

MTRX5700
EXPERIMENTAL ROBOTICS

Assignment 1

Author(s):

KAUSTHUB KRISHNAMURTHY

JAMES FERRIS

SACHITH GUNAWARDHANA

SID:

312086040

311220045

440623630

Due: March 25, 2015

1 Question 1

Refer to the MATLAB code in Appendix 9.a which was used to generate the homogeneous transformation matrices required.

1.a

The homogeneous transformation matrix for $\alpha = 10^\circ$, $\beta = 20^\circ$, $\gamma = 30^\circ$, ${}^A P_B = \{1 \ 2 \ 3\}^T$ looks like:

$$A = \begin{pmatrix} 0.9254 & 0.0180 & 0.3785 & 1.0000 \\ 0.1632 & 0.8826 & -0.4410 & 2.0000 \\ -0.3420 & 0.4698 & 0.8138 & 3.0000 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1.b

The homogeneous transformation matrix for $\alpha = 10^\circ$, $\beta = 30^\circ$, $\gamma = 30^\circ$, ${}^A P_B = \{3 \ 0 \ 0\}^T$ looks like:

$$B = \begin{pmatrix} 0.8529 & 0.0958 & 0.5133 & 3.0000 \\ 0.1504 & 0.8963 & -0.4172 & 0 \\ -0.5000 & 0.4330 & 0.7500 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1.c

The homogeneous transformation matrix for $\alpha = 90^\circ$, $\beta = 180^\circ$, $\gamma = -90^\circ$, ${}^A P_B = \{0 \ 0 \ 1\}^T$ looks like:

$$C_1 = \begin{pmatrix} -0 & -0 & -1 & 0 \\ -1 & -0 & 0 & 0 \\ -0 & 1 & -0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In Comparison: The homogeneous transformation matrix for $\alpha = 90^\circ$, $\beta = 180^\circ$, $\gamma = 270^\circ$, ${}^A P_B = \{0 \ 0 \ 1\}^T$ looks like:

$$C_2 = \begin{pmatrix} -0 & 0 & -1 & 0 \\ -1 & -0 & 0 & 0 \\ -0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Comparing these two we find that there is no functional difference between the two generated matrices. However we can notice that two of the zeroes are negative in C_1 but are positive in C_2 meaning that the approach to the orientation is different in C_1 and C_2 even if they result in the same thing.

Code Listing

See Appendix A [9.2]

2 Question 2

Refer to the MATLAB code in Appendix 9.a which was used to generate the homogeneous transformation matrices required.

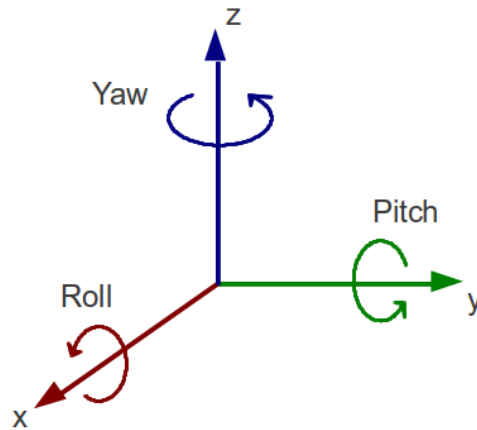


Figure 1: Roll Pitch and Yaw definitions (relative to axis names)

2.a Validity Check

We need to prove that the determinant of the matrix is equal to 1 and that the inverse of the matrix is equal to the transpose to prove validity.

2.a.i R_1

$$R_1 = \begin{pmatrix} 0.7500 & -0.4330 & -0.5000 \\ 0.2165 & 0.8750 & -0.4330 \\ 0.6250 & 0.2165 & 0.7500 \end{pmatrix}$$

$$\det(R_1) = 1.0000$$

$$R_1^T = \begin{pmatrix} 0.7500 & 0.2165 & 0.6250 \\ -0.4330 & 0.8750 & 0.2165 \\ -0.5000 & -0.4330 & 0.7500 \end{pmatrix}$$

$$R_1^{-1} = R_1^T \therefore R_1 \text{ is valid}$$

2.a.ii R_2

$$R_2 = \begin{pmatrix} 0.7725 & -0.4460 & -0.5150 \\ 0.2165 & 0.8750 & -0.4330 \\ 0.6000 & 0.2078 & 0.7200 \end{pmatrix}$$

$$\det(R_2) = 0.9888$$

$$R_2^{-1} = \begin{pmatrix} 0.7281 & 0.2165 & 0.6510 \\ -0.4204 & 0.8750 & 0.2255 \\ -0.4885 & -0.4330 & 0.7813 \end{pmatrix}$$

$$R_2^T = \begin{pmatrix} 0.7725 & 0.2165 & 0.6000 \\ -0.4460 & 0.8750 & 0.2078 \\ -0.5150 & -0.4330 & 0.7200 \end{pmatrix}$$

$R_2^{-1} \cong R_2^T \therefore R_2$ is *valid* within the limits of practical numerical applications.

