# Section2:

Please toggle the following variables in the biggening of the code to check for the desired output – addload = true/false (to add/remove 1kg mass at the end-effector ) [line 11] addrotationalinertia = true/false; (to consider or not the effect of rotational inertia) [line12]

**Problem-1:**

**a**].
To calculate the homogenous transformation matrices representing the pose of each frame with respect to the previous one, we will consider

L1 = 0.3m
L2 = 0.3m
L3 = 0.15m
m1 = 5Kg
m2 = 1kg m3
= 1kg

Coordinates of {1} in {0}: (0, 0, L1/2)
Coordinates of {2} in {0}: (0, L2/2, L1)
Coordinates of {3} in {0}: (0, L2, L1- L3/2)

Calculations are as follows –

Fenil Desai (WPI ID: 249812226)
RBE 501 - Robot Dynamics (Fall-2021)

# Robot Dynamics : Homework 4

## Problem-1:-

→ M01 -

rotation $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  Translation $= [0 \;\; 0 \;\; L1/2]^T$

$$M_{01} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

→ M12 -

rotation $-90°$ ~~abou~~ around X-axis.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-90°) & -\sin(-90°) \\ 0 & \sin(-90°) & \cos(-90°) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Translation $T = [0, (L2/2)(L1/2)]^T$

$$M12 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & (L2/2) \\ 0 & -1 & 0 & (L1/2) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

\

→ M23 –
Rotation of $-90^\circ$ around x axis.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-90) & -\sin(-90) \\ 0 & \sin(-90) & \cos(-90) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Translation $T = [0 \;\; L3/2 \;\; L2/2]^T$

$$M23 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & L3/2 \\ 0 & -1 & 0 & L2/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

→ M34 –
Seeing rotation by dot-product method.

$$R = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Translation $T = [0 \;\; 0 \;\; L3/2]^T$

$$M34 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & L3/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

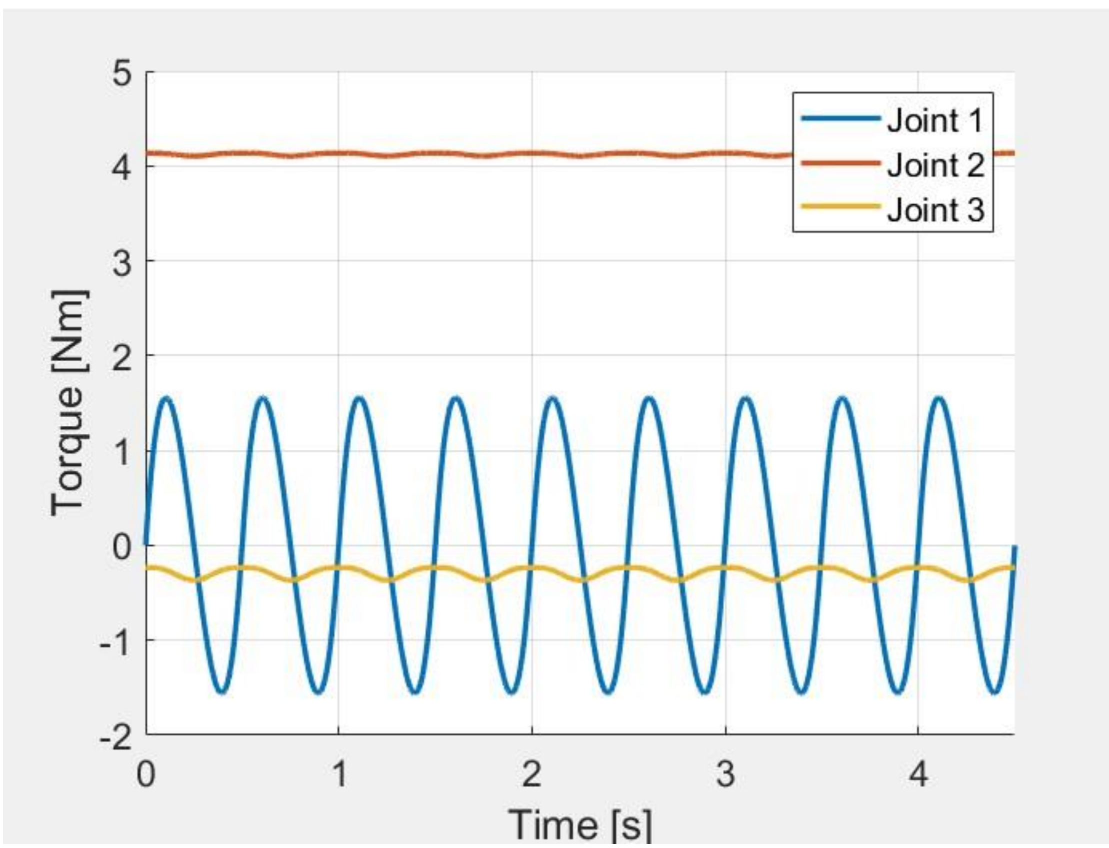
**b**].

