EE5106/ME5402 Advanced Robotics

Part 1 CA- Group 2

AY 2021/2022 Semester 2

Group Members:

Del Rosario Justin Kenneth Macatangay (A0247107J)

Liu Weihao (A0232935A)

Voo Zhi Wei (A0140452U)

Solutions for Question 1:

Q1(A) solution:

The DH frame mapping and assignment is shown below for the PUMA 600 manipulator:

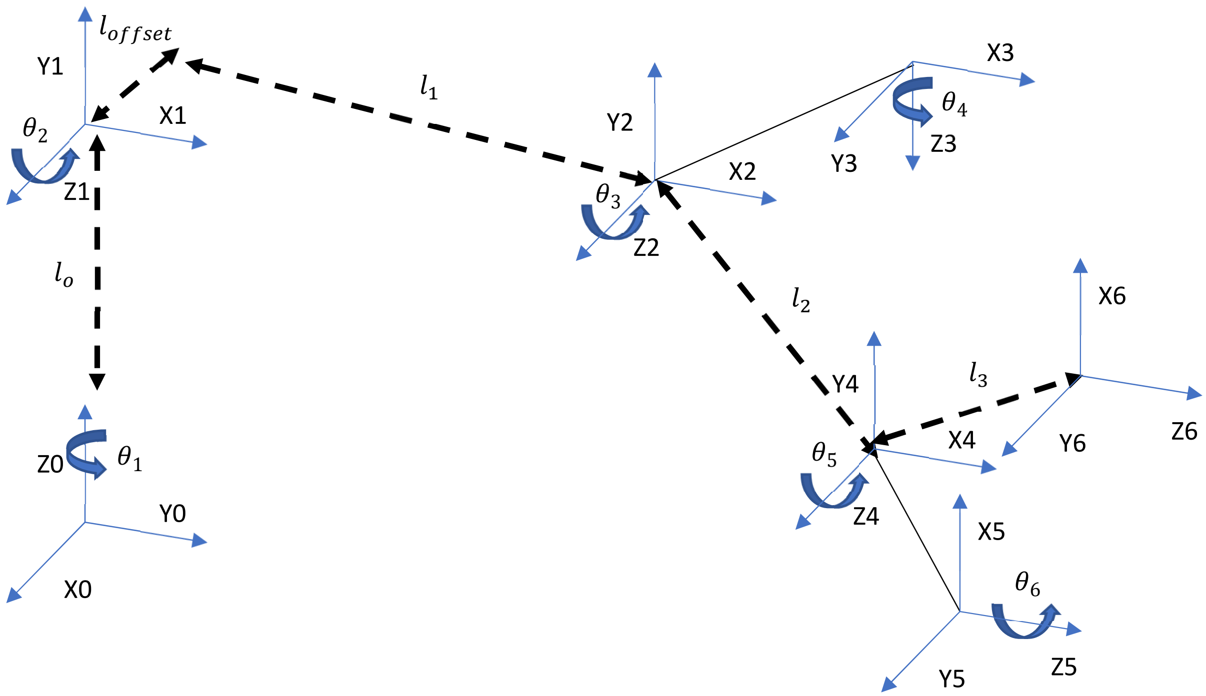


Figure 1: DH Frame Mapping for PUMA 600 Robot

After mapping the frame orientations and locations, the DH parameters table is defined below at table 1:

Table 1: DH Parameters Table

Link number, i	θ_i	α_i(in degrees)	d_i	a_i
1	θ_1	90	l_o	0
2	θ_2	0	$-l_{offset}$	l_1
3	θ_3	90	0	0
4	θ_4	-90	l_2	0
5	θ_5	90	0	0
6	θ_6	0	l_3	0

For this solution, the following variables of the manipulator are assigned with values below. These lengths are based on the reference paper "An exact kinematic model of PUMA 600 manipulator" (listed at References).

$$l_o = 0.660\ m \quad l_{ooffset} = 0.149\ m \quad l_1 \& l_2 = 0.432\ m \quad l_3 = 0.05639\ m$$

Using the values for each link on the D-H parameters table, the homogenous transformation matrix for each link is derived. The general form of the transformation for a link in between joint i and joint i-1 is as follows:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Using the MATLAB code generated for Question 1(A) to programmatically derive all homogenous transformation matrices for each link, the program results are shown below:

$$ {}^{0}_{1}T = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 c_2 \\ s_2 & c_2 & 0 & l_1 s_2 \\ 0 & 0 & 1 & -l_{offset} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{3}_{4}T = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}T = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The code also generates the forward kinematic equation of the manipulator by solving the equation below:

$$ {}^{0}_{6}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T $$

$$ {}^{0}_{6}T = \begin{bmatrix} n_x & t_x & b_x & p_x \\ n_y & t_y & b_y & p_y \\ n_z & t_z & b_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The equation of each element of the forward kinematic matrix are specified below:

$$ n_x = s_6(s_1c_4 - c_1s_4c_{23}) + c_6(c_5(c_1c_4c_{23} + s_1s_4) - c_1s_5s_{23}) $$

$$ n_y = -s_6(s_1s_4c_{23} + c_1c_4) - c_6(s_1s_5s_{23} + c_5(c_1s_4 - s_1c_4c_{23})) $$

$$ n_z = c_6(c_4c_5s_{23} + s_5c_{23}) - s_4s_6s_{23} $$

$$ t_x = c_6(s_1c_4 - c_1s_4c_{23}) - s_6(c_5(c_1c_4c_{23} + s_1s_4) - c_1s_5s_{23}) $$

$t_y = s_6(c_5(c_1s_4 - s_1c_4c_{23}) + s_1s_5s_{23}) - c_6(s_1s_4c_{23} + c_1c_4)$

$t_z = -s_6(c_4c_5s_{23} + s_5c_{23}) - c_6s_4s_{23}$

$b_x = c_1c_5s_{23} + s_5(c_1c_4c_{23} + s_1s_4)$

$b_y = s_1c_5s_{12} - s_5(c_1s_4 - s_1c_4c_{23})$

$b_z = c_4s_5s_{23} - c_5c_{23}$

$p_x = l_1c_1c_2 + l_2c_1s_{23} - l_{offset}s_1 + l_3(s_5(c_1c_4c_{23} + s_1s_4) + c_1c_5s_{23})$

$p_y = l_2s_1s_{23} + l_1c_2s_1 + l_{offset}c_1 - l_3(s_5(c_1s_4 - s_1c_4c_{23}) - s_1c_5s_{23})$

$p_z = l_1s_2 + l_0 - l_2c_{23} - l_3(c_5c_{23} - c_4s_5s_{23})$

Q1(B) solution:

For solving inverse kinematics of PUMA 600, the end-effector transformation matrix below will serve as input:

$$ {}^0_6T = \begin{bmatrix} 0.8750 & -0.4330 & 0.2165 & -0.0378 \\ -0.2165 & -0.7500 & -0.6250 & 0.1762 \\ 0.4330 & 0.5000 & -0.7500 & 0.4596 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \text{T-End Effector} $$

In approaching the inverse kinematics, the frame which can be considered as the wrist point of the manipulator is identified. In this case, the origin of frame 4 will be used as wrist point and will be identified as point W.

Let T-end effector matrix be:

$$ {}^0_6T = \begin{bmatrix} n_x & t_x & b_x & p_x \\ n_y & t_y & b_y & p_y \\ n_z & t_z & b_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

To get position and orientation of point W with respect to frame 1:

$$ {}^0_1T^{-1}\,{}^0_6T\,{}^5_6T^{-1}\,{}^4_5T^{-1} = {}^1_4T = {}^1_2T\,{}^2_3T\,{}^3_4T \rightarrow \text{Eq. 1.0} $$

Solving for the 4th column of matrix at the LHS of equation 1.0:

$$\begin{bmatrix} \frac{P_y S_1}{C_1^2+S_1^2} - l_3\left(\frac{b_x C_1}{C_1^2+S_1^2} + \frac{b_y S_1}{C_1^2+S_1^2}\right) + \frac{P_x C_1}{C_1^2+S_1^2} \\ P_z - l_3 b_z - l_o \\ l_3\left(\frac{b_y C_1}{C_1^2+S_1^2} - \frac{b_x S_1}{C_1^2+S_1^2}\right) + \frac{P_x S_1}{C_1^2+S_1^2} - \frac{P_y C_1}{C_1^2+S_1^2} \\ 1 \end{bmatrix}$$

Simplifying the equations, the LHS 4th column is equal to: $(C_1^2 + S_1^2 = 1)$

$$\begin{bmatrix} P_y S_1 - l_3 b_x C_1 + l_3 b_y S_1 + P_x C_1 \\ P_z - l_3 b_z - l_o \\ l_3 b_y C_1 - l_3 b_x S_1 + P_x S_1 - P_y C_1 \\ 1 \end{bmatrix}$$

Solving for the 4th column of matrix ${}^{1}_{4}T$ which is at the RHS of equation 1.0:

$$\begin{bmatrix} l_2(C_2 S_3 + C_3 S_2) + l_1 C_2 \\ l_1 S_2 - l_2(C_2 C_3 - S_2 S_3) \\ -loffset \\ 1 \end{bmatrix}$$

Simplifying the equations:

$$\begin{bmatrix} l_2(S_{23}) + l_1 C_2 \\ l_1 S_2 - l_2(C_{23}) \\ -loffset \\ 1 \end{bmatrix}$$

To get θ_1, we equate 3rd equation of LHS and 3rd equation of RHS:

$$l_3 b_y C_1 - l_3 b_x S_1 + P_x S_1 - P_y C_1 = -l_{offset}$$

$$(l_3 b_y - p_y)C_1 + (P_x - l_3 b_x)S_1 = -l_{offset} \rightarrow \text{Eq. 1.1}$$

Equation 1.1 is in the form $acos\theta + bsin\theta = c$ which has a general solution of:

$$\theta_1 = cos^{-1}\left(\frac{c}{\pm\sqrt{a^2+b^2}}\right) + tan^{-1}\left(\frac{b}{a}\right) \rightarrow Eq. 1.2$$

θ_1 can be found by utilizing equation 1.2 with:

$$a = l_3 b_y - p_y \rightarrow \text{Eq. } 1.3$$
$$b = P_x - l_3 b_x \rightarrow \text{Eq. } 1.4$$
$$c = -l_{offset} \rightarrow \text{Eq. } 1.5$$

To get θ_2, equate the remaining equations of LHS 4th column Eq. 1.0 and remaining equations of RHS 4th column Eq. 1.0:

$$P_y S_1 - l_3 b_x C_1 + l_3 b_y S_1 + P_x C_1 = l_2(S_{23}) + l_1 C_2$$
$$P_z - l_3 b_z - l_o = l_1 S_2 - l_2(C_{23})$$

Let:

$$A = P_y S_1 - l_3 b_x C_1 + l_3 b_y S_1 + P_x C_1$$
$$B = P_z - l_3 b_z - l_o$$

Then:

$$l_2(S_{23}) = A - l_1 C_2 \rightarrow \text{Eq. } 1.6$$
$$l_2(C_{23}) = l_1 S_2 - B \rightarrow \text{Eq. } 1.7$$

Squaring both sides of Eq. 1.6 and 1.7 then adding their LHS and RHS:

$$l_2^2 S_{23}^2 + l_2^2 C_{23}^2 = A^2 - 2Al_1 C_2 + l_1^2 C_2^2 + l_1^2 S_2^2 - -2l_1 C S_2 B + B^2$$
$$l_2^2(l_2^2 S_{23}^2 + C_{23}^2) = A^2 + B^2 + l_1^2(S_2^2 + C_2^2) - 2l_1(AC_2 + BS_2)$$
$$AC_2 + BS_2 = \frac{A^2+B^2+l_1^2-l_2^2}{2l_1} \rightarrow \text{Eq. } 1.8$$

Equation 1.8 is in the form $acos\theta + bsin\theta = c$ which has a general solution of:

$$\theta_2 = cos^{-1}\left(\frac{c}{\sqrt{a^2+b^2}}\right) + tan^{-1}\left(\frac{b}{a}\right) \rightarrow Eq. 1.9$$

θ_2 can be found by utilizing equation 1.6 with:

$$a = A \rightarrow Eq. 1.10$$
$$b = B \rightarrow \text{Eq. } 1.11$$
$$c = \frac{A^2+B^2+l_1^2-l_2^2}{2l_1} \rightarrow \text{Eq. } 1.12$$

To solve for θ_3, utilize eq. 1.7:

$$l_2(C_{23}) = l_1 S_2 - B$$
$$C(\theta_2 + \theta_3) = \frac{l_1 S_2 - B}{l_2}$$

$$(\theta_3) = cos^{-1}(\frac{l_1S_2 - B}{l_2}) - \theta_2 \rightarrow \text{eq. 1.13}$$

Solving for the remaining joint angles θ_4, θ_5 and θ_6, the transformation matrix ${}^3_6T$ is defined with the following equation below:

$${}^3_6T = {}^0_3T^{-1}\,{}^0_6T \rightarrow \text{ eq. 1.14}$$

Using MATLAB code, we get the value of ${}^3_6T$ which is:

$${}^3_6T = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -S_4C_6 - C_4S_5C_6 & C_4S_5 & l_3C_4S_5 \\ C_4S_6 + S_4C_5C_6 & C_4C_6 - S_4C_5S_6 & S_4S_5 & l_3S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 & l_2 + l_3C_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \text{eq. 1.15}$$

Solving RHS of eq. 1.14, we define the result as:

$${}^0_3T^{-1}\,{}^0_6T = \begin{bmatrix} n'_x & t'_x & b'_x & p'_x \\ n'_y & t'_y & b'_y & p'_y \\ n'_z & t'_z & b'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \text{eq. 1.16}$$

The value of ${}^0_3T^{-1}$ can be solved already since the values of $\theta_1, \theta_2, \theta_3$ are already known at this point. ${}^0_6T$ is the end effector transformation matrix which is given at the start of the inverse kinematics process.

