ENPM 662 HW3

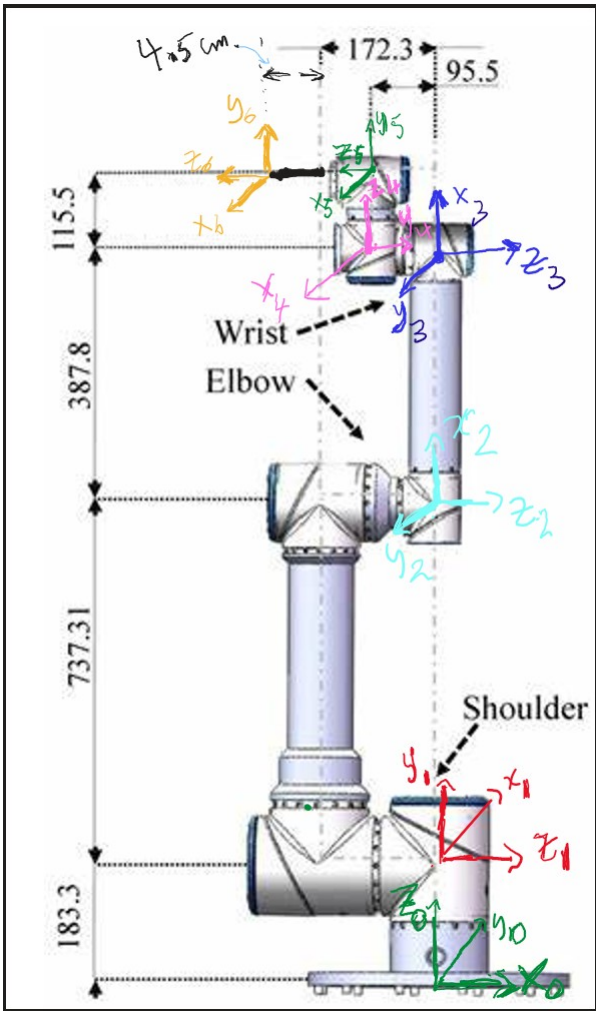
Kent-Diens Joseph

INTRODUCTION

The goal of this assignment was to simulate a UR3 robot with a pen at it's end-effector, drawing a shape consisting of a semi-circle and a rectangle, while maintaining 5 Nm of force in the process, and to calculate the torque at the joints required for this purpose to maintain the robot's weight. The drawing had to be completed in 200 seconds. Plots of each of the torque applied at each of the six joints had to also be made.

CALCULATIONS

The DH frames and DH tables were set-up as shown below.



Frames	Theta [rad]	alpha [rad]	a [m]	d [m]
0 → 1	Theta1 + pi/2	Pi/2	0	0.1833
1 → 2	Theta2 + pi/2	0	0.73731	0
2 → 3	theta3	0	0.3878	0
3 → 4	Theta4 + pi/2	Pi/2	0	-0.0955
4 → 5	theta5	Pi/2	0	0.1155
5 → 6	theta6	0	0	0.1218

From the information in the DH table, the homogeneous transformation matrices for each frames with respect to the base frame was calculated, then the Jacobian matrix was calculated using those matrices and Method 2 from Lecture 8 [2]. Similar to HW2, the drawing shape was split into 5 segments: one half of the semi-circle with radius 5cm, one vertical line of length = 5cm, one 10 cm horizontal line, the second vertical 5cm line, and finally the second half of the semi-circle with radius 5cm. The end-effector pen's starting position was assumed to be at the top of the semi-circle. The time to complete each segments was allocated as 40 seconds to ensure the robot completed the drawing in 200 seconds. This 40 second duration was divided into 1000 points, which resulted in a time_delta of 0.04 seconds used for the numerical integration. Five parametric equations for each line segment was used similar to HW2, with the constant angular velocity of the two halves of the semi-circle, omega, updated to = (pi/2)rad/40 sec, the constant velocity of the two vertical lines were updated to 0.05m/40sec, and that of the horizontal line, to 0.1m/40 sec.

Segment1: First Half of semi-circle with Radius 5 cm

```
x = C_x
y = C_y + R*sp.cos(omega*t +np.pi/2)
z = C_z + R*sp.sin(omega*t +np.pi/2)
```

Segment 2: First Vertical Line on the Left:

```
x2 = C_x
y2 = all_ye_points[-1]
z2 = all_ze_points[-1] - velocity * t  # (motion in -z direction, hence the minus velocity)
```

Segment3: 10cm Horizontal Line:

```
x3 = C_x
y3 = all_ye_points[-1] + velocity * t
z3 = all_ze_points[-1]
```

Segment 4: Second 5cm Vertical Line on the Right:

x4 = C_x

y4 = all_ye_points[-1]

z4 = all_ze_points[-1] + velocity * t

Segment 5: Second Half of the Semi-Circle

x5 = C_x

y5 = (all_ye_points[-1]-R) + R*sp.cos(0+ omga*t)

z5 = all_ze_points[-1] + R*sp.sin(0+ omga*t)

