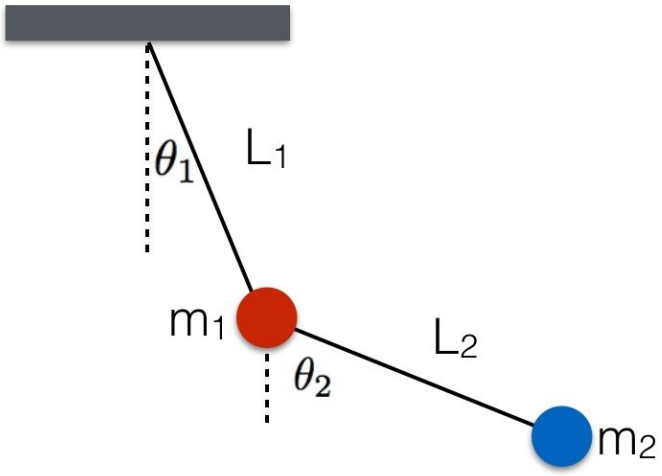
PHYS 3790 Challenge 1
Double Pendulum

X.J.J. Xu

Sept 10, 2018

Make a code to simulate the motion of a double pendulum. Discuss the trajectories as a function of initial conditions using phase space trajectories (velocity vs. position, and angular velocity vs. angle). Plot and discuss the Lyapunov exponent and the Poincare map of the double pendulum.



We can write the positions of the two masses as (x_1, y_1) and (x_2, y_2). Let the length of the connecting rods be l_1 and l_2. Let the angles between the rods and vertical axis be θ_1 and θ_2. Beware that $\dot{\theta}$ and ω are used interexchangeably to represent angular velocity.

$$x_1 = l_1 sin\theta_1$$

$$y_1 = -l_1 cos\theta_1$$

$$x_2 = x_1 + l_2 sin\theta_2$$

$$y_2 = y_1 - l_2 cos\theta_2$$

The corresponding time derivatives are:

$$\dot{x}_1 = \dot{\theta}_1 l_1 cos\theta_1$$

$$\dot{y_1} = \dot{\theta}_1 l_1 sin\theta_1$$
$$\dot{x_2} = \dot{\theta}_1 l_1 cos\theta_1 + \dot{\theta}_2 l_2 cos\theta_2$$
$$\dot{y_2} = \dot{\theta}_1 l_1 sin\theta_1 + \dot{\theta}_2 l_2 sin\theta_2$$

The easiest to solve the system is probably using a Lagrangian. The masses are m_1 and m_2. And the energies of the whole system would be:

$$T = \frac{1}{2} m_1 \dot{v_1}^2 + \frac{1}{2} m_2 \dot{v_2}^2$$
$$V = m_1 g y_1 + m_1 g y_2$$

Apply corresponding expressions of position to obtain the Lagrangian.

$$L = T - V$$
$$L = \frac{1}{2} m_1 \dot{v_1}^2 + \frac{1}{2} m_2 \dot{v_2}^2 - m_1 g y_1 - m_1 g y_2$$
$$L = \frac{1}{2}(m_1+m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 cos(\theta_1 - \theta_2) + (m_1+m_2) g l_1 cos(\theta_1) + m_2 g l_2 cos(\theta_2)$$

By using the Euler-Lagrange equation for both θ_1 and θ_2, we can obtain a pair of coupled second order differential equations. And this pair of equation of motion can be rearranged to a system of first order equations.[1] Let $\delta = \theta_1 - \theta_2$ and for simplicity, let $l_1 = l_2 = m_1 = m_2 = 1$, making it dimensionless.

$$\omega_1 = \dot{\theta}_1$$
$$\omega_2 = \dot{\theta}_2$$
$$\dot{\omega}_1 = \frac{\omega_1^2 sin(2\delta) + 2\omega_2^2 sin\delta + 2g cos\theta_2 sin\delta + 2g sin\theta_1}{-2 - 2sin^2\delta}$$
$$\dot{\omega}_2 = \frac{\omega_2^2 sin(2\delta) + 4\omega_1^2 sin\delta + 4g cos\theta_1 sin\delta}{2 + 2sin^2\delta}$$

1 Velocity vs position

We can make some plots with the following code. Parts of it is imported from the matplotlib library.[2]

```
%matplotlib inline
from numpy import sin, cos
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate

tmin = 0
```

[1] *Classical dynamics of particles and systems* Stephen Thornton (Academic Press, 1965)
[2] *Matplotlib examples* https://matplotlib.org (2018)

```
tmax = 50
G = 9.8
L1 = L2 = 1.0
M1 = M2 = 1.0
dt = 0.001
t = np.arange(tmin, tmax, dt)

# initial conditions
th1 = 0.0
w1 = 0.0
th2 = 90.0
w2 = 460.0
# initial state
state = np.radians([th1, w1, th2, w2])

def ddt(state, t):
    dydx = np.zeros_like(state)
    dydx[0] = state[1]
    dif = state[2] - state[0]
    den1 = (M1 + M2)*L1 - M2*L1*cos(dif)*cos(dif)
    dydx[1] = (M2*L1*state[1]*state[1]*sin(dif)*cos(dif) +
               M2*G*sin(state[2])*cos(dif) +
               M2*L2*state[3]*state[3]*sin(dif) -
               (M1 + M2)*G*sin(state[0]))/den1
    dydx[2] = state[3]
    den2 = (L2/L1)*den1
    dydx[3] = (-M2*L2*state[3]*state[3]*sin(dif)*cos(dif) +
               (M1 + M2)*G*sin(state[0])*cos(dif) -
               (M1 + M2)*L1*state[1]*state[1]*sin(dif) -
               (M1 + M2)*G*sin(state[2]))/den2
    return dydx

y = integrate.odeint(ddt, state, t)
x1 = L1*sin(y[:, 0])
y1 = -L1*cos(y[:, 0])
x2 = L2*sin(y[:, 2]) + x1
y2 = -L2*cos(y[:, 2]) + y1

# Plot
fig, ax = plt.subplots()
fig.set_size_inches(12,8)
ax.plot(x1, y1, label='x1 vs y1')
ax.plot(x2, y2, label='x2 vs y2')
plt.title('Position vs velocity')
```

```
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
# plt.axvline(x=0.5, linestyle='--')
legend = ax.legend()
plt.show()
```

We can play with different initial conditions and make plots. If we just give the lower bob a small push, as one would expect, nothing very interesting would happen as shown in Figure 1 - the bobs hover at the same height, barely moving.