2.a.iii R_3

$$R_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0.8660 & 0.5000 & 0 \\ -0.5000 & 0.8660 & 0 \end{pmatrix}$$

$$\det(R_3) = 1$$

$$R_3^{-1} = \begin{pmatrix} 0 & 0.8660 & -0.5000 \\ 0 & 0.5000 & 0.8660 \\ 1 & 0 & 0 \end{pmatrix}$$

$$R_3^T = \begin{pmatrix} 0 & 0.8660 & -0.5000 \\ 0 & 0.5000 & 0.8660 \\ 1 & 0 & 0 \end{pmatrix}$$

$$R_3^{-1} = R_3^T \therefore R_3 \text{ is } \textit{valid}$$

2.a.iv R_4

$$R_4 = \begin{pmatrix} -0.7500 & -0.2165 & -0.6250 \\ 0.4330 & -0.8750 & -0.2165 \\ 0.5000 & 0.4330 & -0.7500 \end{pmatrix}$$

$$\det(R_4) = -1$$

$$R_4^{-1} = \begin{pmatrix} -0.7500 & 0.4330 & 0.5000 \\ -0.2165 & -0.8750 & 0.4330 \\ -0.6250 & -0.2165 & -0.7500 \end{pmatrix}$$

$$R_4^T = \begin{pmatrix} -0.7500 & 0.4330 & 0.5000 \\ -0.2165 & -0.8750 & 0.4330 \\ -0.6250 & -0.2165 & -0.7500 \end{pmatrix}$$

Although $R_4^{-1} = R_4^T$, $\det(R_4) \neq 1 \therefore R_4$ is *invalid*

2.b Roll/Pitch/Yaw Angles

Roll Angle is α , Pitch Angle is β , Yaw Angle is γ .

Assumption: The believability of the angles can be determined purely mathematically. In reality these angles may not necessarily be believable depending on the application at hand. However, we cannot account for this without further information about the system.

2.b.i R_1

The following values are believable

$$\alpha_1 = 16.1021^\circ$$

$$\beta_1 = -36.6822^\circ$$

$$\gamma_1 = 16.1016^\circ$$

2.b.ii R_2

The following values are believable

$$\alpha_2 = 15.0665^\circ$$

$$\beta_2 = -36.8699^\circ$$

$$\gamma_2 = 15.0552^\circ$$

2.b.iii R_3

The following values are believable

$$\alpha_3 = 90^\circ$$

$$\beta_3 = 30^\circ$$

$$\gamma_3 = 89.5611^\circ$$

2.b.iv R_4

The following values are not believable as R_4 is an invalid rotational matrix

$$\alpha_4 = 150^\circ$$

$$\beta_4 = -30^\circ$$

$$\gamma_4 = 29.9990^\circ$$

2.c Angle Estimation

For a matrix such as R_2 we can adjust our matrix values slightly (making them less accurate i.e. to fewer decimal values) to still get a reasonable estimation of our angles. We can realise this by seeing that the determinant is so close to 1. R_4 , however, cannot give us a reasonable estimate for the angles as the determinant is too far from the required value of 1.

Code Listing

See Appendix A [9.2]

3 Question 3

DH Notation

i	θ	d	r	α
1	θ_1	67	100	$\pi/2$
2	θ_2	0	250	0
3	θ_2	0	0	$\pi/2$
4	θ_4	0	250	$-\pi/2$
5	θ_5	0	0	$-\pi/2$
6	θ_6	0	245	0

Table 1: D-H Variable Notation

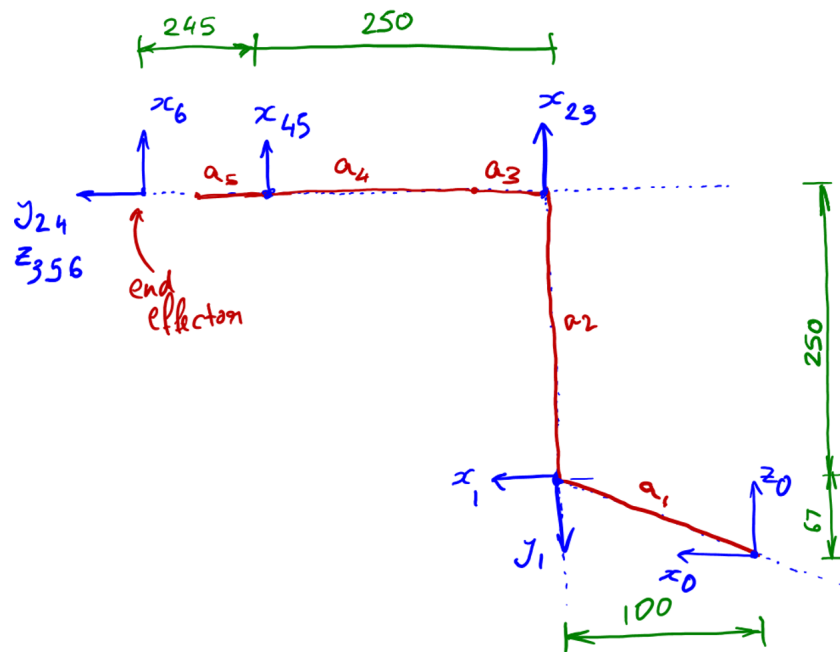


Figure 2: D-H Standard for Forward Kinematics Coordinate Systems

Assumption

Effectively taking the zero coordinate system above joint J1 allows us to eliminate one transformation from the system without drastically altering the kinematic solution behind it. The solution I provide below will be the transformation matrix with respect to my coordinate system 0.

