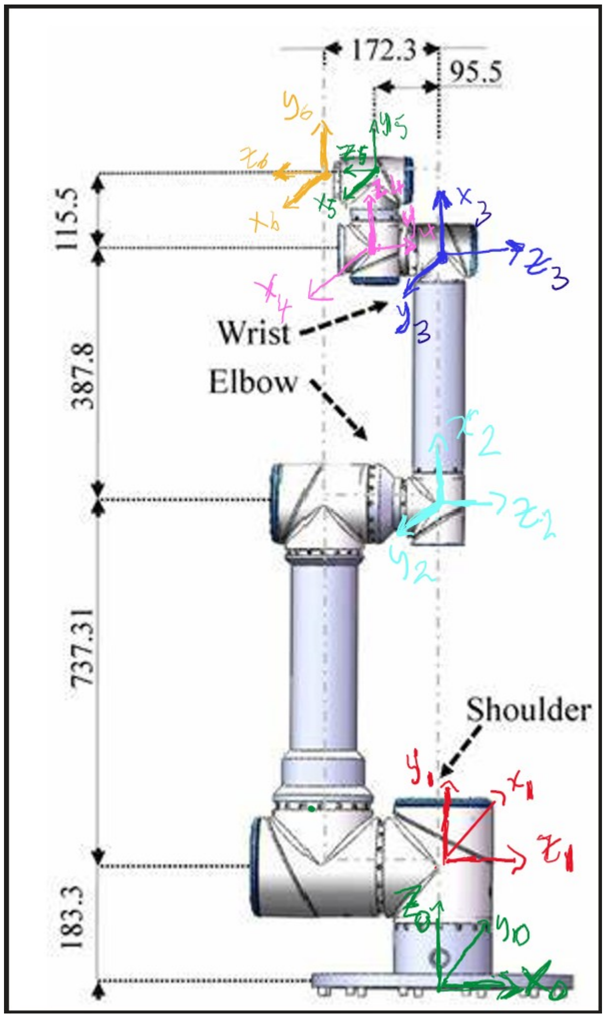
Kent-Diens Joseph
ENPM 662 HW2

PROBLEM 1

The DH frames were set as shown in the picture below:



Note that the frame 1 and frame 2 where shifted along the link so that the x-axis can be perpendicular to the current and previous z-axis frames, and also intersect them. Also note that when the diagonal axis point to the upper right, then it means the axis is going inside the plane, and when the diagonal axis is pointing to the lower left, then it is going outside of the plane.

The DH table was set-up as shown below. Note that the UR3 image in problem did not include a dimension for the small space in front of the end-effector and the dash-lines in front of it. As a result, the end-effector was assumed to be right at the dash-line, and d was determined by subtracting 95.5 mm from 172.3 mm. Also note that all the dimensions given in the picture were converted to meters.

	Theta[rad]	**alpha[rad]**	**a[m]**	**d[m]**
0→1	Theta1 + pi/2	Pi/2	0	0.1833
1 → 2	Theta2 + pi/2	0	0.73731	0
2 → 3	theta3	0	0.3878	0
3 → 4	Theta4 + pi/2	Pi/2	0	-0.0955
4 → 5	theta5	Pi/2	0	0.1155
5 → 6	theta6	0	0	0.0768

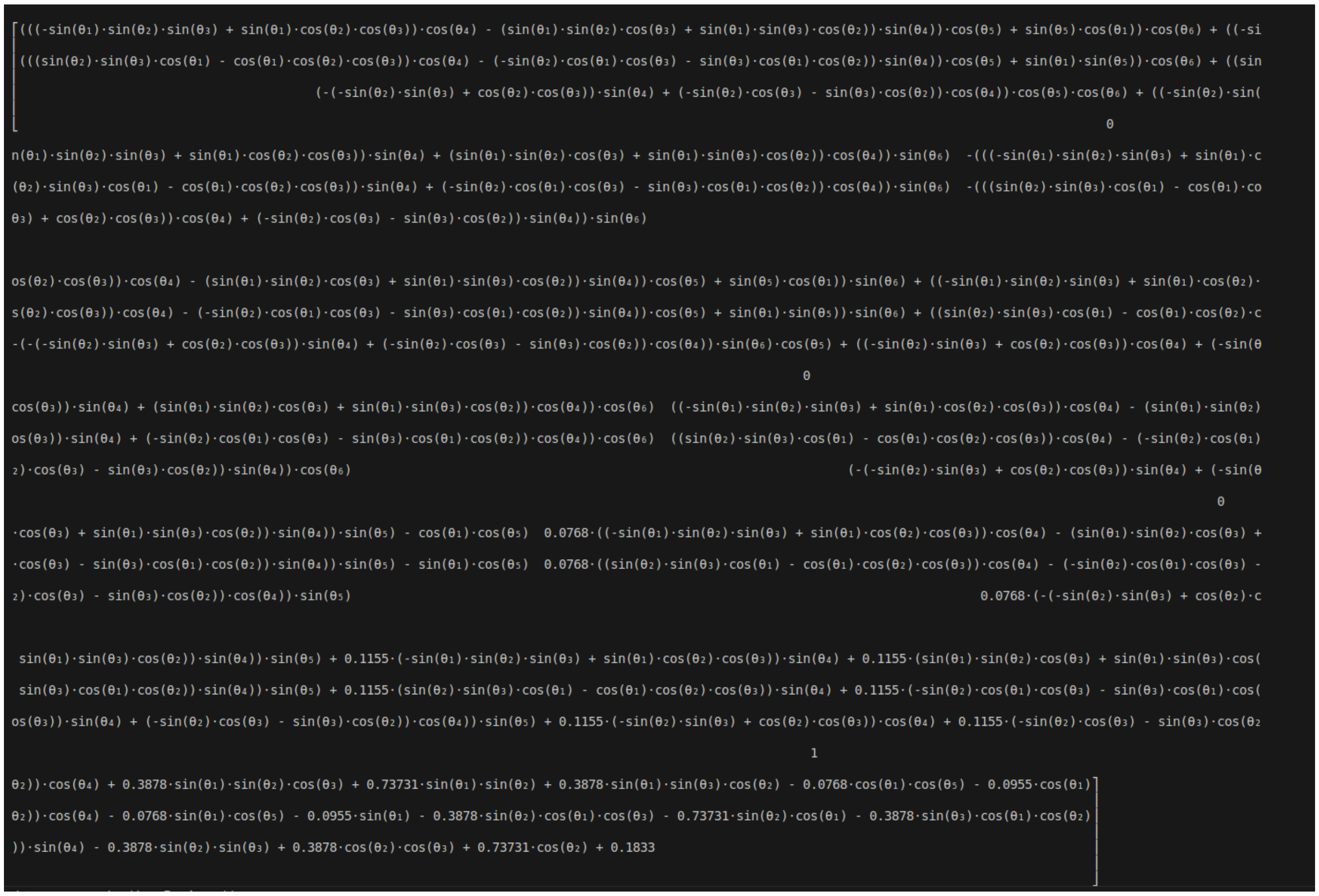
The following equation from the Textbook by Spong was used to find the successive link transformation matrices, with the data in each row of the DH table.

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

The Homogeneous Transformation matrix of the end-effector with respect to the base frame (transM_6_0) was found by multiplying all those link matrices. This was applied in the code as shown below:

```
transM_1_0 = calc_ht_matrix(dh_table[0])
transM_2_1 = calc_ht_matrix(dh_table[1])
transM_3_2 = calc_ht_matrix(dh_table[2])
transM_4_3 = calc_ht_matrix(dh_table[3])
transM_5_4 = calc_ht_matrix(dh_table[4])
transM_6_5 = calc_ht_matrix(dh_table[5])
transM_6_0 = transM_1_0 * transM_2_1 * transM_3_2 * transM_4_3 * transM_5_4 * transM_6_5
```

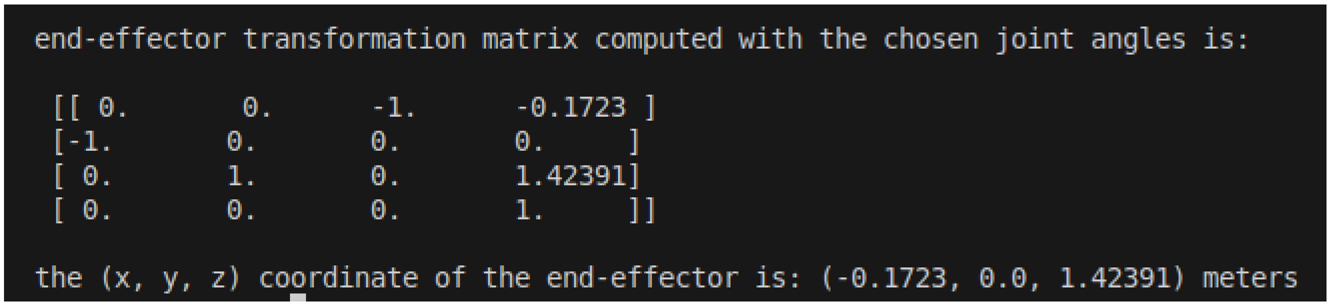
where “calc_ht_matrix()” is a function that takes the link info from the DH table and computes the Spong equation with them (see code for more info). Below is the end-effector homogeneous transformation matrix with respect to the base frame (taken from the terminal):



```
⎡(((-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (sin(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄))·cos(θ₅) + sin(θ₅)·cos(θ₁))·cos(θ₆) + ((-si
⎢(((sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄))·cos(θ₅) + sin(θ₁)·sin(θ₅))·cos(θ₆) + ((sin
⎢                                      (-(-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·sin(θ₄) + (-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·cos(θ₄))·cos(θ₅)·cos(θ₆) + ((-sin(θ₂)·sin(
⎣                                                                                                                                                         0

n(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + (sin(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄))·sin(θ₆)  -(((-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·c
(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + (-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄))·sin(θ₆)  -(((sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·co
θ₃) + cos(θ₂)·cos(θ₃))·cos(θ₄) + (-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·sin(θ₄))·sin(θ₆)


os(θ₂)·cos(θ₃))·cos(θ₄) - (sin(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄))·cos(θ₅) + sin(θ₅)·cos(θ₁))·sin(θ₆) + ((-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·
s(θ₂)·cos(θ₃))·cos(θ₄) - (-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄))·cos(θ₅) + sin(θ₁)·sin(θ₅))·sin(θ₆) + ((sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·c
-(-(-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·sin(θ₄) + (-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·cos(θ₄))·sin(θ₆)·cos(θ₅) + ((-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·cos(θ₄) + (-sin(θ
                                                                                                                       0
cos(θ₃))·sin(θ₄) + (sin(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄))·cos(θ₆)  ((-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (sin(θ₁)·sin(θ₂)
os(θ₃))·sin(θ₄) + (-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄))·cos(θ₆)  ((sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (-sin(θ₂)·cos(θ₁)
₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·sin(θ₄))·cos(θ₆)                                      (-(-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·sin(θ₄) + (-sin(θ
                                                                                                                                                           0
·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄))·sin(θ₅) - cos(θ₁)·cos(θ₅)  0.0768·((-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (sin(θ₁)·sin(θ₂)·cos(θ₃) +
·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄))·sin(θ₅) - sin(θ₁)·cos(θ₅)  0.0768·((sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (-sin(θ₂)·cos(θ₁)·cos(θ₃) -
₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·cos(θ₄))·sin(θ₅)                                                            0.0768·(-(-sin(θ₂)·sin(θ₃) + cos(θ₂)·c


 sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄))·sin(θ₅) + 0.1155·(-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 0.1155·(sin(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(
 sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄))·sin(θ₅) + 0.1155·(sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 0.1155·(-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(
os(θ₃))·sin(θ₄) + (-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·cos(θ₄))·sin(θ₅) + 0.1155·(-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·cos(θ₄) + 0.1155·(-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂
                                                                                                                        1
θ₂))·cos(θ₄) + 0.3878·sin(θ₁)·sin(θ₂)·cos(θ₃) + 0.73731·sin(θ₁)·sin(θ₂) + 0.3878·sin(θ₁)·sin(θ₃)·cos(θ₂) - 0.0768·cos(θ₁)·cos(θ₅) - 0.0955·cos(θ₁)⎤
θ₂))·cos(θ₄) - 0.0768·sin(θ₁)·cos(θ₅) - 0.0955·sin(θ₁) - 0.3878·sin(θ₂)·cos(θ₁)·cos(θ₃) - 0.73731·sin(θ₂)·cos(θ₁) - 0.3878·sin(θ₃)·cos(θ₁)·cos(θ₂)⎥
))·sin(θ₄) - 0.3878·sin(θ₂)·sin(θ₃) + 0.3878·cos(θ₂)·cos(θ₃) + 0.73731·cos(θ₂) + 0.1833                                                            ⎥
                                                                                                                                                   ⎦
```

