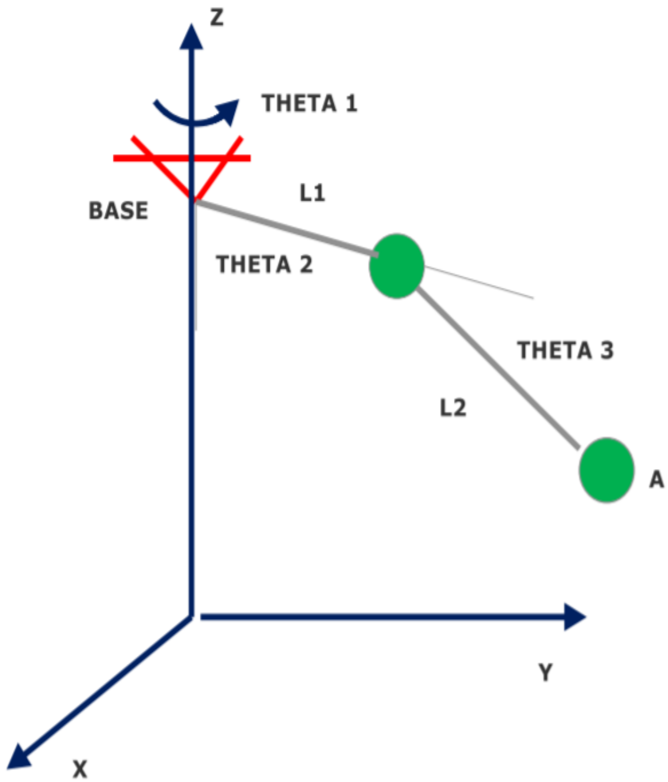


PROJECT 3

ABSTRACT

This project aims to simulate the dynamics of a 2DOF robotic mechanism

Joshua Bukaty
Birahim Ndiaye
Benedict Cutri
Rizve Chowdhury

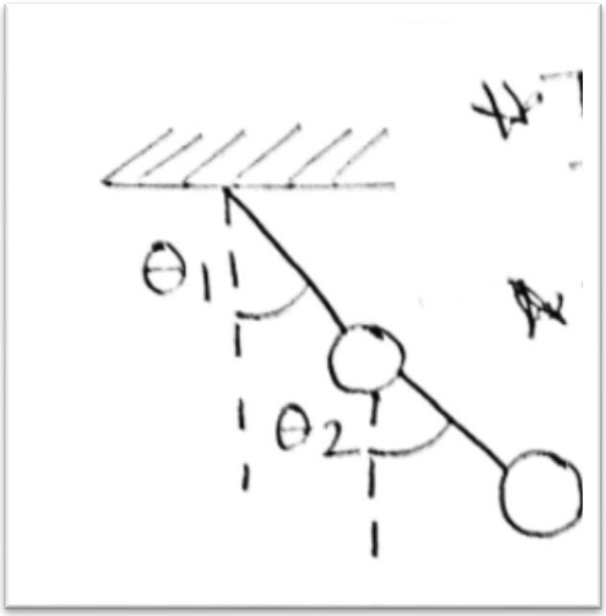
Robotics I

Responsibilities:

- **Presentation:**
  Joshua Bukaty
  Birahim Ndiaye
  Benedict Cutri
- **Simulation:**
  Rizve Chowdhury

Section-1: Derivation of System Dynamics

Before we began any treatment of the system, we noticed that the system acts very similar to a double pendulum and choose to model the system as that. As can be seen the diagram below, the actuator was modeled as a double pendulum.



To derive the equations-of-motion for this mechanism, we used the Lagrangian formulism as follows (the system was assumed to be conservative):

- Derive the total kinetic energy for the system
- Derive the total potential energy for the system
- Find the Lagrangian for the system.
- Utilize the Euler-Lagrange equations for each DOF.

Using the axes convention in the diagram above, the position of each of the masses are as follows:

$$\overrightarrow{r_1} = L_1\cos(\theta_1)\,\hat{x} + L_1\sin(\theta_1)\,\hat{y}$$

$$\overrightarrow{r_2} = (L_1\cos(\theta_1) + L_2\cos(\theta_2))\hat{x} + (L_1\sin(\theta_1) + L_2\sin(\theta_2))\widehat{y}$$