For each of those equations the derivative of the x,y,z with respect to time was taken and combined with the end-effector's angular velocity about the base coordinates wx,wy,wz = (0, 0, 0), to form the 6x1 X_dot column vector. The X_dot for each segment was used in the numerical integration loop with the following equation, to find the joint angles at each delta_t time interval:

q_next = q_current + (inverse_Jacobian*X_dot)*delta_t

The key equation for the robot dynamics is:

$$M(q)q_{ddot} + C(q, q_{dot})q_{dot} + g(q) = tau + (J(q))^T * F$$

where

$$F = [F_x, F_y, F_z, T_x, T_y, T_z]^T$$

For this problem, the UR3 robot is drawing the shape on a plane parallel to the Y0-Z0 plane. As a result the 5Nm that it exerts is in the negative x0 direction, but the reaction force from the wall onto the pen is in the positive x0 direction. Thus the vector F can be written as:

$$F = [5, 0, 0, 0, 0, 0]^T$$

The robot's motion was assumed to be quasi-static, so q_{dot} and q_{ddot} were each assumed to approximately = 0. This resulted in the first to terms in the dynamic equation to also reduce to 0. After further simplifying the equation, an expression for the torque tau was found as follows:

$$tau = g(q) + (J(q))^T * F$$

The equation for g(q) was found from the textbook [3] as:

$$g(q) = \text{partial derivative of } P/ \text{ with respect to } q_i$$

where P is the potential Energy, which is determined using the following equation from the textbook[3]:

$$P = \text{sum } (m_i * g^T * r_{ci}), \text{ for } i=1 \text{ to } i=n$$

In this last equation $g^T = [0, 0, 9.81]$ and is the transpose of the gravity vector, r_{ci} is the coordinate of the center of mass of each link with respect to the base frame, m_i is the mass of each link in kg for i=1 to 6 for the six links.

The mass and center-of-mass coordinates for each link, for the robot without the tool, were found from the Universal Robot website [1]. Those values were adjusted to be with respect to the DH frame set-up for this problem. The table below shows this data:

Link	**DH frame #**	**COM [m]**	**Mass [kg]**
1	1	(0, -0.02, 0)	2
2	2	(-0.13, 0, -0.1157)	3.42
3	3	(-0.05, 0, -0.0238)	1.26
4	4	(0, 0, 0.01)	0.8
5	5	(0, 0, 0.01)	0.8
6	6	(0, 0, -0.02)	0.35

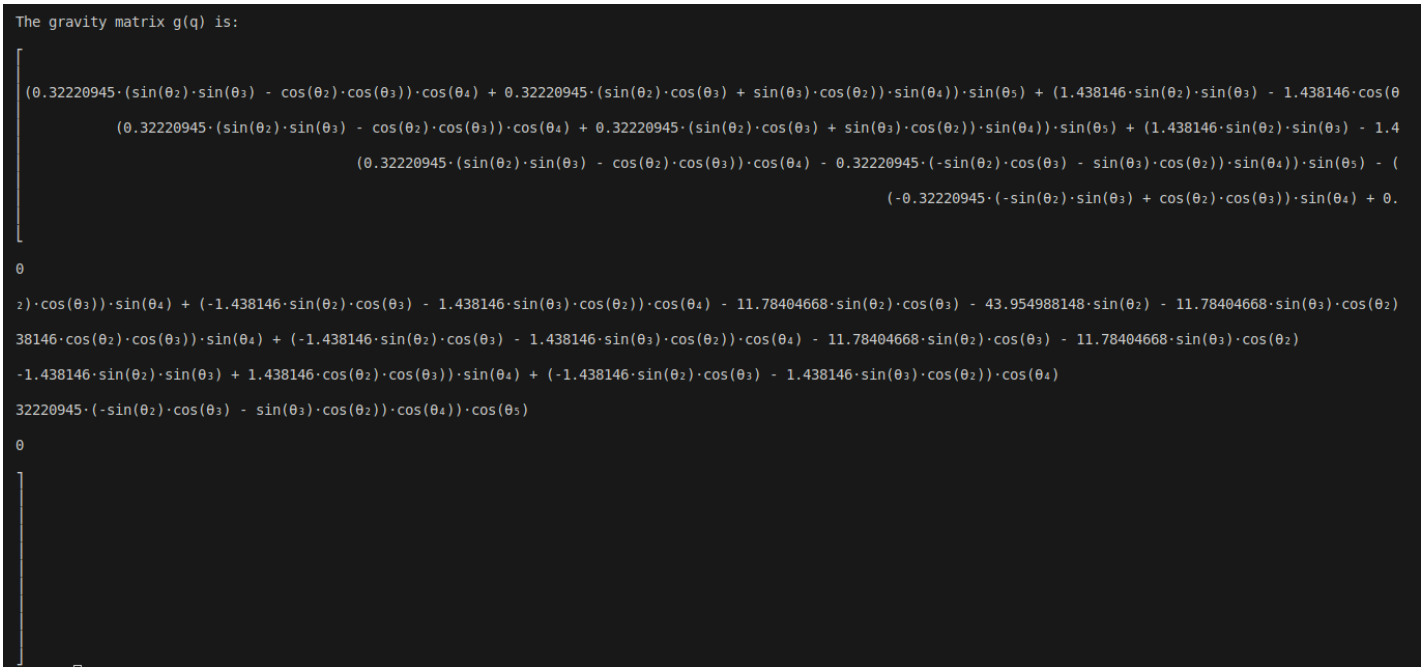
With the tool added, and the last frame shifted to the tip of the pen, the -0.045m was added to the z component of the COM of link 6: (0, 0, -0.065). The mass of the pen was assumed to be 50g or 0.05 kg. The pen was assumed to be a cylinder. As a result, the pen's COM with respect to the frame 6 at the tip of the pen was found to be : (0, 0, -0.045/2) = (0, 0, -0.0225)m. To simplify the calculations, the mass of link 6 and of the pen were combined to equal 0.35+0.05= 0.4 kg. The center-of-mass of the combined mass was found as follows:

