Task 1.a Deriving lagrangian equation of one dimention

A one-dimensional system with mass m where y is the generalized coordinate and a particle is constrained to vertical movement. Newton's second law can be discribes as:

$$m\ddot{y} = f - mg \tag{1}$$

The left side of equation can be written as:

$$m\ddot{y} = \frac{d}{dt}(m\ddot{y}) = \frac{d}{dt}\frac{\partial}{\partial \dot{y}}(\frac{1}{2}m\dot{y}^2) = \frac{d}{dt}\frac{\partial K}{\partial \dot{y}} \tag{2}$$

where $K = \frac{1}{2}m\dot{y}^2$ is the kinetic energy.

We can also describe the gravitational force as well:

$$mg = \frac{\partial}{\partial y}(mgy) = \frac{\partial P}{\partial y} \tag{3}$$

where $P = mgy$ is the potential energy.

If we define:

$$L = K - P = \frac{1}{2}m\dot{y}^2 - mgy \tag{4}$$

$$\frac{\partial L}{\partial \dot{y}} = m\dot{y} = \frac{\partial}{\partial \dot{y}}(\frac{1}{2}m\dot{y}^2) = \frac{\partial}{\partial \dot{y}}K \tag{5}$$

$$\frac{\partial L}{\partial y} = -mg = \frac{\partial}{\partial y}(-mgy) = -\frac{\partial}{\partial \dot{y}}P \tag{6}$$

From equation(1),(5) and (6) we can describe the equation as follow:

$$m\ddot{y} = f - mg \tag{7}$$

$$m\ddot{y} + mg = f \tag{8}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = f \tag{9}$$

Task 1.b Derive the Lagrangian for the two link CrustCrawler

The Lagrangian equation is describe as follow:

$$\mathcal{L} = \mathcal{K} - \mathcal{P} \tag{10}$$

where $\mathcal{K}$ is kinetic energy and $\mathcal{P}$ is potential energy.

Potential energy for each mass and summing the together:

$$\mathcal{P} = P_1 + P_2 \tag{11}$$

$$P_1 = m_1 g \frac{L_1}{2} \tag{12}$$

$$P = m_1 g \frac{L_1}{2} + m_2 g (L_1 \pm L_2 s_2) \tag{13}$$

$\mathcal{K}$ is kinetic energy and we have the following equation.

$$\mathcal{K} = \frac{1}{2} \dot{q}^T [\sum_{i=1}^{n} (m_i J_{vi}^T J_{vi} + J_{\omega i}^T R_i I_i R_i^T J_{\omega i})] \dot{q} \tag{14}$$

We have the Jacobian for calculating velocities.

$$J = \begin{bmatrix} -s_1(L_2c_2 + L_3c_{23}) & -c_1(L_2s_2 + L_3s_{23}) & -c_1(L_3s_{23}) \\ -c_1(L_2c_2 + L_3c_{23}) & -s_1(L_2s_2 + L_3s_{23}) & -s_1(L_3s_{23}) \\ 0 & (L_2c_2 + L_3c_{23}) & (L_3c_{23}) \\ 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix}$$

Since it is the third joint that has been removed, the third column and all instances of L_3 in J should be set to zero. Therefore we can derive the Jacobian as below:

$$J = \begin{bmatrix} -L_2s_1c_2 & -L_2c_1s_2 & 0 \\ -L_2c_1c_2 & -L_2s_1s_2 & 0 \\ 0 & L_2c_2 & 0 \\ 0 & s_1 & 0 \\ 0 & -c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

For i = 1;

$$J_{v1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} ; J_{v1}^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$J_{\omega 1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} ; J_{\omega 1}^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$J_{v1}^T J_{v1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$J_{\omega 1}^T I J_{\omega 1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} I_{1,x} & 0 & 0 \\ 0 & I_{1,y} & 0 \\ 0 & 0 & I_{1,z} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & I_{1,z} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} I_{1,z} & 0 \\ 0 & 0 \end{bmatrix}$$