Now if we start the lower bob at 90 degrees and give it a bigger push, there will be some periodic swinging - see Figure 2. Interestingly, varying angular velocity of the lower bob from 1 to 200 will result in paths that are very similar to the paths shown in Figure 2.

However, if we initiate the lower bob, still at 90 degrees, with a initial angular velocity of 200, then the pattern of the paths will drastically change into what is shown in Figure 3. It is still periodic but the lower bob swing with much greater vertical amplitude.

Finally, giving the lower bob an initial angular velocity of 460 will induce chaotic behavior in the pendulum as shown in Figure 4. Also notice that the upper pendulum covers a complete elliptical orbit, which is not the case for the previous initial conditions.

2 Angular velocity vs angle

We can make plots with the following code:

```
# Plot
fig, ax = plt.subplots()
fig.set_size_inches(12,8)
ax.plot(y[:,0], y[:,1], label='theta1 vs omega1')
ax.plot(y[:,2], y[:,3], label='theta2 vs omega2')
plt.title('Angle vs angular velocity')
plt.xlabel('Theta')
plt.ylabel('Omega')
plt.grid()
legend = ax.legend()
plt.show()
```

We will use the same set of initial conditions as the first section, producing Figures 5-8. In Figure 5 and Figure 6, orbits appear to be very periodic. Figure 7 displays distinct path flows - 4 loops for the lower bob and 4 for the upper bob. However, Figure 8 shows the most obvious case of chaos, which is also shown by its Lyapunov exponent.

3 Lyapunov Exponents

The Lyapunov exponent is used to provide a quantitative measurement of sensitivity. Say two nearby initial conditions are separated by d_0 and then this distance changes with time such that it is described by $d(t) = d_0 e^{\lambda t}$. The average Lyapunov exponent λ is calculated to be:[3]

$$\lambda = \frac{1}{\tau} \sum_{i=0}^{n-1} ln \left| \frac{d(t_i)}{d_0} \right|$$

Where τ is the total interval of time evaluated over.

```
# Lyapunov exponent
lya = 0
dt = 0.001
n = 50000
tau = int(n*dt)

# parameter to look at
# 0: theta1; 1: omega1; 2: theta2; 3: omega2
para = 3
# please manually edit state1 and state2 and d_0 sorry

state1 = np.radians([0, 0, 0, 10])
data1 = integrate.odeint(ddt, state1, t)
# x11 = L1*sin(data1[:, 0])
# y11 = -L1*cos(data1[:, 0])
# x12 = L2*sin(data1[:, 2]) + x11
# y12 = -L2*cos(data1[:, 2]) + y11

state2 = np.radians([0, 0, 0, 20])
data2 = integrate.odeint(ddt, state2, t)
# x21 = L1*sin(data2[:, 0])
# y21 = -L1*cos(data2[:, 0])
# x22 = L2*sin(data2[:, 2]) + x21
# y22 = -L2*cos(data2[:, 2]) + y21

d_0 = 10
for i in range(100):
    di = (180/np.pi) * (data1[i,para] - data2[i,para])
    #print(di)
    lya += np.log(abs(di)/d_0)

print('Lyapunov exponent = ', lya/tau)
```

Let us use $(\theta_1, \omega_1, \theta_2, \omega_2)$ to express the sets of initial conditions. We will use $\tau = 50$,

[3] *Chaos from Simplicity: An Introduction to the Double Pendulum* Joe Chen (University of Canterbury, 2008)

$n = 50000$ and $dt = 0.001$. The Lyapunov exponent can indicate different behaviours.
Case 1: $\lambda < 0$

Trajectories in phase space will converge, suggesting the system is dissipative. E.g. (0,0,0,10) and (0,0,0,20) will yield $\lambda_1 = -0.066$.

Case 2: $\lambda = 0$

Distances between orbits will remain unchanged.
Hard to do for the double pendulum.

Case 3: $\lambda > 0$

Orbits will diverge and this is an indicator of chaos.
E.g. (0,0,90,450) and (0,0,90,460) will yield $\lambda_2 = 0.18$.
As mentioned in section one, this set of initial conditions shows the first onset of chaotic and non-periodic behaviour.

4 Poincare maps

Poincare maps are useful for studying swirling flows, such as the flow near a periodic orbit. Consider an n-dimensional system:[4]

$$\dot{x} = f(x)$$

Let S be an n - 1 dimensional surface of section which is transverse to the flow. The map P is a mapping from S to itself. If $x_k \subset S$ denotes the kth intersection, then the Poincare map is defined by

$$x_{k+1} = P(x_k)$$

E.g. a fixed point x^* will be mapped to itself $P(x^*) = x^*$. The following code has been written to create Poincare maps:

```
def poincare(x, y):
    Sx = []
    Sy = []

    # Variables for tracking flow cycles
    sign = -1
    cycle = 0
    go = False

#     for i in range(n):
#         if np.sign(x[i]) != sign:
#             sign = np.sign(x[i])
#             cycle += 1
#         if cycle == 2:
#             Sx.append(y[i])
#             if go == True:
#                 Sy.append(y[i])
```