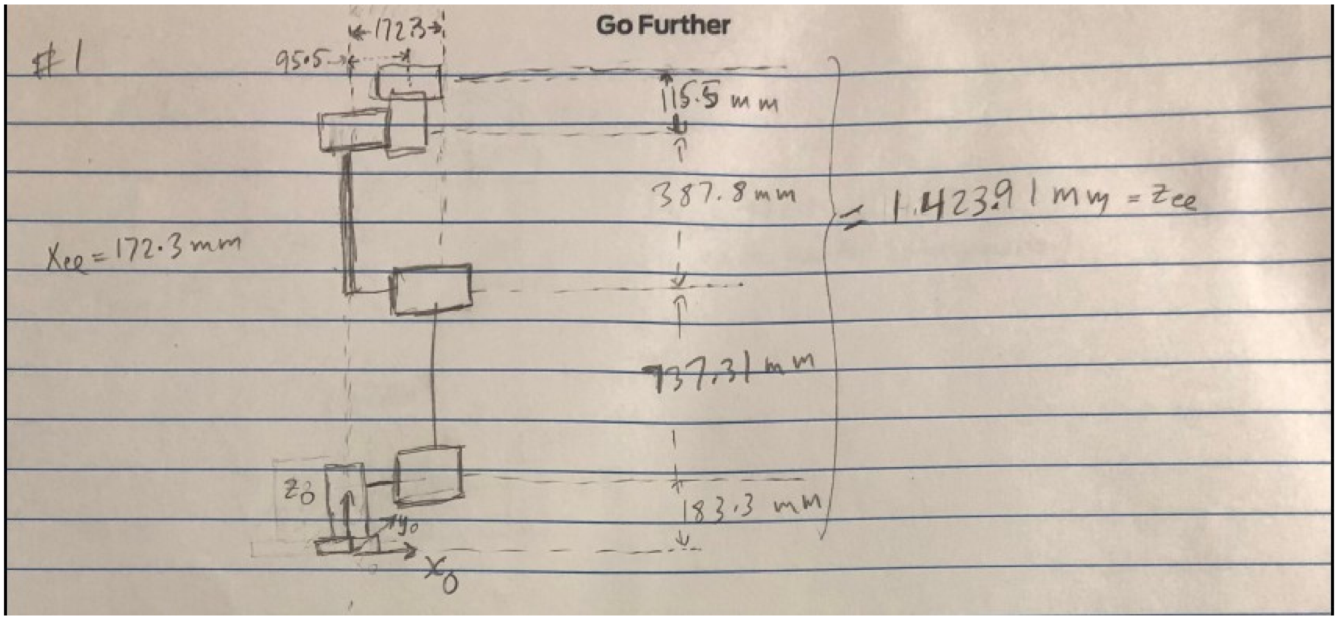
Forward Position Kinematics Validation
To validate the forward position kinematics equations obtained in the previous part, different values for the joint angles were chosen, then the end-effector's coordinates were found geometrically using the dimension provided, then found by computing the end-effector's transformation using those joint angles into the transformation matrix using the code, and some of them were also computed using the Matlab Robotics Toolbox by Prof. Peter Corke. The end-effector coordinates of in the transformation matrix are given by the top 3 elements in the last column of the matrix. The coordinates from the three approaches were then compared and were found to produce the same result.

- Initial position
This is the same position as the robot's home position with theta1 = theta2= theta3 = theta4 = theta5= theta6 = 0 . Note that theta1 is the angle for the revolute joint at the base, theta2 is for the next joint and associated with DH frame x1,y1,z1, thetat3 is associated with DH frame x2,y2,z2, and so on. The coordinates of the end-effector with respect to the base frame found with the geometrical method are: (-0.1723, 0.0, 1.42391) meters. This matches the coordinates found using the transformation matrix and the code (image below is from the terminal output):

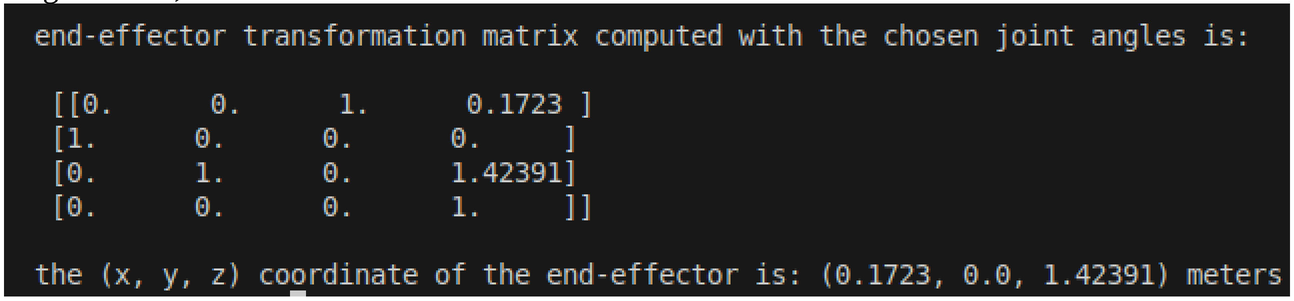


```
end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.        0.       -1.       -0.1723 ]
 [-1.        0.        0.        0.      ]
 [ 0.        1.        0.        1.42391]
 [ 0.        0.        0.        1.      ]]

the (x, y, z) coordinate of the end-effector is: (-0.1723, 0.0, 1.42391) meters
```

- 1st Joint Angle Set (that are different than the home position)
theta1=180, theta2=0,theta3= 0, theta4=0, theta5= 0, theta6=0
The image below (with dimensions in mm) shows the robot with the revolute joint at the base with 180 degrees. Using the dimensions that were provided, the end-effector coordinates converted to meters are: (0.1723, 0.0, 1.42391)

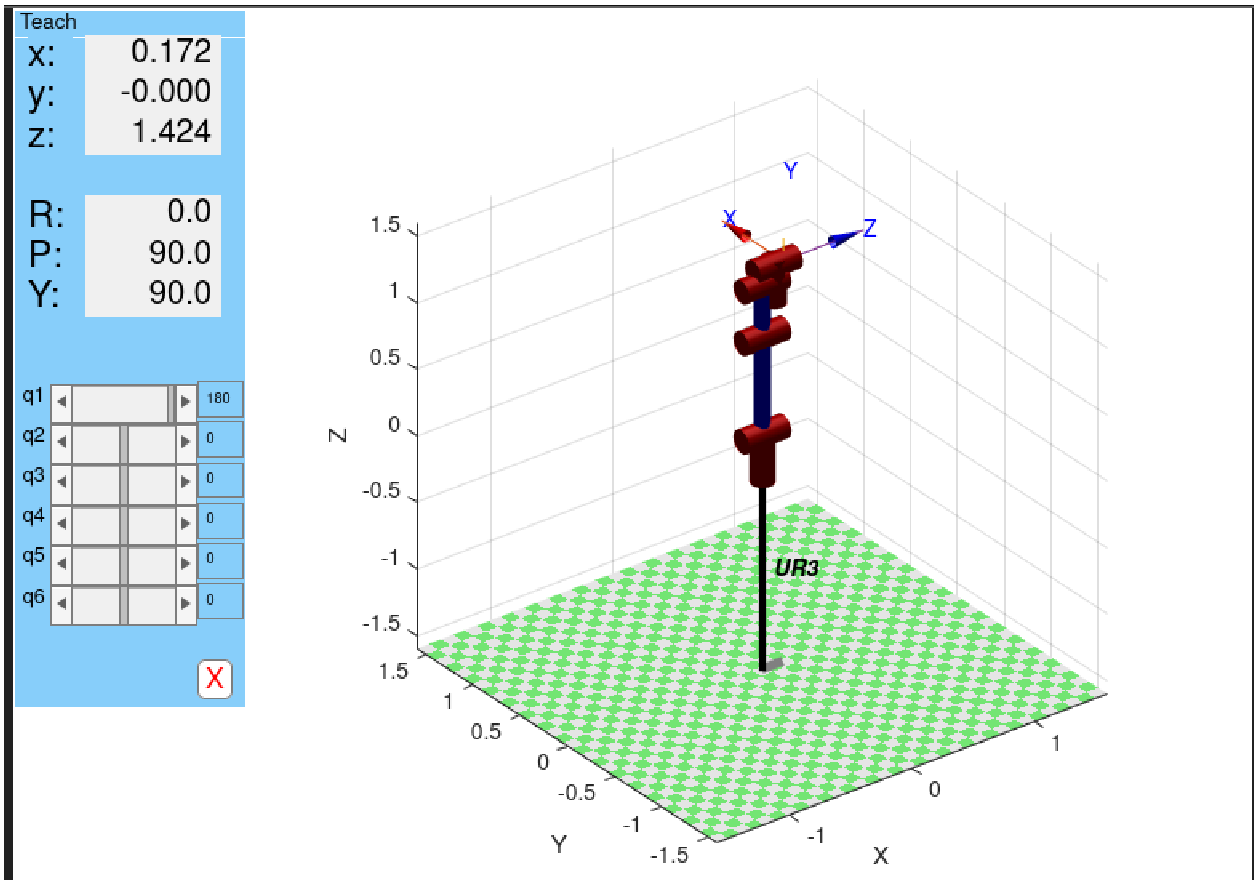


Using the code, the end-effector coordinates are the same.



```
end-effector transformation matrix computed with the chosen joint angles is:

 [[0.       0.       1.       0.1723 ]
 [1.       0.       0.       0.      ]
 [0.       1.       0.       1.42391]
 [0.       0.       0.       1.     ]]

the (x, y, z) coordinate of the end-effector is: (0.1723, 0.0, 1.42391) meters
```

Using the Peter Corke Matlab Robotics Toolbox, the same result was found as can be seen in the image below.