To get the remaining joint angles, the elements at the LHS of eq. 1.14 are equated to the elements at the RHS of eq. 1.14. For θ_5, element (3,3) of LHS is equal to element (3,3) of RHS:

$$\theta_5 = cos^{-1}(b'_z) \rightarrow \text{eq. 1.17}$$

For θ_4, element (1,3) of LHS is equal to element (1,3) of RHS:

$$\theta_4 = cos^{-1}(\frac{b'_x}{S_5}) \rightarrow \text{eq. 1.18}$$

For θ_6, element (3,1) of LHS is equal to element (3,1) of RHS:

$$\theta_6 = cos^{-1}(-\frac{n'_z}{S_5}) \rightarrow \text{eq.1.19}$$

The equations to obtain all joint angles for a given end-effector matrix are implemented in the MATLAB code. Using the end-effector matrix as input, the code should provide all angle set solutions that will allow the manipulator to reach the desired end-effector position along with plots of the manipulator position and orientation for each set solution. The end-effector matrix input is shown below again:

$${}^0_6T = \begin{bmatrix} 0.8750 & -0.4330 & 0.2165 & -0.0378 \\ -0.2165 & -0.7500 & -0.6250 & 0.1762 \\ 0.4330 & 0.5000 & -0.7500 & 0.4596 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The MATLAB code output provides the following angle solution set with values in terms of degrees for the given end effector T matrix:

```
teta_values_up =

  -33.3981  -50.2998  -63.4916   40.3114  141.5123   14.3981
  -33.3981  -50.2998  -63.4916 -139.6886 -141.5123 -165.6019
  -33.3981 -203.7914  243.4916  100.5949   24.1800 -120.7782
  -33.3981 -203.7914  243.4916  -79.4051  -24.1800   59.2218
 -299.9890   19.1525  -65.1991   64.4188   33.6587   29.9164
 -299.9890   19.1525  -65.1991 -115.5812  -33.6587 -150.0836
 -299.9890 -136.0466  245.1991  138.5673  130.9349 -120.0300
 -299.9890 -136.0466  245.1991  -41.4327 -130.9349   59.9700
```

There are 8 sets of angles that can reach the same position and orientation of the given end-effector. Each row of “teta_values_up” represent 1 solution set and the angles are arranged from left to right starting from $\theta_1$to θ_6.

The X-Y, Y-Z and X-Z plots of the manipulator position and orientation for each solution set are shown below:

