NATIONAL UNIVERSITY OF SINGAPORE

ME5402 ADVANCED ROBOTICS

Computing Team Project

Assignment – AY21/22 Semester 1

Name	Student Number	E-mail Address
Jin Hoontae	A0243155L	E0816449@u.nus.edu
Tan Jinji	A0229553Y	E0679986@u.nus.edu
Zhang Junlei	A0243221X	E0816515@u.nus.edu

Table of Contents

1. Question 1

In this question, a six degree-of-freedom robot manipulator, PUMA600, was given. Based on the frames indicated in the figure, we did the following parts: 1) Determination of the link parameters and derivation of the kinematic equations, 2) Computation of the inverse kinematics of the PUMA 600 by using MATLAB, and 3) The number of solutions available for the given endpoint location.

1.1. Link parameters and the kinematic equation of the robot.

The kinematic parameters were identified according to the Denavit-Hartenberg (DH) convention with the pre-determined coordinate frames (frame 0 – 6). **Table 1** shows the DH convention for PUMA 600.

Table 1. The DH parameter table of PUMA 600

Link number	θ_i	d_i	α_i	a_i
1	θ_1	d_0	$+90°$	0
2	θ_2	$-d_{offset}$	$0°$	l_1
3	θ_3	0	$+90°$	0
4	θ_4	d_2	$-90°$	0
5	θ_5	0	$+90°$	0
6	θ_6	d_3	$0°$	0

The DH transformation matrix can be expressed as:

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

Based on the constructed table and **Eq. 1** above, the following matrices could be obtained:

$$ {}^{0}_{1}A(\theta_1) = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}A(\theta_2) = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & l_1 cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & l_1 sin\theta_2 \\ 0 & 0 & 1 & -d_{offset} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{2}_{3}A(\theta_3) = \begin{bmatrix} cos\theta_3 & 0 & sin\theta_3 & 0 \\ sin\theta_3 & 0 & -cos\theta_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{3}_{4}A(\theta_4) = \begin{bmatrix} cos\theta_4 & 0 & -sin\theta_4 & 0 \\ sin\theta_4 & 0 & cos\theta_4 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{4}_{5}A(\theta_5) = \begin{bmatrix} cos\theta_5 & 0 & sin\theta_5 & 0 \\ sin\theta_5 & 0 & -cos\theta_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}A(\theta_6) = \begin{bmatrix} cos\theta_6 & -sin\theta_6 & 0 & 0 \\ sin\theta_6 & cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence, by using the forward kinematic equation below, we could compute the matrix of the manipulator arm that represents the end-effector position and orientation.

$$ {}^{0}_{6}T = {}^{0}_{1}A(\theta_1)\ {}^{1}_{2}A(\theta_2){}^{2}_{3}A(\theta_3){}^{3}_{4}A(\theta_4){}^{4}_{5}A(\theta_5){}^{5}_{6}A(\theta_6) $$

Since, the matrix is complex and lengthy, it will not be written here. The matrix can be found by typing *T0_6* in the command after running the MATLAB script. Below is the script used to compute the matrix of the manipulator arm.

```
syms t1 t2 t3 t4 t5 t6 d0 d_off d2 d3 l1 % t(n): angle theta
% The matrices were hand-calculated based on the DH parameters
A0_1 = [cos(t1) 0 sin(t1) 0; sin(t1) 0 -cos(t1) 0; 0 1 0 d0; 0 0 0 1];
A1_2 = [cos(t2) -sin(t2) 0 l1*cos(t2); sin(t2) cos(t2) 0 l1*sin(t2); 0 0 1 d_off; 0 0 0 1];
A2_3 = [cos(t3) 0 sin(t3) 0; sin(t3) 0 -cos(t3) 0; 0 1 0 0; 0 0 0 1];
A3_4 = [cos(t4) 0 -sin(t4) 0; sin(t4) 0 cos(t4) 0; 0 -1 0 d2; 0 0 0 1];
A4_5 = [cos(t5) 0 sin(t5) 0; sin(t5) 0 -cos(t5) 0; 0 1 0 0; 0 0 0 1];
A5_6 = [cos(t6) -sin(t6) 0 0; sin(t6) cos(t6) 0 0; 0 0 1 d3; 0 0 0 1];

% Answer %
T0_6 = simplify(A0_1*A1_2*A2_3*A3_4*A4_5*A5_6);
```

1.2. Computation of the inverse kinematics of the PUMA 600.

For simplicity purposes, an assumption needed to be made before proceeding with the computation of the inverse kinematics; that is, we pre-defined all the joint angle variables (θ_i) equal to $\frac{\pi}{6}$. Moreover, the length variables (d_i and a_i) were additionally guesstimated to obtain the numeric matrices (${}^{0}_{6}T$). **Table 2** shows the variable values pre-defined to solve the inverse kinematics.

Table 2. Table of the pre-defined values of PUMA 600 for inverse kinematics

Link number	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{\alpha_i}$	$\boldsymbol{a_i}$
1	$\pi/6$	500	$+90°$	0
2	$\pi/6$	-200	$0°$	500
3	$\pi/6$	0	$+90°$	0
4	$\pi/6$	400	$-90°$	0
5	$\pi/6$	0	$+90°$	0
6	$\pi/6$	150	$0°$	0

Hence, the numeric matrix could be obtained as below:

$$ {}^{0}_{6}T = \begin{bmatrix} 0.2522 & 0.1044 & 0.9620 & 719.3029 \\ -0.7874 & -0.5558 & 0.2667 & 602.9285 \\ 0.5625 & -0.8248 & -0.0580 & 541.2981 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Subsequently, an inverse matrix multiplication was used to compute the joint variables one by one in such a way:

$$ {}^{0}_{6}T = {}^{0}_{1}A(\theta_1)\ {}^{1}_{2}A(\theta_2){}^{2}_{3}A(\theta_3){}^{3}_{4}A(\theta_4){}^{4}_{5}A(\theta_5){}^{5}_{6}A(\theta_6) $$

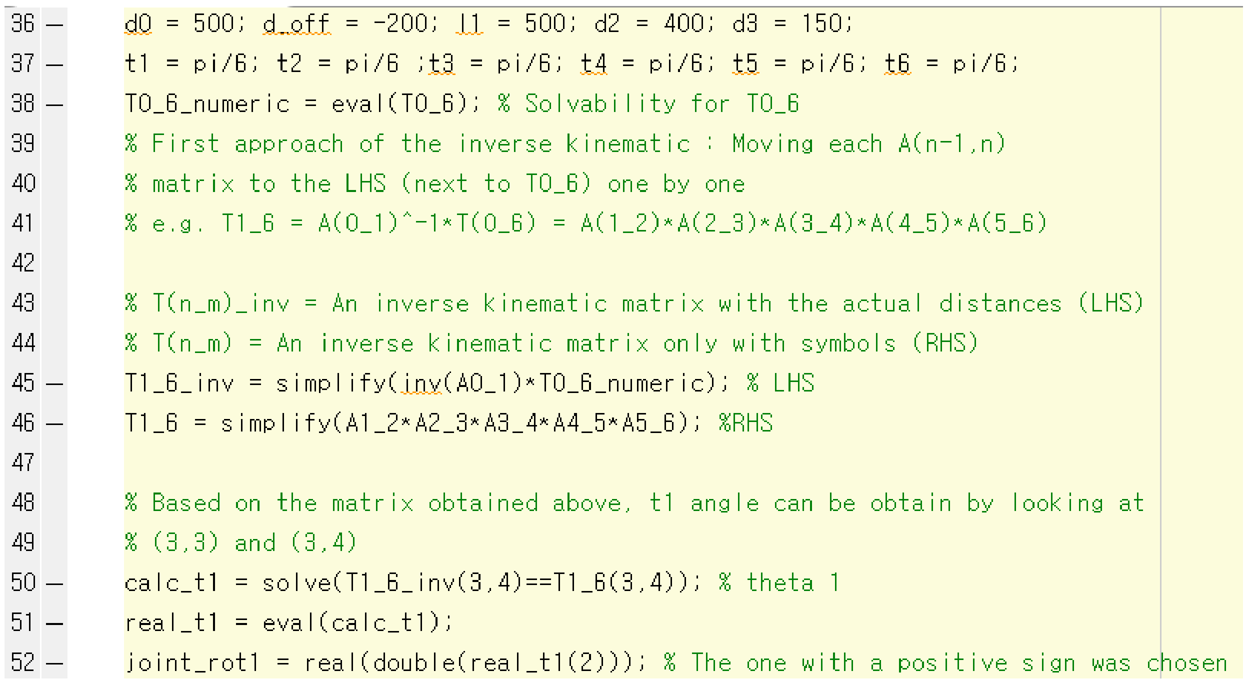
$${}^0_1A(\theta_1)^{-1}\,{}^0_6T = {}^1_2A(\theta_2)\,{}^2_3A(\theta_3)\,{}^3_4A(\theta_4)\,{}^4_5A(\theta_5)\,{}^5_6A(\theta_6) = {}^1_6T$$

$${}^1_2A(\theta_2)^{-1}\,{}^0_1A(\theta_1)^{-1}\,{}^0_6T = {}^2_3A(\theta_3)\,{}^3_4A(\theta_4)\,{}^4_5A(\theta_5)\,{}^5_6A(\theta_6) = {}^2_6T$$

$$\vdots$$

$${}^4_5A(\theta_5)^{-1}\,{}^3_4A(\theta_4)^{-1}\,{}^2_3A(\theta_3)^{-1}\,{}^1_2A(\theta_2)^{-1}\,{}^0_1A(\theta_1)^{-1}\,{}^0_6T = {}^5_6T$$

In this manner, all the joint variables (θ_i) were computed successfully. Below is the script that shows how each variable was obtained.



```
36 -    d0 = 500; d_off = -200; l1 = 500; d2 = 400; d3 = 150;
37 -    t1 = pi/6; t2 = pi/6 ;t3 = pi/6; t4 = pi/6; t5 = pi/6; t6 = pi/6;
38 -    T0_6_numeric = eval(T0_6); % Solvability for T0_6
39      % First approach of the inverse kinematic : Moving each A(n-1,n)
40      % matrix to the LHS (next to T0_6) one by one
41      % e.g. T1_6 = A(0_1)^-1*T(0_6) = A(1_2)*A(2_3)*A(3_4)*A(4_5)*A(5_6)
42
43      % T(n_m)_inv = An inverse kinematic matrix with the actual distances (LHS)
44      % T(n_m) = An inverse kinematic matrix only with symbols (RHS)
45 -    T1_6_inv = simplify(inv(A0_1)*T0_6_numeric); % LHS
46 -    T1_6 = simplify(A1_2*A2_3*A3_4*A4_5*A5_6); %RHS
47
48      % Based on the matrix obtained above, t1 angle can be obtain by looking at
49      % (3,3) and (3,4)
50 -    calc_t1 = solve(T1_6_inv(3,4)==T1_6(3,4)); % theta 1
51 -    real_t1 = eval(calc_t1);
52 -    joint_rot1 = real(double(real_t1(2))); % The one with a positive sign was chosen
```

Line 45 represents ${}^1_6T$ matrix in which the numeric matrix ${}^0_6T$ computed previously is multiplied by ${}^0_1A(\theta_1)$ that contains the symbolic variables. Line 46 shows ${}^1_6T$ matrix that does not contain numeric values, but it was only computed using the symbolic variables. As the numeric ${}^1_6T$ matrix needed to be equal to the symbolic ${}^1_6T$, a careful observation was required to find a proper index to compute θ_1. Based on the observation, the index (3,4) was found to be the simplest among the other indexes for computing the variable. *calc_t1* in line 50 solves the given equation with respect to θ_1 and *real_t1* in line 51 computes the desirable solution. Two solutions were obtained in line 51: -2.2700 and +0.5236. The positive value was chosen. The rest of the joint variables ($\theta_2, \theta_3, \theta_4, \theta_5$ *and* θ_6) were obtained in the same fashion. Hence, the steps to computing the variables were taken as follows:

1. Determine which joint variable to compute

2. Inverse-multiply "${}^0_6T = {}^0_1A(\theta_1)\ {}^1_2A(\theta_2)\,{}^2_3A(\theta_3)\,{}^3_4A(\theta_4)\,{}^4_5A(\theta_5)\,{}^5_6A(\theta_6)$" one by one

3. Observe the numeric and symbolic matrices and select a proper index to solve for θ_i

4. If two values (one for + and the other for -) are obtained for θ_i, select the positive one as all of the pre-defined joint angle values are positive.

The values obtained by the inverse kinematic approach are:

$$\begin{matrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \theta_5\\ \theta_6\end{matrix} = \begin{bmatrix}0.5236\\0.5236\\0.5236\\0.5236\\0.5236\\0.5236\end{bmatrix}$$

The result suggests that the values obtained were exactly equal to the pre-defined value ($\frac{\pi}{6}$), which proved that the inverse kinematic approach was accurately performed. The detailed computation process for the remaining joint angles can be found in **Section 4.1.**

Additionally, the plots of the joints and links of the robot in X-Y, X-Z and Y-Z planes were made to examine a schematic view of PUMA 600 according to the pre-defined values in **Table 2**. **Fig. 1** shows three 2-d plots and one 3-d plot. The MATLAB function for plotting the PUMA 600 diagrams below is shown in **Section 4.2.**

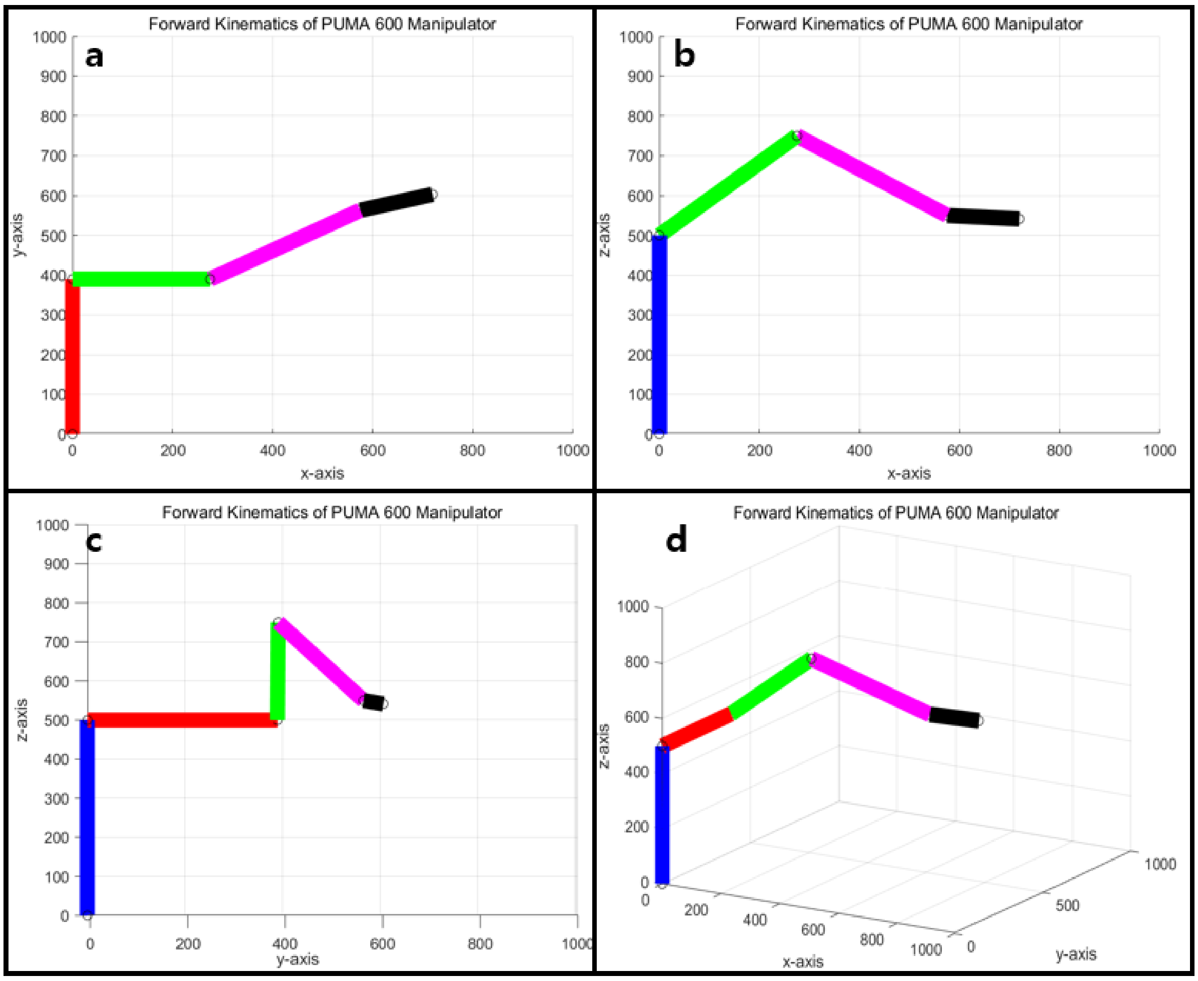


Figure 1. The plots of PUMA 600 in 2- and 3-dimensions (a: X-Y plane, b: X-Z plane, c: Y-Z plane, and d: X-Y-Z plane)

The coordinates of the endpoint shown in the figure were equal to the previously obtained position vector $[719.3029\ 602.9285\ 541.2981]^T$, which is the position vector of ${}_6^0T$ matrix.