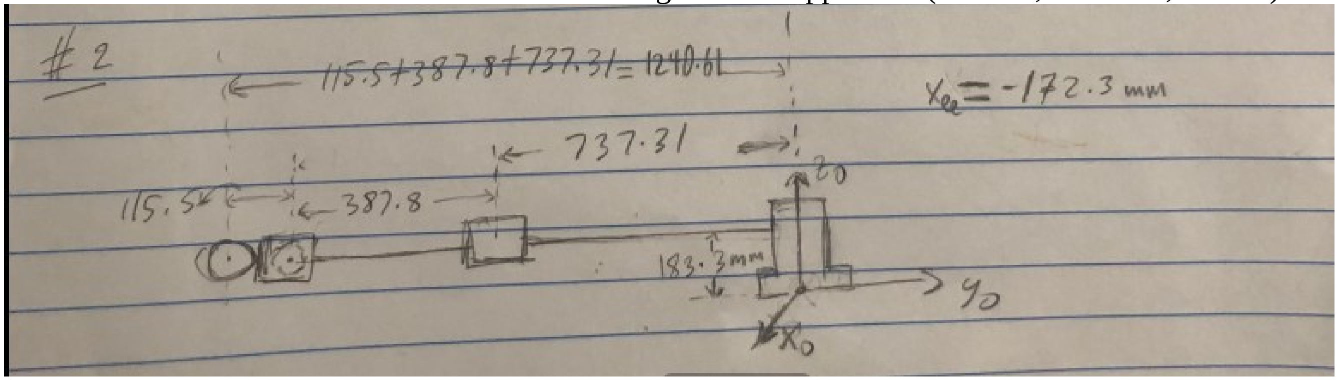

Teach

x: 0.172
y: -0.000
z: 1.424

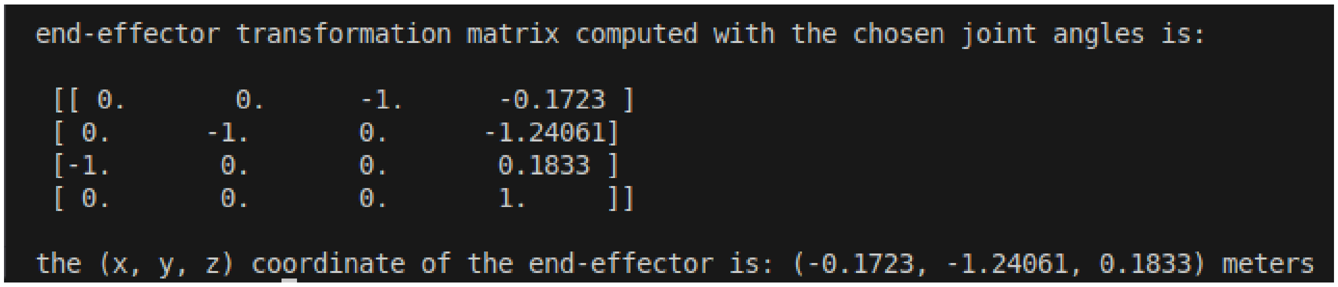
R: 0.0
P: 90.0
Y: 90.0

q1 180
q2 0
q3 0
q4 0
q5 0
q6 0

- Second Joint Angle Set
theta1= 0, theta2=90, theta3=0, theta4 = 0, theta5 = 0, theta6 = 0
With those angles, the robot moved in the y-z plane, so the sketch shows a view of the robot from this plane. As a result, the x-coordinate is the same as that of the initial home position. The end-effector coordinates in meters w.r.t the base frame with the geometric approach: (-0.1723, -1.24061, 0.1833)

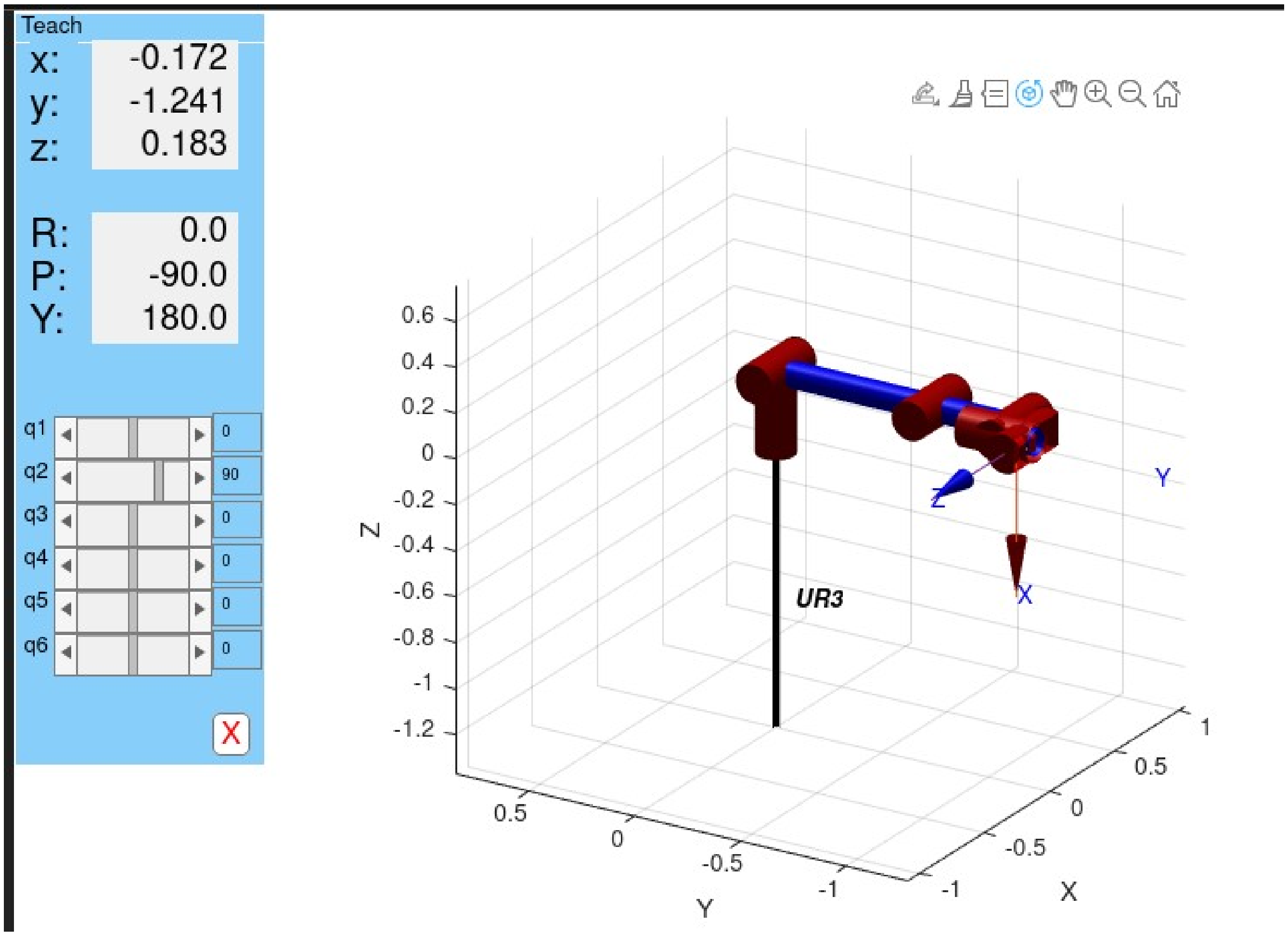


Using the code, the end-effector's coordinates were found to be the same:



```
end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.        0.       -1.       -0.1723 ]
 [ 0.       -1.        0.       -1.24061]
 [-1.        0.        0.        0.1833 ]
 [ 0.        0.        0.        1.     ]]

the (x, y, z) coordinate of the end-effector is: (-0.1723, -1.24061, 0.1833) meters
```

The same result was found using the Matlab robotics toolbox.

