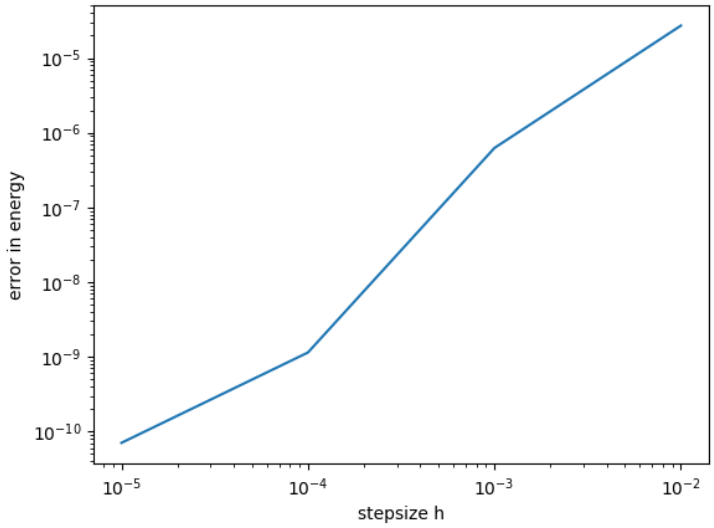

Figure 12: Third orbit energy error with leapfrog algorithm

The errors decreas with decreasing time steps for both integration schemes. For the euler algorithm we expect a decreasing with $O(h^2)$, for the Leapfrog algorithm $O(h^3)$. This is what we get. The errors of the Leapfrog integration are much smaller than those of the Euler integration. However, the slope of the errorr - curve is steeper which we also expect. Another difference between the error - plots, of the two integration variations, is, that for the Euler integration we get straight lines whereas for the Leapfrog integration the errors seem to scatter around the expected line.