THE UNIVERSITY OF SYDNEY

ACADEMIC BOARD POLICY: ACADEMIC DISHONESTY AND PLAGIARISM

COMPLIANCE STATEMENT

INDIVIDUAL / COLLABORATIVE WORK

I/We certify that:

(1) I/We have read and understood the *University of Sydney Academic Board Policy: Academic Dishonesty and Plagiarism*;

(2) I/We understand that failure to comply with the *Academic Board Policy: Academic Dishonesty and Plagiarism* can lead to the University commencing proceedings against me/us for potential student misconduct under Chapter 8 of the *University of Sydney By-Law 1999* (as amended);

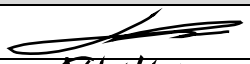
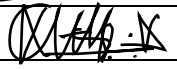
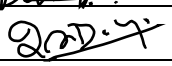
(3) This Work is substantially my/our own, and to the extent that any part of this Work is not my/our own I/we have indicated that it is not my/our own by Acknowledging the Source of that part or those parts of the Work;

(4) No part of this Work has been previously submitted for summative assessment, whether in this Unit of Study or another Unit of Study (unless the Examiner has given specific approval for this to occur);

(5) I/We accept that the Work submitted with this Compliance Statement is the version of the Work that will be assessed.

Complete all fields in the following table for yourself *(in the case of individual work)*, *OR* for ALL members of your group *(in the case of collaborative work)*.

Typewritten name(s) in the signature column will be accepted as signature(s) for electronic submissions only.

Name	Signature	SID	Date
James Ferris		311220045	25/3/15
Kausthub Krishnamurthy		312086040	25/3/15
Sachith Gunawardhana		440623630	25/3/15

MTRX5700

Experimental Robotics

Assignment 1

Author(s):
Kausthub Krishnamurthy
James Ferris
Sachith Gunawardhana

SID:
312086040
311220045
440623630

Due: March 25, 2015

1 Question 1

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°, ${}^A P_B = \{0\ 0\ 1\}^T$
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 [9.1]

2 Question 2

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

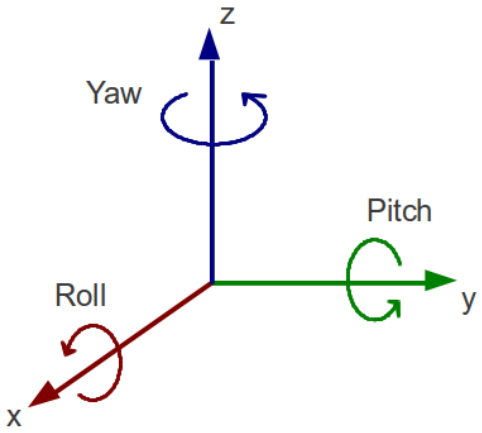


Figure 1: Roll Pitch and Yaw definitions (relative to axis names)

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

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

$det(R_1) = 1.0000$

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

$R_1^{-1} = R_1^T \therefore R_1$ 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
$\alpha_1 = 16.1021^\circ$
$\beta_1 = -36.6822^\circ$
$\gamma_1 = 16.1016^\circ$

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

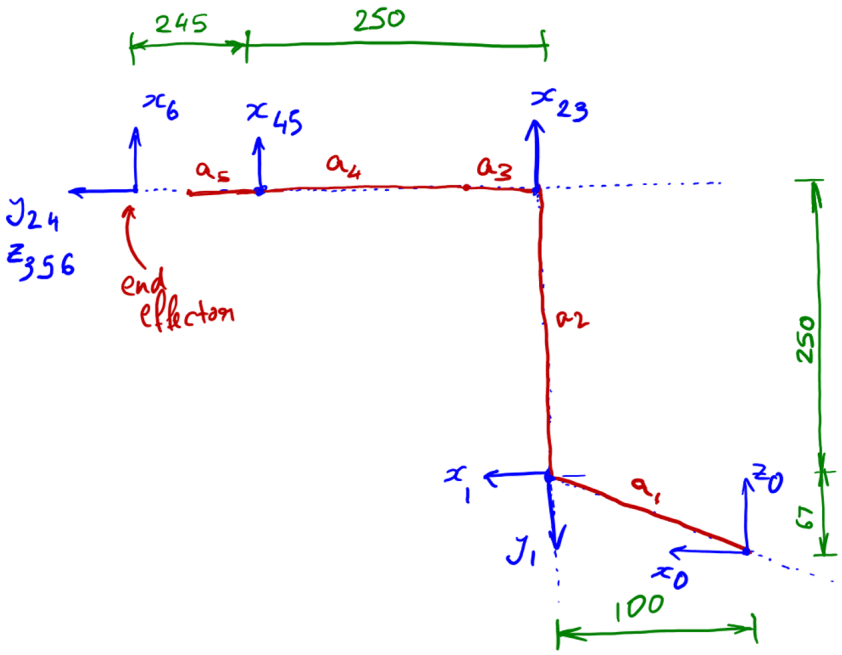


Figure 2: D-H Standard for Forward Kinematics Coordinate Systems

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:

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

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 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