O3 = [ 0 0 0; 0 0 0; 0 0 0]
I1 = O3
I2 = O3
I3 = O3
I = [1 0 0; 0 1 0; 0 0 1]
G1 = [I1, O3; O3, (m1*I)] (Spatial Inertia Matrix for link 1)
G2 = [I2, O3; O3, (m3*I)] (Spatial Inertia Matrix for link 2)
G3 = [I3, O3; O3, (m3*I)] (Spatial Inertia Matrix for link 3)

**c**]. I can see a robot collapsing under its own weight. **d**]. Joint Torques:

[0.000000 4.414500 0.000000] Nm

This means, that the actuator at the second joint has to apply a torque of 4.414500 Nm to prevent the robot from collapsing under its own weight and the rest of the joints has no role to play here in this configuration. **e**]. I can see the robot moving along the prescribed trajectory. -TORQUE PROFILE



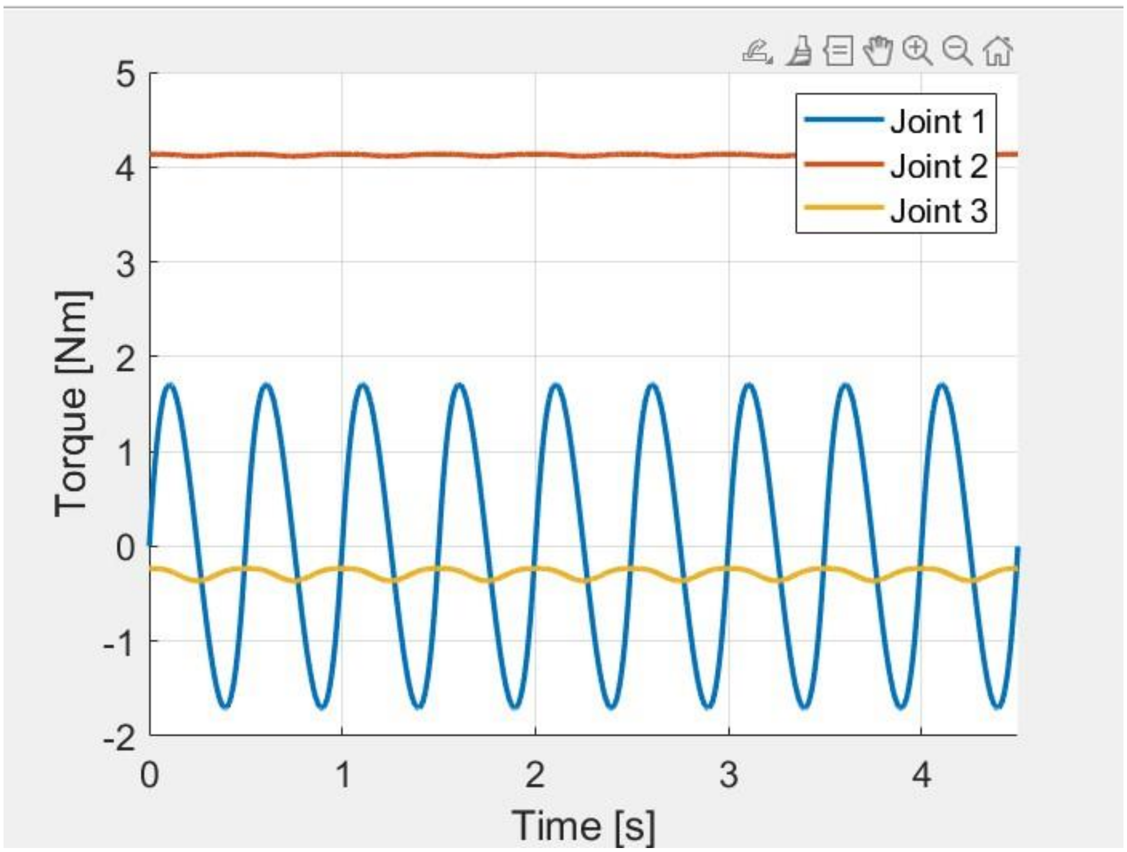

**f**]. O3 = [0 0 0; 0 0 0; 0 0 0]

Ixx1 = (m1*((w*w) + (L1*L1)))/12

Iyy1 = (m1*((l*l) + (L1*L1)))/12
Izz1 = (m1*((l*l) + (w*w)))/12
Ixx2 = (m2*((w*w) + (L2*L2)))/12
Iyy2 = (m2*((l*l) + (L2*L2)))/12
Izz2 = (m2*((l*l) + (w*w)))/12
Ixx3 = (m3*((w*w) + (L3*L3)))/12
Iyy3 = (m3*((l*l) + (L3*L3)))/12
Izz3 = (m3*((l*l) + (w*w)))/12
I1 = Ixx1 0 0; 0 Iyy1 0; 0 0 Izz1]
I2 = [Ixx2 0 0; 0 Iyy2 0; 0 0 Izz2]
I3 = [Ixx3 0 0; 0 Iyy3 0; 0 0 Izz3]

I = [1 0 0; 0 1 0; 0 0 1
G1 = [I1, O3; O3, (m1*I)]; (Spatial Inertia Matrix for link 1)
G2 = [I2, O3; O3, (m3*I)]; (Spatial Inertia Matrix for link 2)
G3 = [I3, O3; O3, (m3*I)]; (Spatial Inertia Matrix for link 3)