Taking the derivative of these equations, we get:

$$\overrightarrow{V_1} = -L_1\dot{\theta}_2\sin(\theta_2)\,\hat{x} + L_1\dot{\theta}_2\cos(\theta_2)\,\hat{y}$$

$$\overrightarrow{V_2} = -(L_1\dot{\theta}_1\sin(\theta_1)+L_2\dot{\theta}_2\sin(\theta_2))\,\hat{x} + ((L_1\dot{\theta}_1\cos(\theta_1)+L_2\dot{\theta}_2\cos(\theta_2))\hat{y}$$

With this, the kinetic energy of each of the masses can be found:

$$T_1 = \frac{1}{2}m_1\overrightarrow{V_1}\cdot\overrightarrow{V_1} = \frac{1}{2}m_1L_1^2\dot{\theta}_1^2$$

$$T_2 = \frac{1}{2}m_2\overrightarrow{V_2}\cdot\overrightarrow{V_2} = \frac{1}{2}m_2(L_1^2\dot{\theta}_1^2 + 2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2) + L_2^2\dot{\theta}_2^2)$$

The potential energy of each of the masses is:

$$U_1 = -m_1gL_1\cos(\theta_1)$$

$$U_2 = -m_2g(L_1\cos(\theta_1) + L_2\cos(\theta_2))$$

From these equations, the Lagrangian is found to be:

$$L = \Sigma_{i=1}^2 T_i - \Sigma_{i=1}^2 U_i$$

The Euler-Lagrange equations are given by (for each DOF):

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}_i}\right) - \frac{\partial L}{\partial\theta_i} = F_{\theta_i}$$

where F is the generalized force for that particular DOF. For both our DOF, θ_1 and θ_2, the generalized forces are the torques.

Applying the Euler-Lagrange equations, we can generate the equations-of-motion as:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where the following notation was used:

$$A_{11} = m_1 L_1^2 + m_2 L_1^2$$

$$A_{12} = m_2 L_1 L_2 \cos(\theta_1 - \theta_2)$$

$$A_{21} = m_2 L_1 L_2 \cos(\theta_1 - \theta_2)$$

$$A_{22} = m_2 L_2^2$$

$$b_1 = -g L_1 \sin(\theta_1)\,(m_1 + m_2) - m_2 L_1 L_2 \dot{\theta}_2^{\,2} \sin(\theta_1 - \theta_2) + \tau_1$$

$$b_2 = -g L_2 \sin(\theta_2)\, m_2 + m_2 L_1 L_2 \dot{\theta}_1^{\,2} \sin(\theta_1 - \theta_2) + \tau_2$$

Here, τ_1 and τ_2 are the torque inputs to the respective joints. This term can also account for disturbance torques and damping terms.

Given values for θ_1, $\dot{\theta}_1$, θ_2 and $\dot{\theta}_2$, we can find the acceleration vector $\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$. Using this, we can step-forward in time to simulate the system as done in the next section.

Section-2: Simulation of System Dynamics

Methodology:

A fourth-order Runge-Kutta technique was used to carry out the numerical integration. The algorithm was as follows:

1.) Initialize time i = 0
2.) Initialize using the values of θ_1, $\dot{\theta}_1$, θ_2 and $\dot{\theta}_2$ at time t = 0.01i
3.) Using these values find $\begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix}$ at time t = 0 by solving the system of equations

$$\begin{bmatrix}A_{11} & A_{12} \\ A_{21} & A_{22}\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix} = \begin{bmatrix}b_1 \\ b_2\end{bmatrix}$$

4.) Use Runge-Kutta to find θ_1, $\dot{\theta}_1$, θ_2 and $\dot{\theta}_2$ at time t = 0.01i
5.) Repeat above steps 2-4 until t = 5 minutes = 300 seconds.

