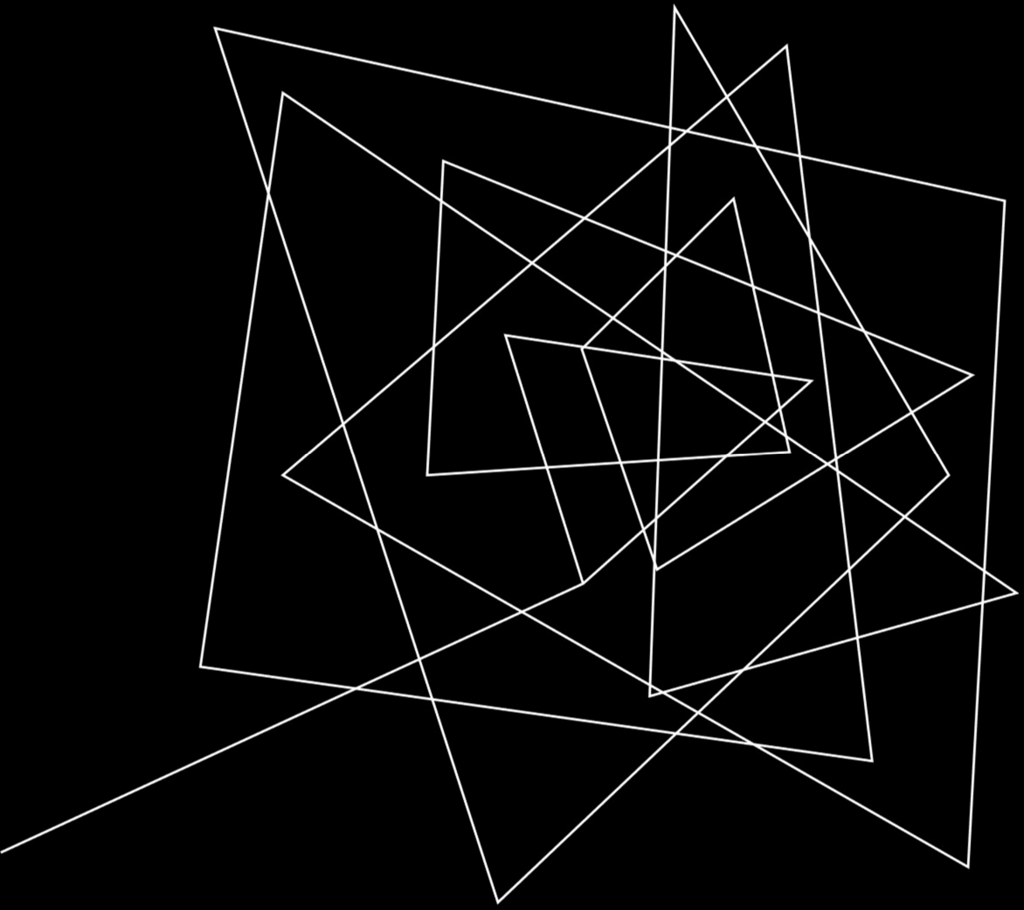


VIRTUAL WORK METHOD & PROJECTIONS

PARAMETERS

Parameter	Magnitude (units)
L1, L2, L3	1
W1, W2, W3	0.1
M1, M2, M3	1
J1_1, J1_2, J2_,1, J2_2, J3_1, J3_2	0
J1_3	(1/12) * M1 * (L1^2 + W1^2)
J2_3	(1/12) * M2 * (L2^2 + W2^2)
J3_3	(1/12) * M3 * (L3^2 + W3^2)
Tau_1, Tau_2, Tau_3	Our forced inputs!
K – spring constant	Variable
B – damping constant	variable

BASIC DEFINITIONS

1. Defining the position vectors for center of mass in respective frames.

rc_1, rc_2, rc_3 = $\begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix}$

2. Find the rotation matrices around z – axis for each transformation.

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. Find the angular velocities with respect to each frame.

Omega01_11 = $\begin{bmatrix} 0 & -\sin^2(\theta_1(t))\frac{d}{dt}\theta_1(t) - \cos^2(\theta_1(t))\frac{d}{dt}\theta_1(t) & 0 \\ \sin^2(\theta_1(t))\frac{d}{dt}\theta_1(t) + \cos^2(\theta_1(t))\frac{d}{dt}\theta_1(t) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Omega02_22 = $\begin{bmatrix} 0 & -\frac{d}{dt}\theta_1(t) - \frac{d}{dt}\theta_2(t) & 0 \\ \frac{d}{dt}\theta_1(t) + \frac{d}{dt}\theta_2(t) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Omega03_33 = $\begin{bmatrix} 0 & -\frac{d}{dt}\theta_1(t) - \frac{d}{dt}\theta_2(t) - \frac{d}{dt}\theta_3(t) & 0 \\ \frac{d}{dt}\theta_1(t) + \frac{d}{dt}\theta_2(t) + \frac{d}{dt}\theta_3(t) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