combined_COM_x = (COM_LINK6_x * M6 + COM_PEN_x * PEN_MASS)/ (M6 + PEN_MASS)
combined_COM_y = (COM_LINK6_y * M6 + COM_PEN_y * PEN_MASS)/ (M6 + PEN_MASS)
combined_COM_z = (COM_LINK6_z * M6 + COM_PEN_z * PEN_MASS)/ (M6 + PEN_MASS)

This yielded the combined COM of (0, 0, -0.0596875) meters. The table was updated with this result.

Link	DH frame #	COM [m]	Mass [kg]
1	1	(0, -0.02, 0)	2
2	2	(-0.13, 0, -0.1157)	3.42
3	3	(-0.05, 0, -0.0238)	1.26
4	4	(0, 0, 0.01)	0.8
5	5	(0, 0, 0.01)	0.8
6+pen	6	(0, 0, -0.0596875)	0.4

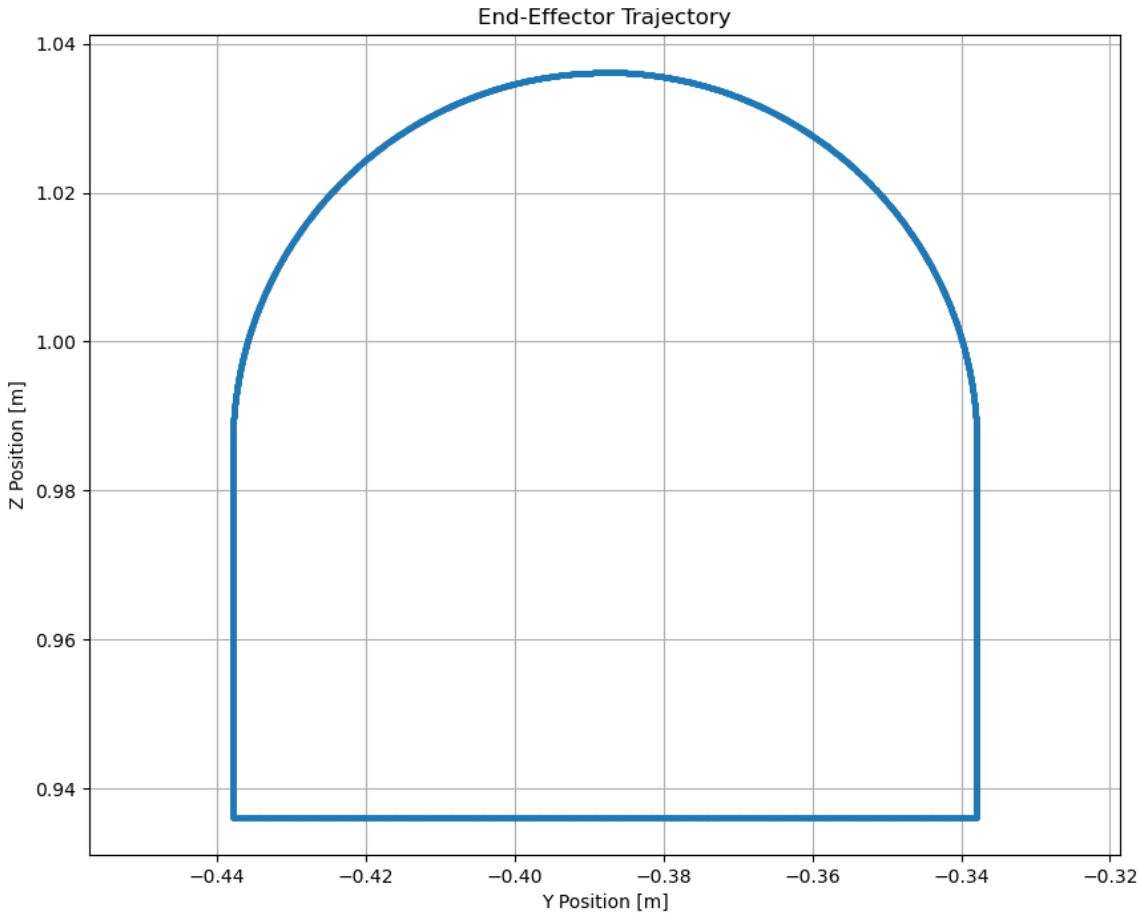
Those COMs were converted to homogeneous forms by adding a 1 after their z-coordinate values. The transformation matrix for each frame with respect to the base frame was then multiplied by each corresponding COM, to find the homogeneous COM with respect to the base frame. The COM coordinates were then found by taking the top 3 values in the resulting homogeneous 4x1 vector. These COMs and the masses where then used in the Potential energy equation. The gravity matrix was then found by taking the partial derivative of the potential energy with respect to each joint angle (theta1 to theta6). The image below shows the gravity matrix print-out from the terminal.



