Simulation of A Double Pendulum

Tania Patra (20PH40055), Subhadeep Bej (20PH40048), Krishnendu Barman (20PH40020)

Introduction: In this essay we demonstrate the chaotic behaviour of a double pendulum. We will work out the lagrangian of the system, derive the equations of motion of the system using Euler-Lagrange equations. We will then calculate a solution to the second order ODEs numerically using Python's scipy.integrate pack and simulate the dynamics of the system using Python. We wil demonstrate how second order, non-linear ODEs makes the system very sensitive to the initial conditions and motion of the bob chaotic.

Derivation of equation of motions of the Double Pendulum system:

The double pendulum system is shown in the figure-1. We will denote the upper bob of mass m_1 by subscript 1 and the lower one of mass m_2 by subscript by 2. Here l denotes the length of the rod, x denotes the horizontal position of pendulum mass, y denotes the vertical position of pendulum mass and θ is the angle between pendulum and vertical reference.

By using trigonometry we can write for the positions (x_1,x_2,y_1,y_2) and angles (θ_1, θ_2), we get,

$x_1 = l_1 \sin\theta_1$ .....(1)

$y_1 = -l_1 \cos\theta_1$ .....(2)

$x_2 = x_1 + l_2 \sin\theta_2$ .....(3)

$y_2 = y_1 - l_2 \cos\theta_2$ .....(4)

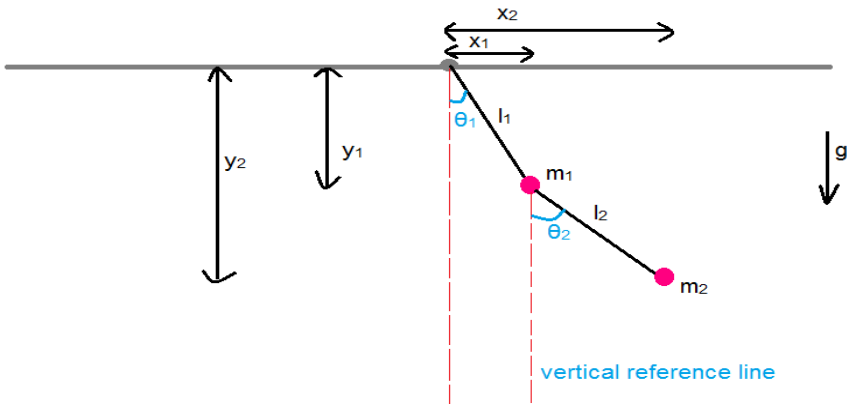


Fig 1: DOUBLE PENDULUM SYSTEM

So the kinetic energy, $T = (\frac{1}{2}).m_1.[(\dot{x}_1)^2 + (\dot{x}_2)^2] + (1/2).m_2.[(\dot{y}_1)^2 + (\dot{y}_2)^2]$

Potential energy of the system, $V = m_1 g y_1 + m_2 g y_2$

So lagrangian of the system, L = T- V

Lagrange's equation of motions are,

$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} = 0$; i=1,2.

From which we get two equations,

$(m_1 + m_2).(l_1)^2.(\ddot{\theta_2}) + m_2 l_1 l.(\ddot{\theta_1})\cos(\theta_1 - \theta_2) - m_2 l_1 l_2(\dot{\theta}_1)^2 \sin(\theta_1 - \theta_2) + (m_1 + m_2).l_1.g.\sin\theta_1 = 0$

And,

$m_2 l_2 l_2(\ddot{\theta_2}) + m_2 l_1 l.(\ddot{\theta_1})\cos(\theta_1 - \theta_2) - m_2 l_1 l_2(\dot{\theta}_1)^2 \sin(\theta_1 - \theta_2) + m_2 l_2 g.\sin\theta_2 = 0$

These two 2nd order differential equations have 4 unknowns and numerically it can not be solved. So next we will use Hamilton's Equations technique to get the 4 first order differential equations.