[4] *Nonlinear dynamics and chaos* Stephen Strogatz (CRC Press, 2014)

```
#                 go = True
#                 cycle = 0
    for i in range(n):
        if np.sign(y[i]) != sign:
            sign = np.sign(y[i])
            cycle += 1
        if cycle == 2:
            Sx.append(x[i])
            if go == True:
                Sy.append(x[i])
            go = True
            cycle = 0
    return Sx, Sy

# Sx1, Sy1 = poincare(x1, y1)
# Sx2, Sy2 = poincare(x2, y2)
Sx1, Sy1 = poincare(y[:,0], y[:,1])
Sx2, Sy2 = poincare(y[:,2], y[:,3])

# print(Sx)
# print(Sy)

# Plot Poincare map
fig, ax = plt.subplots()
fig.set_size_inches(12,8)
# plt.scatter(Sx1[0:-1], Sy1, label='x1 vs y1')
# plt.scatter(Sx2[0:-1], Sy2, label='x2 vs y2')
plt.scatter(Sx1[0:-1], Sy1, label='theta1 vs omega1')
plt.scatter(Sx2[0:-1], Sy2, label='theta2 vs omega2')
plt.title('Poincare map')
# plt.xlabel('y1')
# plt.ylabel('y1*')
# plt.xlabel('Omega')
# plt.ylabel('Omega*')
plt.xlabel('Theta')
plt.ylabel('Theta*')
plt.legend()
plt.grid()
plt.show()
```

We can choose a surface S where $x = 0$ in the phase space plot of Figure 2-4. We can choose a surface where $\theta = 0$ in the phase space of Figure 6-7. For Figure 8, it is more suitable to find the Poincare map for θ instead of ω. The corresponding Poincare maps are Figure 9-14. It is worth pointing out that Figure 11 demonstrates chaos, as presented by the other plots for initial conditions (0,0,90,460).