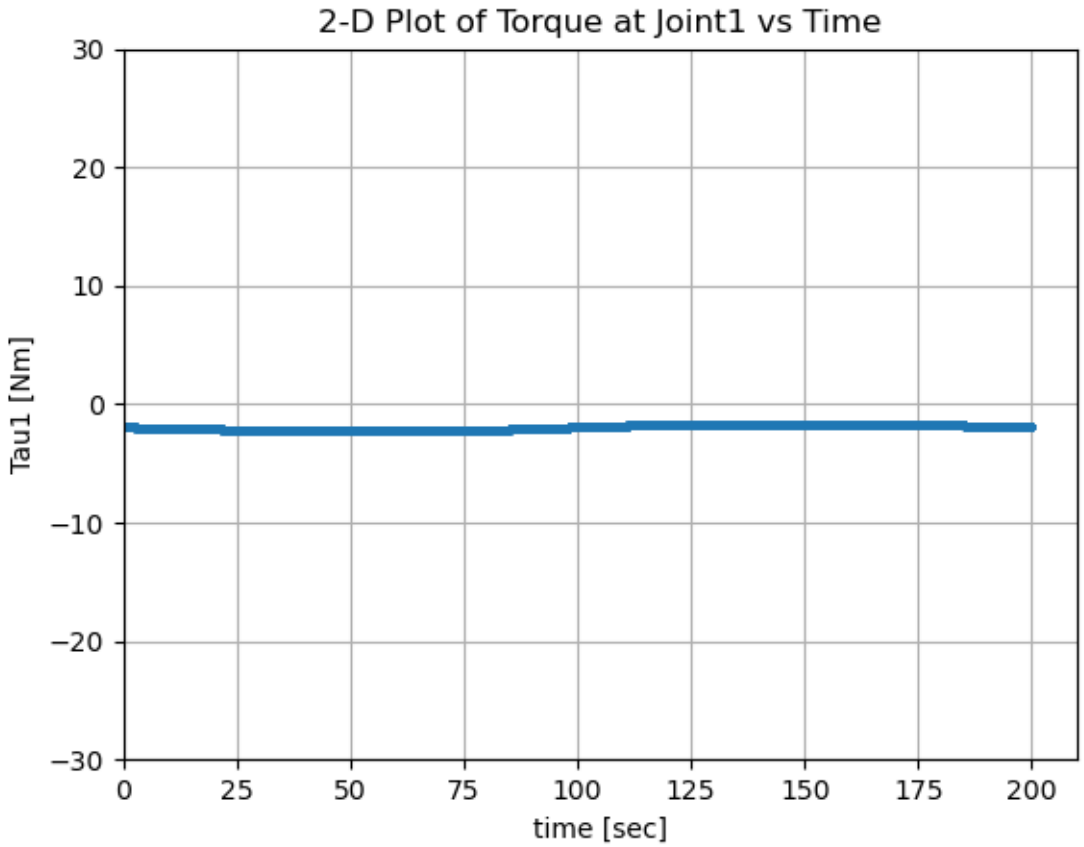
Finally, the gravity matrix along with the Jacobian matrix, which was found in hw2, was used to find the equation for the joint torque vector (see code snippet in the Appendix section). This equation was then used to find the initial joint torque angles at he initial joint angles (tht1, tht2, tht3, tht4, tht5, tht6 = 0.0, 0.0, pi/2, -pi/2, 0.0, 0.0). The joint torques were stored into a lists. The same Euler method of numerical integration that was used in HW2 was used to find the joint angles at each time delta, then the torque and end-effector positions were calculated and stored into lists. The lists containing the end-effector positions where used to draw the shape, and the lists containing the joints torque where used to plot them versus time. The following shows the plots of the shape dawned by the pen, followed by the joint torques.