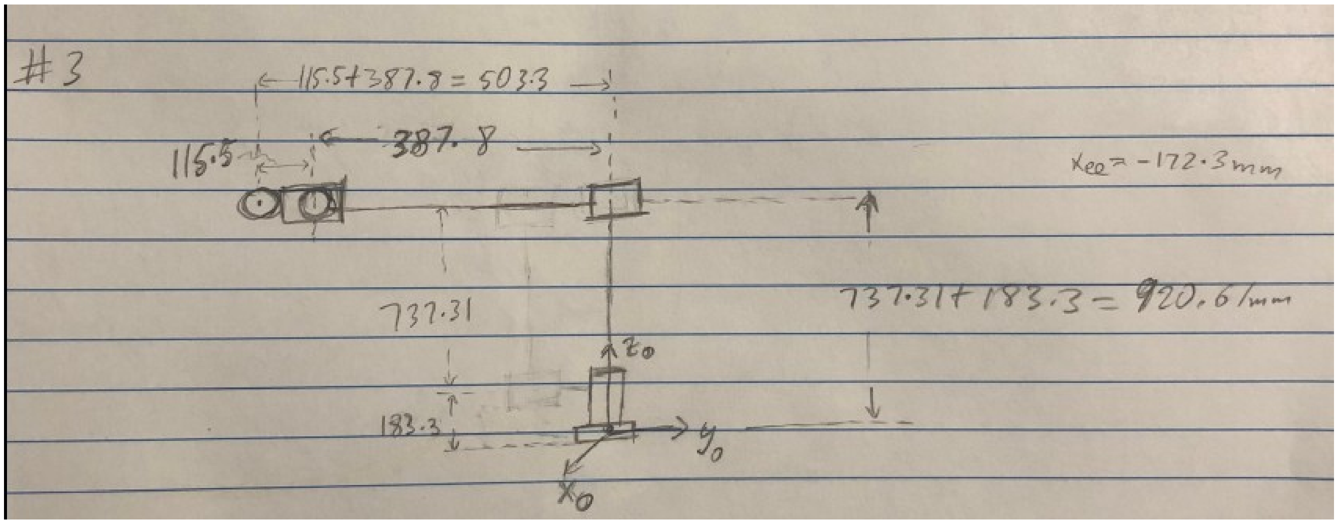


Teach
x: -0.172
y: -1.241
z: 0.183

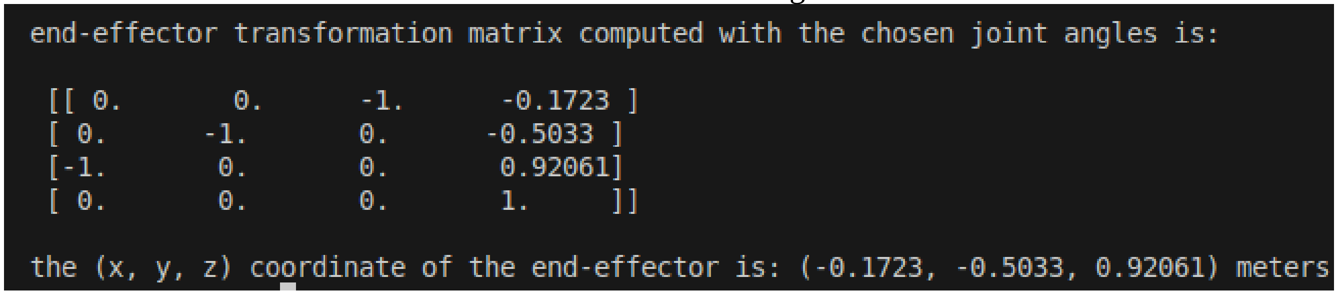
R: 0.0
P: -90.0
Y: 180.0

q1 0
q2 90
q3 0
q4 0
q5 0
q6 0

- 3rd Joint Angle Set
theta1 = 0, theta2 = 0, theta3 = 90, theta4 = 0, theta5 = 0, theta6 = 0
Using the geometric approach the (x, y, z) coordinate of the end-effector in meters is:
(-0.1723, -0.5033, 0.92061)

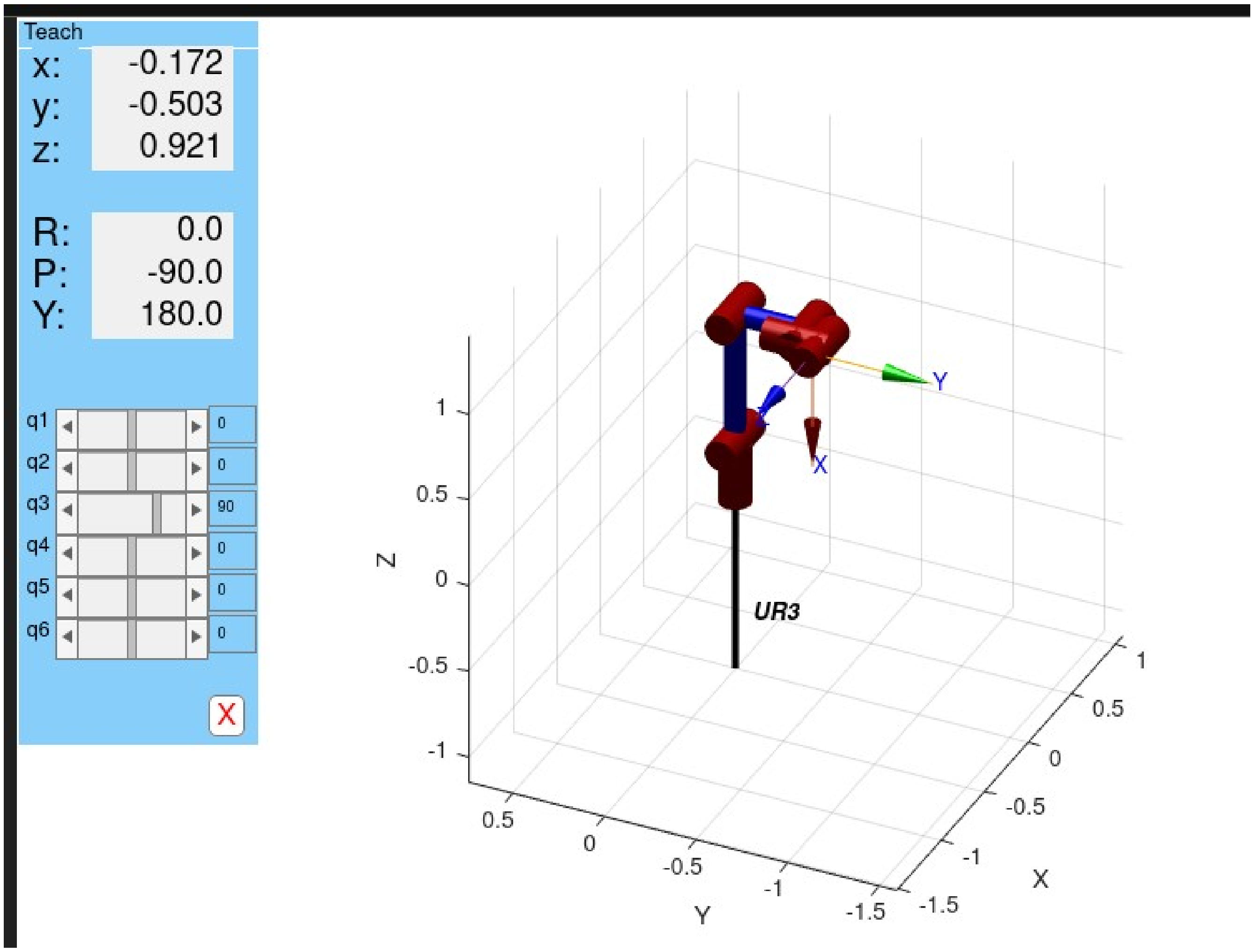


The end-effector's coordinates was found to be the same using the code:

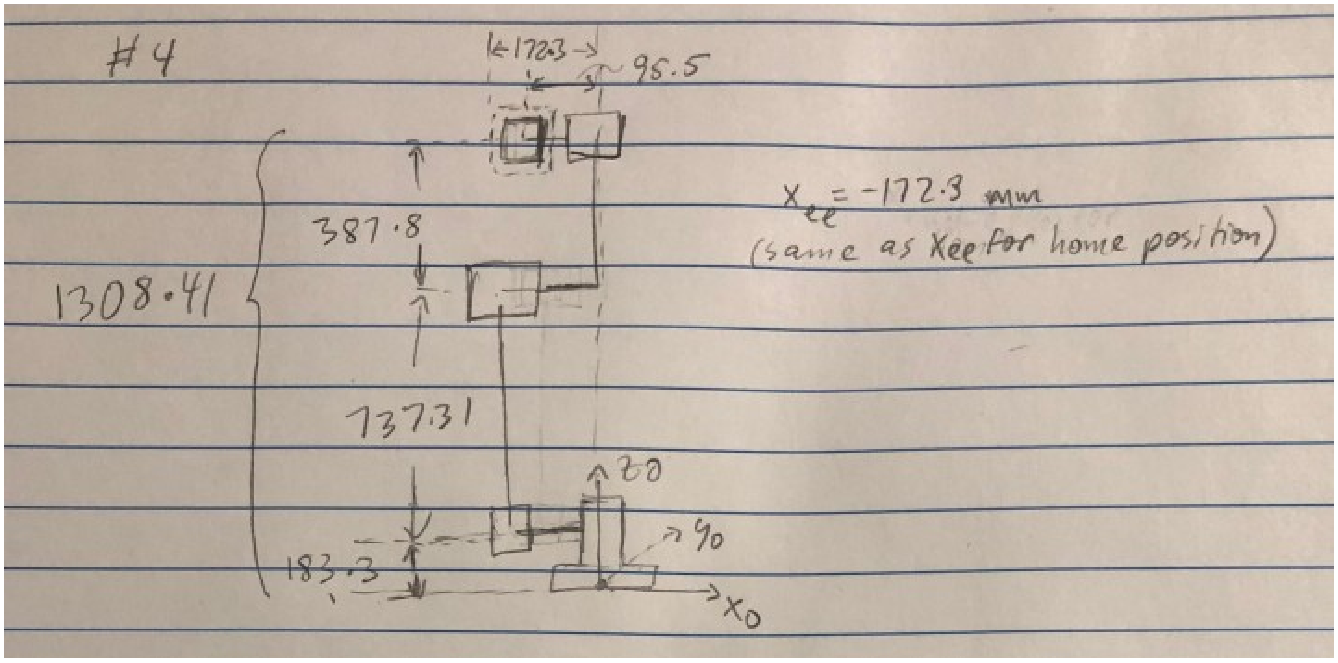


```
end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.       0.      -1.      -0.1723 ]
 [ 0.      -1.       0.      -0.5033 ]
 [-1.       0.       0.       0.92061]
 [ 0.       0.       0.       1.     ]]

the (x, y, z) coordinate of the end-effector is: (-0.1723, -0.5033, 0.92061) meters
```

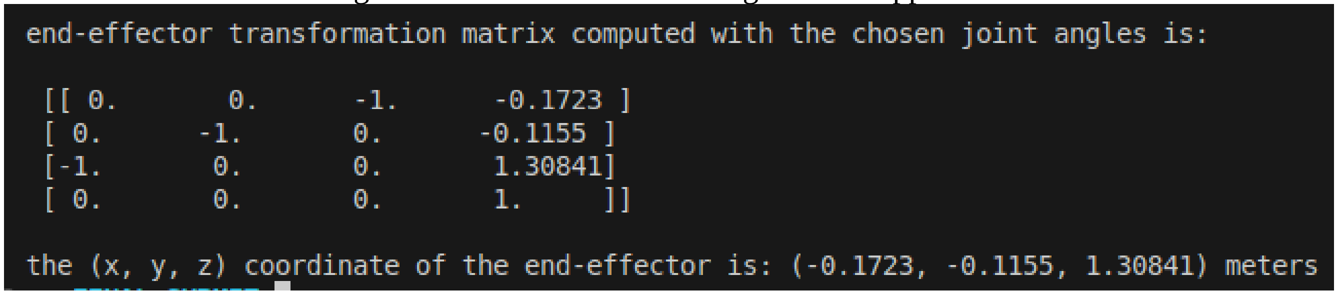
The same result was found using the Matlab robotics toolbox.