5 Appendix

Figure 1: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 0, \omega_2 = 1$

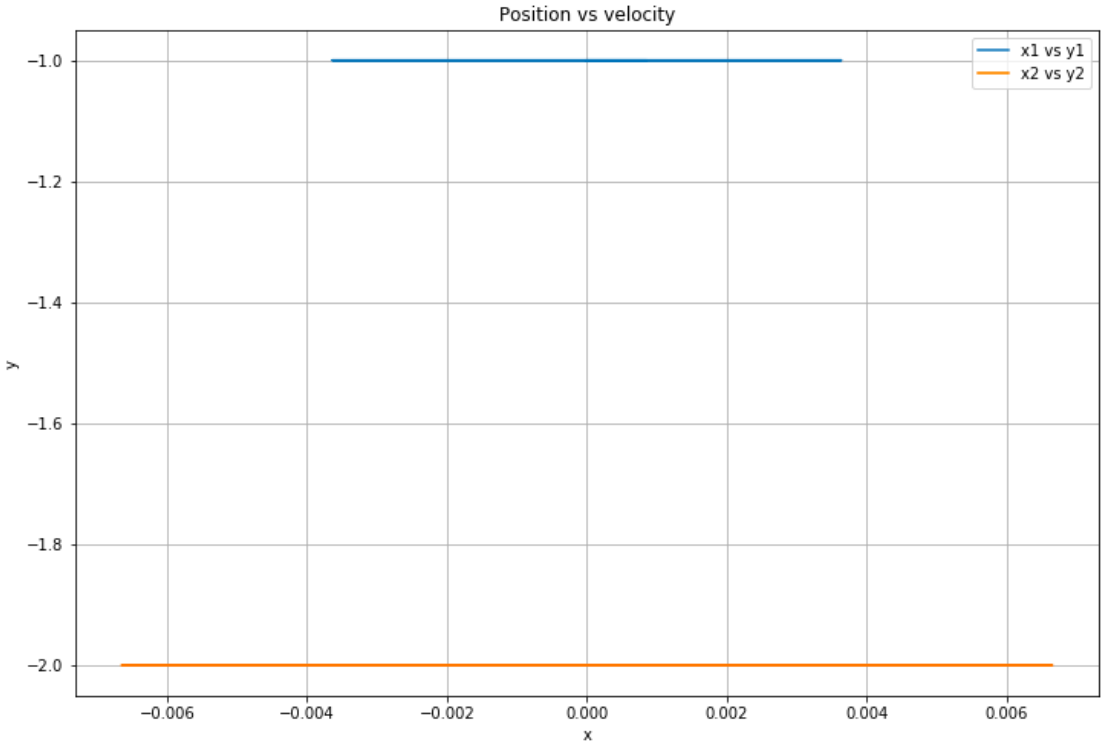


Figure 2: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 1$

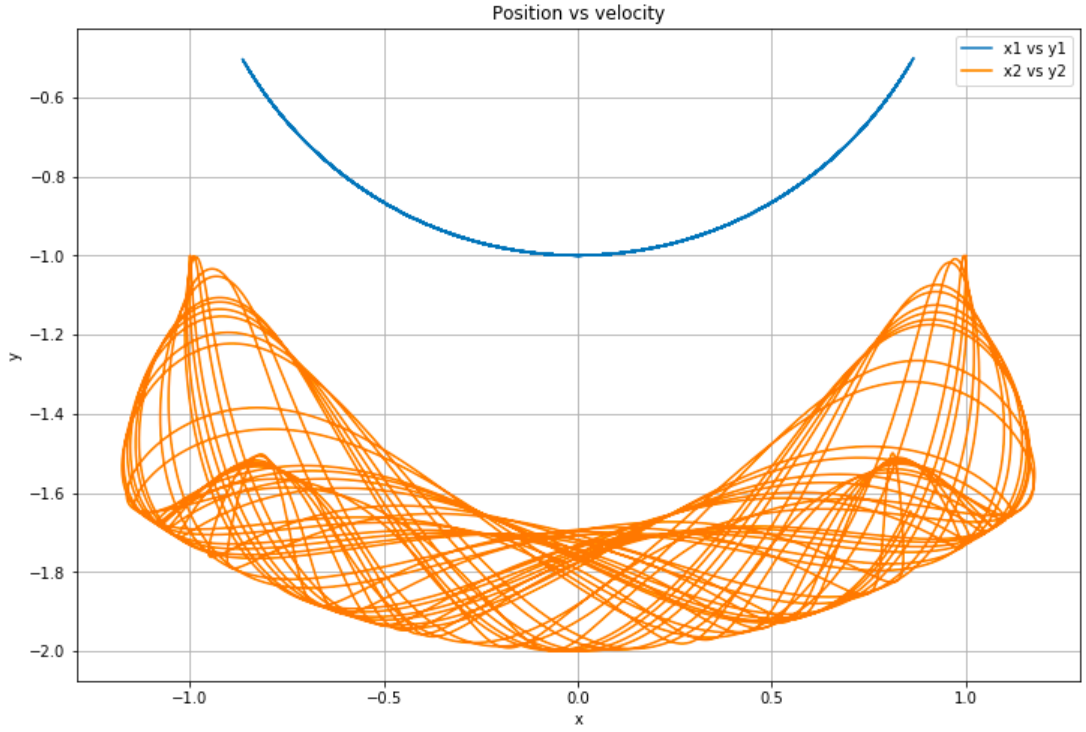


Figure 3: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 200$

