University of Bristol

Faculty of Engineering

Robotic Fundamentals

Serial and Parallel Robot Kinematics

NISHANT RAMAKURU

Supervisor:
Aghil Jafari

Bristol, December 2019

Abstract

The use of manipulator robots in industry has risen exponentially in the last decade due to advances in the control algorithms that enable complex tasks to be automated. These robots come in different types and sizes, and usually the structures and the number of joints associated with the robot determine its type. The aim of this assignment is first to simulate a Lynxomotion manipulator robot, the robot consists of five revolute joints and one prismatic joint at the end effector to mimic the grapping action. The second task is to model a 3RRR parallel robot which consists of a closed-loop kinematic chain where the end-effector is positioned in the base by varies independent kinematic chains. Parallel robots have grasped significant interest due to their high rigidity and accuracy as well as the ability of withstanding high payloads. Ones the mechanism of these robots is described, the report will explain the methods taken to model the kinematics as well as workspace analysis performed.

Table of Contents

1 Introduction

Manipulator robots are an electronically controlled mechanisms that are composed from varies parts such as joints, links and grippers. These robots interact with the environment and perform tasks that are either repetitive or humans are unable of doing. Robots manipulator or as may be referred to as robot arms are mainly used in industry and manufacturing. To control these robots, kinematic modelling of such structure must be carried out. There are two main problems with kinematic modelling, the forward and inverse kinematics. Forward-kinematics is concerned about finding the position of the end-effector when each joint value is known and the inverse kinematics is concerned about finding each joint variable when the position and orientation of the end-effector is given as a target point in the Cartesian space[1]. This project is done in accordance with the coursework for the Robotics Fundamentals module. The objective of this project is to demonstrate the principles of forward and inverse kinematics of both serial and parallel robotic structures. This is done through a sequence of assignments split up into parts. This document outlines the methods used to derive both the forward kinematics and inverse kinematics for a 5 Degrees of Freedom Lynxmotion robot arm. From this a MATLAB model of the arm will be created. The model will allow us to perform tasks such as plotting the workspace and planning movement sequences. Finally results from the model will be tested on the robot.

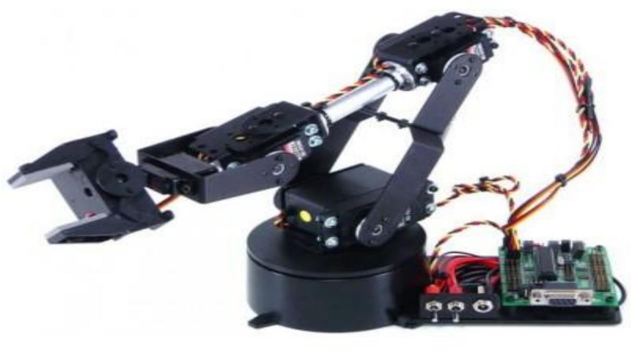

Figure 1: 5DOF Lynxmotion robot arm [2]

In this report, modelling and simulation of two types of robot manipulators are explored, these types are serial and parallel robots. The structure by which these robots are constructed determine its type. Serial robots consist of multiple links that serially linked, and the end-effector is at the end of the last link. However, parallel robots are constructed from links that are parallelly linked and the end-effector is the base of the plane that connected these links together.

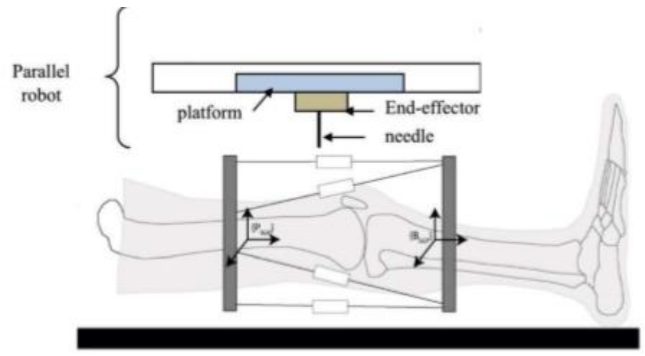

Figure 2: Needle positioning parallel robot

The report starts by explaining the method taken to model Lynxmotion robot arm and the corresponding mathematical analysis are then derived, then the methodology moves on to explain the steps taken to determine the inverse kinematics of the parallel robot used as a needle positioning mechanism. The results section will provide figures showing the obtained results from modelling both serial and parallel robots. Finally, the discussion will provide statements about the findings and a conclusion showing possible improvements will be explored.