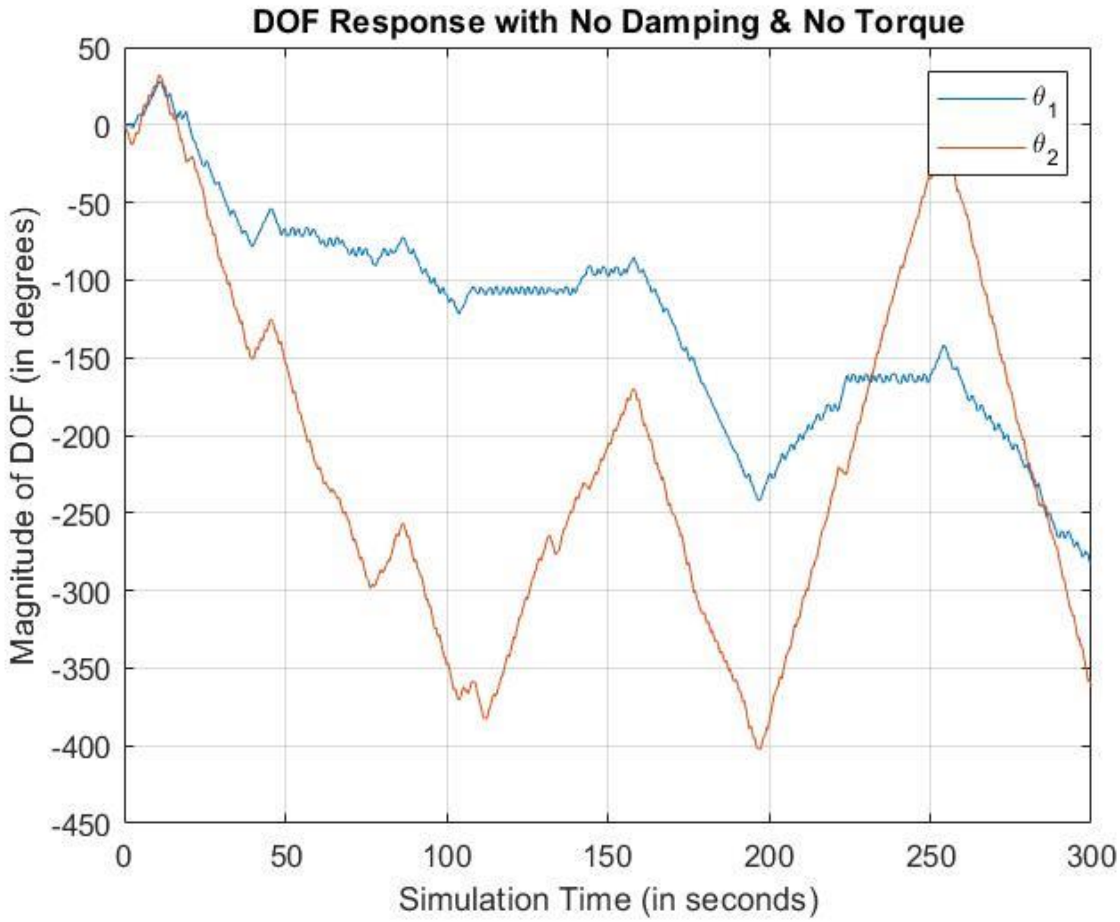
The MATLAB implementation of the system dynamics with no damping and no input is reconstructed below (Plotting commands are also included to visualize results):

```
%Initialize the state of the system
X_int = [2,0,0,0]';
%Specify the step-size
step_size = 1E-2;
%Specify the end time of 5 minutes
end_time = 5*60;
%Create a time vector
t = 0:step_size:end_time;
%Pre-allocate a state vector matrx
X = zeros(4,length(t)+1);
%Assign the initial state value to state vector matrix
X(:,1) = X_int;
%Some system parameters
L_1 = 0.5;
L_2 = 0.5;

for i = 1:length(t)
K_1 = step_size*SystemDyn(X(:,i));
K_2 = step_size*SystemDyn(X(:,i)+0.5*K_1);
K_3 = step_size*SystemDyn(X(:,i)+0.5*K_2);
K_4 = step_size*SystemDyn(X(:,i)+K_3);
X(:,i+1) = X(:,i) + (K_1+2*K_2+2*K_3+K_4)/6;
end

X_1 = @(X) L_1*cos(X(1));
Y_1 = @(X) L_1*sin(X(1));
X_2 = @(X) L_1*cos(X(1)) + L_2*cos(X(3));
Y_2 = @(X) L_1*sin(X(1))+ L_2*sin(X(3));
h = figure;
axis tight manual
```