Code Listing

See Appendix A [9.3]

Resulting Transformation Matrix

The resulting transformation matrix can be represented as:

$${}^0\mathbf{T}_6 = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = {}^0\mathbf{A}_1 {}^1\mathbf{A}_2 {}^2\mathbf{A}_3 {}^3\mathbf{A}_4 {}^4\mathbf{A}_5 {}^5\mathbf{A}_6$$

$$\mathbf{n}_x = -\sin \theta_6 (\cos \theta_4 \sin \theta_1 - \sin \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3)) - \cos \theta_6 (\cos \theta_5 (\sin \theta_1 \sin \theta_4 + \cos \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3)) + \sin \theta_5 (\cos \theta_1 \cos \theta_2 \sin \theta_3 + \cos \theta_1 \cos \theta_3 \sin \theta_2))$$

$$\mathbf{s}_x = \sin \theta_6 (\cos \theta_5 (\sin \theta_1 \sin \theta_4 + \cos \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3)) + \sin \theta_5 (\cos \theta_1 \cos \theta_2 \sin \theta_3 + \cos \theta_1 \cos \theta_3 \sin \theta_2)) - \cos \theta_6 (\cos \theta_4 \sin \theta_1 - \sin \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3))$$

$$\mathbf{a}_x = \sin \theta_5 (\sin \theta_1 \sin \theta_4 + \cos \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3)) - \cos \theta_5 (\cos \theta_1 \cos \theta_2 \sin \theta_3 + \cos \theta_1 \cos \theta_3 \sin \theta_2)$$

$$\mathbf{p}_x = 100 \cos \theta_1 + 250 \cos \theta_1 \cos \theta_2 - 250 \sin \theta_1 \sin \theta_4 - 250 \sin \theta_6 (\cos \theta_4 \sin \theta_1 - \sin \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3)) - 250 \cos \theta_6 (\cos \theta_5 (\sin \theta_1 \sin \theta_4 + \cos \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3)) + \sin \theta_5 (\cos \theta_1 \cos \theta_2 \sin \theta_3 + \cos \theta_1 \cos \theta_3 \sin \theta_2)) - 250 \cos \theta_4 (\cos \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_1 \cos \theta_2 \cos \theta_3)$$

$$\mathbf{n}_y = \sin \theta_6 (\cos \theta_1 \cos \theta_4 + \sin \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1)) + \cos \theta_6 (\cos \theta_5 (\cos \theta_1 \sin \theta_4 - \cos \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1)) - \sin \theta_5 (\cos \theta_2 \sin \theta_1 \sin \theta_3 + \cos \theta_3 \sin \theta_1 \sin \theta_2))$$

$$\mathbf{s}_y = \cos \theta_6 (\cos \theta_1 \cos \theta_4 + \sin \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1)) - \sin \theta_6 (\cos \theta_5 (\cos \theta_1 \sin \theta_4 - \cos \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1)) - \sin \theta_5 (\cos \theta_2 \sin \theta_1 \sin \theta_3 + \cos \theta_3 \sin \theta_1 \sin \theta_2))$$

$$\mathbf{a}_y = -\sin \theta_5 (\cos \theta_1 \sin \theta_4 - \cos \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1)) - \cos \theta_5 (\cos \theta_2 \sin \theta_1 \sin \theta_3 + \cos \theta_3 \sin \theta_1 \sin \theta_2)$$

$$\mathbf{p}_y = 100 \sin \theta_1 + 250 \cos \theta_2 \sin \theta_1 + 250 \cos \theta_1 \sin \theta_4 + 250 \sin \theta_6 (\cos \theta_1 \cos \theta_4 + \sin \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1)) - 250 \cos \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1) + 250 \cos \theta_6 (\cos \theta_5 (\cos \theta_1 \sin \theta_4 - \cos \theta_4 (\sin \theta_1 \sin \theta_2 \sin \theta_3 - \cos \theta_2 \cos \theta_3 \sin \theta_1)) - \sin \theta_5 (\cos \theta_2 \sin \theta_1 \sin \theta_3 + \cos \theta_3 \sin \theta_1 \sin \theta_2))$$

$$\mathbf{n}_z = \cos \theta_6 (\sin \theta_5 (\cos \theta_2 \cos \theta_3 - \sin \theta_2 \sin \theta_3) + \cos \theta_4 \cos \theta_5 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2)) - \sin \theta_4 \sin \theta_6 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2)$$

$$\mathbf{s}_z = -\sin \theta_6 (\sin \theta_5 (\cos \theta_2 \cos \theta_3 - \sin \theta_2 \sin \theta_3) + \cos \theta_4 \cos \theta_5 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2)) - \cos \theta_6 \sin \theta_4 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2)$$

