A report on

APPLE FRUIT PICKER ROBOT

ME G511: MECHANISMS & ROBOTICS

Project work carried out by

ADITYA MAJALI (Group Leader)	2017B5A41031G
DEVANSHU WAKHALE	2017A4PS0293G
DEVESH DIMBLE	2017B4A40052G
VENUGOPALAN IYENGAR	2017B5A40765G

Submitted in partial fulfillment of
ME G511 (Mechanisms & Robotics) Course

Under the supervision of
Dr. Pravin M. Singru

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)**

November 2020

Table of Contents

Chapter No.	**Title of Chapter**	**Page No.**
Chapter 1	Idea of the project	3
Chapter 2	Design of Layout / Work cell	4
Chapter 3	Direct Kinematics Solution	13
Chapter 4	Inverse Kinematics Solution	17
Chapter 5	Jacobian and velocity analysis	21
Chapter 6	Dynamic Analysis	26
Chapter 7	Final Conclusion	27
Chapter 8	References	28
Chapter 9	Appendices	29

Contribution of each member to the project

Specific contributions:
Aditya Majali:
AutoCAD - Design Template for the overall design of the Arm and Arm length specifications

Devesh Dimble:
SOLIDWORKS - Design of the whole arm using the design template from AutoCAD

Venugopalan Iyengar:
Blender 3D - Animation and Simulation of the Arm on a Vehicle in a Tree Environment

Devanshu Wakhale:
AutoCAD - Design Template of the overall design of the Arm and Arm length specifications

Contributions by all:
Forward Kinematics, Inverse Kinematics, Jacobian and Velocity analysis, Dynamics Analysis, MATLAB Codes

Chapter 1

Idea of the project

"APPLE FRUIT PICKER"

1.1 Problem statement:

During the fruit harvest season, costs incurred due to the hiring of labor can be reduced by employing a semi-autonomous robot that can detect and pluck the ripe fruits at the right time and drop them in a carton for further processing.

1.2 Brief Description:

The increasing amounts of investments in the field of agriculture have generated the need to reduce the costs incurred due to the hiring of labor to work on the farmland. This need for labor is more frequent in plantations that grow fruit-bearing trees, especially during the harvest season as these generally spread across many acres. To facilitate the task of plucking ripe fruits at the right time, a robot can be used that can work round the clock. The robot will be responsible for picking the apples from the trees accurately and dropping them in the collecting carton or bag.

1.3 Methodology:

- The robot will determine the existence of a ripe apple fruit based on deep learning models using an image recognition algorithm.
- To measure the distance to the apple, we can use an ultrasonic sensor and feed the distance measurement to the robotic arm so that it can be moved accordingly.
- This bot can be trained for other fruits of similar size and tree characteristics like oranges, etc.

Chapter 2

Design of layout / work cell

2.1 Type of Motion:

The robot which will be used for the task will have parameters R-P-R-R-R-R which is a six degree of freedom robot. The robot can be seen as divided into two motions, one is the movement of the robot from the home position to the position of the apple (fruit), which will use the R-P-R-R sequence of operations. The second motion (Gripper) will utilize the last R-R for pitch and roll motions.

The robot will pluck the fruit and then palletize the collected fruits into a neat tray for further transport or processing.

2.2 Motion sequence:

1. The robot starts at the **home position.**
2. The robot rotates towards the tree (by varying $\theta 1$) so that the camera faces the tree line.
3. The **camera** on the robotic arm detects Apple's **x, y location**.
4. **The gripper** is aligned with the **detected location.** This is done by calculating all the parameters of the arm, from the home position to the apple's x, y position.
5. The depth/distance of the apple from the gripper is detected using the **ultrasonic sensor.**
6. The arm moves forward towards the **apple.**
7. The gripper grasps the apple gently, **then rotates and retracts with the apple.** This is the **plucking** motion.
8. The arm then returns to the **home position.**
9. The robot then moves towards the **pallet and places** the apple on it.
10. It **again returns to the home position**.

2.3 Sensors:

The sensors which will be used are:

1. **Camera(1080 HDR MIPI Camera MCAM400)** for detection of the ripe apple on the tree. (Click here)
2. **Ultrasonic sensor** (U300.R50-GP1J.72N) can be used for distance measurement between the gripper and the apple. The sensor will be placed inside the gripper. The sensor has a range between 0mm and 500mm and can be used effectively for the measurement of depth of apple from the gripper.

2.4 Workspace Sketch:

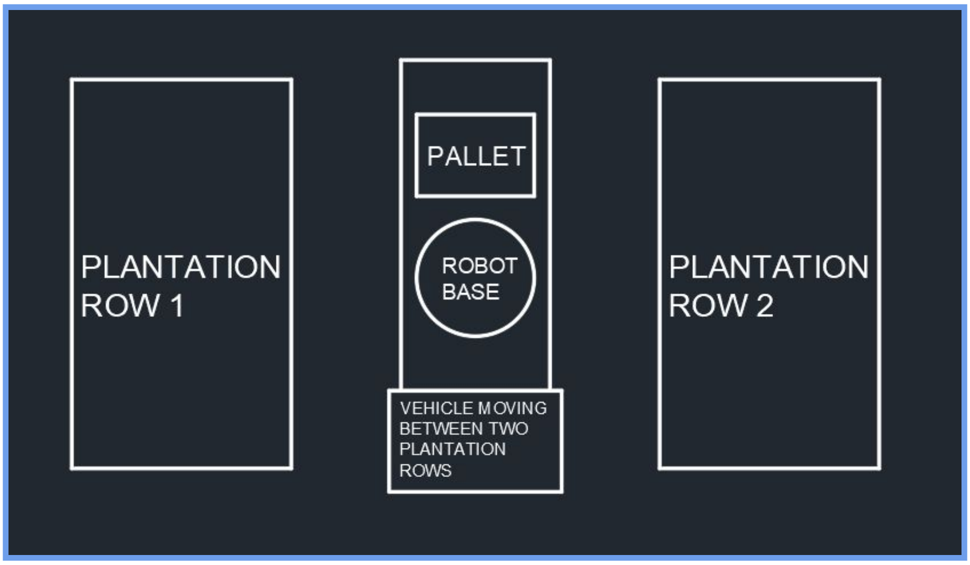


Figure 1: Top view of the layout.

Generally, trees are planted in rows in a plantation with enough space between two rows for a mini-truck / tractor to move. The robot will be mounted on the back of a mini-truck / tractor that will move it around the plantation. The robot will scan one tree at a time and will try to detect a ripe apple while the truck remains stationary in front of the tree. The detected apple will be plucked and palletized. After all the apples from one tree have been plucked, the robot rotates and starts scanning the tree in the opposite row. Hence, two trees can be covered at one location.

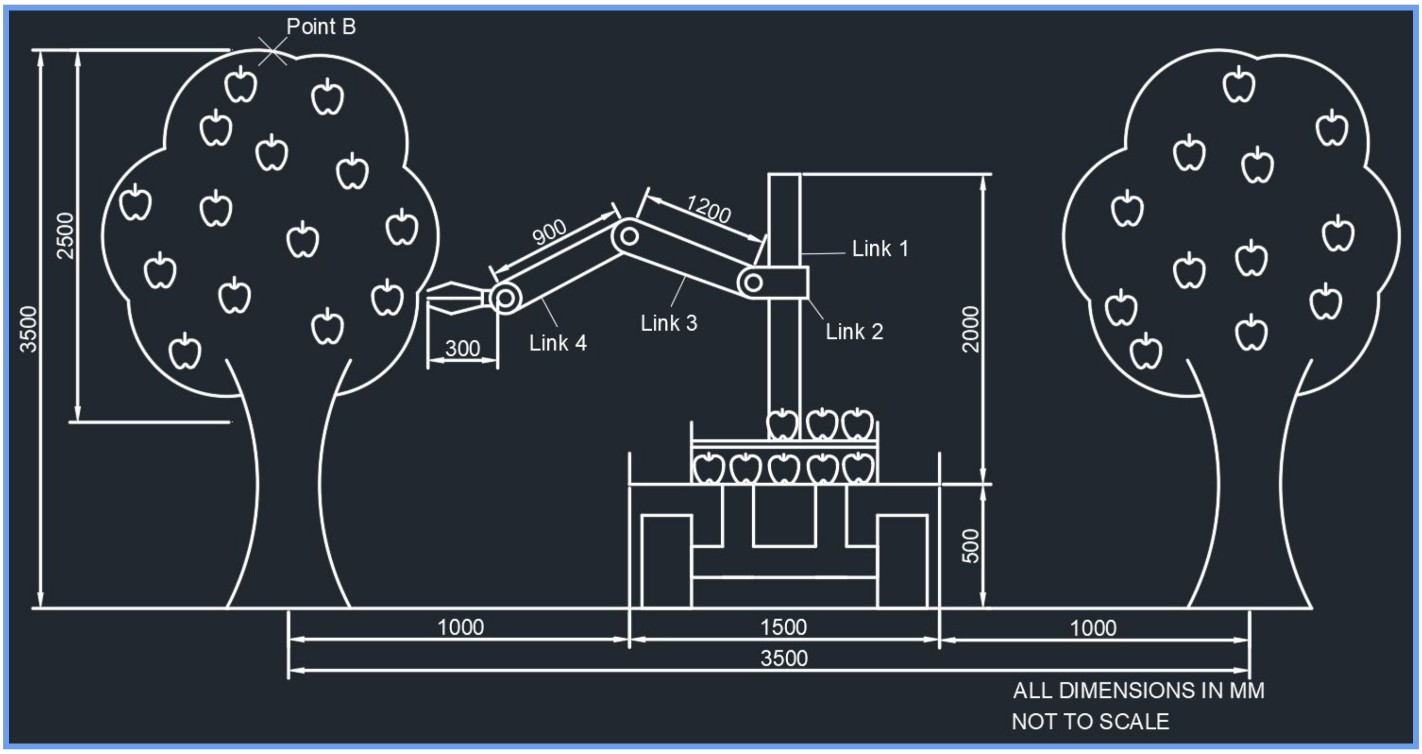


Figure 2(a): Ground view of the layout with dimensions (as seen from behind the vehicle).

Figure 2(b): Ground view of the layout with dimensions (as seen from behind the vehicle).

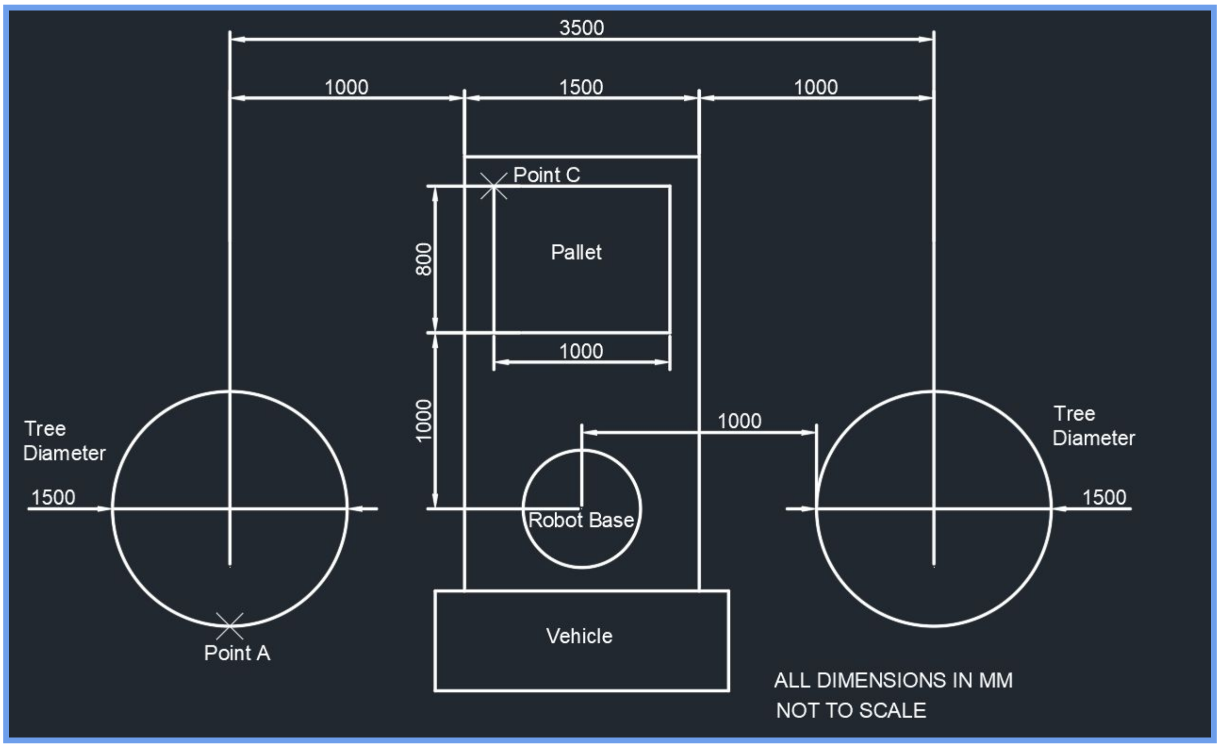


Figure 3(a): Top view of the layout with dimensions.