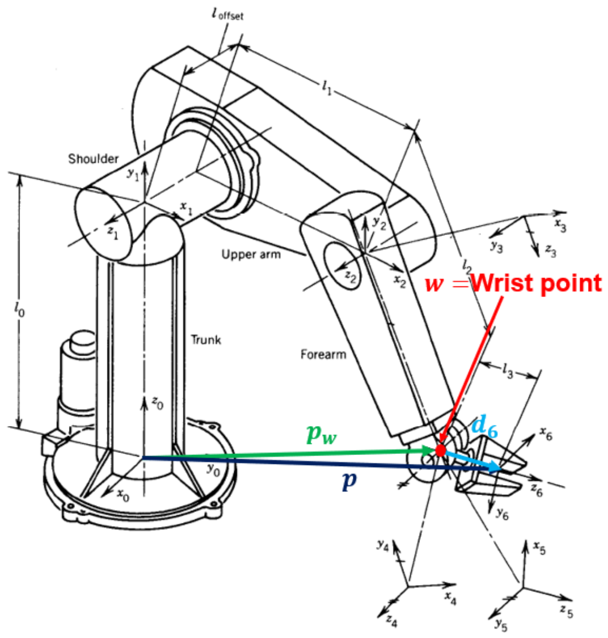
Hence, this suggests that the plotting function was successfully made to produce the PUMA600 diagram with the predefined values. The wrist approach will be deal with in the following section.

1.3. Number of solutions for the given endpoint location

The general transformation of an end-effector in the given figure can be expressed as:

$$ {}_6^0T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

At this point, in order to discover the number of solutions for the given endpoint location, it is necessary to determine the wrist point. Below is an illustration of the wrist point according to the given diagram. As can be seen, the wrist point is located at the intersection of the three terminal revolute axes.



As this wrist point can also be applied to our PUMA600 model, the following relationships can be built:

$$ p_w = p - d_6 a \ \text{ with } \ R = [\, n \; s \; a] $$

Using the homogeneous transformation matrix for the frame 4, we can obtain the position vector (p_w) with respect to the origin frame. Furthermore, the relationship ($p_w = p - d_6 a$) needs to be subsequently constructed for further computation as below:

```
% According to the figure given, the wrist point is at the fourth frame.
% Hence, we compute T0_4 matrix by forward kinmatic transformation as below
T0_4 = simplify(A0_1*A1_2*A2_3*A3_4);
% In order to find out the position of the wrist, we use the (4,1),(4,2) and (4,3) in the matrix.
p0_4a = T0_4(1:3,4); % the position of the wrist from T0_4
d3 = 150; %mm
p0_4b = T0_6(1:3,4) - d3*T0_6(1:3,3); % pw = p - d3a
```

According to the results obtained, the position vector (*p0_4a*) is:

$$p_{0\_4a} = \begin{bmatrix} p_{wx} \\ p_{wy} \\ p_{wz} \end{bmatrix} = \begin{bmatrix} d_2(c_1c_2s_3 + c_1c_3s_2) + d_{offset}s_1 + l_1c_1c_2 \\ d_2(c_2s_1s_3 + c_3s_1s_2) - d_{odffset}c_1 + l_1c_2s_1 \\ d_0 - d_2c_{23} + l_1s_2 \end{bmatrix} = \begin{bmatrix} d_2s_{23}c_1 + d_{offset}s_1 + l_1c_1c_2 \\ d_2s_1s_{23} - d_{odffset}c_1 + l_1c_2s_1 \\ d_0 - d_2c_{23} + l_1s_2 \end{bmatrix}$$

where $c_n = \cos(n)\ and\ s_n = \sin(n)$

From the relationship ($p_w = p - d_6a$), the following equation can be also written based on the geometry of the diagram:

$$p_w = \begin{bmatrix} p_{wx} \\ p_{wy} \\ p_{wz} \end{bmatrix} = \begin{bmatrix} p_x - l_3a_x \\ p_y - l_3a_y \\ p_z - l_3a_z \end{bmatrix} \text{ where } p_w = p_{0\_4a}$$

Based on the figure given and p_{0_4a}, d_{offset} can be used to construct a unique relationship with respect to θ_1 as below:

$$d_{offset} = s_1p_{wx} - c_1p_{wy}$$

In order to check the existence of θ_1, the pre-defined parameter values ($d_{offset} = -200mm$, $p_{wx} = 575mm$ and $p_{wy} = 562.92mm$) were inputted into the equation.

```
% Number of solutions for t1
t1_eq = sin(t1)*575.0000 - cos(t1)*562.9165 == -200;
t1_sol = eval(solve(dd,t1));
```

From this computation, two values were obtained for θ_1:

$$\theta_1 = \begin{bmatrix} -2.1156 \\ 0.5236 \end{bmatrix} = \begin{bmatrix} -121.2^\circ \\ 30^\circ \end{bmatrix} \rightarrow \text{2 solutions for } \theta_1$$

In a similar manner, another relationship with respect to θ_2 and θ_3 can be expressed as:

$$d_2s_{23} + l_1c_2 = c_1p_{wx} + s_1p_{wy} \quad [2]$$

$$d_0 - d_2c_{23} + l_1s_2 = p_{wz} \quad [3]$$

To further simplify the above relationships, we suppose:

$$X = c_1p_{wx} + s_1p_{wy}$$

$$Y = d_0 - p_{wz}$$

Hence, we can re-construct the previous relationships as:

For **Eq. 2**,

$$d_2s_{23} + l_1c_2 = X$$

$$s_{23} = {(X - l_1c_2)}/{d_2} \quad [4]$$

$$s_{23}^{2} = {(X - l_1c_2)/_{d_2}}^{2} \qquad [5]$$

For **Eq. 3**,

$$d_0 - d_2c_{23} + l_1s_2 = p_{wz}$$

$$c_{23} = (Y + l_1s_2)/_{d_2} \qquad [6]$$

$$c_{23}^{2} = {(Y + l_1s_2)/_{d_2}}^{2} \qquad [7]$$

Eq 5. + **Eq. 7** produces:

$$s_{23}^{2} + c_{23}^{2} = {(X - l_1c_2)/_{d_2}}^{2} + {(Y + l_1s_2)/_{d_2}}^{2} = 1$$

By expanding the equation, we obtain:

$$(X - l_1c_2)^2 + (Y + l_1s_2)^2 = d_2^{2}$$

$$X^2 - 2Xl_1c_2 + (l_1c_2)^2 + Y^2 + 2Yl_1s_2 + (l_1s_2)^2 = d_2^{2}$$

$$-2Xl_1c_2 + 2Yl_1s_2 = d_2^{2} - l_1^{2} - X^2 - Y^2$$

$$c_2 + s_2 = (d_2^{2} - l_1^{2} - X^2 - Y^2)/(2Yl_1 - 2Xl_1) \qquad [8]$$

Hence, there exist two solutions for **Eq. 8** as the RHS is a constant value in the working space. As for θ_3, **Eq. 4** and **Eq. 6** should be used so that the following equation is expressed:

$$\textbf{Eq. 4:}\ s_{23} = (X - l_1c_2)/_{d_2}$$

$$\textbf{Eq. 5:}\ c_{23} = (Y + l_1s_2)/_{d_2}$$

$$\frac{\boldsymbol{Eq.4}}{\boldsymbol{Eq.5}} = \frac{s_{23}}{s_{23}} = \tan\left(\theta_2 + \theta_3\right) = \frac{(X - l_1c_2)}{(Y + l_1s_2)}$$

To ensure that the above equation works, there should be only one solution for θ_3 because θ_2 is placed in both LHS and RHS, meaning that its value stays fixed.

Moving onto the next step where the remaining joint angles ($\theta_4, \theta_5\ and\ \theta_6$) need to be identified, we construct the following relationship:

$${}^{0}_{6}T = {}^{0}_{3}A\,{}^{3}_{6}A$$

$$({}^{0}_{3}A)^{-1}\,{}^{0}_{6}T = {}^{3}_{6}A \quad \text{where} \quad {}^{0}_{3}A^{-1}(1{:}3,1{:}3) = {}^{0}_{3}A^{T}(1{:}3,1{:}3)$$

Only considering the rotational homogeneous matrices, we can then obtain:

$$({}^{0}_{3}R)^{T}\,{}^{0}_{6}R = {}^{3}_{6}R$$

$$\begin{bmatrix} c_1c_{23} & s_1 & c_1s_{23} \\ s_1c_{23} & -c_1 & s_1s_{23} \\ s_{23} & 0 & -c_{23} \end{bmatrix}^T \begin{bmatrix} n_x & a_x & o_x \\ n_y & a_y & o_y \\ n_z & a_z & o_z \end{bmatrix} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_6s_4 - c_4c_5s_6 & c_4s_5 \\ c_4s_6 + c_5c_6s_4 & c_4c_6 - c_5s_4s_6 & s_4s_5 \\ -c_6s_5 & s_{56} & c_5 \end{bmatrix}$$

$$\begin{bmatrix} c_1c_{23} & s_1c_{23} & s_{23} \\ s_1 & -c_1 & 0 \\ c_1s_{23} & s_1s_{23} & -c_{23} \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_6s_4 - c_4c_5s_6 & c_4s_5 \\ c_4s_6 + c_5c_6s_4 & c_4c_6 - c_5s_4s_6 & s_4s_5 \\ -c_6s_5 & s_{56} & c_5 \end{bmatrix}$$

The LHS can be expanded to:

$$\begin{bmatrix} n_zs_{23} + n_xc_1c_{23} + n_ys_1c_{23} & o_zs_{23} + o_xc_1c_{23} + o_ys_1c_{23} & a_zs_{23} + a_xc_1c_{23} + a_ys_1c_{23} \\ n_xs_1 - n_yc_1 & o_xs_1 - o_yc_1 & a_xs_1 - a_yc_1 \\ n_xc_1s_{23} - n_zc_{23} + n_ys_1s_{23} & o_xc_1s_{23} - o_zc_{23} + o_ys_1s_{23} & a_xc_1s_{23} - a_zc_{23} + a_ys_1s_{23} \end{bmatrix}$$

As the number of solutions for θ_1, θ_2 and θ_3 are known, the LHS can be considered *constant*. Hence, by carefully analyzing the LHS and RHS matrices, the number of solutions for the other joint angle variables can be obtained as below:

To find $\boldsymbol{\theta_4}$:

$$LHS(1,3) = RHS(1,3) \rightarrow a_zs_{23} + a_xc_1c_{23} + a_ys_1c_{23} = c_4s_5 \qquad [9]$$

$$LHS(2,3) = RHS(2,3) \rightarrow a_xs_1 - a_yc_1 = s_4s_5 \qquad [10]$$

$$\frac{Eq.10}{Eq.9} = \frac{a_xs_1 - a_yc_1}{a_zs_{23} + a_xc_1c_{23} + a_ys_1c_{23}} = \frac{s_4s_5}{c_4s_5} = \frac{s_4}{c_4}$$

Hence, we eventually obtain:

$$\tan(\theta_4) = \frac{s_4}{c_4} = \frac{a_xs_1 - a_yc_1}{a_zs_{23} + a_xc_1c_{23} + a_ys_1c_{23}} \qquad [11]$$

Eq. 11 technically produces two values (two possible solutions) for the fourth joint variable (θ_4).

To find $\boldsymbol{\theta_5}$:

$$LHS(3,3) = RHS(3,3)$$

$$a_xc_1s_{23} - a_zc_{23} + a_ys_1s_{23} = c_5$$

,which can be re-written as:

$$\cos(\theta_5) = a_xc_1s_{23} - a_zc_{23} + a_ys_1s_{23} \rightarrow \textit{two solutions}$$

To find $\boldsymbol{\theta_6}$:

$$LHS(3,1) = RHS(3,1)$$

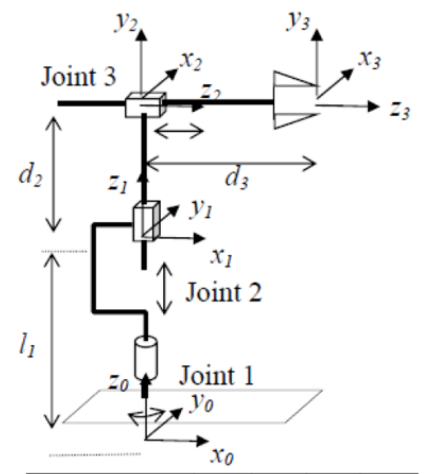
$$n_xc_1s_{23} - n_zc_{23} + n_ys_1s_{23} = -c_6s_5$$

$$\frac{-n_xc_1s_{23} + n_zc_{23} - n_ys_1s_{23}}{s_5} = c_6$$

$$\cos(\theta_6) = \frac{-n_x c_1 s_{23} + n_z c_{23} - n_y s_1 s_{23}}{s_5} \rightarrow \textit{two solutions}$$

2. Question 2

In question 2, a diagram of a 3-DOF manipulator with one-revolute joint and two-prismatic joints and its D-H table were given as shown in **Fig. 2**.



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

Figure 2. 3-DOF manipulator and its D-H table

Based on the information given, we were asked to compute the Jacobian matrix and the equivalent joints' torques/forces corresponding to the endpoint force. The following two sections will demonstrate how the tasks were performed.

2.1. Jacobian matrix regarding the endpoint

Based on the D-H table, the transformation matrices for each link were created. As can be seen in the script below, a function named "*DH_mat(link)*" was made in advance to compute the desired matrices rather automatically.

```
function TransformMat = DH_mat(link)
t = link.dh(1);
d = link.dh(2);
a = link.dh(3);
alpha = link.dh(4);
TransformMat = [cos(t)  , -sin(t)*cos(alpha)  , sin(t)*sin(alpha)  , a*cos(t);
                sin(t)  ,  cos(t)*cos(alpha)  , -cos(t)*sin(alpha)  , a*sin(t);
                  0     ,       sin(alpha)     ,      cos(alpha)     ,    d     ;
                  0     ,         0            ,          0          ,    1     ;];
end
```

```
link(1).dh = [t1 l1 0 0]; % ".dh" creates a matrix containing the DH details for a certain link
link(2).dh = [pi/2 d2 0 pi/2];
link(3).dh = [0 d3 0 0];
% With the details given above, we use the forward kinematic to create
% matrices as below.
for i = 1:3 % function for the matrix creation can be found at the bottom (TransformMat)
    link(i).mat = DH_mat(link(i)); %link(1-3).mat contains the link matrices (A0_1, A1_2 and A2_3).
end
```

Hence, the following transformation matrices were obtained:

$$ {}^0_1A = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1_2A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^2_3A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For computing the Jacobian matrix that relates the joint velocities to the linear velocity of the endpoint, the formulae differ depending on the type of the joint. Moreover, the elements of the Jacobian matrix are, in general, functions of joint displacements. Below is the summary for the computation of manipulator Jacobian for different link types.