$$\mathbf{a}_z = \cos \theta_5 (\cos \theta_2 \cos \theta_3 - \sin \theta_2 \sin \theta_3) - \cos \theta_4 \sin \theta_5 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2)$$

$$\mathbf{p}_z = 250 \sin \theta_2 + 250 \cos \theta_6 (\sin \theta_5 (\cos \theta_2 \cos \theta_3 - \sin \theta_2 \sin \theta_3) + \cos \theta_4 \cos \theta_5 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2)) + 250 \cos \theta_4 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2) - 250 \sin \theta_4 \sin \theta_6 (\cos \theta_2 \sin \theta_3 + \cos \theta_3 \sin \theta_2) + 67$$

World Coordinate Transformation

If it is necessary to calculate the transformation matrix with respect to the "real" (0, 0, 0) world coordinate system (as defined by the assignment document) we can take the following matrix:

$${}^wA_0 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 253 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and multiply it with our Transformation Matrix 0T_6 :

$${}^wT_6 = {}^wA_0 \cdot {}^0T_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 253 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^wT_6 = \begin{pmatrix} -n_y & -s_y & a_y & p_y \\ n_x & s_x & a_x & p_x + 253 \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This can be used for inverse kinematics problems by substituting our end effector position values into wT_6 in order to calculate the joint angles. We will see more about this in Question 6.

4 Question 4

4.a Modified DH Notation

i	r	α	d	θ
1	0	0	0	θ_1
2	0	$\pi/2$	0	θ_1
3	10	$\pi/2$	0	θ_1
4	12	$-\pi/2$	0	θ_4
5	0	$\pi/2$	10.5	θ_1