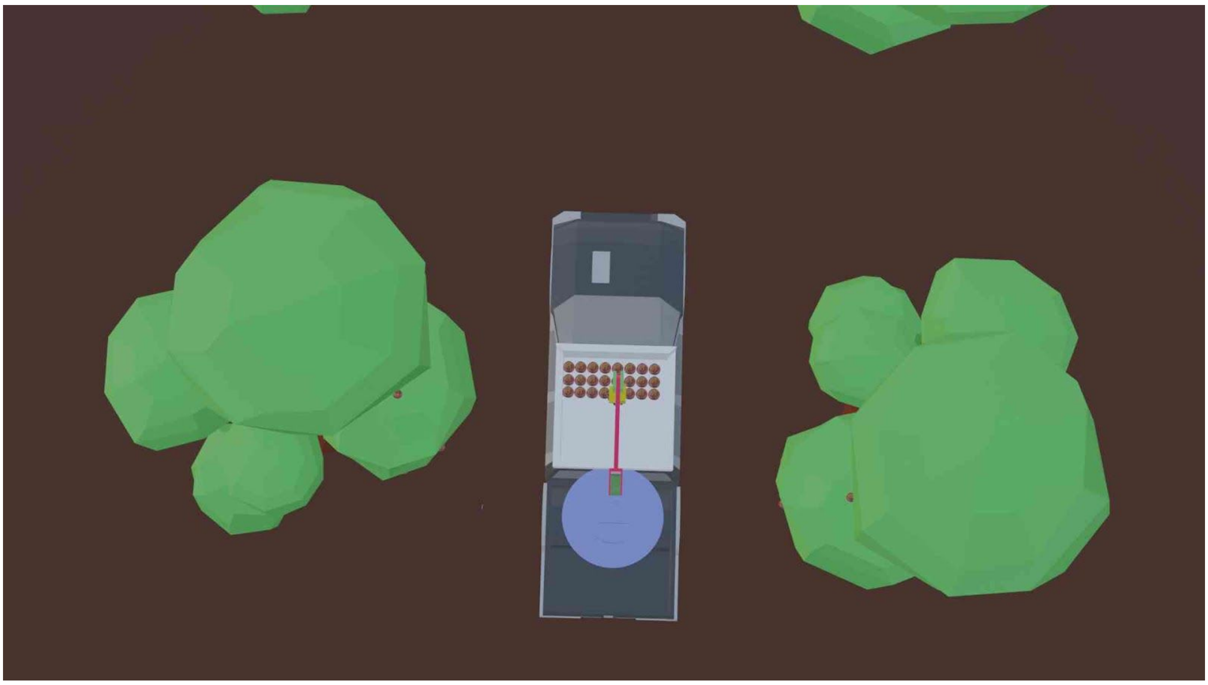


Figure 3(b): Top view of the layout.

2.5 Link length calculations:

Before calculating the link lengths, it is necessary to find the maximum and minimum extensions that the arm of the robot is expected to reach.

As it can be seen from the Figure 3 above, ***the minimum distance from the robot base in the workspace is 1000 mm***.

The maximum distance, say X, is calculated by using the Pythagoras theorem, for two possible locations in the workspace:

A. For the first location, the Point A (see figure 3) on the circumference of the tree is chosen (Figure 4 below which depicts the top view of the layout).

From pythagoras theorem,

$$x = \sqrt{1750^2 + 750^2} = 1903.94mm$$

Hence the maximum distance is ≈ 1900 mm.

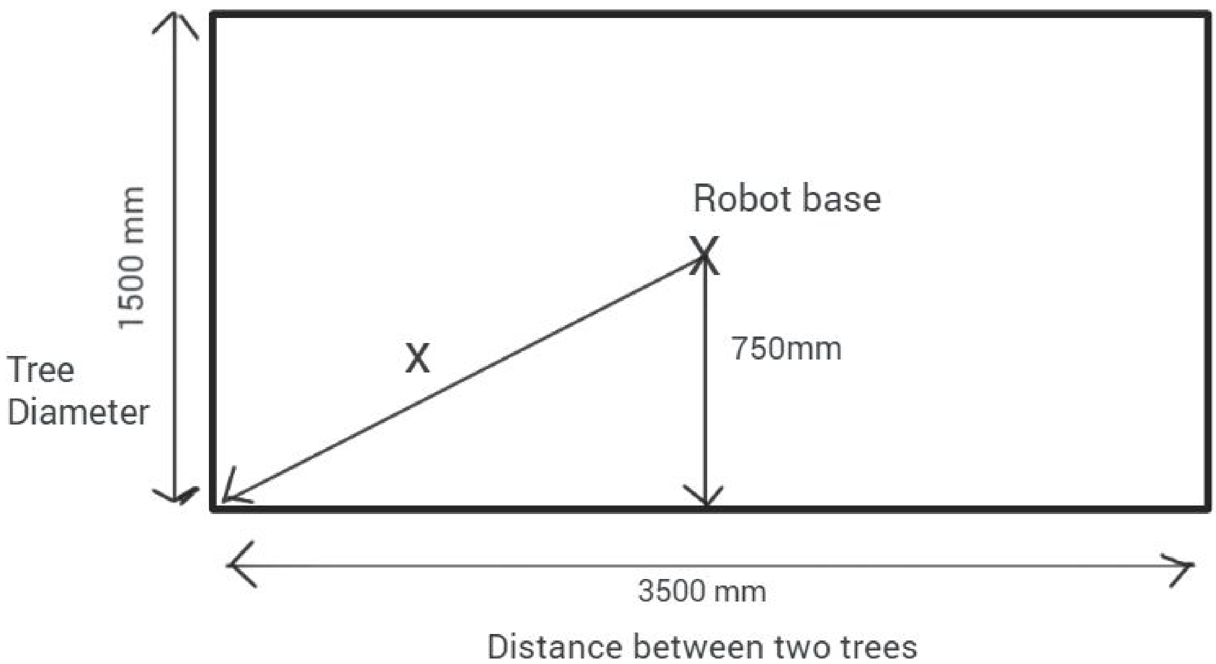


Figure 4: Calculating maximum extension of arm for Point A

B. For the second location, the Point B (see figure 2) on the top of the tree has been chosen. Firstly, to reach this position, ***the Link 1 should be 2000 mm in height***, so that the arm can extend further. (Figure 5 below, which shows the front view of the layout)

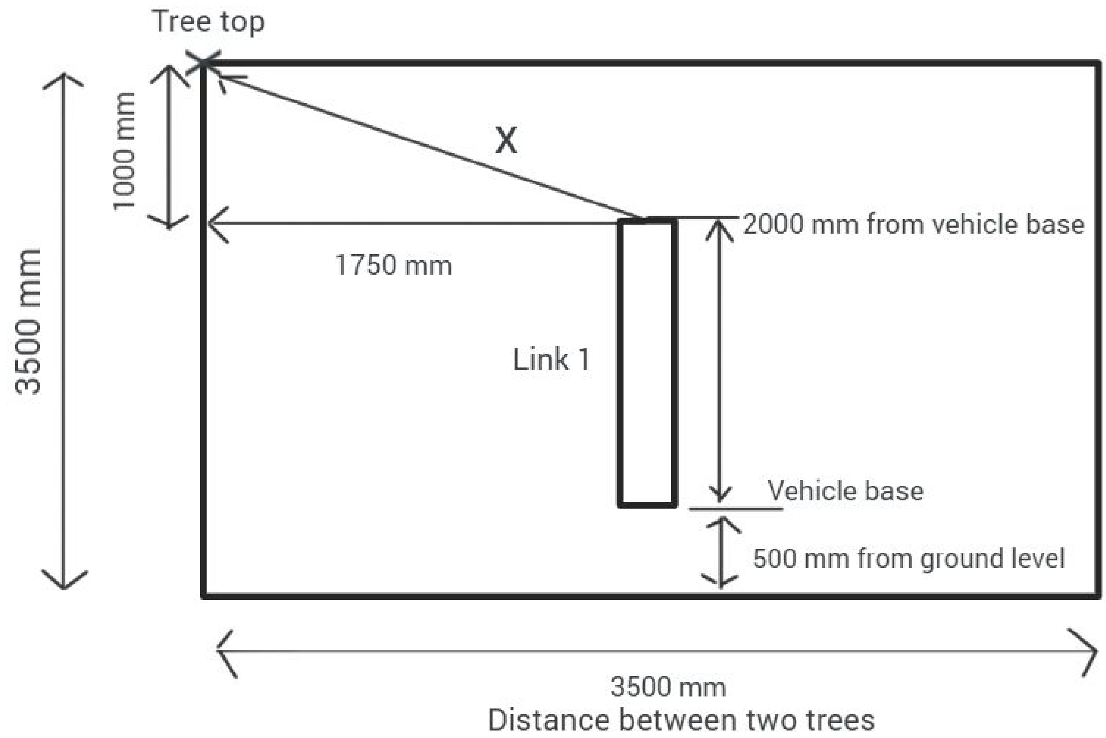


Figure 5: Calculating maximum extension of arm for Point B.

For finding the distance to the top of tree, from pythagoras theorem,

$$x = \sqrt{1750^2 + 1000^2} = 2015.56mm$$

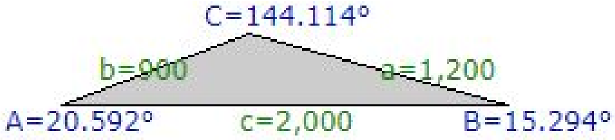
Hence, in this case, the maximum distance is ≈ 2015 mm.

To be on the safer side, we can say that the maximum extension for ***the links 3 and 4 combined will be 2000 mm***.

Now that the maximum and the minimum distances are known, we can assume that the length of ***Link 3 = 1200 mm***, and ***Link 4 = 900 mm*** (as taking the length of link 4 = 800 mm would not yield a feasible solution for the triangle thus formed).