For translational joints,

$$ J_i = \begin{bmatrix} J_{Li} \\ J_{Ai} \end{bmatrix} = \begin{bmatrix} b_{i-1} \\ 0 \end{bmatrix} $$

For rotational joints,

$$ J_i = \begin{bmatrix} J_{Li} \\ J_{Ai} \end{bmatrix} = \begin{bmatrix} b_{i-1} \times r_{i-1,e} \\ b_{i-1} \end{bmatrix} $$

where b_{i-1} is the unit vector along z-axis of frame (i-1), $r_{i-1,e}$ is the position vector from O_{i-1} to end-effector (both expressed in $O_0X_0Y_0Z_0$)

To obtain the elements of the Jacobian matrices, hence, forward kinematic matrix multiplication was done to compute (${}^0_2T$ and ${}^0_3T$).

```
% Construct T0_2 and T0_3 to obtain r(i-1)_e and b(i-1)_e subsequently
T0_2 = link(1).mat*link(2).mat;
T0_3 = T0_2*link(3).mat;
```

$$ {}^0_2T = \begin{bmatrix} -sin\theta_1 & 0 & cos\theta_1 & 0 \\ cos\theta_1 & 0 & sin\theta_1 & 0 \\ 0 & 1 & 0 & d_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^0_3T = \begin{bmatrix} -sin\theta_1 & 0 & cos\theta_1 & d_3\cos\,\theta_1 \\ cos\theta_1 & 0 & sin\theta_1 & d_3 sin\theta_1 \\ 0 & 1 & 0 & d_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Based on the results computed thus far, the Jacobian elements could be obtained as below.

```
% b and r
b0 = [0 ; 0 ; 1];
b1 = link(1).mat(1:3,3);
b2 = T0_2(1:3,3);

r2_end = d3*T0_3(1:3,1)
r1_end = d2*T0_2(1:3,1) + r2_end ;
r0_end = l1*b0 + r1_end;
```

$$b_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad b_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad b_2 = \begin{bmatrix} cos\theta_1 \\ sin\theta_1 \\ 0 \end{bmatrix}$$

$$r_{2,e} = \begin{bmatrix} -d_3 sin\theta_1 \\ d_3 cos\theta_1 \\ 0 \end{bmatrix} \quad r_{1,e} = \begin{bmatrix} -d_2 sin\theta_1 - d_3 sin\theta_1 \\ d_2 cos\theta_1 + d_3 cos\theta_1 \\ 0 \end{bmatrix} \quad r_{0,e} = \begin{bmatrix} -d_2 sin\theta_1 - d_3 sin\theta_1 \\ d_2 cos\theta_1 + d_3 cos\theta_1 \\ l_1 \end{bmatrix}$$

With the elements obtained above, the following Jacobian matrix could be constructed.

```
% Substitute all the vectors into the Jacobian formula
% Prismatic : [b(i-1) ; 0], Revolute : [b(i-1)xr(i-1)_end ; b(i-1)]
% Joint 1: Prismatic, Joint 2: Prismatic, Joint 3: Prismatic
J = [cross(b0,r0_end),b1,b2;b0,[0;0;0],[0;0;0]]; % Answer
```

$$J = \begin{bmatrix} b_0 \times r_{0,e} & b_1 & b_2 \\ b_0 & O_{3\times1} & O_{3\times1} \end{bmatrix} = \begin{bmatrix} -d_2 cos\theta_1 - d_3 cos\theta_1 & 0 & cos\theta_1 \\ -d_2 sin\theta_1 - d_3 sin\theta_1 & 0 & sin\theta_1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

2.2. Equivalent joint's torques and forces

With the given values ($\theta_1 = 0, d_2 = 1m, d_3 = 1m$), the numeric Jacobian matrix could be obtained as below.

```
J_numeric = subs(J,[t1, d2, d3], [0, 1, 1]);
```

$$J_{numeric} = \begin{bmatrix} -2 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Assuming that the joints are frictionless, the joint torque (τ) can be formulated as:

$$\tau = J^T F$$

An arbitrary end-point force (F) can be written as $[f_{n,n+1}, n_{n,n+1}]^T$. In this question, according to the matrix multiplication rule ($(m \times n) \cdot (n \times k) = (m \times k)$), the force vector at the end-effector could be written as:

$$F = \begin{bmatrix} F_{3\times1} \\ N_{3\times1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ n_1 \\ n_2 \\ n_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

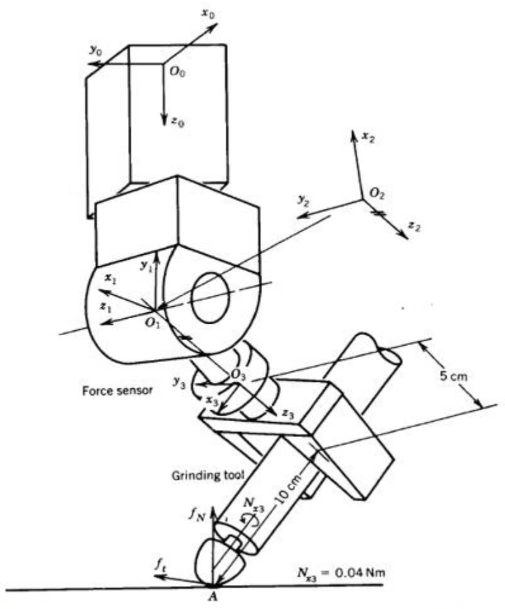
Hence, the joint torque (τ) could be obtained as below:

```
syms f1 f2 f3 n1 n2 n3
F = subs([f1;f2;f3;n1;n2;n3],[f1, f2, f3, n1, n2, n3], [1, 2, 3, 0, 0, 0]);
torque = J_numeric.'*F % Answer
```

$$\tau = \begin{bmatrix} -2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix} Nm$$

3. Question 3

In this question, a PUMA600 of three joints was given. Based on the given D-H table that describes the kinematic configuration of the wrist joints, we were asked to answer three sub-questions: 1) Derive the 6×3 Jacobian matrix at point A, 2) Derive the equivalent joint torques acting on the tool tip during the grinding operation, and 3) Write a program to move the tool in a desired manner. The given diagram and table can be seen in **Fig. 3**.



Link number	α_i	a_i	d_i
1	-90°	0	400 mm
2	+90°	0	0
3	0	0	100 mm

Figure 3. Diagram and D-H table for PUMA 600 of three joints

3.1. Jacobian matrix at point A

As the question is majorly focused on the tool tip (A), an additional frame needed to be considered. Hence, a new D-H table was constructed as below

Table 3. A new D-H table for the PUMA600 with an additional frame at point A.

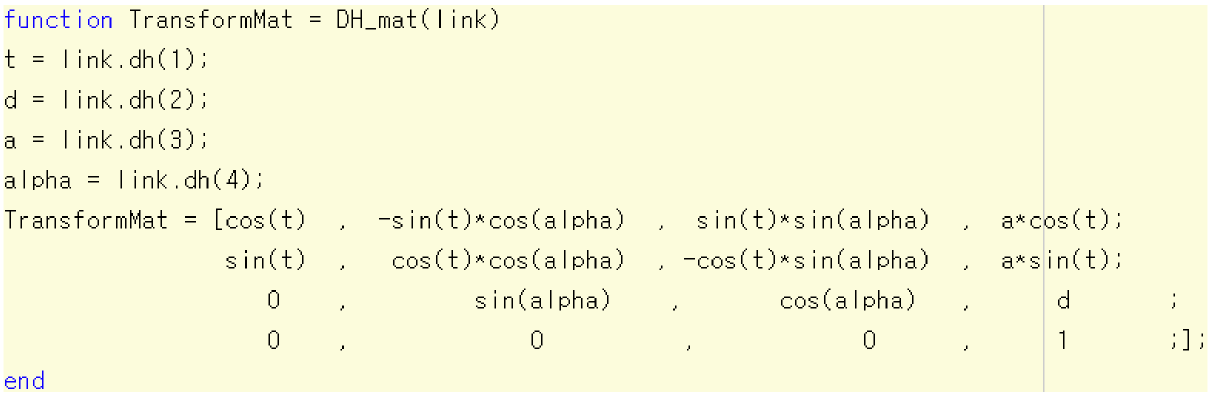
Link number	θ_i	$d_i(mm)$	α_i	$a_i(mm)$
1	θ_1	400	$-90°$	0
2	θ_2	0	$+90°$	0
3	θ_3	100	0	0
4	0	50	0	100

Using **Eq. 1** (pg.2) for computing the transformation matrices according to the D-H table, we obtained the following matrices:

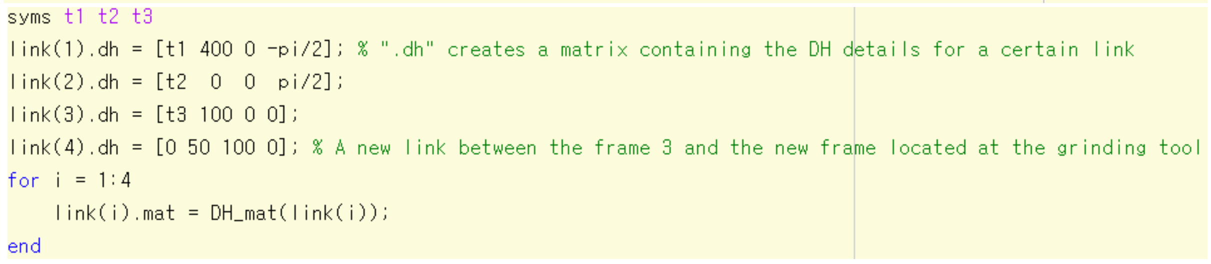
$$ {}^{0}_{1}A = \begin{bmatrix} cos\theta_1 & 0 & -sin\theta_1 & 0 \\ sin\theta_1 & 0 & cos\theta_1 & 0 \\ 0 & -1 & 0 & 400 \\ 0 & 0 & 0 & 1s \end{bmatrix} \quad {}^{1}_{2}A = \begin{bmatrix} cos\theta_2 & 0 & sin\theta_2 & 0 \\ sin\theta_2 & 0 & -cos\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}A = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & 0 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}_{3}A = \begin{bmatrix} 1 & 0 & 0 & 500 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 50 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

In the script, the matrices were computed as below.



```
function TransformMat = DH_mat(link)
t = link.dh(1);
d = link.dh(2);
a = link.dh(3);
alpha = link.dh(4);
TransformMat = [cos(t)   ,  -sin(t)*cos(alpha)  ,  sin(t)*sin(alpha)  ,  a*cos(t);
                sin(t)   ,   cos(t)*cos(alpha)  , -cos(t)*sin(alpha)  ,  a*sin(t);
                  0      ,          sin(alpha)     ,       cos(alpha)    ,     d       ;
                  0      ,              0          ,           0         ,     1       ;];
end
```



```
syms t1 t2 t3
link(1).dh = [t1 400 0 -pi/2]; % ".dh" creates a matrix containing the DH details for a certain link
link(2).dh = [t2  0  0  pi/2];
link(3).dh = [t3 100 0 0];
link(4).dh = [0 50 100 0]; % A new link between the frame 3 and the new frame located at the grinding tool
for i = 1:4
    link(i).mat = DH_mat(link(i));
end
```

Next, the forward kinematic multiplication was done to compute ${}^{0}_{i}T$, which was required to obtain b_{i-1} *and* $r_{i-1,e}$ vectors for constructing the Jacobian matrix. The desired vectors were obtained as below:

```
% b and r
b0 = [0 ; 0 ; 1];
b1 = link(1).mat(1:3,3);
b2 = T0_2(1:3,3);

r2_end = 100*T0_3(1:3,1);
r1_end = 0*T0_2(1:3,1) + r2_end ;
r0_end = 400*b0 + r1_end;
```

$$b_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad b_1 = \begin{bmatrix} -sin\theta_1 \\ cos\theta_1 \\ 0 \end{bmatrix} \quad b_2 = \begin{bmatrix} cos\theta_1 sin\theta_2 \\ sin\theta_1 sin\theta_2 \\ cos\theta_2 \end{bmatrix}$$

$$r_{2,e} = r_{1,e} = \begin{bmatrix} 100cos\theta_1 cos\theta_2 cos\theta_3 - 100sin\theta_1 sin\theta_3 \\ 100cos\theta_1 sin\theta_3 + 100cos\theta_2 cos\theta_3 sin\theta_1 \\ -100cos\theta_3 sin\theta_2 \end{bmatrix}$$

$$r_{0,e} = \begin{bmatrix} 100cos\theta_1 cos\theta_2 cos\theta_3 - 100sin\theta_1 sin\theta_3 \\ 100cos\theta_1 sin\theta_3 + 100cos\theta_2 cos\theta_3 sin\theta_1 \\ 400 - 100cos\theta_3 sin\theta_2 \end{bmatrix}$$

Thus, with the unit and position vectors obtained, the Jacobian matrix was built as below:

```
% Substitute all the vectors into the Jacobian formula
% Prismatic : [b(i-1) ; 0], Revolute : [b(i-1)xr(i-1)_end ; b(i-1)]
% Joint 1: Prismatic, Joint 2: Prismatic, Joint 3: Prismatic
J = simplify([cross(b0,r0_end),cross(b1,r1_end),cross(b2,r2_end);b0,b1,b2]); % Answer
```

```
J =

[- 100*cos(t1)*sin(t3) - 100*cos(t2)*cos(t3)*sin(t1), -100*cos(t1)*cos(t3)*sin(t2), - 100*cos(t3)*sin(t1) - 100*cos(t1)*cos(t2)*sin(t3)]
[  100*cos(t1)*cos(t2)*cos(t3) - 100*sin(t1)*sin(t3), -100*cos(t3)*sin(t1)*sin(t2),   100*cos(t1)*cos(t3) - 100*cos(t2)*sin(t1)*sin(t3)]
[                                                  0,        -100*cos(t2)*cos(t3),                                 100*sin(t2)*sin(t3)]
[                                                  0,                   -sin(t1),                                     cos(t1)*sin(t2)]
[                                                  0,                    cos(t1),                                     sin(t1)*sin(t2)]
[                                                  1,                          0,                                             cos(t2)]
```

3.2. The equivalent joint torques at the tool tip

For the derivation of the torques, several elements needed to be computed. Hence, multiple assumptions needed to be made before proceeding with the torque derivation.

3.2.1. Angles and lengths regarding the frame 2, 3 and 4

Fig. 4 shows the virtual lines/assumed lengths for the PUMA diagram given. As can be seen in the figure, the length of the line (l_{AB}) was set to be 150mm, and a relationship ($l_{AB} \perp l_{BC}$) was established. Also, the angle between l_{AB} and h was assumed to be 50°. These assumptions will be used continuously to write a program of the endpoint displacement subsequently. By applying basic trigonometric functions, the desired variables (a, B, h, r and c) could be obtained.