Table 2: Modified D-H Variable Notation

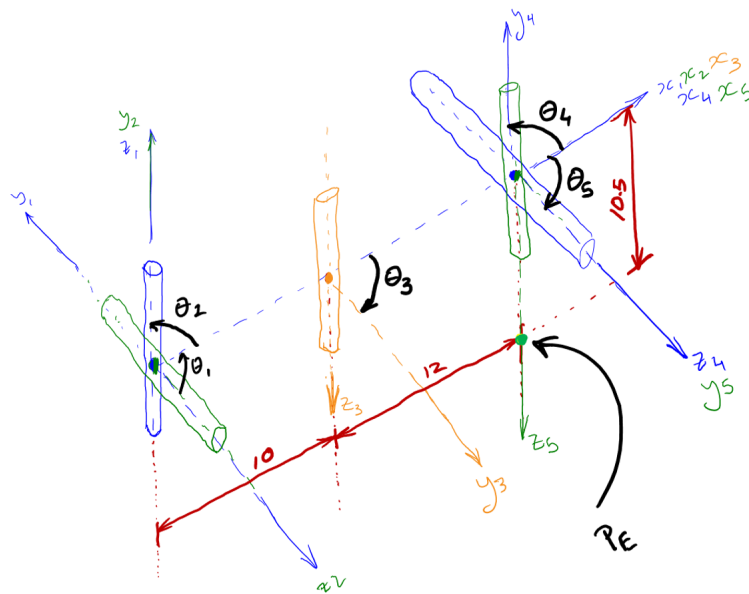


Figure 3: Derived Coordinate Systems

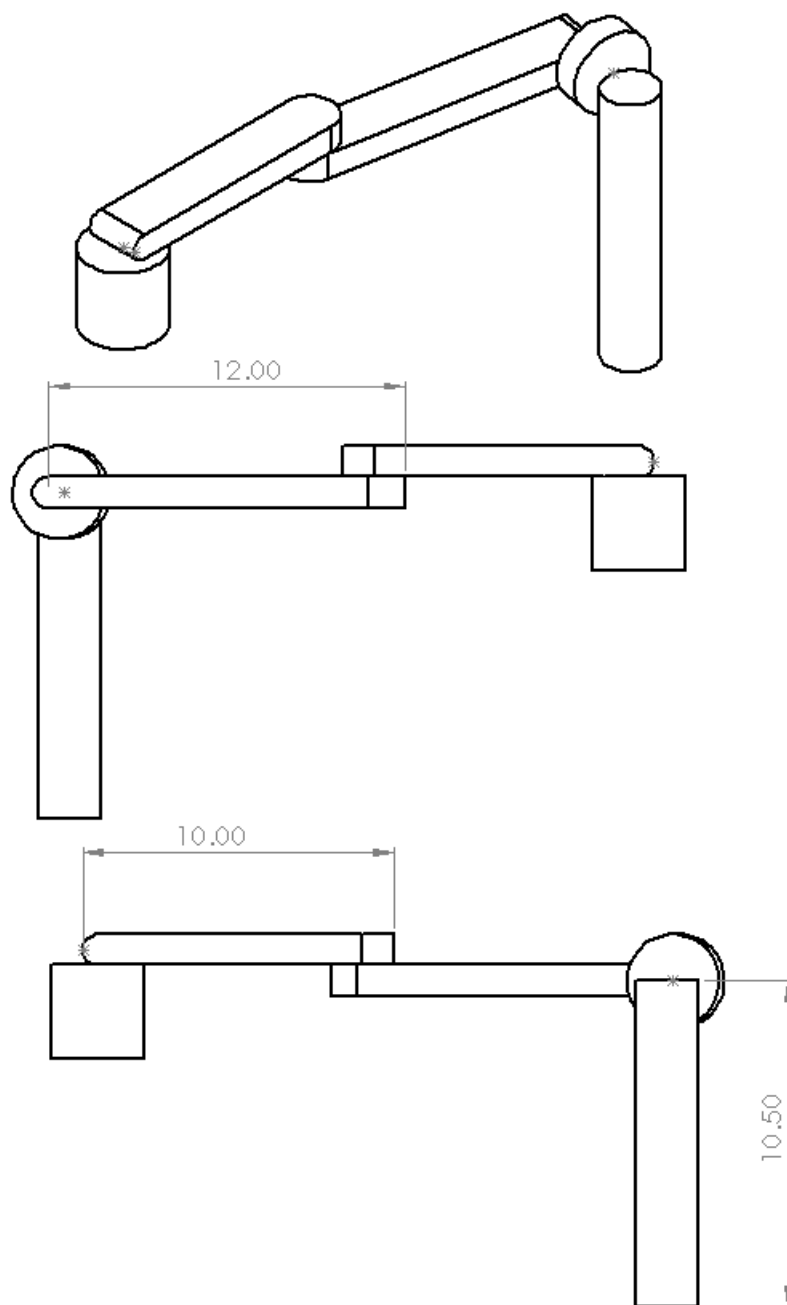


Figure 4: Derived Robot Arm Geometry

5 Question 5

5.a

5.a.i Workspace

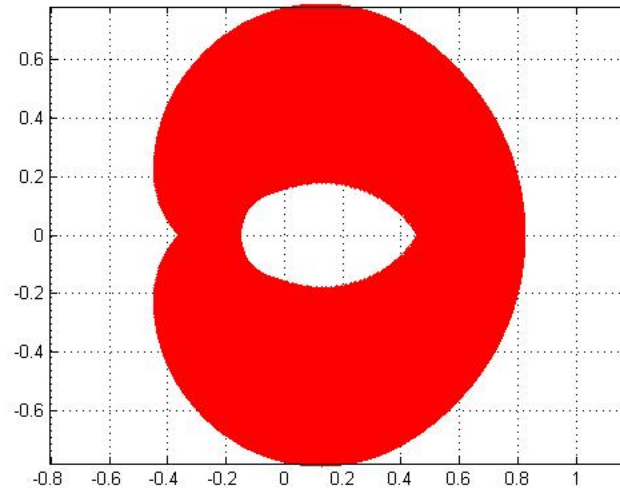


Figure 5: Workspace Plot (See Appendix A [9.4.i])

5.a.ii Configuration Space

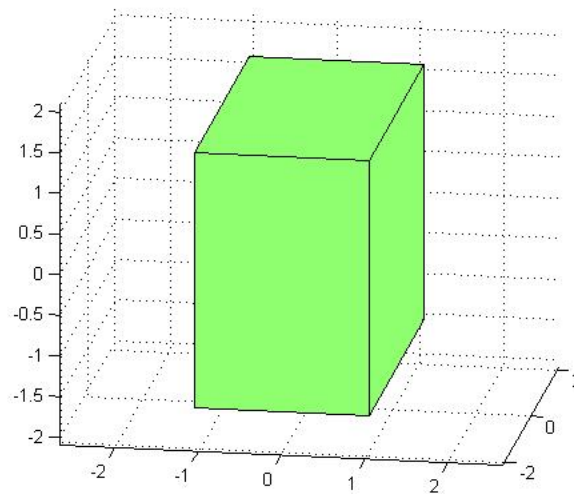


Figure 6: Configuration Space Plot (See Appendix A [9.4.ii])

5.b Singularities

6 Question 6

6.a

6.b

```
sa2 =
-133.57970015587177746184941710562 -56.676681927833899035216072316143
sa3 =
166.90301822803787842663334478947 13.096981771962121573366655210527
sa5 =
-123.32331807216610408729072583881 -46.420299844128225660657381049333
sb2 =
-121.68117590050949105729393864272 -45.154934789134486545751233766901
sb3 =
166.52624111137500451154270487582 13.473758888624995488457295124181
sb5 =
-134.84506521086551657675556438805 -58.318824099490512065212859512228
```

6.c

7 Question 7

7.a Qualitative Analysis of Block Stacking Method

The following code describes a general function which can move a block from any location and place it as part of the tower. The function takes as input the x and y coordinates of the block, and the block number. As long as it is known where the block is and how many blocks have already been placed, this function will place the block in its appropriate position of the tower (within limits of the robotic arm, of course).

This function is not optimised - it will not give the tower building time, which requires previous knowledge of the blocks and their locations. This function was designed with the mentality that nothing is previously known about the system - when a block is found, it must be added to the tower, no matter where that block is or how many blocks there are.