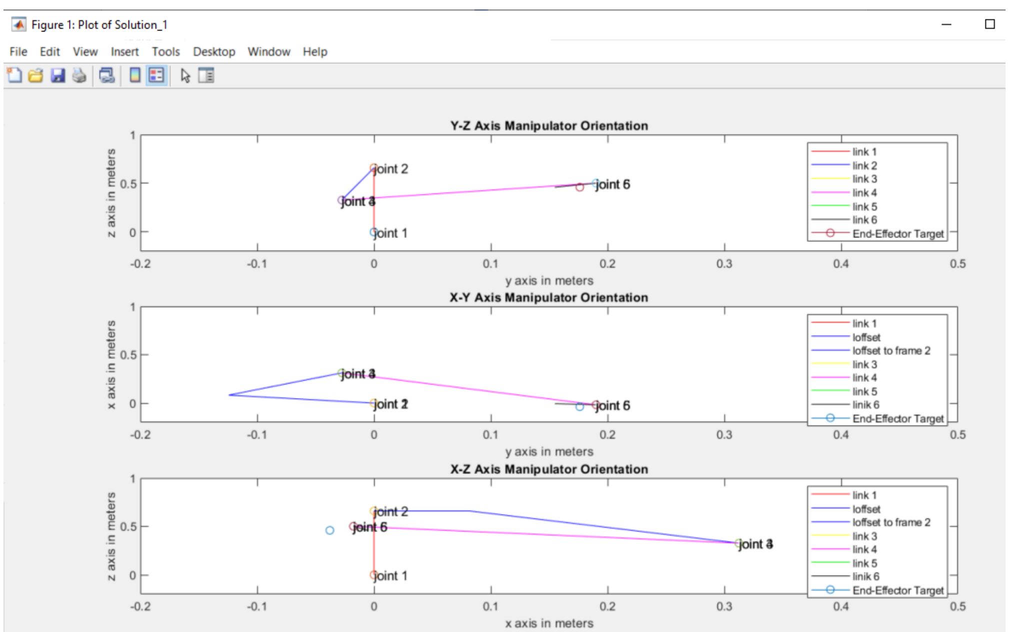


Figure 2: Manipulator Orientation Plot Solution Set 1

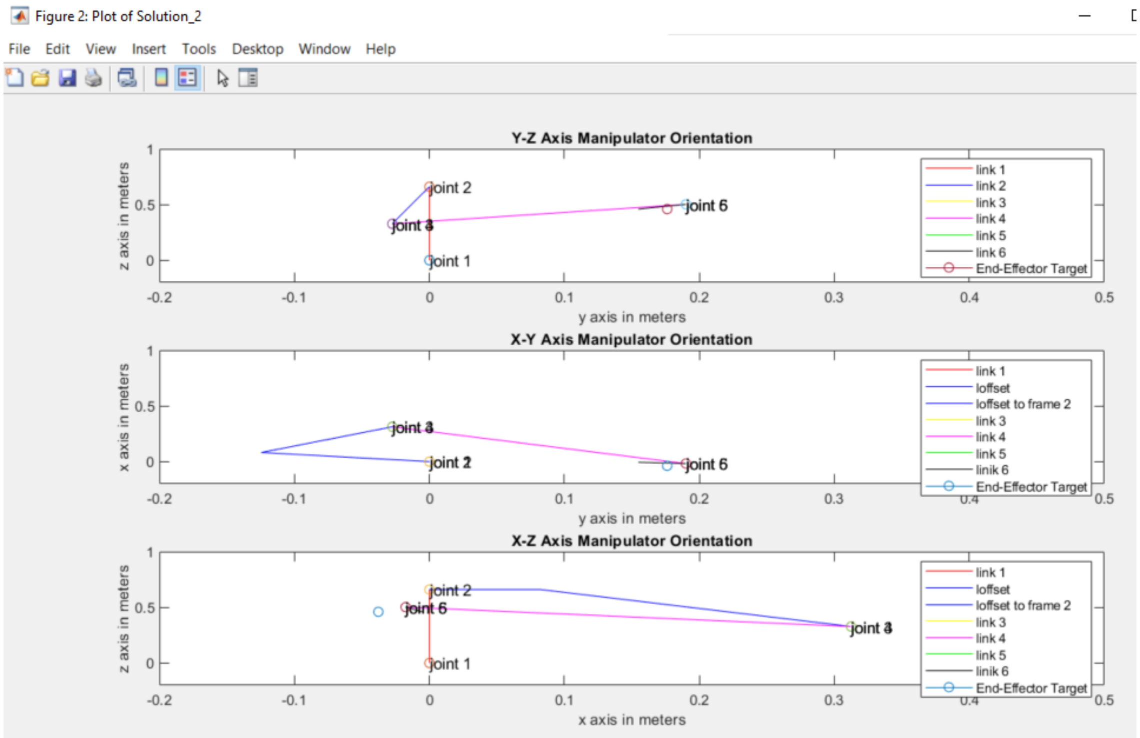


Figure 3: Manipulator Orientation Plot Solution Set 2

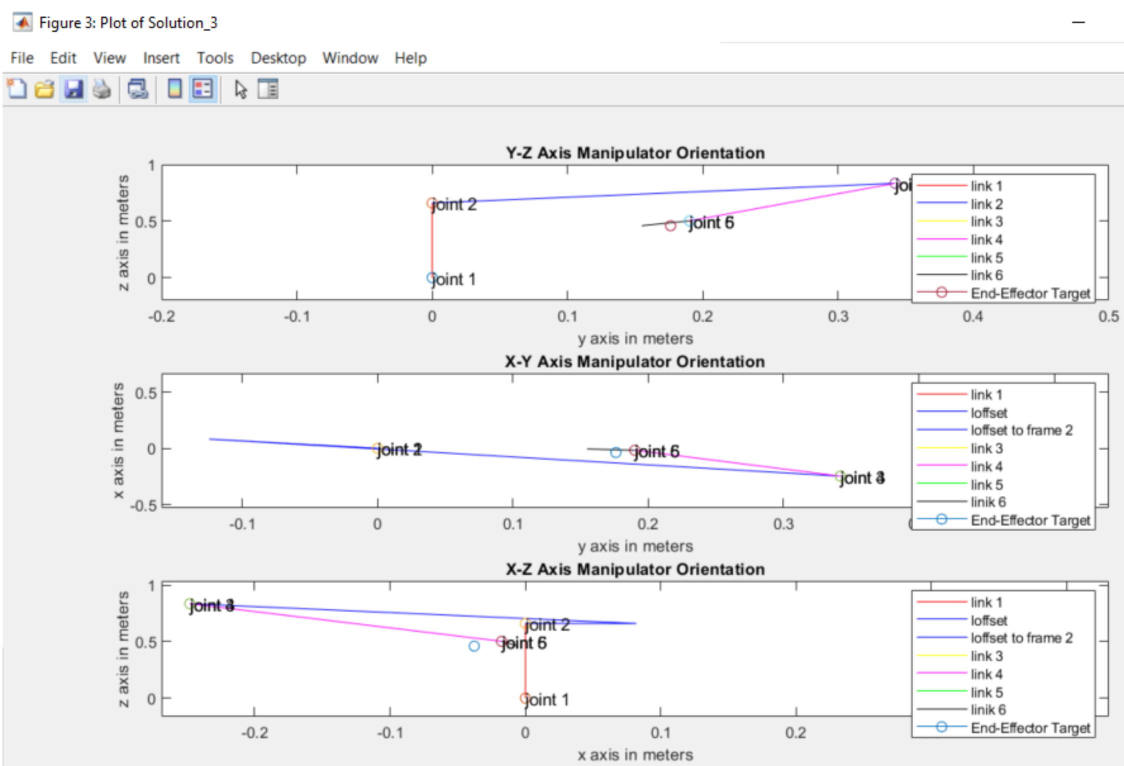


Figure 4: Manipulator Orientation Plot Solution Set 3

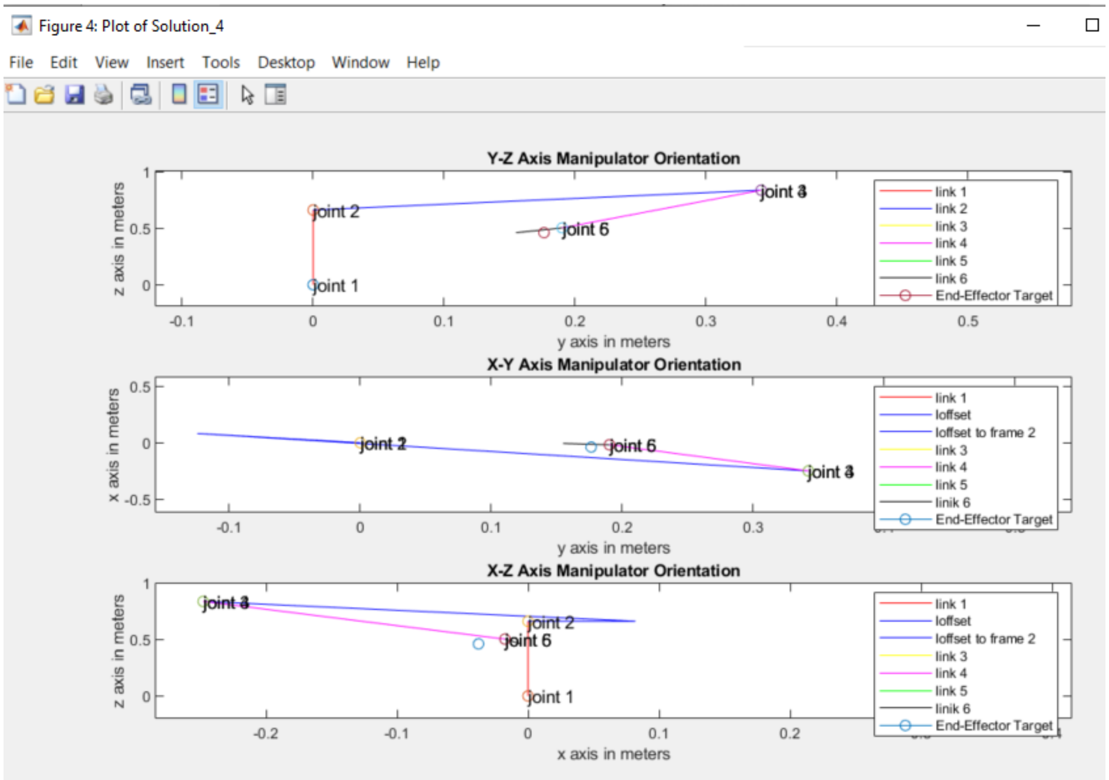


Figure 5: Manipulator Orientation Plot Solution Set 4

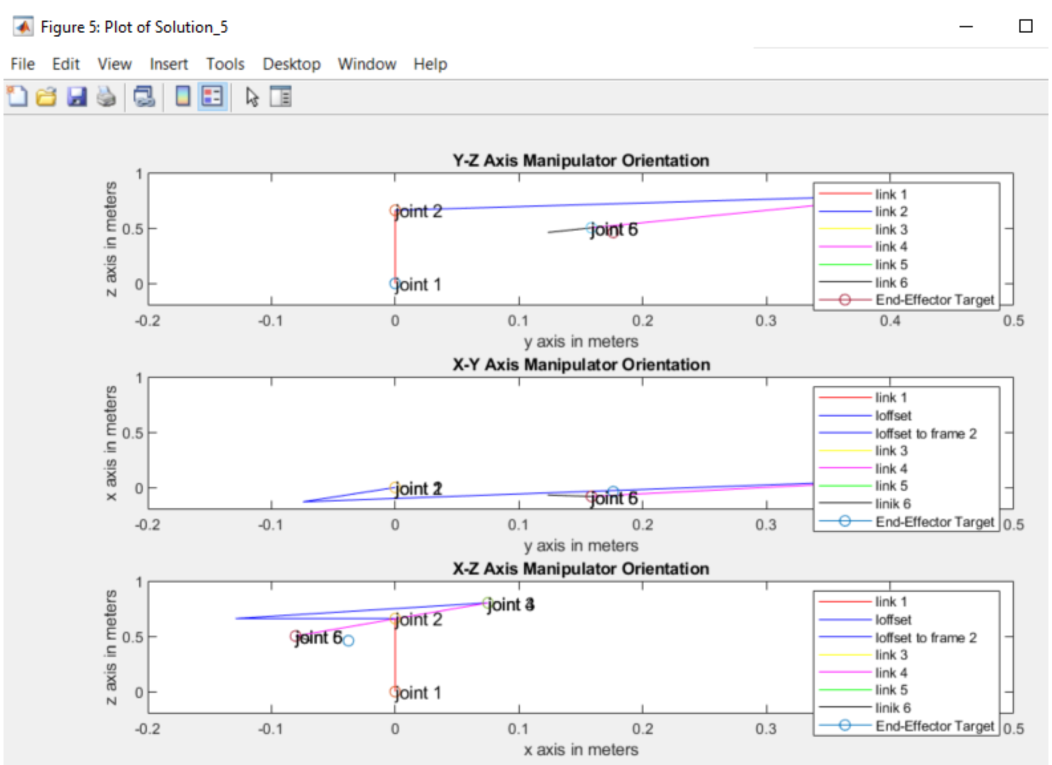


Figure 6: Manipulator Orientation Plot Solution Set 5

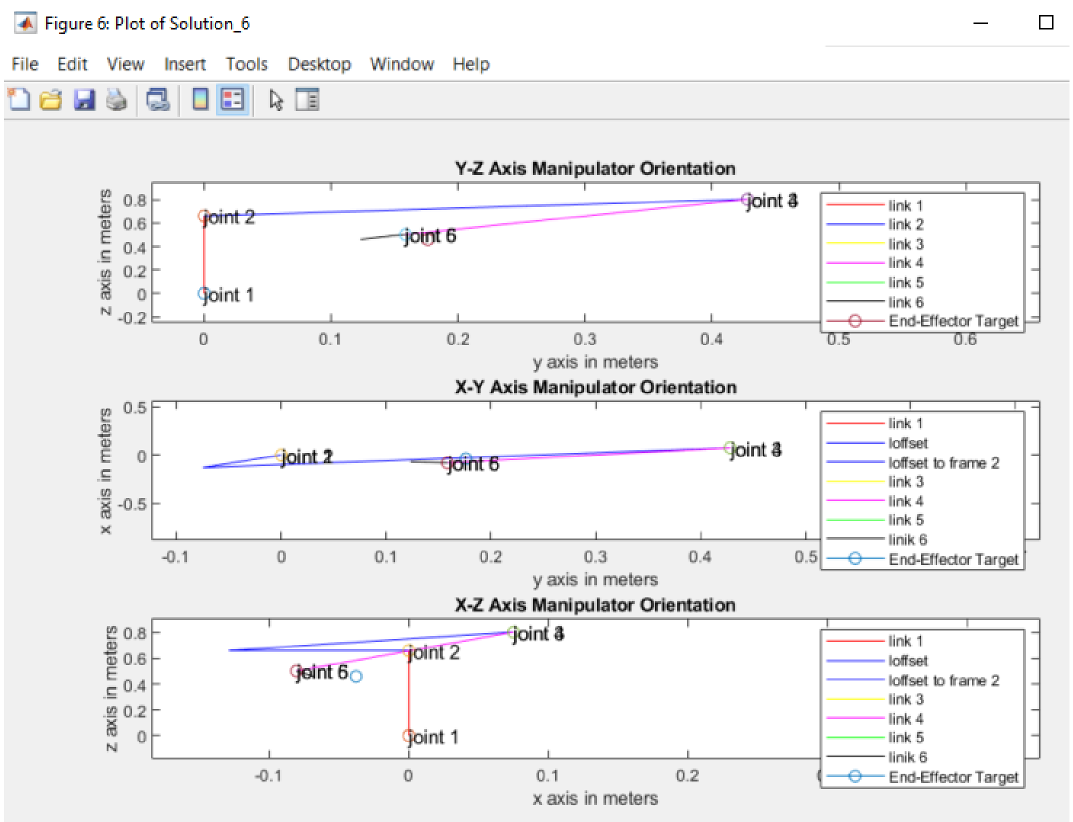


Figure 7: Manipulator Orientation Plot Solution Set 6

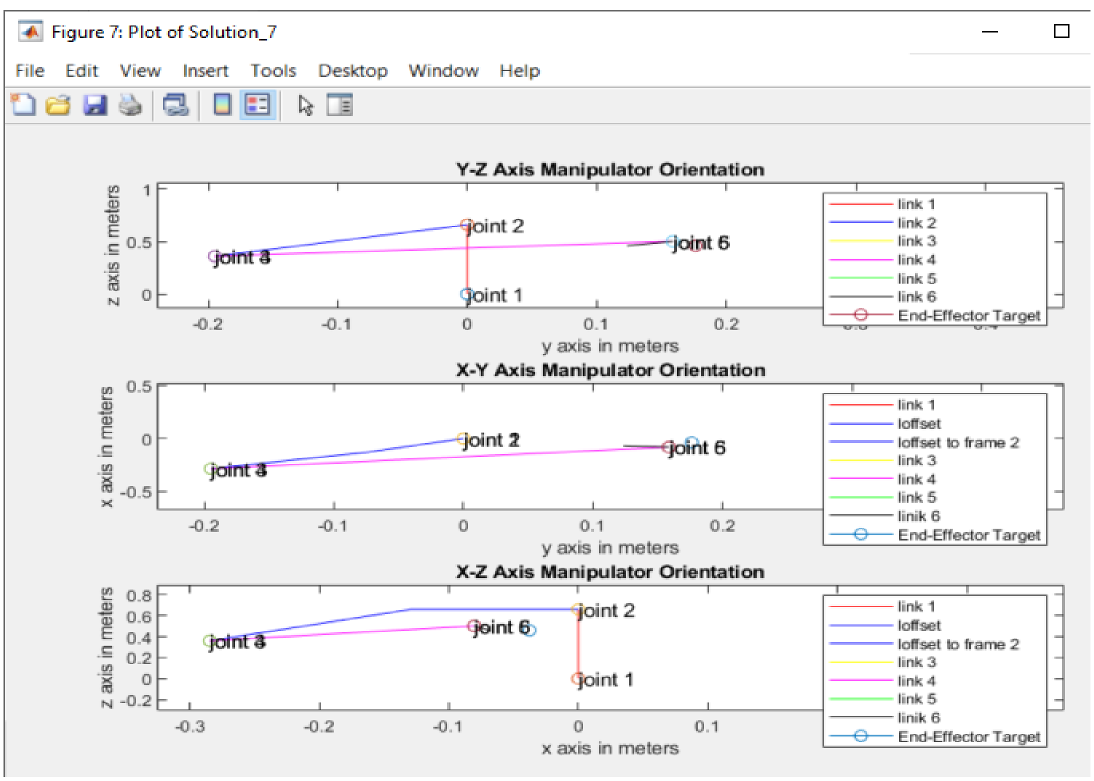


Figure 8: Manipulator Orientation Plot Solution Set 7

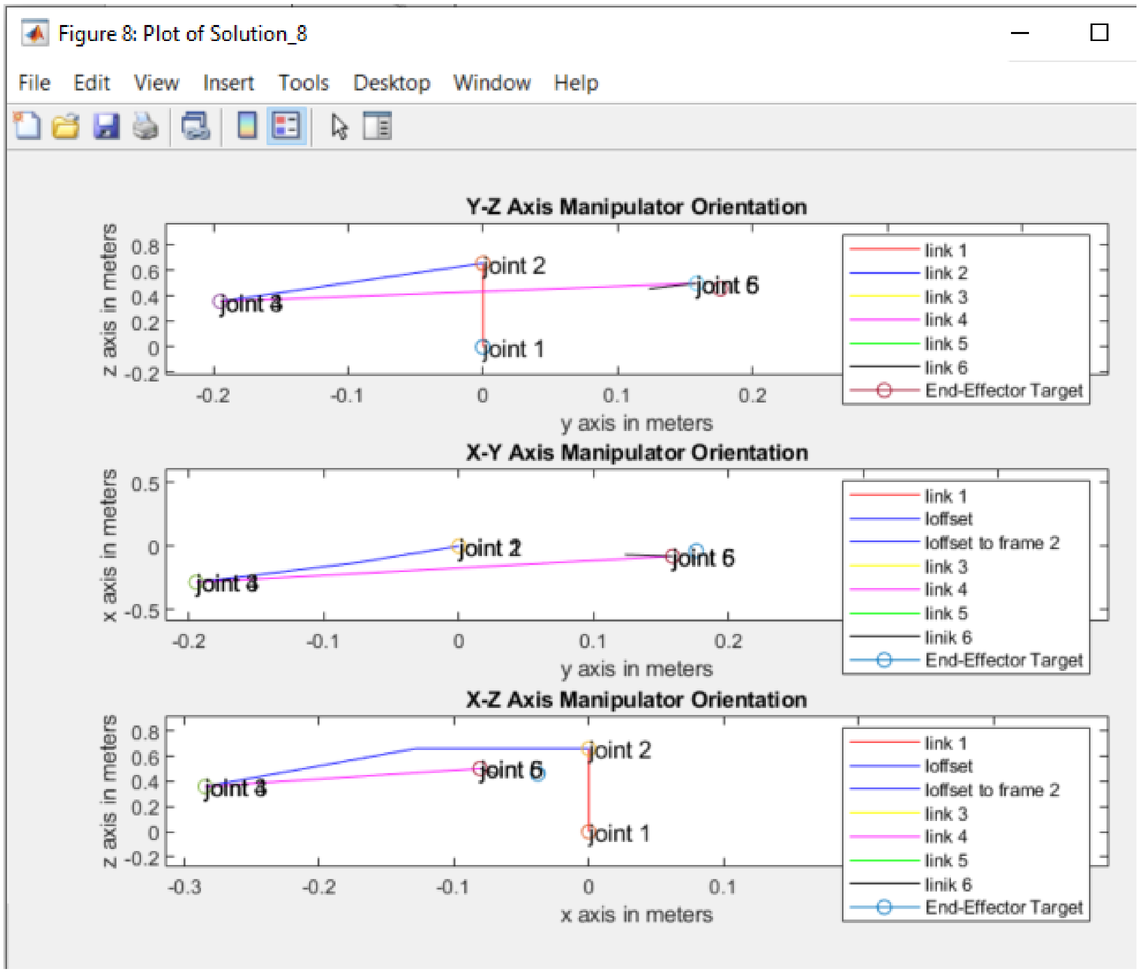


Figure 9: Manipulator Orientation Plot Solution Set 8

Q1(C) solution:

By looking at the equations formed during inverse kinematics, the number of solutions for a given T-End Effector matrix (assuming joints can rotate by 360 degrees) can be defined. Looking at all the equations for all angles from inverse kinematics, they are COSINE in form and will have 2 possible solution values since cos(x)=cos(-x). The total number solution sets generated from the equations is 2^8 = 64 but not all of these are valid. After substituting each angle set to the forward kinematic equation matrix and comparing it to the given end effector transformation matrix, the MATLAB code can identify <u>8 valid solution sets</u> possible for a given end-effector position and orientation. The MATLAB code result below shows that θ_1 has 2 values, θ_2 has 4 values, $\theta_3$has 4 values and θ_4, θ_5 and θ_6 each have 8 values on the solution set.

```
teta_values_up =
```

-33.3981	-50.2998	-63.4916	40.3114	141.5123	14.3981
-33.3981	-50.2998	-63.4916	-139.6886	-141.5123	-165.6019
-33.3981	-203.7914	243.4916	100.5949	24.1800	-120.7782
-33.3981	-203.7914	243.4916	-79.4051	-24.1800	59.2218
-299.9890	19.1525	-65.1991	64.4188	33.6587	29.9164
-299.9890	19.1525	-65.1991	-115.5812	-33.6587	-150.0836
-299.9890	-136.0466	245.1991	138.5673	130.9349	-120.0300
-299.9890	-136.0466	245.1991	-41.4327	-130.9349	59.9700
θ_1	θ_2	θ_3	θ_4	θ_5	θ_6

Solutions for Question 2

Q2(a) solution

The D-H parameters is shown in below table 2

Table 2: DH Parameters Table

Link number, i	θ_i	α_i	d_i	a_i
1	θ_1	0	l_1	0
2	$\pi/2$	$\pi/2$	d_2	0
3	0	0	d_3	0

Using the values for each link on the D-H parameters table, the homogenous transformation matrix for each link is derived. The general form of the transformation for a link in between joint i and joint i-1 is as follows:

$$^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using the MATLAB code generated for Question 1(A) to programmatically derive all homogenous transformation matrices for each link, the program results are shown below:

$$^0_1A = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^1_2A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^2_3A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can get the forward kinematic equation of the manipulator:

$$^0_2T = {}^0_1A\,{}^1_2A = \begin{bmatrix} -s_1 & 0 & c_1 & 0 \\ c_1 & 0 & s_1 & 0 \\ 0 & 1 & 0 & d_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^0_3T = {}^0_2T\,{}^2_3A = \begin{bmatrix} -s_1 & 0 & c_1 & d_3 c_1 \\ c_1 & 0 & s_1 & d_3 s_1 \\ 0 & 1 & 0 & d_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, we can get joint axes directions (expressed in $O_0x_0y_0z_0$):

$$b_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, b_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, b_2 = \begin{bmatrix} c_1 \\ s_1 \\ 0 \end{bmatrix}$$

$$r_{0,e} = l_1 b_0 + d_2 b_1 + d_3 b_2 = \begin{bmatrix} d_3 c_1 \\ d_3 s_1 \\ l_1 + d_2 \end{bmatrix}$$

$$r_{1,e} = d_2 b_1 + d_3 b_2 = \begin{bmatrix} d_3 c_1 \\ d_3 s_1 \\ d_2 \end{bmatrix}$$

$$r_{2,e} = d_3 b_2 = \begin{bmatrix} d_3 c_1 \\ d_3 s_1 \\ 0 \end{bmatrix}$$

Because joint 1 is rotational, joint 2 and 3 are prismatic. We can compute the Manipulator Jacobian:

For revolute joint 1:

$$\begin{bmatrix} J_{L1} \\ J_{A1} \end{bmatrix} = \begin{bmatrix} b_0 \times r_{0,e} \\ b_0 \end{bmatrix} = \begin{bmatrix} -d_3 s_1 \\ d_3 c_1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

For prismatic joint 2:

$$\begin{bmatrix} J_{L2} \\ J_{A2} \end{bmatrix} = \begin{bmatrix} b_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

For prismatic joint 3:

$$\begin{bmatrix} J_{L3} \\ J_{A3} \end{bmatrix} = \begin{bmatrix} b_2 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 \\ s_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Hence, the Jacobian matrix can be expressed as:

$$J=\begin{bmatrix} J_L \\ J_A \end{bmatrix}=\begin{bmatrix} -d_3s_1 & 0 & c_1 \\ d_3c_1 & 0 & s_1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

The liner velocity part:

$$J_L=\begin{bmatrix} -d_3s_1 & 0 & c_1 \\ d_3c_1 & 0 & s_1 \\ 0 & 1 & 0 \end{bmatrix}$$

Q2(b) solution

Because the endpoint is the origin of frame3, the moment arm of the force acting on the endpoint is 0. So, the torques is also 0. Assume that the torques/forces on the endpoint is ${}^0F$, the equivalent joints' torques/forces corresponding to the endpoint force can be expressed as:

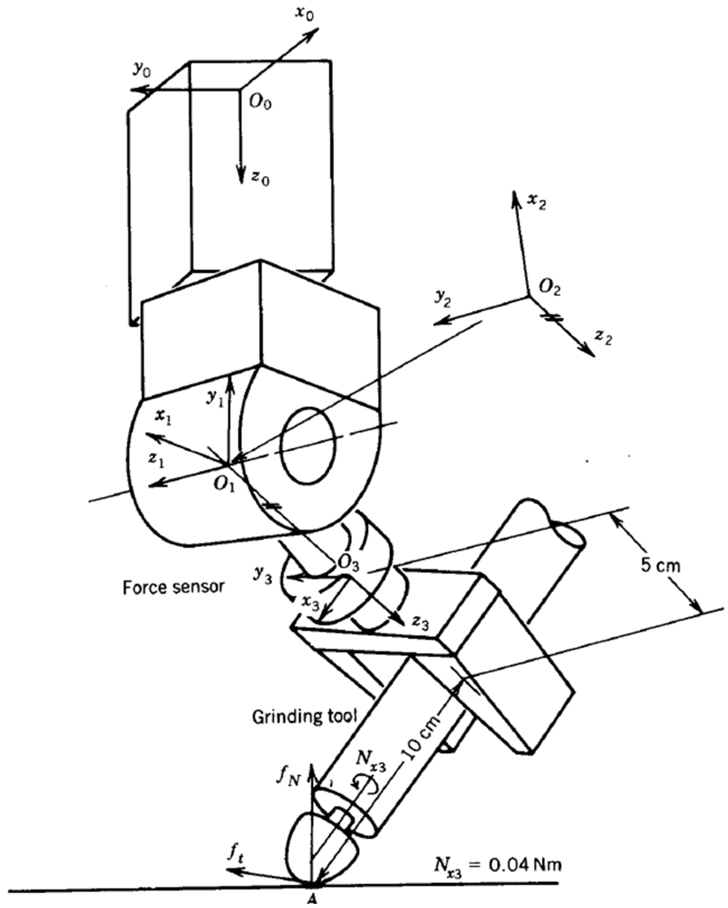
$$\tau=J^TF=\begin{bmatrix} -d_3s_1 & d_3c_1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ c_1 & s_1 & 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} f_x \\ f_y \\ f_z \\ 0 \\ 0 \\ 0 \end{bmatrix}=\begin{bmatrix} (f_yc_1-f_xs_1)d_3 \\ f_z \\ f_xc_1+f_ys_1 \end{bmatrix}$$

When ${}^0F=\begin{bmatrix}1 & 2 & 3\end{bmatrix}^T$, joint coordinates $\theta_1=0,\ d_2=1\ m$ and $d_3=1\ m$, we can get:

$$\tau=\begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

Solutions for Question 3

Q3(a) solution



Link Number	α_i	a_i	d_i	θ_i
1	-90°	0	0.4 m	θ_1
2	+90°	0	0	θ_2
3	0	0	0.1 m	θ_3
End Effector (EE)	0	0.1 m	0.05 m	0

To derive the 6x3 Jacobian matrix associated with the relationship between joint displacements and the position and orientation of the tool at point A, we first derive the transformation matrix from adjacent links.

$$^{i-1}_{\ \ \ i}A = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{0}_{1}A = \begin{bmatrix} cos\theta_1 & 0 & -sin\theta_1 & 0 \\ sin\theta_1 & 0 & cos\theta_1 & 0 \\ 0 & -1 & 0 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{1}_{2}A = \begin{bmatrix} cos\theta_2 & 0 & sin\theta_2 & 0 \\ sin\theta_2 & 0 & -cos\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}A = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & 0 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{EE}A = \begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.05 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{2}T = {}^{0}_{1}A \cdot {}^{1}_{2}A = \begin{bmatrix} c_1c_2 & -s_1 & s_2c_1 & 0 \\ s_1c_2 & c_1 & s_1s_2 & 0 \\ -s_2 & 0 & c_2 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{3}T = {}^{0}_{2}T \cdot {}^{2}_{3}A = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -s_3c_1c_2 - s_1c_3 & s_2c_1 & 0.1s_2c_1 \\ s_1c_2c_3 + c_1s_3 & -s_1s_3c_2 + c_1c_3 & s_1s_2 & 0.1s_1s_2 \\ -s_2c_3 & s_2s_3 & c_2 & 0.1c_2 + 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{EE}T = {}^{0}_{3}T \cdot {}^{3}_{EE}A = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -s_3c_1c_2 - s_1c_3 & s_2c_1 & 0.1(c_1c_2c_3 - s_1s_3) + 0.15s_2c_1 \\ s_1c_2c_3 + c_1s_3 & -s_1s_3c_2 + c_1c_3 & s_1s_2 & 0.1(s_1c_2c_3 + c_1s_3) + 0.15s_1s_2 \\ -s_2c_3 & s_2s_3 & c_2 & -0.1s_2c_3 + 0.15c_2 + 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Using

$$ \begin{bmatrix} J_{Li} \\ J_{Ai} \end{bmatrix} = \begin{bmatrix} b_{i-1} \times r_{i-1,e} \\ b_{i-1} \end{bmatrix}, where\ i = 1, 2, 3 \text{ (for all revolute joints)}, $$

$$ b_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} $$

$$ b_1 = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} $$

$$ b_2 = \begin{bmatrix} s_2c_1 \\ s_1s_2 \\ c_2 \end{bmatrix} $$

$$ r_{0,e} = {}^{0}_{3}d - {}^{0}_{0}d = {}^{0}_{3}d = \begin{bmatrix} 0.1s_2c_1 \\ 0.1s_1s_2 \\ 0.1c_2 + 0.4 \end{bmatrix} $$

$$ r_{1,e} = {}^{0}_{3}d - {}^{0}_{1}d = \begin{bmatrix} 0.1s_2c_1 \\ 0.1s_1s_2 \\ 0.1c_2 \end{bmatrix} $$

$$r_{2,e} = {}^{0}_{3}d - {}^{0}_{2}d = \begin{bmatrix} 0.1s_2c_1 \\ 0.1s_1s_2 \\ 0.1c_2 \end{bmatrix}$$

