Introduction to Computational Physics - Exercise 2

Simon Groß-Bölting, Lorenz Vogel, Sebastian Willenberg

8. April 2020

In the first exercise form the worksheet we woked on the following code. In the following exercise we will have a look at the errors of the euler scheme.

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

class Body:
    def __init__(self,mass,position,velocity):
        self.mass = mass              # mass of the body
        self.position = position      # inital position vector
        self.velocity = velocity      # inital velocity vector

def total_energy(s,w):
    ''' Function to compute the total energy of the system '''
    energy = np.zeros(np.shape(s)[0])
    for i in range(0,np.shape(s)[0]):
        energy[i] = (0.5*np.linalg.norm(w[i])**2)-(1/np.linalg.norm(s[i]))
    return energy

def angular_momentum(s,w):
    angular_momentum = np.zeros((np.shape(s)[0],3))
    for i in range(0,np.shape(s)[0]):
        angular_momentum[i] = np.cross(s[i],w[i])
    return angular_momentum

def laplace_runge_lenz(s,w):
    LRL = np.zeros((np.shape(s)[0],3))
    for i in range(0,np.shape(s)[0]):
        LRL[i] = np.cross(w[i], np.cross(s[i],w[i]))-s[i]
    return LRL

def eccentricity(LRL):
    ''' Function to compute the eccentricity
        from the Laplace-Runge-Lenz vector '''
    return np.linalg.norm(LRL, axis=1)

def relative_error(energy):
    rel_error = np.zeros(np.shape(s)[0])
    for i in range(0,np.shape(s)[0]):
        rel_error[i] = abs(energy[i]-energy[0])/abs(energy[0])
    return rel_error


def two_body_problem(body1,body2,G,dt,N):
    ''' Numerical Simulation of the Two-Body Problem: This function computes
        the relative motion of two point-like bodies under their mutual
        gravitational influence using a step-by-step Euler integration procedure
        Input:  two bodies of the class "Body", gravitational constant G,
                characteristic length scale R0, time steps dt and
                number of time steps N
        Output:  '''

    # compute the total mass of the two bodies and set the
    # characteristic length scale as the inital seperation
    M = body1.mass+body2.mass
```

```python
    R0 = np.linalg.norm(body1.position-body2.position)
    h = dt/np.sqrt(R0**3/(G*M))

    s = np.zeros((int(N),3))
    s[0] = (body1.position-body2.position)/R0
    print(s[0])

    w = np.zeros((int(N),3))
    w[0] = (body1.velocity-body2.velocity)/np.sqrt(G*M/R0)

    for i in range(1,int(N)):
        s[i] = s[i-1]+(w[i-1]*dt)
        w[i] = w[i-1]-(s[i-1]/np.linalg.norm(s[i-1])**3.)*dt

    return (s,w)


v0 = 1 #np.sqrt(1.*2./1.)
body1 = Body(1., np.array([1.,0.,0.]), np.array([0.,v0,0.]))
body2 = Body(1., np.array([0.,0.,0.]), np.array([0.,0.,0.]))
s, w = two_body_problem(body1, body2, 1., 1e-3, 1e4)

energy = total_energy(s,w)
rel_error = relative_error(energy)
angular_momentum = angular_momentum(s,w)
eccentricity = eccentricity(laplace_runge_lenz(s,w))


# Plot the results
fig, axs = plt.subplots(2,2,figsize=(12,8), constrained_layout=True)
fig.suptitle(r'Forward Euler Method: Numerical Simulation of the Two-Body Problem')

axs[0,0].plot(s[:,0], s[:,1], 'b.', markersize = 1, label='Body 1')
axs[0,0].plot([0], [0], 'rx', label='Body 2')
axs[0,0].set_title(r'Orbits')
axs[0,0].set_xlabel(r'$x$-axis'); axs[0,0].set_ylabel(r'$y$-axis')
axs[0,0].axis('equal')

axs[0,1].plot(range(0,len(eccentricity)), eccentricity, 'b.', markersize = 1, label='Eccentricity')
axs[0,1].set_title(r'Eccentricity $\vert\,\vec{e}_i\,\vert=\vert\,\vec{w}_i\times (\vec{s}_i\times\vec{w}_i)-\vec{s}_i\,\vert$ of the Orbit')
axs[0,1].set_xlabel(r'time step $i$'); axs[0,1].set_ylabel(r'eccentricity $\vert\,\vec{e}_i\,\vert$')

axs[1,0].plot(range(0,len(energy)), energy, 'b.', markersize = 1, label='Total Energy')
axs[1,0].set_title(r'Total Energy $E_i=(w_i^2\,/\,2)-(1/s_i)$')
axs[1,0].set_xlabel(r'time step $i$'); axs[1,0].set_ylabel(r'total energy $E_i$')

axs[1,1].plot(range(0,len(rel_error)), rel_error, 'b.', markersize = 1, label='Relative Error')
axs[1,1].set_title(r'Relative Error $\epsilon_i(h)=\vert\,E_i-E_0\,\vert\,/\,\vert\,E_0\,\vert$ in the Total Energy $E_i$')
axs[1,1].set_xlabel(r'time step $i$'); axs[1,1].set_ylabel(r'relative error $\epsilon_i$')

for ax in fig.get_axes():
    ax.grid(); ax.legend(loc='best')

fig.savefig('figures/Two-Body-Problem.pdf', format='pdf', dpi=250)
plt.show(); plt.clf(); plt.close()
```

2 Error Analysis of Euler Scheme

(a) After varying the inital velocity three times we can see that the duble logarithmic plot of the function is always a straight line with the same slope but different starting points. Therefore all the lines are parallel. This differs quite a lot from what we would expect. The law of conservation of energy predicts, that the energy in the system should remain constant. But we actually see that the energy in the system increases.

(b) After implementing the Leapfrog scheme we can see, that the energy remains constant for initial velocities that are smaller or equal to $v_0 = 2\sqrt{\frac{GM}{R_0}}$. This means that the conservation of enrgy applies here. If we pllot the motion in this case we can see that the circular remains equidistant to the center. But if we look at initial velocities that are greater than $v_0 = 2\sqrt{\frac{GM}{R_0}}$ we can see that the Leapfrog scheme also deviates from the constant energy. If we look at the plot we can see that it actually decreases.