7.b CODE

```

1
2 ' Wrapper Functions...
3 ' Don 't change anything in this section...
4
5 #define MoveBlockHeight (40)
6 #define BlockWidth (80)
7 Function SetupArm()
8     Motor On
9     If (0) Then
10         Power High
11         SpeedS 2000 ' 80
12         AccelS 500 ' 50
13         Accel 50, 50 ' 15, 15
14         Speed 15
15         SpeedR 75
16         AccelR 200
17     Else
18         Power Low
19         SpeedS 80
20         AccelS 50

```

```

21         Accel 15, 15
22         Speed 3
23     EndIf
24     TLSet 1, XY(0, 0, 180, 0, 0, 0)
25     Tool 1
26 Fend
27 Function GoHome()
28     Move LJM(Here :Z(MoveBlockHeight + BlockWidth))
29     Move LJM(Here :U(90) :V(0) :W(180)) ROT
30     Move LJM(XY(0, 360, MoveBlockHeight + BlockWidth, 90, 0, 180))
31 Fend
32 Function CloseGripper
33     On 10
34     Wait 0.2
35 Fend
36 Function OpenGripper
37     Off 10
38     Wait 0.2
39 Fend
40 Function SetToolHeight(Height As Real)
41     If (Height < 0) Then
42         Height = 0
43     EndIf
44     'Go Here :Z(MoveBlockHeight + Height * BlockWidth) LJM
45     Move LJM(Here :Z(MoveBlockHeight + Height * BlockWidth))
46 Fend
47 Function PositionTool(XPos As Real, YPos As Real)
48     P1 = Here
49     Real z
50     z = CZ(P1)
51     'Go LJM(Here :X(XPos) :Y(YPos))
52     Move LJM(Here :X(XPos) :Y(YPos))
53 Fend
54 Function SetToolAngle(Angle As Real)
55     If (Angle < 0) Then
56         Angle = 0
57     EndIf
58     If (Angle > 180) Then
59         Angle = 180
60     EndIf
61     Move LJM(Here :U(90 + Angle) :V(0) :W(180)) ROT
62 Fend
63 ' ... end wrapper functions
64 ' .....
65
66 Function BuildBlock(x As Real, y As Real, BlockNumber As Integer)
67     SetToolHeight(4)
68     PositionTool(x, y)
69     SetToolHeight(0)
70     CloseGripper
71     SetToolHeight(4 + 0.1)
72     PositionTool(0, 490)
73     SetToolHeight((BlockNumber - 1))
74     OpenGripper
75 Fend
76
77 Function main
78     SetupArm()
79     GoHome
80     BuildBlock(0, 290, 1)
81     BuildBlock(300, 290, 2)
82     BuildBlock(300, 390, 3)
83     BuildBlock(-300, 390, 4)
84     BuildBlock(-300, 290, 5)
85
86 Fend

```

8 Question 8

8.a

8.b

8.c

9 Appendix

9.1 Question 1 Code Listings

```
1 close all
2 clear
3 clc
4
5 DEGREES = pi/180;
6 RADIANS = 180/pi;
7
8 fprintf('a\n');
9 roll = 10*DEGREES; %alpha
10 pitch = 20*DEGREES; %beta
11 yaw = 30*DEGREES; %gamma
12
13 aPb = [1 2 3];
14 aRb = [cos(roll)*cos(pitch), cos(roll)*sin(pitch)*sin(yaw)-sin(roll)*cos(yaw), cos(roll)*sin(pitch)*
    ↪ cos(yaw)+sin(roll)*sin(yaw);
15     sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos
    ↪ (yaw)-cos(roll)*sin(yaw);
16     -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
17 %transpose(aPb)
18
19 aTb = [aRb transpose(aPb); 0 0 0 1]
20
21 fprintf('b\n');
22
23 roll = 10*DEGREES; %alpha
24 pitch = 30*DEGREES; %beta
25 yaw = 30*DEGREES; %gamma
26
27 aPb = [3 0 0];
28 aRb = [cos(roll)*cos(pitch), cos(roll)*sin(pitch)*sin(yaw)-sin(roll)*cos(yaw), cos(roll)*sin(pitch)*
    ↪ cos(yaw)+sin(roll)*sin(yaw);
29     sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos
    ↪ (yaw)-cos(roll)*sin(yaw);
30     -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
31 %transpose(aPb)
32
33 aTb = [aRb transpose(aPb); 0 0 0 1]
34
35 fprintf('c\n');
36
37 roll = 90*DEGREES; %alpha
38 pitch = 180*DEGREES; %beta
39 yaw = -90*DEGREES; %gamma
40
41 aPb = [0 0 1];
42 aRb = [cos(roll)*cos(pitch), cos(roll)*sin(pitch)*sin(yaw)-sin(roll)*cos(yaw), cos(roll)*sin(pitch)*
    ↪ cos(yaw)+sin(roll)*sin(yaw);
43     sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos
    ↪ (yaw)-cos(roll)*sin(yaw);
44     -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
45 %transpose(aPb)
46
47 aTb = [aRb transpose(aPb); 0 0 0 1]
48
49 roll = 90*DEGREES; %alpha
50 pitch = 180*DEGREES; %beta
51 yaw = 270*DEGREES; %gamma
52 aPb = [0 0 1];
53 aRb = [cos(roll)*cos(pitch), cos(roll)*sin(pitch)*sin(yaw)-sin(roll)*cos(yaw), cos(roll)*sin(pitch)*
    ↪ cos(yaw)+sin(roll)*sin(yaw);
54     sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos
    ↪ (yaw)-cos(roll)*sin(yaw);
55     -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
56 %transpose(aPb)
57 aTb = [aRb transpose(aPb); 0 0 0 1]
```