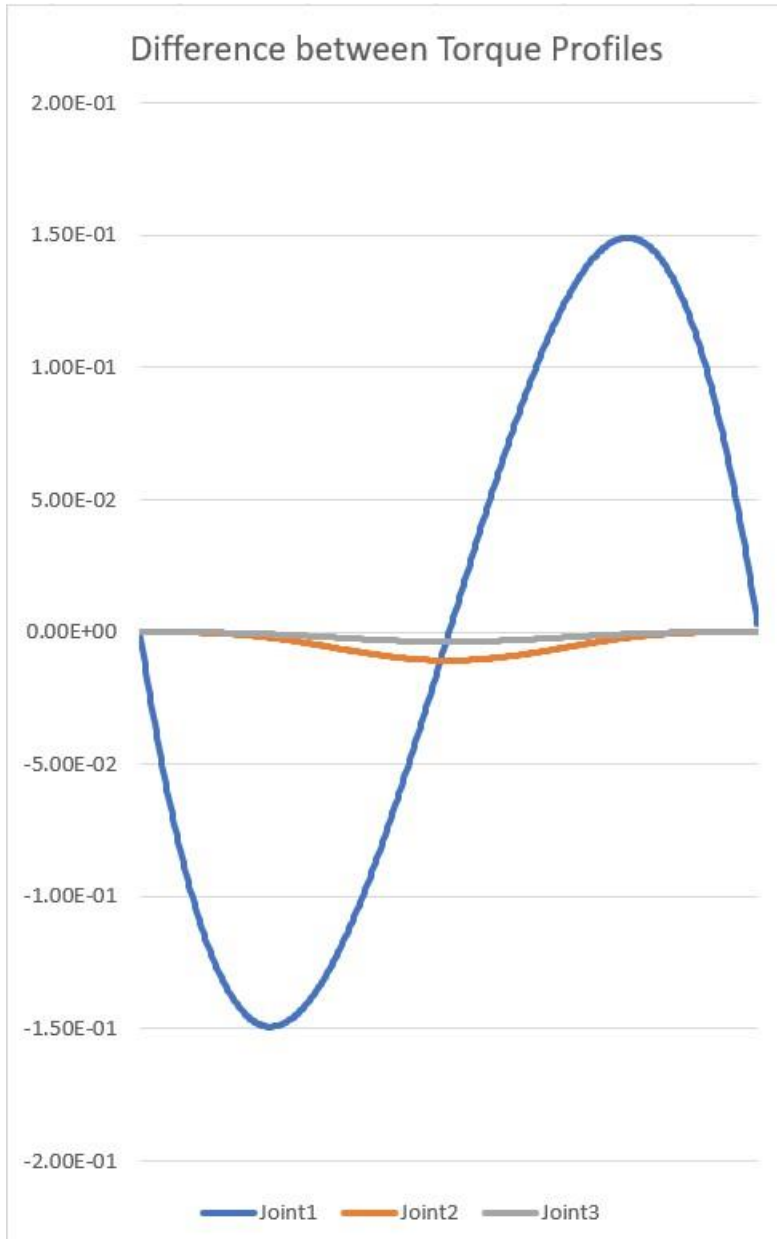
TORQUE PROFILE -



As it can be seen that very little difference is seen between the two torque profiles obtained when all the rotation inertia was zero and when it was considered to be non-zero.

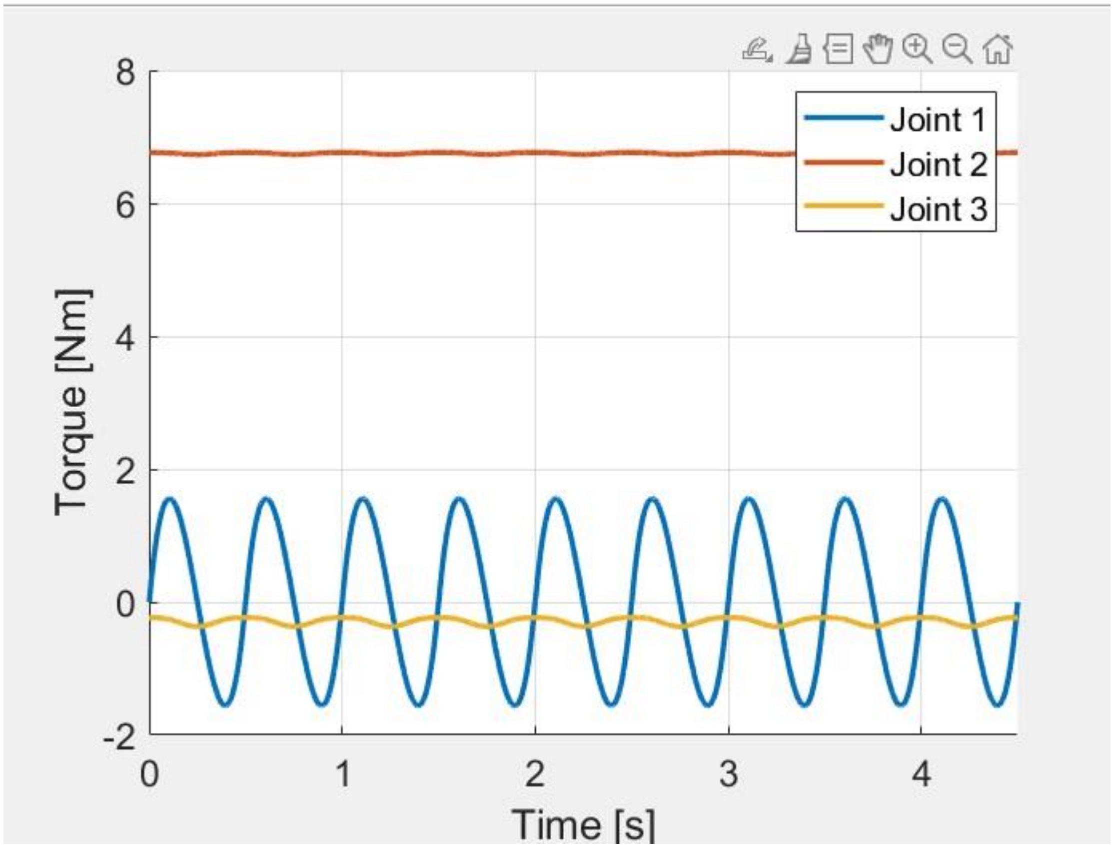
To see the difference between the two torque profiles I have plotted their difference (torque at no rotational inertia – torque at given rotational inertia.) in Microsoft Excel and the following graph was obtained from observing that data.