Plots

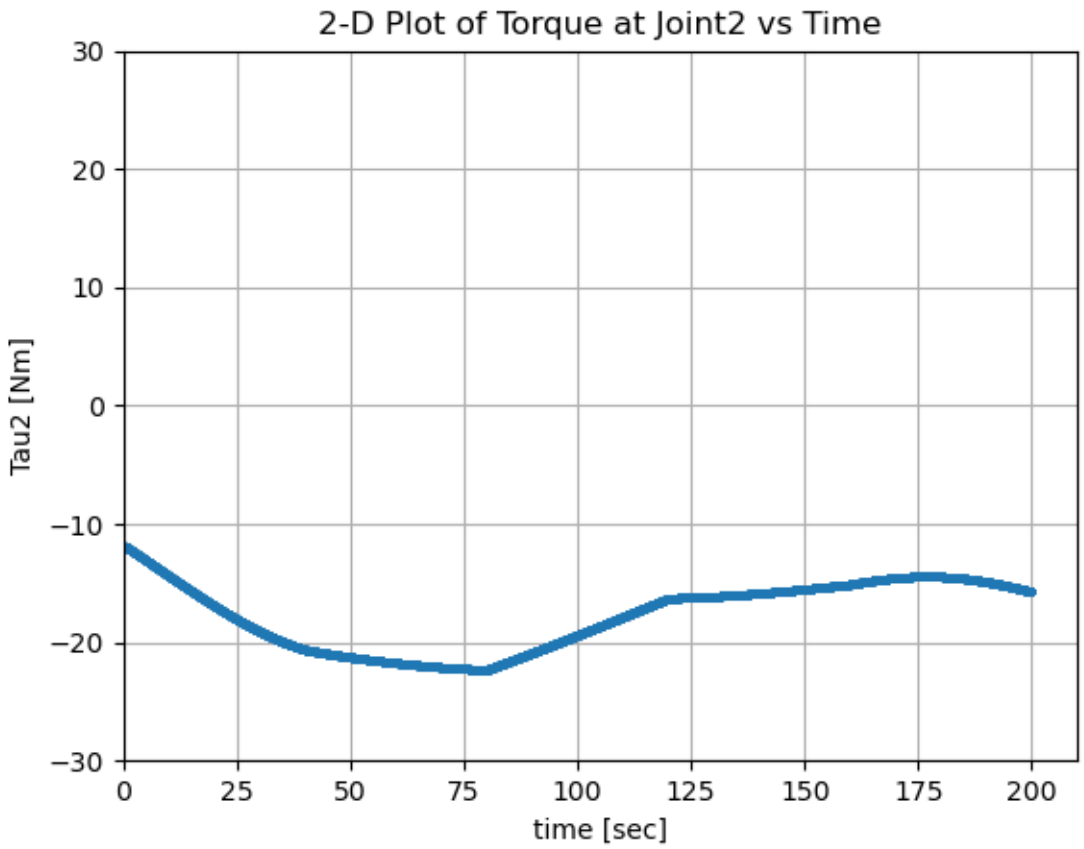
Plot of the end-effector pen drawing



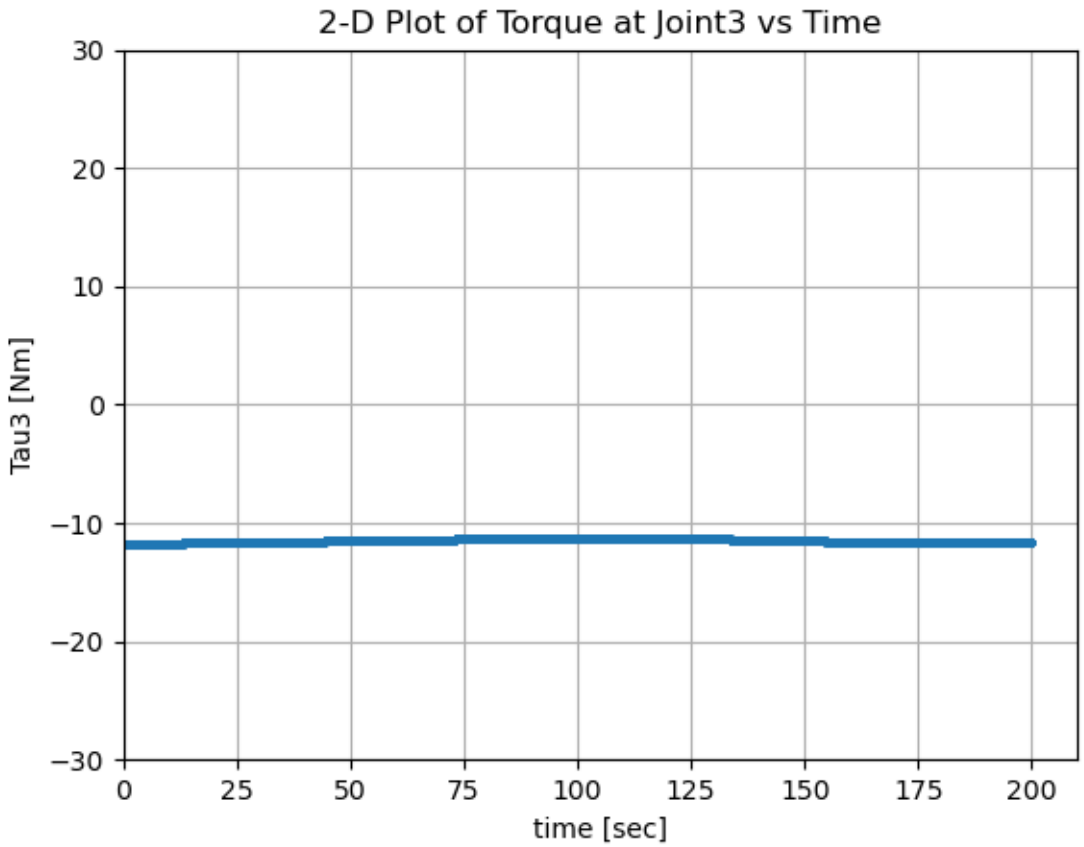
Joint 1 Torque Plot



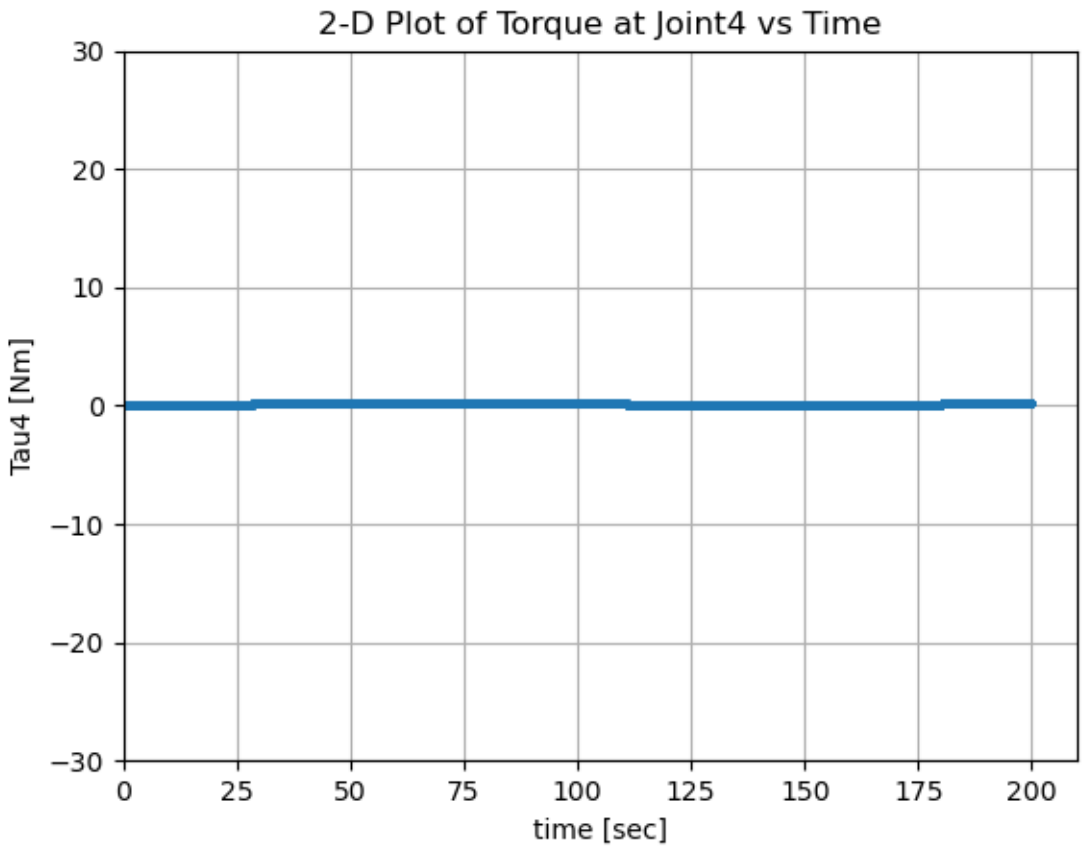
Joint2 Torque Plot



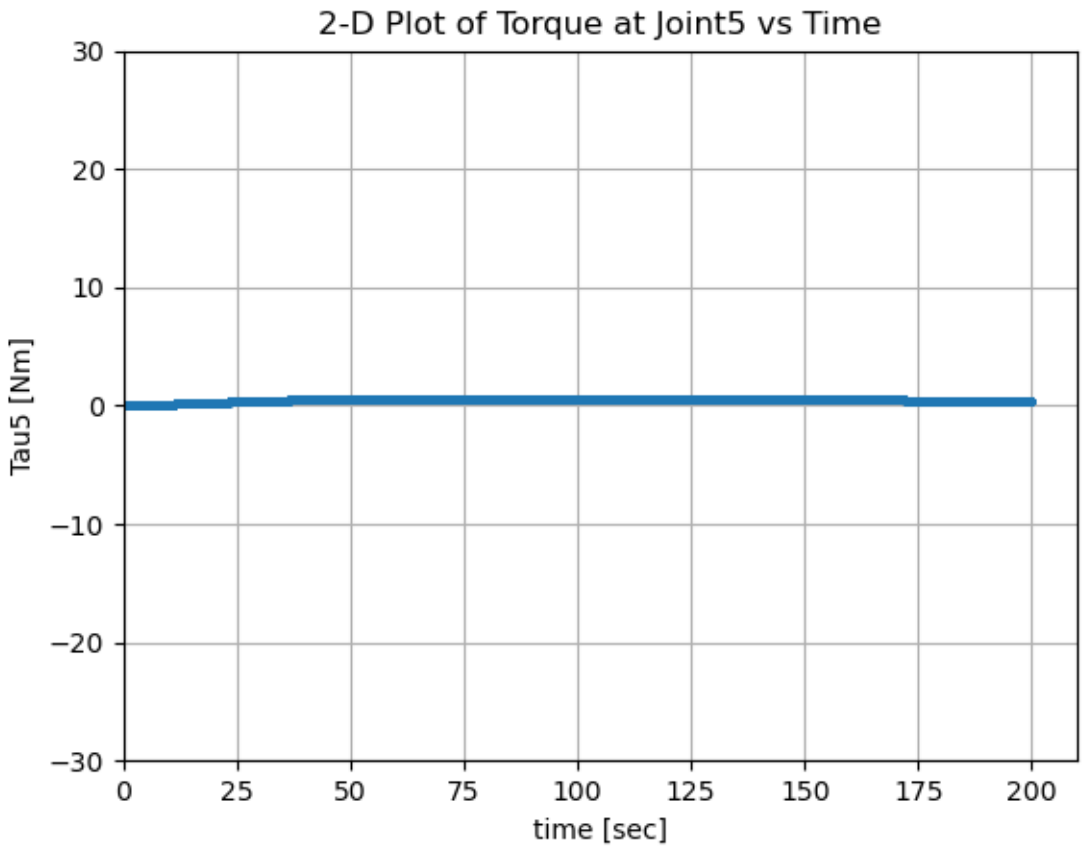
Joint 3 Torque Plot



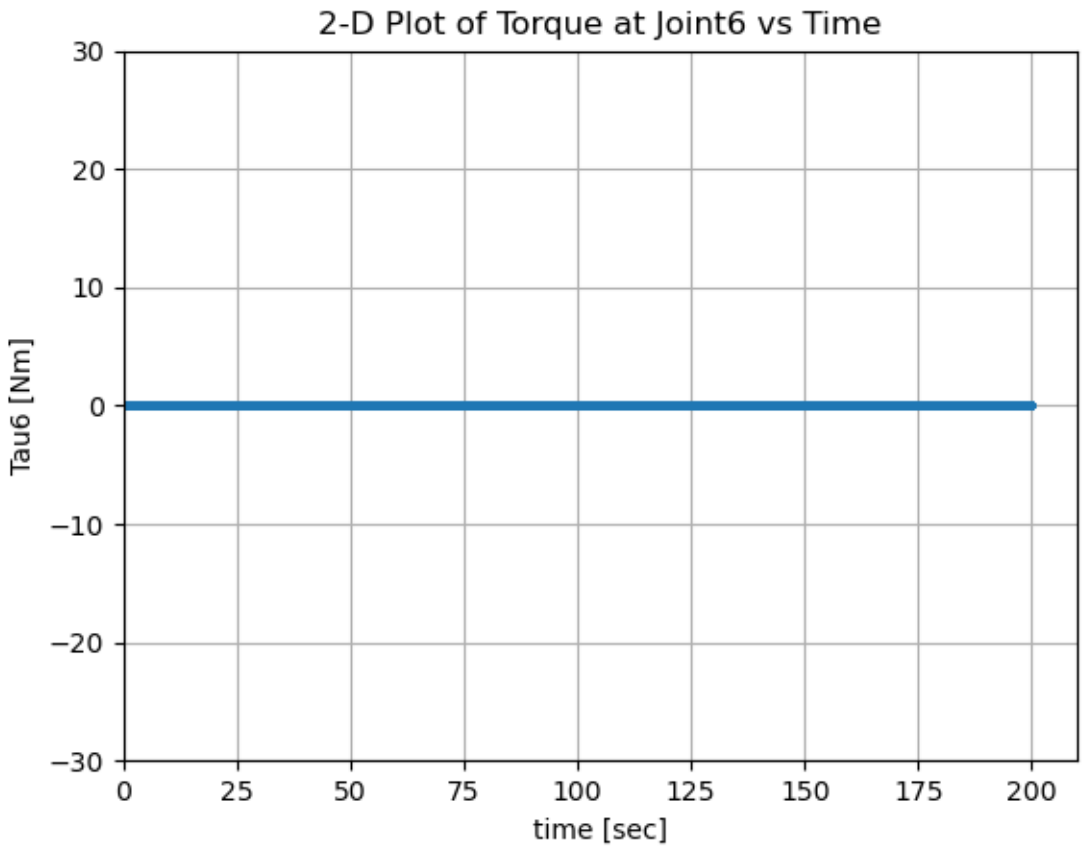
Joint 4 Torque Plot



Joint 5 Torque Plot



Joint 6 Torque



The following is a summary of the data points in the plot that is printed-out when the code is run.

The max torque at joint1 is -1.69 Nm, the minimum torque: -2.19 Nm, and the mean: -1.9394 Nm

The max torque at joint2 is -11.78 Nm, the minimum torque: -22.32 Nm, and the mean: -17.6579 Nm

The max torque at joint3 is -11.25 Nm, the minimum torque: -11.78 Nm, and the mean: -11.4905 Nm

The max torque at joint4 is 0.22 Nm, the minimum torque: -0.0 Nm, and the mean: 0.1462 Nm

The max torque at joint5 is 0.6 Nm, the minimum torque: -0.0 Nm, and the mean: 0.4846 Nm