9.2 Question 2 Code Listings

```
1 close all
2 clear
3 clc
4
5 DEGREES = pi/180;
6 RADIANS = 180/pi;
7
8 R1 = [0.7500, -0.4330, -0.5000; 0.2165, 0.8750, -0.4330; 0.6250, 0.2165, 0.7500]
9
10
11 determinantR1 = det(R1)
12 inverseR1 = inv(R1)
13 transposeR1 = transpose(R1)
14
15 beta1 = asin(-1*R1(3,1));
16 gamma1 = asin(R1(3,2)/cos(beta1));
17 alpha1 = acos(R1(1,1)/cos(beta1));
18
19 alpha1 = alpha1*RADIANS
20 beta1 = beta1*RADIANS
21 gamma1 = gamma1*RADIANS
22
23 R2 = [0.7725, -0.4460, -0.5150; 0.2165, 0.8750, -0.4330; 0.6000, 0.2078, 0.7200]
24 determinantR2 = det(R2)
25 inverseR2 = inv(R2)
26 transposeR2 = transpose(R2)
27
28 beta2 = asin(-1*R2(3,1));
29 gamma2 = asin(R2(3,2)/cos(beta2));
30 alpha2 = acos(R2(1,1)/cos(beta2));
31
32 alpha2 = alpha2*RADIANS
33 beta2 = beta2*RADIANS
34 gamma2 = gamma2*RADIANS
35
36 R3 = [0, 0, 1; 0.8660, 0.500, 0; -0.500, 0.8660, 0]
37 determinantR3 = det(R3)
38 inverseR3 = inv(R3)
39 transposeR3 = transpose(R3)
40
41 beta3 = asin(-1*R3(3,1));
42 gamma3 = asin(R3(3,2)/cos(beta3));
43 alpha3 = acos(R3(1,1)/cos(beta3));
44
45 alpha3 = alpha3*RADIANS
46 beta3 = beta3*RADIANS
47 gamma3 = gamma3*RADIANS
48
49 R4 = [-0.7500, -0.2165, -0.6250; 0.4330, -0.8750, -0.2165; 0.500, 0.4330, -0.7500]
50 determinantR4 = det(R4)
51 inverseR4 = inv(R4)
52 transposeR4 = transpose(R4)
53
54 beta4 = asin(-1*R4(3,1));
55 gamma4 = asin(R4(3,2)/cos(beta4));
56 alpha4 = acos(R4(1,1)/cos(beta4));
57
58 alpha4 = alpha4*RADIANS
59 beta4 = beta4*RADIANS
60 gamma4 = gamma4*RADIANS
```


9.3 Question 3 Code Listings

```
1 close all
2 clear all
3 clc
4
5 %%uninitialised symbolic values
6 th = sym('th',[1 6]);
7 d = sym('d',[1 6]);
8 al = sym('al',[1 6]);
9 r = sym('a',[1 6]);
10
11 %%initialised known values
12 %%d_i
13 d(1) = 67;
14 d(2) = 0;
15 d(3) = 0;
16 d(4) = 0;
17 d(5) = 0;
18 d(6) = 0;
19
20 %%alpha_i
21 al(1) = pi()/2;
22 al(2) = 0;
23 al(3) = -pi()/2;
24 al(4) = pi()/2;
25 al(5) = -pi()/2;
26 al(6) = 0;
27
28 %%r_i
29 r(1) = 100;
30 r(2) = 250;
31 r(3) = 0;
32 r(4) = 250;
33 r(5) = 0;
34 r(6) = 245;
35
36 %%individual link to link transformation matrices
37 A_01 = [cos(th(1)) -sin(th(1))*cos(al(1)) sin(th(1))*sin(al(1)) r(1)*cos(th(1));sin(th(1)) cos(th(1))
    ↪ )*cos(al(1)) -cos(th(1))*sin(al(1)) r(1)*sin(th(1));0 sin(al(1)) cos(al(1)) d(1);0 0 0 1];
38 A_12 = [cos(th(2)) -sin(th(2))*cos(al(2)) sin(th(2))*sin(al(2)) r(2)*cos(th(2));sin(th(2)) cos(th(2))
    ↪ )*cos(al(2)) -cos(th(2))*sin(al(2)) r(2)*sin(th(2));0 sin(al(2)) cos(al(2)) d(2);0 0 0 1];
39 A_23 = [cos(th(3)) -sin(th(3))*cos(al(3)) sin(th(3))*sin(al(3)) r(3)*cos(th(3));sin(th(3)) cos(th(3))
    ↪ )*cos(al(3)) -cos(th(3))*sin(al(3)) r(3)*sin(th(3));0 sin(al(3)) cos(al(3)) d(3);0 0 0 1];
40 A_34 = [cos(th(4)) -sin(th(4))*cos(al(4)) sin(th(4))*sin(al(4)) r(4)*cos(th(4));sin(th(4)) cos(th(4))
    ↪ )*cos(al(4)) -cos(th(4))*sin(al(4)) r(4)*sin(th(4));0 sin(al(4)) cos(al(4)) d(4);0 0 0 1];
41 A_45 = [cos(th(5)) -sin(th(5))*cos(al(5)) sin(th(5))*sin(al(5)) r(5)*cos(th(5));sin(th(5)) cos(th(5))
    ↪ )*cos(al(5)) -cos(th(5))*sin(al(5)) r(5)*sin(th(5));0 sin(al(5)) cos(al(5)) d(5);0 0 0 1];
42 A_56 = [cos(th(6)) -sin(th(6))*cos(al(6)) sin(th(6))*sin(al(6)) r(6)*cos(th(6));sin(th(6)) cos(th(6))
    ↪ )*cos(al(6)) -cos(th(6))*sin(al(6)) r(6)*sin(th(6));0 sin(al(6)) cos(al(6)) d(6);0 0 0 1];
43
44 %%transformation matrix for end effector pose with respect to coordinate
45 %%system zero
46 T_06 = A_01*A_12*A_23*A_34*A_45*A_56
```