Using properties of triangle, it can be shown that these link lengths are able to reach the maximum and the minimum distances:-

Link 3 => side a = 1200 mm Link 4 => side b = 900 mm



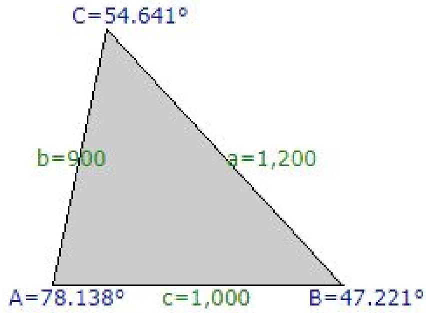


Fig. 6 : Maximum extension **Fig. 7 : Minimum extension**

2.6 <u>Robot Sketch</u>:

A. This is the configuration we want to achieve using the robot in **home position**:

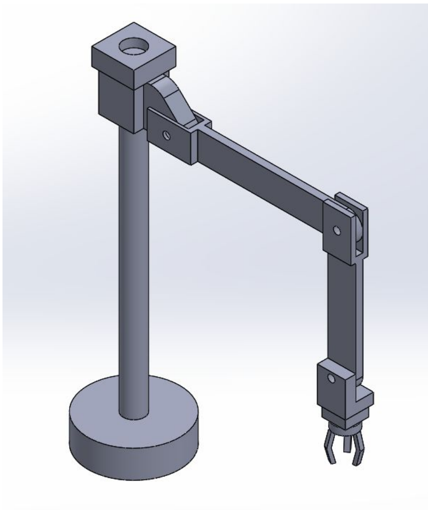
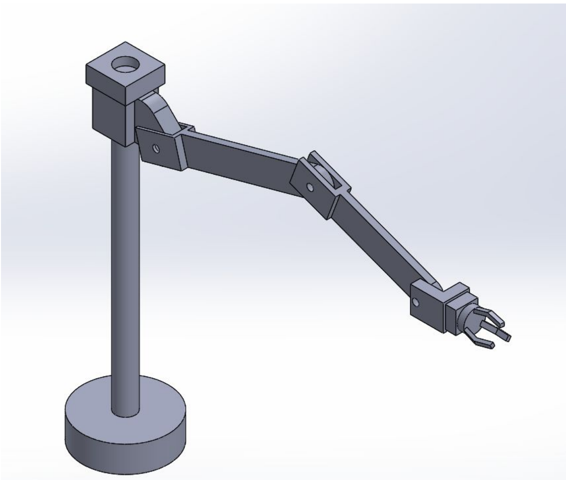


Fig.8 : Home position

B. The robot will look like this when it is plucking the apple:



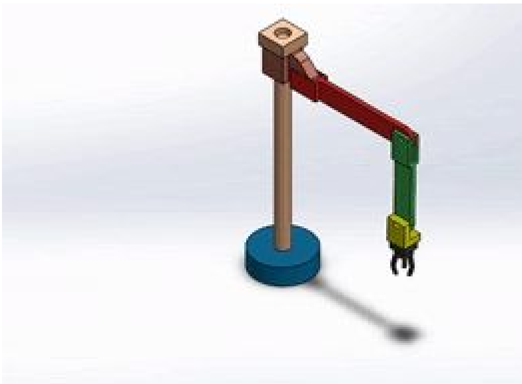


Fig. 9 (to the left): Robot-maximum extension
Fig. 10 (top): Robot moving to pluck apple

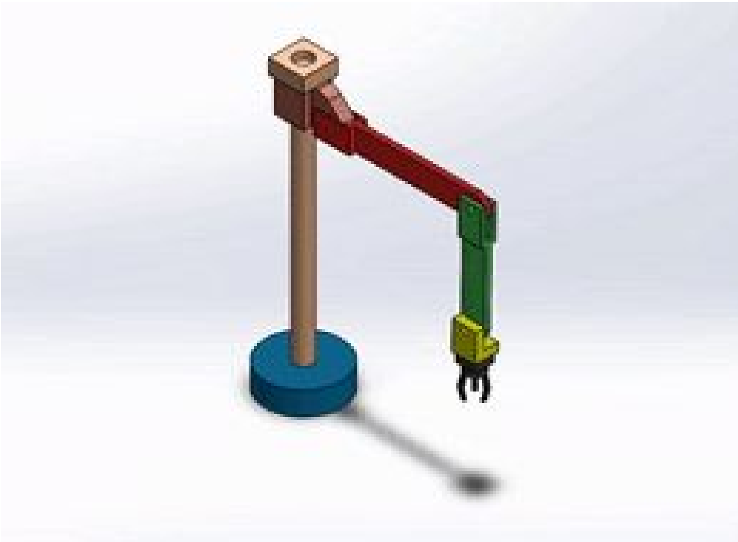


Fig. 11 (top): Robot moving to palletize apple

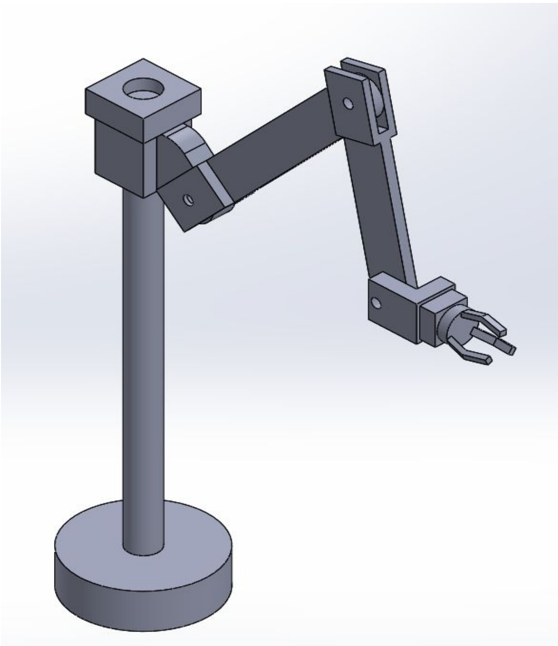


Fig. 12 : Robot-minimum extension

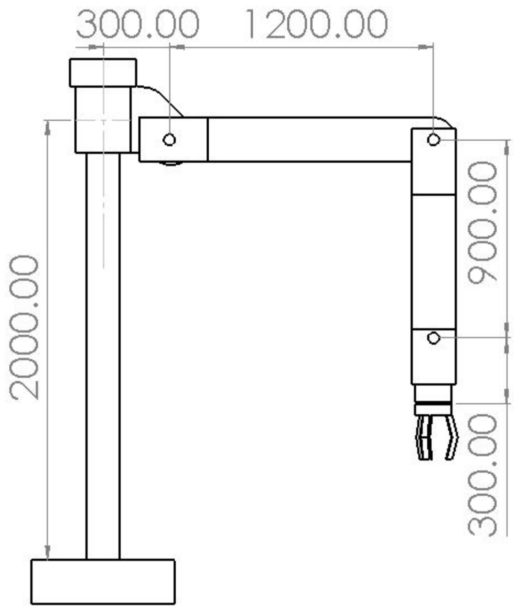


Figure 13: Drawing with dimensions

Figure 14: Isometric view of the home position

Chapter 3

Forward Kinematics

3.1 Denavit-Hartenberg frame link assignment:

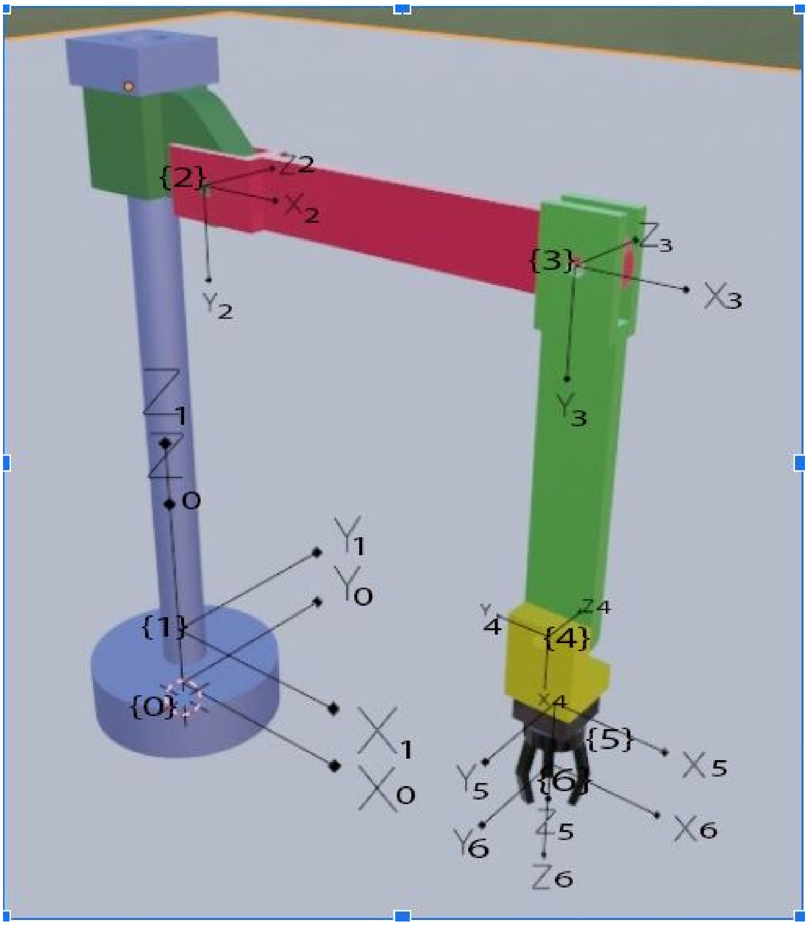


Figure 15 : DH frame link assignment

3.2 Joint link parameter table:

Link i	**θi** (degrees)	**αi** (degrees)	**ai**	**di**	**qi**
1	**t1**	**0**	**0**	**0**	**t1**
2	**0**	**-90**	**a2**	**d2**	**d2**
3	**t3**	**0**	**a3**	**0**	**t3**
4	**t4+90**	**0**	**a4**	**d4**	**t4**
5	**t5-90**	**-90**	**0**	**d5**	**t5**
6	**t6**	**0**	**0**	**d6**	**t6**

Table 1 : Joint link parameter table

The values of link lengths and joint distances as obtained from the CAD model are:

a2 = 300mm; a3 = 1200 mm; a4 = 900 mm; d4 = -50mm ; d5 = 100mm ; d6 = 300mm;

3.3 <u>Transformation Matrices</u>: (See Appendix A1)

$$^{0}T_1 = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^{3}T_4 = \begin{bmatrix} -S4 & -C4 & 0 & -900*S4 \\ C4 & -S4 & 0 & 900*C4 \\ 0 & 0 & 1 & -50 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}T_2 = \begin{bmatrix} 1 & 0 & 0 & 300 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^{4}T_5 = \begin{bmatrix} S5 & 0 & C5 & 0 \\ -C5 & 0 & S5 & 0 \\ 0 & -1 & 0 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}T_3 = \begin{bmatrix} C3 & -S3 & 0 & 1200*C3 \\ S3 & C3 & 0 & 1200*S3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^{5}T_6 = \begin{bmatrix} C6 & -S6 & 0 & 0 \\ S6 & C6 & 0 & 0 \\ 0 & 0 & 1 & 300 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.4 <u>End effector matrix</u>:

$$^{0}T_6 = \begin{bmatrix} S1*S6+C1*C345*C6 & S1*C6-C1*C345*S6 & -C1*S345 & 300*C1*(1+4*C3-3*S34-S345)-50*S1 \\ C345*C6*S1-C1*S6 & -C1*C6-C345*S1*S6 & -S1*S345 & 300*S1*(1+4*C3-3*S34-S345)+50*C1 \\ -C6*S345 & S6*S345 & -C345 & d2-300*C345-900*C34-1200*S3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where:

- S1 = sin(t1)
- S34 = sin(t3 + t4)
- S345 = sin(t3 + t4 + t5)

3.5 Enumeration of Forward Kinematics for different positions:

1. **Forward Kinematics at home position:**

At home position, the gripper faces vertically downwards (please refer to figure 8) . The home position parameter table looks like the following:

Link i	**θi (degrees)**	**αi (degrees)**	**ai (mm)**	**di (mm)**	**qi**
1	0	0	0	0	t1
2	0	-90	300	2000	d2
3	0	0	1200	0	t3
4	90	0	900	-50	t4
5	-90	-90	0	100	t5
6	0	0	0	300	t6

Table 2 : Parameters for home position

2. **Forward Kinematics during plucking motion (figure 9):**

The position parameter table for plucking motion looks like the following:

Link i	**θi (degrees)**	**αi (degrees)**	**ai (mm)**	**di (mm)**	**qi**
1	90	0	0	0	t1
2	0	-90	300	2000	d2
3	15.294	0	1200	0	t3
4	-35.886+90	0	900	-50	t4
5	20.592-90	-90	0	100	t5
6	270	0	0	300	t6