BASIC DEFINITIONS

4. Find linear velocities

VC_0 = $$\begin{bmatrix} -0.5\sin\left(\theta_1(t)\right)\frac{d}{dt}\theta_1(t) \\ 0.5\cos\left(\theta_1(t)\right)\frac{d}{dt}\theta_1(t) \\ 0 \end{bmatrix}$$

VC_1 = $$\begin{bmatrix} -0.5\left(\frac{d}{dt}\theta_1(t)+\frac{d}{dt}\theta_2(t)\right)\sin\left(\theta_1(t)+\theta_2(t)\right)-\sin\left(\theta_1(t)\right)\frac{d}{dt}\theta_1(t) \\ 0.5\left(\frac{d}{dt}\theta_1(t)+\frac{d}{dt}\theta_2(t)\right)\cos\left(\theta_1(t)+\theta_2(t)\right)+\cos\left(\theta_1(t)\right)\frac{d}{dt}\theta_1(t) \\ 0 \end{bmatrix}$$

VC_2 =

$$\begin{bmatrix} -\left(\frac{d}{dt}\theta_1(t)+\frac{d}{dt}\theta_2(t)\right)\sin\left(\theta_1(t)+\theta_2(t)\right)-0.5\left(\frac{d}{dt}\theta_1(t)+\frac{d}{dt}\theta_2(t)+\frac{d}{dt}\theta_3(t)\right)\sin\left(\theta_1(t)+\theta_2(t)+\theta_3(t)\right)-\sin\left(\theta_1(t)\right)\frac{d}{dt}\theta_1(t) \\ \left(\frac{d}{dt}\theta_1(t)+\frac{d}{dt}\theta_2(t)\right)\cos\left(\theta_1(t)+\theta_2(t)\right)+0.5\left(\frac{d}{dt}\theta_1(t)+\frac{d}{dt}\theta_2(t)+\frac{d}{dt}\theta_3(t)\right)\cos\left(\theta_1(t)+\theta_2(t)+\theta_3(t)\right)+\cos\left(\theta_1(t)\right)\frac{d}{dt}\theta_1(t) \\ 0 \end{bmatrix}$$

5. Initialize the generalized coordinates

$$\begin{bmatrix} \frac{d}{dt}\theta_1(t) \\ \frac{d}{dt}\theta_2(t) \\ \frac{d}{dt}\theta_3(t) \end{bmatrix}$$

6. Find Jacobian matrix (18 * 3)

BASIC DEFINITIONS

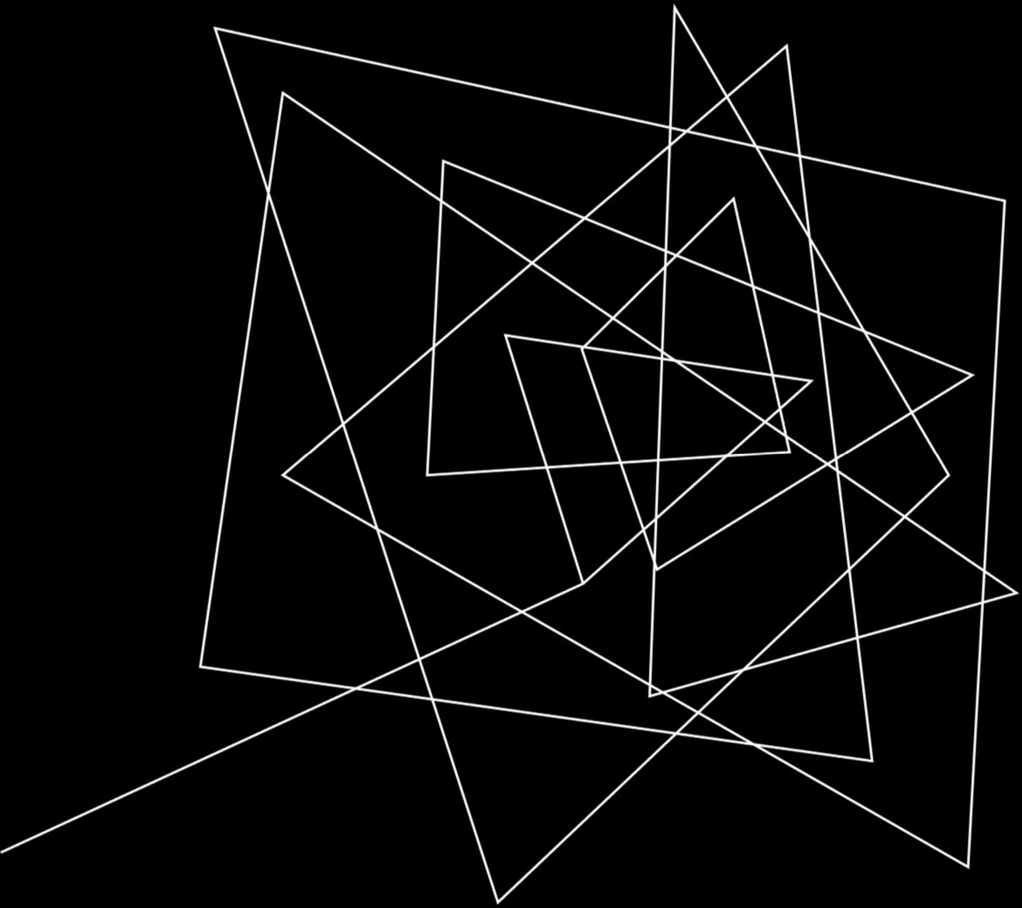
7. Initialize the mass matrix

$$\begin{bmatrix}
m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & m_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & m_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & m_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & m_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & m_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_{23} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_{33}
\end{bmatrix}$$