For i = 2;

$$J_{v2} = \begin{bmatrix} -s_1 L_2 c_2 & -c_1 L_2 s_2 \\ -c_1 L_2 c_2 & -s_1 L_2 s_2 \\ 0 & L_2 c_2 \end{bmatrix}; J_{v2}^T = \begin{bmatrix} -s_1 L_2 c_2 & -c_1 L_2 c_2 & 0 \\ -c_1 L_2 s_2 & -s_1 L_2 s_2 & L_2 c_2 \end{bmatrix}$$

$$J_{v2}^T J_{v2} = \begin{bmatrix} -s_1 L_2 c_2 & -c_1 L_2 c_2 & 0 \\ -c_1 L_2 s_2 & -s_1 L_2 s_2 & L_2 c_2 \end{bmatrix} \begin{bmatrix} -s_1 L_2 c_2 & -c_1 L_2 s_2 \\ -c_1 L_2 c_2 & -s_1 L_2 s_2 \\ 0 & L_2 c_2 \end{bmatrix} = \begin{bmatrix} L_2^2 c_2^2 & 2 s_1 c_1 L_2^2 s_2 c_2 \\ 2 s_1 c_1 L_2^2 s_2 c_2 & L_2^2 \end{bmatrix}$$

$$J_{\omega 2} = \begin{bmatrix} 0 & s_1 \\ 0 & -c_1 \\ 1 & 0 \end{bmatrix}; J_{\omega 2}^T = \begin{bmatrix} 0 & 0 & 1 \\ s_1 & -c_1 & 0 \end{bmatrix}$$

We have $J_{\omega 2}^T \mathcal{I} J_{\omega 2}$, where $\mathcal{I} = R I_i R^T$.

Using that $R = R_{M2} = R_2^0 = R_0^1 R_{z,\theta 2}$, where R_2^0 is extracted from A_2 in forward kinematics.

$$R = R_2^0 = \begin{bmatrix} c_1 c_2 & -c_1 s_2 & -s_1 \\ s_1 c_2 & -s_1 s_2 & c_1 \\ -s_2 & -c_2 & 0 \end{bmatrix}; R^T = {R_2^0}^T = \begin{bmatrix} c_1 c_2 & s_1 c_2 & -s_1 \\ -c_1 s_2 & -s_1 s_2 & -c_1 \\ -s_1 & c_1 & 0 \end{bmatrix}$$

and therefore;

$$R^T J_{\omega 2} = \begin{bmatrix} c_1 c_2 & s_1 c_2 & -s_1 \\ -c_1 s_2 & -s_1 s_2 & -c_1 \\ -s_1 & c_1 & 0 \end{bmatrix} \begin{bmatrix} 0 & s_1 \\ 0 & -c_1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -s_1 & 0 \\ -c_1 & 0 \\ 0 & -s_1^2 + c_1^2 \end{bmatrix}$$

Using the equality of $J_{\omega 2}^T R = (R^T J_{\omega 2})^T$,

$$J_{\omega 2}^T R = \begin{bmatrix} -s_1 & -c_1 & 0 \\ 0 & 0 & -s_1^2 + c_1^2 \end{bmatrix}$$

$$J_{\omega 2}^T R I = \begin{bmatrix} -s_1 & -c_1 & 0 \\ 0 & 0 & -s_1^2 + c_1^2 \end{bmatrix} \begin{bmatrix} I_{2,x} & 0 & 0 \\ 0 & I_{2,y} & 1 \\ 0 & 0 & I_{2,z} \end{bmatrix} = \begin{bmatrix} -s_1 I_{2,x} & -c_1 I_{2,y} & 0 \\ 0 & 0 & (-s_1^2 + c_1^2) I_{2,z} \end{bmatrix}$$

$$(J_{\omega 2}^T RI)(R^T J_{\omega 2}) = \begin{bmatrix} -s_1 I_{2,x} & -c_1 I_{2,y} & 0 \\ 0 & 0 & (-s_1^2 + c_1^2) I_{2,z} \end{bmatrix} \begin{bmatrix} -s_1 & 0 \\ -c_1 & 0 \\ 0 & -s_1^2 + c_1^2 \end{bmatrix} = \begin{bmatrix} s_1^2 I_{2,x} + c_1^2 I_{2,y} & 0 \\ 0 & (s^4 - 2s_1^2 c_1^2 - c_1^4) I_{2,z} \end{bmatrix}$$