Table 3 : Parameters for plucking motion

3. Forward Kinematics during palletizing:

During these calculations, we are placing the apple at the far end (corner) of the palette at point C(1800,500) according to the robot base (please refer to figure 3) . The position parameter table for palletizing motion looks like the following:

Link i	θi (degrees)	ɑi (degrees)	ai (mm)	di (mm)	qi
1	13.99	0	0	0	t1
2	0	-90	300	1760	d2
3	56.63	0	1200	0	t3
4	-100.083+90	0	900	-50	t4
5	-30.282-90	-90	0	100	t5
6	0	0	0	300	t6

Table 4 : Parameters for palletizing motion

Chapter 4

Inverse Kinematics

4.1 Inverse Kinematics Solution (See Appendix A2)

The inverse kinematics solution is found as follows:

$$T_E = \begin{bmatrix} n_x & o_x & a_x & d_x \\ n_y & o_y & a_y & d_y \\ n_z & o_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{0}T_6$$

Eq. 1

- **Step-1**

Pre-multiply Eq. 1 with inverse of 0T1:

Taking the element (2,3), we get:

=> ay*C1 - ax*S1 = 0

=> **θ1 = Atan2d(ay, ax)** Eq. 2

- **Step-2**

Taking elements (2,1) and (2,2) from the previous matrix, we get:

=> S6 = -(ny*C1 - nx*S1)

=> C6 = -(oy*C1 - ox*S1)

=> **θ6 = Atan2d(-ny*C1 + nx*S1, -oy*C1 + ox*S1)** Eq. 3

- **Step-3**

Pre-multiply Eq. 1 with inverse of 1T2, 0T1 and post-multiply with 5T6:

Taking elements (1,3) and (2,3), we get:

=> S345 = -ax*C1 - ay*S1

=> C345 = -az,

=> **θ345 = Atan2d(-ax*C1 - ay*S1, -az)** Eq. 4

- **Step-4**

  Pre-multiply Eq. 1 with inverse of 2T3, 1T2, 0T1, and post-multiply with inverse of 5T6:

  Taking elements (1,2) and (2,2), we get:

  => ox*C1*C3*C6 - nz*S3*S6 - oz*C6*S3 + nx*C1*C3*S6 + oy*C3*C6*S1 + ny*C3*S1*S6 = 0
  => -ox*C1*C6*S3 - nz*C3*S6 - oz*C3*C6 - nx*C1*S3*S6 - oy*C6*S1*S3 - ny*S1*S3*S6 = 0

  Add both equations and simplify:

  => (C3 - S3)*[ox*C1*C6 + nx*C1*S6 + oy*C6*S1 + ny*S1*S6] = (C3 + S3)*[nz*S6 + oz*C6]

  => (C3 - S3) * k1= (C3 + S3) * k2

  where, **k1 = ox*C1*C6 + nx*C1*S6 + oy*C6*S1 + ny*S1*S6**

  and, **k2 = nz*S6 + oz*C6,** are constants.

  => C3 * (k1 - k2) = S3 * (k1 + k2)

  => S3 / C3 = (k1 - k2) / (k1 + k2)

  => **θ3 = Atan2d((k1 - k2), (k1 + k2))** Eq. 5

- **Step-5**

  Post-multiply Eq. 1 with inverse of 5T6:

  Taking the element (1,4), we get:

  => k3 - 900*C1*(S34) = dx - ax*300

  where, **k3 = C1*(300 + 1200*C3) + 50*S1 - 100*S1**, is a constant.

  => S34 = k4

  => C34 = ± sqrt(1-(k4^2))

  where, **k4 = -(dx - ax*300 - k3)/900*C1,** is a constant.

  => **θ3 + θ4 = Atan2d(k4, ± sqrt(1-(k4^2))** Eq. 6

  From Eq. 5:

  => **θ4 = Atan2d(k4, ± sqrt(1-(k4^2)) - Atan2d((k1 - k2), (k1 + k2))** Eq. 7

There can be two solutions of in eq.7 for $\theta 4$ (One taking the positive sign and one with the negative sign)

From Eq. 4, and Eq. 6:

=> **$\theta 5$ = Atan2d(-ax*C1 - ay*S1, -az) - Atan2d(k4, ± sqrt(1-($k4^2$))** Eq. 8

Here in Eq.8 in the second term on the right hand side, it is visible that there can be two solutions for $\theta 5$ (one taking the positive sign and one taking the negative sign).

- **Step-6**

Taking the element (3,4) from the previous matrix, we get:

=> **d2 = dz - az*300 + 900*C34 + 1200*S3** Eq. 9

Hence, the inverse kinematics solution is now complete. The joint variables have been found in the above equations 1 to 9, using the constant k1, k2, k3 and k4. The inverse kinematics solution is summarised below.

$\theta 1$	Atan2d(ay, ax)
$\theta 6$	Atan2d(-ny*C1 + nx*S1, -oy*C1 + ox*S1)
$\theta 345$	Atan2d(-ax*C1 - ay*S1, -az)
k1	ox*C1*C6 + nx*C1*S6 + oy*C6*S1 + ny*S1*S6
k2	nz*S6 + oz*C6
$\theta 3$	Atan2d((k1 - k2), (k1 + k2))
k3	C1*(300 + 1200*C3) + 50*S1 - 100*S1
k4	-(dx - ax*300 - k3)/900*C1
$\theta 3 + \theta 4$	Atan2d(k4, ± sqrt(1-($k4^2$))
$\theta 4$	Atan2d(k4, ± sqrt(1-($k4^2$)) - Atan2d((k1 - k2), (k1 + k2))
$\theta 5$	Atan2d(-ax*C1 - ay*S1, -az) - Atan2d(k4, ± sqrt(1-($k4^2$))
d2	dz - az*300 + 900*C34 + 1200*S3

Table 5 : Inverse Kinematics Solution

4.2 <u>Existence of solutions:</u>

A close examination of the equations and constants obtained above reveals that the solutions to the inverse kinematics problem exist only if the term k4 has a value in the range [-1, 1].

4.3 <u>Multiplicity of solutions</u>:

qi	**Multiplicity**
$\theta 1$	1
d2	2
$\theta 3$	1
$\theta 4$	2
$\theta 5$	2
$\theta 6$	1

Table 6 : Multiplicity of solutions

Chapter 5

Jacobian and Velocity Analysis

5.1 Jacobian and Velocities (See Appendix A3)

The link velocities obtained are specified in the google doc. We have not included them in this document since the matrices are too huge and the formatting of the page doesn't support them to be enlisted here.

$$J1 = \begin{bmatrix} 300*S1(S345-1)-1200*C3*S1-50*C1+900*S1*S34 \\ 300*C1-50*S1+1200*C1*C3-300*S345*C1-900*C1*S34 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad J3 = \begin{bmatrix} -C1*(300*C345+900*C34+1200*S3) \\ -S1*(300*C345+900*C34+1200*S3) \\ 300*S345+900*S34-1200*C3 \\ -S1 \\ C1 \\ 0 \end{bmatrix}$$

$$J4 = \begin{bmatrix} -C1*(300*C345+900*C34) \\ -S1*(300*C345+900*C34) \\ 300*S345+900*S34 \\ -S1 \\ C1 \\ 0 \end{bmatrix} \qquad J5 = \begin{bmatrix} -300*C345*C1 \\ -300*C345*S1 \\ 300*S345 \\ -S1 \\ C1 \\ 0 \end{bmatrix}$$

$$J6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -S345*C1 \\ -S345*S1 \\ -C345 \end{bmatrix}$$

Combining all the above jacobian matrices, we get:

Jacobian matrix, J =

$$J = \begin{bmatrix} 300*S1(S345-1)-1200*C3*S1-50*C1+900*S1*S34, & 0, & -C1*(300*C345+900*C34+1200*S3), & -C1*(300*C345+900*C34), & -300*C345*C1, & 0 \\ 300*C1-50*S1+1200*C1*C3-300*S345*C1-900*C1*S34, & 0, & -S1*(300*C345+900*C34+1200*S3), & -S1*(300*C345+900*C34) & -300*C345*S1, & 0 \\ 0, & 1, & 300*S345+900*S34-1200*C3, & 300*S345+900*S34 & 300*S345, & 0 \\ 0, & 0, & -S1, & -S1 & -S1 & -S345*C1 \\ 0, & 0, & C1, & C1 & C1 & -S345*S1 \\ 1 & 0 & 0 & 0 & 0 & -C345 \end{bmatrix}$$

Considering only first four columns and first four rows corresponding to links which are not a part of wrist we get :

Jprime1 =

$$Jprime1 = \begin{bmatrix} 300*S345*S1-300*S1-1200*C3*S1-50*C1+900*S1*S34, & 0, & -C1*(300*C345+900*C34+1200*S3), & -C1*(300*C345+900*C34) \\ 300*C1-50*S1+1200*C1*C3-300*S345*C1-900*C1*S34, & 0, & -S1*(300*C345+900*C34+1200*S3), & -S1*(300*C345+900*C34) \\ 0, & 1, & 300*S345+900*S34-1200*C3, & 300*S345+900*S34 \\ 0, & 0, & -S1, & -S1 \end{bmatrix}$$

The determinant is given by :

$$|Jprime1| = 360000*sin(theta1)*[(3*cos(2*theta3+theta4))/2+2*sin(2*theta3)+cos(2*theta3+theta4+theta5)/2-cos(theta4+theta5)/2-(3*cos(theta4))/2+sin(theta3)]$$

The inverse of this jacobian can be found out using the code mentioned in the appendix. We have not elaborated it here for brevity.

Considering the last 2 columns and 4th and 5th rows of the Jacobian to account for the wrist, we have:

$$Jprime2 = \begin{bmatrix} -S1 & -S345*C1 \\ C1 & -S345*S1 \end{bmatrix}$$

DetJprime2 =

$$|Jprime2| = sin(theta3+theta4+theta5)$$

invJprime2 =

[-sin(t1),   cos(t1)]

[-cos(t1)/sin(t3 + t4 + t5),   -sin(t1)/sin(t3 + t4 + t5)]

5.2 **Singularities**:

- Equating |Jprime1| to zero, we have:
    1) $\sin(\Theta_1) = 0$ i.e. $\Theta_1 = 0$ or π or
    2) $(3\cos(2\Theta_3 + \Theta_4))/2 + 2\sin(2\Theta_3) + \cos(2\Theta_3 + \Theta_4 + \Theta_5)/2 - \cos(\Theta_4 + \Theta_5)/2 - (3\cos(\Theta_4))/2 + \sin(\Theta_3) = 0$

       By observation, this equation collapses to zero when $\Theta_3 = 0$.
       To verify that there is no other value of Θ_3, we ran a python script which looped over all values of Θ_3, Θ_4 and Θ_5 from 0 to 2π. We found that apart from $\Theta_3 = 0$, there are no other possibilities for which the expression becomes 0. We have uploaded the python code and the csv file (singularity.csv) of singularities generated in this drive link. Essentially, for $\Theta_3=0$, at all values of Θ_4 and Θ_5, a singularity is occurring.

- Equating |Jprime2| to zero :
  We can clearly see from the above two J' (J-prime) matrices, that the singularities exist where $(\Theta_3+\Theta_4+\Theta_5) = 0, 90, 180, 270$. (See fig. 14) This means that the number of combinations of the three quantities is infinite. In this case we have no option but to analyze the torques near these values to pull out pseudo singularities.