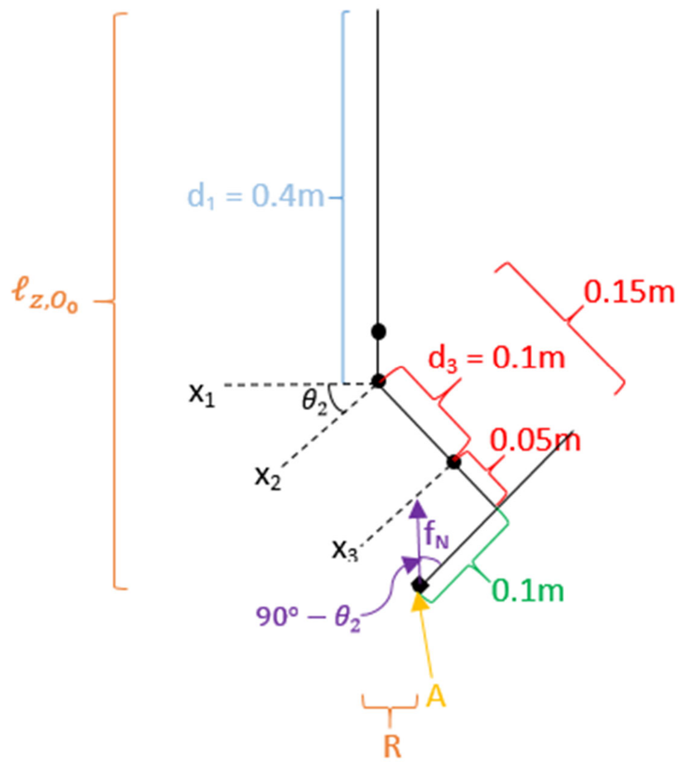
$$b_0 \times r_{0,e} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0.1s_2c_1 \\ 0.1s_1s_2 \\ 0.1c_2 + 0.4 \end{bmatrix} = \begin{bmatrix} -0.1s_1s_2 \\ 0.1s_2c_1 \\ 0 \end{bmatrix}$$

$$b_1 \times r_{1,e} = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0.1s_2c_1 \\ 0.1s_1s_2 \\ 0.1c_2 \end{bmatrix} = \begin{bmatrix} 0.1c_1c_2 \\ 0.1s_1c_2 \\ -0.1s_2 \end{bmatrix}$$

$$b_2 \times r_{2,e} = \begin{bmatrix} s_2c_1 \\ s_1s_2 \\ c_2 \end{bmatrix} \times \begin{bmatrix} 0.1s_2c_1 \\ 0.1s_1s_2 \\ 0.1c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Therefore, $J = \begin{bmatrix} J_{Li} \\ J_{Ai} \end{bmatrix} = \begin{bmatrix} -0.1s_1s_2 & 0.1c_1c_2 & 0 \\ 0.1s_2c_1 & 0.1s_1c_2 & 0 \\ 0 & -0.1s_2 & 0 \\ 0 & -s_1 & s_2c_1 \\ 0 & c_1 & s_1s_2 \\ 1 & 0 & c_2 \end{bmatrix}$

Q3(b) solution



For joint torques derivation, we use $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{F}$

To derive the 6x1 force vector, we assume **f_t = 10 N** and **f_N = 10 N**

Finding a reasonable angle to assume for θ_2

If R = 0 (radius of circle drawn by tooltip on work surface is 0) and point A is directly below origin of frame 0 (O_0),

Let ℓ_{z,O_1} be displacement from O_1 to work surface

$\ell_{z,O_1} = -\sqrt{0.15^2 + 0.1^2} = \frac{-\sqrt{13}}{20} or - 0.180$ (in metres)

$$\frac{\ell_{z,O_1}}{sin90°} = \frac{0.1}{sin\theta_2}$$

$\theta_2 \approx -33.69°$

From above, we learn that for R to be > 0, θ_2 must be < -33.69°

Since link 3 is slanted downwards at an angle from the horizontal (as shown in Figure 3 in Question), θ_2 > -90°

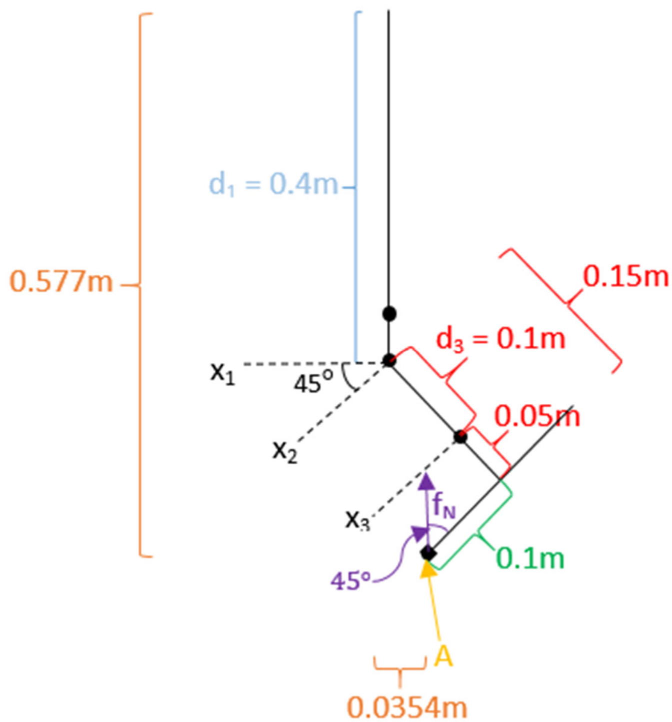
$-90° < \theta_2 < -33.69°$

We assume **θ_2 = -45°**

Using θ_2 derived, we can find ℓ_{z,O_0} and R, the radius of path of the tool tip on the work surface

$|\ell_{z,O_0}| = 0.4 + 0.15\sin(45°) + 0.1sin(45°) = \frac{16+5\sqrt{2}}{40}$ or 0.57678 (in metres)

R = $0.15\cos(45°) - 0.1cos(45°) = \frac{\sqrt{2}}{40}$ or 0.03536 (in metres)



$F_{x_3} = -f_N \times \sin(\theta_2)$

$F_{y_3} = f_t$

$F_{z_3} = -f_N \times \cos(\theta_2)$

$N_{x_3} = 0.04 - 0.05f_t$

$N_{y_3} = [0.1 \times f_N \times \cos(\theta_2)] - [0.05 \times f_N \times \sin(\theta_2)]$

$N_{z_3} = 0.1f_t$

$$\begin{bmatrix} {}^3_3f \\ {}^3_3n \end{bmatrix} = \begin{bmatrix} F_{x_3} \\ F_{y_3} \\ F_{z_3} \\ N_{x_3} \\ N_{y_3} \\ N_{z_3} \end{bmatrix} = \begin{bmatrix} -f_N(s_2) \\ f_t \\ -f_N(c_2) \\ 0.04 - 0.05f_t \\ 0.1f_N(c_2) - 0.05f_N(s_2) \\ 0.1f_t \end{bmatrix}$$

Transforming vectors f & n via Rotation Matrices,

$$F = \begin{bmatrix} {}^0_3f \\ {}^0_3n \end{bmatrix} = \begin{bmatrix} {}^0_3R & 0 \\ 0 & {}^0_3R \end{bmatrix} \begin{bmatrix} {}^3_3f \\ {}^3_3n \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -s_3c_1c_2 - s_1c_3 & s_2c_1 \\ s_1c_2c_3 + c_1s_3 & -s_1s_3c_2 + c_1c_3 & s_1s_2 \\ -s_2c_3 & s_2s_3 & c_2 \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -s_3c_1c_2 - s_1c_3 & s_2c_1 \\ s_1c_2c_3 + c_1s_3 & -s_1s_3c_2 + c_1c_3 & s_1s_2 \\ -s_2c_3 & s_2s_3 & c_2 \end{bmatrix} \end{bmatrix} \begin{bmatrix} -f_N(s_2) \\ f_t \\ -f_N(c_2) \\ 0.04 - 0.05f_t \\ 0.1f_N(c_2) - 0.05f_N(s_2) \\ 0.1f_t \end{bmatrix}$$

$$= \begin{bmatrix} f_Ns_2(s_1s_3 - c_1c_2c_3) - f_t(s_1c_3 + s_3c_1c_2) - f_Ns_2c_1c_2 \\ f_t(c_1c_3 - s_1s_3c_2) - f_Ns_2(s_3c_1 + s_1c_2c_3) - f_Ns_1s_2c_2 \\ -f_Nc_2{}^2 + f_Ns_2{}^2c_3 + f_ts_2s_3 \\ (0.05f_t - 0.04)(s_1s_3 - c_1c_2c_3) - (s_1c_3 + s_3c_1c_2)(0.1f_Nc_2 - 0.05f_Ns_2) + 0.1f_ts_2c_1 \\ (c_1c_3 - s_1s_3c_2)(0.1f_Nc_2 - 0.05f_Ns_2) - (0.05f_t - 0.04)(s_3c_1 + s_1c_2c_3) + 0.1f_ts_1s_2 \\ 0.1f_tc_2 + s_2s_3(0.1f_Nc_2 - 0.05f_Ns_2) + s_2c_3(0.05f_t - 0.04) \end{bmatrix}$$

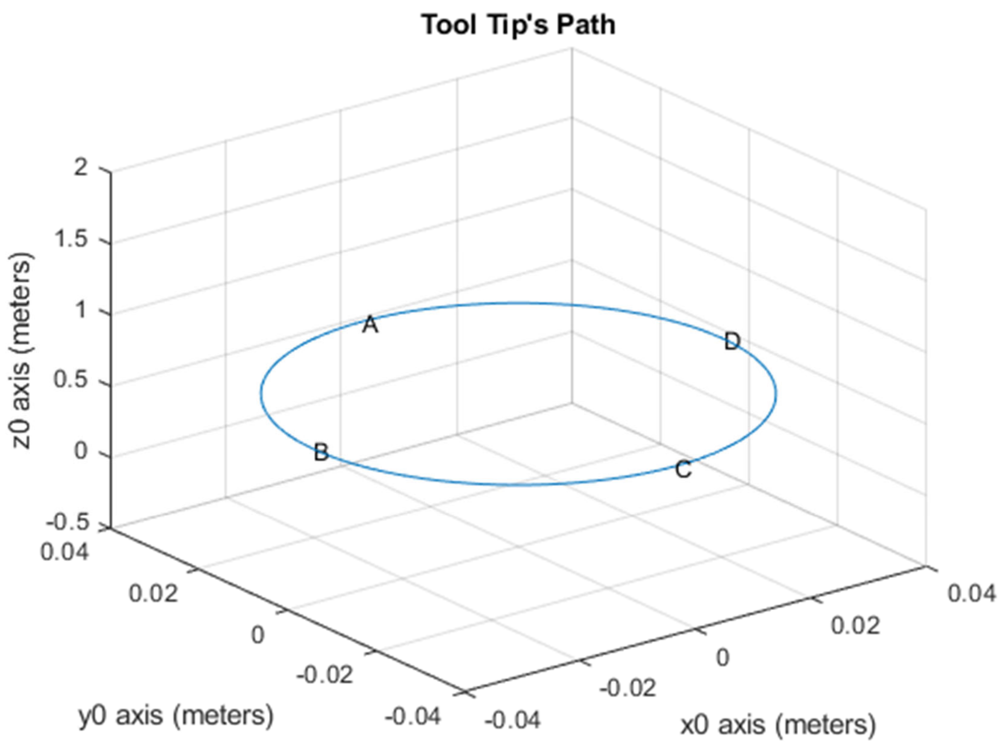
$$\tau = J^TF = \begin{bmatrix} -0.1s_1s_2 & 0.1c_1c_2 & 0 \\ 0.1s_2c_1 & 0.1s_1c_2 & 0 \\ 0 & -0.1s_2 & 0 \\ 0 & -s_1 & s_2c_1 \\ 0 & c_1 & s_1s_2 \\ 1 & 0 & c_2 \end{bmatrix}^T F$$

$$= \mathit{Solved\ in\ Q3\ MATLAB\ code\ (as\ tau)\ due\ to\ complexity}$$

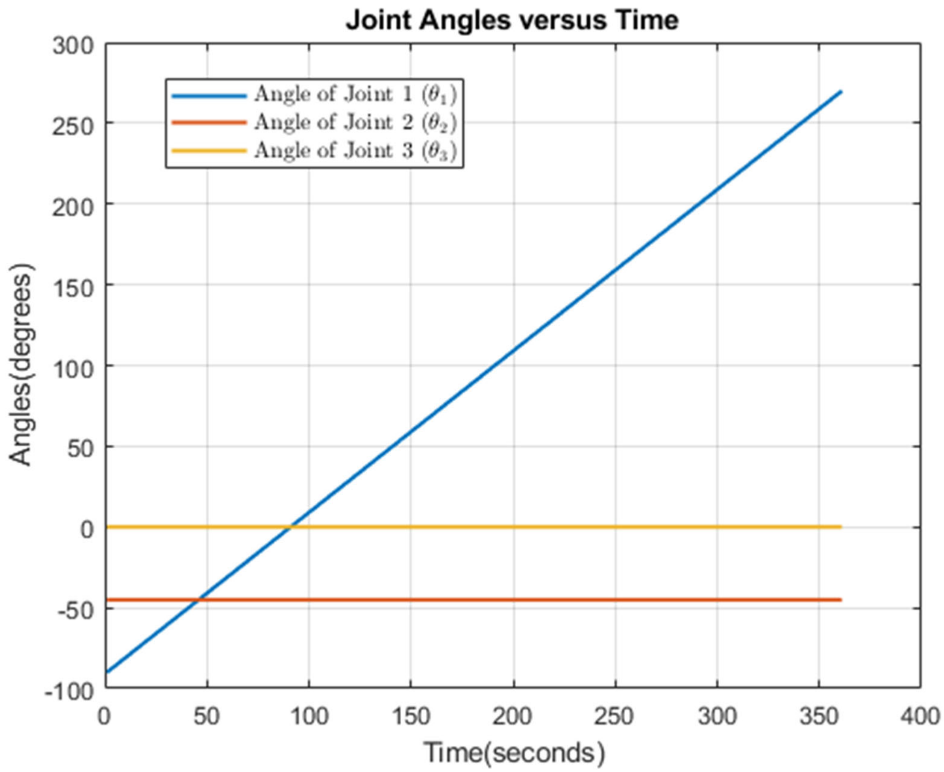
Q3(c) solution

To plot a circle on the working surface starting from Point A, points B, C, and D, and then back to Point A, we assume that the tool is fixed at the same angle at joint 2 and joint 3, and only joint 1 is rotating freely in 360° (i.e. **θ_2 = -45°**, **θ_3 = 0°**, **-90° < θ_1 < 270°**)

