Exercise 1: Two particles

In this exercise we are going to animate the movement of two charged particles.

a) Write down Coulomb's law.

Solution.

$$\mathbf{F} = \frac{1}{4\pi\epsilon_0}\frac{q_1 q_2}{r^2}\hat{\mathbf{r}} \tag{1}$$

b) Code a function that takes the postions and charges of two particles, and then returns the force between them.

Solution.

```
def F(q1, q2, r1, r2):
    """
    This function takes in the positions and charges of two particles,
    and returns the force.
    """
    r21 = r1 - r2
    R21 = np.linalg.norm(r21, axis=0)
    F21 = 1/(4*np.pi*eps0)*q1*q2*r21/R21**3
    return F21
```

c) Using your numeric method of choice, simulete how the particles interact. It can be a good idea to set $\epsilon_0 = 1$. Try for different initial conditions, plot the result and see if it makes sence.

Hint. You can try the initial conditions:

```
r0_1 = np.array([0, 0])
r0_2 = np.array([-1, -1])
v0_1 = np.array([0, 0])
v0_2 = np.array([2, 1.5])
Q1 = 1
Q2 = 3
m1 = 0.2
m2 = 0.1
eps0 = 1
dt = 1e-4
tstart = 0
tslutt = 2
```

Solution. Expanding on the code over, I wrote the simulation using Euler-Chromer.

```
#Define values we are going to use
r0_1 = np.array([0, 0])
r0_2 = np.array([-1, -1])
v0_1 = np.array([0, 0])
v0_2 = np.array([2, 1.5])
Q1 = 1
Q2 = 3
m1 = 0.2
m2 = 0.1
eps0 = 1
dt = 1e-4
tstart = 0
tslutt = 2
N = int((tslutt - tstart)/dt)
#Define som arrays we are going to fill
pos1 = np.zeros((2, N))
pos2 = np.zeros((2, N))
vel1 = np.zeros((2, N))
vel2 = np.zeros((2, N))
#Set initial conditions
pos1[:, 0] = r0_1
pos2[:, 0] = r0_2
vel1[:, 0] = v0_1
vel2[:, 0] = v0_2
for i in range(N-1):
    """
    Using Euler-Chromer
    """
    a1 = F(Q1, Q2, pos1[:, i], pos2[:, i])/m1
    a2 = -a1*m1/m2
    vel1[:, i+1] = a1*dt + vel1[:, i]
    vel2[:, i+1] = a2*dt + vel2[:, i]
    pos1[:, i+1] = vel1[:, i+1]*dt + pos1[:, i]
    pos2[:, i+1] = vel2[:, i+1]*dt + pos2[:, i]
plt.plot(pos1[0], pos1[1], color='r', \label='particle1')
plt.plot(pos2[0], pos2[1], color='b', \label='particle2')
plt.x\label('Posisjon i x-retning')
plt.y\label('Posision i y-retning')
plt.legend(loc=1)
plt.show()
```

d) Now that you know the positions it can be a good idea to animate the particles to get a better look at the result. Animate the motion of the particles, does the animation agree with your expectations? Try for different initial conditions.

Hint. You can use matplotlib to plot the animation.

```
from matplotlib import animation
```

Solution. Expanding the code we have written so far.

```
#Creating the figure we are going to use
fig = plt.figure()
ax = plt.axes(xlim=(-2, 2), ylim=(-2, 2))
#The two particles
particle1, = ax.plot([], [], 'ro', \label='Particle1')
particle2, = ax.plot([], [], 'bo', \label='Particle2')
points = [particle1, particle2]
def init():
    #Init function
    ax.clear
    return points
#This adjusts the speed of the animation
speed = 20
def animate(i):
    #The function that animates
    ax.set_title('Tid = %f' % (i*dt*speed))
    points[0].set_data(pos1[:, i*speed])
    points[1].set_data(pos2[:, i*speed])
    return points
#Plotting
anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=int(N/speed), interval=1, blit=False)
plt.x\label('Position in x-direction')
plt.y\label('Position in y-direction')
plt.legend(loc=1)
plt.show()
```