**Also, as we can see the torque profile is similar when the robot travels from one waypoints to another for all the waypoints. So, in my study I only considered the difference between the torque profiles in both the cases (with and without rotational inertia) only when the robot travels from the first waypoint to the second.
And it is safe to consider the difference between the torque profiles will show similar difference when robot travels to any jth waypoint from ith waypoint in the given setting.



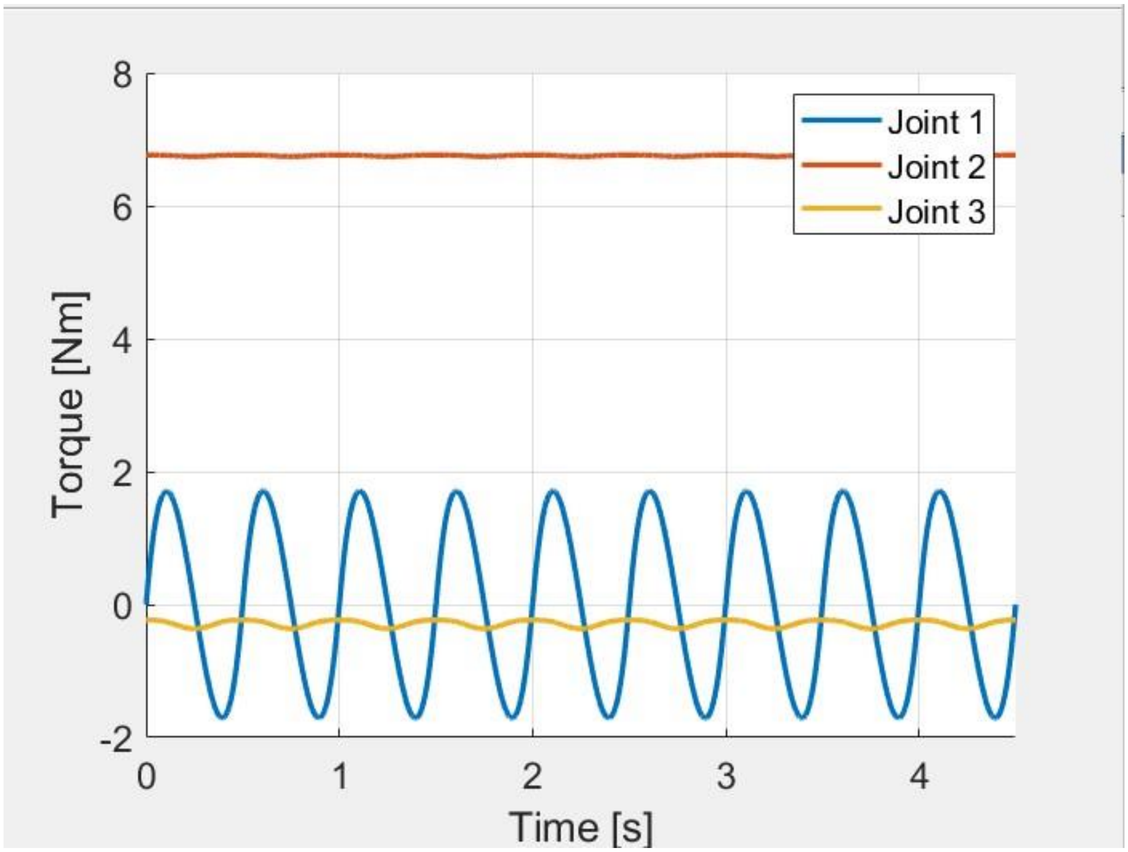

**g**]. To repeat the step [e] without any rotational inertia I modified the Ftipmat (wrench applied by the end-effector on the environment at all points as seen from the end-effector frame.) load = [0 0 0 -9.81 0 0] (in end-effector frame) Ftipmat = repmat(load, [N 1])

Torque profile obtained is as follows –




Now let us add Rotational inertia as given in question [f] and lest run the same code for question [g].

