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
$\alpha = 10°$, $\beta = 20°$, $\gamma = 30°$, ${}^{A}P_{B} = \{1\ 2\ 3\}^{T}$
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
$\alpha = 10°$, $\beta = 30°$, $\gamma = 30°$, ${}^{A}P_{B} = \{3\ 0\ 0\}^{T}$
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
$\alpha = 90°$, $\beta = 180°$, $\gamma =$ -90°, ${}^{A}P_{B} = \{0\ 0\ 1\}^{T}$
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
$\alpha = 90°$, $\beta = 180°$, $\gamma = 270°$, ${}^{A}P_{B} = \{0\ 0\ 1\}^{T}$
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.

2 Question 2

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

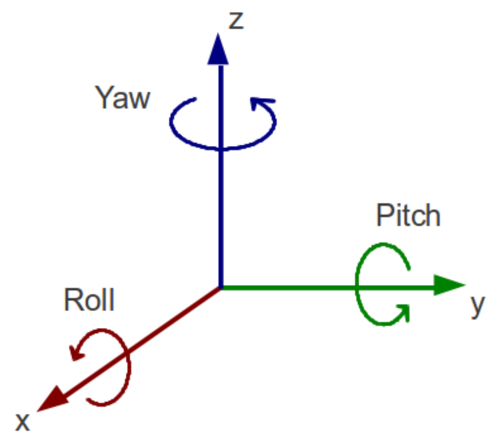

Figure 1: Roll Pitch and Yaw definitions (relative to axis names)

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

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

$det(R_1) = 1.0000$

$$R_1^{-1} = \begin{pmatrix} 0.7500 & 0.2165 & 0.6250 \\ -0.4330 & 0.8750 & 0.2165 \\ -0.5000 & -0.4330 & 0.7500 \end{pmatrix}$$

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

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

2.a.ii R_2

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

$det(R_2) = 0.9888$

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

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

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

2.a.iii R_3

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

$det(R_3) = 1$

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

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

$R_3^{-1} = R_3^T \therefore R_3$ is *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
α_1 = 16.1021°
β_1 = -36.6822°
γ_1 = 16.1016°

2.b.ii R_2

The following values are believable
α_2 = 15.0665°
β_2 = -36.8699°
γ_2 = 15.0552°

2.b.iii R_3

The following values are believable
α_3 = 90°
β_3 = 30°
γ_3 = 89.5611°

2.b.iv R_4

The following values are not believable as R_4 is an invalid rotational matrix
α_4 = 150°
β_4 = -30°
γ_4 = 29.9990°

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.

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

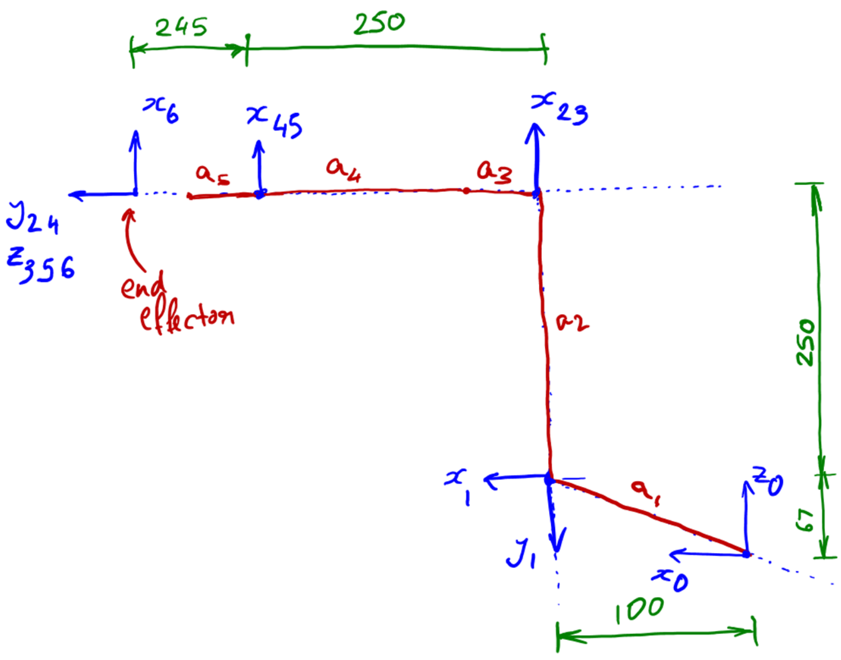

Figure 2: D-H Standard for Forward Kinematics Coordinate Systems

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.