Then from equation (14) we get:

$$\mathcal{K} = \frac{1}{2}\dot{q}^T [(m_1 J_{vi}^T J_{v1} + J_{\omega 1}^T R_1 I_1 R_1^T J_{\omega 1}) + (m_2 J_{v2}^T J_{v2} + J_{\omega 2}^T R_2 I_2 R_2^T J_{\omega 2})]\dot{q} \tag{15}$$

$$\mathcal{K} = \frac{1}{2}\dot{q}^T [(m_1 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} I_{1,z} & 0 \\ 0 & 0 \end{bmatrix}) + (m_2 \begin{bmatrix} L_2^2 c_2^2 & 2s_1 c_1 L_2^2 s_2 c_2 \\ 2s_1 c_1 L_2^2 s_2 c_2 & L_2^2 \end{bmatrix} + \begin{bmatrix} s_1^2 I_{2,x} + c_1^2 I_{2,y} & 0 \\ 0 & (s^4 - 2s_1^2 c_1^2 - c_1^4) I_{2,z} \end{bmatrix})]\dot{q} \tag{16}$$

$$\mathcal{K} = \frac{1}{2}\dot{q}^T [\begin{bmatrix} I_{1,z} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y} & m_2 2s_1 c_1 L_2^2 s_2 c_2 \\ m_2 2s_1 c_1 L_2^2 s_2 c_2 & m_2 L_2^2 + (s^4 - 2s_1^2 c_1^2 - c_1^4) I_{2,z} \end{bmatrix}]\dot{q} \tag{17}$$

$$\mathcal{K} = \frac{1}{2} \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix} \begin{bmatrix} I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y} & m_2 2s_1 c_1 L_2^2 s_2 c_2 \\ m_2 2s_1 c_1 L_2^2 s_2 c_2 & m_2 L_2^2 + (s^4 - 2s_1^2 c_1^2 - c_1^4) I_{2,z} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{18}$$

$$\mathcal{K} = \frac{1}{2} \begin{bmatrix} (I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y})\dot{\theta}_1 + (m_2 2s_1 c_1 L_2^2 s_2 c_2)\dot{\theta}_2 & (m_2 2s_1 c_1 L_2^2 s_2 c_2)\dot{\theta}_1 + (m_2 L_2^2 + (s^4 - 2s_1^2 c_1^2 - c_1^4) I_{2,z})\dot{\theta}_2 \end{bmatrix} \tag{19}$$

$\mathcal{K} = \frac{1}{2}[(I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y})\dot{\theta}_1^2 + (m_2 2s_1 c_1 L_2^2 s_2 c_2)\dot{\theta}_1\dot{\theta}_2 + (m_2 2s_1 c_1 L_2^2 s_2 c_2)\dot{\theta}_1\dot{\theta}_2 + (m_2 L_2^2 + (s^4 - 2s_1^2 c_1^2 - c_1^4) I_{2,z})\dot{\theta}_2^2]$

Then from equation (10),(13) ,(19) we get:

$\mathcal{L} = \frac{1}{2}(I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y})\dot{\theta}_1^2 + (m_2 s_1 c_1 L_2^2 s_2 c_2)\dot{\theta}_1\dot{\theta}_2 + \frac{1}{2}(m_2 L_2^2 + (s_1^4 - 2s_1^2 c_1^2 - c_1^4) I_{2,z})\dot{\theta}_2^2 - m_1 g \frac{L_1}{2} + m_2 g(L_1 \pm L_2 s_2)$

Task 1.c Deriving the dynamical model for the two link CrustCrawler using the Euler-Lagrange Equations

The dynamical model of robot can be describe:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \tau \tag{20}$$

For the manipulatior, the equation should be:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} - \frac{\partial \mathcal{L}}{\partial \theta_1} = \tau_1 \tag{21}$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} - \frac{\partial \mathcal{L}}{\partial \theta_2} = \tau_2 \tag{22}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y})\dot{\theta}_1$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (I_{1,z} - m_2 L_2^2 2 c_2 s_2 \dot{\theta}_2 + 2 s_1 c_1 I_{2,x}\dot{\theta}_1 - 2 c_1 s_1 I_{2,y}\dot{\theta}_1)\dot{\theta}_1 + (I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y})\ddot{\theta}_1$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = 0$$

$$\tau_1 = (I_{1,z} - m_2 L_2^2 2 c_2 s_2 \dot{\theta}_2 + 2 s_1 c_1 I_{2,x}\dot{\theta}_1 - 2 c_1 s_1 I_{2,y}\dot{\theta}_1)\dot{\theta}_1 + (I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y})\ddot{\theta}_1$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = (m_2 s_1 c_1 L_2^2 s_2 c_2)\dot{\theta}_1\dot{\theta}_2 + (m_2 L_2^2 + (s_1^4 - 2 s_1^2 c_1^2 - c_1^4) I_{2,z})\dot{\theta}_2$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = (m_2 L_2^2 + (s_1^4 - 2 s_1^2 c_1^2 - c_1^4) I_{2,z})\ddot{\theta}_2$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = 0$$