8. Initialize the frame rotation matrix

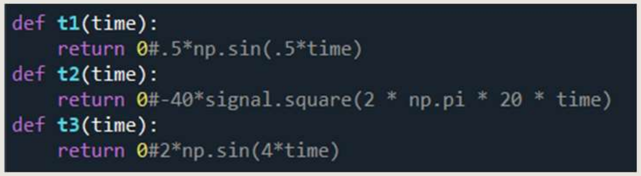
9. Initialize the external forces matrix: G.T

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \tau_1 - \tau_2 & 0 & 0 & \tau_2 - \tau_3 & 0 & 0 & \tau_3 \end{bmatrix}$$

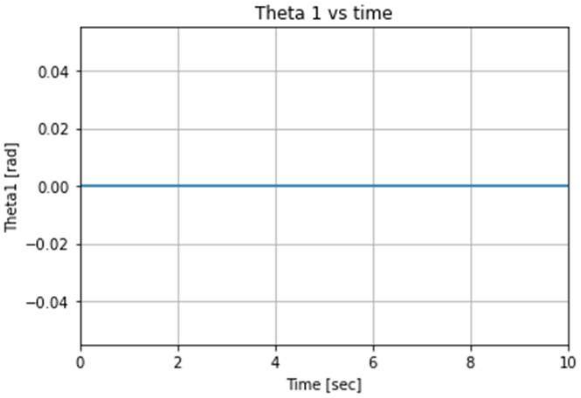


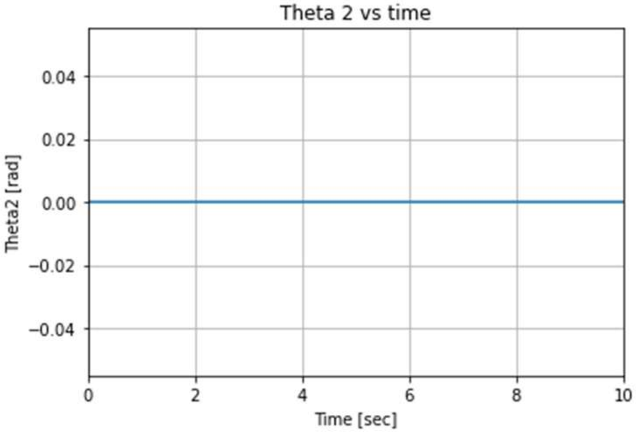
RESULTS AND DISCUSSION

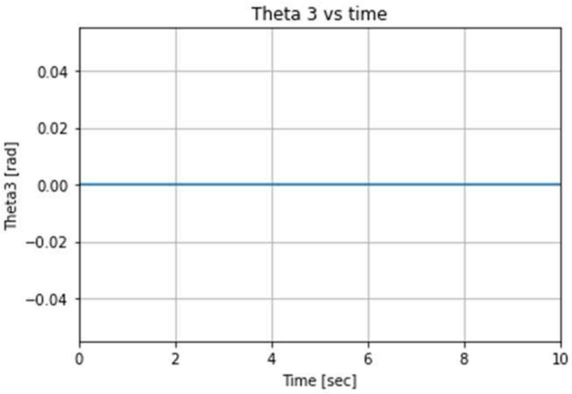
TEST 0 – UNIT TEST



```
def t1(time):
    return 0#.5*np.sin(.5*time)
def t2(time):
    return 0#-40*signal.square(2 * np.pi * 20 * time)
def t3(time):
    return 0#2*np.sin(4*time)
```

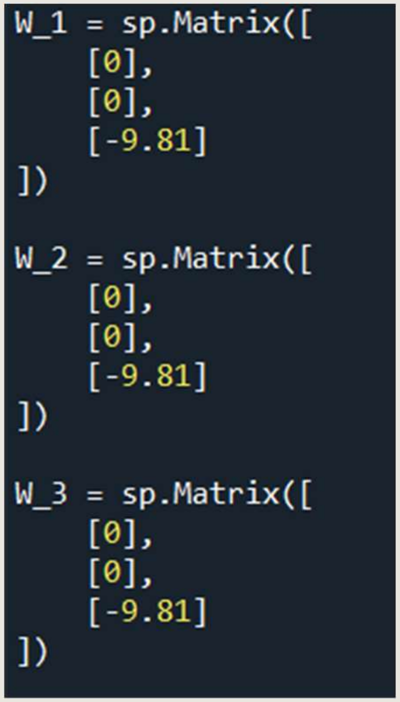






With no torques, the system should be stationary and accordingly the angles don't change from their initial conditions of 0 degrees.