Resulting Transformation Matrix

The resulting transformation matrix can be represented as:

$$\mathbf{{}^0T_6} = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{{}^0A_1{}^1A_2{}^2A_3{}^3A_4{}^4A_5{}^5A_6}$$

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

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

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

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

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

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

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

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

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

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

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

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

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:

$$^{\mathbf{W}}\mathbf{A_0} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 253 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and multiply it with our Transformation Matrix $^{0}T_6$:

$$^{W}T_6 = {}^{W}A_0 \cdot {}^{0}T_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 253 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

This can be used for inverse kinematics problems by substituting our end effector position values into $^{W}T_6$ in order to calculate the joint angles. We will see more about this in Question 6.

4 Question 4

4.a

4.b

4.c

5 Question 5

5.a Workspace

```
%q5 - workspace
%http://au.mathworks.com/help/fuzzy/examples/modeling-inverse-kinematics-in-a-robotic-arm.html
close all
clear
clc

mm = 10^-3;
DEGREES = pi/180;
RADIANS = 180/pi;

L1 = 250*mm;
L2= 250*mm;
L3 = 315*mm;

t1 = linspace(-pi/3,pi/3);
t2 = linspace(-2*pi/3, 2*pi/3);
t3 = linspace(-pi/2, pi/2);

[T1, T2, T3] = meshgrid(t1,t2,t3);

X = L1*cos(T1)+L2*cos(T1+T2)+L3*cos(T1+T2+T3);  %typo here put in final
Y = L1*sin(T1)+L2*sin(T1+T2)+L3*sin(T1+T2+T3);

plot(X(:), Y(:), 'r.'); %last column of X and Y
axis equal
grid
```

5.b Configuration Space

```
%q5 - configuration space
close all
clear
clc

mm = 10^-3;
DEGREES = pi/180;
RADIANS = 180/pi;

X = [-pi/3, pi/3];
Y = [-2*pi/3, 2*pi/3];
Z = [-pi/2, pi/2];

config_space_3dof(X,Y,Z);
```

```
function config_space_3dof(X,Y,Z)

    hold on
    grid
    axis equal

```

```matlab
    cornersX = [X(2), X(2), X(1), X(1)];
    cornersY = [Y(2), Y(1), Y(1), Y(2)];
    cornersZ = [Z(1), Z(1), Z(1), Z(1)];
    fill3(cornersX, cornersY, cornersZ,1);

    cornersX = [X(2), X(2), X(1), X(1)];
    cornersY = [Y(2), Y(1), Y(1), Y(2)];
    cornersZ = [Z(2), Z(2), Z(2), Z(2)];
    fill3(cornersX, cornersY, cornersZ,1);

    cornersX = [X(2), X(2), X(2), X(2)];
    cornersY = [Y(2), Y(1), Y(1), Y(2)];
    cornersZ = [Z(1), Z(1), Z(2), Z(2)];
    fill3(cornersX, cornersY, cornersZ,1);

    cornersX = [X(1), X(1), X(1), X(1)];
    cornersY = [Y(2), Y(1), Y(1), Y(2)];
    cornersZ = [Z(1), Z(1), Z(2), Z(2)];
    fill3(cornersX, cornersY, cornersZ,1);

    cornersX = [X(1), X(2), X(2), X(1)];
    cornersY = [Y(2), Y(2), Y(2), Y(2)];
    cornersZ = [Z(1), Z(1), Z(2), Z(2)];
    fill3(cornersX, cornersY, cornersZ,1);

    cornersX = [X(1), X(2), X(2), X(1)];
    cornersY = [Y(1), Y(1), Y(1), Y(1)];
    cornersZ = [Z(1), Z(1), Z(2), Z(2)];
    fill3(cornersX, cornersY, cornersZ,1);

    hold off
end
```