The max torque at joint6 is 0.0 Nm, the minimum torque: 0.0 Nm, and the mean: 0.0 Nm

CONCLUSION

In conclusion, using the dynamic equations, and the forward and inverse kinematics equations, the simulation of the UR3 drawing the shape while maintaining 5Nm of force on the wall was successful. Based on the summary data, the joint 2 had the largest absolute value in torque of 17.6579 Nm. This makes sense considering the fact that joint 3 had an initial angle of 90 degrees and joint 4 an initial angle of -90 degrees, and the fact that joint 2's axis of rotation is perpendicular to the plane where the drawing took place, joint 2 had to be actuated with the most torque to help the movement of all the links above it.

REFERENCES

1- https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/

2- Monfaredi, Reza, ENPM 662 Lecture 8

3- Spong, Mark, Robot Modeling and Control.

APPENDIX

Dynamics Equations & Calculations From the Code

```
# Adding 1 at the end of each COM coordinate to convert them into homogeneous form.

for link in COM_WITH_PEN.keys():
COM_WITH_PEN[link].append(1)

# Homogeneous center-of-mass coordinates, converted to column vectors w.r.t the base
frame
r1_0h = transforms_wrt_base[0] * sp.transpose(sp.Matrix([COM_WITH_PEN['link1']]))
r2_0h = transforms_wrt_base[1] * sp.transpose(sp.Matrix([COM_WITH_PEN['link2']]))
r3_0h = transforms_wrt_base[2] * sp.transpose(sp.Matrix([COM_WITH_PEN['link3']]))
r4_0h = transforms_wrt_base[3] * sp.transpose(sp.Matrix([COM_WITH_PEN['link4']]))
r5_0h = transforms_wrt_base[4] * sp.transpose(sp.Matrix([COM_WITH_PEN['link5']]))
r6_0h = transforms_wrt_base[5] * sp.transpose(sp.Matrix([COM_WITH_PEN['link6_plus_pen']]))
# Taking the x,y,z coordinates by taking the top 3 elements
R1_0 = r1_0h[0:3,-1]
R2_0 = r2_0h[0:3,-1]
R3_0 = r3_0h[0:3,-1]