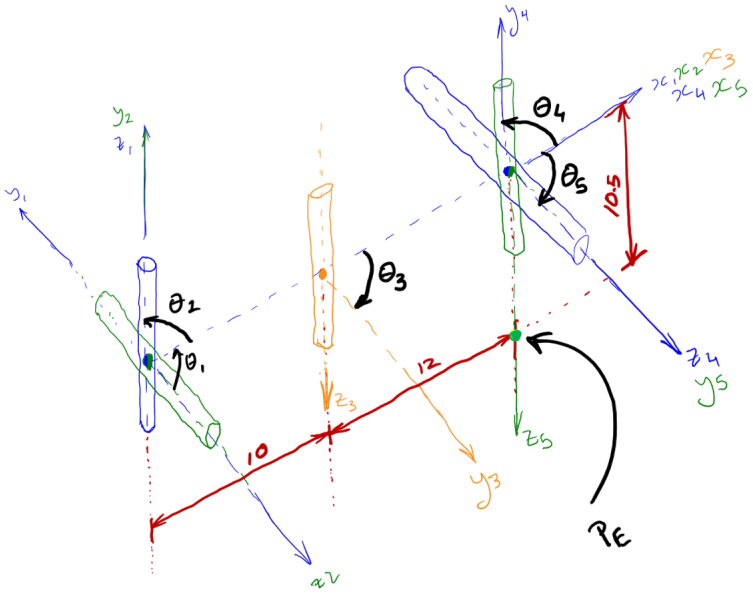


Figure 3: Derived Coordinate Systems

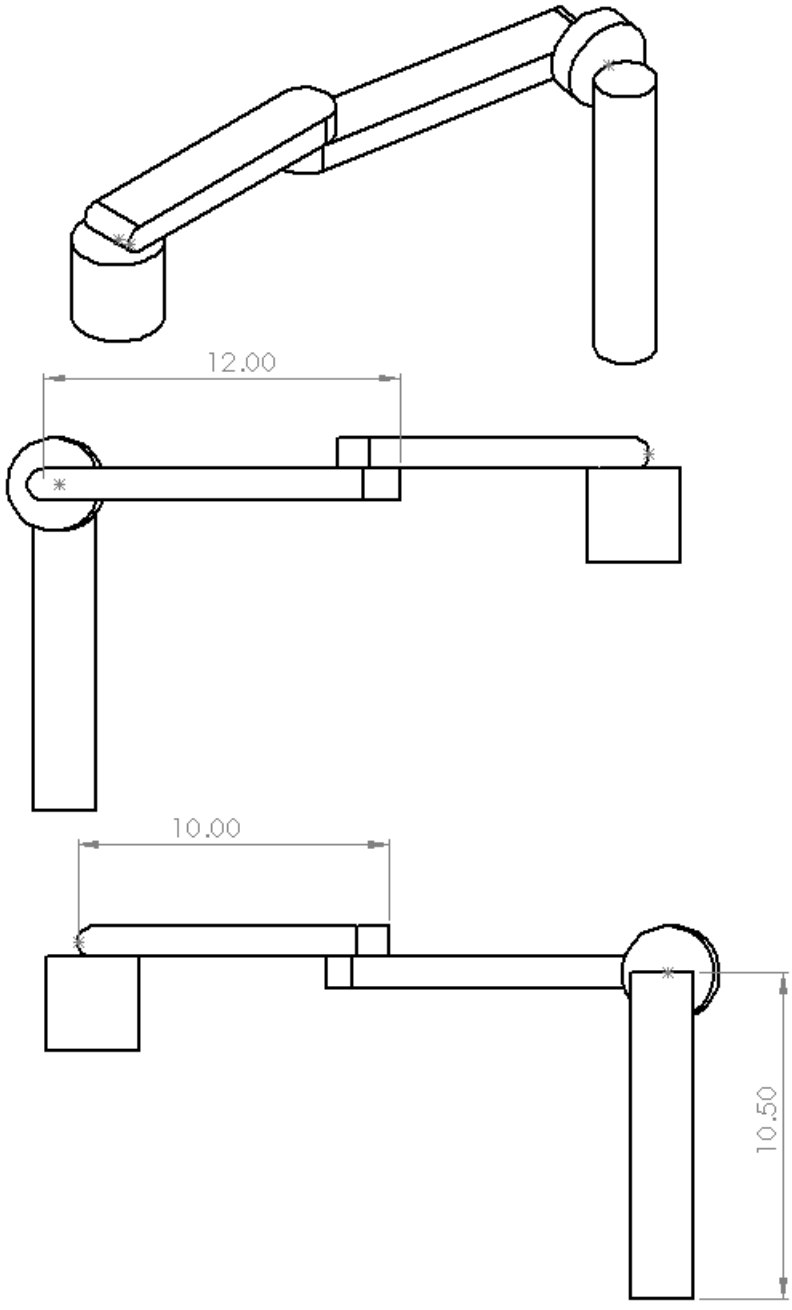


Figure 4: Derived Robot Arm Geometry

5 Question 5

5.a

5.a.i Workspace

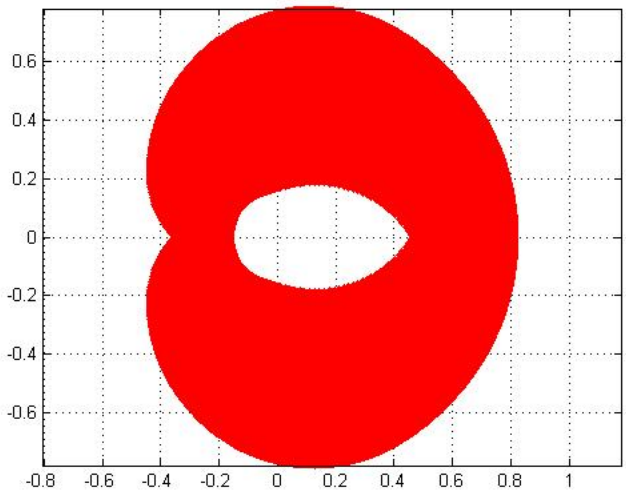


Figure 5: Workspace Plot (See Appendix A [9.4.i])

5.a.ii Configuration Space

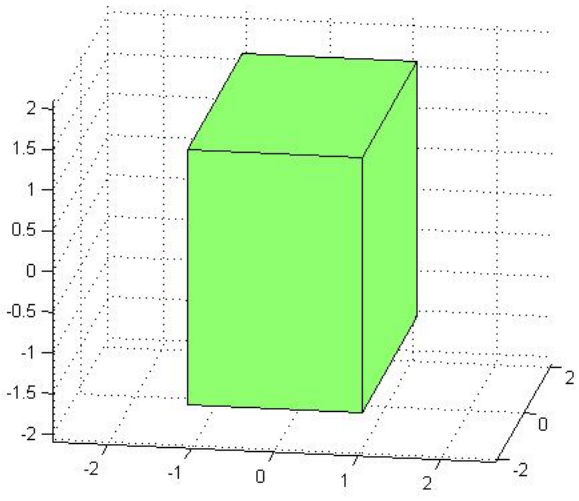


Figure 6: Configuration Space Plot (See Appendix A [9.4.ii])

5.b Singularities

Considering link L1 and link L2, take end point of link L2 as P(x,y)
$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2))$
$x' = (-L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2))\theta' - L_2 \sin(\theta_1 + \theta_2)\theta_2'$
$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$
$y' = (L_1 \cos\theta_1 - L_2 \cos(\theta_1 + \theta_2))\theta' - L_2 \cos(\theta_1 + \theta_2)\theta_2'$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{pmatrix} \cdot \begin{pmatrix} \theta_1' \\ \theta_2' \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = (J_1) \cdot \begin{pmatrix} \theta_1' \\ \theta_2' \end{pmatrix}$$

where $J_1 =$

$$\begin{pmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{pmatrix}$$

$det(J_1) = (-L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2)) \cdot (L_2 \cos(\theta_1 + \theta_2)) + (L_2 \sin(\theta_1 + \theta_2)) \cdot (L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2))$
$det(J_1) = L_1 L_2 \sin\theta_3$