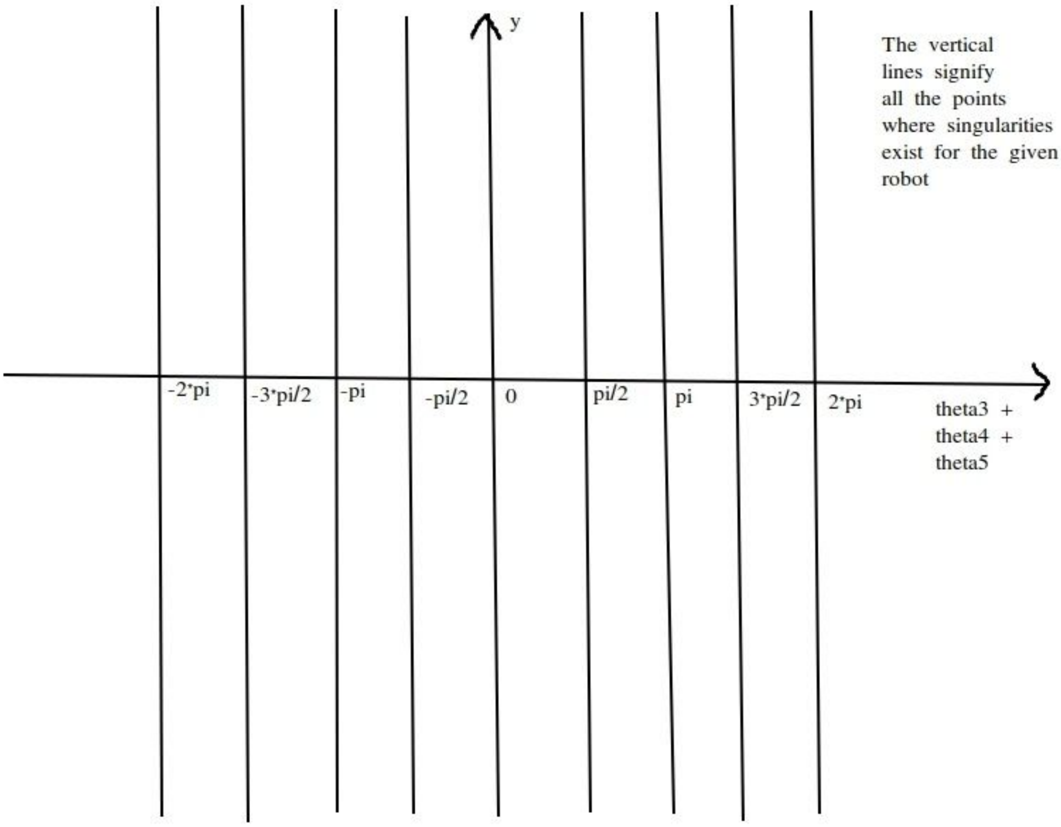


Fig. 16. Singularities for Jprime2

Chapter 6

Dynamic Analysis

6.1 Newton-Euler formulation (See Appendix A4)

We ran the code for dynamic analysis and got the equations for τ.

- τ(simplified)
- Forces(f)
- Moment(η)

The torques for all joints are independent of theta6 (θ_6).

6.2 Torque Graphs

Using the code in appendix A4, we plotted the following torque graphs for different joints. Since the torque is dependent on various joint parameters like (θ_1,d_2,θ_3,θ_4,θ_5) we varied them one by one and plotted all the graphs seen below.