1.1 Objectives

1.1.1 Part 1

- Derive DH representation of the forward kinematics of Lynx motion robot arm.
- Analyse the workspace of the end-effector and plot 2D and 3D views of the workspace.
- Derive the inverse kinematics for the Lynxmotion robot
- Plan a task in MATLAB with at least 5 positions

- Solve the inverse kinematics of these 5 positions
- Create an appropriate plot and animation for the motion of the robot
- Implement three different trajectories
- Implement a free motion between the points
- Implement straight line trajectory between points
- Aet an obstacle between two points
- Implement an object avoidance trajectory

1.1.2 Part 2

- Solve and implement parallel robot inverse kinematics
- Calculate the joint coordiantes from the cartesian parameters of the platform centre.
- Plot the parallel robot workspace for a given orientation.

2 Methodology and Analysis

The study of kinematics is concerned about describing motion without taken the forces that cause it into account. It is concerned about the position and its higher derivatives such as velocity and acceleration [3]. There are two types of solutions to the kinematics regarding robotic arms, the first is the Forward Kinematics FK where the position of the end-effector is determined by using the values of the joint angles. However, the inverse kinematic solution is concerned about the value of each joint angle that will cause the end-effector of the manipulator to be in certain position. Therefore, forward kinematic can be thought of as the transformation from joint space to the cartesian space whereas the inverse-kinematics solution is the transformation from the cartesian space to the joint space.[3]

2.1 Part 1

In this part of the report, the method taken to model Lynxmotion robot arm is explored. This section shows the necessary steps taken to model kinematics of the robot arm.

2.1.1 Forward Kinematics

As previously mentioned, the forward kinematics problem is concerned about the position and orientation of the end-effector of the manipulator when the joint variables are known. Therefore, the objective of the forward kinematics is to find the cumulative effect of each joint variable to the position and orientation of the end-effector.

The first step to determine the forward kinematics of the robotic arm is to assign a coordinate frame for all the joints. As shown below the Lynxomotion robot arm has five rotational joints and a sliding grip, Joint 1 represents the shoulder providing rotation around z1 in the x1y1 plane. The second joint represents the upper arm of the robot having rotation around z2 in the x2y2 plane. Furthermore, Joint 3 acts as the forearm and joint 4 represents the wrist of the manipulator, these provide angular rotation around z3 and z4 in the x3y3 and x4y4 planes respectively. The last joint represents the grip.

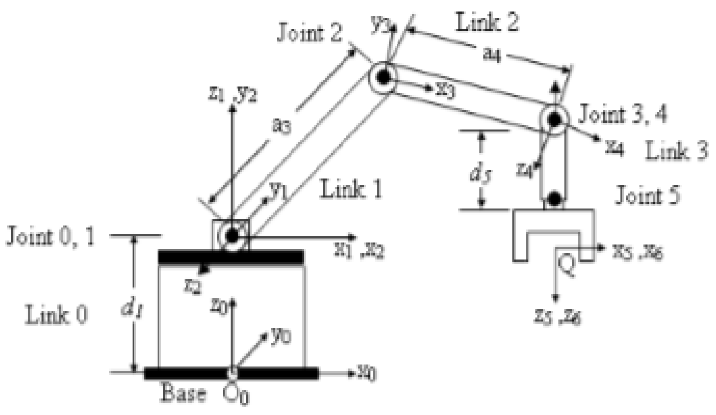

Figure 3 Robot Arm Frame Assignment [4]

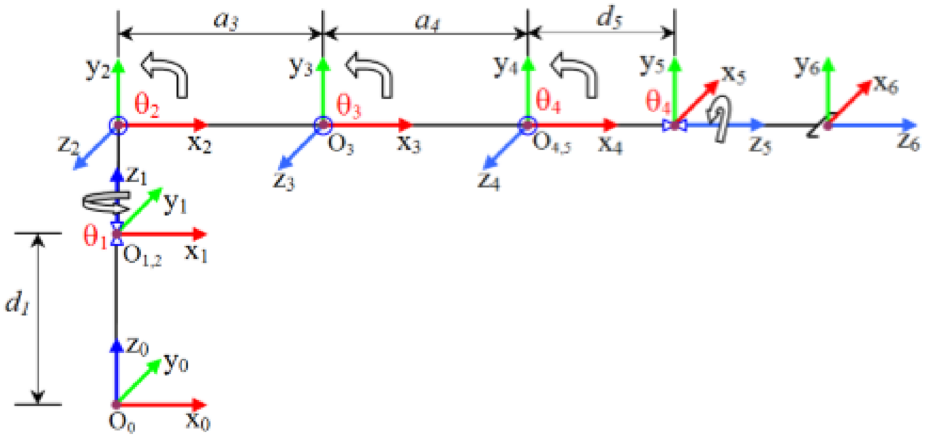

Figure 4 Assigning coordinate frames.