R4_0 = r4_0h[0:3,-1]
R5_0 = r5_0h[0:3,-1]
R6_0 = r6_0h[0:3,-1]

# gravity
gravity = sp.symbols("gravity")
gravity = sp.Matrix([[0.0], [0.0], [9.81]])
gravity_T = sp.transpose(gravity)
# Potential Energy potential_energy = (m*g^T)*r_cm
potential_energy = sp.symbols("potential_energy")
potential_energy = M1*gravity_T*R1_0 + M2*gravity_T*R2_0 + M3*gravity_T*R3_0 +
M4*gravity_T*R4_0 + M5*gravity_T*R5_0 + M6_PLUS_PEN*gravity_T*R6_0
gravity_matrix = sp.Matrix([[sp.diff(potential_energy, theta1)], [sp.diff(potential_energy,
theta2)], [sp.diff(potential_energy,theta3)],
[sp.diff(potential_energy, theta4)], [sp.diff(potential_energy, theta5)],
[sp.diff(potential_energy,theta6)]])

# force vector F = [Fx, Fy, Fz, Tx, Ty, Tz]
force = sp.symbols("force")
force = sp.Matrix([[5.0], [0.0], [0.0], [0.0], [0.0], [0.0]])
#Torque
torque = sp.symbols("torque")
torque = gravity_matrix - sp.transpose(jacobian)*force
```