Similarly we can find that
$\therefore det(J_1) = L_2 L_3 \sin\theta_3$

Finding singularities Singularities occurs when $det(J) = 0$
For J_1**:**
$det(J_1) = L_2 L_3 \sin\theta_3 = 0$
$\therefore$ **when** $\theta_3 = 0$ **or** π
However, the range of $\theta 3$ **must be between** $(-\pi/2)$ **and** $(\pi/2)$
$\therefore \theta_3 = 0$

Similarly for J_2**:**
$det(J_2) = L_1 L_2 \sin\theta_2 = 0$
$\therefore$ **when** $\theta_2 = 0$ **or** π
However, the range of $\theta 2$ **must be between** $(-2\pi/3)$ **and** $(2\pi/3)$
$\therefore \theta_2 = 0$
So for the whole system singularities occurs according to the intersection of results of A B:
i.e. for all points where $\theta_2 = 0$ **and** $\theta_3 = 0$

6 Question 6

6.a Inverse Kinematics

6.a.i Theoretical Method

In Question 3 we used forward kinematics to derive the Transformation matrix ($^0T_6$) for this robot arm with respect to our assumed "zero" coordinate system (see Figure 2).

Our aim for this question is to derive the transformation matrix specific to two separate end effector points (P_{e_a} P_{e_b}) which can be done in two ways.

One method (as mentioned in Question 3) could be to apply another transformation matrix $^wA_0$ and then continue simultaneously equating.
newline

According to our method we know that our Z_0 points are based 253 higher than the Z_w points in space and rotated about the Z axis by $\pi/2$. We need to account for this by modifying our end effector point values from:
newline

$P_{e_a} = (0, 490, 40)$
$P_{e_b} = (0, 490, 120)$
newline
To:
$P_{e_a} = (0, 490, -213)$
$P_{e_b} = (0, 490, -133)$
Since our end effector must point vertically downwards our $\sum\theta = -\pi/2$ in order to achieve this. We then simultaneously equate this with our equations for P_{e_x} and P_{e_y} with their equivalent values from our transformation matrix $^0T_6$. Substituting $\theta_1 = \theta_4 = \theta_6 = 0$ (as we do not rotate around any axis parallel to the Z_0 axis) we can solve for $\theta_2, \theta_3,$ and θ_5.
We do this once to find all the angles for Point A and again for all the angles for point B. We can then compare these angle values with the simulator jkoint angle readings in Q6 Part B.

6.a.ii Results

Our Matlab output looks as follows:
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
We can interpret this as Points A and B can each have two theoretical values for the angles (i.e. two possible configurations) to reach the desired end effector pose.

Point	θ_2	θ_3	θ_5
P_{e_a} Path 1	-56.677	13.096	-46.420
P_{e_a} Path 2	-133.579	166.903	-123.323
P_{e_b} Path 1	-45.155	13.473	-58.319
P_{e_b} Path 2	-121.681	166.526	-134.845

Table 3: Potential Angle Values for End Effector Positions A and B

6.a.iii Code Listing

See Appendix A [9.5]

6.b Simulation

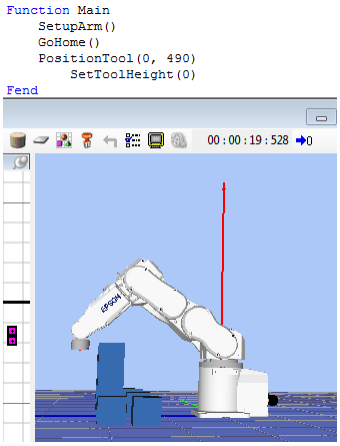


Figure 7: Simulator Point A

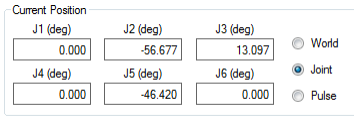


Figure 8: Simulator Point A Angles

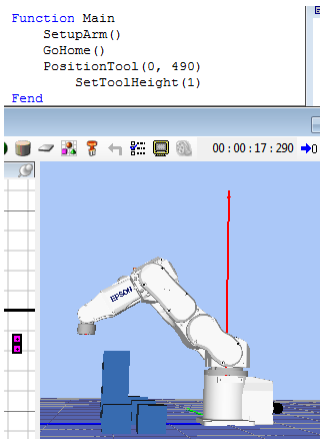


Figure 9: Simulator Point B

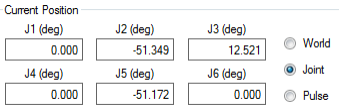


Figure 10: Simulator Point B Angles

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

See the Appendix for the Epson robotic arm code.

8 Question 8

8.a

For complete working out see the Appendix.
Brief overview of methodology:
By differentiating the position of link 2 we obtain its velocity. From there we can determine the systems Kinetic Energy.
Potential Energy can be found straight from the effect of gravity.
From there, we can find the Lagrangian $L = K - P$.
We can then determine $T1$ and $T2$ from the Lagrangian.

Final Results:
After substituting in for m_i, l_i, I_i, we obtain the following:

$$\begin{aligned} T_1 = \alpha_1[5.1625 + 3\cos\theta_2] + \alpha_2[0.7625 + 1.5\cos\theta_2] \\ -\omega_1\omega_2[3\sin\theta_2] - \omega_2^2[3\sin\theta_2] \\ +6.5g\cos\theta_1 + 1.5g\cos(\theta_1 + \theta_2) \end{aligned}$$

and

$$\begin{aligned} T_2 = \alpha_1[0.7625 + 1.5\cos\theta_2] + \alpha_2[0.2] \\ +\omega_1^2[1.5\sin\theta_2] + 1.5g\cos(\theta_1 + \theta_2) \end{aligned}$$

where

$g = 9.81$ **is the acceleration due to gravity.**
$\omega_i = \frac{d}{dt}\theta_i$ **is the angular velocity.**
$\alpha_i = \frac{d}{dt}\omega_i$ **is the angular acceleration.**