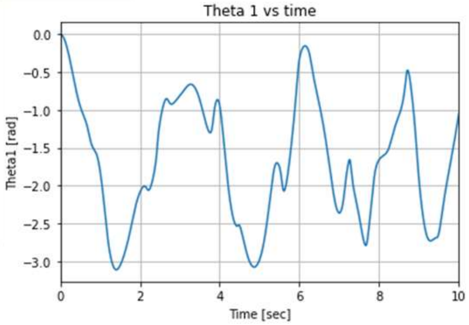
TEST 0 – UNIT TEST

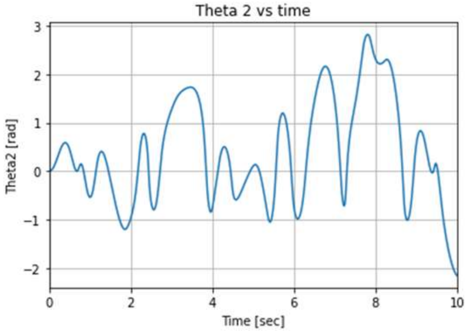


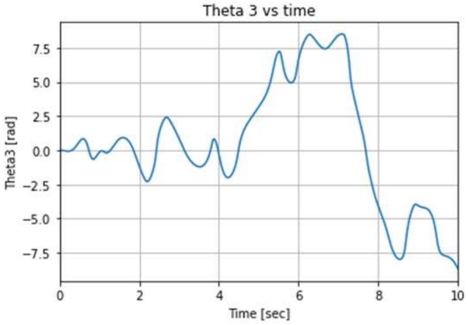
```
W_1 = sp.Matrix([
    [0],
    [0],
    [-9.81]
])

W_2 = sp.Matrix([
    [0],
    [0],
    [-9.81]
])

W_3 = sp.Matrix([
    [0],
    [0],
    [-9.81]
])
```

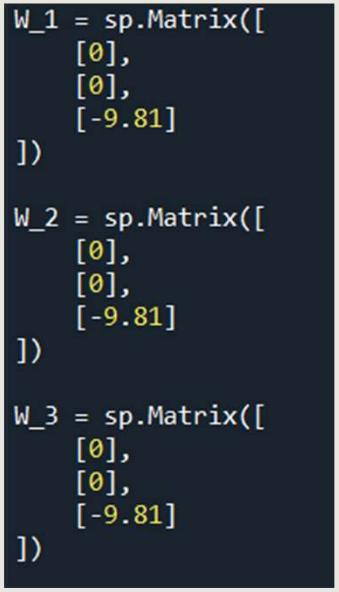






The second test case was giving weight to all linkages and having 0 torques so the linkages can behave as free pendulums. All masses were fixed at 1 and the force was applied in the negative z direction. As expected all the masses behaved like pendulums. Due to the chaotic nature of the system, the above condition gets worse as the number of pendulums increase. Therefore, the length and mass of the second pendulum were made .0001 to see if the first pendulum would still behave like an actual pendulum. The test results are seen in next slide.