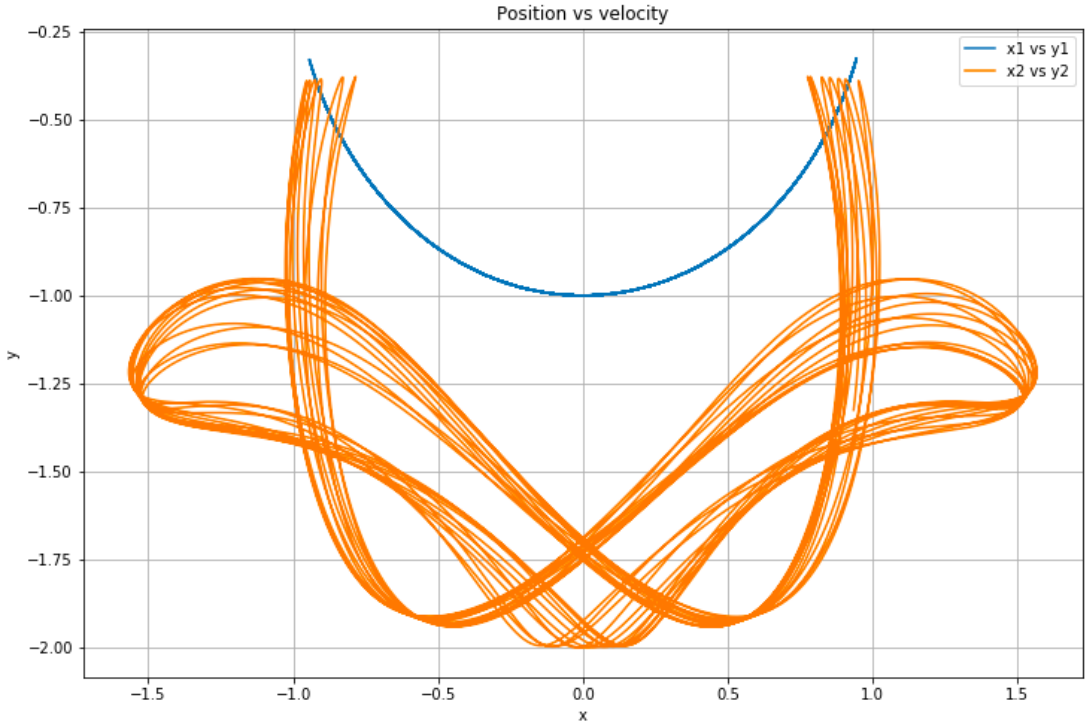


Figure 4: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 460$

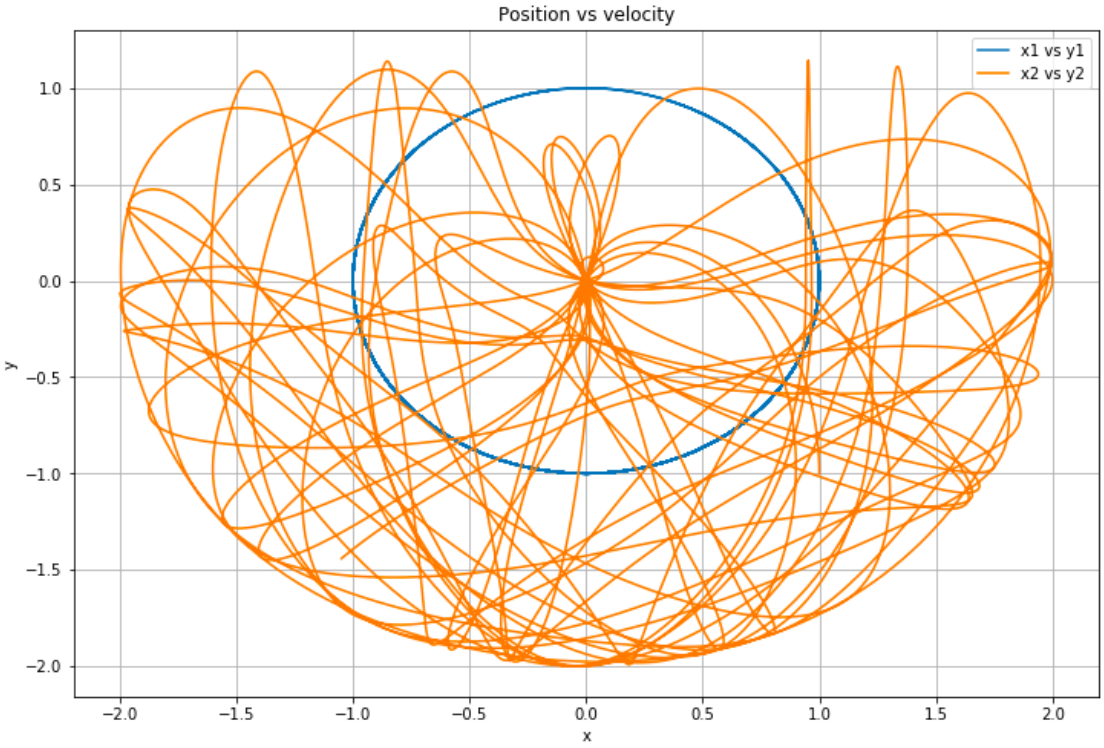


Figure 5: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 0, \omega_2 = 1$

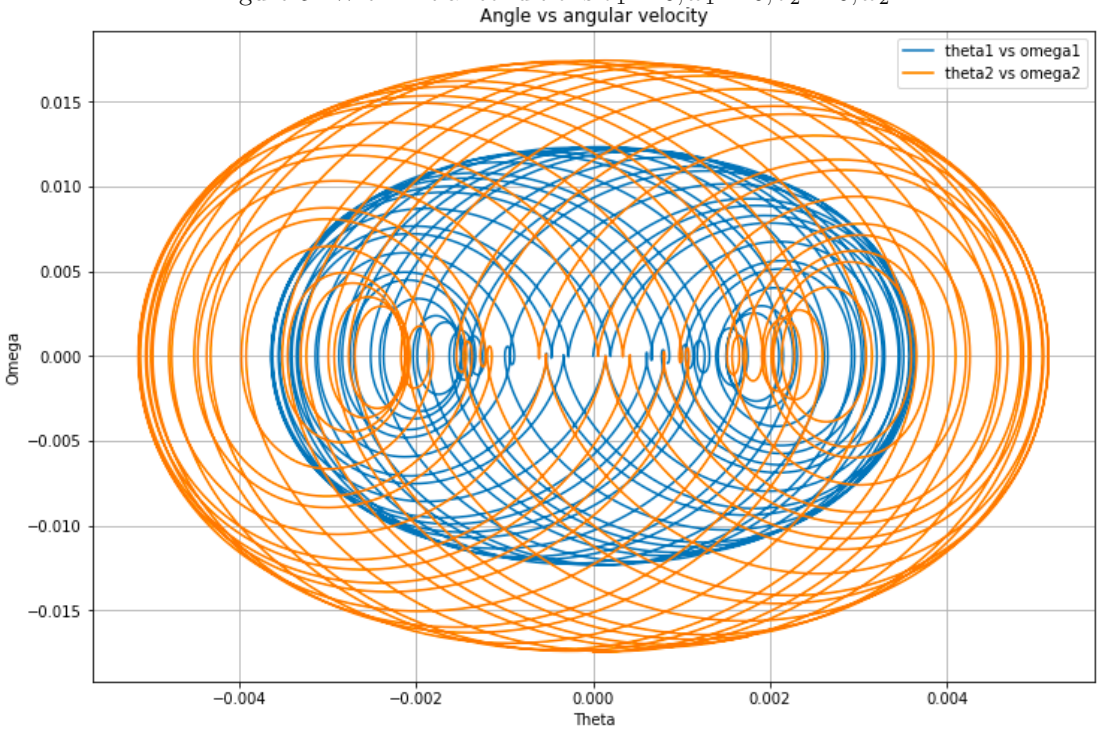


Figure 6: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 1$