Torque profile is as seen below –

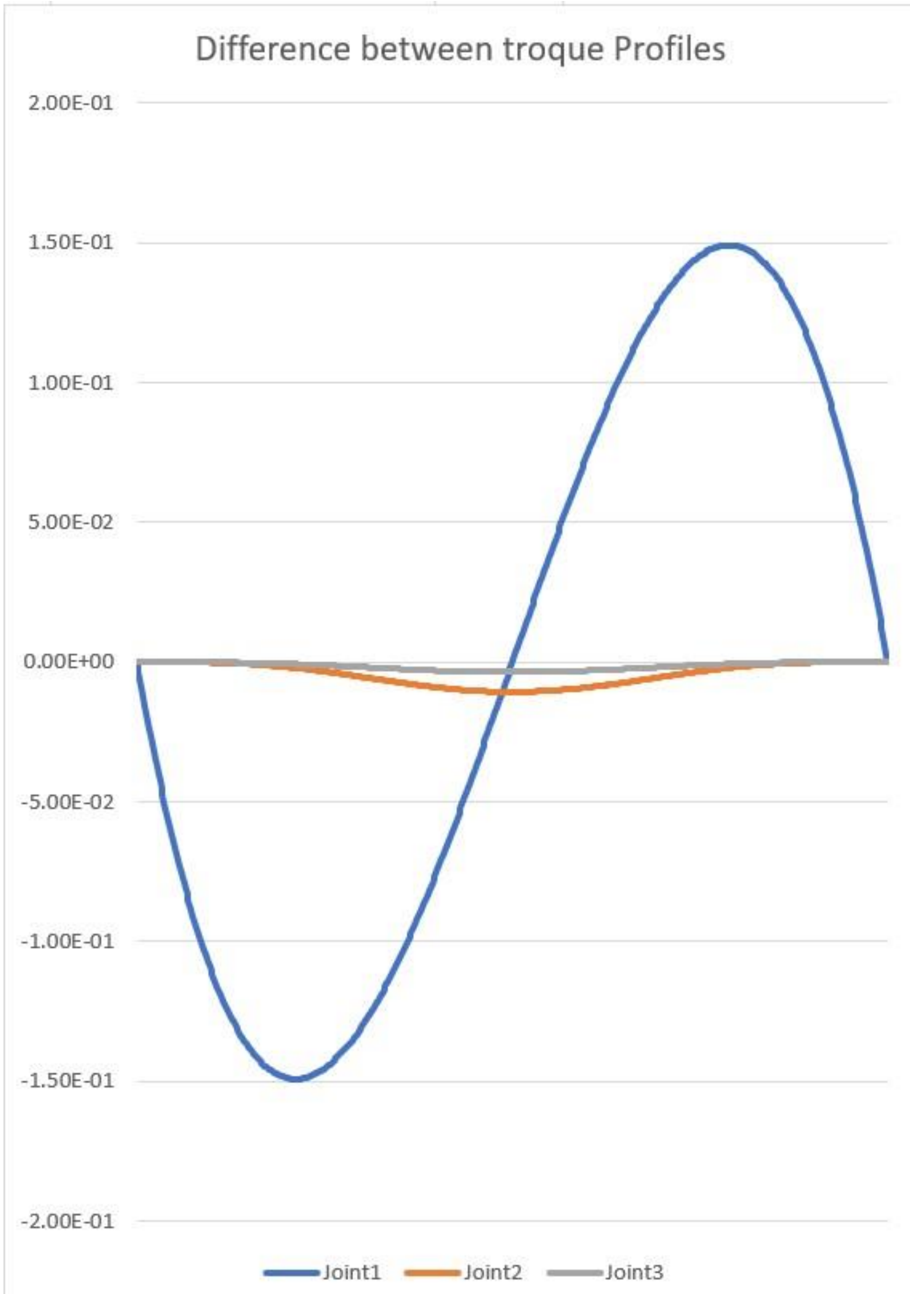



As it can be seen that very little difference is seen between the two torque profiles obtained when all the rotation inertia was zero and when it was considered to be non-zero.

To see the difference between the two torque profiles I have plotted their difference (torque at no rotational inertia – torque at given rotational inertia.) in Microsoft Excel and the following graph was obtained from observing that data.

**Also, as we can see the torque profile is similar when the robot travels from one waypoints to another for all the waypoints. So, in my study I only considered the difference between the torque profiles in both the cases (with and without rotational inertia) only when the robot travels from the first waypoint to the second.
And it is safe to consider the difference between the torque profiles will show similar difference when robot travels to any jth waypoint from ith waypoint in the given setting.

Fenil Desai (WPI ID: 249812226)

**Problem-2:**

**a**].

O3 = [0 0 0; 0 0 0; 0 0 0]

I = [1 0 0; 0 1 0; 0 0 1]
G1 = [Ib1 O3; O3 (m1*I)] (Spatial Inertia Matrix for link 1)
G2 = [Ib2 O3; O3 (m2*I)] (Spatial Inertia Matrix for link 2)
G3 = [Ib3 O3; O3 (m3*I)] (Spatial Inertia Matrix for link 3)