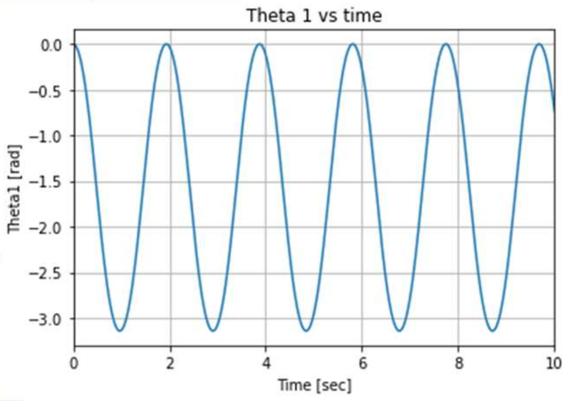
TEST 0 – UNIT TEST

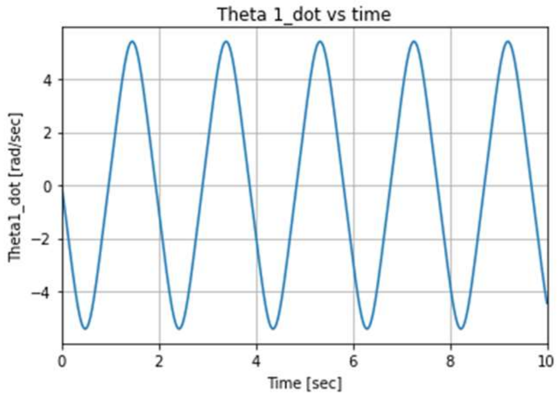


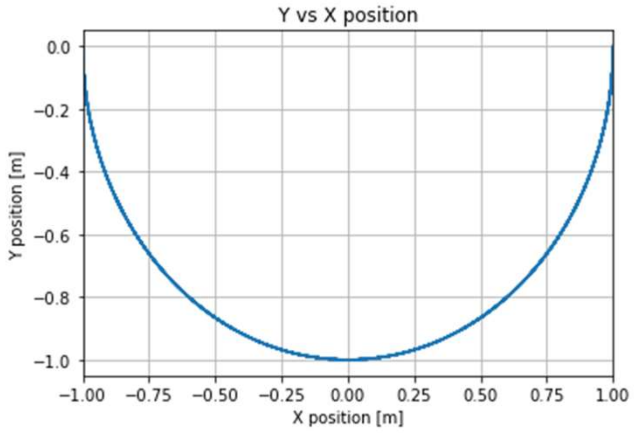
```
W_1 = sp.Matrix([
    [0],
    [0],
    [-9.81]
])

W_2 = sp.Matrix([
    [0],
    [0],
    [-9.81]
])

W_3 = sp.Matrix([
    [0],
    [0],
    [-9.81]
])
```

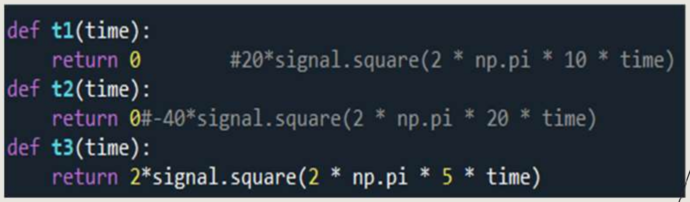




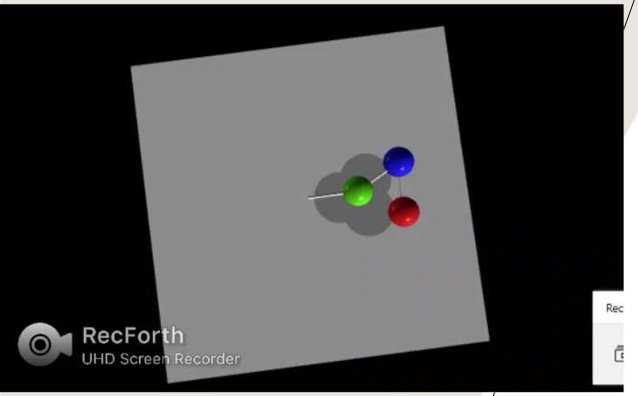


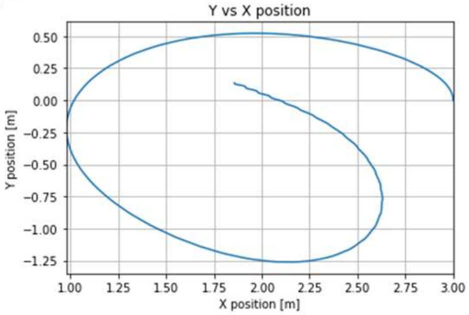
As expected, the first mass behaves exactly like a pendulum when effects of second and thirst pendulum are made negligible.

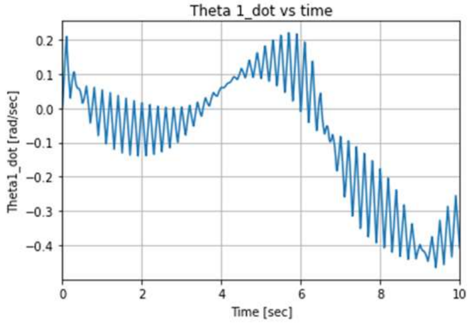
TEST 1 – SINUSOIDAL TORQUES

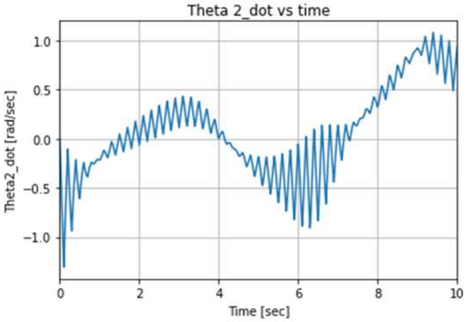


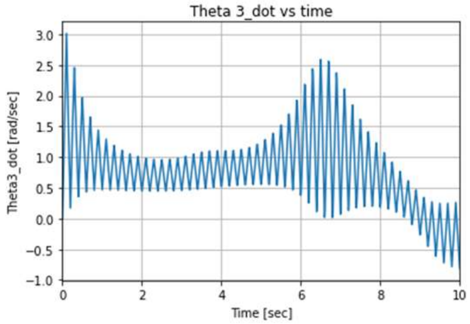
```
def t1(time):
    return 0        #20*signal.square(2 * np.pi * 10 * time)
def t2(time):
    return 0#-40*signal.square(2 * np.pi * 20 * time)
def t3(time):
    return 2*signal.square(2 * np.pi * 5 * time)
```