8.b

Rearranging the above equations so that α_i is the subject we obtain the following:

$$\alpha_1 = \frac{T_1 - \alpha_2[0.7625 + 1.5\cos\theta_2] + \omega_1\omega_2[3\sin\theta_2] + \omega_2^2[3\sin\theta_2] - 6.5g\cos(\theta_1+\theta_2)}{5.1625 + 3\cos\theta_2}$$

$$==> Equation1$$

$$\alpha_2 = \frac{T_2 - \alpha_1[0.7625 + 1.5\cos\theta_2] - \omega_1^2[1.5\sin\theta_2] - 1.5g\cos(\theta_1+\theta_2)}{0.2}$$

$$==> Equation2$$

Through matlab we can simultaneously solve $Equations$ 1 and 2 to obtain a description of α_i without depending on the other α_i.
See the Appendix for Question 8 for the matlab code.
This enables us to utilise the following simulink model:

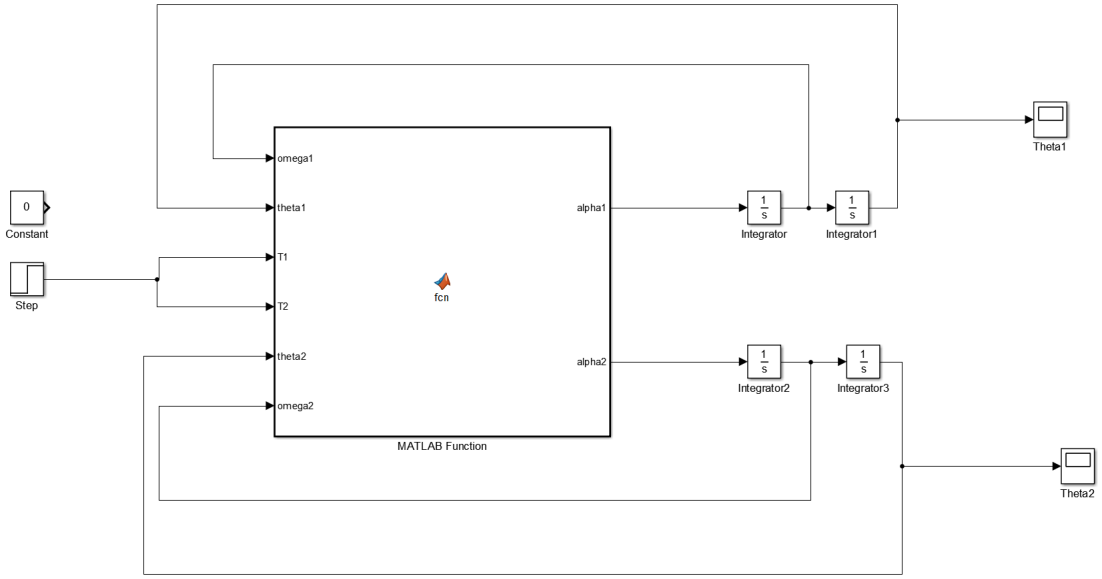


Figure 11: Simulink Model

For a constant zero input torque, we obtain for θ_1 the following output:

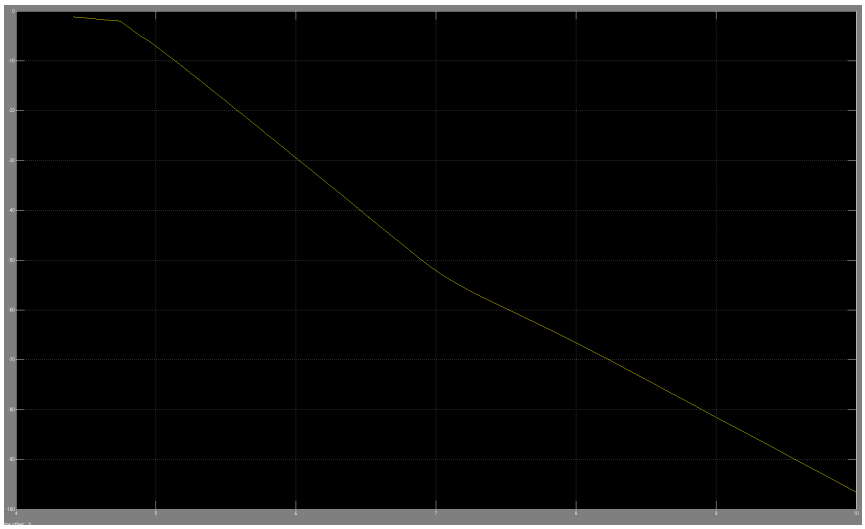


Figure 12: Theta 1 for constant zero torque

and for θ_2:

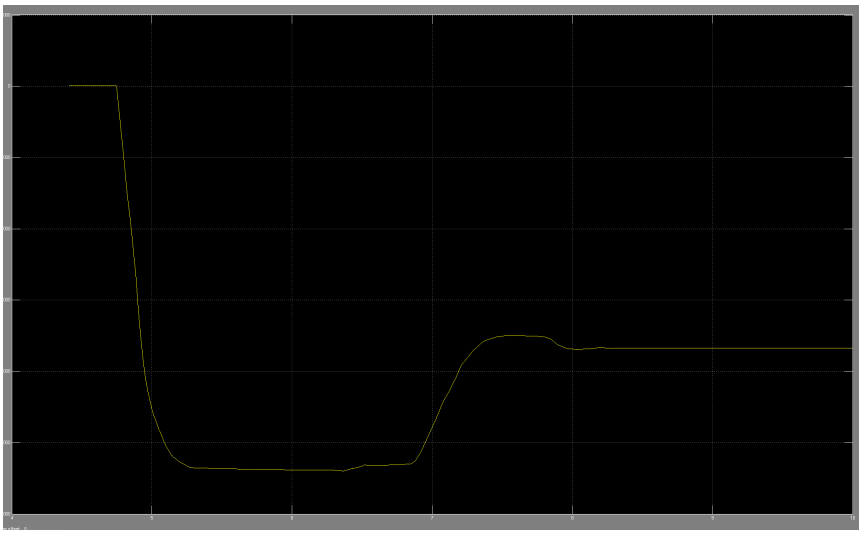


Figure 13: Theta 2 for constant zero torque

For a step input torque, we obtain for θ_1:

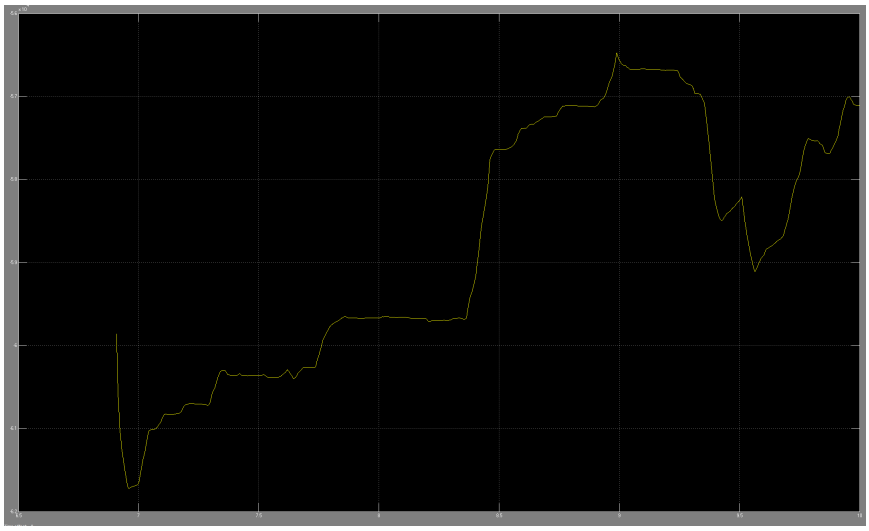


Figure 14: Theta 1 for step torque

and for θ_2:

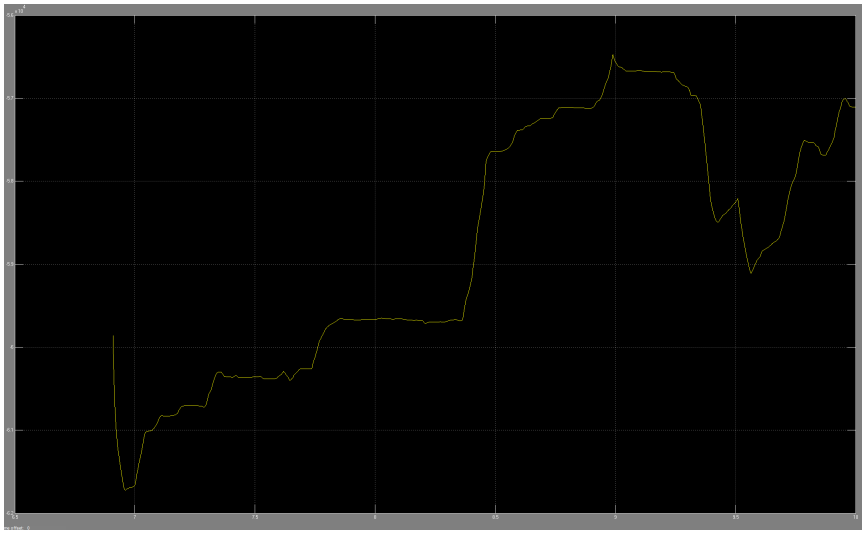


Figure 15: Theta 2 for step torque

8.c

NA

8.d

NA

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 Question 5 Code Listings

9.4.i Workspace Plot

```
%%Q5 - Determine the workspace of the system
close all;clear;clc

%Declare constants
mm = 10^-3;
L1 = 250*mm;
L2= 250*mm;
L3 = 315*mm;

%Generate an array of 100 values for each boundary conditions
theta1 = linspace(-pi/3,pi/3,100);
theta2 = linspace(-2*pi/3, 2*pi/3,100);
theta3 = linspace(-pi/2, pi/2,100);

%Create a grid matrix of the boundary values
[THETA1, THETA2, THETA3] = meshgrid(theta1,theta2,theta3);

%Using the above grid matrix, find X and Y
X = L1*cos(THETA1)+L2*cos(THETA1+THETA2)+L3*cos(THETA1+THETA2+THETA3);
Y = L1*sin(THETA1)+L2*sin(THETA1+THETA2)+L3*sin(THETA1+THETA2+THETA3);

%Take the final column of the resulting X and Y matrices
%These are the coordinates for a scatter plot
plot(X(:), Y(:), 'r.');
axis equal
grid
xlabel('x');
ylabel('y');
title('Workspace of the System');

%http://au.mathworks.com/help/fuzzy/examples/modeling-inverse-kinematics-in-a-robotic-arm.html
```

9.4.ii Configuration Space Plot

```
%%Q5 - Determine the Configuration Space of the System
close all;clear;clc

%Declare the boundaries of the angles (theta1, theta2, theta3)
theta1 = [-pi/3, pi/3];
theta2 = [-2*pi/3, 2*pi/3];
theta3 = [-pi/2, pi/2];

%Send to a function that will generate the graph
config_space_3dof(theta1, theta2, theta3);
title('Configuration Space of the System');
xlabel('-pi/3 < theta1 < pi/3');
ylabel('-2pi/3 < theta2 < 2pi/3');
zlabel('-pi/2 < theta3 < pi/2');
```

```
%%Function to plot the configuration space of a 3DOF system
%Creates a rectangular polygon with corners defined by the boundary values
%of the system

%Takes as input 3 arrays with 2 elements in each - these
%are the boundaries of the x, y, and z values respectively
function config_space_3dof(X,Y,Z)

    hold on
    grid
    axis equal
```

```
12
13      %Generate one face of the rectangular polygon, with corners described
14      %by the system boundaries.
15      cornersX = [X(2), X(2), X(1), X(1)];
16      cornersY = [Y(2), Y(1), Y(1), Y(2)];
17      cornersZ = [Z(1), Z(1), Z(1), Z(1)];
18      fill3(cornersX, cornersY, cornersZ,1);
19
20      %Generate the rest of the faces, building the polygon
21      cornersX = [X(2), X(2), X(1), X(1)];
22      cornersY = [Y(2), Y(1), Y(1), Y(2)];
23      cornersZ = [Z(2), Z(2), Z(2), Z(2)];
24      fill3(cornersX, cornersY, cornersZ,1);
25
26      cornersX = [X(2), X(2), X(2), X(2)];
27      cornersY = [Y(2), Y(1), Y(1), Y(2)];
28      cornersZ = [Z(1), Z(1), Z(2), Z(2)];
29      fill3(cornersX, cornersY, cornersZ,1);
30
31      cornersX = [X(1), X(1), X(1), X(1)];
32      cornersY = [Y(2), Y(1), Y(1), Y(2)];
33      cornersZ = [Z(1), Z(1), Z(2), Z(2)];
34      fill3(cornersX, cornersY, cornersZ,1);
35
36      cornersX = [X(1), X(2), X(2), X(1)];
37      cornersY = [Y(2), Y(2), Y(2), Y(2)];
38      cornersZ = [Z(1), Z(1), Z(2), Z(2)];
39      fill3(cornersX, cornersY, cornersZ,1);
40
41      cornersX = [X(1), X(2), X(2), X(1)];
42      cornersY = [Y(1), Y(1), Y(1), Y(1)];
43      cornersZ = [Z(1), Z(1), Z(2), Z(2)];
44      fill3(cornersX, cornersY, cornersZ,1);
45
46      hold off
47  end
```