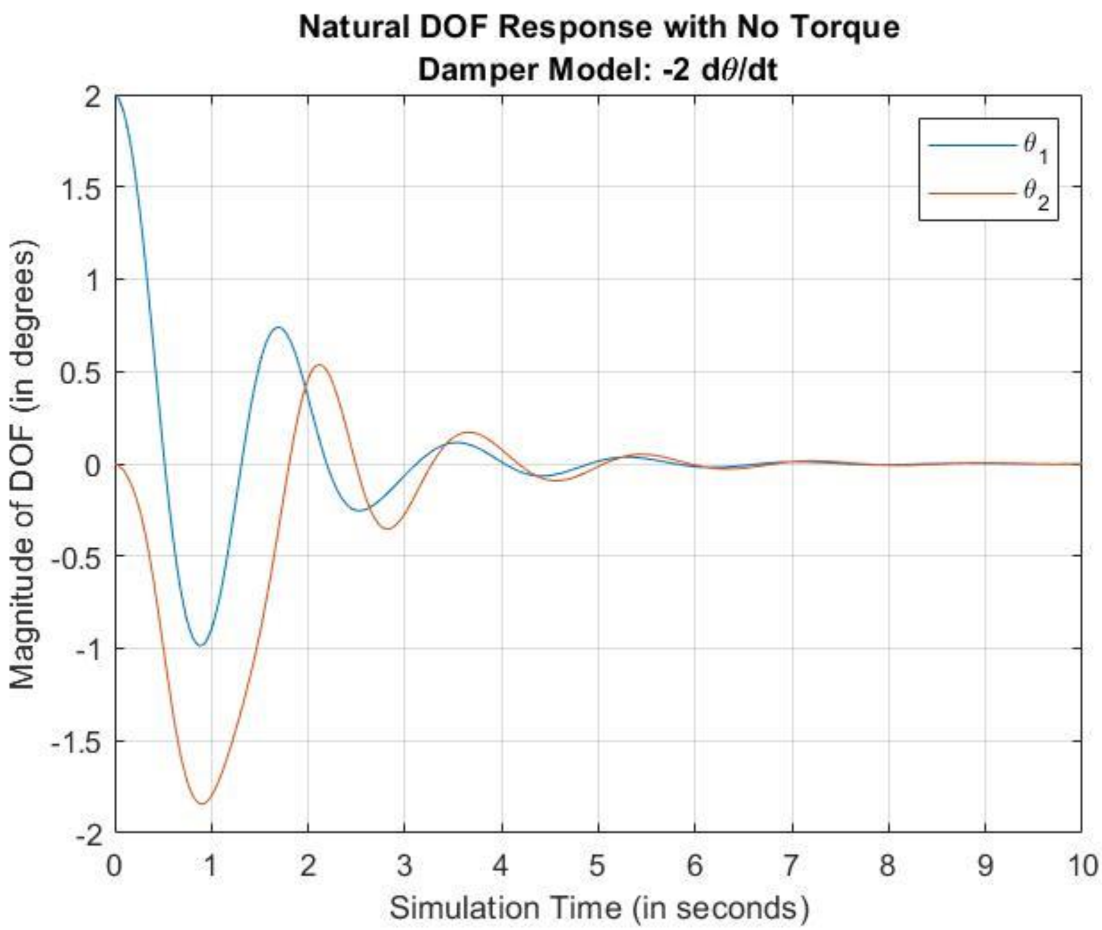
```
plot([0,Y_1(X(:,1)),Y_2(X(:,1))],[0,-X_1(X(:,1)),-X_2(X(:,1))])
ax = gca;
ax.NextPlot = 'replaceChildren';
loops = 800;
M(loops) = struct('cdata',[],'colormap',[]);
h.Visible = 'off';
for i = 1:loops
plot([0,Y_1(X(:,i+1)),Y_2(X(:,i+1))]',[0,-X_1(X(:,i+1)),-
X_2(X(:,i+1))]','-o','MarkerSize',10)
hold on
yline(0)
hold off
title('Double Pendulum')
xlabel('X')
ylabel('Y')
ylim([-2,2])
xlim([-2,2])
drawnow
M(i) = getframe;
end
h.Visible = 'on';

function X_dot = SystemDyn(X)
%System Parameters
m_1 = 5;
m_2 = 2.5;
L_1 = 0.5;
L_2 = 0.5;
g = 9.8;
X_dot = zeros(4,1);

%Creation of A matrix (See Project Document)
A_11 = m_1*L_1^2 + m_2*L_1^2;
A_12 = m_2*L_1*L_2*cos(X(3)-X(1));
A_21 = m_2*L_1*L_2*cos(X(3)-X(1));
A_22 = m_2*L_2^2;

A = [A_11, A_12;
    A_21, A_22];

%Creation of b vecotr (See Project Document)
b_1 = -(m_1+m_2)*g*L_1*sin(X(1))-m_2*L_1*L_2*X(4)^2*sin(X(3)-X(1));
b_2 = -m_2*g*L_2*sin(X(3))+m_2*L_1*L_2*X(2)^2*sin(X(3)-X(1));
b = [b_1;b_2];
Acc = inv(A)*b;

%Constructs the X_dot vector
X_dot(1) = X(2);
X_dot(2) = Acc(1);
X_dot(3) = X(4);
X_dot(4) = Acc(2);
```