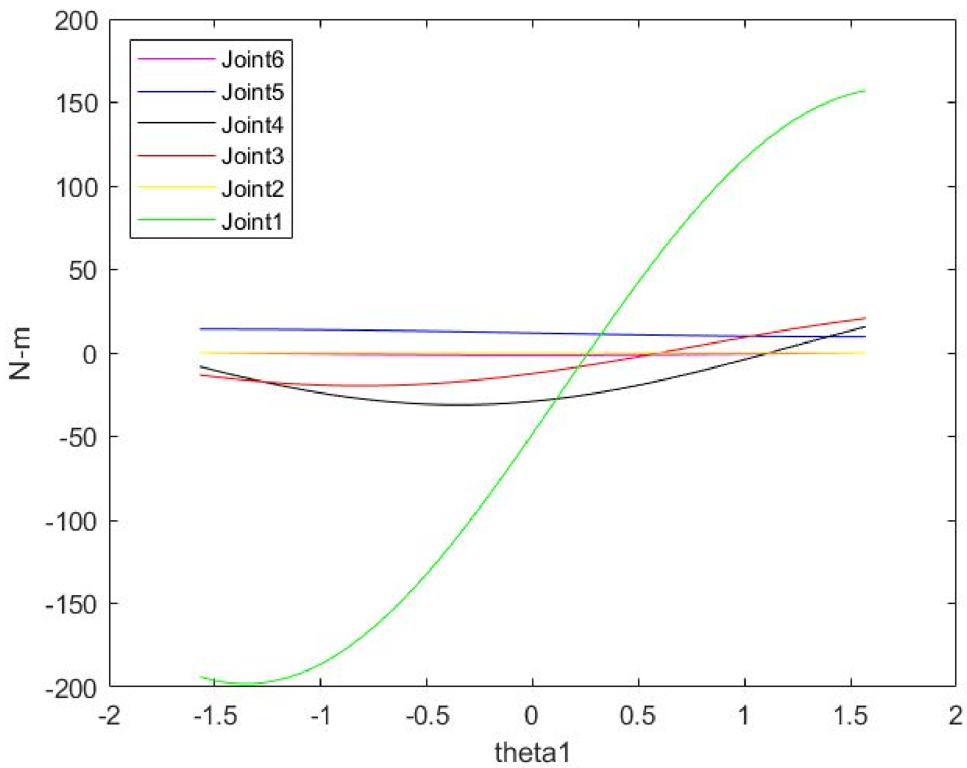


Figure 17: Torque at joints w.r.t Θ_1

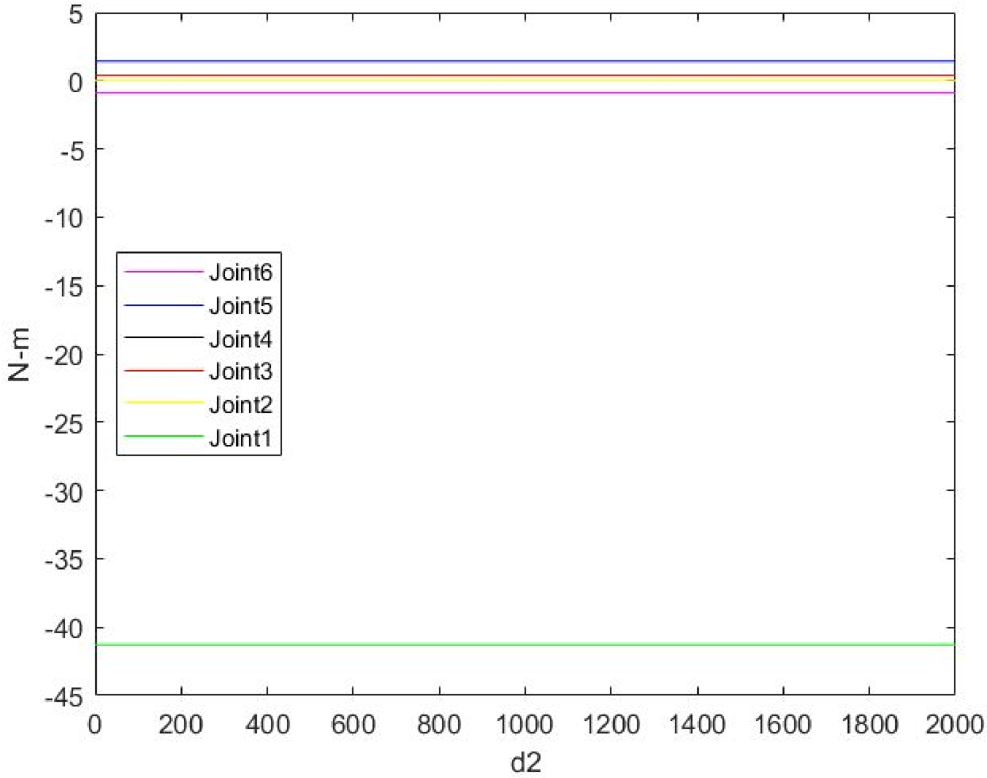


Figure 18: Torque at joints w.r.t d2

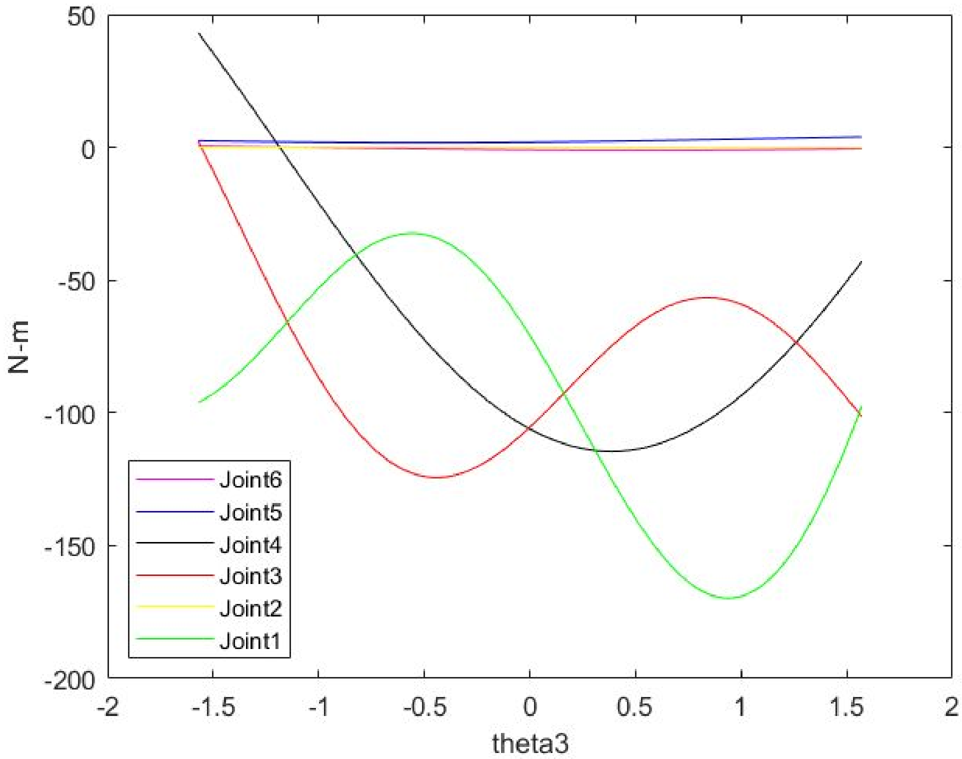


Figure 19: Torque at joints w.r.t Θ_3

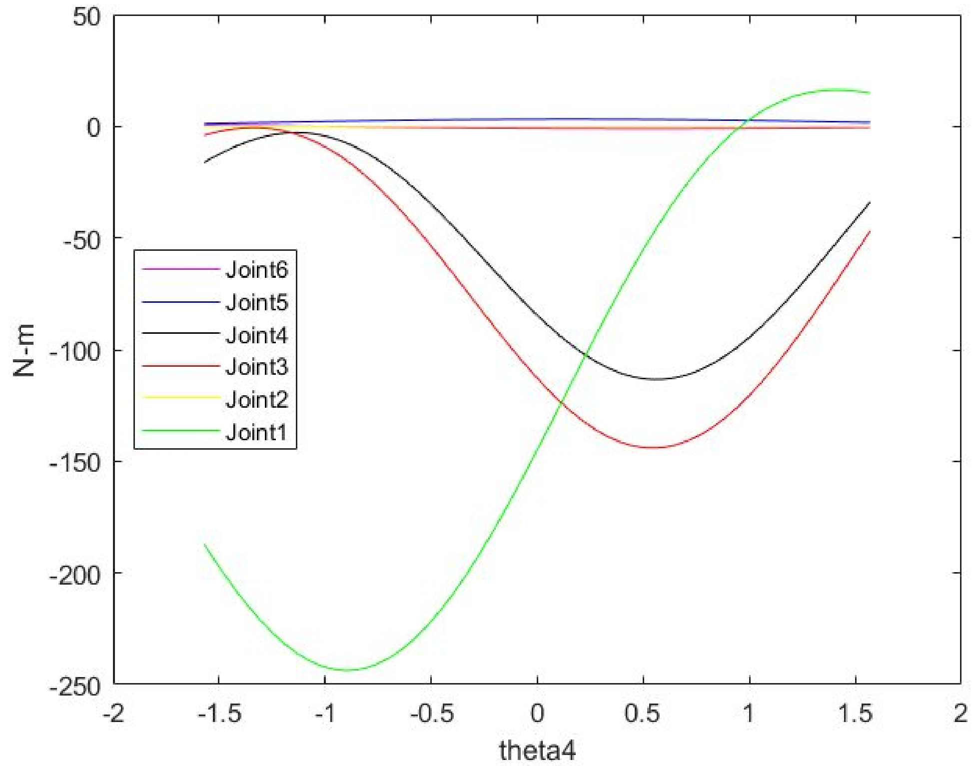


Figure 20: Torque at joints w.r.t Θ_4

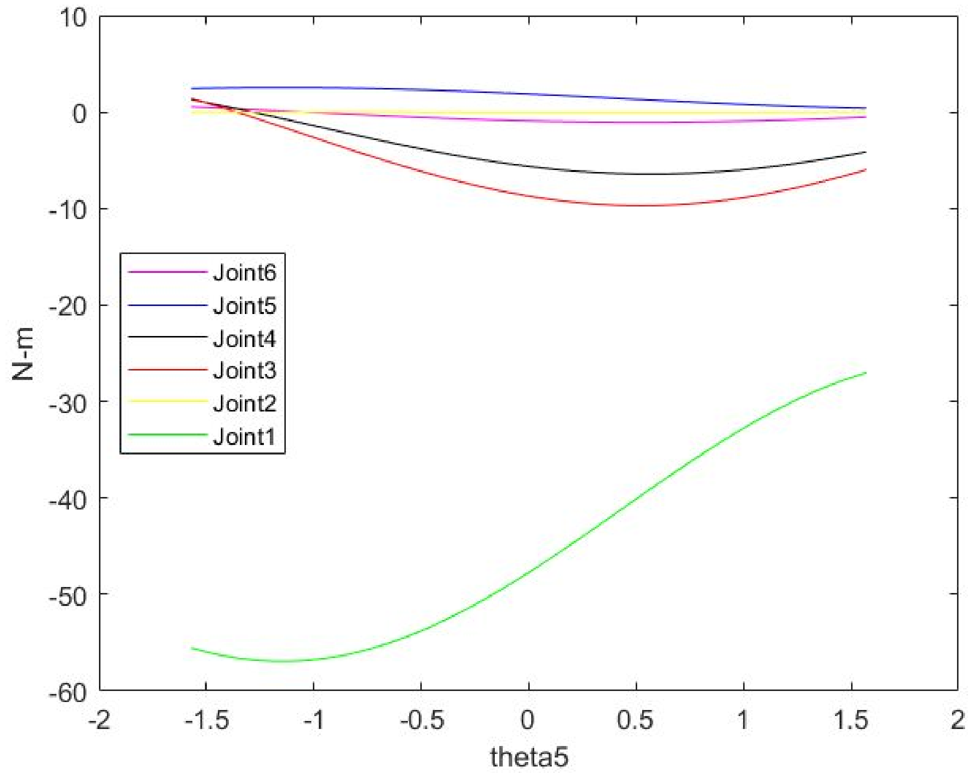


Figure 21: Torque at joints w.r.t Θ_5

6.3 <u>Recommendations for selection of motors</u>

- **<u>Joint - 1:</u>** From figure number 20, it is evident that the maximum torque required for joint 1 occurs when we consider its variance with theta4 (θ_4). The maximum torque required can be found out using simple matlab code using max/min function. The maximum torque required is **243.6176 N-m**. Hence, we can recommend a 300 N-m motor for joint1 (Base rotation).
- **<u>Joint - 3 & 4:</u>** From figure number 20, it is evident that the maximum torque required for joint 3 is **144.0253 N-m** and from figure 19, we know that the maximum torque for joint 4 is **114.4722 N-m.** Hence, we can recommend a 150 N-m motor for these joints. (Shoulder and Elbow joint)
- **<u>Joint - 5 & 6:</u>** The maximum torque requirement for joint 5 from figure 17 is **14.2362 N-m.** Hence, we can recommend a 20 N-m motor for this joint. For joint 6, from figure 18, the maximum torque requirement is **0.918 N-m.** But since this is the end point of the gripper, the apple will be placed within the gripper, so we will still recommend a **10 N-m** motor.
- **<u>Joint - 2:</u>** For the joint 2, the motion can be achieved either by using a linear pneumatic actuator, or a lead-screw linear actuator.

Chapter 7

Final Conclusion

The apple fruit picker is a generalized solution to the fruit orchards around the globe. The gripper size can be changed as per requirement of the job. Once started, the arm can pick up the fruits and palletize them into trays. These trays can then be further carried to the warehouses for further shipment.

The arm can also be used in pick and place tasks, in warehouses, etc.

Furthermore, the camera feed can be fed to a pre-trained neural network so that it will detect a ripe or a raw fruit. This will again increase the efficiency, as a lot of work goes into educating the labourers for picking the ripe fruits. This in turn increases the quality from source i.e the warehouse and helps to segregate the fruits, avoiding overly ripe fruits turn bad and disrupt the whole container.

This project is still in its infancy and some physical prototyping is required to identify some design errors which might need rectification. However, the project proves to be promising and is worthy of consideration for prototyping.

Chapter 8

References

- Mittal. R. K, and I. J Nagrath, *Robotics And Control*, Tata Mc Graw-Hill, 2003.
- Baeten, Johan & Donné, Kevin & Boedrij, Sven & Beckers, Wim & Claesen, Eric. (2007). Autonomous Fruit Picking Machine: A Robotic Apple Harvester. Springer Tracts in Advanced Robotics. 42. 10.1007/978-3-540-75404-6_51.
- “Automatic fruit picker demonstration by FF Robotics” YouTube, GoodFruitGrower, 3rd April 2017, https://youtu.be/UaL3UxUclKY.
- “Robotic apple picker trials continue in Washington by FF Robotics” YouTube, GoodFruitGrower, 28th October 2016 https://youtu.be/mS0coCmXiYU.
- “Robotics Arms Race by FF Robotics” YouTube, GoodFruitGrower, 30th November 2019 https://youtu.be/-PtqZA2enkQ.

Chapter 9

Appendices

A1: Chapter 3 MATLAB Code for forward kinematics

MATLAB code used to enumerate the forward kinematics

```
clc
clear variables
syms theta alph a d;
syms t1 d2 t3 t4 t5 t6;

% Input joint link parameter table below in the order [a alpha d theta]
linkVar = [0  0  0  t1;  300  sym(-pi/2)  d2  0;  1200  0  0  t3;  900  0  -50
sym(t4+pi/2); 0 sym(-pi/2) 100 sym(t5-pi/2); 0 0 300 t6];

T = @(a, alph, d, theta)[cos(theta) -sin(theta)*cos(alph)
sin(theta)*sin(alph) a*cos(theta); sin(theta) cos(theta)*cos(alph)
-cos(theta)*sin(alph) a*sin(theta); 0 sin(alph) cos(alph) d; 0 0 0 1];

[n,~] = size(linkVar);
Tf(:,:,1) = sym(eye(4));
t(:,:,1) = sym(eye(4));

for i=1:n

    t(:,:,i+1) = T(linkVar(i,1),linkVar(i,2),linkVar(i,3),linkVar(i,4));
    fprintf('%dT%d = \n', i-1, i)
    disp(simplify(t(:,:,i+1)))

    Tf(:,:,i+1) = simplify(Tf(:,:,i)*t(:,:,i+1));
    fprintf('0T%d = \n', i)
    disp(simplify(Tf(:,:,i+1)))
end
```

A2: Chapter 4 Inverse Kinematics Solution

MATLAB code used to find inverse kinematics solution

```
clc
clear variables
syms t1 d2 t3 t4 t5 t6;
syms nx ox ax dx ny oy ay dy nz oz az dz
T = @(a, alph, d, theta)[cos(theta) -sin(theta)*cos(alph)
sin(theta)*sin(alph) a*cos(theta); sin(theta) cos(theta)*cos(alph)
-cos(theta)*sin(alph) a*sin(theta); 0 sin(alph) cos(alph) d; 0 0 0 1];

Te = [nx ox ax dx; ny oy ay dy; nz oz az dz; 0 0 0 1];     % End effector
matrix
Tf =
[sin(t1)*sin(t6)+cos(t3+t4+t5)*cos(t1)*cos(t6),cos(t6)*sin(t1)-cos(t3+t4+t5)*
cos(t1)*sin(t6),-sin(t3+t4+t5)*cos(t1),300*cos(t1)-50*sin(t1)+1200*cos(t1)*co
s(t3)-300*sin(t3+t4+t5)*cos(t1)-900*cos(t1)*cos(t3)*sin(t4)-900*cos(t1)*cos(t
4)*sin(t3);cos(t3+t4+t5)*cos(t6)*sin(t1)-cos(t1)*sin(t6),-cos(t1)*cos(t6)-cos
(t3+t4+t5)*sin(t1)*sin(t6),-sin(t3+t4+t5)*sin(t1),50*cos(t1)+300*sin(t1)+1200
*cos(t3)*sin(t1)-300*sin(t3+t4+t5)*sin(t1)-900*cos(t3)*sin(t1)*sin(t4)-900*co
s(t4)*sin(t1)*sin(t3);-sin(t3+t4+t5)*cos(t6), sin(t3+t4+t5)*sin(t6),
-cos(t3+t4+t5), d2-300*cos(t3+t4+t5)-900*cos(t3+t4)-1200*sin(t3); 0, 0, 0,
1]; % 0T6 matrix

% Use inverse from here
% inv(T(0, 0, 0, t1)) inv(T(300, sym(-pi/2), d2, 0)) inv(T(1200, 0, 0, t3))
% inv(T(900, 0, -50, sym(t4+pi/2))) inv(T(0, sym(-pi/2), 100, sym(t5-pi/2)))
inv(T(0, 0, 300, t6))
%% Step - 1 & 2
disp('For Step 1 & 2')
disp('LHS = ')
disp(simplify(inv(T(0,0,0,t1))*Te))   % Perform pre multiplication
disp('RHS = ')
disp(simplify(inv(T(0,0,0,t1))*Tf))   % Perform pre multiplication

%% Step - 3
disp('For Step 3')
disp('LHS = ')
disp(simplify(inv(T(300,sym(-pi/2),d2,0))*inv(T(0,0,0,t1))*Te*inv(T(0,0,300,t
6)))) % Perform pre/post multiplication
disp('RHS = ')
disp(simplify(inv(T(300,sym(-pi/2),d2,0))*inv(T(0,0,0,t1))*Tf*inv(T(0,0,300,t
6)))) % Perform pre/post multiplication

%% Step - 4
disp('For Step 4')
disp('LHS = ')
```

```
disp(simplify(inv(T(1200,0,0,t3))*inv(T(300,sym(-pi/2),d2,0))*inv(T(0,0,0,t1)
)*Te*inv(T(0,0,300,t6)))) % Perform pre/post multiplication
disp('RHS = ')
disp(simplify(inv(T(1200,0,0,t3))*inv(T(300,sym(-pi/2),d2,0))*inv(T(0,0,0,t1)
)*Tf*inv(T(0,0,300,t6)))) % Perform pre/post multiplication

%% Step - 5 & 6
disp('For Step 5 & 6')
disp('LHS = ')
disp(simplify(Te*inv(T(0, 0, 300, t6))))   % Perform post multiplication
disp('RHS = ')
disp(simplify(Tf*inv(T(0, 0, 300, t6))))   % Perform post multiplication
```