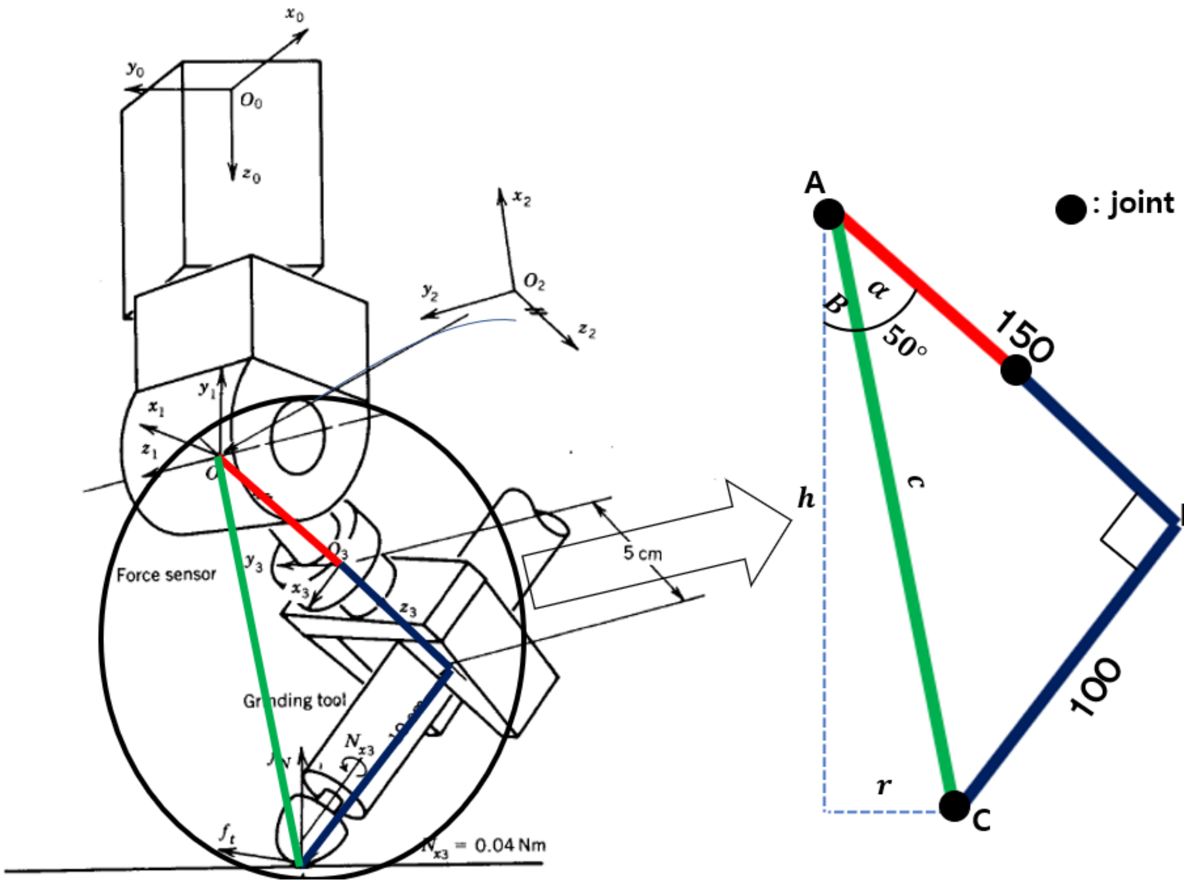


Figure 4. The PUMA 600 diagram with virtual lines and assumed lengths

```
% To obtain the numeric magnitudes, we need to obtain some configurations
a = atan(100/150);
c = sqrt(100^2+150^2);
b = (5*pi/18) - a; % (5*pi/18), assumed angle based on the figure
r = c*sin(b);
h = c*cos(b);
h_total = 400 + h;
```

$$a = 0.5880 = 33.69° \quad B = 0.2847 = 16.31°$$

$$h = 173.0226mm \quad c = 180.2776mm \quad r = 50.6279mm$$

h_total in the script is the total linear height from the working surface to O_0. Hence, for the total height, an assumption was made that the position coordinates (x_1 and y_1) of the frame 1 are the same as those of (x_0 and y_0).

3.2.2. Force/Moment magnitudes and directions, and joint angles

A set of forces (f_t, f_N *and* N_{x_3}) needed to be examined carefully to compute the joint torques. **Fig. 5** shows the directions of f_N *and* N_{x_3} according to the PUMA600 diagram. As can be seen in the figure, two assumptions were made: 1) f_N is acting in the opposite direction of z_0 (meaning that its direction is perpendicular to the working surface), 2) N_{x_3} is acting in the opposite direction of x_3. As it was unclear about which direction f_t was acting, a further observation was needed.

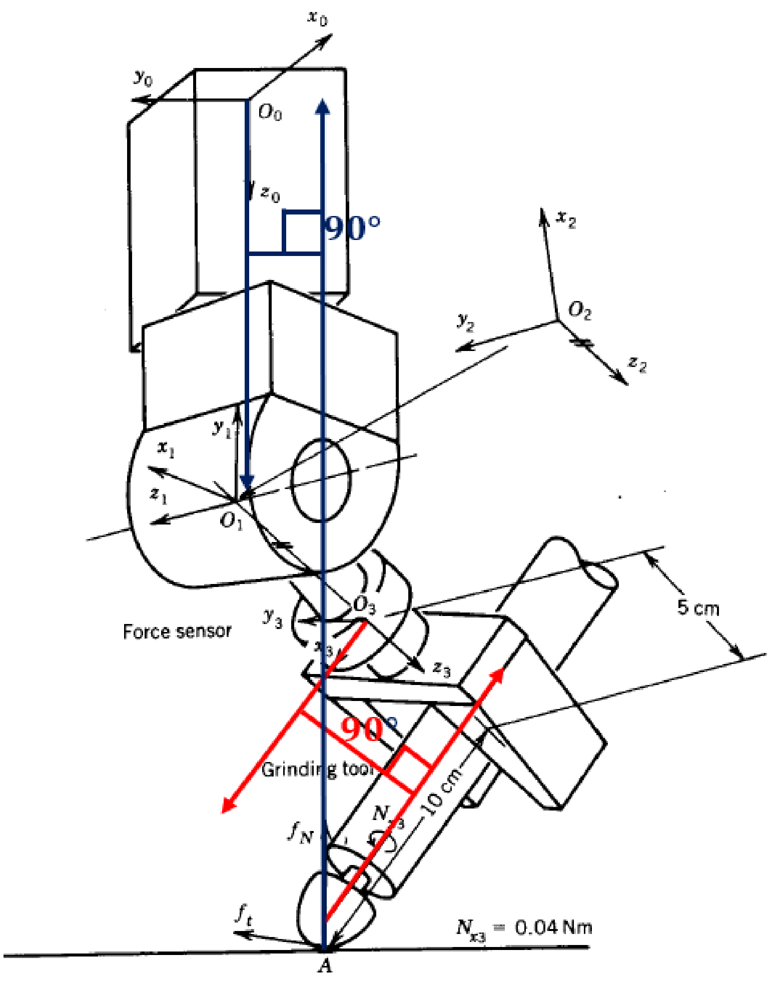


Figure 5. Directions of f_N *and* N_{x_3}

After the further observation, multiple planes were drawn to eventually discover the direction of f_t. **Fig. 6** illustrates how each plane is correlated with each other. Based on the virtual planes created in the figure, several assumptions were drawn. Firstly, as mentioned in the previously section, the two plane center lines (l_{p1} *and* l_{p2}) are perpendicular to each other, meaning that each plane is also perpendicular to each other. This first assumption leads to discover the direc-

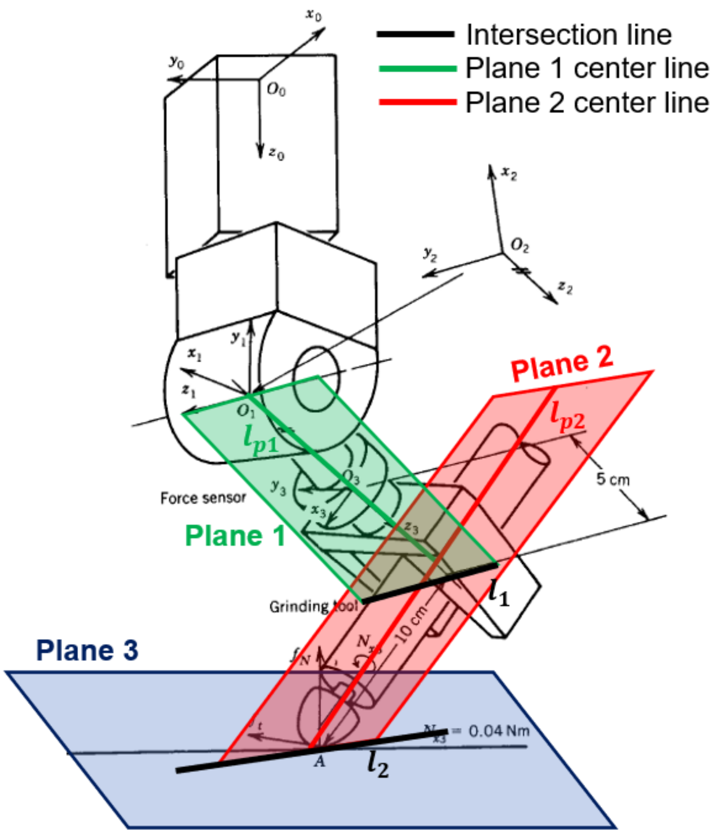


Figure 6. Multiple virtual planes drawn in the PUMA600 diagram

tion of the velocity of the tool; that is, the intersection line, l_2, represents the direction of the tool velocity. Hence, the second assumption drawn was that the intersection line, l_2, is always parallel to the axis of z_1 during the grinding operation. Thus, in total, four assumptions were drawn from the lines (**Fig. 5)** and the virtual plans (**Fig. 6)** as below:

1. The direction of $N_{x_3} \parallel x_3$

2. The direction of $f_N \parallel z_0$

3. $l_{p1} \perp l_{p2}, hence\ Plane1 \perp \text{Plane2}$

4. l_2 and the direction of the tool velocity are always parallel to z_1 during the grinding operation.

The assumptions made thus far are sufficient to compute the direction of f_t after assigning each force (f_t and f_N) with reasonable force magnitudes (15N for both forces).

```
ft = 15; fN = 15; Nx3 = 0.04; % moment and forces pre-defined
ft_direction = -link(1).mat(1:3,3); fN_direction = [0;0;-1];
R0_3 = T0_3(1:3,1:3); % Rotation Matrix of the frame 3
P0_3 = T0_3(1:3,4); % Position at the frame 3
fN = [(ft*ft_direction)+(fN*fN_direction); Nx3*R0_3(:,1)]; %R0_3(:,1) : first column because the moment is along x-axis of the frame 3
```

Strictly following the four assumptions, we obtained numeric values of the desired vectors.

1. As the direction of N_{x_3} is parallel to x_3, its direction can be written as:

$$Dir_{N_{x_3}} = {}^{0}_{3}T(1:3,1) = \begin{bmatrix} cos\theta_1 cos\theta_2 cos\theta_3 - sin\theta_1 sin\theta_3 \\ cos\theta_1 sin\theta_3 + cos\theta_2 cos\theta_3 sin\theta_1 \\ -cos\theta_3 sin\theta_2 \end{bmatrix}$$

2. As the direction of f_N is parallel to z_0, its direction can be written as:

$$Dir_{f_N} = -z_0 = -\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

3. As the direction of the tool velocity is parallel to z_1, the direction of f_t can be written as:

$$Dir_{f_t} = {}^{0}_{1}A(1:3,3) = \begin{bmatrix} sin\theta_1 \\ -cos\theta_1 \\ 0 \end{bmatrix}$$

Hence, we obtain:

$$f = f_N * Dir_{f_N} + f_t * Dir_{f_t} = \begin{bmatrix} 15 sin\theta_1 \\ -15 cos\theta_1 \\ -15 \end{bmatrix}$$

$$N = N_{x_3} * Dir_{N_{x_3}} = \begin{bmatrix} \frac{cos\theta_1 cos\theta_2 cos\theta_3 - sin\theta_1 sin\theta_3}{25} \\ \frac{cos\theta_1 sin\theta_3 + cos\theta_2 cos\theta_3 sin\theta_1}{25} \\ -\frac{cos\theta_3 sin\theta_2}{25} \end{bmatrix}$$

$$F = \begin{bmatrix} f \\ N \end{bmatrix} = \begin{bmatrix} 15 sin\theta_1 \\ -15 cos\theta_1 \\ -15 \\ \frac{cos\theta_1 cos\theta_2 cos\theta_3 - sin\theta_1 sin\theta_3}{25} \\ \frac{cos\theta_1 sin\theta_3 + cos\theta_2 cos\theta_3 sin\theta_1}{25} \\ -\frac{cos\theta_3 sin\theta_2}{25} \end{bmatrix}$$

To compute the joint angles, instead of using the cumbersome kinematic inverse approach (as in **Section 1.2**), a mathematical optimization method was employed as it is known to effectively solve inverse kinematics for any generic robot arm.

```
% Computation of the initial joint angles by using the optimisation method
x0 = [0 0 0]; % Random values for the optimisation of the nonliear equations of P0_4
initial_joint_angles = double(fsolve(@optim,x0,optimoptions('fsolve','Display','iter')));
```

x0 in the above script is an initial guess that determines the search direction, which can either lead to a local minimum, or a global minimum. By using the *fsolve* function in MATLAB, the initial joint angles were computed:

$$\theta_1 = -0.4696 = -26.89°$$

$$\theta_2 = -0.6559 = -37.58°$$

$$\theta_3 = 0.3335 = 19.11°$$

Hence, F can be re-written as:

```
% Force computation
Ft = double(subs(fN,[t1,t2,t3],initial_joint_angles)); % The force and moments on the end-effector
```

$$F = \begin{bmatrix} f \\ N \end{bmatrix} = \begin{bmatrix} 15sin\theta_1 \\ -15cos\theta_1 \\ -15 \\ \dfrac{cos\theta_1 cos\theta_2 cos\theta_3 - sin\theta_1 sin\theta_3}{25} \\ \dfrac{cos\theta_1 sin\theta_3 + cos\theta_2 cos\theta_3 sin\theta_1}{25} \\ -\dfrac{cos\theta_3 sin\theta_2}{25} \end{bmatrix} = \begin{bmatrix} -6.7842 \\ -13.3781 \\ -15.0000 \\ 0.0326 \\ -0.0019 \\ 0.0231 \end{bmatrix}$$

With the answers obtained thus far, equivalent joint torques (τ) can be computed as below:

$$\tau = J^T F \cdot 10^{-3} \text{ where } F = \begin{bmatrix} f \\ N \end{bmatrix}$$

As the analytical solution is over-complicated, only the numerical solution is computed:

$$\tau = \begin{bmatrix} -1.1232 \\ 1.1233 \\ -1.1179 \end{bmatrix} Nm$$

The codes for the above-described computation can be found in **Section 4.3**.

3.3. Simulation of the tool movement

Before writing a simulation script for the tool movement, the joint angles obtained previously in **Section 3.2.2** were double-checked roughly by drawing a 3-D tool diagram and comparing it to the PUMA 600 diagram given in the question. **Fig. 7** shows two diagrams for a comparison. As can be observed, the two diagrams look similar, following the assumption ($l_{p1} \perp l_{p2}$). The reason why l_{AB} is shorter in **Fig. 7(a)** is because the length, pre-defined in **Section 3.2.1**, happened to be shorter than the actual length in the given diagram, which can be fixed simply if the length is set to be longer.