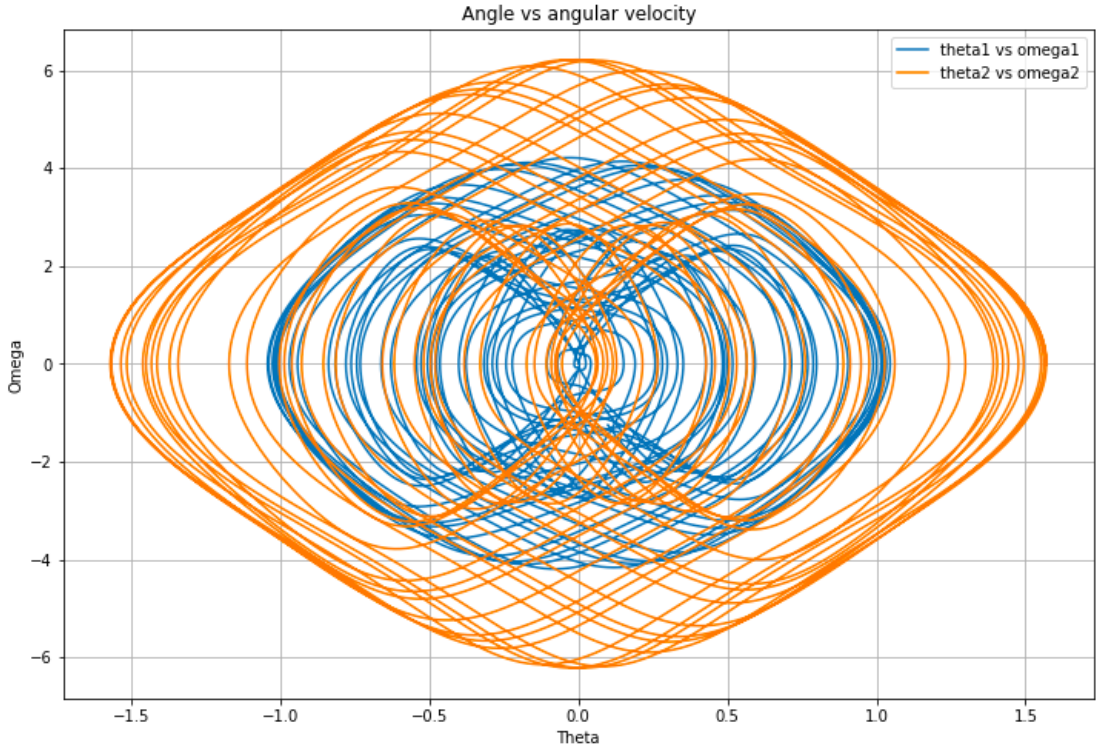


Figure 7: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 200$

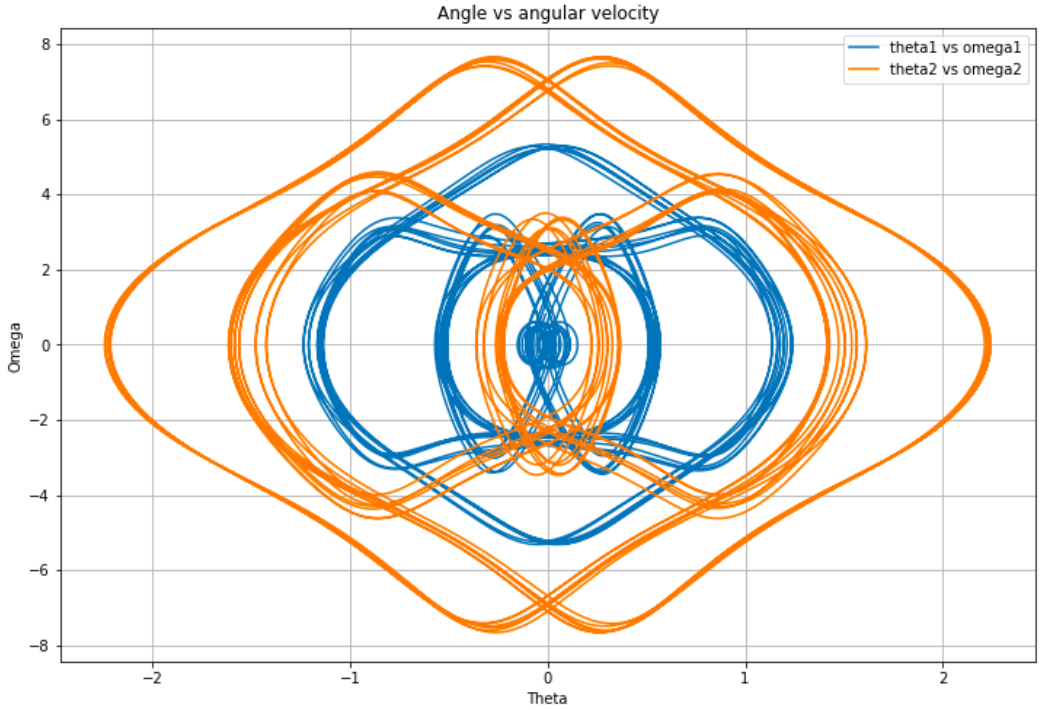


Figure 8: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 460$

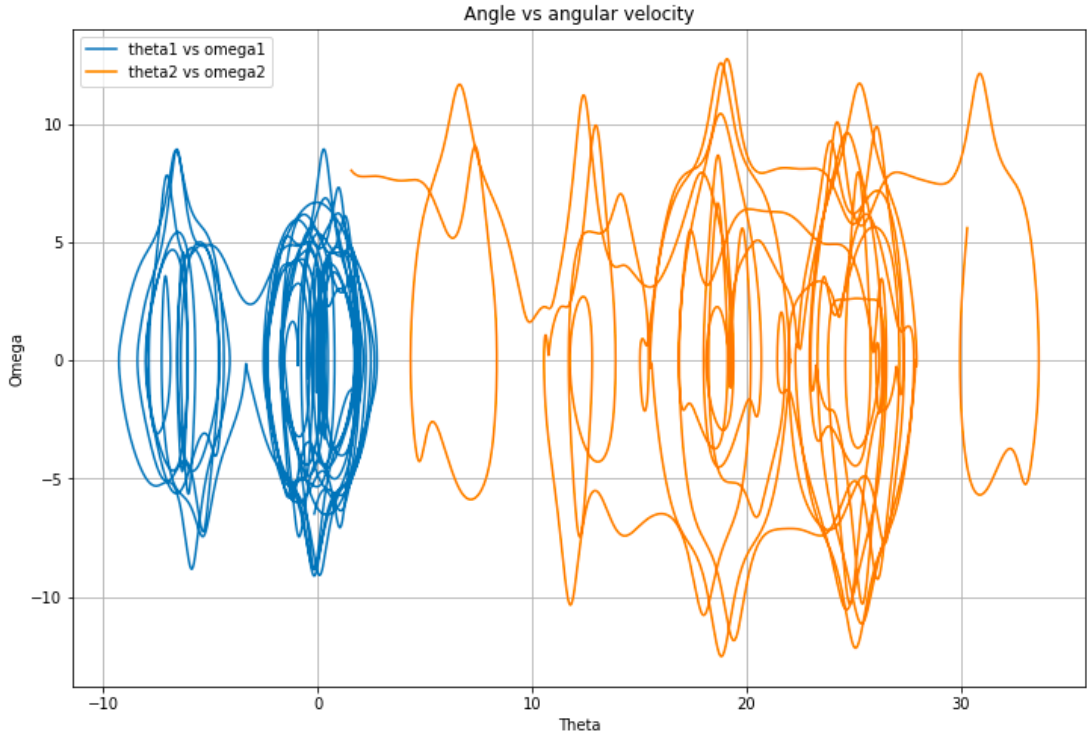