A3: **Chapter 5 Jacobian and Singularities**

Generalized MATLAB code to find link velocities and Jacobian

```
clc
clear variables

%% Inputs (only make changes here)
syms t1 a2 d2 t3 a3 t4 a4 d4 t5 d5 t6 d6   %    Variables    in    joint    link
parameter table
syms t1dot d2dot t3dot t4dot t5dot t6dot        % Joint velocities

% List the joint variables q_i
q_i = [t1 d2 t3 t4 t5 t6];

% Input joint link parameter table below in the order [a alpha d theta]
linkVar  =  [0  0  0  t1;  300  sym(-pi/2)  d2  0;  1200  0  0  t3;  900  0  -50
sym(t4+pi/2); 0 sym(-pi/2) 100 sym(t5-pi/2); 0 0 300 t6]; % a alpha d theta

% Define the joint velocities
dots = [t1dot d2dot t3dot t4dot t5dot t6dot];

% Define the type of joint, where 0 = prismatic, and 1 = revolute
joints = [1 0 1 1 1 1];

%% Initialize

zCap = [0 0 1]';
iDi = [0 0 0 1]';
omega = [0 0 0]';

%% Finding linear and angular link velocities

T = @(a, alph, d, theta)[cos(theta) -sin(theta)*cos(alph)
sin(theta)*sin(alph) a*cos(theta); sin(theta) cos(theta)*cos(alph)
-cos(theta)*sin(alph) a*sin(theta); 0 sin(alph) cos(alph) d; 0 0 0 1];

[r,~] = size(linkVar);
Tf = eye(4);

for i=1:r

    t = T(linkVar(i,1),linkVar(i,2),linkVar(i,3),linkVar(i,4));
    tsave = Tf;
    Tf = simplify(Tf*t);
    fprintf('0T%d = \n', i)
    disp(simplify(Tf))

    % Find linear velocity
    vel = [0 0 0 0]';
    for j = 1:i
        vel = vel + simplify(diff(Tf, q_i(j))*iDi*dots(j));
    end
    fprintf('V%d = \n', i)
    disp(vel(1:3))
```

```
    % Find angular velocity
    omega = omega + joints(i)*simplify(tsave(1:3,1:3)*zCap*dots(i));
    fprintf('omega%d = \n', i)
    disp(omega)

end

%% Finding the Jacobian
tsave = Tf;
Tf = eye(4);
J = sym(zeros(r));

for i=1:r

    Pi_1 = Tf(1:3,3);
    if joints(i)==0
        fprintf('J%d = \n', i)
        J_single = [Pi_1; zeros(3,1)];
        disp(J_single)
        J(:,i) = J_single;
    else
        Pi_1_n = tsave(:,4) - Tf(:,4);
        fprintf('J%d = \n', i)
        J_single = [simplify(cross(Pi_1, Pi_1_n(1:3))); Pi_1];
        disp(J_single)
        J(:,i) = J_single;
    end
    t = T(linkVar(i,1),linkVar(i,2),linkVar(i,3),linkVar(i,4));
    Tf = simplify(Tf*t);

end

fprintf('Final Jacobian, J = \n')
disp(J)

% Taking first 4 rows of the first 4 columns
Jprime1 = J(1:4, 1:4);
disp('Jprime1 =')
disp(Jprime1)
disp('|Jprime1| = ')
disp(simplify(det(Jprime1)))
disp('invJprime1 = ')
disp(simplify(inv(Jprime1)));

% Taking 4th and 5th rows of the last 2 columns:
Jprime2 = J(4:5,5:6);
disp('Jprime2 =')
disp(Jprime2)
disp('|Jprime2| = ')
disp(simplify(det(Jprime2)))
disp('invJprime2 = ')
disp(simplify(inv(Jprime2)));
```