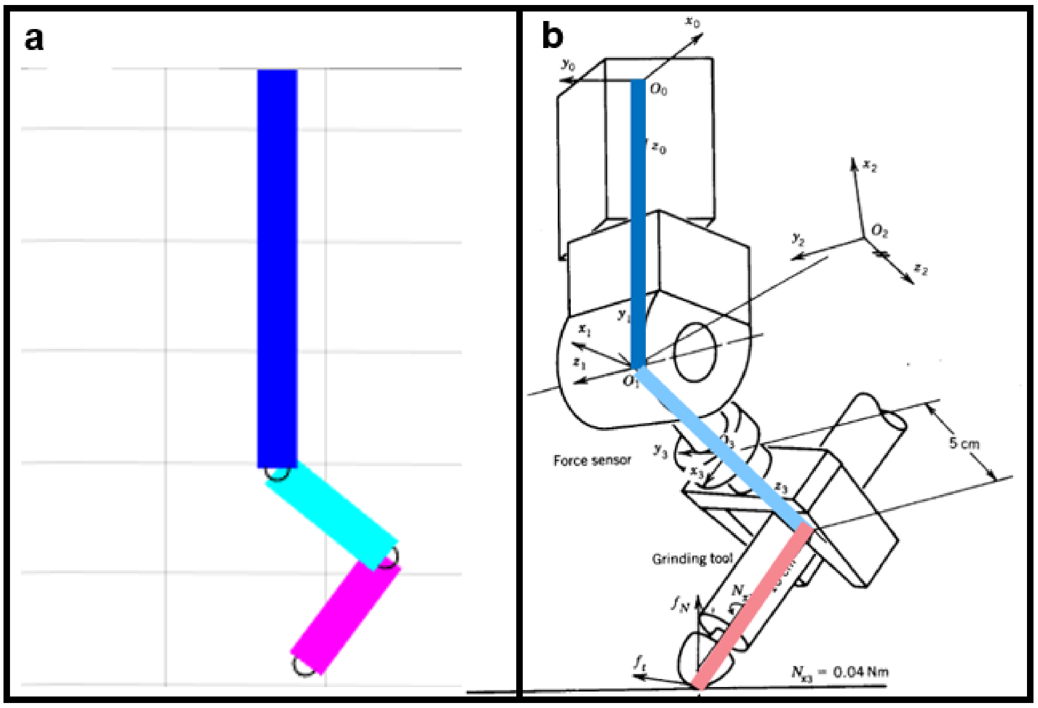


Figure 7. A comparison between the 3-D drawn plot and the given diagram (a: the 3-D drawn plot, b: the given diagram)

Before constructing a simulation of the end-point position, several assumptions were made as follows:

1. The angles $(a \ and \ B)$ obtained in **Section 3.2.1** remain fixed during the movement.

2. The manipulator rotates only about z_0-axis.

3. The joint mechanism of the manipulator is frictionless.

4. The operation time is 60 seconds.

5. The joint variables $(\theta_2 \ and \ \theta_3)$ remain fixed and θ_1 rotates while in operation.

3.3.1. The path of the tool tip

The following script was used to generate the necessary variables.

```
syms t % t for simulation (each angle varies with respect to t)
% X : position of the frame 3 with time (t)
X = 100*[cos(t*(pi/180))*sin(t*(pi/180)) ; sin(t*(pi/180))*sin(t*(pi/180)) ; cos(t*(pi/180)) + 4];
P_real = simplify(P0_3);
% Construct equations for the position of the frame 3 to obtain the
% answers of "t"
equation = [X(1)==P_real(1) X(2)==P_real(2) X(3)==P_real(3)];
answer = solve(equation,[t1, t2, t3]);

% from the answer, we obtained the following joint angles with respect to time.
% t1_new will be used for plotting
t1_new = 2*atan(tan((pi*t)/360)); %From the answer obtained above
```

The position vector (${}^{0}_{3}P$) was used to compute the joint angle rate per second as below:

$$ {}^{0}_{3}P = \begin{bmatrix} 100\cos(\theta_1)\sin(\theta_2) \\ 100\sin(\theta_1)\,sin(\theta_2) \\ 100\cos(\theta_2) + 400 \end{bmatrix} $$

Given that the vector increases by the variable (t) per second, a new vector was created:

$$ {}^{0}_{3}X = \begin{bmatrix} 100\cos(t)\sin(t) \\ 100\sin(t)\sin(t) \\ 100\cos(t) + 400 \end{bmatrix} $$

Solutions for the joint variables ($\theta_1,\ \theta_2\ and\ \theta_3$) were obtained:

$$ {}^{0}_{3}P = {}^{0}_{3}X \rightarrow \begin{bmatrix} 100\cos(\theta_1)\sin(\theta_2) \\ 100\sin(\theta_1)\,sin(\theta_2) \\ 100\cos(\theta_2) + 400 \end{bmatrix} = \begin{bmatrix} 100\cos(t)\sin(t) \\ 100\sin(t)\sin(t) \\ 100\cos(t) + 400 \end{bmatrix} $$

$$ \theta_1 = 2\arctan(\tan\left(\left(\frac{\pi t}{360}\right)\right)) $$

$$ \theta_2 = \theta_3 = 0 $$

The initial joint angles ($\theta_1 = -0.4696, \theta_2 = -0.6559\ and\ \theta_3 = 0.3335$) obtained in **Section 3.2.2**, were used as the starting point on the work surface. As previously assumed, the joint angles ($\theta_2\ and\ \theta_3$) remain fixed whereas the other one (θ_1) is the only joint angle that changes with respect to time while in operation. Hence, the new joint angle (θ_1) with respect time can be denoted as:

$$ \theta_1 = -0.4696 + 2\arctan(\tan\left(\left(\frac{\pi t}{360}\right)\right)) $$

Thus, the path of the tool tip on the work surface for the period of 60 seconds could be plotted as shown in **Fig. 8**.

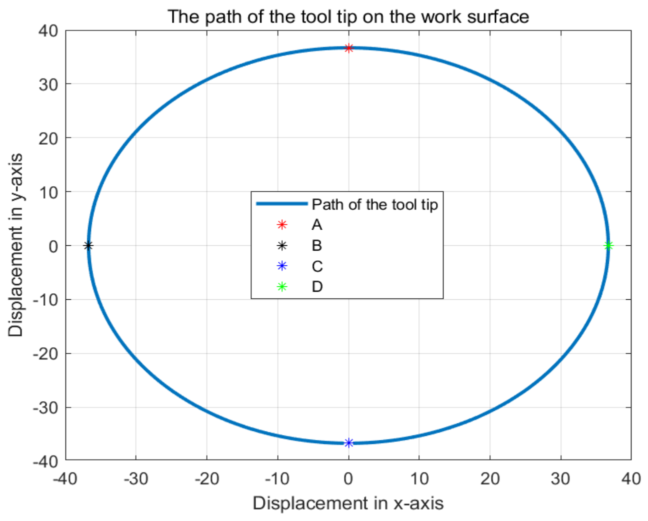


Figure 8. The path of the tool tip on the work surface

Furthermore, by using the new joint angles, a graph for the respective joint angles versus time could be illustrated as in **Fig. 9**.

$$\theta_1 = -0.4696 + 2\arctan(\tan\left(\left(\frac{\pi t}{360}\right)\right))$$

$$\theta_2 = \theta_3 = 0$$

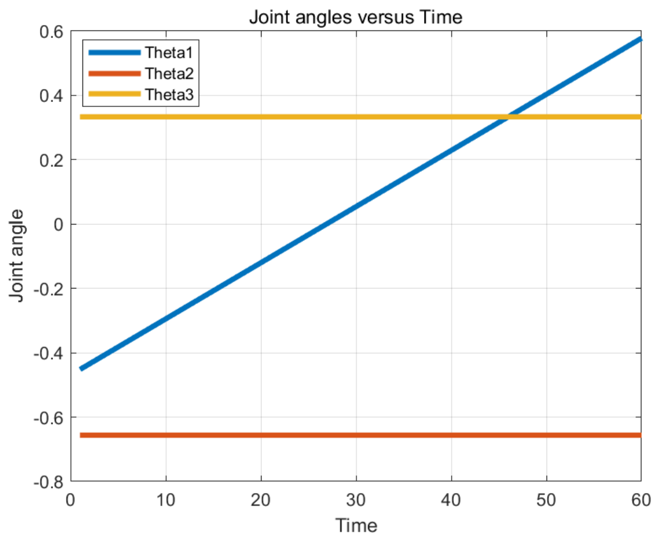


Figure 9. The change in each joint angle with respect to time

3.3.2. Respective joint rates and joint torques versus time

Joint rates (change in velocity for each joint) can be computed by **Eq. 12**:

$$\dot{q} = J^{-1}\dot{p} \qquad [12]$$

where $\dot{q}$ is $n \times 1$ joint velocity vector, J is the Jacobian matrix, and $\dot{p}$ is an end-effect velocity vector. As the Jacobian matrix was not a square matrix, The *Pseudo-inverse* was used to obtain its inverse matrix. Using the below script, a graph for the joint rates versus time could be obtained as shown in **Fig. 10(a)**. The joint rate 2 completely overlaps with the joint rate 3 with the value 0 as they were assumed to remain fixed while in motion.

```
J_new = simplify(subs(J,[t1,t2,t3],[initial_joint_angles(1)+t1_new,initial_joint_angles(2),initial_joint_angles(3)]));
J_inv = pinv(J_new); % Pseudo-inverse so that we can calculate dq=J^-1*dp
P0_3_change = subs(P0_3,[t1, t2,t3],[initial_joint_angles(1)+t1_new, initial_joint_angles(2),initial_joint_angles(3)]);
Velocity = diff(P0_3_change,t); % Velocity at the frame 3
q = simplify(J_inv*[Velocity;0;0;1]); % Joint rate
```

For the joint moments with respect to time, the following equations were used:

```
f0_3 = F_sym;
R3_0 = transpose(R0_3);
f3_3 = [R3_0 zeros(3,3) ; zeros(3,3) R3_0]*f0_3;
f3_3 = eval(subs(f3_3,[t1, t2,t3],[initial_joint_angles(1)+t1_new, initial_joint_angles(2),initial_joint_angles(3)]));
torque_tooltip = transpose(J_new)*f3_3;
```

$$\begin{bmatrix} {}^B_B f \\ {}^B_B n \end{bmatrix} = \begin{bmatrix} {}^3_0 R & O_{3\times3} \\ O_{3\times3} & {}^3_0 R \end{bmatrix} \begin{bmatrix} {}^A_B f \\ {}^A_B n \end{bmatrix}$$

$$
{}^{3}_{0}R = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & c_1s_3 + c_2c_3s_1 & -c_3s_2 \\ -c_3s_1 - c_1c_2s_3 & c_1c_3 - c_2s_1s_3 & s_2s_3 \\ c_1s_2 & s_1s_2 & c_2 \end{bmatrix}
$$

$$
\tau_{Tool\_tip} = J^T \begin{bmatrix} {}^{B}_{B}f \\ {}^{B}_{B}n \end{bmatrix} \cdot 10^{-3}
$$

where A is the orientation frame and B is the frame of the tool tip. Consequently, the joint moments with respect to time were plotted as shown in **Fig. 10(b)**. The full codes for plotting the simulation graphs can be found in **Section 4.4**.

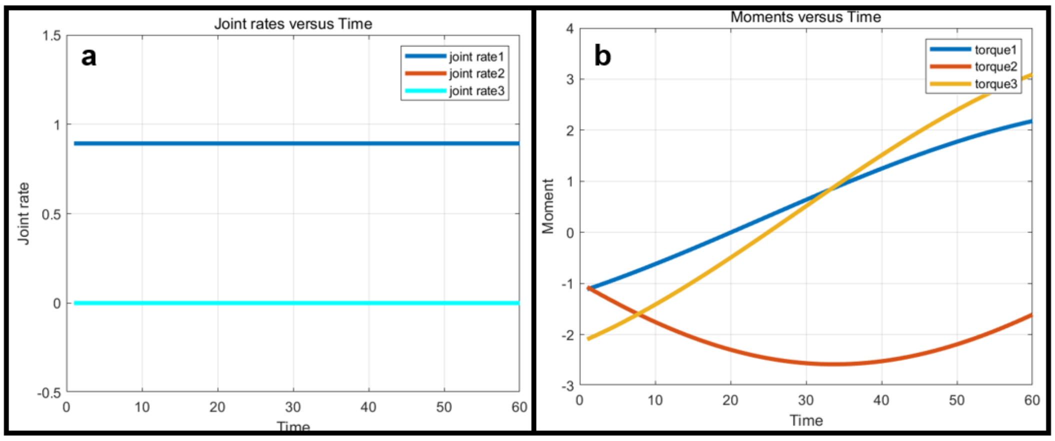
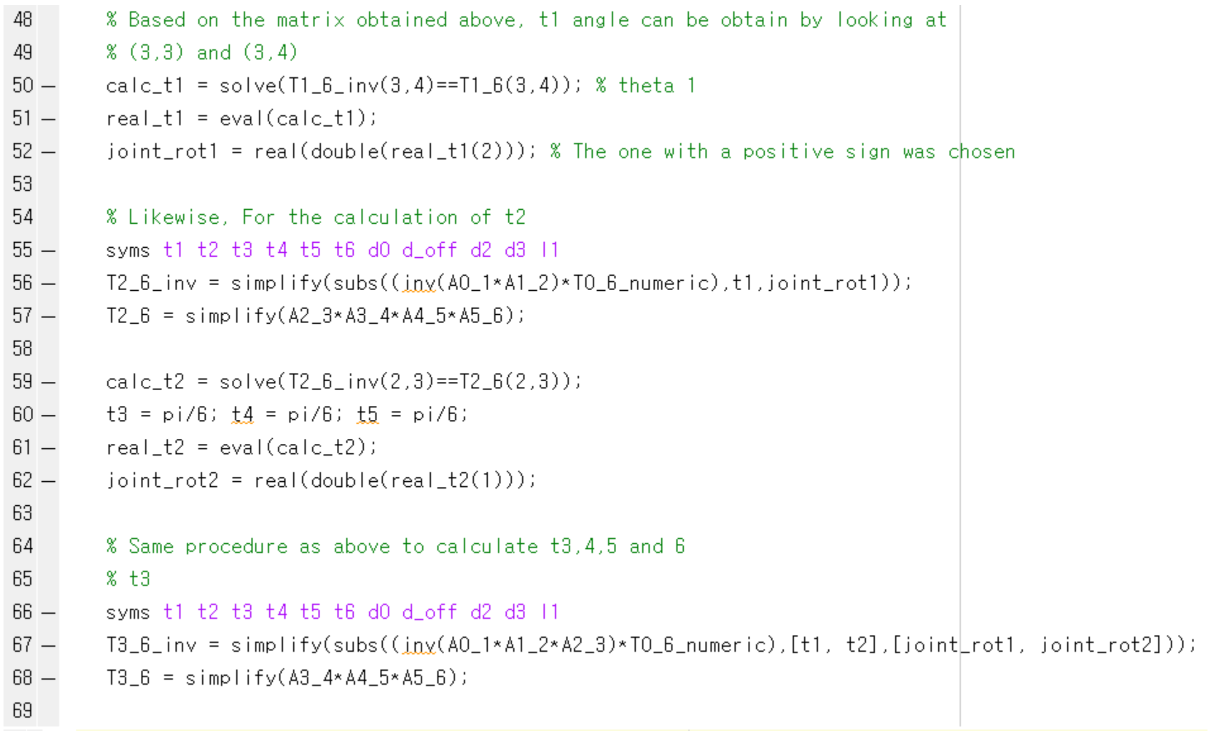


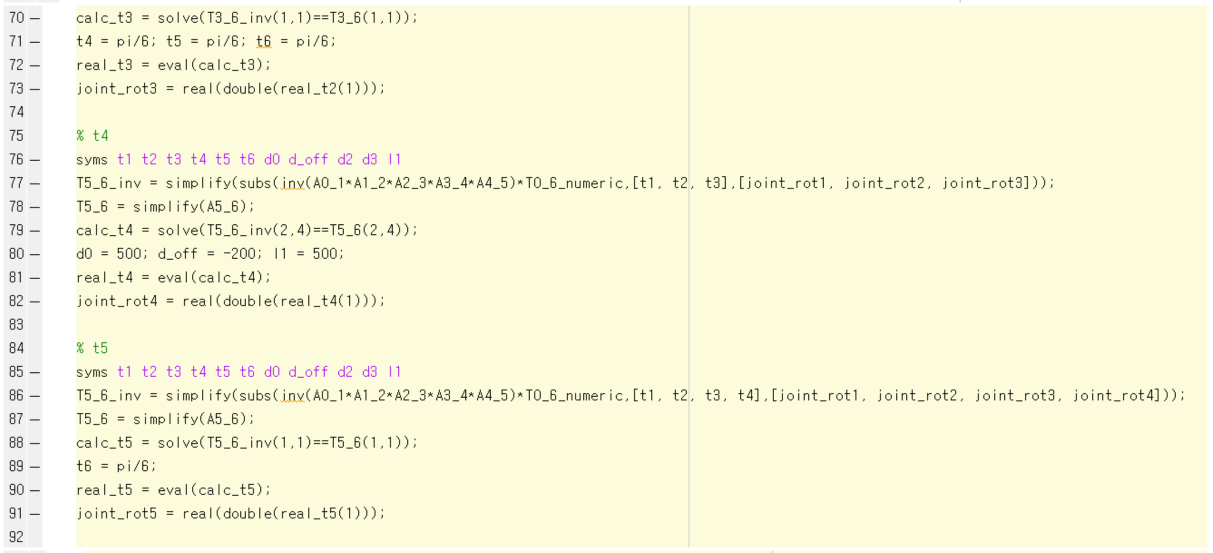
Figure 10. A graph for the joint rates and joint torques versus time