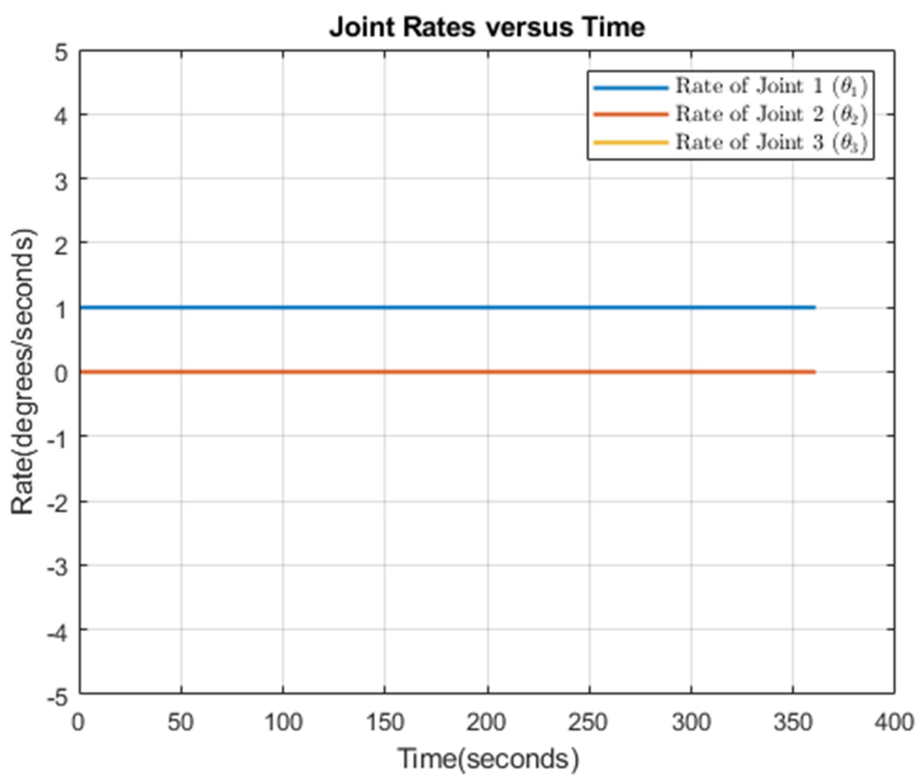
(i) Plot of the path of the tool tip on the work surface;



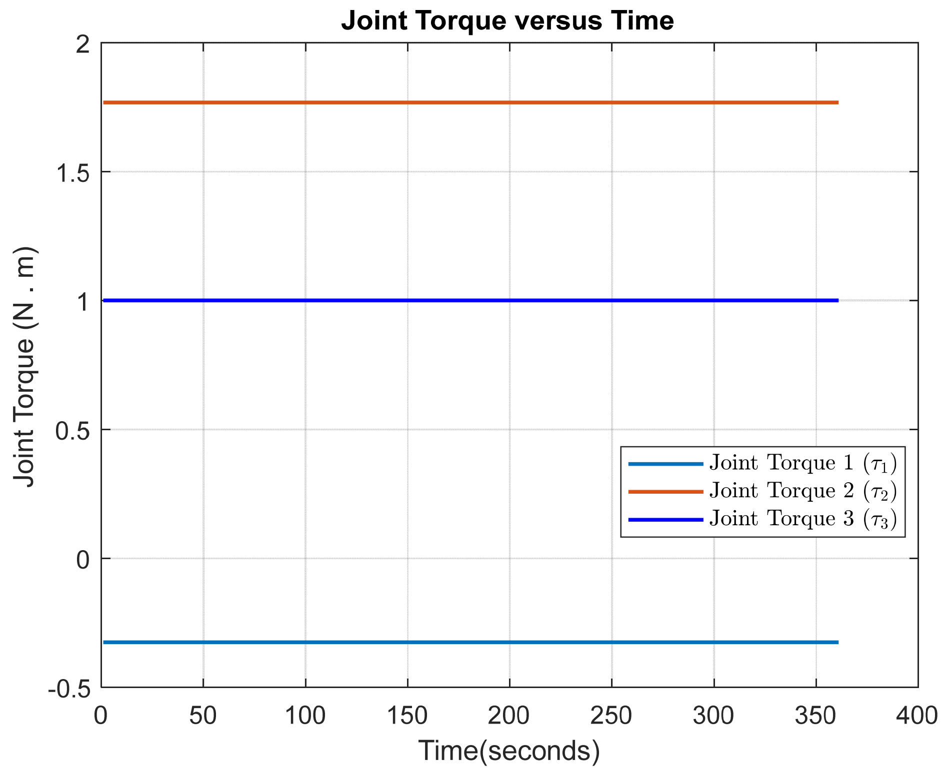
(ii) Plots of the respective joint angles versus time;



(iii) Plots of the respective joint rates versus time; and



(iv) Plots of the respective joint torques versus time.



Joint Torque versus Time
Joint Torque (N . m)
2
1.5
1
0.5
0
-0.5
0
50
100
150
200
250
300
350
400
Time(seconds)
Joint Torque 1 (τ_1)
Joint Torque 2 (τ_2)
Joint Torque 3 (τ_3)

Appendix

Q1(A) Solution MATLAB Code

Note: Please run CA1Q1a.mlx code to view solutions for Q1(A)

```
clear all
syms teta alpha d a
syms teta1 teta2 teta3 teta4 teta5 teta6
syms l0 loffset l1 l2 l3 d3
syms nx ny nz tx ty tz bx by bz px py pz
link_no=6;  %assign number of links for manipulator
T_link_all = []; %array where all T matrices of each link will be
concatenated.
```

```
%This portion of the code below is for forward kinematics of PUMA 600



for n= 1:link_no  %for loop to calculate T of manipulator
 %if else statement for assigning values to T matrix from DH table. Teta
 %and alpha values are in degrees
    if n==1
        teta=teta1; alpha=90; d=l0; a=0;
    elseif n==2
        teta=teta2; alpha=0; d=-loffset; a=l1;
    elseif n==3
        teta=teta3; alpha=90; d=0; a=0;
    elseif n==4
       teta=teta4; alpha=-90; d=l2; a=0;
    elseif n==5
      teta=teta5; alpha=90; d=0; a=0;
    else
     teta=teta6; alpha=0; d=l3;  a=0;
    end


T1=[cosd(teta) -sind(teta)*cosd(alpha) sind(teta)*sind(alpha)
a*cosd(teta)]; %this is the 1st row of T matrix using DH notation
T2=[sind(teta) cosd(teta)*cosd(alpha) -cosd(teta)*sind(alpha) a*sind(teta)]; %
this is the 2nd row of T matrix using DH notation
T3=[0 sind(alpha) cosd(alpha) d]; %this is the 3rd row of T matrix using DH
notation
T4=[0 0 0 1]; %this is the 4th row of T matrix using DH notation
T_link=[T1; T2; T3; T4]; %build T matrix for the link
T_link_all=[T_link_all T_link]; %concatenate all T links into 1 array
if n~= 1
T_manipulator = T_manipulator*T_link; %perform dot product of resulting T
matrix with previous iteration T matrix.
```

```
else
    T_manipulator = T_link;
end
 end


T0_1=T_link_all(1:4,1:4)
T1_2=T_link_all(1:4,5:8)
T2_3=T_link_all(1:4,9:12)
T3_4=T_link_all(1:4,13:16)
T4_5=T_link_all(1:4,17:20)
T5_6=T_link_all(1:4,21:24)


T_manipulator=simplify(T_manipulator) %final result of T_manipulator will be
forward kinematic equation.
```