5.c Singularities

5.d

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

```
'''''''''''''''''''''''''''''''''''''''''''
' Wrapper Functions...                    '
' Don 't change anything in this section...'
'''''''''''''''''''''''''''''''''''''''''''
#define MoveBlockHeight (40)
#define BlockWidth (80)
Function SetupArm()
    Motor On
    If (0) Then
        Power High
        SpeedS 2000 ' 80
        AccelS 500 ' 50
        Accel 50, 50 ' 15, 15
        Speed 15
        SpeedR 75
        AccelR 200
    Else
        Power Low
        SpeedS 80
        AccelS 50
        Accel 15, 15
        Speed 3
    EndIf
    TLSet 1, XY(0, 0, 180, 0, 0, 0)
    Tool 1
Fend
Function GoHome()
    Move LJM(Here :Z(MoveBlockHeight + BlockWidth))
    Move LJM(Here :U(90) :V(0) :W(180)) ROT
    Move LJM(XY(0, 360, MoveBlockHeight + BlockWidth, 90, 0, 180))
Fend
Function CloseGripper
    On 10
    Wait 0.2
Fend
Function OpenGripper
    Off 10
    Wait 0.2
Fend
Function SetToolHeight(Height As Real)
    If (Height < 0) Then
        Height = 0
    EndIf
    'Go Here :Z(MoveBlockHeight + Height * BlockWidth) LJM
    Move LJM(Here :Z(MoveBlockHeight + Height * BlockWidth))
Fend
```

```
Function PositionTool(XPos As Real, YPos As Real)
    P1 = Here
    Real z
    z = CZ(P1)
    'Go LJM(Here :X(XPos) :Y(YPos))
    Move LJM(Here :X(XPos) :Y(YPos))
Fend
Function SetToolAngle(Angle As Real)
    If (Angle < 0) Then
        Angle = 0
    EndIf
    If (Angle > 180) Then
        Angle = 180
    EndIf
    Move LJM(Here :U(90 + Angle) :V(0) :W(180)) ROT
Fend
' ... end wrapper functions                  '
''''''''''''''''''''''''''''''''''''''''''''

Function BuildBlock(x As Real, y As Real, BlockNumber As Integer)
    SetToolHeight(4)
    PositionTool(x, y)
    SetToolHeight(0)
    CloseGripper
    SetToolHeight(4 + 0.1)
    PositionTool(0, 490)
    SetToolHeight((BlockNumber - 1))
    OpenGripper
Fend

Function main
    SetupArm()
    GoHome
    BuildBlock(0, 290, 1)
    BuildBlock(300, 290, 2)
    BuildBlock(300, 390, 3)
    BuildBlock(-300, 390, 4)
    BuildBlock(-300, 290, 5)

Fend
```

8 Question 8

8.a

8.b

8.c

9 Appendix

9.1 Question 1 Code Listings

```
close all
clear
clc

DEGREES = pi/180;
RADIANS = 180/pi;

fprintf('a)\n');
roll = 10*DEGREES; %alpha
pitch = 20*DEGREES; %beta
yaw = 30*DEGREES; %gamma

aPb = [1 2 3];
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);
    sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos(yaw)-cos(roll)*sin(yaw);
    -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
%transpose(aPb)

aTb = [aRb transpose(aPb); 0 0 0 1]

fprintf('b)\n');

roll = 10*DEGREES; %alpha
pitch = 30*DEGREES; %beta
yaw = 30*DEGREES; %gamma

aPb = [3 0 0];
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);
    sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos(yaw)-cos(roll)*sin(yaw);
    -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
%transpose(aPb)

aTb = [aRb transpose(aPb); 0 0 0 1]

fprintf('c)\n');

roll = 90*DEGREES; %alpha
pitch = 180*DEGREES; %beta
yaw = -90*DEGREES; %gamma

aPb = [0 0 1];
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);
    sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos(yaw)-cos(roll)*sin(yaw);
    -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
%transpose(aPb)

aTb = [aRb transpose(aPb); 0 0 0 1]

roll = 90*DEGREES; %alpha
pitch = 180*DEGREES; %beta
yaw = 270*DEGREES; %gamma
aPb = [0 0 1];
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);
    sin(roll)*cos(pitch), sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw), sin(roll)*sin(pitch)*cos(yaw)-cos(roll)*sin(yaw);
    -sin(pitch), cos(pitch)*sin(yaw), cos(pitch)*cos(yaw)];
%transpose(aPb)
aTb = [aRb transpose(aPb); 0 0 0 1]
```