G4 = [Ib4 O3; O3 (m4*I)] (Spatial Inertia Matrix for link 4)
G5 = [Ib5 O3; O3 (m5*I)] (Spatial Inertia Matrix for link 5) G6
= [Ib6 O3; O3 (m6*I)] (Spatial Inertia Matrix for link 6)

**b**].
Inverse Kinematics –

I implemented the Damped-Least-Square method to find out the deltaQ and update it iteratively.

```
                    J_a = jacoba(S_space,M,currentQ);
deltaQ = J_a'*pinv(J_a*J_a' + (lambda^2)*eye(3))*(targetPose - currentPose);
```

here,

- J_a is the analytical Jacobian
- deltaQ is a vector whose elements represent the change in joint values from the previous joint values.
- pinv(k) is Moore-Penrose pseudo-inverse of the matrix k.
- targetPose is twist of the final desire pose of the end effector.
- CurrentPose is the twist of the current pose (at current iteration) of the end effector. - lambda is the damping factor.
- eye(3) give 3*3 Identity matrix

After trial-and-error I found lamda = 0.5 to best fit the solutions and the error smoothly converges to zero when we use this lambda for all the configurations of the target pose.

Quintic Polynomial step –

Here I used N = 50, which defines the number of intermediate steps between two waypoints and hence defines the speed at which the robot moves.

Inverse Dynamics –