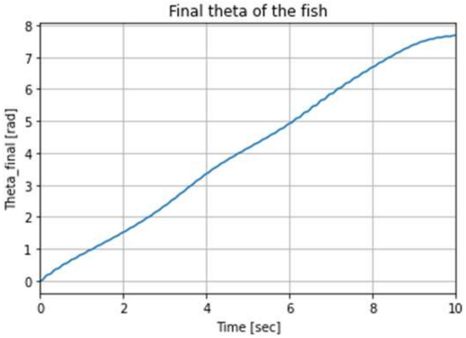






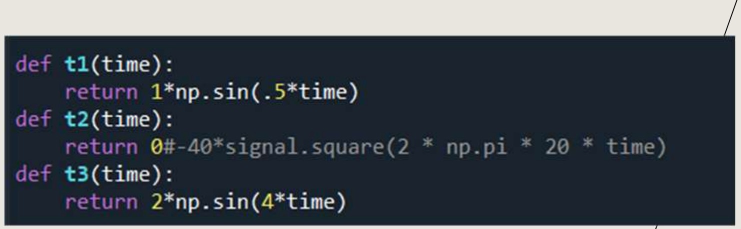




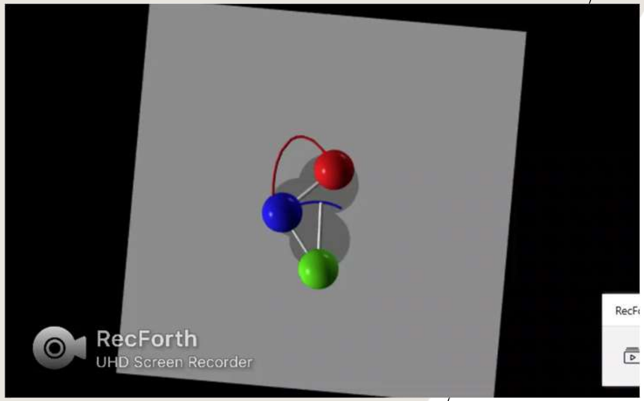


With torque input as a square function, the system has a lot of oscillation in its velocities. Practically this does not happens with a fish since the motion is pretty smooth. So clearly the square input torque is not the way to go. The reason for the smooth y vs x position is because the frequencies are so small that the fish moves really slow (as seen in the video).

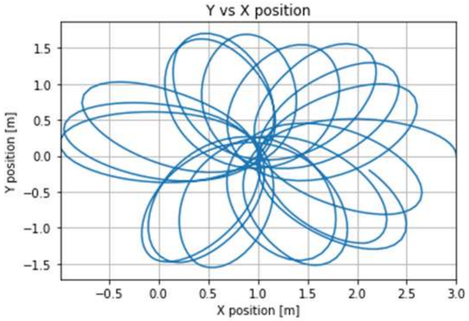
TEST 2 – SQUARE TORUQES

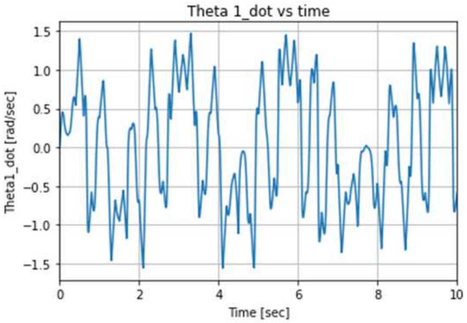


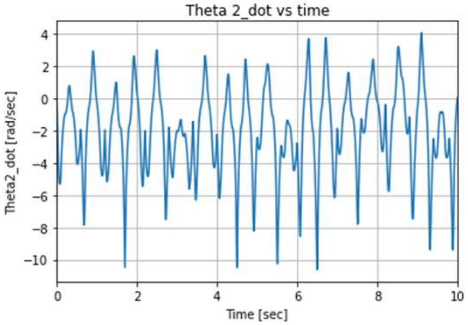
```
def t1(time):
    return 1*np.sin(.5*time)
def t2(time):
    return 0#-40*signal.square(2 * np.pi * 20 * time)
def t3(time):
    return 2*np.sin(4*time)
```

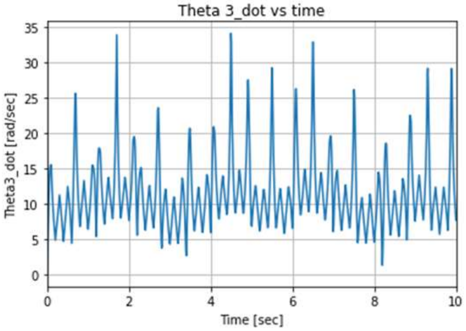


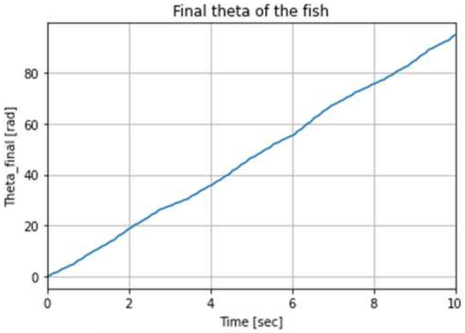
Switching torques to sin waves helps the system to reduce oscillation also making the system smoother. However, the system still overlaps with itself which is not good.