Figure 9: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 1$

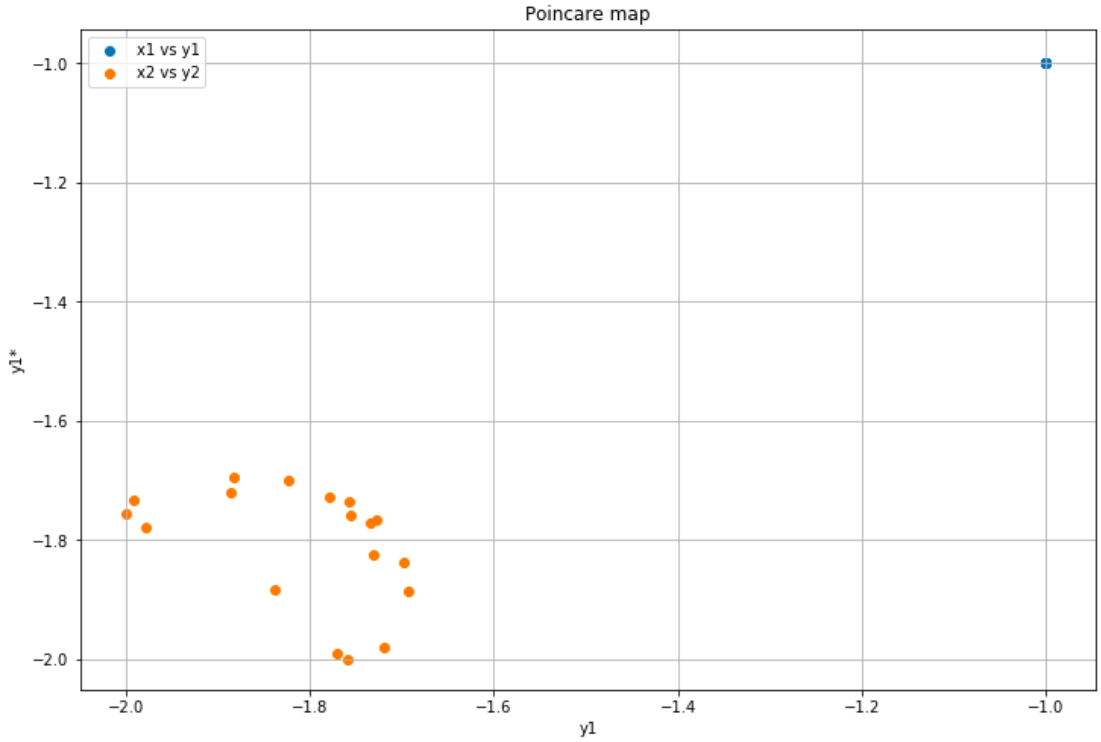


Figure 10: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 200$

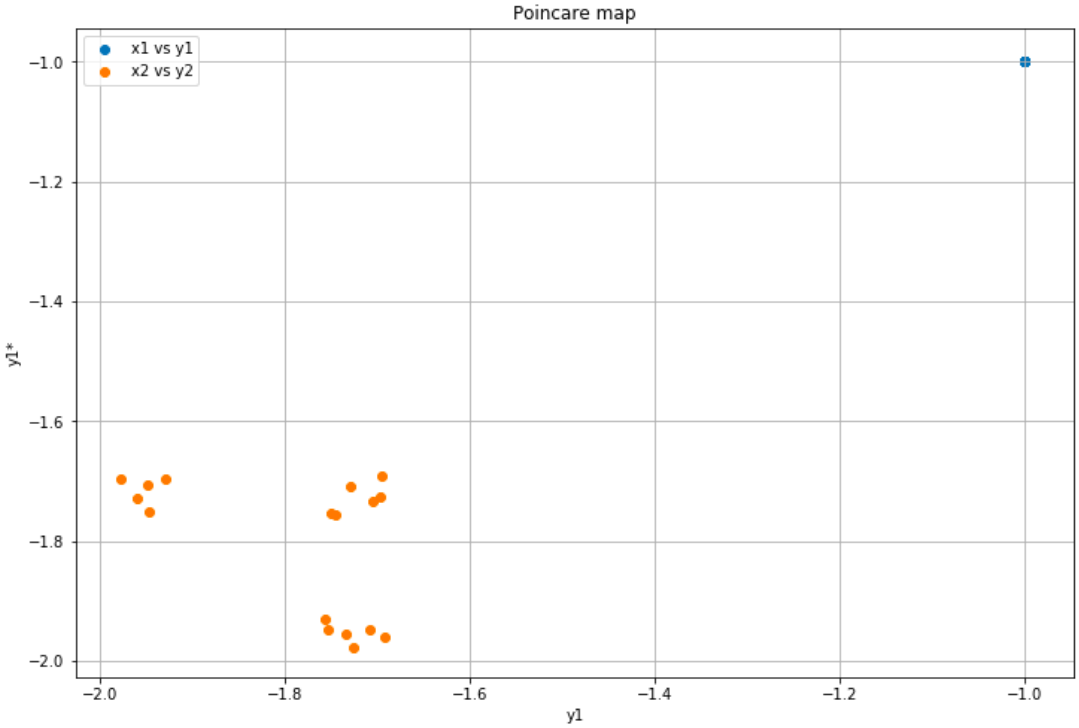


Figure 11: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 460$

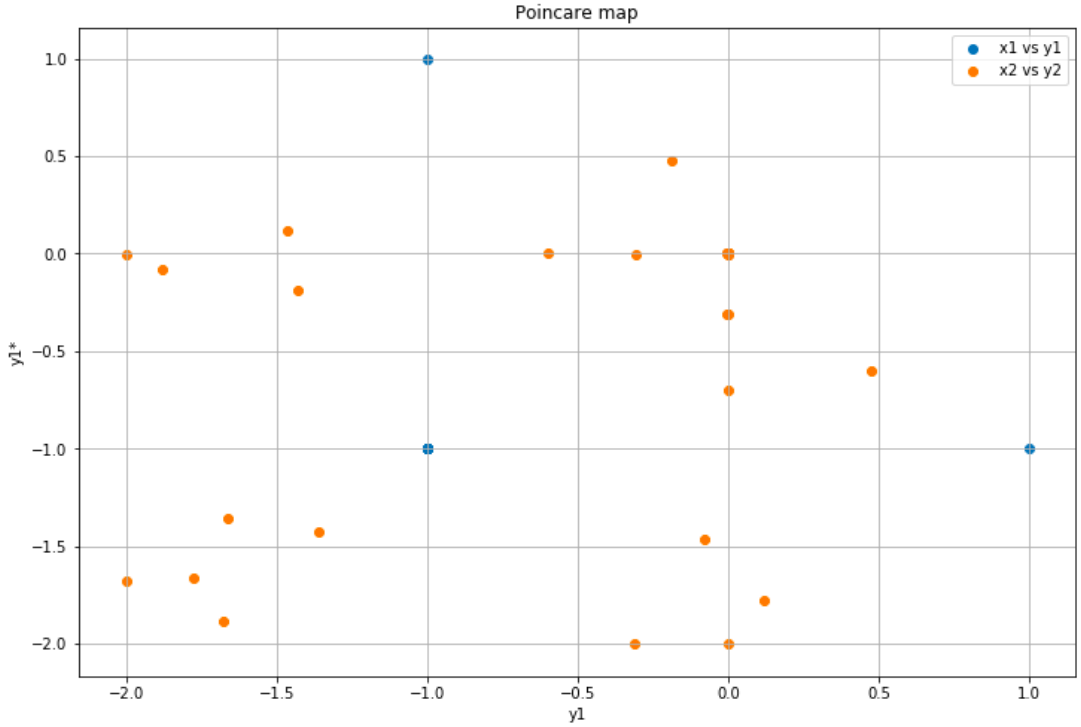


