Report Assignment 1

Diagram

Description automatically generated with medium confidence**Submitted by:** Prasham Patel

**Student ID:** 871563809

Fig 1. 2-link system

**Generalize co-ordinates**

The angles are in radians

**Kinetic Energy**

K.E for link1 = 0.5(m1(r1·1’)2 + I1(1’)2)

K.E for link2 = 0.5·m2[(L1· 1)2 + (r2(1 + 2))2 + 2· 1·L1·r2(1 + 2)·cos(θ2)] + 0.5·I2(1 + 2)2

K.E = 0.5(m1(r1·1)2 + I1(1)2) + 0.5·m2[(L1· 1)2 + 0.5·m2[(L1· 1)2 + (r2(1 + 2))2 + 2· 1·L1·r2(1 + 2)·cos(θ2)] + I2(1 + 2)2

**Potential Energy**

P.E for link 1 = m1·g·r1·cos(θ1)

P.E for link2 = m2·g(L1·cos(θ1) + r2·cos(θ1 + θ2))

P.E = m1·g·r1·cos(θ1) - m2·g(L1·cos(θ1) + r2·cos(θ1 + θ2))

**Langrangian function**

L = K.E - P.E

L = 0.5(m1(r1·1)2 + I1(1)2) + 0.5·m2[(L1· 1)2 + 0.5·m2[(L1· 1)2 + (r2(1 + 2))2 + 2· 1·L1·r2(1 + 2)·cos(θ2)] + I2(1 + 2)2 - m1·g·r1·cos(θ1) - m2·g(L1·cos(θ1) + r2·cos(θ1 + θ2))

**Partial differentiations**

Equations are calculated with help of following command in MATLAB commands

>> dl\_dtheta1 = jacobian(L, theta1);

= g\*m2\*(r2\*sin(theta1 + theta2) + l1\*sin(theta1)) + g\*m1\*r1\*sin(theta1)

>> dl\_dtheta2 = jacobian(L, theta2);

= g\*m2\*r2\*sin(theta1 + theta2) - l1\*m2\*r2\*theta\_dot1\*sin(theta2)\*(theta\_dot1 + theta\_dot2)

>> dl\_dtheta\_dot1 = jacobian(L, theta\_dot1);

= theta\_dot1\*I1^2 + m1\*theta\_dot1\*r1^2 + (I2\*(2\*theta\_dot1 + 2\*theta\_dot2))/2 + (m2\*(r2^2\*(2\*theta\_dot1 + 2\*theta\_dot2) + 2\*l1^2\*theta\_dot1 + 2\*l1\*r2\*cos(theta2) \* (theta\_dot1 + theta\_dot2) + 2\*l1\*r2\*theta\_dot1\*cos(theta2)))/2

>> dl\_dtheta\_dot2 = jacobian(L, theta\_dot2);

= (I2\*(2\*theta\_dot1 + 2\*theta\_dot2))/2 + (m2\*((2\*theta\_dot1 + 2\*theta\_dot2)\*r2^2 + 2\*l1\*theta\_dot1\*cos(theta2)\*r2))/2

>> ddl\_dtheta\_dot1\_dt = jacobian(dl\_dtheta\_dot1, [theta1; theta\_dot1])\* [theta\_dot1; theta\_ddot1] + jacobian(dl\_dtheta\_dot1, [theta2; theta\_dot2])\*[theta\_dot2; theta\_ddot2];

= theta\_ddot1\*(I1^2 + m1\*r1^2 + I2 + (m2\*(2\*l1^2 + 4\*cos(theta2)\*l1\*r2 + 2\*r2^2))/2) + theta\_ddot2\*(I2 + (m2\*(2\*r2^2 + 2\*l1\*cos(theta2)\*r2))/2) - (m2\*theta\_dot2\*(2\*l1\*r2\*sin(theta2) \*(theta\_dot1 + theta\_dot2) + 2\*l1\*r2\*theta\_dot1\*sin(theta2)))/2

>> ddl\_dtheta\_dot2\_dt = jacobian(dl\_dtheta\_dot2, [theta1; theta\_dot1])\*[theta\_dot1; theta\_ddot1] + jacobian(dl\_dtheta\_dot2, [theta2; theta\_dot2])\*[theta\_dot2; theta\_ddot2];

= theta\_ddot2\*(m2\*r2^2 + I2) + theta\_ddot1\*(I2 + (m2\*(2\*r2^2 + 2\*l1\*cos(theta2)\*r2))/2) - l1\*m2\*r2\*theta\_dot1\*theta\_dot2\*sin(theta2)

**Euler\_langrange Equation**

u1 = -

u1 is calculated with help of following MATLAB command

>> u1 = ddl\_dtheta\_dot1\_dt - dl\_dtheta1;

u1 = theta\_ddot1\*(I1^2 + m1\*r1^2 + I2 + (m2\*(2\*l1^2 + 4\*cos(theta2)\*l1\*r2 + 2\*r2^2))/2) + theta\_ddot2\*(I2 + (m2\*(2\*r2^2 + 2\*l1\*cos(theta2)\*r2))/2) - (m2\*theta\_dot2\*(2\*l1\*r2\*sin(theta2) \*(theta\_dot1 + theta\_dot2) + 2\*l1\*r2\*theta\_dot1\*sin(theta2)))/2 - g\*m2\*(r2\*sin(theta1 + theta2) + l1\*sin(theta1)) - g\*m1\*r1\*sin(theta1)

u2 = -

u2 is calculated with help of following MATLAB command

>> u2 = ddl\_dtheta\_dot2\_dt - dl\_dtheta2;

u2 = theta\_ddot2\*(m2\*r2^2 + I2) + theta\_ddot1\*(I2 + (m2\*(2\*r2^2 + 2\*l1\*cos(theta2)\*r2))/2) - g\*m2\*r2\*sin(theta1 + theta2) + l1\*m2\*r2\*theta\_dot1\*sin(theta2)\*(theta\_dot1 + theta\_dot2) - l1\*m2\*r2\*theta\_dot1\*theta\_dot2\*sin(theta2)