4. Appendix

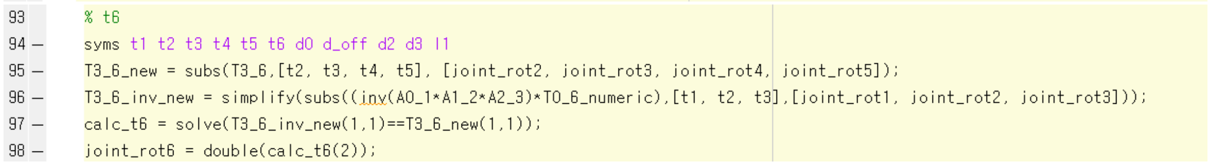
4.1. Question 1 - Inverse kinematics for the joint angles



```
% Based on the matrix obtained above, t1 angle can be obtain by looking at
% (3,3) and (3,4)
calc_t1 = solve(T1_6_inv(3,4)==T1_6(3,4)); % theta 1
real_t1 = eval(calc_t1);
joint_rot1 = real(double(real_t1(2))); % The one with a positive sign was chosen

% Likewise, For the calculation of t2
syms t1 t2 t3 t4 t5 t6 d0 d_off d2 d3 l1
T2_6_inv = simplify(subs((inv(A0_1*A1_2)*T0_6_numeric),t1,joint_rot1));
T2_6 = simplify(A2_3*A3_4*A4_5*A5_6);

calc_t2 = solve(T2_6_inv(2,3)==T2_6(2,3));
t3 = pi/6; t4 = pi/6; t5 = pi/6;
real_t2 = eval(calc_t2);
joint_rot2 = real(double(real_t2(1)));

% Same procedure as above to calculate t3,4,5 and 6
% t3
syms t1 t2 t3 t4 t5 t6 d0 d_off d2 d3 l1
T3_6_inv = simplify(subs((inv(A0_1*A1_2*A2_3)*T0_6_numeric),[t1, t2],[joint_rot1, joint_rot2]));
T3_6 = simplify(A3_4*A4_5*A5_6);

```

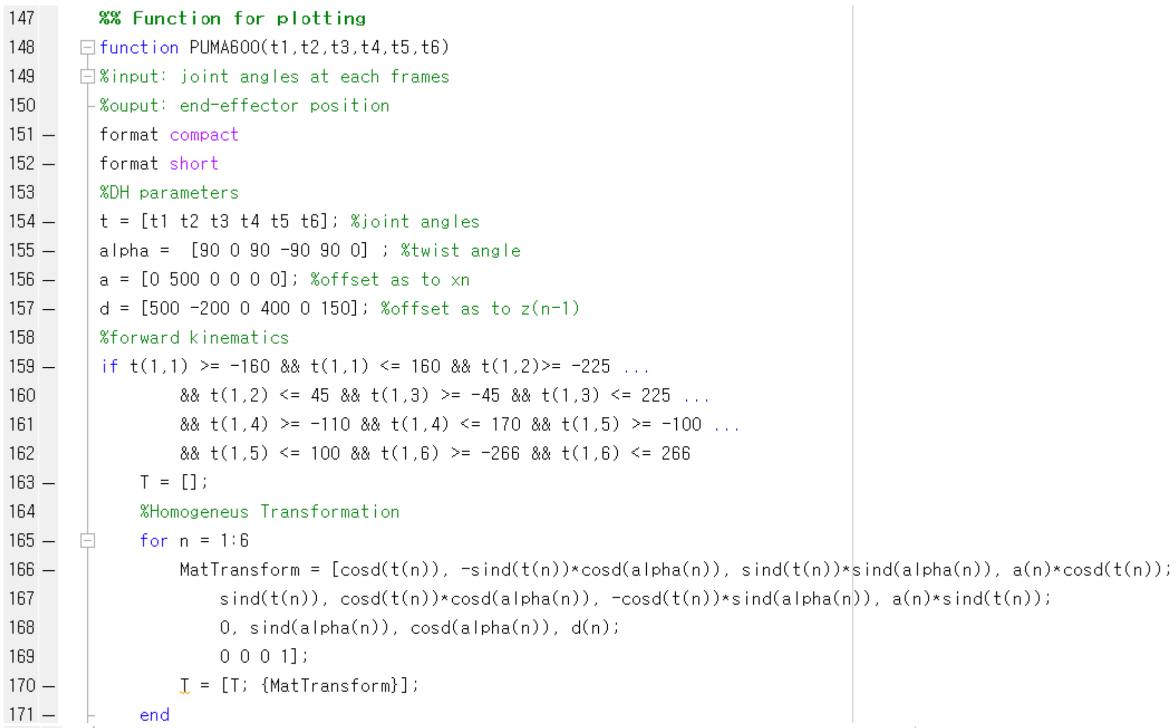


```
calc_t3 = solve(T3_6_inv(1,1)==T3_6(1,1));
t4 = pi/6; t5 = pi/6; t6 = pi/6;
real_t3 = eval(calc_t3);
joint_rot3 = real(double(real_t2(1)));

% t4
syms t1 t2 t3 t4 t5 t6 d0 d_off d2 d3 l1
T5_6_inv = simplify(subs(inv(A0_1*A1_2*A2_3*A3_4*A4_5)*T0_6_numeric,[t1, t2, t3],[joint_rot1, joint_rot2, joint_rot3]));
T5_6 = simplify(A5_6);
calc_t4 = solve(T5_6_inv(2,4)==T5_6(2,4));
d0 = 500; d_off = -200; l1 = 500;
real_t4 = eval(calc_t4);
joint_rot4 = real(double(real_t4(1)));

% t5
syms t1 t2 t3 t4 t5 t6 d0 d_off d2 d3 l1
T5_6_inv = simplify(subs(inv(A0_1*A1_2*A2_3*A3_4*A4_5)*T0_6_numeric,[t1, t2, t3, t4],[joint_rot1, joint_rot2, joint_rot3, joint_rot4]));
T5_6 = simplify(A5_6);
calc_t5 = solve(T5_6_inv(1,1)==T5_6(1,1));
t6 = pi/6;
real_t5 = eval(calc_t5);
joint_rot5 = real(double(real_t5(1)));

```

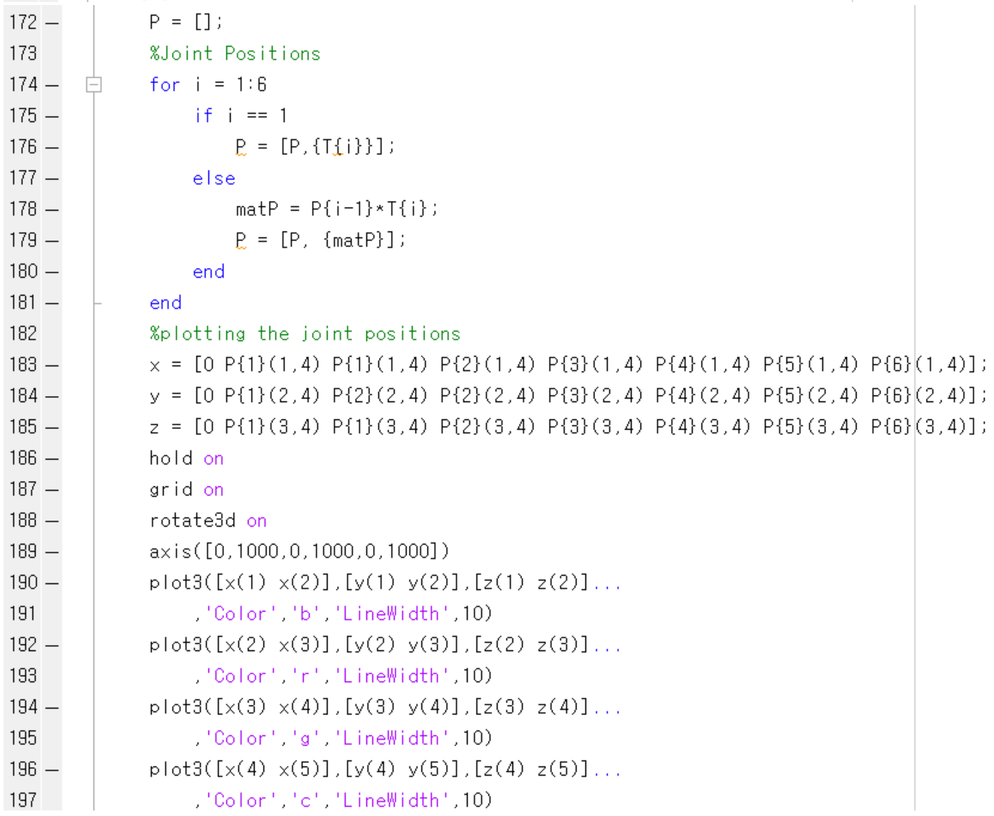


```
% t6
syms t1 t2 t3 t4 t5 t6 d0 d_off d2 d3 l1
T3_6_new = subs(T3_6,[t2, t3, t4, t5], [joint_rot2, joint_rot3, joint_rot4, joint_rot5]);
T3_6_inv_new = simplify(subs((inv(A0_1*A1_2*A2_3)*T0_6_numeric),[t1, t2, t3],[joint_rot1, joint_rot2, joint_rot3]));
calc_t6 = solve(T3_6_inv_new(1,1)==T3_6_new(1,1));
joint_rot6 = double(calc_t6(2));
```

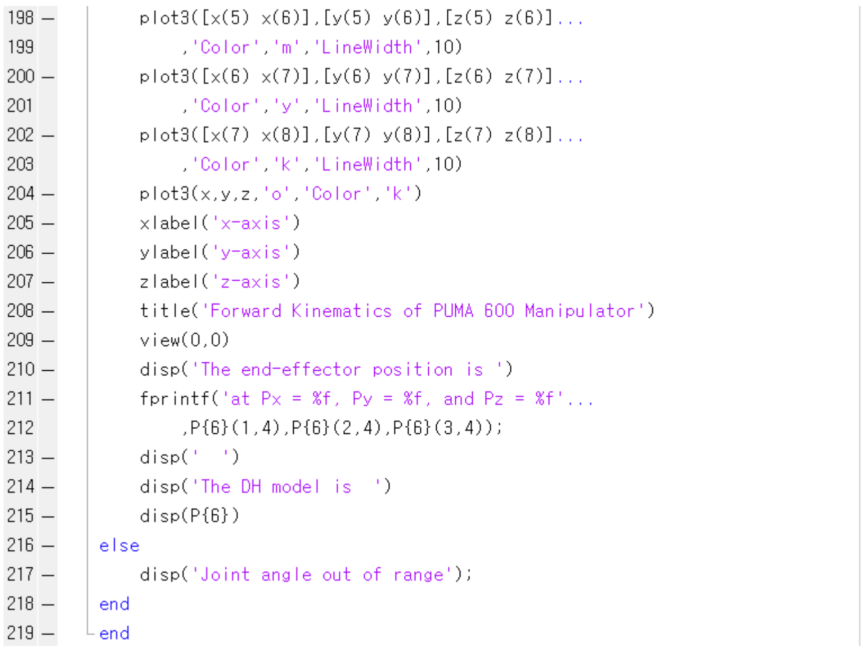
4.2. Question 1 - The Function for plotting the PUMA 600 diagram



```
%% Function for plotting
function PUMA600(t1,t2,t3,t4,t5,t6)
%input: joint angles at each frames
%ouput: end-effector position
format compact
format short
%DH parameters
t = [t1 t2 t3 t4 t5 t6]; %joint angles
alpha =  [90 0 90 -90 90 0] ; %twist angle
a = [0 500 0 0 0 0]; %offset as to xn
d = [500 -200 0 400 0 150]; %offset as to z(n-1)
%forward kinematics
if t(1,1) >= -160 && t(1,1) <= 160 && t(1,2)>= -225 ...
        && t(1,2) <= 45 && t(1,3) >= -45 && t(1,3) <= 225 ...
        && t(1,4) >= -110 && t(1,4) <= 170 && t(1,5) >= -100 ...
        && t(1,5) <= 100 && t(1,6) >= -266 && t(1,6) <= 266
    T = [];
    %Homogeneus Transformation
    for n = 1:6
        MatTransform = [cosd(t(n)), -sind(t(n))*cosd(alpha(n)), sind(t(n))*sind(alpha(n)), a(n)*cosd(t(n));
            sind(t(n)), cosd(t(n))*cosd(alpha(n)), -cosd(t(n))*sind(alpha(n)), a(n)*sind(t(n));
            0, sind(alpha(n)), cosd(alpha(n)), d(n);
            0 0 0 1];
        T = [T; {MatTransform}];
    end
```

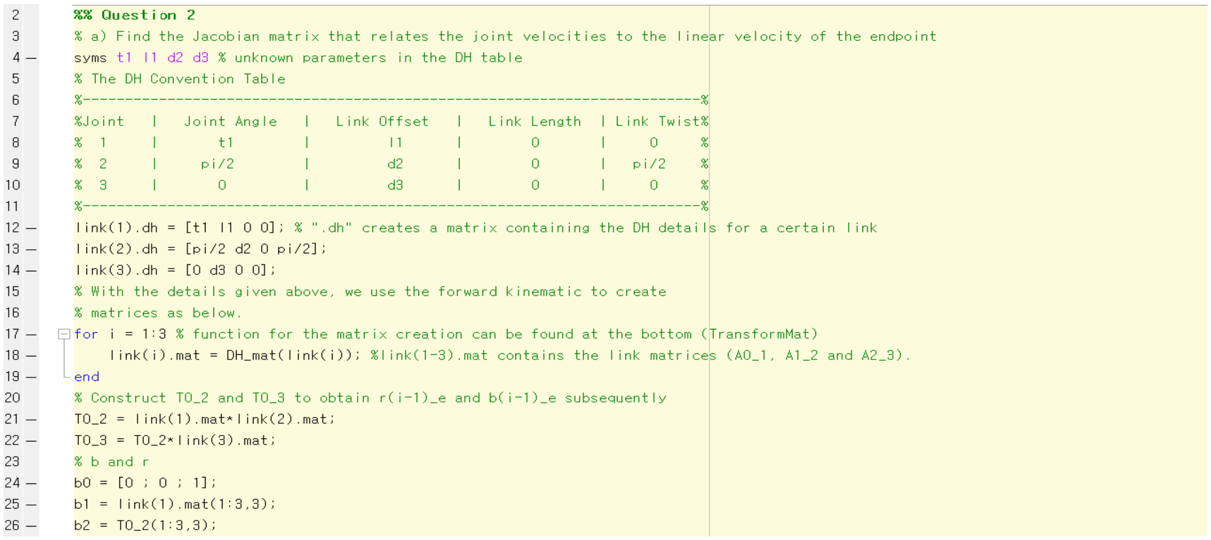


```
    P = [];
    %Joint Positions
    for i = 1:6
        if i == 1
            P = [P,{T{i}}];
        else
            matP = P{i-1}*T{i};
            P = [P, {matP}];
        end
    end
    %plotting the joint positions
    x = [0 P{1}(1,4) P{1}(1,4) P{2}(1,4) P{3}(1,4) P{4}(1,4) P{5}(1,4) P{6}(1,4)];
    y = [0 P{1}(2,4) P{2}(2,4) P{2}(2,4) P{3}(2,4) P{4}(2,4) P{5}(2,4) P{6}(2,4)];
    z = [0 P{1}(3,4) P{1}(3,4) P{2}(3,4) P{3}(3,4) P{4}(3,4) P{5}(3,4) P{6}(3,4)];
    hold on
    grid on
    rotate3d on
    axis([0,1000,0,1000,0,1000])
    plot3([x(1) x(2)],[y(1) y(2)],[z(1) z(2)]...
        ,'Color','b','LineWidth',10)
    plot3([x(2) x(3)],[y(2) y(3)],[z(2) z(3)]...
        ,'Color','r','LineWidth',10)
    plot3([x(3) x(4)],[y(3) y(4)],[z(3) z(4)]...
        ,'Color','g','LineWidth',10)
    plot3([x(4) x(5)],[y(4) y(5)],[z(4) z(5)]...
        ,'Color','c','LineWidth',10)
```