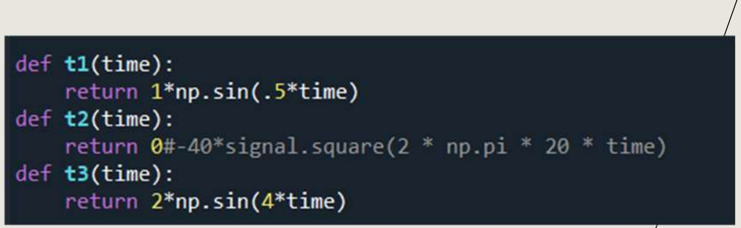




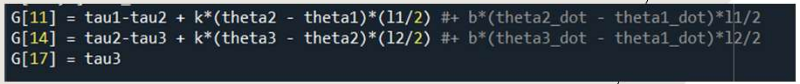




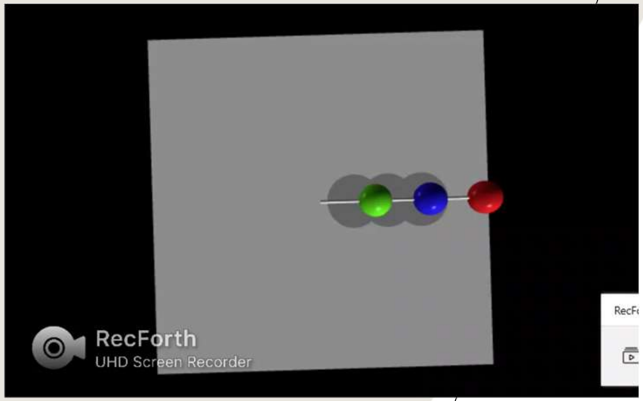
TEST 3 – SINUSOIDAL TORQUES WITH SPRINGS

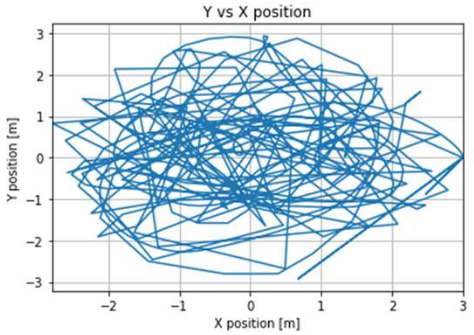


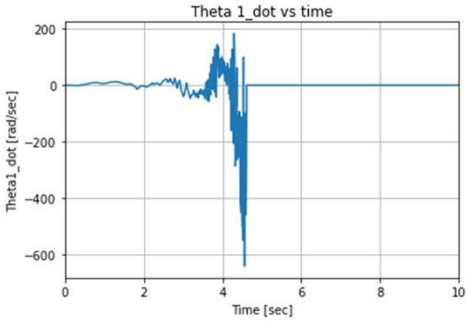
```
def t1(time):
    return 1*np.sin(.5*time)
def t2(time):
    return 0#-40*signal.square(2 * np.pi * 20 * time)
def t3(time):
    return 2*np.sin(4*time)
```

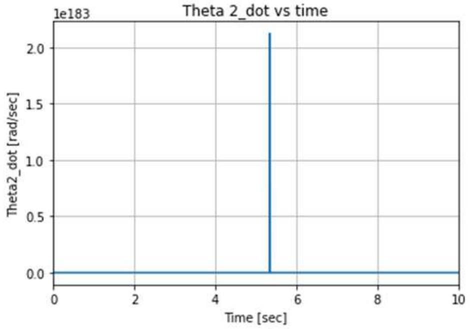


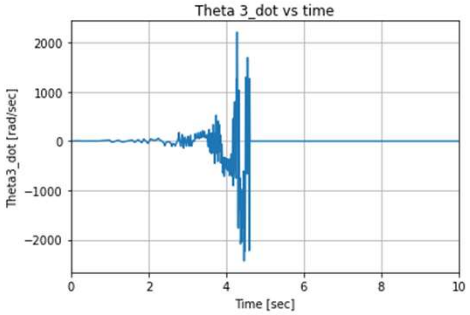
```
G[11] = tau1-tau2 + k*(theta2 - theta1)*(l1/2) #+ b*(theta2_dot - theta1_dot)*l1/2
G[14] = tau2-tau3 + k*(theta3 - theta2)*(l2/2) #+ b*(theta3_dot - theta1_dot)*l2/2
G[17] = tau3
```