**State Space representation**

u1 and u2 are solved using the following MATLAB Command

>> sol = solve([u1==0, u2==0], [theta\_ddot1, theta\_ddot2]);

>> theta\_ddot1 = sol.theta\_ddot1;

= (l1\*m2^2\*r2^3\*theta\_dot1^2\*sin(theta2) + l1\*m2^2\*r2^3\*theta\_dot2^2\*sin(theta2) + g\*l1\*m2^2\*r2^2\*sin(theta1) + I2\*g\*l1\*m2\*sin(theta1) + I2\*g\*m1\*r1\*sin(theta1) + 2\*l1\*m2^2\*r2^3\*theta\_dot1\*theta\_dot2\*sin(theta2) + l1^2\*m2^2\*r2^2\*theta\_dot1^2\*cos(theta2)\*sin(theta2) - g\*l1\*m2^2\*r2^2\*sin(theta1 + theta2)\*cos(theta2) + I2\*l1\*m2\*r2\*theta\_dot1^2\*sin(theta2) + I2\*l1\*m2\*r2\*theta\_dot2^2\*sin(theta2) + g\*m1\*m2\*r1\*r2^2\*sin(theta1) + 2\*I2\*l1\*m2\*r2\*theta\_dot1\*theta\_dot2\*sin(theta2))/(I1^2\*I2 + I1^2\*m2\*r2^2 + l1^2\*m2^2\*r2^2 + I2\*l1^2\*m2 + I2\*m1\*r1^2 - l1^2\*m2^2\*r2^2\*cos(theta2)^2 + m1\*m2\*r1^2\*r2^2)

>> theta\_ddot2 = sol.theta\_ddot2 ;

= -(l1\*m2^2\*r2^3\*theta\_dot1^2\*sin(theta2) - I1^2\*g\*m2\*r2\*sin(theta1 + theta2) + l1^3\*m2^2\*r2\*theta\_dot1^2\*sin(theta2) + l1\*m2^2\*r2^3\*theta\_dot2^2\*sin(theta2) - g\*l1^2\*m2^2\*r2\*sin(theta1 + theta2) + g\*l1\*m2^2\*r2^2\*sin(theta1) + I2\*g\*l1\*m2\*sin(theta1) + I2\*g\*m1\*r1\*sin(theta1) + I1^2\*l1\*m2\*r2\*theta\_dot1^2\*sin(theta2) + 2\*l1\*m2^2\*r2^3\*theta\_dot1\*theta\_dot2\*sin(theta2) + 2\*l1^2\*m2^2\*r2^2\*theta\_dot1^2\*cos(theta2)\*sin(theta2) + l1^2\*m2^2\*r2^2\*theta\_dot2^2\*cos(theta2)\*sin(theta2) - g\*l1\*m2^2\*r2^2\*sin(theta1 + theta2)\*cos(theta2) + g\*l1^2\*m2^2\*r2\*cos(theta2)\*sin(theta1) - g\*m1\*m2\*r1^2\*r2\*sin(theta1 + theta2) + I2\*l1\*m2\*r2\*theta\_dot1^2\*sin(theta2) + I2\*l1\*m2\*r2\*theta\_dot2^2\*sin(theta2) + g\*m1\*m2\*r1\*r2^2\*sin(theta1) + 2\*l1^2\*m2^2\*r2^2\*theta\_dot1\*theta\_dot2\*cos(theta2)\*sin(theta2) + l1\*m1\*m2\*r1^2\*r2\*theta\_dot1^2\*sin(theta2) + 2\*I2\*l1\*m2\*r2\*theta\_dot1\*theta\_dot2\*sin(theta2) + g\*l1\*m1\*m2\*r1\*r2\*cos(theta2)\*sin(theta1))/(I1^2\*I2 + I1^2\*m2\*r2^2 + l1^2\*m2^2\*r2^2 + I2\*l1^2\*m2 + I2\*m1\*r1^2 - l1^2\*m2^2\*r2^2\*cos(theta2)^2 + m1\*m2\*r1^2\*r2^2)

**Plotting state trajectory**

The ode function for getting the state space values and used state space representation derived above in the function. The command to simulate in ODE45 and plot is as below:

>> [t, y] = ode45(@ode\_2link, [0, 10], [pi/6, p1/4, 0, 0]);

>> plot(t, y);

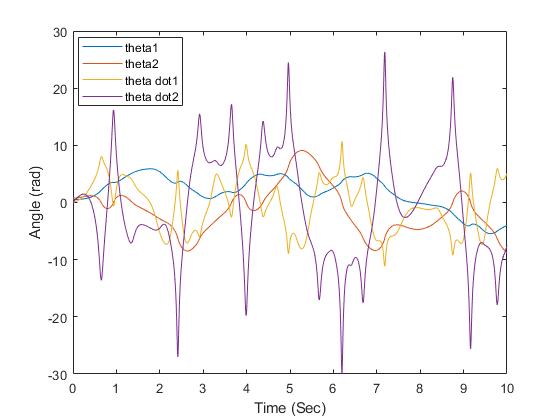


Fig 2. State space trajectory plot