$$\tau_2 = (m_2 L_2^2 + (s_1^4 - 2 s_1^2 c_1^2 - c_1^4) I_{2,z})\ddot{\theta}_2$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} (I_{1,z} - m_2 L_2^2 2 c_2 s_2 \dot{\theta}_2 + 2 s_1 c_1 I_{2,x}\dot{\theta}_1 - 2 c_1 s_1 I_{2,y}\dot{\theta}_1)\dot{\theta}_1 + (I_{1,z} + m_2 L_2^2 c_2^2 + s_1^2 I_{2,x} + c_1^2 I_{2,y})\ddot{\theta}_1 \\ (m_2 L_2^2 + (s_1^4 - 2 s_1^2 c_1^2 - c_1^4) I_{2,z})\ddot{\theta}_2 \end{bmatrix} \tag{23}$$

Task 2.a The Potential energy for the three-link manipulator

The python code for Potential energy is describe as follow:

Listing 1: Python kode

```
from sympy.interactive import printing
printing.init_printing(use_latex=True)
from sympy import Eq, solve_linear_system, Matrix
from numpy import linalg
import numpy as np
import sympy as sp

def potential_energy(theta_1, theta_2, theta_3):
    theta_1 = (theta_1/180.0)*np.pi
    theta_2 = (theta_2/180.0)*np.pi
    theta_3 = (theta_3/180.0)*np.pi
    L_1 = 2
    L_2 = 3
    L_3 = 0
    m1 = 0.3833
    m2 = 0.2724
    m3 = 0.1406

    s1 = np.sin(theta_1)
    c1 = np.cos(theta_1)
    s2 = np.sin(theta_2)
    c2 = np.cos(theta_2)
    s3 = np.sin(theta_3)
    c3 = np.cos(theta_3)

    R_1 = Matrix([[c1,0,s1],[s1,0,-c1],[0,1,0]])
    R_2 = Matrix([[c1*c2,-c1*s2,s1],[s1*c2,-s1*s2,-c1],[s2,c2,0]])
    R_3 = Matrix([[c1*c2*c3-c1*s2*s3,-c1*c2*s3-c1*s2*c3,s1],[s1*c2*c3-s1*s2*s3,-s1*c2*s3-s1
    R_z1 = Matrix([[c1,-s1,0],[s1,c1,0],[0,0,1]])
    R_z2 = Matrix([[c2,-s2,0],[s2,c2,0],[0,0,1]])
    R_z3 = Matrix([[c3,-s3,0],[s3,c3,0],[0,0,1]])
```

```
    r_c1 = R_1 * R_z1
    r_c2 = R_2 * R_z2
    r_c3 = R_3 * R_z3

    g = [0,0,9.81]
    G = Matrix([g])

    P1 = m1*G*(L_1/2)*r_c1
    P2 = m2*G*(L_2/2)*r_c2
    P3 = m3*G*(L_3/2)*r_c3
    P = P1 + P2 + P3

    return P

pe = potential_energy(0,30,60)
print(pe)
```

Task 2.b The Kinetic energy for the three-link manipulator

The python code for Kinetic energy is describe as follow:

Listing 2: Python kode

```
from sympy.interactive import printing
printing.init_printing(use_latex=True)
from sympy import Eq, solve_linear_system, Matrix
from numpy import linalg
import numpy as np
import sympy as sp

def kinetic_energy(theta_1, theta_2, theta_3):

    theta_1 = (theta_1/180.0)*np.pi
    theta_2 = (theta_2/180.0)*np.pi
    theta_3 = (theta_3/180.0)*np.pi
    L_1 = 2
    L_2 = 3
    L_3 = 4
    m1 = 0.3833
    m2 = 0.2724
    m3 = 0.1406



    s1 = np.sin(theta_1)
    c1 = np.cos(theta_1)
    s2 = np.sin(theta_2)
    c2 = np.cos(theta_2)
    s3 = np.sin(theta_3)
    c3 = np.cos(theta_3)
    s23 = np.sin(theta_2 +  theta_3)
    c23 = np.cos(theta_2 +  theta_3)

    row1 = [-s1*(L_2*c2 + L_3*c23), -c1*(L_2*s2 + L_3*s23), -c1*(L_3*s23)]
    row2 = [c1*(L_2*c2 + L_3*c23), s1*(L_2*s2 + L_3*s23), -s1*(L_3*s23)]
    row3 = [0, (L_2*c2 + L_3*c23), L_3*c23]
```

```
row4 = [0, s1, s1]
row5 = [0, -c1, -c1]
row6 = [1, 0, 0]

J = Matrix((row1, row2, row3,row4,row5,row6))

R_1 = Matrix([[c1,0,s1],[s1,0,-c1],[0,1,0]])
R_2 = Matrix([[c1*c2,-c1*s2,s1],[s1*c2,-s1*s2,-c1],[s2,c2,0]])
R_3 = Matrix([[c1*c2*c3-c1*s2*s3,-c1*c2*s3-c1*s2*c3,s1],[s1*c2*c3-s1*s2*s3,-s1*c2*s3-s1
R_z1 = Matrix([[c1,-s1,0],[s1,c1,0],[0,0,1]])
R_z2 = Matrix([[c2,-s2,0],[s2,c2,0],[0,0,1]])
R_z3 = Matrix([[c3,-s3,0],[s3,c3,0],[0,0,1]])

J_v1 = Matrix([[0,0],[0,0],[0,0]])
J_v1_t = J_v1.transpose()
J_o1 = Matrix([[0,0],[0,0],[1,0]])
J_o1_t = J_o1.transpose()
R_1 = Matrix([[c1,0,s1],[s1,0,-c1],[0,1,0]])
R_1_t = R_1.transpose()
I_1_x,I_1_y,I_1_z = sp.symbols('I_1_x I_1_y I_1_z')
i_1 = Matrix([[I_1_x,0,0],[0,I_1_y,0],[0,0,I_1_z]])
I_1 = R_1*i_1*R_1_t

J_v2 = Matrix([[-s1*(L_2*c2 + L_3*c23),-c1*(L_2*s2 + L_3*s23)],[c1*(L_2*c2 + L_3*c23),s
J_v2_t = J_v1.transpose()
J_o2 = Matrix([[0,s1],[0,-c1],[1,0]])
J_o2_t = J_o2.transpose()
R_2 = Matrix([[c1*c2,-c1*s2,s1],[s1*c2,-s1*s2,-c1],[s2,c2,0]])
R_2_t = R_1.transpose()
I_2_x,I_2_y,I_2_z = sp.symbols('I_2_x I_2_y I_2_z')
i_2 = Matrix([[I_2_x,0,0],[0,I_2_y,0],[0,0,I_2_z]])
I_2 = R_2*i_2*R_2_t

J_v3 = Matrix((row1, row2, row3))
J_v3_t = J_v3.transpose()
J_o3 = Matrix([[0, s1, s1],[0, -c1, -c1],[1, 0, 0]])
```

```
    J_o3_t = J_o1.transpose()
    R_3 = Matrix([[c1*c2*c3-c1*s2*s3,-c1*c2*s3-c1*s2*c3,s1],[s1*c2*c3-s1*s2*s3,-s1*c2*s3-s1
    R_3_t = R_1.transpose()
    I_3_x,I_3_y,I_3_z = sp.symbols('I_3_x I_3_y I_3_z')
    i_3 = Matrix([[I_3_x,0,0],[0,I_3_y,0],[0,0,I_3_z]])
    I_3 = R_3*i_3*R_3_t

    J_v_1 = m1*J_v1_t*J_v1
    J_o_1 = J_o1_t*I_1*J_o1
    J_v_2 = m2*J_v2_t*J_v2
    J_o_2 = J_o2_t*R_2*I_2*R_2_t*J_o2
    J_v_3 = m3*J_v3_t*J_v3
    J_o_3 = J_o3_t*R_3*I_3*R_3_t*J_o3
    q_dot = Matrix([theta_1, theta_2, theta_3])
    q_dot_t = Matrix([[theta_1, theta_2, theta_3]])
    K = (1/2)*q_dot_t*((J_v_1 + J_o_1)+(J_v_2 + J_o_2)+(J_v_3 + J_o_3))*q_dot
    return K

ke = kinetic_energy(0,30,60)
print(ke)
```