Q1(B) Solution MATLAB Code

Note: Please run CA1Q1b.m to view solutions for Q1(B)

```
clear all
syms teta alpha d a
syms teta1 teta2 teta3 teta4 teta5 teta6
syms l0 loffset l1 l2 l3 d3
syms nx ny nz tx ty tz bx by bz px py pz
link_no=6;  %assign number of links for manipulator
T_link_all = []; %array where all T matrices of each link will be concatenated.


%Assume the following linik lengths in meters (from IEEE paper for PUMA 600(listed
as reference):
lo = .660;
loffset = .149;
l1 = .432;
l2 = .432;
l3 = .05639;


%This portion of the code below is for inverse kinematics of PUMA 600

%Given T-End Effector
T_EF=[0.8750 -0.4330 0.2165 -0.0378; -0.2165 -0.7500 -0.6250 0.1762; 0.4330 0.5000
-0.7500 0.4596; 0 0 0 1]

%initialize matrix that will contain teta values for all solutions
teta_values = [];

num_sol = 8; %all teta value equations (6 equations) are COS in form, so total
combination is 64. We use 8 here to compute teta 1,2 and 3 combinations first.
2^3=8

for n = 1:num_sol
```

```
    switch n
        case 1
            teta1_sign= 1; teta2_sign=1; teta3_sign=1;
        case 2
            teta1_sign= 1; teta2_sign=1; teta3_sign=-1;
        case 3
            teta1_sign= 1; teta2_sign=-1; teta3_sign=1;
        case 4
            teta1_sign= 1; teta2_sign=-1; teta3_sign=-1;
        case 5
            teta1_sign= -1; teta2_sign=-1; teta3_sign=-1;
        case 6
            teta1_sign= -1; teta2_sign=1; teta3_sign=1;
        case 7
            teta1_sign= -1; teta2_sign=1; teta3_sign=-1;
        otherwise
            teta1_sign= -1; teta2_sign=-1; teta3_sign=1;
    end

    a1 = (l3*T_EF(2,3)) - T_EF(2,4); %equation 1.3 at documentation
    b1 = T_EF(1,4)-(l3*T_EF(1,3)); %equation 1.4 at documentation
    c1 = -loffset; %equation 1.5 at documentation

    %compute for teta1 using general equation of acos(x)+bsin(x)=c for
    %equation 1.3 at documentation
    teta1_solved = (teta1_sign*acosd(c1/(sqrt(a1^2+b1^2))))+atan2d(b1,a1); % Eq.
1.2

    a2 = T_EF(2,4)*sind(teta1_solved)- l3*T_EF(1,3)*cosd(teta1_solved) +
l3*T_EF(2,3)*sind(teta1_solved) + T_EF(1,4)*cosd(teta1_solved); %Eq. 1.10 at
documentation
    b2 = T_EF(3,4) - l3*T_EF(3,3) -lo; %Eq. 1.11 at documentation
    c2 = (a2^2 +b2^2 +(l1^2) - (l2^2)) / (2*l1); %Eq. 1.12 at documentation

    %compute for teta2 using general equation of acos(x)+bsin(x)=c for
    %equation 2.0
    teta2_solved = (teta2_sign*acosd(c2/(sqrt(a2^2+b2^2))))+atan2d(b2,a2); %Eq.
1.9


    %compute for teta3
    teta3_solved =(teta3_sign*acosd(((l1*sind(teta2_solved)-b2))/l2)) -
teta2_solved; %Eq.1.13 at documentation

    %Solve for T3_6 Homogenous Matrix with computed teta1, teta2 and teta3
    %values

    T_link_U_all = [];

    for m= 1:3 %for loop to calculate T0_1, T0_2 and T0_3 of manipulator
    %if else statement for assigning values to T matrix from DH table
      if m==1
        teta=teta1_solved; alpha=90; d=lo; a=0;
      elseif m==2
        teta=teta2_solved; alpha=0; d=-loffset; a=l1;
      else
        teta=teta3_solved; alpha=90; d=0; a=0;
      end
```

```
      T1=[cosd(teta) -sind(teta)*cosd(alpha) sind(teta)*sind(alpha)
a*cosd(teta)]; %this is the 1st row of T matrix using DH notation
      T2=[sind(teta) cosd(teta)*cosd(alpha) -cosd(teta)*sind(alpha)
a*sind(teta)]; % this is the 2nd row of T matrix using DH notation
      T3=[0 sind(alpha) cosd(alpha) d]; %this is the 3rd row of T matrix using DH
notation
      T4=[0 0 0 1]; %this is the 4th row of T matrix using DH notation
      T_link=[T1; T2; T3; T4]; %build T matrix for the link
     if m~= 1
      T_link_U = T_link_U*T_link; %perform dot product of resulting T matrix with
previous iteration T matrix.
     else
    T_link_U = T_link;
     end
     T_link_U_all = [T_link_U_all T_link_U];
    end

    T0_1_up = T_link_U_all(1:4,1:4);
    T0_2_up = T_link_U_all(1:4,5:8);
    T0_3_up = T_link_U_all(1:4,9:12);

    T3_6_up = inv(T0_3_up)*T_EF; %Get T3_6 in terms of T_EF Eq.1.16

    %Solve for remaining teta angles teta 4, teta 5 and teta 6 by getting
    %all possible solutions using solved teta 1, teta 2 and teta 3
    for m = 1:num_sol
        switch m
            case 1
                teta4_sign= 1; teta5_sign=1; teta6_sign=1;
            case 2
                teta4_sign= 1; teta5_sign=1; teta6_sign=-1;
            case 3
                teta4_sign= 1; teta5_sign=-1; teta6_sign=1;
            case 4
                teta4_sign= 1; teta5_sign=-1; teta6_sign=-1;
            case 5
                teta4_sign= -1; teta5_sign=-1; teta6_sign=-1;
            case 6
                teta4_sign= -1; teta5_sign=1; teta6_sign=1;
            case 7
                teta4_sign= -1; teta5_sign=1; teta6_sign=-1;
            otherwise
                teta4_sign= -1; teta5_sign=-1; teta6_sign=1;
            end

         %compute for teta5
         teta5_solved = teta5_sign*acosd(T3_6_up(3,3)); %eq.1.17

        %compute for teta4
        teta4_solved = teta4_sign*acosd(T3_6_up(1,3)/sind(teta5_solved)); %eq.1.18

        %compute for teta6
        teta6_solved = teta6_sign*acosd(-T3_6_up(3,1)/sind(teta5_solved)); %eq.
1.19

        teta_values = [teta_values; teta1_solved teta2_solved teta3_solved
teta4_solved teta5_solved teta6_solved];
    end
end
```

```
teta_values; %this is the collection of all teta angle solution sets computed


%Check if each solution set is valid by plugging the teta angles calculated to
forward kinematic
%homogenous matrix code from Q1A:

teta_values_up=[];
for i=1:size(teta_values,1)
    teta_values_current = teta_values(i,:);
    %check if solution is valid or not
    for n= 1:link_no  %for loop to calculate T of manipulator
     %if else statement for assigning values to T matrix from DH table
        if n==1
            teta=teta_values_current(1); alpha=90; d=lo; a=0;
        elseif n==2
            teta=teta_values_current(2); alpha=0; d=-loffset; a=l1;
        elseif n==3
            teta=teta_values_current(3); alpha=90; d=0; a=0;
        elseif n==4
           teta=teta_values_current(4); alpha=-90; d=l2; a=0;
        elseif n==5
          teta=teta_values_current(5); alpha=90; d=0; a=0;
        else
         teta=teta_values_current(6); alpha=0; d=l3;  a=0;
        end

        T1=[cosd(teta) -sind(teta)*cosd(alpha) sind(teta)*sind(alpha)
a*cosd(teta)]; %this is the 1st row of T matrix using DH notation
        T2=[sind(teta) cosd(teta)*cosd(alpha) -cosd(teta)*sind(alpha)
a*sind(teta)]; % this is the 2nd row of T matrix using DH notation
        T3=[0 sind(alpha) cosd(alpha) d]; %this is the 3rd row of T matrix using
DH notation
        T4=[0 0 0 1]; %this is the 4th row of T matrix using DH notation
        T_link=[T1; T2; T3; T4]; %build T matrix for the link
        T_link_all=[T_link_all T_link]; %concatenate all T links into 1 array
        if n~= 1
            T_manipulator = T_manipulator*T_link; %perform dot product of
resulting T matrix with previous iteration T matrix.
        else
            T_manipulator = T_link;
        end
    end
    T_manipulator;


% Compare T_End effector Matrix from inverse kinematics input and T matrix
% of angle sets by chcking if the difference of their elements are less
% than 0.1
    T_diff_logic_all=[];
    for j=1:4
        for k=1:4
            T_diff_logic= abs(T_EF(j,k)-T_manipulator(j,k))<0.1;
            T_diff_logic_all = [T_diff_logic_all T_diff_logic];
        end
    end

    if all(T_diff_logic_all) == true
        teta_values_up=[teta_values_up; teta_values_current];
```

```
    end
end
teta_values_up % add all valid solution set of angles to this matrix

%Plot all valid solutions:
for p=1:size(teta_values_up,1);
    teta_values_plot=teta_values_up(p,:);
     T_link_U_all = [];
         for o= 1:link_no %for loop to calculate T0_1, T0_2, T0_3, T0_4, T0_5 and
T0_6 of angle set solution for plotting
        %if else statement for assigning values to T matrix from DH table
             if o==1
                teta=teta_values_plot(1); alpha=90; d=lo; a=0;
            elseif o==2
                teta=teta_values_plot(2); alpha=0; d=-loffset; a=l1;
            elseif o==3
                teta=teta_values_plot(3); alpha=90; d=0; a=0;
            elseif o==4
                teta=teta_values_plot(4); alpha=-90; d=l2; a=0;
            elseif o==5
                teta=teta_values_plot(5); alpha=90; d=0; a=0;
            else
                teta=teta_values_plot(6); alpha=0; d=l3;  a=0;
            end

          T1=[cosd(teta) -sind(teta)*cosd(alpha) sind(teta)*sind(alpha)
a*cosd(teta)]; %this is the 1st row of T matrix using DH notation
          T2=[sind(teta) cosd(teta)*cosd(alpha) -cosd(teta)*sind(alpha)
a*sind(teta)]; % this is the 2nd row of T matrix using DH notation
          T3=[0 sind(alpha) cosd(alpha) d]; %this is the 3rd row of T matrix using
DH notation
          T4=[0 0 0 1]; %this is the 4th row of T matrix using DH notation
          T_link=[T1; T2; T3; T4]; %build T matrix for the link
          if o~= 1
            T_link_U = T_link_U*T_link; %perform dot product of resulting T matrix
with previous iteration T matrix.
          else
            T_link_U = T_link;
          end
            T_link_U_all = [T_link_U_all T_link_U];
        end

        T0_1_up=T_link_U_all(1:4,1:4);
        T0_2_up=T_link_U_all(1:4,5:8);
        T0_3_up=T_link_U_all(1:4,9:12);
        T0_4_up=T_link_U_all(1:4,13:16);
        T0_5_up=T_link_U_all(1:4,17:20);
        T0_6_up=T_link_U_all(1:4,21:24);

        %Plot Solution for this iteration
        name = strcat('Plot of Solution','_',string(p));
        figure('Name',name);

        %Subplot for Y-Z Axis
        subplot(3,1,1);


        %plot frame 0 to frame 1
        pos_joints_y = [0 T0_1_up(2,4)];
```

```
pos_joints_z = [0 T0_1_up(3,4)];
plot(pos_joints_y,pos_joints_z,'r-')
hold on

%plot frame 1 to frame 2
pos_joints_y = [T0_1_up(2,4) T0_2_up(2,4)];
pos_joints_z = [T0_1_up(3,4) T0_2_up(3,4)];
plot(pos_joints_y,pos_joints_z,'b-')
hold on

%plot frame 2 to frame 3
pos_joints_y = [T0_2_up(2,4) T0_3_up(2,4)];
pos_joints_z = [T0_2_up(3,4) T0_3_up(3,4)];
plot(pos_joints_y,pos_joints_z,'y-')
hold on

%plot frame 3 to frame 4
pos_joints_y = [T0_3_up(2,4) T0_4_up(2,4)];
pos_joints_z = [T0_3_up(3,4) T0_4_up(3,4)];
plot(pos_joints_y,pos_joints_z,'m-')
hold on

%plot frame 4 to frame 5
pos_joints_y = [T0_4_up(2,4) T0_5_up(2,4)];
pos_joints_z = [T0_4_up(3,4) T0_5_up(3,4)];
plot(pos_joints_y,pos_joints_z,'g-')
hold on

%plot frame 5 to frame 6
pos_joints_y = [T0_5_up(2,4) T0_6_up(2,4)];
pos_joints_z = [T0_5_up(3,4) T0_6_up(3,4)];
plot(pos_joints_y,pos_joints_z,'k-')
hold on

%plot end effector location
pos_joints_y = [T_EF(2,4)];
pos_joints_z = [T_EF(3,4)];
plot(pos_joints_y,pos_joints_z,'o-')

   %plot joints location
pos_joints_y = [0];
pos_joints_z = [0];
plot(pos_joints_y,pos_joints_z,'o-')
text(pos_joints_y,pos_joints_z,'joint 1')

pos_joints_y = [T0_1_up(2,4)];
pos_joints_z = [T0_1_up(3,4)];
plot(pos_joints_y,pos_joints_z,'o-')
text(pos_joints_y,pos_joints_z,'joint 2')

pos_joints_y = [T0_2_up(2,4)];
pos_joints_z = [T0_2_up(3,4)];
plot(pos_joints_y,pos_joints_z,'o-')
text(pos_joints_y,pos_joints_z,'joint 3')

pos_joints_y = [T0_3_up(2,4)];
pos_joints_z = [T0_3_up(3,4)];
plot(pos_joints_y,pos_joints_z,'o-')
text(pos_joints_y,pos_joints_z,'joint 4')
```

```
        pos_joints_y = [T0_4_up(2,4)];
        pos_joints_z = [T0_4_up(3,4)];
        plot(pos_joints_y,pos_joints_z,'o-')
        text(pos_joints_y,pos_joints_z,'joint 5')

        pos_joints_y = [T0_5_up(2,4)];
        pos_joints_z = [T0_5_up(3,4)];
        plot(pos_joints_y,pos_joints_z,'o-')
        text(pos_joints_y,pos_joints_z,'joint 6')

        title('Y-Z Axis Manipulator Orientation')
        xlabel('y axis in meters');
        ylabel('z axis in meters');
        axis([-.2 .5 -0.2 1]);
        legend('link 1','link 2','link 3', 'link 4', 'link 5', 'link 6','End-
Effector Target');

        %Subplot for X-Y Axis
        subplot(3,1,2);

        %plot frame 0 to frame 1
        pos_joints_x = [0 T0_1_up(1,4)];
        pos_joints_y = [0 T0_1_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'r-')
        hold on

         %plot frame 1 to loffset_origin
        pos_joints_x = [T0_1_up(1,4) -loffset*sind(teta_values_plot(1))];
        pos_joints_y = [T0_1_up(2,4) -loffset*cosd(teta_values_plot(1))];
        plot(pos_joints_y,pos_joints_x,'b-')
        hold on

        %plot loffset_origin to frame 2
        pos_joints_x = [-loffset*sind(teta_values_plot(1)) T0_2_up(1,4)];
        pos_joints_y = [-loffset*cosd(teta_values_plot(1)) T0_2_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'b-')
        hold on

        %plot frame 2 to frame 3
        pos_joints_x = [T0_2_up(1,4) T0_3_up(1,4)];
        pos_joints_y = [T0_2_up(2,4) T0_3_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'y-')
        hold on

        %plot frame 3 to frame 4
        pos_joints_x = [T0_3_up(1,4) T0_4_up(1,4)];
        pos_joints_y = [T0_3_up(2,4) T0_4_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'m-')
        hold on

        %plot frame 4 to frame 5
        pos_joints_x = [T0_4_up(1,4) T0_5_up(1,4)];
        pos_joints_y = [T0_4_up(2,4) T0_5_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'g-')
        hold on

        %plot frame 5 to frame 6
        pos_joints_x = [T0_5_up(1,4) T0_6_up(1,4)];
```

```
        pos_joints_y = [T0_5_up(2,4) T0_6_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'k-')
        hold on

        %plot end effector location
        pos_joints_x = [T_EF(1,4)];
        pos_joints_y = [T_EF(2,4)];
        plot(pos_joints_y,pos_joints_x,'o-')

        %plot joints location
        pos_joints_x = [0];
        pos_joints_y = [0];
        plot(pos_joints_y,pos_joints_x,'o-')
        text(pos_joints_y,pos_joints_x,'joint 1')

        pos_joints_x = [T0_1_up(1,4)];
        pos_joints_y = [T0_1_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'o-')
        text(pos_joints_y,pos_joints_x,'joint 2')

        pos_joints_x = [T0_2_up(1,4)];
        pos_joints_y = [T0_2_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'o-')
        text(pos_joints_y,pos_joints_x,'joint 3')

        pos_joints_x = [T0_3_up(1,4)];
        pos_joints_y = [T0_3_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'o-')
        text(pos_joints_y,pos_joints_x,'joint 4')

        pos_joints_x = [T0_4_up(1,4)];
        pos_joints_y = [T0_4_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'o-')
        text(pos_joints_y,pos_joints_x,'joint 5')

        pos_joints_x = [T0_5_up(1,4)];
        pos_joints_y = [T0_5_up(2,4)];
        plot(pos_joints_y,pos_joints_x,'o-')
        text(pos_joints_y,pos_joints_x,'joint 6')


        title('X-Y Axis Manipulator Orientation')
        xlabel('y axis in meters');
        ylabel('x axis in meters');
        axis([-.2 .5 -0.2 1]);
        legend('link 1','loffset','loffset to frame 2', 'link 3', 'link 4', 'link
5','linik 6',"End-Effector Target");

        %Subplot for X-Z Axis
        subplot(3,1,3);

        %plot frame 0 to frame 1
        pos_joints_x = [0 T0_1_up(1,4)];
        pos_joints_z = [0 T0_1_up(3,4)];
        plot(pos_joints_x,pos_joints_z,'r-')
        hold on

         %plot frame 1 to loffset_origin
        pos_joints_x = [T0_1_up(1,4) -loffset*sind(teta_values_plot(1))];
```

```
pos_joints_z = [T0_1_up(3,4) lo];
plot(pos_joints_x,pos_joints_z,'b-')
hold on

%plot loffset_origin to frame 2
pos_joints_x = [-loffset*sind(teta_values_plot(1)) T0_2_up(1,4)];
pos_joints_z = [lo T0_2_up(3,4)];
plot(pos_joints_x,pos_joints_z,'b-')
hold on

%plot frame 2 to frame 3
pos_joints_x = [T0_2_up(1,4) T0_3_up(1,4)];
pos_joints_z = [T0_2_up(3,4) T0_3_up(3,4)];
plot(pos_joints_x,pos_joints_z,'y-')
hold on

%plot frame 3 to frame 4
pos_joints_x = [T0_3_up(1,4) T0_4_up(1,4)];
pos_joints_z = [T0_3_up(3,4) T0_4_up(3,4)];
plot(pos_joints_x,pos_joints_z,'m-')
hold on

%plot frame 4 to frame 5
pos_joints_x = [T0_4_up(1,4) T0_5_up(1,4)];
pos_joints_z = [T0_4_up(3,4) T0_5_up(3,4)];
plot(pos_joints_x,pos_joints_z,'g-')
hold on

%plot frame 5 to frame 6
pos_joints_x = [T0_5_up(1,4) T0_6_up(1,4)];
pos_joints_z = [T0_5_up(3,4) T0_6_up(3,4)];
plot(pos_joints_x,pos_joints_z,'k-')
hold on

%plot end effector location
pos_joints_x = [T_EF(1,4)];
pos_joints_z = [T_EF(3,4)];
plot(pos_joints_x,pos_joints_z,'o-')

%plot joints location
pos_joints_x = [0];
pos_joints_z = [0];
plot(pos_joints_x,pos_joints_z,'o-')
text(pos_joints_x,pos_joints_z,'joint 1')

pos_joints_x = [T0_1_up(1,4)];
pos_joints_z = [T0_1_up(3,4)];
plot(pos_joints_x,pos_joints_z,'o-')
text(pos_joints_x,pos_joints_z,'joint 2')

pos_joints_x = [T0_2_up(1,4)];
pos_joints_z = [T0_2_up(3,4)];
plot(pos_joints_x,pos_joints_z,'o-')
text(pos_joints_x,pos_joints_z,'joint 3')

pos_joints_x = [T0_3_up(1,4)];
pos_joints_z = [T0_3_up(3,4)];
plot(pos_joints_x,pos_joints_z,'o-')
text(pos_joints_x,pos_joints_z,'joint 4')
```

```
        pos_joints_x = [T0_4_up(1,4)];
        pos_joints_z = [T0_4_up(3,4)];
        plot(pos_joints_x,pos_joints_z,'o-')
        text(pos_joints_x,pos_joints_z,'joint 5')

        pos_joints_x = [T0_5_up(1,4)];
        pos_joints_z = [T0_5_up(3,4)];
        plot(pos_joints_x,pos_joints_z,'o-')
        text(pos_joints_x,pos_joints_z,'joint 6')

        title('X-Z Axis Manipulator Orientation')
        xlabel('x axis in meters');
        ylabel('z axis in meters');
        axis([-.2 .5 -0.2 1]);
        legend('link 1','loffset','loffset to frame 2', 'link 3', 'link 4', 'link
5','linik 6',"End-Effector Target");


end
hold off;
```