- 4th Joint Angle Set
theta1= 0, theta2 = 0, theta3 = 0, theta4 = 90, theta5 = 0, theta6= 0
In this configuration, the end-effector is coming out of the page (in the -y direction). The (x, y, z) coordinate of the end-effector in meters using the geometric approach is is: (-0.1723, -0.1155, 1.30841)

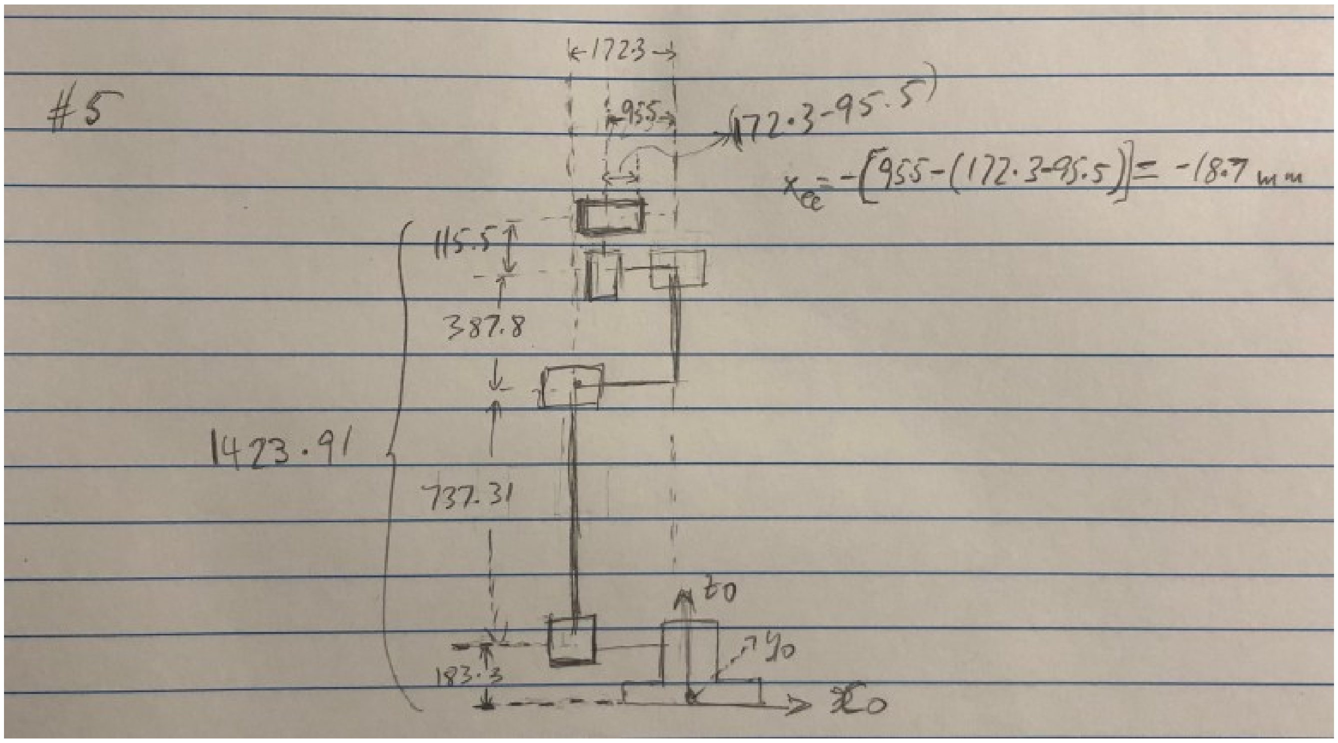


The coordinates found using the code are the same as the geometric approach.

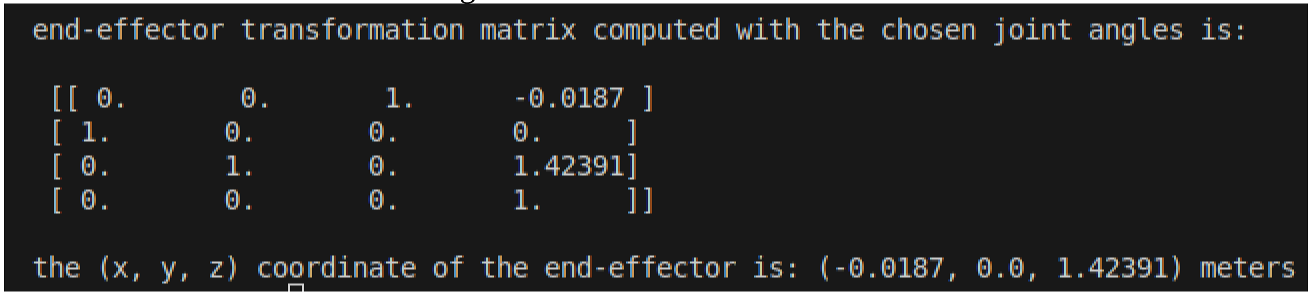


```
end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.       0.      -1.      -0.1723 ]
 [ 0.      -1.       0.      -0.1155 ]
 [-1.       0.       0.       1.30841]
 [ 0.       0.       0.       1.     ]]

the (x, y, z) coordinate of the end-effector is: (-0.1723, -0.1155, 1.30841) meters
```

- 5th Joint Angle Set
theta1= 0, theta2= 0, theta3 = 0, theta4= 0, theta5= 180, theta6 = 0
the (x, y, z) coordinate of the end-effector in meters using the geometric approach is:
(-0.0187, 0.0, 1.42391)



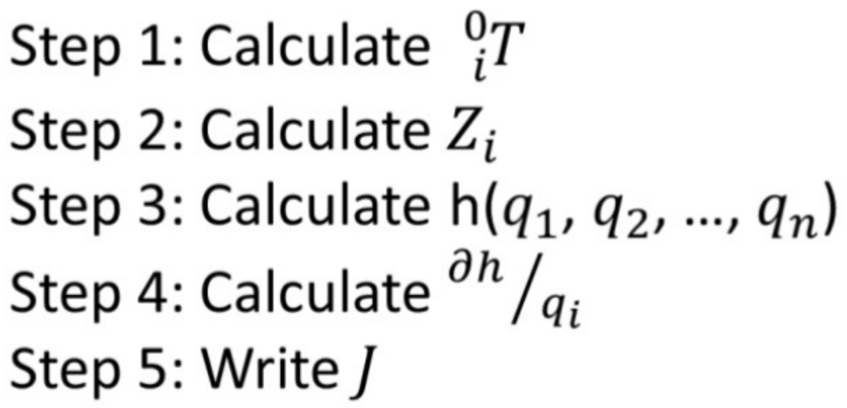
The same coordinate was found using the code:



```
end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.       0.       1.      -0.0187 ]
 [ 1.       0.       0.       0.     ]
 [ 0.       1.       0.       1.42391]
 [ 0.       0.       0.       1.     ]]

the (x, y, z) coordinate of the end-effector is: (-0.0187, 0.0, 1.42391) meters
```

PROBLEM 2

Method 2 from lecture 8 was used to calculate the Jacobian matrix for the UR3 robot.



Step 1: Calculate ${}^{0}_{i}T$

Step 2: Calculate Z_i

Step 3: Calculate h($q_1, q_2, \ldots, q_n$)

Step 4: Calculate ${}^{\partial h}/_{q_i}$

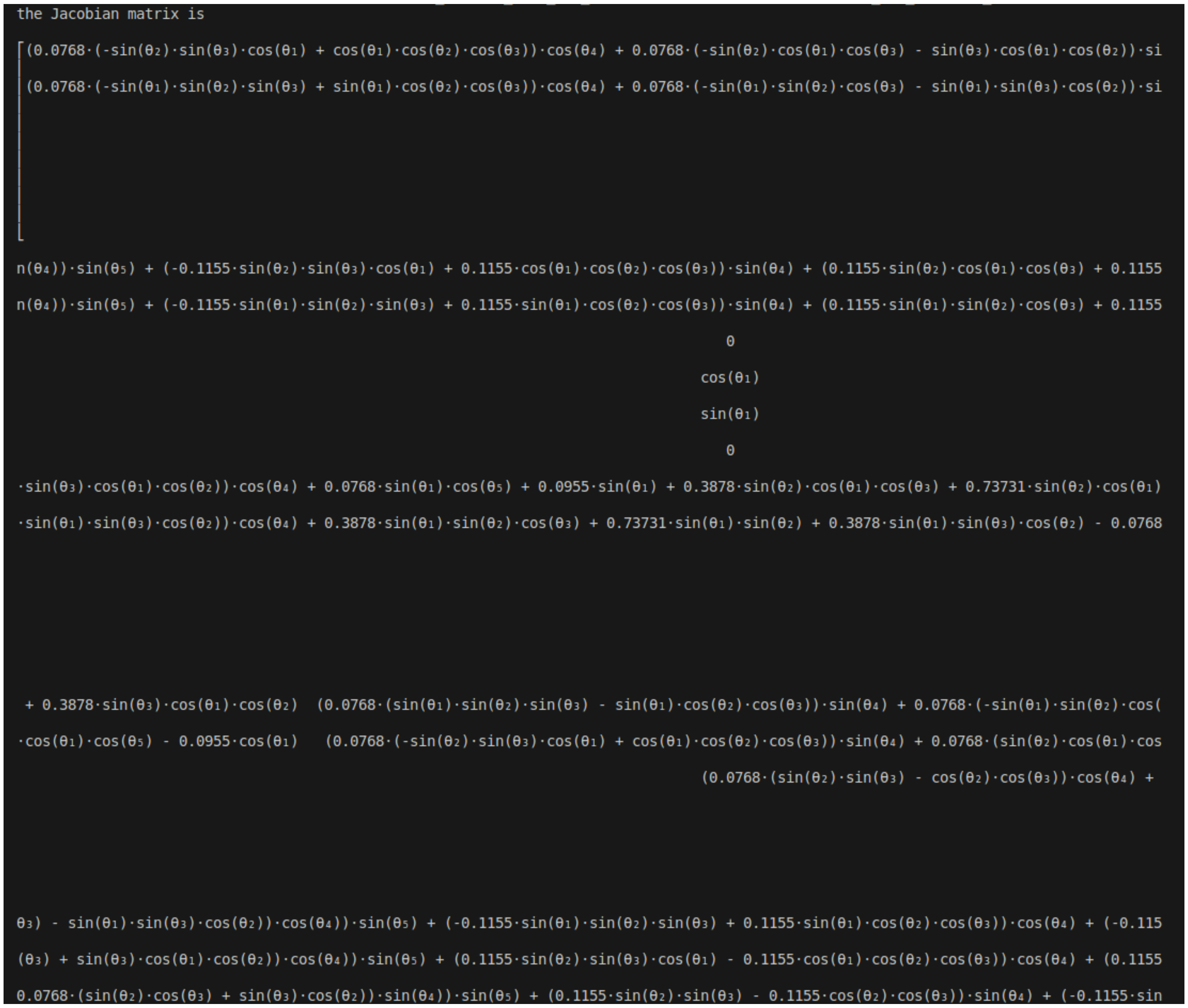
Step 5: Write J

For step1, the same equation for the homogeneous transformation of the link with respect to the previous link, was used to calculated the homogeneous transformation matrix for the first frame with respect to the base frame. (TransM_1_0) The same equation was reused to calculate the homogeneous transform matrix from the second frame with respect to the first frame (TransM_2_1). Then to find the homogeneous transformation matrix of the second frame with respect to the base frame, TransM_0 was multiplied by TransM_2_1. The same steps were repeated for the subsequent frames, including the frame at the end-effector.