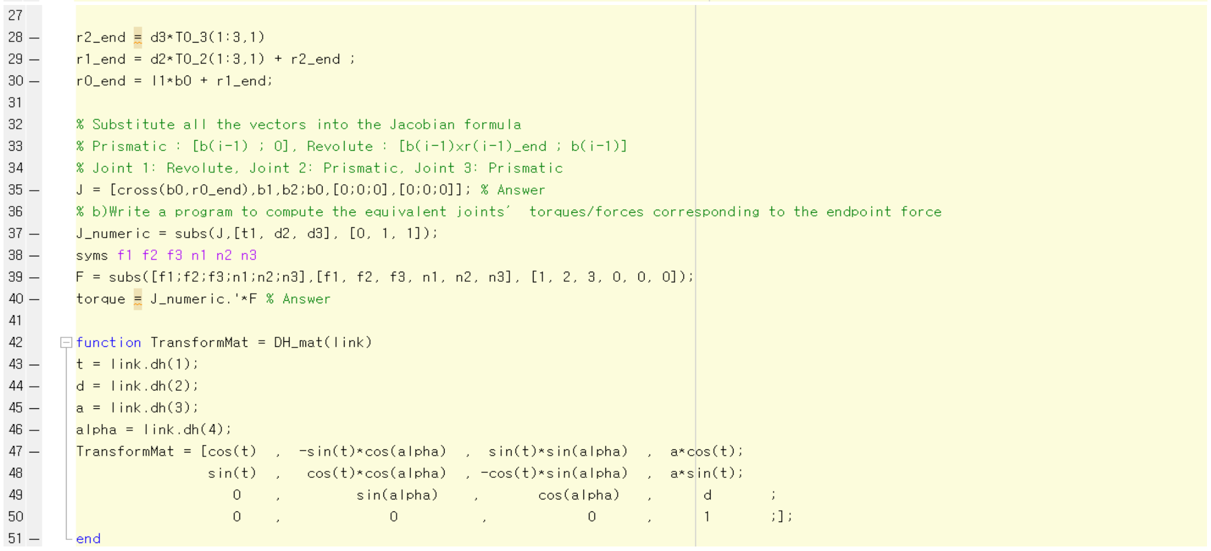


```matlab
        plot3([x(5) x(6)],[y(5) y(6)],[z(5) z(6)]...
            ,'Color','m','LineWidth',10)
        plot3([x(6) x(7)],[y(6) y(7)],[z(6) z(7)]...
            ,'Color','y','LineWidth',10)
        plot3([x(7) x(8)],[y(7) y(8)],[z(7) z(8)]...
            ,'Color','k','LineWidth',10)
        plot3(x,y,z,'o','Color','k')
        xlabel('x-axis')
        ylabel('y-axis')
        zlabel('z-axis')
        title('Forward Kinematics of PUMA 600 Manipulator')
        view(0,0)
        disp('The end-effector position is ')
        fprintf('at Px = %f, Py = %f, and Pz = %f'...
            ,P{6}(1,4),P{6}(2,4),P{6}(3,4));
        disp('  ')
        disp('The DH model is  ')
        disp(P{6})
    else
        disp('Joint angle out of range');
    end
end
```

4.3. Question 2 – Full Codes

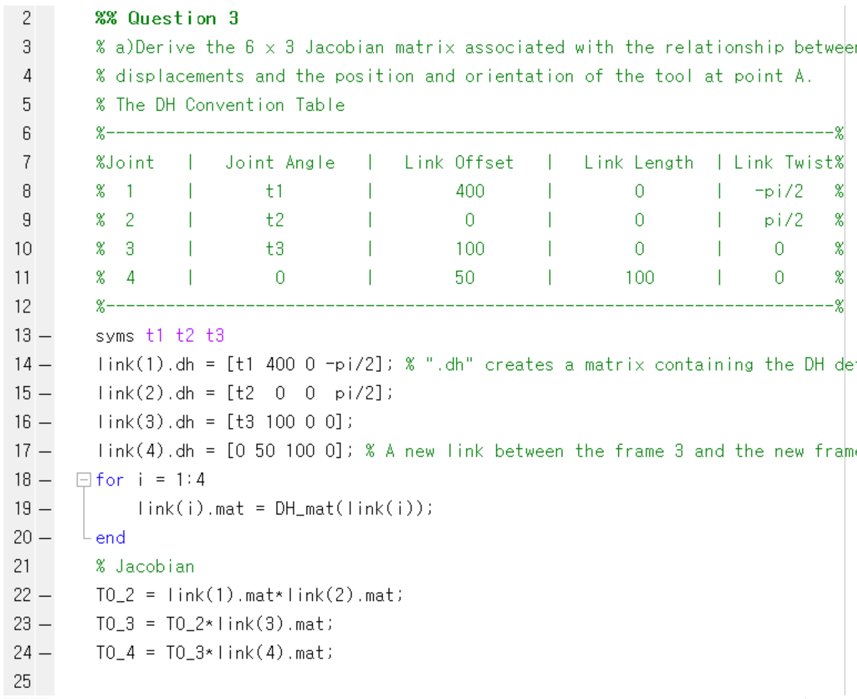


```matlab
%% Question 2
% a) Find the Jacobian matrix that relates the joint velocities to the linear velocity of the endpoint
syms t1 l1 d2 d3 % unknown parameters in the DH table
% The DH Convention Table
%-----------------------------------------------------------------------%
%Joint    |   Joint Angle    |   Link Offset    |   Link Length   | Link Twist%
%  1      |       t1         |        l1        |        0        |     0     %
%  2      |      pi/2        |        d2        |        0        |   pi/2    %
%  3      |       0          |        d3        |        0        |     0     %
%-----------------------------------------------------------------------%
link(1).dh = [t1 l1 0 0]; % ".dh" creates a matrix containing the DH details for a certain link
link(2).dh = [pi/2 d2 0 pi/2];
link(3).dh = [0 d3 0 0];
% With the details given above, we use the forward kinematic to create
% matrices as below.
for i = 1:3 % function for the matrix creation can be found at the bottom (TransformMat)
    link(i).mat = DH_mat(link(i)); %link(1-3).mat contains the link matrices (A0_1, A1_2 and A2_3).
end
% Construct T0_2 and T0_3 to obtain r(i-1)_e and b(i-1)_e subsequently
T0_2 = link(1).mat*link(2).mat;
T0_3 = T0_2*link(3).mat;
% b and r
b0 = [0 ; 0 ; 1];
b1 = link(1).mat(1:3,3);
b2 = T0_2(1:3,3);
```

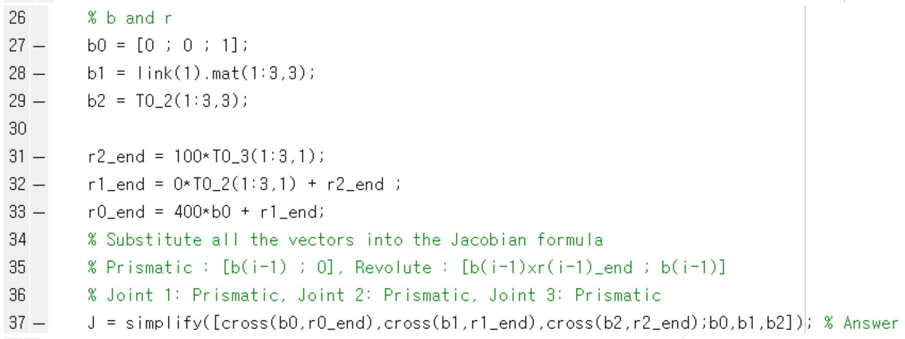


```matlab

r2_end = d3*T0_3(1:3,1)
r1_end = d2*T0_2(1:3,1) + r2_end ;
r0_end = l1*b0 + r1_end;

% Substitute all the vectors into the Jacobian formula
% Prismatic : [b(i-1) ; 0], Revolute : [b(i-1)xr(i-1)_end ; b(i-1)]
% Joint 1: Revolute, Joint 2: Prismatic, Joint 3: Prismatic
J = [cross(b0,r0_end),b1,b2;b0,[0;0;0],[0;0;0]]; % Answer
% b)Write a program to compute the equivalent joints'  torques/forces corresponding to the endpoint force
J_numeric = subs(J,[t1, d2, d3], [0, 1, 1]);
syms f1 f2 f3 n1 n2 n3
F = subs([f1;f2;f3;n1;n2;n3],[f1, f2, f3, n1, n2, n3], [1, 2, 3, 0, 0, 0]);
torque = J_numeric.'*F % Answer

function TransformMat = DH_mat(link)
t = link.dh(1);
d = link.dh(2);
a = link.dh(3);
alpha = link.dh(4);
TransformMat = [cos(t)  ,  -sin(t)*cos(alpha)  ,  sin(t)*sin(alpha)  ,  a*cos(t);
                sin(t)  ,   cos(t)*cos(alpha)  , -cos(t)*sin(alpha)  ,  a*sin(t);
                  0     ,        sin(alpha)    ,       cos(alpha)    ,      d      ;
                  0     ,           0          ,           0         ,      1      ;];
end
```

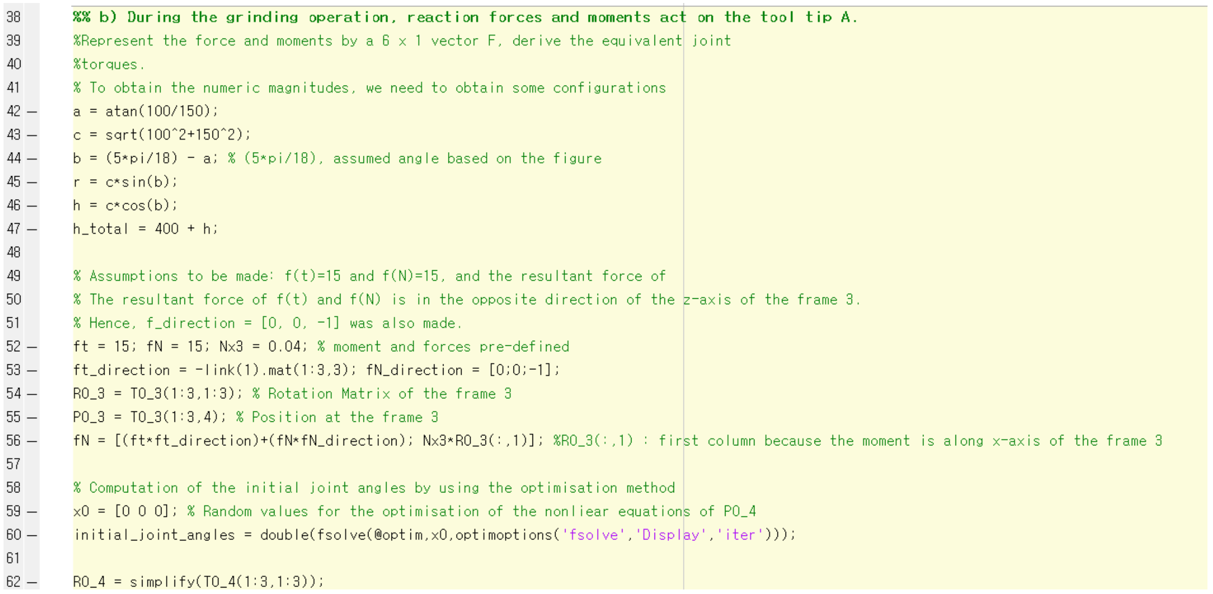
4.4. Question 3 – Derivation of the equivalent joint torques



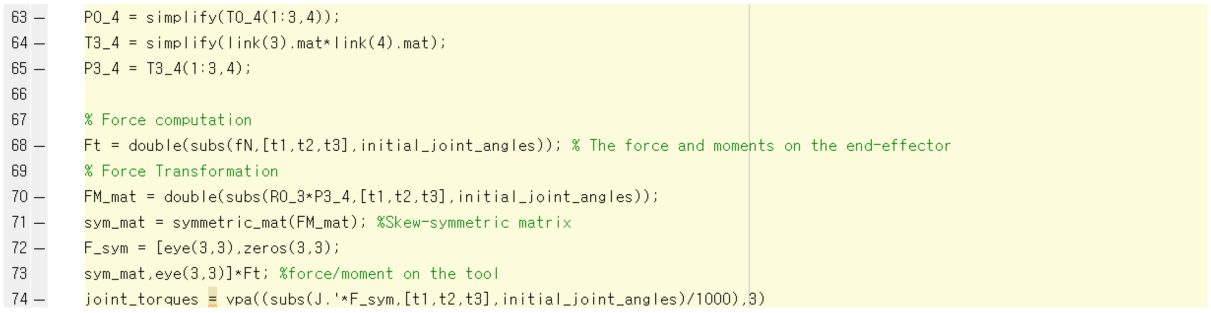
```
%% Question 3
% a)Derive the 6 x 3 Jacobian matrix associated with the relationship between joint
% displacements and the position and orientation of the tool at point A.
% The DH Convention Table
%-------------------------------------------------------------------------%
%Joint   |   Joint Angle   |   Link Offset   |   Link Length   | Link Twist%
%  1     |       t1        |       400       |        0        |   -pi/2   %
%  2     |       t2        |        0        |        0        |    pi/2   %
%  3     |       t3        |       100       |        0        |     0     %
%  4     |        0        |       50        |       100       |     0     %
%-------------------------------------------------------------------------%
syms t1 t2 t3
link(1).dh = [t1 400 0 -pi/2]; % ".dh" creates a matrix containing the DH details for a certain link
link(2).dh = [t2  0  0  pi/2];
link(3).dh = [t3 100 0 0];
link(4).dh = [0 50 100 0]; % A new link between the frame 3 and the new frame located at the grinding tool
for i = 1:4
    link(i).mat = DH_mat(link(i));
end
% Jacobian
T0_2 = link(1).mat*link(2).mat;
T0_3 = T0_2*link(3).mat;
T0_4 = T0_3*link(4).mat;

```



```
% b and r
b0 = [0 ; 0 ; 1];
b1 = link(1).mat(1:3,3);
b2 = T0_2(1:3,3);

r2_end = 100*T0_3(1:3,1);
r1_end = 0*T0_2(1:3,1) + r2_end ;
r0_end = 400*b0 + r1_end;
% Substitute all the vectors into the Jacobian formula
% Prismatic : [b(i-1) ; 0], Revolute : [b(i-1)xr(i-1)_end ; b(i-1)]
% Joint 1: Prismatic, Joint 2: Prismatic, Joint 3: Prismatic
J = simplify([cross(b0,r0_end),cross(b1,r1_end),cross(b2,r2_end);b0,b1,b2]); % Answer
```