The generalized momentum(p_1,p_2) corresponding to the generalized coordinates θ_1, θ_2 are ,

$p_1 = \frac{\partial L}{\partial \dot{\theta}_1} = (m_1 + m_2)(l_1)^2.\dot{\theta}_1 + m_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2)$ .....(5)

$p_2 = \frac{\partial L}{\partial \dot{\theta}_2} = m_2(l_2)^2.\dot{\theta}_2 + m_2 l_1 l_2.\cos(\theta_1 - \theta_2)$ .....(6)

Hamiltonian [$=p_i\theta_i$ - L] of the system will be,

$H = (1/2).(m_1 + m_2)(l_1)^2.(\dot{\theta}_1)^2 + (1/2)m_2([l_2\dot{\theta}_2]^2 + 2l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) - (m_1+m_2)gl_1\cos\theta_1 - m_2 l_2 g.\cos\theta_2$
.....(7)

Putting the values of $\dot{\theta}_1$, $\dot{\theta}_2$ from the equations (5),(6),

$H = [m_2(l_2p_2)^2 + (m_1 + m_2).(l_2p_2)^2 - 2m_2 l_1 l_2 p_1 p_2.\cos(\theta_1 - \theta_2)] / [2m_2(l_1l_2)^2.(m_1+m_2\sin^2(\theta_1 - \theta_2))]$

$- (m_1+m_2)gl_1\cos\theta_1 - m_2 g l_2 \cos\theta_2$ .....(8)

By using Hamiltonian equations $\frac{\partial H}{\partial p_k} = \dot{\theta_k}$ and $\frac{\partial H}{\partial \theta_k} = -\dot{p_k}$ (k=1,2), we get 4 following 1st differential equations,

$\dot{\theta_1} = [l_2p_1 - l_1p_2\cos(\theta_1 - \theta_2)]/[(l_1)^2 l_2(m_1 + m_2\sin^2(\theta_1 - \theta_2))]$ .....(9)

$\dot{\theta_2} = [p_2 l_1(m_1 + m_2) - m_2 p_1 l_2\cos(\theta_1 - \theta_2)]/[(l_2)^2 l_1 m_2(m_1 + m_2\sin^2(\theta_1 - \theta_2))]$ .....(10)

$\dot{p_1} = -(m_1 + m_2)gl_1\sin\theta_1 - [p_1p_2.\cos(\theta_1 - \theta_2)]/[l_1l_2(m_1 + m_2\sin^2(\theta_1 - \theta_2))^2] + A$ .....(11)

$\dot{p_2} = -m_2 g l_2 \sin\theta_2 + [p_1p_2.\cos(\theta_1 - \theta_2)]/[l_1l_2(m_1 + m_2\sin^2(\theta_1 - \theta_2))^2] - A$ .....(12)

Where ,

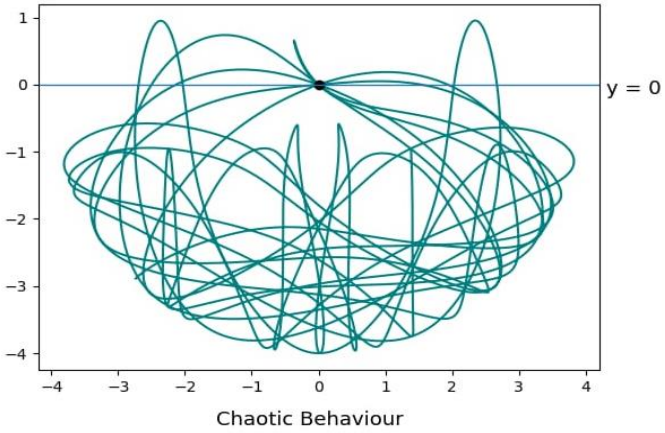
$A = \{[(l_2)^2 m_2(p_1)^2 + (m_1 + m_2).(l_1p_2)^2 - l_1l_2m_2p_1p_2.\cos(\theta_1 - \theta_2)]\sin[2(\theta_1 - \theta_2)]\} / \{2.[l_1l_2(m_1 + m_2\sin^2(\theta_1 - \theta_2))]^2\}$

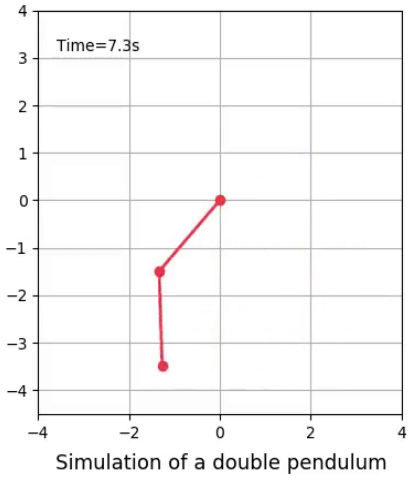
Finding a numerical solution:

In this section, we animate the dynamics of the double pendulum. Following code is written in Python.