9.4 Question 5 Code Listings

9.4.i Workspace Plot

```
1 %q5 - workspace
2 %http://au.mathworks.com/help/fuzzy/examples/modeling-inverse-kinematics-in-a-robotic-arm.html
3 close all
4 clear
5 clc
6
7 mm = 10^-3;
8 DEGREES = pi/180;
9 RADIANS = 180/pi;
10
11 L1 = 250*mm;
12 L2 = 250*mm;
13 L3 = 315*mm;
14
15 t1 = linspace(-pi/3,pi/3);
16 t2 = linspace(-2*pi/3, 2*pi/3);
17 t3 = linspace(-pi/2, pi/2);
18
19 [T1, T2, T3] = meshgrid(t1,t2,t3);
20
21 X = L1*cos(T1)+L2*cos(T1+T2)+L3*cos(T1+T2+T3); %typo here put in final
22 Y = L1*sin(T1)+L2*sin(T1+T2)+L3*sin(T1+T2+T3);
23
24 plot(X(:), Y(:), 'r. '); %last column of X and Y
25 axis equal
26 grid
```

9.4.ii Configuration Space Plot

```
1 %q5 - configuration space
2 close all
3 clear
4 clc
5
6 mm = 10^-3;
7 DEGREES = pi/180;
8 RADIANS = 180/pi;
9
10 X = [-pi/3, pi/3];
11 Y = [-2*pi/3, 2*pi/3];
12 Z = [-pi/2, pi/2];
13
14 config_space_3dof(X,Y,Z);
```

```
1 function config_space_3dof(X,Y,Z)
2
3     hold on
4     grid
5     axis equal
6
7     cornersX = [X(2), X(2), X(1), X(1)];
8     cornersY = [Y(2), Y(1), Y(1), Y(2)];
9     cornersZ = [Z(1), Z(1), Z(1), Z(1)];
10    fill3(cornersX, cornersY, cornersZ,1);
11
12    cornersX = [X(2), X(2), X(1), X(1)];
13    cornersY = [Y(2), Y(1), Y(1), Y(2)];
14    cornersZ = [Z(2), Z(2), Z(2), Z(2)];
15    fill3(cornersX, cornersY, cornersZ,1);
16
```

```
17     cornersX = [X(2), X(2), X(2), X(2)];
18     cornersY = [Y(2), Y(1), Y(1), Y(2)];
19     cornersZ = [Z(1), Z(1), Z(2), Z(2)];
20     fill3(cornersX, cornersY, cornersZ,1);
21
22     cornersX = [X(1), X(1), X(1), X(1)];
23     cornersY = [Y(2), Y(1), Y(1), Y(2)];
24     cornersZ = [Z(1), Z(1), Z(2), Z(2)];
25     fill3(cornersX, cornersY, cornersZ,1);
26
27     cornersX = [X(1), X(2), X(2), X(1)];
28     cornersY = [Y(2), Y(2), Y(2), Y(2)];
29     cornersZ = [Z(1), Z(1), Z(2), Z(2)];
30     fill3(cornersX, cornersY, cornersZ,1);
31
32     cornersX = [X(1), X(2), X(2), X(1)];
33     cornersY = [Y(1), Y(1), Y(1), Y(1)];
34     cornersZ = [Z(1), Z(1), Z(2), Z(2)];
35     fill3(cornersX, cornersY, cornersZ,1);
36
37     hold off
38 end
```

9.5 Q6