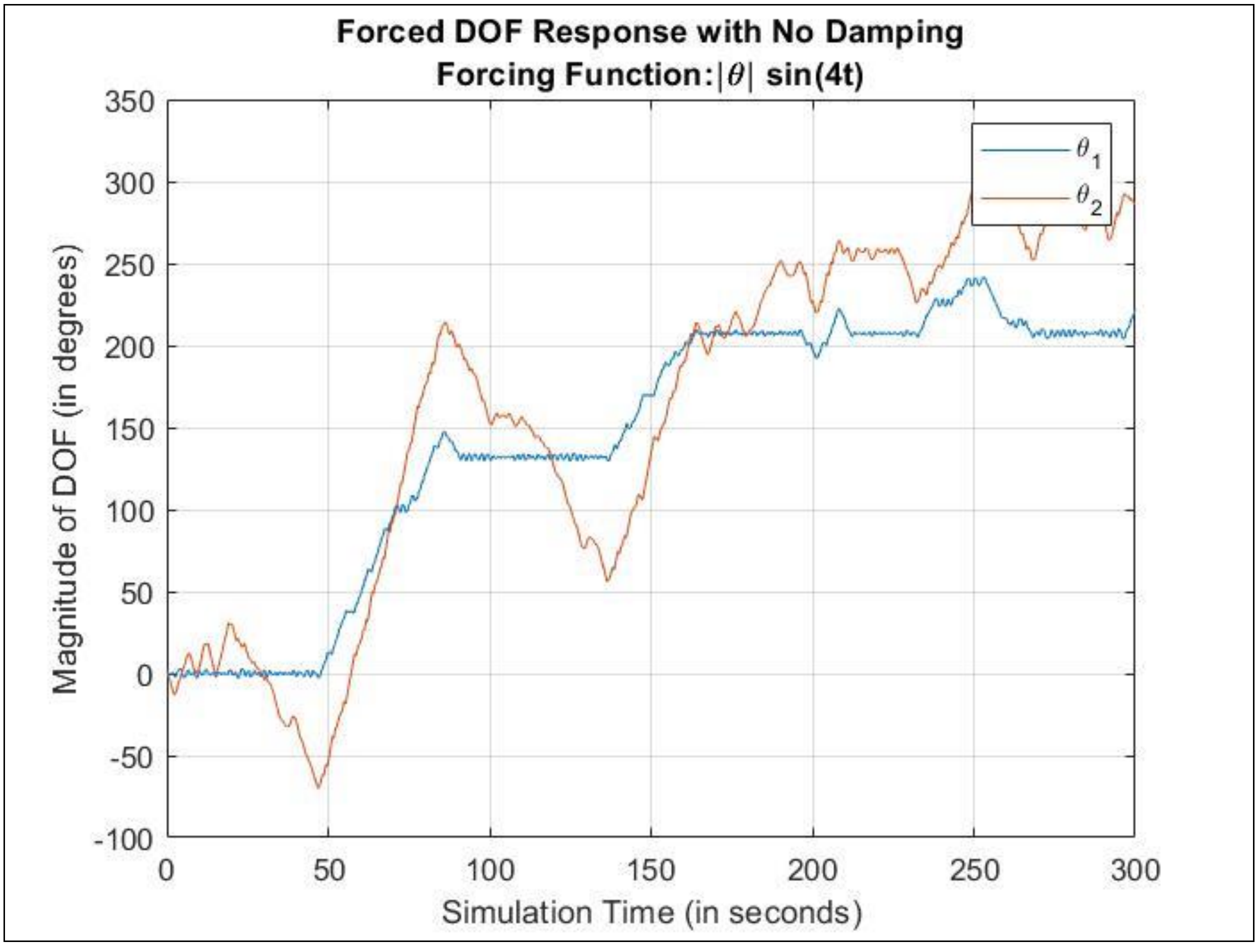
```
end
```

Results:



DOF Response with No Damping & No Torque
Magnitude of DOF (in degrees)
Simulation Time (in seconds)
θ_1
θ_2
50
0
-50
-100
-150
-200
-250
-300
-350
-400
-450
0
50
100
150
200
250
300


Natural DOF Response with No Torque
Damper Model: -2 dθ/dt
θ1
θ2
Magnitude of DOF (in degrees)
Simulation Time (in seconds)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5
6
7
8
9
10


Forced DOF Response with No Damping
Forcing Function: $|\theta|$ sin(4t)
θ_1
θ_2
Magnitude of DOF (in degrees)
Simulation Time (in seconds)
350
300
250
200
150
100
50
0
-50
-100
0
50
100
150
200
250
300