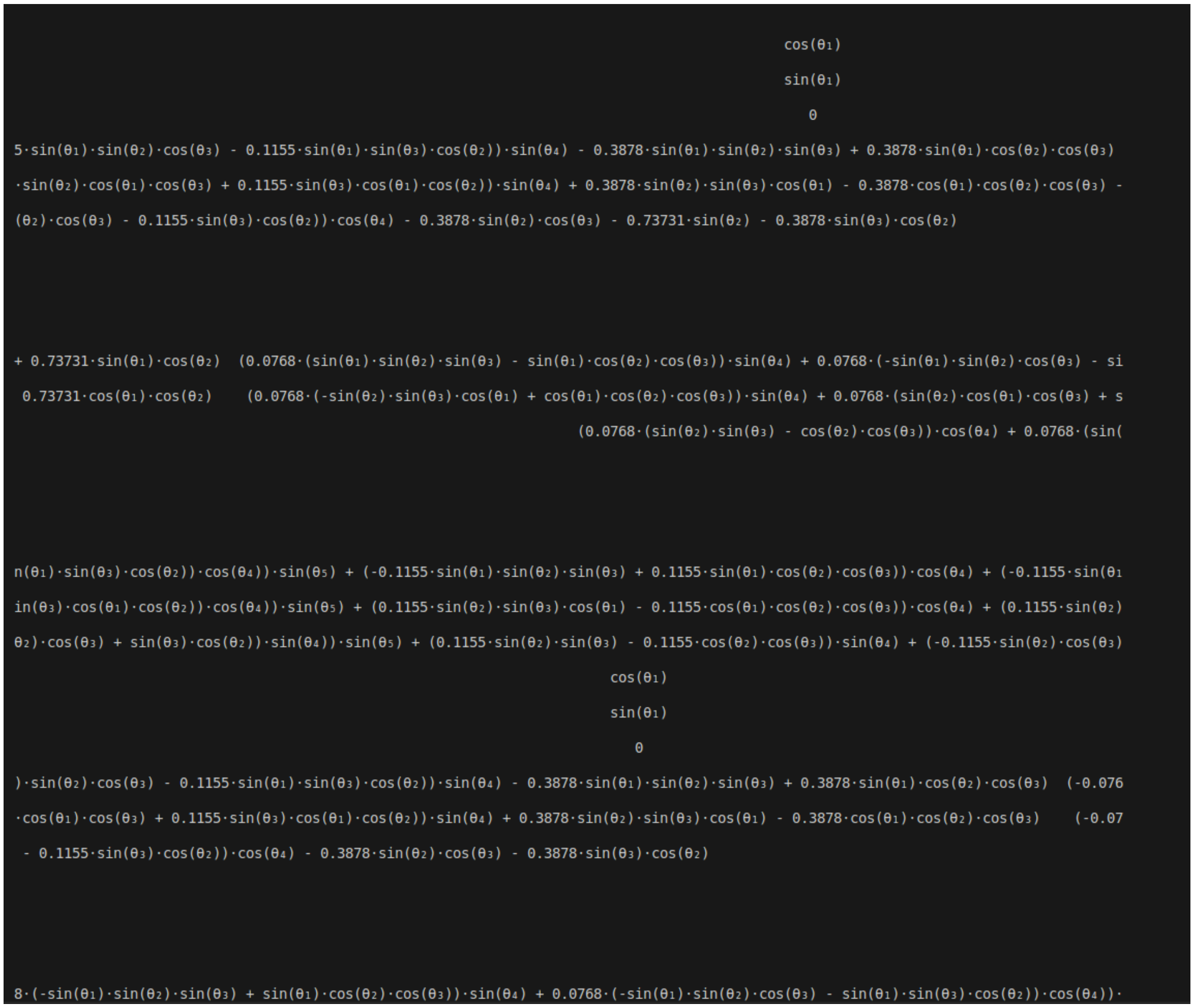
TransM_1_0 = found using the DH with homogeneous transformation matrix equation
TransM_2_0 = TransM_1_0 * TransM_2_1
TransM_3_0 = TransM_2_0 * TransM_3_2
TransM_4_0 = TransM_3_0 * TransM_4_3
TransM_5_0 = TransM_4_0 * TransM_5_4
TransM_6_0 = TransM_5_0 * TransM_6_5

(Note, to translate the above notation used to the one used by the book, the first number after the term "TransM" would be upper-script elements, and the second element would be the subscript element)

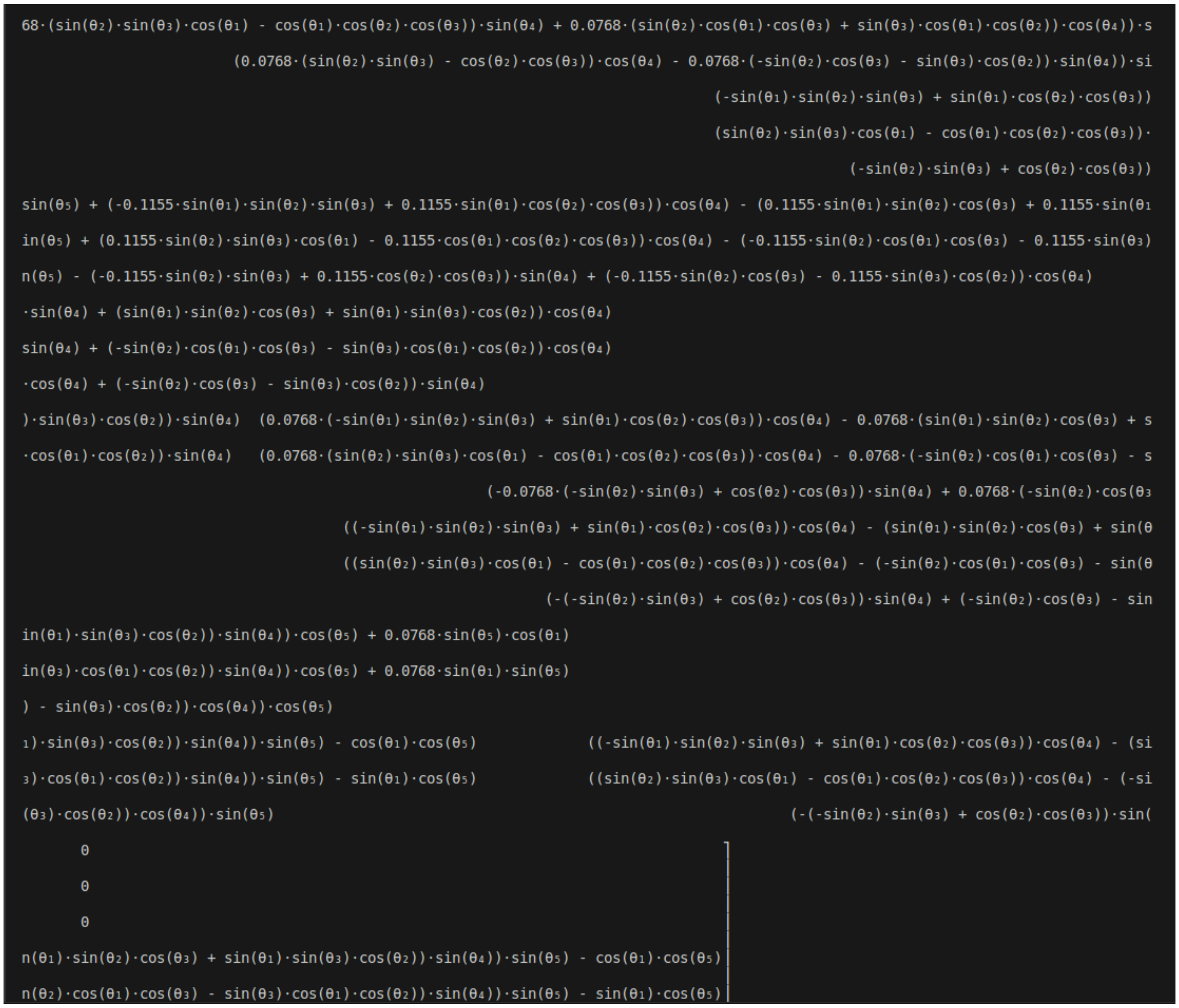
TransM_6_0 is the end-effectors homogeneous transformation matrix. The partial derivative of the elements in the last column and first 3 rows of this matrix were taken with respect to each of the joint angle theta1 through theta6. The results were combined to form the top 3x6 portion of the final Jacobian matrix (Steps 3 & 4). Then the z components were taken from each of the homogeneous transform matrices with respect to the base frame (TransM_1_0 to TransM_6_0), which is the elements in the third column and first three rows of the homogeneous matrices, to form the bottom 3x6 portion of the Jacobian matrix (Step 2). The final 6x6 Jacobian matrix was found by stacking the top and bottom 3x6 matrices together. (see comments in the code for more information). The following 3 pages show the full Jacobian matrix taken from the Terminal.