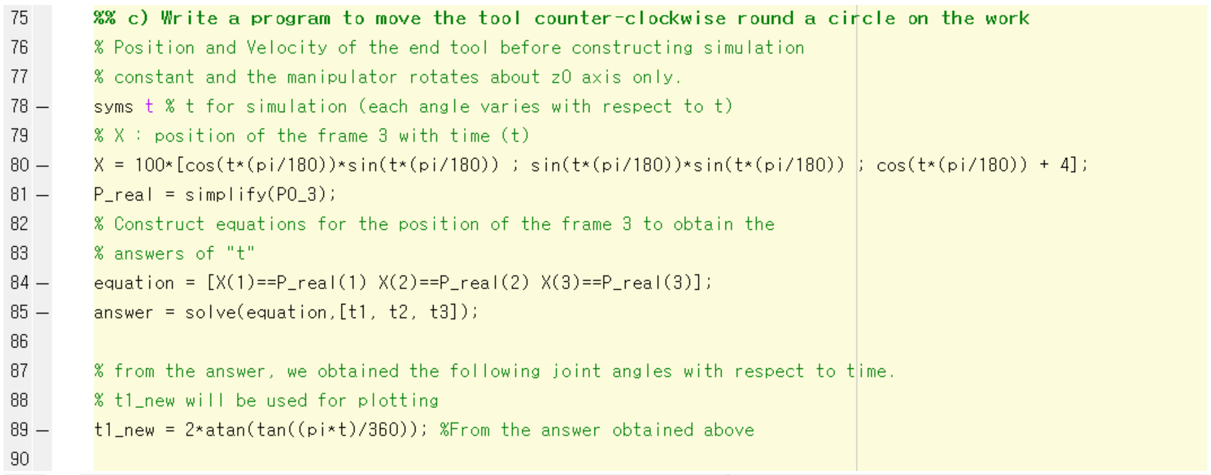


```
%% b) During the grinding operation, reaction forces and moments act on the tool tip A.
%Represent the force and moments by a 6 x 1 vector F, derive the equivalent joint
%torques.
% To obtain the numeric magnitudes, we need to obtain some configurations
a = atan(100/150);
c = sqrt(100^2+150^2);
b = (5*pi/18) - a; % (5*pi/18), assumed angle based on the figure
r = c*sin(b);
h = c*cos(b);
h_total = 400 + h;

% Assumptions to be made: f(t)=15 and f(N)=15, and the resultant force of
% The resultant force of f(t) and f(N) is in the opposite direction of the z-axis of the frame 3.
% Hence, f_direction = [0, 0, -1] was also made.
ft = 15; fN = 15; Nx3 = 0.04; % moment and forces pre-defined
ft_direction = -link(1).mat(1:3,3); fN_direction = [0;0;-1];
R0_3 = T0_3(1:3,1:3); % Rotation Matrix of the frame 3
P0_3 = T0_3(1:3,4); % Position at the frame 3
fN = [(ft*ft_direction)+(fN*fN_direction); Nx3*R0_3(:,1)]; %R0_3(:,1) : first column because the moment is along x-axis of the frame 3

% Computation of the initial joint angles by using the optimisation method
x0 = [0 0 0]; % Random values for the optimisation of the nonliear equations of P0_4
initial_joint_angles = double(fsolve(@optim,x0,optimoptions('fsolve','Display','iter')));

R0_4 = simplify(T0_4(1:3,1:3));
```

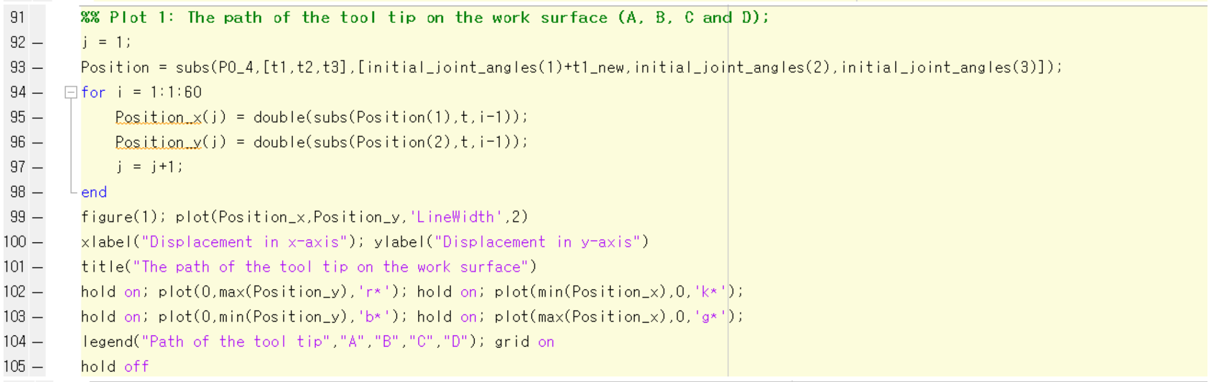


```matlab
P0_4 = simplify(T0_4(1:3,4));
T3_4 = simplify(link(3).mat*link(4).mat);
P3_4 = T3_4(1:3,4);

% Force computation
Ft = double(subs(fN,[t1,t2,t3],initial_joint_angles)); % The force and moments on the end-effector
% Force Transformation
FM_mat = double(subs(R0_3*P3_4,[t1,t2,t3],initial_joint_angles));
sym_mat = symmetric_mat(FM_mat); %Skew-symmetric matrix
F_sym = [eye(3,3),zeros(3,3);
sym_mat,eye(3,3)]*Ft; %force/moment on the tool
joint_torques = vpa((subs(J.'*F_sym,[t1,t2,t3],initial_joint_angles)/1000),3)
```

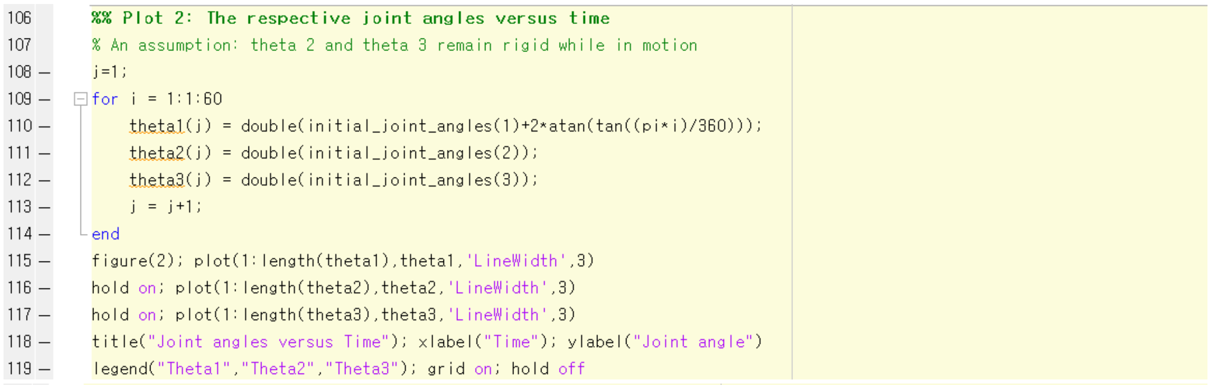
4.5. Question 3 – Simulation of PUMA 600



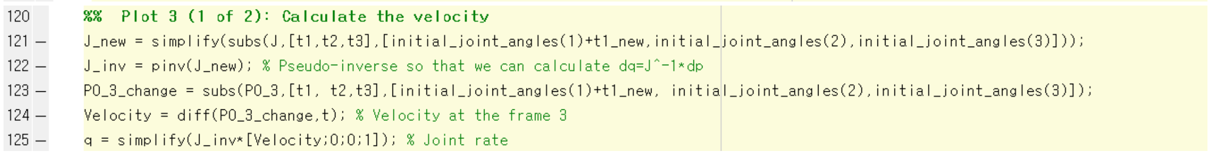
```matlab
%% c) Write a program to move the tool counter-clockwise round a circle on the work
% Position and Velocity of the end tool before constructing simulation
% constant and the manipulator rotates about z0 axis only.
syms t % t for simulation (each angle varies with respect to t)
% X : position of the frame 3 with time (t)
X = 100*[cos(t*(pi/180))*sin(t*(pi/180)) ; sin(t*(pi/180))*sin(t*(pi/180)) ; cos(t*(pi/180)) + 4];
P_real = simplify(P0_3);
% Construct equations for the position of the frame 3 to obtain the
% answers of "t"
equation = [X(1)==P_real(1) X(2)==P_real(2) X(3)==P_real(3)];
answer = solve(equation,[t1, t2, t3]);

% from the answer, we obtained the following joint angles with respect to time.
% t1_new will be used for plotting
t1_new = 2*atan(tan((pi*t)/360)); %From the answer obtained above
```



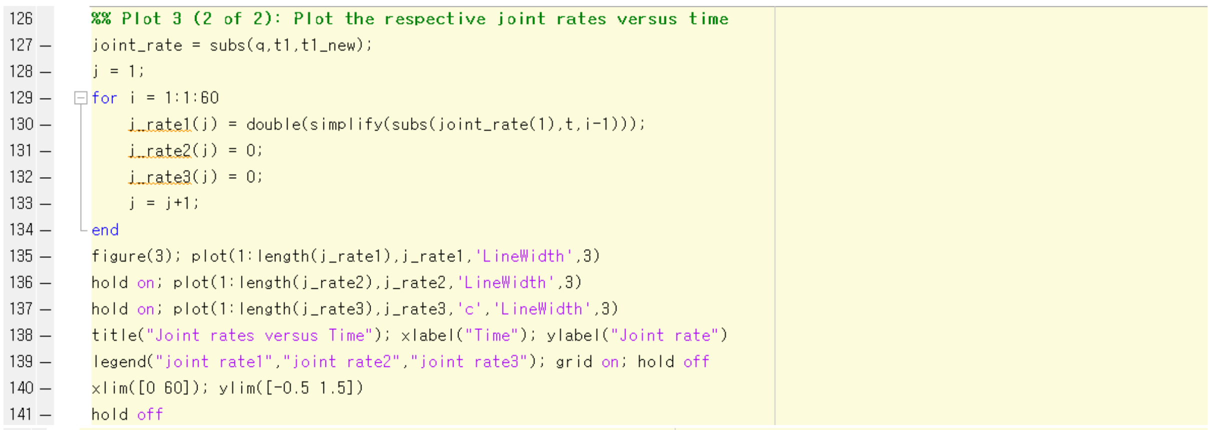
```matlab
%% Plot 1: The path of the tool tip on the work surface (A, B, C and D);
j = 1;
Position = subs(P0_4,[t1,t2,t3],[initial_joint_angles(1)+t1_new,initial_joint_angles(2),initial_joint_angles(3)]);
for i = 1:1:60
    Position_x(j) = double(subs(Position(1),t,i-1));
    Position_y(j) = double(subs(Position(2),t,i-1));
    j = j+1;
end
figure(1); plot(Position_x,Position_y,'LineWidth',2)
xlabel("Displacement in x-axis"); ylabel("Displacement in y-axis")
title("The path of the tool tip on the work surface")
hold on; plot(0,max(Position_y),'r*'); hold on; plot(min(Position_x),0,'k*');
hold on; plot(0,min(Position_y),'b*'); hold on; plot(max(Position_x),0,'g*');
legend("Path of the tool tip","A","B","C","D"); grid on
hold off
```



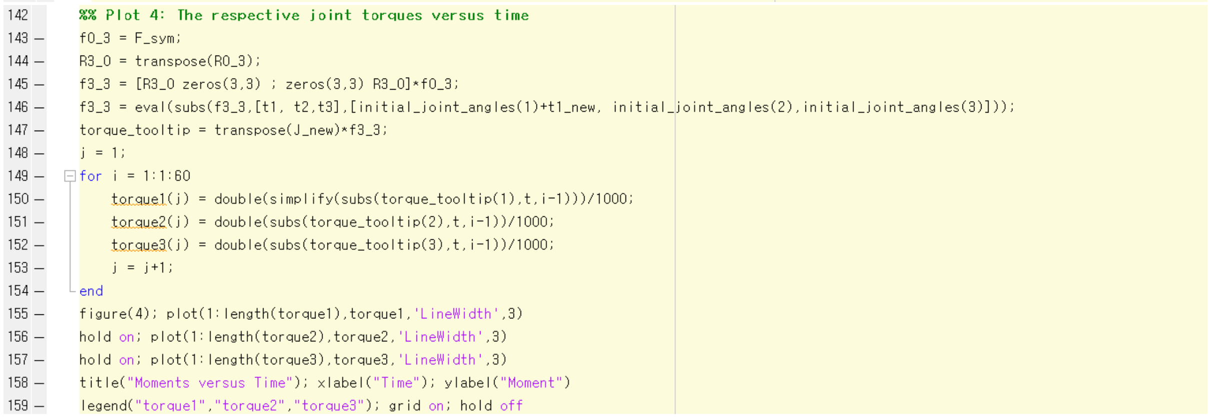
```matlab
%% Plot 2: The respective joint angles versus time
% An assumption: theta 2 and theta 3 remain rigid while in motion
j=1;
for i = 1:1:60
    theta1(j) = double(initial_joint_angles(1)+2*atan(tan((pi*i)/360)));
    theta2(j) = double(initial_joint_angles(2));
    theta3(j) = double(initial_joint_angles(3));
    j = j+1;
end
figure(2); plot(1:length(theta1),theta1,'LineWidth',3)
hold on; plot(1:length(theta2),theta2,'LineWidth',3)
hold on; plot(1:length(theta3),theta3,'LineWidth',3)
title("Joint angles versus Time"); xlabel("Time"); ylabel("Joint angle")
legend("Theta1","Theta2","Theta3"); grid on; hold off
```



```matlab
%% Plot 3 (1 of 2): Calculate the velocity
J_new = simplify(subs(J,[t1,t2,t3],[initial_joint_angles(1)+t1_new,initial_joint_angles(2),initial_joint_angles(3)]));
J_inv = pinv(J_new); % Pseudo-inverse so that we can calculate dq=J^-1*dp
P0_3_change = subs(P0_3,[t1, t2,t3],[initial_joint_angles(1)+t1_new, initial_joint_angles(2),initial_joint_angles(3)]);
Velocity = diff(P0_3_change,t); % Velocity at the frame 3
q = simplify(J_inv*[Velocity;0;0;1]); % Joint rate
```



```
%% Plot 3 (2 of 2): Plot the respective joint rates versus time
joint_rate = subs(q,t1,t1_new);
j = 1;
for i = 1:1:60
    j_rate1(j) = double(simplify(subs(joint_rate(1),t,i-1)));
    j_rate2(j) = 0;
    j_rate3(j) = 0;
    j = j+1;
end
figure(3); plot(1:length(j_rate1),j_rate1,'LineWidth',3)
hold on; plot(1:length(j_rate2),j_rate2,'LineWidth',3)
hold on; plot(1:length(j_rate3),j_rate3,'c','LineWidth',3)
title("Joint rates versus Time"); xlabel("Time"); ylabel("Joint rate")
legend("joint rate1","joint rate2","joint rate3"); grid on; hold off
xlim([0 60]); ylim([-0.5 1.5])
hold off
```



```
%% Plot 4: The respective joint torques versus time
f0_3 = F_sym;
R3_0 = transpose(R0_3);
f3_3 = [R3_0 zeros(3,3) ; zeros(3,3) R3_0]*f0_3;
f3_3 = eval(subs(f3_3,[t1, t2,t3],[initial_joint_angles(1)+t1_new, initial_joint_angles(2),initial_joint_angles(3)]));
torque_tooltip = transpose(J_new)*f3_3;
j = 1;
for i = 1:1:60
    torque1(j) = double(simplify(subs(torque_tooltip(1),t,i-1)))/1000;
    torque2(j) = double(subs(torque_tooltip(2),t,i-1))/1000;
    torque3(j) = double(subs(torque_tooltip(3),t,i-1))/1000;
    j = j+1;
end
figure(4); plot(1:length(torque1),torque1,'LineWidth',3)
hold on; plot(1:length(torque2),torque2,'LineWidth',3)
hold on; plot(1:length(torque3),torque3,'LineWidth',3)
title("Moments versus Time"); xlabel("Time"); ylabel("Moment")
legend("torque1","torque2","torque3"); grid on; hold off
```