9.2 Question 2 Code Listings

```
close all
clear
clc

DEGREES = pi/180;
RADIANS = 180/pi;

R1 = [0.7500, -0.4330, -0.5000; 0.2165,0.8750, -0.4330; 0.6250, 0.2165, 0.7500]


determinantR1 = det(R1)
inverseR1 = inv(R1)
transposeR1 = transpose(R1)

beta1 = asin(-1*R1(3,1));
gamma1 = asin(R1(3,2)/cos(beta1));
alpha1 = acos(R1(1,1)/cos(beta1));

alpha1 = alpha1*RADIANS
beta1 = beta1*RADIANS
gamma1 = gamma1*RADIANS

R2 = [0.7725, -0.4460, -0.5150; 0.2165, 0.8750, -0.4330; 0.6000, 0.2078, 0.7200]
determinantR2 = det(R2)
inverseR2 = inv(R2)
transposeR2 = transpose(R2)

beta2 = asin(-1*R2(3,1));
gamma2 = asin(R2(3,2)/cos(beta2));
alpha2 = acos(R2(1,1)/cos(beta2));

alpha2 = alpha2*RADIANS
beta2 = beta2*RADIANS
gamma2 = gamma2*RADIANS

R3 = [0, 0, 1; 0.8660, 0.500, 0; -0.500, 0.8660, 0]
determinantR3 = det(R3)
inverseR3 = inv(R3)
transposeR3 = transpose(R3)

beta3 = asin(-1*R3(3,1));
gamma3 = asin(R3(3,2)/cos(beta3));
alpha3 = acos(R3(1,1)/cos(beta3));

alpha3 = alpha3*RADIANS
beta3 = beta3*RADIANS
gamma3 = gamma3*RADIANS

R4 = [-0.7500, -0.2165, -0.6250; 0.4330, -0.8750, -0.2165; 0.500, 0.4330, -0.7500]
determinantR4 = det(R4)
inverseR4 = inv(R4)
transposeR4 = transpose(R4)

beta4 = asin(-1*R4(3,1));
gamma4 = asin(R4(3,2)/cos(beta4));
alpha4 = acos(R4(1,1)/cos(beta4));

alpha4 = alpha4*RADIANS
beta4 = beta4*RADIANS
gamma4 = gamma4*RADIANS
```

9.3 Question 3 Code Listings

```
close all
clear all
clc

%%uninitialised symbolic values
th = sym('th',[1 6]);
d = sym('d', [1 6]);
al = sym('al',[1 6]);
r = sym('a',[1 6]);

%%initialised known values
%%d_i
d(1) = 67;
d(2) = 0;
d(3) = 0;
d(4) = 0;
d(5) = 0;
d(6) = 0;

%%alpha_i
al(1) = pi()/2;
al(2) = 0;
al(3) = -pi()/2;
al(4) = pi()/2;
al(5) = -pi()/2;
al(6) = 0;

%%r_i
r(1) = 100;
r(2) = 250;
r(3) = 0;
r(4) = 250;
r(5) = 0;
r(6) = 245;

%%individual link to link transformation matrices
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];
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];
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];
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];
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];
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];

%%transformation matrix for end effector pose with respect to coordinate
%%system zero
T_06 = A_01*A_12*A_23*A_34*A_45*A_56
```

9.4 Q4

9.5 Q5

9.6 Q6

9.7 Q7

9.8 Q8