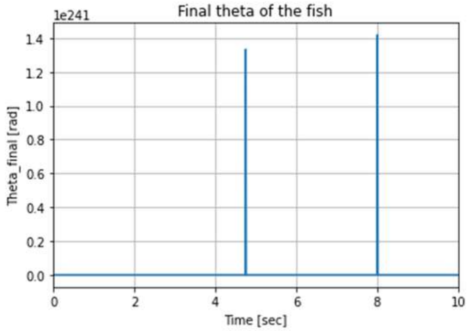






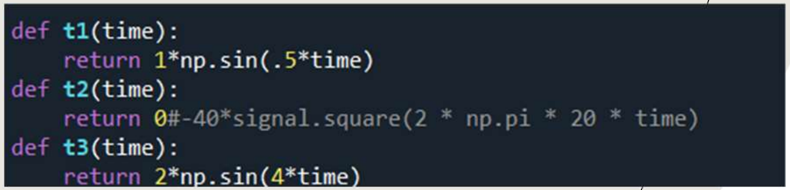




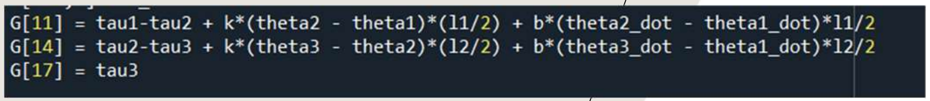


As expected, the springs helped the system in not overlapping. However, the springs energy never dies and makes the system extremely chaotic as time goes on. Clearly the graphs suggest the same because the magnitudes are extreme.

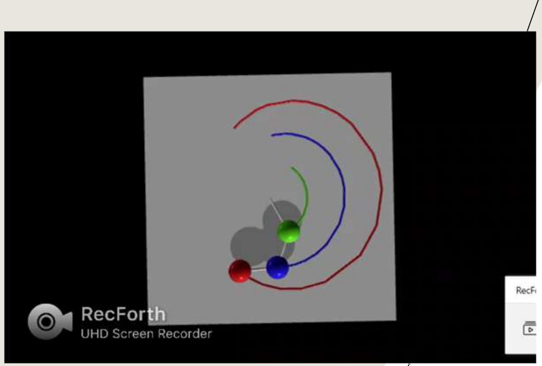
TEST 4 – SIN TORQUES WITH SPRINGS AND DAMPERS

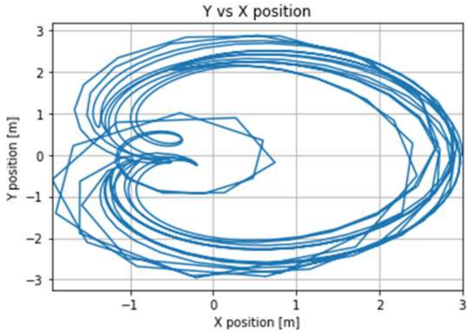


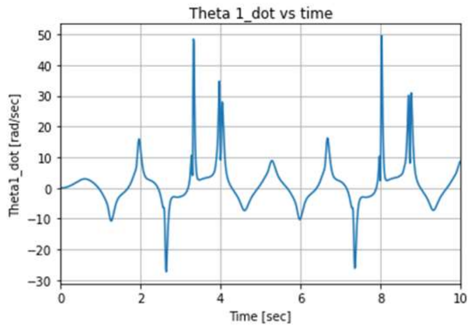
```
def t1(time):
    return 1*np.sin(.5*time)
def t2(time):
    return 0#-40*signal.square(2 * np.pi * 20 * time)
def t3(time):
    return 2*np.sin(4*time)
```

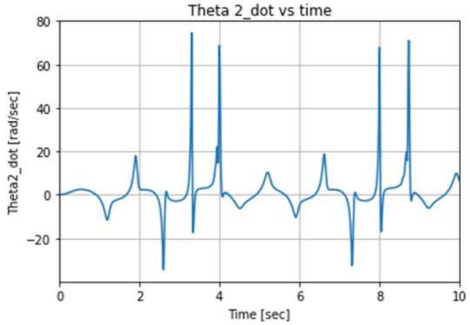


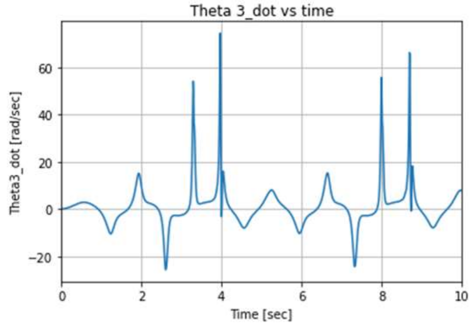
```
G[11] = tau1-tau2 + k*(theta2 - theta1)*(l1/2) + b*(theta2_dot - theta1_dot)*l1/2
G[14] = tau2-tau3 + k*(theta3 - theta2)*(l2/2) + b*(theta3_dot - theta1_dot)*l2/2
G[17] = tau3
```

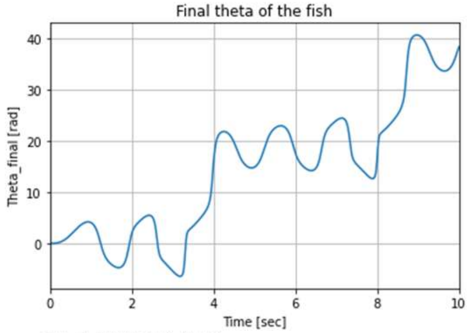












As expected, adding dampers along with springs helps the system loose the energy built up by the springs and cools the system down. In fact, this works best in replicating the motion of the fish. Even though it does