A4: **Chapter 6 Dynamic Analysis**

MATLAB code to find the forces, torques and plot the torque graphs

Disclaimer - This code will take about 10 minutes to run at first. It will output all the torque graphs in separate windows. It will print the progress as it runs.

```
%% Code
clear variable;
clc;
n = 6;
% n = no. of joints
syms theta alpha d a Ct St Ca Sa [1 n] ;
Temp = eye(4);
syms nx ny nz ox oy oz ax ay az dx dy dz
for i = 1:n
    a = [0,300,1200,900,0,0];
    alpha = [0,-90,0,0,-90,0];
    d = [0,d2,0,-50,100,360];
    theta = [theta1,0,theta3,theta4+pi/2,theta5-pi/2,theta6];
    Ca(i) = cosd(alpha(i));
    Sa(i) = sind(alpha(i));
    Ct(i) = cos(theta(i));
    St(i) = sin(theta(i));
     T = simplify([Ct(i),-St(i)*Ca(i),St(i)*Sa(i),a(i)*Ct(i);
         St(i),Ct(i)*Ca(i),-Ct(i)*Sa(i),a(i)*St(i);
         0,Sa(i),Ca(i),d(i);
         0,0,0,1]);

    TT(:,:,i) = T;
    Temp = simplify(Temp * T);
    Teff(:,:,i) = Temp;

end
fprintf("Forward kinematics done \n")
%% Newton Euler - Forward iteration
syms  D DD th thh [1 n]
% D is d_dot and DD is d_dot_dot
% th is theta_dot & thh is theta_dot_dot
m = [2;5;3;2;0.5;0.5];
W00 = [0;0;0];
Wdot00 = [0;0;0];
vdot00 = [0;-9.83;0];
Lby2 = [0,1000,600,450,100,55];
L = Lby2.*2;

r(:,:,1) = [0;0;-Lby2(1)];
r(:,:,2) = [0;-Lby2(2);0];
r(:,:,3) = [-Lby2(3);0;0];
r(:,:,4) = [-Lby2(4);0;0];
r(:,:,5) = [0;0;-Lby2(5)];
r(:,:,6) = [0;0;-Lby2(6)];

I(:,:,1) = [0,0,0,0; 0,0,0,0; 0,0,0,0; 0,0,0,m(1)];
I(:,:,2) = [0,0,0,0; 0,(1/3)*m(2)*L(2).^2,0,-m(2)*L(2)/2; 0,0,0,0;
     0,-m(2)*L(2)/2,0,m(2)];
```

```
I(:,:,3) = [(1/3)*m(3)*L(3).^2,0,0,-m(3)*L(3)/2; 0,0,0,0; 0,0,0,0;
      -m(3)*L(3)/2,0,0,m(3)];
I(:,:,4) = [(1/3)*m(4)*L(4).^2,0,0,-m(4)*L(4)/2; 0,0,0,0; 0,0,0,0;
      -m(4)*L(4)/2,0,0,m(4)];
I(:,:,5) = [0,0,0,0; 0,0,0,0; 0,0,(1/3)*m(5)*L(5).^2,-m(5)*L(5)/2;
      0,0,-m(5)*L(5)/2,m(5)];
I(:,:,6) = [0,0,0,0; 0,0,0,0; 0,0,(1/3)*m(6)*L(6).^2,-m(6)*L(6)/2;
      0,0,-m(6)*L(6)/2,m(6)];


ques = ['R','P','R','R','R','R'];
for i = 1: n

    if ques(i) == 'R'
    %for revolute joint
        if i==1
            W(:,:,1) = TT(1:3,1:3,1)\(W00 + [0;0;1]*th(1));
            Wdot(:,:,1) = TT(1:3,1:3,1)\(Wdot00 + [0;0;1].*thh(1) +
      cross(W00,[0;0;1].*th(1)));
            vdot(:,:,1) = simplify(TT(1:3,1:3,1)\ vdot00 +
      cross(Wdot(:,:,1),Teff(1:3,1:3,1)\TT(1:3,4,1)) +
      cross(W(:,:,1),cross(W(:,:,1),Teff(1:3,1:3,1)\TT(1:3,4,1))));
        else
            W(:,:,i) = TT(1:3,1:3,i)\(W(:,:,i-1) + [0;0;1]*th(i));
            Wdot(:,:,i) = TT(1:3,1:3,i)\(Wdot(:,:,i-1) + [0;0;1]*thh(i) +
      cross(W(:,:,i-1),[0;0;1].*th(i)));
            vdot(:,:,i) = simplify(TT(1:3,1:3,i)\ vdot(:,:,i-1) +
      cross(Wdot(:,:,i),Teff(1:3,1:3,i)\TT(1:3,4,i)) +
      cross(W(:,:,i),cross(W(:,:,i),Teff(1:3,1:3,i)\TT(1:3,4,i))));
         end

    else
      %for prismatic joint
          if i==1
             W(:,:,1) = TT(1:3,1:3,1)\W00;
             Wdot(:,:,1)= TT(1:3,1:3,1)\Wdot00;
             vdot(:,:,1) = simplify(TT(1:3,1:3,1)\ (vdot00 + [0;0;1].*DD(1))
      + cross(2*W(:,:,1),TT(1:3,1:3,1)\([0;0;1].*d(1))) +
      cross(Wdot(:,:,1),Teff(1:3,1:3,1)\TT(1:3,4,1)) +
      cross(W(:,:,1),cross(W(:,:,1),Teff(1:3,1:3,1)\TT(1:3,4,1))));
          else
             W(:,:,i) = TT(1:3,1:3,1)\W(:,:,i-1);
             Wdot(:,:,i)= TT(1:3,1:3,1)\Wdot(:,:,i-1);
             vdot(:,:,i) = simplify(TT(1:3,1:3,i)\ (vdot(:,:,i-1) +
      [0;0;1].*DD(i)) + cross(2.*W(:,:,i),TT(1:3,1:3,i)\([0;0;1].*d(i))) +
      cross(Wdot(:,:,i),Teff(1:3,1:3,i)\TT(1:3,4,i)) +
      cross(W(:,:,i),cross(W(:,:,i),Teff(1:3,1:3,i)\TT(1:3,4,i))));
          end
     end
end
for i = 1 : n
    vdashdot(:,:,i) = simplify(vdot(:,:,i) + cross(W(:,:,i),r(:,:,i)) +
     cross(W(:,:,i),cross(W(:,:,i),r(:,:,i))));
end
fprintf("Forward iteration done \n")
%% Newton Euler - Backward iteration
```

```
syms tau [1 n]
f = sym(zeros(3,n));
nn = sym(zeros(3,n));
F = sym(zeros(3,n));
N = sym(zeros(3,n));
for i = n:-1:1
    F(:,i) =  simplify(m(i).* vdashdot(:,:,i));
    N(:,i) = simplify(I(1:3,1:3,i)*Wdot(:,:,i) +
     cross(W(:,:,i),I(1:3,1:3,i)*W(:,:,i)));
    if i == n
        f(:,i) = F(:,n);
        nn(:,i) = cross(Teff(1:3,1:3,i)\TT(1:3,4,i)+
     Teff(1:3,1:3,i)\r(:,:,i),F(:,i))+ N(:,i);

    else
        f(:,i) = simplify(F(:,i) + TT(1:3,1:3,i+1)*f(:,i+1));
        nn(:,i) = simplify(TT(1:3,1:3,i+1)*nn(:,i+1)+
     cross(Teff(1:3,1:3,i)\TT(1:3,4,i),(TT(1:3,1:3,i+1)*f(:,i+1))) +
     cross(Teff(1:3,1:3,i)\TT(1:3,4,i)+ Teff(1:3,1:3,i)\r(:,:,i),F(:,i))+
     N(:,i));
    end

    if ques(i) == 'R'
        tau(i) = simplify(transpose(nn(:,i))*inv(TT(1:3,1:3,i))*[0;0;1]);
    else
        tau(i) = simplify(transpose(f(:,i))*inv(TT(1:3,1:3,i))*[0;0;1]);
    end
    fprintf("Backward iteration iteration %d done \n",i)
end
%% All the terms
%w
W0 = W00;
W1 = W(:,:,1);
W2 = W(:,:,2);
W3 = W(:,:,3);
W4 = W(:,:,4);
W5 = W(:,:,5);
W6 = W(:,:,6);
%wdot
Wdot0 = Wdot00;
Wdot1 = Wdot(:,:,1);
Wdot2 = Wdot(:,:,2);
Wdot3 = Wdot(:,:,3);
Wdot4 = Wdot(:,:,4);
Wdot5 = Wdot(:,:,5);
Wdot6 = Wdot(:,:,6);
%vdot
vdot0 = vdot00;
vdot1 = vdot(:,:,1);
vdot2 = vdot(:,:,2);
vdot3 = vdot(:,:,3);
vdot4 = vdot(:,:,4);
vdot5 = vdot(:,:,5);
vdot6 = vdot(:,:,6);
%vdashdot
```

```
vdashdot1 = vdashdot(:,:,1);
vdashdot2 = vdashdot(:,:,2);
vdashdot3 = vdashdot(:,:,3);
vdashdot4 = vdashdot(:,:,4);
vdashdot5 = vdashdot(:,:,5);
vdashdot6 = vdashdot(:,:,6);

F1 = F(:,1);
F2 = F(:,2);
F3 = F(:,3);
F4 = F(:,4);
F5 = F(:,5);
F6 = F(:,6);

N1 = N(:,1);
N2 = N(:,2);
N3 = N(:,3);
N4 = N(:,4);
N5 = N(:,5);
N6 = N(:,6);

%force
f1 = f(:,1);
f2 = f(:,2);
f3 = f(:,3);
f4 = f(:,4);
f5 = f(:,5);
f6 = f(:,6);
%eta
n1 = simplify(nn(:,1));
n2 = simplify(nn(:,2));
n3 = simplify(nn(:,3));
n4 = simplify(nn(:,4));
n5 = simplify(nn(:,5));
n6 = simplify(nn(:,6));
%Torque
tau1 = 0.001*tau(1);
tau2 = 0.001*tau(2);
tau3 = 0.001*tau(3);
tau4 = 0.001*tau(4);
tau5 = 0.001*tau(5);
tau6 = 0.001*tau(6);
fprintf("All torques found \n")
%MOI
I1 = I(:,:,1);
I2 = I(:,:,2);
I3 = I(:,:,3);
I4 = I(:,:,4);
I5 = I(:,:,5);
I6 = I(:,:,6);
%% Plotting the Graph: theta1
fprintf("Plotting the graphs of torques with respect to theta1 \n")
Angle1 = linspace(-pi/2,pi/2,100);
tor61 =
     subs(tau6,[th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5,th5
     ,thh5,theta6,th6,thh6],[0.03,0,1000,0,0,pi/3,0,0,pi/3,0,0,0,0,0,0,0,0]);
```

```
tor51 =
      subs(tau5,[th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5,th5
      ,thh5,theta6,th6,thh6],[0.03,0,1000,0,0,pi/3,0,0,pi/3,0,0,0,0,0,0,0,0]);
tor41 =
      subs(tau4,[th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5,th5
      ,thh5,theta6,th6,thh6],[0.03,0,1000,0,0,pi/3,0,0,pi/3,0,0,0,0,0,0,0,0]);
tor31 =
      subs(tau3,[th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5,th5
      ,thh5,theta6,th6,thh6],[0.03,0,1000,0,0,pi/3,0,0,pi/3,0,0,0,0,0,0,0,0]);
tor21 =
      subs(tau2,[th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5,th5
      ,thh5,theta6,th6,thh6],[0.03,0,1000,0,0,pi/3,0,0,pi/3,0,0,0,0,0,0,0,0]);
tor11 =
      subs(tau1,[th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5,th5
      ,thh5,theta6,th6,thh6],[0.03,0,1000,0,0,pi/3,0,0,pi/3,0,0,0,0,0,0,0,0]);

fig1 = figure;
plot(Angle1,subs(tor61,theta1,Angle1),'-m')
hold on
plot(Angle1,subs(tor51,theta1,Angle1),'-b')
plot(Angle1,subs(tor41,theta1,Angle1),'-k')
plot(Angle1,subs(tor31,theta1,Angle1),'-r')
plot(Angle1,subs(tor21,theta1,Angle1),'-y')
plot(Angle1,subs(tor11,theta1,Angle1),'-g')
hold off
xlabel('theta1')
ylabel('N-m')
legend('Joint6','Joint5','Joint4','Joint3','Joint2','Joint1','location','nort
      hwest')
%% Plotting the Graph: d2
length2 = linspace(0,2000,1000);
tor62 =
      subs(tau6,[theta1,th1,thh1,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5
      ,th5,thh5,theta6,th6,thh6],[-pi/4,0,0,10,0,pi/3,0,0,0,0,0,0,0,0,0,0,0]);
tor52 =
      subs(tau5,[theta1,th1,thh1,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5
      ,th5,thh5,theta6,th6,thh6],[-pi/4,0,0,10,0,pi/3,0,0,0,0,0,0,0,0,0,0,0]);
tor42 =
      subs(tau4,[theta1,th1,thh1,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5
      ,th5,thh5,theta6,th6,thh6],[-pi/4,0,0,10,0,pi/3,0,0,0,0,0,0,0,0,0,0,0]);
tor32 =
      subs(tau3,[theta1,th1,thh1,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5
      ,th5,thh5,theta6,th6,thh6],[-pi/4,0,0,10,0,pi/3,0,0,0,0,0,0,0,0,0,0,0]);
tor22 =
      subs(tau2,[theta1,th1,thh1,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5
      ,th5,thh5,theta6,th6,thh6],[-pi/4,0,0,10,0,pi/3,0,0,0,0,0,0,0,0,0,0,0]);
tor12 =
      subs(tau1,[theta1,th1,thh1,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,theta5
      ,th5,thh5,theta6,th6,thh6],[-pi/4,0,0,10,0,pi/3,0,0,0,0,0,0,0,0,0,0,0]);

fprintf("Plotting the graphs of torques with respect to d2 \n")
fig2 = figure;
plot(length2,subs(tor62,d2,length2),'-m')
hold on
plot(length2,subs(tor52,d2,length2),'-b')
plot(length2,subs(tor42,d2,length2),'-k')
```

```
plot(length2,subs(tor32,d2,length2),'-r')
plot(length2,subs(tor22,d2,length2),'-y')
plot(length2,subs(tor12,d2,length2),'-g')
hold off
xlabel('d2')
ylabel('N-m')
legend('Joint6','Joint5','Joint4','Joint3','Joint2','Joint1','location','west
    ')
%% Plotting the Graph: theta3
Angle3 = linspace(-pi/2,pi/2,100);
tor63 =
    subs(tau6,[theta1,th1,thh1,d2,D2,DD2,th3,thh3,theta4,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,0.1,0,pi/3,0,0,0,0,0,0,0,0]);
tor53 =
    subs(tau5,[theta1,th1,thh1,d2,D2,DD2,th3,thh3,theta4,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,0.1,0,pi/3,0,0,0,0,0,0,0,0]);
tor43 =
    subs(tau4,[theta1,th1,thh1,d2,D2,DD2,th3,thh3,theta4,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,0.1,0,pi/3,0,0,0,0,0,0,0,0]);
tor33 =
    subs(tau3,[theta1,th1,thh1,d2,D2,DD2,th3,thh3,theta4,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,0.1,0,pi/3,0,0,0,0,0,0,0,0]);
tor23 =
    subs(tau2,[theta1,th1,thh1,d2,D2,DD2,th3,thh3,theta4,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,0.1,0,pi/3,0,0,0,0,0,0,0,0]);
tor13 =
    subs(tau1,[theta1,th1,thh1,d2,D2,DD2,th3,thh3,theta4,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,0.1,0,pi/3,0,0,0,0,0,0,0,0]);
fprintf("Plotting the graphs of torques with respect to theta3 \n")
fig3 = figure;
plot(Angle3,subs(tor63,theta3,Angle3),'-m')
hold on
plot(Angle3,subs(tor53,theta3,Angle3),'-b')
plot(Angle3,subs(tor43,theta3,Angle3),'-k')
plot(Angle3,subs(tor33,theta3,Angle3),'-r')
plot(Angle3,subs(tor23,theta3,Angle3),'-y')
plot(Angle3,subs(tor13,theta3,Angle3),'-g')
hold off
xlabel('theta3')
ylabel('N-m')
legend('Joint6','Joint5','Joint4','Joint3','Joint2','Joint1','location','sout
    hwest')
%% Plotting the Graph: theta4
Angle4 = linspace(-pi/2,pi/2,100);
tor64 =
    subs(tau6,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0.1,0,0,0,0,0,0,0]);
tor54 =
    subs(tau5,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0.1,0,0,0,0,0,0,0]);
tor44 =
    subs(tau4,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0.1,0,0,0,0,0,0,0]);
tor34 =
    subs(tau3,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,th4,thh4,theta5,th5
    ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0.1,0,0,0,0,0,0,0]);
```

```
tor24 =
      subs(tau2,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,th4,thh4,theta5,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0.1,0,0,0,0,0,0,0]);
tor14 =
      subs(tau1,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,th4,thh4,theta5,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0.1,0,0,0,0,0,0,0]);
fprintf("Plotting the graphs of torques with respect to theta4 \n")
fig4 = figure;
plot(Angle4,subs(tor64,theta4,Angle4),'-m')
hold on
plot(Angle4,subs(tor54,theta4,Angle4),'-b')
plot(Angle4,subs(tor44,theta4,Angle4),'-k')
plot(Angle4,subs(tor34,theta4,Angle4),'-r')
plot(Angle4,subs(tor24,theta4,Angle4),'-y')
plot(Angle4,subs(tor14,theta4,Angle4),'-g')
hold off
xlabel('theta4')
ylabel('N-m')
legend('Joint6','Joint5','Joint4','Joint3','Joint2','Joint1','location','west
      ')
%% Plotting the Graph: theta5
Angle5 = linspace(-pi/2,pi/2,100);
tor65 =
      subs(tau6,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0,0,0,0.1,0,0,0,0]);
tor55 =
      subs(tau5,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0,0,0,0.1,0,0,0,0]);
tor45 =
      subs(tau4,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0,0,0,0.1,0,0,0,0]);
tor35 =
      subs(tau3,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0,0,0,0.1,0,0,0,0]);
tor25 =
      subs(tau2,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0,0,0,0.1,0,0,0,0]);
tor15 =
      subs(tau1,[theta1,th1,thh1,d2,D2,DD2,theta3,th3,thh3,theta4,th4,thh4,th5
      ,thh5,theta6,th6,thh6],[-pi/4,0,0,1000,0,0,pi/3,0,0,0,0,0,0.1,0,0,0,0]);
fprintf("Plotting the graphs of torques with respect to theta5 \n")
fig5 = figure;
plot(Angle5,subs(tor65,theta5,Angle5),'-m')
hold on
plot(Angle5,subs(tor55,theta5,Angle5),'-b')
plot(Angle5,subs(tor45,theta5,Angle5),'-k')
plot(Angle5,subs(tor35,theta5,Angle5),'-r')
plot(Angle5,subs(tor25,theta5,Angle5),'-y')
plot(Angle5,subs(tor15,theta5,Angle5),'-g')
hold off
xlabel('theta5')
ylabel('N-m')
legend('Joint6','Joint5','Joint4','Joint3','Joint2','Joint1','location','west
      ')
% Maximum torque
maxim1 = min(eval(subs(tor14,linspace(-pi/2,pi/2,100))));
```

```
fprintf("The maximum torque required for joint1, after looking at all graphs
     is : %f  \n",abs(maxim1))
maxim3 = min(eval(subs(tor34,linspace(-pi/2,pi/2,100))));
fprintf("The maximum torque required for joint3, after looking at all graphs
     is : %f  \n",abs(maxim3))
maxim4 = min(eval(subs(tor43,linspace(-pi/2,pi/2,100))));
fprintf("The maximum torque required for joint4, after looking at all graphs
     is : %f  \n",abs(maxim4))
maxim5 = min(eval(subs(tor51,linspace(-pi/2,pi/2,100))));
fprintf("The maximum torque required for joint5, after looking at all graphs
     is : %f  \n",abs(maxim5))
maxim6 = min(eval(subs(tor62,linspace(-pi/2,pi/2,100))));
fprintf("The maximum torque required for joint6, after looking at all graphs
     is : %f  \n",abs(maxim6))
```