9.5 Question 6 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
%%initializing thetas 1, 4, and 6 as we know the robot cannot turn about
%%those angles for the specified end effector position
th(1) = 0;
%th(2) = 0;
% th(3) = 0;
th(4) = 0;
% th(5) = 0;
th(6) = 0;

d(1) = 67;
d(2) = 0;
d(3) = 0;
d(4) = 0;
d(5) = 0;
d(6) = 0;

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

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

% T11 = T_06(1, 1);
% T12 = T_06(1, 2);
% T13 = T_06(1, 3);
% T14 = T_06(1, 4);
% T21 = T_06(2, 1);
% T22 = T_06(2, 2);
```

```
% T23 = T_06(2, 3);
% T24 = T_06(2, 4);
% T31 = T_06(3, 1);
% T32 = T_06(3, 2);
% T33 = T_06(3, 3);
% T34 = T_06(3, 4);
% T41 = T_06(4, 1);
% T42 = T_06(4, 2);
% T43 = T_06(4, 3);
% T44 = T_06(4, 4);

%%rectifies against unrealistic solutions.
%%included this line prior to running the slove function
T = vpa(T_06);

%%implement the solve function
%%solve simultaneously
%%Theta2 + 3 + 4 = -90 degrees  %%-->Equation 1
%%Px = 490                      %%-->Equation 2
%%Pz = -213                     %%-->Equation 3
%%Solve for th2, 3, 5 values for block 1 (designated by a) and block 2
%%(designated by b)
sa = solve(th(2) + th(3) + th(5) == -pi/2, T_06(1,4) == 490, T_06(3,4) == -213, th(2),th(3),th(5));
sb = solve(th(2) + th(3) + th(5) == -pi/2, T_06(1,4) == 490, T_06(3,4) == -133, th(2),th(3),th(5));
%%convert to degrees and correct theta
sa2 = vpa(radtodeg(sa.th2)) - 90
sa3 = vpa(radtodeg(sa.th3)) + 90
sa5 = vpa(radtodeg(sa.th5))
sb2 = vpa(radtodeg(sb.th2)) - 90
sb3 = vpa(radtodeg(sb.th3)) + 90
sb5 = vpa(radtodeg(sb.th5))
```

9.6 Question 7 Code Listings

9.6.i Epson Arm Code

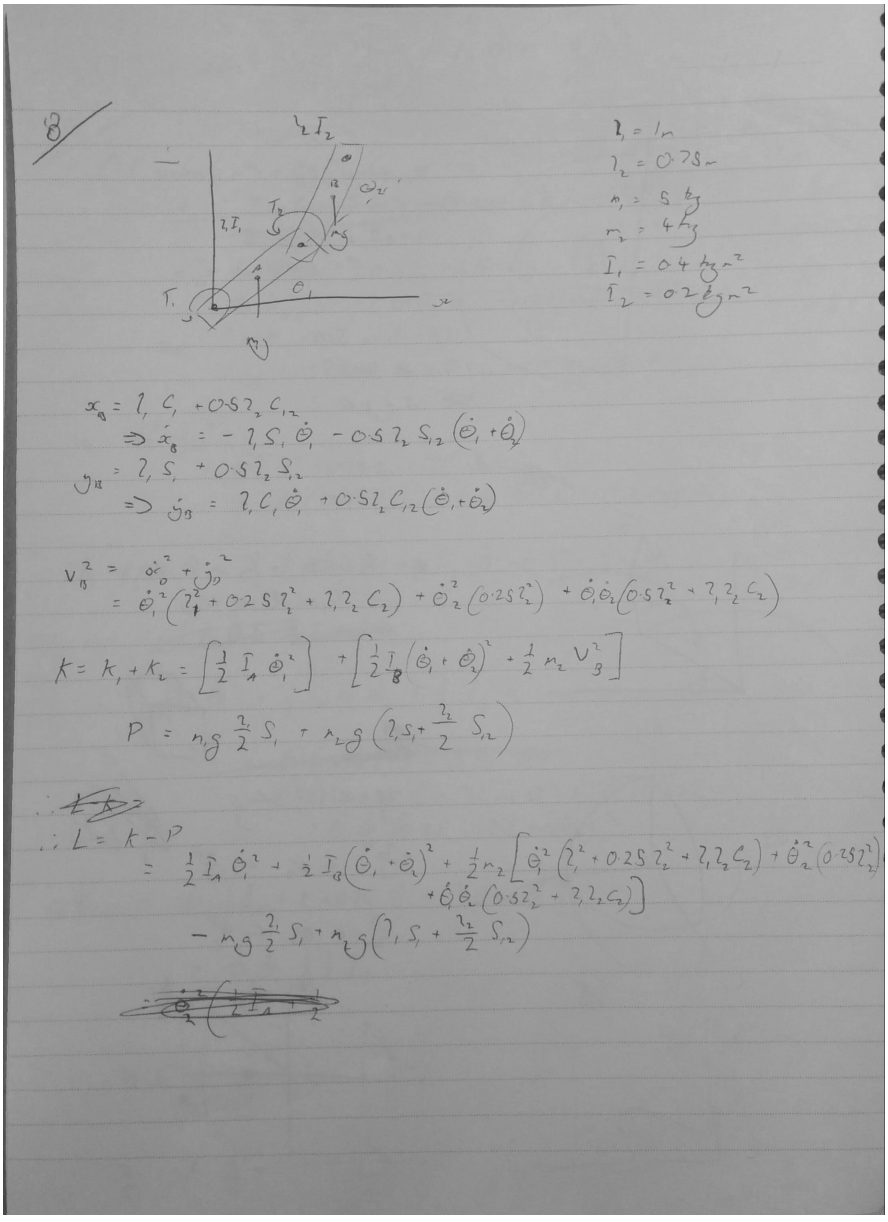
```
''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Wrapper Functions...                                '
' Don 't change anything in this section...'
''''''''''''''''''''''''''''''''''''''''''''''''''''''
#define MoveBlockHeight (40)
#define BlockWidth (80)
Function SetupArm()
    Motor On
    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                          '
''''''''''''''''''''''''''''''''''''''''''''''''''''''

```