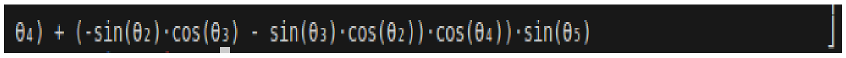
```
the Jacobian matrix is

⎡(0.0768⋅(-sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) + cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) + 0.0768⋅(-sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) - sin(θ₃)⋅cos(θ₁)⋅cos(θ₂))⋅si
⎢(0.0768⋅(-sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) + sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) + 0.0768⋅(-sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) - sin(θ₁)⋅sin(θ₃)⋅cos(θ₂))⋅si
⎢
⎢
⎢
⎢
⎢
⎣

n(θ₄))⋅sin(θ₅) + (-0.1155⋅sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) + 0.1155⋅cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) + (0.1155⋅sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) + 0.1155
n(θ₄))⋅sin(θ₅) + (-0.1155⋅sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) + 0.1155⋅sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) + (0.1155⋅sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) + 0.1155
                                                                                   0
                                                                                cos(θ₁)
                                                                                sin(θ₁)
                                                                                   0
⋅sin(θ₃)⋅cos(θ₁)⋅cos(θ₂))⋅cos(θ₄) + 0.0768⋅sin(θ₁)⋅cos(θ₅) + 0.0955⋅sin(θ₁) + 0.3878⋅sin(θ₂)⋅cos(θ₁)⋅cos(θ₃) + 0.73731⋅sin(θ₂)⋅cos(θ₁)
⋅sin(θ₁)⋅sin(θ₃)⋅cos(θ₂))⋅cos(θ₄) + 0.3878⋅sin(θ₁)⋅sin(θ₂)⋅cos(θ₃) + 0.73731⋅sin(θ₁)⋅sin(θ₂) + 0.3878⋅sin(θ₁)⋅sin(θ₃)⋅cos(θ₂) - 0.0768





 + 0.3878⋅sin(θ₃)⋅cos(θ₁)⋅cos(θ₂)  (0.0768⋅(sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) - sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) + 0.0768⋅(-sin(θ₁)⋅sin(θ₂)⋅cos(
⋅cos(θ₁)⋅cos(θ₅) - 0.0955⋅cos(θ₁)   (0.0768⋅(-sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) + cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) + 0.0768⋅(sin(θ₂)⋅cos(θ₁)⋅cos
                                                                                (0.0768⋅(sin(θ₂)⋅sin(θ₃) - cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) +





θ₃) - sin(θ₁)⋅sin(θ₃)⋅cos(θ₂))⋅cos(θ₄))⋅sin(θ₅) + (-0.1155⋅sin(θ₁)⋅sin(θ₂)⋅sin(θ₃) + 0.1155⋅sin(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) + (-0.115
(θ₃) + sin(θ₃)⋅cos(θ₁)⋅cos(θ₂))⋅cos(θ₄))⋅sin(θ₅) + (0.1155⋅sin(θ₂)⋅sin(θ₃)⋅cos(θ₁) - 0.1155⋅cos(θ₁)⋅cos(θ₂)⋅cos(θ₃))⋅cos(θ₄) + (0.1155
0.0768⋅(sin(θ₂)⋅cos(θ₃) + sin(θ₃)⋅cos(θ₂))⋅sin(θ₄))⋅sin(θ₅) + (0.1155⋅sin(θ₂)⋅sin(θ₃) - 0.1155⋅cos(θ₂)⋅cos(θ₃))⋅sin(θ₄) + (-0.1155⋅sin
```



```
                                                                                         cos(θ₁)

                                                                                         sin(θ₁)

                                                                                            0

5·sin(θ₁)·sin(θ₂)·cos(θ₃) - 0.1155·sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄) - 0.3878·sin(θ₁)·sin(θ₂)·sin(θ₃) + 0.3878·sin(θ₁)·cos(θ₂)·cos(θ₃)

·sin(θ₂)·cos(θ₁)·cos(θ₃) + 0.1155·sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄) + 0.3878·sin(θ₂)·sin(θ₃)·cos(θ₁) - 0.3878·cos(θ₁)·cos(θ₂)·cos(θ₃) -

(θ₂)·cos(θ₃) - 0.1155·sin(θ₃)·cos(θ₂))·cos(θ₄) - 0.3878·sin(θ₂)·cos(θ₃) - 0.73731·sin(θ₂) - 0.3878·sin(θ₃)·cos(θ₂)




+ 0.73731·sin(θ₁)·cos(θ₂)  (0.0768·(sin(θ₁)·sin(θ₂)·sin(θ₃) - sin(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 0.0768·(-sin(θ₁)·sin(θ₂)·cos(θ₃) - si

 0.73731·cos(θ₁)·cos(θ₂)    (0.0768·(-sin(θ₂)·sin(θ₃)·cos(θ₁) + cos(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 0.0768·(sin(θ₂)·cos(θ₁)·cos(θ₃) + s

                                                                   (0.0768·(sin(θ₂)·sin(θ₃) - cos(θ₂)·cos(θ₃))·cos(θ₄) + 0.0768·(sin(




n(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄))·sin(θ₅) + (-0.1155·sin(θ₁)·sin(θ₂)·sin(θ₃) + 0.1155·sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) + (-0.1155·sin(θ₁

in(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄))·sin(θ₅) + (0.1155·sin(θ₂)·sin(θ₃)·cos(θ₁) - 0.1155·cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) + (0.1155·sin(θ₂)

θ₂)·cos(θ₃) + sin(θ₃)·cos(θ₂))·sin(θ₄))·sin(θ₅) + (0.1155·sin(θ₂)·sin(θ₃) - 0.1155·cos(θ₂)·cos(θ₃))·sin(θ₄) + (-0.1155·sin(θ₂)·cos(θ₃)

                                                                       cos(θ₁)

                                                                       sin(θ₁)

                                                                          0

)·sin(θ₂)·cos(θ₃) - 0.1155·sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄) - 0.3878·sin(θ₁)·sin(θ₂)·sin(θ₃) + 0.3878·sin(θ₁)·cos(θ₂)·cos(θ₃)  (-0.076

·cos(θ₁)·cos(θ₃) + 0.1155·sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄) + 0.3878·sin(θ₂)·sin(θ₃)·cos(θ₁) - 0.3878·cos(θ₁)·cos(θ₂)·cos(θ₃)    (-0.07

 - 0.1155·sin(θ₃)·cos(θ₂))·cos(θ₄) - 0.3878·sin(θ₂)·cos(θ₃) - 0.3878·sin(θ₃)·cos(θ₂)




8·(-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 0.0768·(-sin(θ₁)·sin(θ₂)·cos(θ₃) - sin(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄))·
```



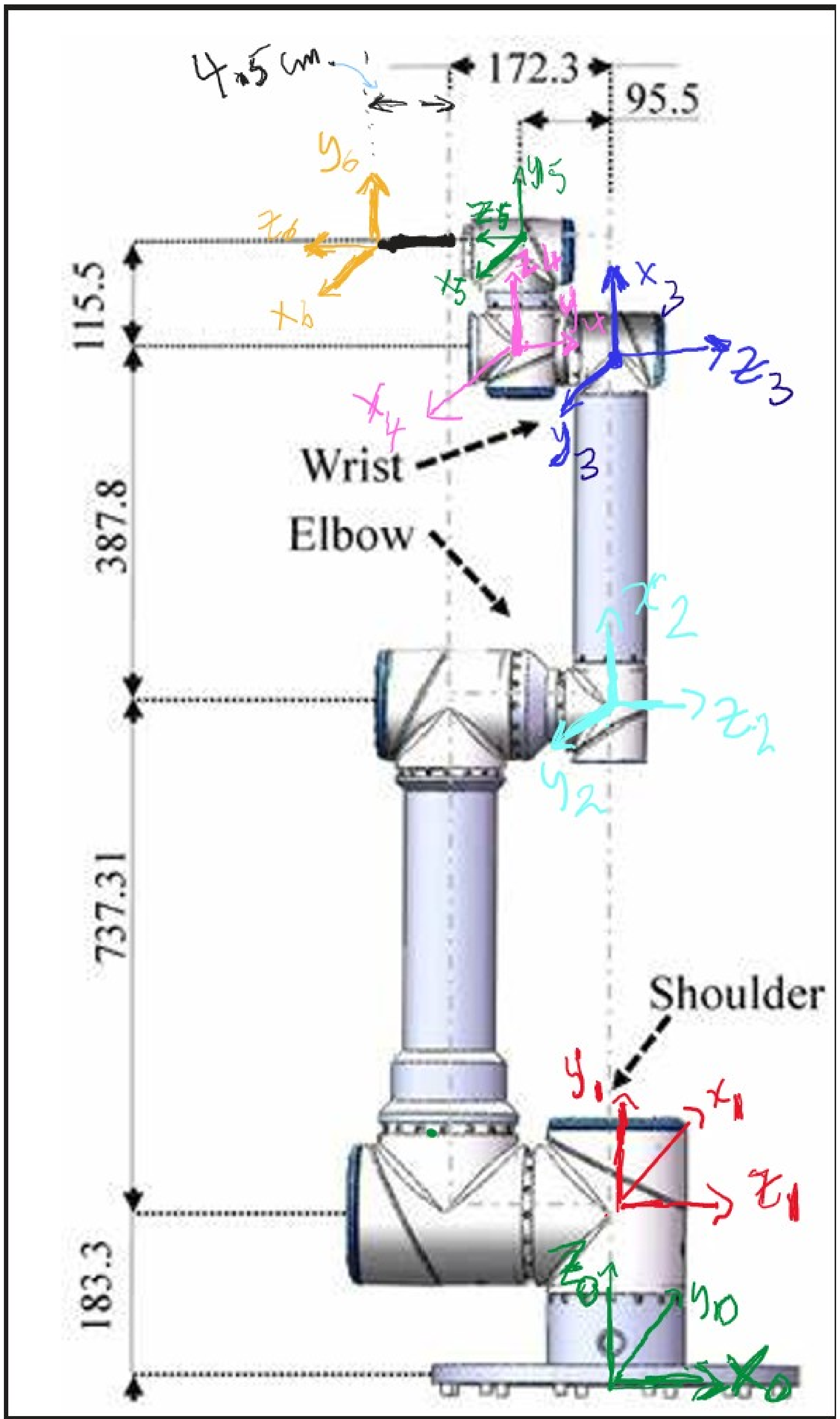
```
68·(sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 0.0768·(sin(θ₂)·cos(θ₁)·cos(θ₃) + sin(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄))·s
                         (0.0768·(sin(θ₂)·sin(θ₃) - cos(θ₂)·cos(θ₃))·cos(θ₄) - 0.0768·(-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·sin(θ₄))·si
                                                                                 (-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))
                                                                                 (sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·
                                                                                                 (-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))
sin(θ₅) + (-0.1155·sin(θ₁)·sin(θ₂)·sin(θ₃) + 0.1155·sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (0.1155·sin(θ₁)·sin(θ₂)·cos(θ₃) + 0.1155·sin(θ₁
in(θ₅) + (0.1155·sin(θ₂)·sin(θ₃)·cos(θ₁) - 0.1155·cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (-0.1155·sin(θ₂)·cos(θ₁)·cos(θ₃) - 0.1155·sin(θ₃)
n(θ₅) - (-0.1155·sin(θ₂)·sin(θ₃) + 0.1155·cos(θ₂)·cos(θ₃))·sin(θ₄) + (-0.1155·sin(θ₂)·cos(θ₃) - 0.1155·sin(θ₃)·cos(θ₂))·cos(θ₄)
·sin(θ₄) + (sin(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄)
sin(θ₄) + (-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄)
·cos(θ₄) + (-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·sin(θ₄)
)·sin(θ₃)·cos(θ₂))·sin(θ₄)  (0.0768·(-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - 0.0768·(sin(θ₁)·sin(θ₂)·cos(θ₃) + s
·cos(θ₁)·cos(θ₂))·sin(θ₄)   (0.0768·(sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - 0.0768·(-sin(θ₂)·cos(θ₁)·cos(θ₃) - s
                                                       (-0.0768·(-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·sin(θ₄) + 0.0768·(-sin(θ₂)·cos(θ₃
                                      ((-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (sin(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ
                                      ((sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ
                                                              (-(-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·sin(θ₄) + (-sin(θ₂)·cos(θ₃) - sin
in(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄))·cos(θ₅) + 0.0768·sin(θ₅)·cos(θ₁)
in(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄))·cos(θ₅) + 0.0768·sin(θ₁)·sin(θ₅)
) - sin(θ₃)·cos(θ₂))·cos(θ₄))·cos(θ₅)
₁)·sin(θ₃)·cos(θ₂))·sin(θ₄))·sin(θ₅) - cos(θ₁)·cos(θ₅)             ((-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (si
₃)·cos(θ₁)·cos(θ₂))·sin(θ₄))·sin(θ₅) - sin(θ₁)·cos(θ₅)             ((sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) - (-si
(θ₃)·cos(θ₂))·cos(θ₄))·sin(θ₅)                                                              (-(-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·sin(
       0                                                                           ⎤
       0                                                                           ⎥
       0                                                                           ⎥
n(θ₁)·sin(θ₂)·cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄))·sin(θ₅) - cos(θ₁)·cos(θ₅)⎥
n(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄))·sin(θ₅) - sin(θ₁)·cos(θ₅)⎥
```



```
θ₄) + (-sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·cos(θ₄))·sin(θ₅)                        ⎦
```