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

G,  L1,  L2,  M1,  M2 = 9.8,  2.0,  2.0,  1.0,  1.0  #parameters in S.I units

t = np.arange(0, 40, dt)   # Time steps at which we're going to calculate theta1, theta2

# Intial Conditions
theta1 = 120.0
omega1 = 0.0
theta2 = -10.0
omega2 = 0.0

# initial state
initial_state = np.radians([theta1, omega1, theta2, omega2])  # Converting angles in degrees to radians

def derivs(state, t):
''' returns [theta1_dot/omega1, theta1_double_dot, theta2_dot/omega2, theta2_double_dot] '''

d = np.zeros_like(state)  # array of zeroes with the same shape as of the variable 'state'
d[0] = state[1] #theta1_dot
delta = state[2] - state[0]  #theta2 - theta1
den1 = (M1+M2) * L1 - M2 * L1 * cos(delta) * cos(delta)

 # Writing the eqn for theta1_doubledot
 d[1] = ((M2 * L1 * state[1] * state[1] * sin(delta) * cos(delta)  + M2 * G * sin(state[2]) * cos(delta)
             + M2 * L2 * state[3] * state[3] * sin(delta)  - (M1+M2) * G * sin(state[0]))
            / den1)

d[2] = state[3] #theta2_dot
den2 = (L2/L1) * den1

# Writing  the eqn for theta2_doubledot
d[3] = ((- M2 * L2 * state[3] * state[3] * sin(delta) * cos(delta) + (M1+M2) * G * sin(state[0]) * cos(delta)
             - (M1+M2) * L1 * state[1] * state[1] * sin(delta) - (M1+M2) * G * sin(state[2])) / den2)
return d

# integrating ODE using scipy.integrate.
y = integrate.odeint(derivs, initial_state, t)
```

```
x1 = L1*sin(y[:, 0])   # Values of x1 for each solution of theta1
y1 = -L1*cos(y[:, 0])  # Values of y1

x2 = L2*sin(y[:, 2]) + x1
y2 = -L2*cos(y[:, 2]) + y1

fig, ax = plt.subplots(subplot_kw= {'xlim' : (-4, 4), 'ylim' :(-4.5, 4)})
ax.grid()
ax.set_aspect('equal')

line, = ax.plot([], [], 'o-', lw=2, color='#e63946')   # drawing a line
time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)
line.set_data([], [])
time_text.set_text('')


def animate(i):
    x_value = [0, x1[i], x2[i]]    # x-values of origin, bob of mass M1, bob of mass M2
    y_value = [0, y1[i], y2[i]]    # y-values of origin, bob of mass M1, bob of mass M2

    line.set_data(x_value, y_value)
    time_text.set_text( f'Time={i*dt:0.1f}s')
    return line, time_text


ani = animation.FuncAnimation(fig, animate, range(1, len(y), 2),
                              interval=dt*700,  blit=True)
ax.annotate(s = "Simulation of a double pendulum", xy=(0.28, 0.05), xycoords='figure fraction',
fontsize=13);
plt.subplots_adjust(bottom=0.15)
plt.show()
# ani.save('simulation_double_pendulum.mp4', fps=120) # Saving the animation


# Plotting the trajectory of the bob of mass M2

fig, ax = plt.subplots()
ax.plot(x2, y2, 'teal')
ax.plot(0, 0, 'o', ms = 6, color='black')
ax.axhline(0,  linewidth=1)
ax.annotate(s = "Chaotic Behaviour", xy=(0.37, 0.04), xycoords='figure fraction', fontsize=13)
ax.annotate(s = "y = 0", xy=(0.91, 0.70), xycoords='figure fraction', fontsize=13)
plt.subplots_adjust(bottom = 0.15)
plt.show()
# plt.savefig('chaotic behaviour.png')
```





Animation can be found at: [Simulation of a double pendulum]

Conclusion:

From the trajectories of the double pendulum system it is evident that the double pendulum is a complex system. When displacements from equilibrium are large the system becomes dramatically chaotic in its motion and demonstrate the fact that deterministic systems are not necessarily predictable. However, as shown, the behaviour of the system can be modeled to a very high degree of accuracy. It is concluded that the double pendulum system exhibits rich dynamic behavior with strong sensitivity to initial conditions.

References :

1) " Course of theoretical physics" by L.D.Landau and E.M.Lifshitz, page :1 to 11

2) Code used can be found in this repository.
3) https://en.m.wikipedia.org/wiki/Double_pendulum
4) Abdalftah Elbori and Ltfei Abdalsmd , Simulation of double pendulum . Journal of Software Engineering and Simulation, Volume 3~ Issue 7(2017) pp: 01-13