```
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
```

9.7 Question 8 Working and Code Listings

9.7.i Part a Working



$T_i = \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i}$ $\quad$ $F_i = \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{x}_i}\right) - \frac{\partial L}{\partial x_i}$

$K = \frac{1}{2} I_A \dot{\theta}_1^2 + \left[\frac{1}{2} I_B(\dot{\theta}_1 + \dot{\theta}_2)^2 + \frac{1}{2} m_2 V_0^2\right]$

$= \frac{1}{2} I_A \dot{\theta}_1^2 + \frac{1}{2} I_B \dot{\theta}_1^2 + I_B \dot{\theta}_1 \dot{\theta}_2 + \frac{1}{2} I_B \dot{\theta}_2^2$

$+ \frac{1}{2} m_2 \left[\dot{\theta}_1^2 (l_1^2 + 0.25 l_2^2 + l_1 l_2 C_2) + \dot{\theta}_2^2 (0.25 l_2^2) + \dot{\theta}_1 \dot{\theta}_2 (0.5 l_2^2 + l_1 l_2 C_2)\right]$

~~$= \dot{\theta}_1^2 \left(\frac{1}{2} I_A + \frac{1}{2} I_B\right) + \dot{\theta}_1 \dot{\theta}_2 (I_B) + \dot{\theta}_2^2 \left(\frac{1}{2} I_B\right)$~~

~~$+ \dot{\theta}_1^2 [$~~

$= \dot{\theta}_1^2 \left[\frac{1}{2} I_A + \frac{1}{2} I_B + \frac{1}{2} m_2 (l_1^2 + 0.25 l_2^2 + l_1 l_2 C_2)\right]$

$+ \dot{\theta}_1 \dot{\theta}_2 \left[I_B + \frac{1}{2} m_2 (0.5 l_2^2 + l_1 l_2 C_2)\right]$

$+ \dot{\theta}_2^2 \left[\frac{1}{2} I_B\right]$

$P = m_1 g \frac{l_1}{2} S_1 + m_2 g \left(l_1 S_1 + \frac{l_2}{2} S_{12}\right)$

$L = K - P$

$= \dot{\theta}_1^2 \left[\frac{1}{2} I_A + \frac{1}{2} I_B + \frac{1}{2} m_2 (l_1^2 + 0.25 l_2^2 + l_1 l_2 C_2)\right]$

$+ \dot{\theta}_1 \dot{\theta}_2 \left[I_B + \frac{1}{2} m_2 (0.5 l_2^2 + l_1 l_2 C_2)\right]$

$+ \dot{\theta}_2^2 \left[\frac{1}{2} I_B\right] - m_1 g \frac{l_1}{2} S_1 - m_2 g \left(l_1 S_1 + \frac{l_2}{2} S_{12}\right)$

$$T_1 = \frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{\theta}_1} - \frac{\partial L}{\partial \theta_1}$$

$$\frac{\partial L}{\partial \theta_1} = -m_1 g \frac{l_1}{2} C_1 - m_2 g l_1 C_1 - m_2 g \frac{l_2}{2} C_{12}$$

$$\frac{\partial L}{\partial \dot{\theta}_1} = \dot{\theta}_1\left[I_A + I_B + m_2\left(l_1^2 + 0.25 l_2^2 + l_1 l_2 C_2\right)\right] + \dot{\theta}_2\left[I_B + \tfrac{1}{2} m_2\left(0.5 l_2^2 + l_1 l_2 C_2\right)\right]$$

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{\theta}_1} = \ddot{\theta}_1\left[I_A + I_B + m_2\left(l_1^2 + 0.25 l_2^2 + l_1 l_2 C_2\right)\right] - m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 + \ddot{\theta}_2\left[I_B + \tfrac{1}{2} m_2\left(0.5 l_2^2 + l_1 l_2 C_2\right)\right] - m_2 l_1 l_2 S_2 \dot{\theta}_2^2$$

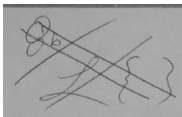
$$\therefore T_1 = \ddot{\theta}_1\left[I_A + I_B + m_2\left(l_1^2 + 0.25 l_2^2 + l_1 l_2 C_2\right)\right] + \ddot{\theta}_2\left[I_B + \tfrac{1}{2} m_2\left(0.5 l_2^2 + l_1 l_2 C_2\right)\right] - m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 l_1 l_2 S_2 \dot{\theta}_2^2 + m_1 g \frac{l_1}{2} C_1 + m_2 g l_1 C_1 + m_2 g \frac{l_2}{2} C_{12}$$

$$\frac{\partial L}{\partial \theta_2} = -\frac{1}{2} m_2 l_1 l_2 S_2 \dot{\theta}_1^2 - \frac{1}{2} m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 g \frac{l_2}{2} C_{12}$$

$$\frac{\partial L}{\partial \dot{\theta}_2} = \dot{\theta}_1 \left[I_3 + \frac{1}{2} m_2 \left(OSL^2 - l_1 l_2 C_2 \right) \right] + \dot{\theta}_2 \left[I_3 \right]$$

$$\frac{\partial}{\partial t} \frac{\partial L}{\partial \dot{\theta}_2} = \ddot{\theta}_1 \left[I_3 + \frac{1}{2} m_2 \left(OSL^2 + l_1 l_2 C_2 \right) \right] - \frac{1}{2} m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 + \ddot{\theta}_2 \left[I_3 \right]$$

$$T_2 = \ddot{\theta}_1 \left[I_3 + \frac{1}{2} m_2 \left(OSL^2 + l_1 l_2 C_2 \right) \right] + \ddot{\theta}_2 \left[I_3 \right] - \frac{1}{2} m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 + \frac{1}{2} m_2 l_1 l_2 S_2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 g \frac{l_2}{2} C_{12}$$



$$T_1 = \ddot{\theta}_1 \left[I_1 + I_{1s} + m_2 \left(l_1^2 + 0.25 l_2^2 + l_1 l_2 c_2 \right) \right]$$
$$+ \ddot{\theta}_2 \left[I_{1s} + \tfrac{1}{2} m_2 \left(0.5 l_2^2 + l_1 l_2 c_2 \right) \right]$$
$$+ -m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 l_1 l_2 s_2 \dot{\theta}_2^2$$
$$+ m_1 g \tfrac{l_1}{2} c_1 + m_2 g l_1 c_1 + m_2 g \tfrac{l_2}{2} c_{12}$$

$$T_2 = \ddot{\theta}_1 \left[I_{1s} + \tfrac{1}{2} m_2 \left(0.5 l_2^2 + l_1 l_2 c_2 \right) \right] + \ddot{\theta}_2 \left[I_{1s} \right]$$
$$- \tfrac{1}{2} m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 + \tfrac{1}{2} m_2 l_1 l_2 s_2 \dot{\theta}_1^2$$
$$+ \tfrac{1}{2} m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 g \tfrac{l_2}{2} c_{12}$$

$$\therefore T_1 = \ddot{\theta}_1 \left[5.1625 + 3 \cos \theta_2 \right] + \ddot{\theta}_2 \left[0.7625 + 1.5 \cos \theta_2 \right]$$
$$- \dot{\theta}_1 \dot{\theta}_2 \left[3 \sin \theta_2 \right] - \dot{\theta}_2^2 \left[3 \sin \theta_2 \right]$$
$$+ 6.5 g \cos \theta_1 + 1.5 g \cos (\theta_1 + \theta_2)$$

$$\therefore T_2 = \ddot{\theta}_1 \left[0.7625 + 1.5 \cos \theta_2 \right] + \ddot{\theta}_2 \left[0.2 \right]$$
$$- \dot{\theta}_1 \dot{\theta}_2 \left[1.5 \sin \theta_2 \right] + \dot{\theta}_1^2 \left[1.5 \sin \theta_2 \right] + 1.5 g \cos (\theta_1 + \theta_2)$$

where g = acceleration due to gravity
$\approx 9.81 \, m s^{-2}$

8b

$$T_1 = \ddot{\theta}_1\left[5.1625 + 3\cos\theta_2\right] + \ddot{\theta}_2\left[0.7625 + 1.5\cos\theta_2\right]$$
$$- \dot{\theta}_1\dot{\theta}_2\left[3\sin\theta_2\right] - \dot{\theta}_2^2\left[3\sin\theta_2\right]$$
$$+ 6.5g\cos\theta_1 + 1.5g\cos(\theta_1+\theta_2)$$

$$\therefore \ddot{\theta}_1 = \frac{\left(T_1 - \ddot{\theta}_2\left[0.7625 + 1.5\cos\theta_2\right] + \dot{\theta}_1\dot{\theta}_2\left[3\sin\theta_2\right] + \dot{\theta}_2^2\left[3\sin\theta_2\right] - 6.5g\cos\theta_1 + 1.5g\cos(\theta_1+\theta_2)\right)}{5.1625 + 3\cos\theta_2}$$

↳ (1)

$$T_2 = \ddot{\theta}_1\left[0.7625 + 1.5\cos\theta_2\right] + \ddot{\theta}_2\left[0.2\right]$$
$$+ \dot{\theta}_1^2\left[1.5\sin\theta_2\right] + 1.5g\cos(\theta_1+\theta_2)$$

$$\therefore \ddot{\theta}_2 = \frac{\left(T_2 - \ddot{\theta}_1\left[0.7625 + 1.5\cos\theta_2\right] - \dot{\theta}_1^2\left[1.5\sin\theta_2\right] - 1.5g\cos(\theta_1+\theta_2)\right)}{0.2}$$

↳ (2)

9.7.ii Part b Code