Figure 12: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 1$

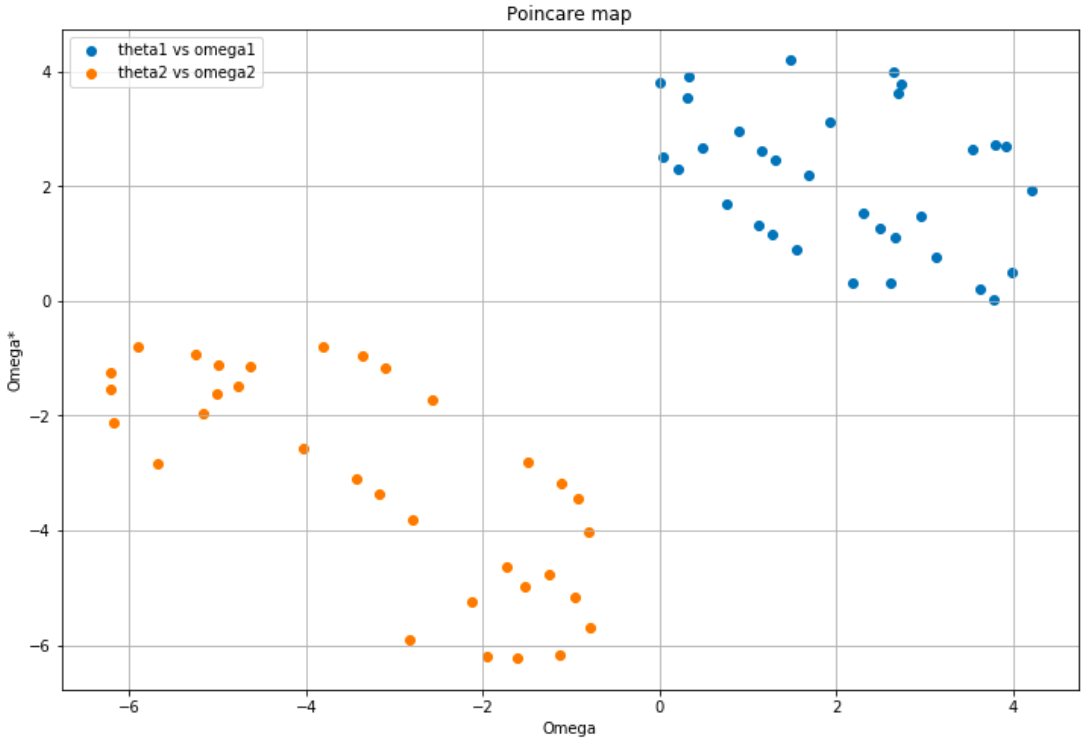


Figure 13: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 200$

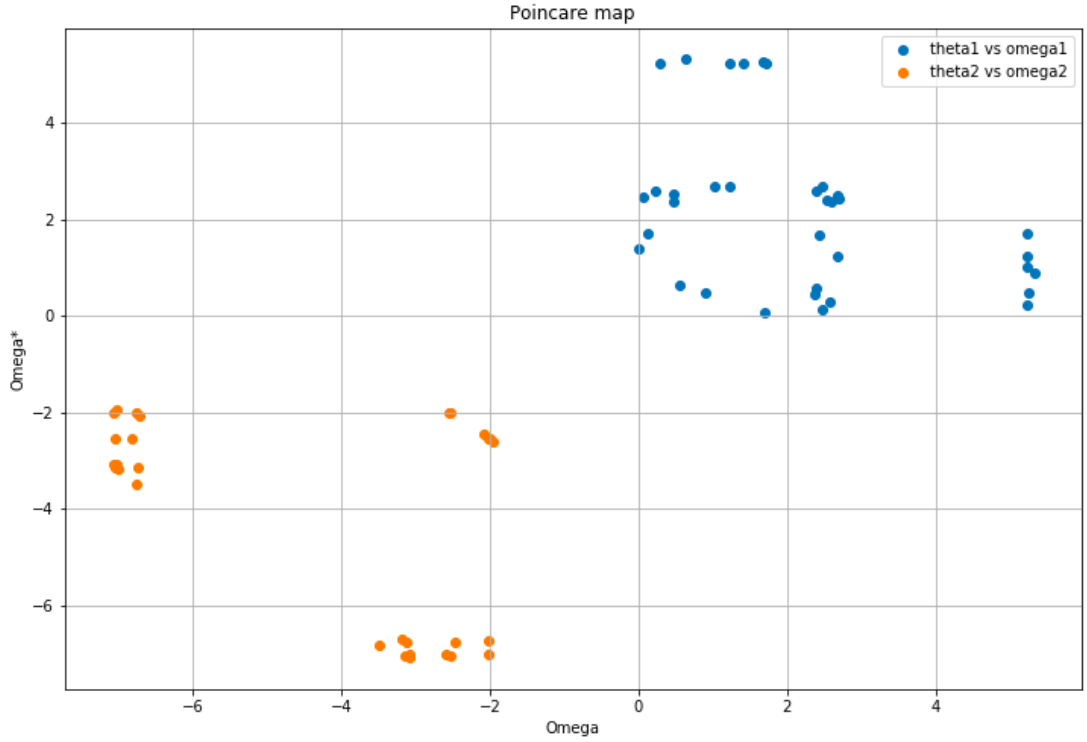


Figure 14: With initial conditions $\theta_1 = 0, \omega_1 = 0, \theta_2 = 90, \omega_2 = 460$