Q2 Solution MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%Author: Liu Weihao
%%%Data: 13 Mar 2022
%%%File name: Question2.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms l1 d2 d3 theta1 fx fy fz
%D-H representation
T1=[cos(theta1),-sin(theta1),0,0;
    sin(theta1),cos(theta1),0,0;
    0,0,1,l1;
    0,0,0,1];
T2=[0,0,1,0;
    1,0,0,0;
    0,1,0,d2;
    0,0,0,1];
T3=[1,0,0,0;
    0,1,0,0;
    0,0,1,d3;
    0,0,0,1];

T = T1*T2*T3;

% unit vector along z-axis expressed in x0y0z0
b0 = [0;0;1];
b1 = [0;0;1];
b2 = [cos(theta1);sin(theta1);0];
% Position vector from Oi-1 to end-effector
r0 = l1*b0+d2*b1+d3*b2;
r1 = d2*b1+d3*b2;
r2 = d3*b2;

% Jacobian matrix
J1 = [cross(b0,r0);b0];
J2 = [b1;zeros(3,1)];
J3 = [b2;zeros(3,1)];
J = [J1,J2,J3];
% Liner velocity part
JL = J(1:3,:);


% Transformation of Forces and moments
F=[fx;
   fy;
   fz;
   0;
   0;
   0];
R= T(1:3,1:3);
p = T(1:3,4:4);
px=[0,-p(3),p(2);p(3),0,-p(1);-p(2),p(1),0];
T_tra=[R,zeros(3,3);px*R,R];

% drive torque
tau_o = J.' * F;

% Substitute the value
```

```
fx = 1;
fy = 2;
fz = 3;
theta1 = 0;
d2 = 1;
d3 = 1;
tau = subs(tau_o);
```

Q3 Solution MATLAB Code

```
N=360;

syms theta1 theta2 theta3 fN ft
p_A = [0.1;0;0.05;1];
A01=[cosd(theta1),0,-sind(theta1),0
     sind(theta1),0,cosd(theta1),0
     0,-1,0,0.4
     0,0,0,1];
A12=[cosd(theta2),0,sind(theta2),0
     sind(theta2),0,-cosd(theta2),0
     0,1,0,0
     0,0,0,1];
A23=[cosd(theta3),-sind(theta3),0,0
     sind(theta3),cosd(theta3),0,0
     0,0,1,0.1
     0,0,0,1];
T02=A01*A12;
T03=A01*A12*A23;
T30=inv(T03);
p_A0=T03*p_A;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%Calculate torque/force%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
F3=[-fN*sind(theta2);
    ft;
    -fN*cosd(theta2);
    0.04-0.05*ft;
    0.1*fN*cosd(theta2)-0.05*fN*sind(theta2);
    0.1*ft];
%%%% Jacobian matrix
b0=[0;0;1];
b1=[-sind(theta1);cosd(theta1);0];
b2=[sind(theta2)*cosd(theta1);sind(theta1)*sind(theta2);cosd(theta2)];
r0e=0.4*b0+0.1*b2;
r1e=0.1*b2;
r2e=0.1*b2;
J1=[cross(b0,r0e);b0];
J2=[cross(b1,r1e);b1];
J3=[cross(b2,r2e);b2];
J=[J1,J2,J3];

F03 = [T03(1:3,1:3) zeros(3); zeros(3) T03(1:3,1:3)]*F3;
tau=J.'*F03;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Insert 9 point between two adjacent points.%%%%%%
%%%%% The surface_path can be seen as moving track%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
surface =[]; % This is the results

theta2 = -45;
theta3 = 0;
fN=10;
ft=10;
k=1;
for i = -90:1:270
```

```
    theta1=i;
    surface(:,k)=double(subs(p_A0));
    theta1_time(k)=theta1;
    joint_tau(:,k) = double(subs(tau));
    k=k+1;
end

theta2_time=-45*ones(N+1);
theta3_time=zeros(N+1);

Theta1_rate=ones(N+1);
Theta2_rate=zeros(N+1);
Theta3_rate=zeros(N+1);


% results
figure(1)
plot3(surface(1,1:N+1),surface(2,1:N+1),surface(3,1:N+1))
text(surface(1,1),surface(2,1),surface(3,1),'A');
text(surface(1,91),surface(2,91),surface(3,91),'B');
text(surface(1,181),surface(2,181),surface(3,181),'C');
text(surface(1,271),surface(2,271),surface(3,271),'D');
title("Tool Tip's Path")
xlabel('x0 axis (meters)')
ylabel('y0 axis (meters)')
zlabel('z0 axis (meters)')
grid on


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% joint angles versus time %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2)
plot(1:N+1,theta1_time, 1:N+1,theta2_time, 1:N+1,theta3_time,'LineWidth', 1.5)

title("Joint Angles versus Time")
xlabel('Time(seconds)')
ylabel('Angles(degrees)')
legend('Angle of Joint 1 ($\theta_1$)','Angle of Joint 2 ($\theta_2$)','Angle of
Joint 3 ($\theta_3$)','interpreter', 'latex')
grid on


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% joint rate versus time %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(3)
% plot(1:N+1,Theta1_rate,'b-','LineWidth', 1.5)
% hold on
% plot(1:N+1,Theta2_rate,'r-','LineWidth', 1.5)
% hold on
% plot(1:N+1,Theta3_rate,'g-','LineWidth', 1.5)
% hold on
plot(1:N+1,Theta1_rate,1:N+1,Theta2_rate,1:N+1,Theta3_rate,'b-','LineWidth', 1.5)
title("Joint Rates versus Time")
xlabel('Time(seconds)')
ylabel('Rate(degrees/seconds)')
legend('Rate of Joint 1 ($\theta_1$)','Rate of Joint 2 ($\theta_2$)','Rate of
Joint 3 ($\theta_3$)','interpreter', 'latex')
```

```
ylim([-5,5])
grid on


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% joint torque versus time %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(4)

plot(1:N+1,joint_tau(1,1:N+1),1:N+1,joint_tau(2,1:N+1),1:N+1,joint_tau(3,1:N+1),'b
-','LineWidth', 1.5)
title("Joint Torque versus Time")
xlabel('Time(seconds)')
ylabel('Joint Torque (N . m)')
legend('Joint Torque 1 ($\tau_1$)', 'Joint Torque 2 ($\tau_2$)','Joint Torque 3
($\tau_3$)', 'interpreter', 'latex')

grid on
```

References

[1] A. Bazerghi, A. Goldenberg and J. Apkarian, An exact kinematic model of PUMA 600 manipulator, IEEE Transactions on Systems, Man, and Cybernetics, vol. -14, no. 3, pp. 483-487, 1984. Accessed on: March 2, 2022 [Online]. Available: https://www.scribd.com/document/383176984/puma-600.