```
%%Code to simultaneously solve for Angular acceleration 1 and 2, without reference to the other

close all;clear;clc;

% Torque 1&2
syms T1
syms T2
% Angles 1&2
syms theta1
syms theta2
% Angular Velocity 1&2
syms omega1
syms omega2
% Angular Acceleration 1&2
syms alpha1
syms alpha2

% Acceleration due to Gravity
g = 9.8;

% alpha1 = (T1 - alpha2*(0.7625 + 1.5*cos(theta2)) + omega1*omega2*(3*sin(theta2)) + omega2*3*sin(
    ↪ theta2) - 6.5*g*cos(theta1) + 1.5*g*cos(theta1 + theta2))/(5.1625 + 3*cos(theta2));
% alpha2 = (T2 - alpha1*(0.7625 + 1.5*cos(theta2) - omega1^2*(1.5*sin(theta2))) - 1.5*g*cos(theta1 +
    ↪  theta2))/0.2;
s = solve(alpha1 == (T1 - alpha2*(0.7625 + 1.5*cos(theta2)) + omega1*omega2*(3*sin(theta2)) + omega2
    ↪ *3*sin(theta2) - 6.5*g*cos(theta1) + 1.5*g*cos(theta1 + theta2))/(5.1625 + 3*cos(theta2)),
    ↪ alpha2 == (T2 - alpha1*(0.7625 + 1.5*cos(theta2) - omega1^2*(1.5*sin(theta2))) - 1.5*g*cos(
    ↪ theta1 + theta2))/0.2 ,alpha1,alpha2);

alpha1 = s.alpha1
alpha2 = s.alpha2

%% Function code for simulink - uses the equations found above to determine angular acceleration

% function [alpha1,alpha2] = fcn(omega1,theta1,T1,T2,theta2,omega2)
%#codegen

% alpha1 = (2*(3200*T1 - 12200*T2 + 226611*cos(theta1 + theta2) - 204048*cos(theta1) - 24000*T2*cos(
    ↪ theta2) + 9600*omega2*sin(theta2) + 353160*cos(theta1 + theta2)*cos(theta2) + 9600*omega1*
    ↪ omega2*sin(theta2)))/(5*(7320*omega1^2*sin(theta2) - 10800*cos(theta2) - 14400*cos(theta2)^2
    ↪ + 14400*omega1^2*cos(theta2)*sin(theta2) + 2887));
% alpha2 = -(2*(12200*T1 - 82600*T2 + 1394982*cos(theta1 + theta2) - 777933*cos(theta1) - 1530360*
    ↪ cos(theta1)*cos(theta2) + 24000*T1*cos(theta2) - 48000*T2*cos(theta2) + 36600*omega2*sin(
    ↪ theta2) + 1059480*cos(theta1 + theta2)*cos(theta2) + 36600*omega1*omega2*sin(theta2) - 24000*
    ↪ T1*omega1^2*sin(theta2) + 72000*omega2*cos(theta2)*sin(theta2) - 353160*omega1^2*cos(theta1 +
    ↪  theta2)*sin(theta2) - 72000*omega1^2*omega2*sin(theta2)^2 - 72000*omega1^3*omega2*sin(theta2
    ↪ )^2 + 1530360*omega1^2*cos(theta1)*sin(theta2) + 72000*omega1*omega2*cos(theta2)*sin(theta2))
    ↪ )/(5*(7320*omega1^2*sin(theta2) - 10800*cos(theta2) - 14400*cos(theta2)^2 + 14400*omega1^2*
    ↪ cos(theta2)*sin(theta2) + 2887));
```