PROBLEM 3

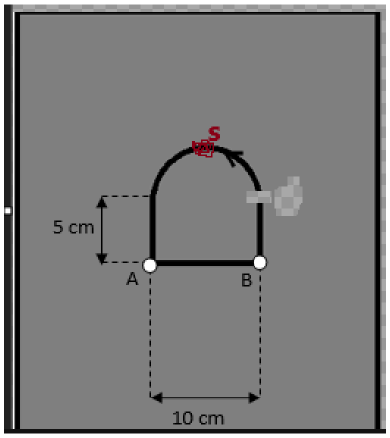
In this problem the goal was to simulate the UR3 robot arm drawing a shape consisting of a semi-circle and a rectangle. The first step taken was to adjust the last DH frame (frame 6) from Problem 1, and place it at the end of the drawing tool. This is shown in the picture below.



Consequently the DH table was also adjusted by adding 4.5 cm (0.045 m) to the d value in the last row. This is highlighted below.

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

The same steps performed in Problem 1 were used to find the new homogeneous transformation of the end-effector tool with respect to the base frame (x0, y0, z0). Then the same Method 2 from Lecture 8, which was described in Problem2 was used to recalculate the Jacobian matrix. The top of the semi-circle would be outside of the robot arms workspace if it started with the home position pictured in the problem description (which is the same as the above picture showing the DH frames), and started drawing at the point on the right edge of the semi-circle. As a potential solution, the starting point of the tool was considered to be at the top of the semi-circle (see image below). However, this approach still resulted in the robot arm being on the edge of its workspace and the tool did not draw the shape entirely accurately. To avoid this issue, at t0 =0, theta3 was set to 90 degrees (or pi/2 radians) and theta4, to -90 degrees (or -pi/2), and theta1=theta2=theta5=theta6=0, which resulted in the robot arm starting at lower position. The values of those angles was used in the end-effector homogeneous transform matrix to compute the end-effector tool starting position (x,y, z), which yielded (-0.2173, -0.3878, 1.03611).



In the code, the drawing was divided into five segments and the robot moved counter-clockwise. First one half of the semi-circle was drawn, then one 5 cm vertical line, then one 10 cm horizontal line, then the other 5 cm vertical line, and finally the other half of the semi-circle. Four seconds were allocated for each segment so that they added up to the 20 sec limit. Parametric equations for each segment were defined, then the numerical integration method was used to find the joint angles and the end-effector tool's position at each time interval. Based on the chosen base frame (x0, y0, z0), the drawing occurred in the y-z plane. Since the tool only moved in one plane, the initial x-coordinate position of the tool end did not change throughout the drawing.

For the first half of the semi-circle the parametric equation for a circle below as used:

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

where C_x, C_y, C_z are the coordinates of the center of the semi-circle, R the radius of the semi-circle which was= 10cm/2 or 5cm, omega is the constant angular speed as the robot moves along the semi-circle, and pi/2 is from the fact that the tool's starting point is at the top of the semi-circle. The coordinate for the center were found using the initial tool position and subtracting the radius R to the Z component:

C_x, C_y, C_z = initial_x_ee, initial_y_ee, initial_z_ee -R

omega was set to (pi/2) radians/4sec.

Then derivatives of the x, y, z parametric equations were taken. The tool was assumed to remain constantly in contact with the wall and did not wobble; so the wx, wy, wz component of the end-effector velocity vector were all set to 0. The parametric equation derivatives were combined with the wx, wy, wz into a single 6x1 vector.

Then the Jacobian matrix and the end-effector velocity vector were both computed with the initial joint angle values.

The equations to do the numerical integration were derived from the equations below:

q_dot = inverse_Jacobian * X_dot
with X_dot = [vx, vy, vz, wx, wy, wz}^T (which is the 6x1 vector described previously).

q_next = q_current + q_dot_next*delta_t

with delta_t = The segment duration time (sec)/the total number of data points (which was chosen as 1000 for each segment).

The above equations yielded the equation below, which was used in a loop to do the numerical integration:

q_next = q_current + (inverse_Jacobian*X_dot)*delta_t

For each small time interval (delta_t), the joint angles found were used in the end-effector tool homogeneous transformation matrix to find the x,y,z coordinate of the tool end's position by looking at the last column of the transform matrix. These values were then stored into lists that were later used to create the plot. The numerical Jacobian matrix was also calculated using the new joint angles, and X_dot was recalculated using the updated time. The loop was exited after all 1000 points were collected.

A similar process was repeated for the next segments, but with updated parametric equations. The following are the parametric equations for each of the segments.

Segment 2:

```
distance = DISTANCE_SEGMENT2  # 5 cm vertical segment
velocity = distance/segment_duration

# Parametric equations for the vertical line
x2 = C_x
y2 = all_ye_points[-1]
z2 = all_ze_points[-1] - velocity * t
```

In the above all_ye_points[-1] and all_ze_points[-1] are the last y and z coordinates, respectively, of the tool's end, which are taken from the collection lists from the previous segment. Also since the line is vertical only, the z changes with time. The term "velocity*t" is the distance covered and the negative sign in front of it, is because the tool is moving down in the -z axis direction.

Segment 3:

```
distance = DISTANCE_SEGMENT3  # 10 cm horizontal line
velocity = distance/segment_duration

# Parametric equations for the vertical line
x3 = C_x
y3 = all_ye_points[-1] + velocity * t
z3 = all_ze_points[-1]
```

Similar to the previous segments, but only the y coordinate changes, since the tool is moving horizontally along the y axis. The "velocity*t" expression has a positive sign in this case since the tool is moving in the +y axis direction.

Segment 4:

```
distance = DISTANCE_SEGMENT2  # second 5cm vertical line
velocity = distance/segment_duration

# Parametric equations for the vertical line
x4 = C_x
y4 = all_ye_points[-1]
z4 = all_ze_points[-1] + velocity * t
```

This is similar to the Segment 2 equations, with the velocity*t term being positive since the tool is moving in the +z-axis direction.

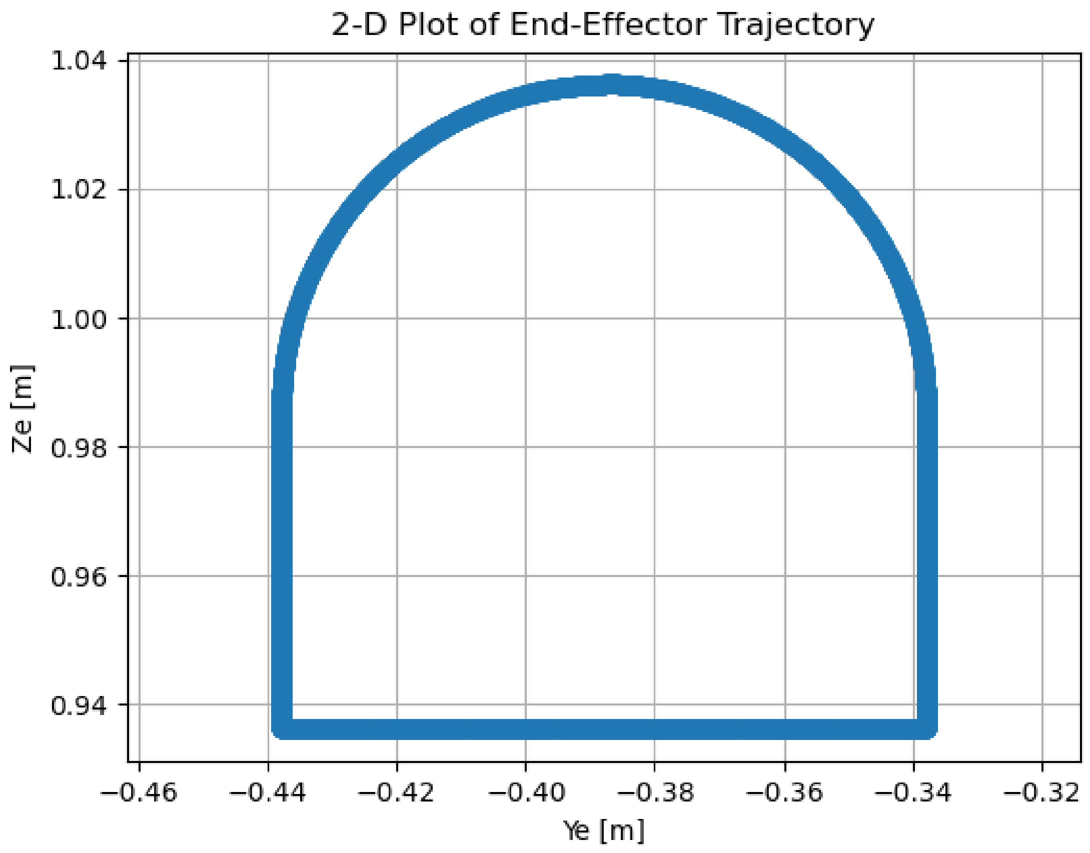
Segment 5 (second half of semi-circle):

```
x5 = C_x
y5 = all_ye_points[-1] + R*sp.cos(0+ omga*t)
z5 = all_ze_points[-1] + R*sp.sin(0+ omga*t)
```

This is similar to the parametric equations of the first half of the semi-circle, but the starting angle is changed to 0 (instead of pi/2), since the tool will start on the side of the semi-circle in this segment.

For each of those segments, the X_dot velocity vector function was recalculated. The initial joint angles were taken from the last Q vector used in the numerical integration loop that preceded the segment. Then starting numerical Jacobian matrices and Xe_dot vectors were recalculated using those joint angles. Then finally, a loop similar to the first segment was used and the lists collecting the tool's positions were updated. Please see the comments in the code for more information.

Below is the resulting plot, after all the segments were drawn.



REFERENCES
1- Mofaredi, Reza, *ENPM 662 Lecuture 8*
2- Spong, Mark, *Robot Modeling and Control*