The standard method of finding the forward kinematics is the homogeneous transformation from the base frame to the end-effector frame. The homogeneous matrix which enables transformation between joints is shown below with R_n^0 representing the rotation or the orientation of the end-effector from the base of the manipulator and d_n^0 is the translation or the coordinates of the end-effector. [4]

However, in this report, the Denavit-Hartenberg method of finding the end-effector position and orientation is used. DH method is mainly used in industry, it enables successful determination of the orientation and the position of the end-effector when joint angles are known. Please note that the DH table below was constructed using the proximal convention. [4]

2.1.1.1 DH table

Link	α_{i-1}	a_{i-1}	d_i	θ_{i-1}
1	**0**	**0**	$d1$	θ_1
2	**Pi/2**	**0**	**0**	θ_2
3	**0**	$a3$	**0**	θ_3
4	**0**	$a4$	**0**	$\theta_4 - pi/2$
5	**-pi/2**	**0**	$d5$	θ_5
6	**0**	**0**	**0**	**Gripper**

Table 1 DH parameter table.

The transformation from the base frame and the gripper frame can be found by substituting the parameters from the DH table into the transformation matrices T1 to T6 shown below.

$$T_1^0 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_2^1 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_{\theta_2} & c_{\theta_2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & a_3 \\ s_{\theta_3} & c_{\theta_3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_4^3 = \begin{bmatrix} c_{\theta_4} & -s_{\theta_4} & 0 & a_4 \\ s_{\theta_4} & c_{\theta_4} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} c_{\theta_5} & -s_{\theta_5} & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ s_{\theta_5} & c_{\theta_5} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{Gripper} = \begin{bmatrix} c_{\theta_6} & -s_{\theta_6} & 0 & 0 \\ s_{\theta_6} & c_{\theta_6} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 5 Transformation matrices from base frame to end-effector frame.

The solution of the forward kinematics of the robot is obtained by multiplying these transformation matrices T_1^0 x T^1_2 x T^2_3 x T^3_4 x T^4_5 x $T^5_{Gripper}$.

$$T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 = \begin{bmatrix} n_x & o_x & a_x & d_x \\ n_y & o_y & a_y & d_y \\ n_z & o_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 6 Forward kinematics solution

Where the first three columns in the obtained matrix correspond to the orientation of the end effector and the last column represents the position of the end-effector.

2.1.2 Inverse Kinematics

It is vital for the robot to determine the corresponding angle values when a target position in the Cartesian space is given. To do this, the inverse kinematics of the robot must be solved, the inverse kinematic problem is concerned about determining the joint angle values when the position of the end-effector is known.

2.1.2.1 Geometric Approach

The most straight forward way of solving the inverse kinematics of the robot arm is the geometric approach. In this approach, the problem is divided into multiple 2D problems and then a solution is derived for each. For this 5DOF robot, the problem is divided into two separate problems, the first solution would be to solve for $\theta 1$ as viewed from above in the XY plane. The second problem would be to solve for $\theta 2,\ \theta 3\ and\ \ \theta 4$ by considering the manipulator structure as a 2D problem in the XZ plane, where X is the hypotenuse formed by the position of the end-effector in the XY plane.

- **Find $\theta 1$**

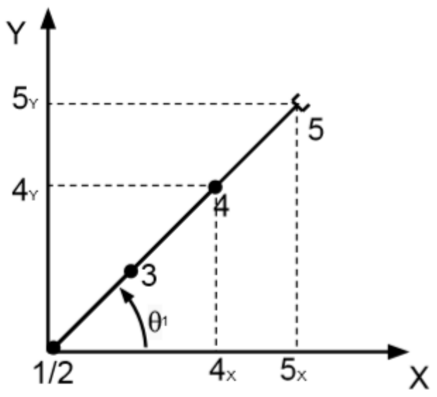

Figure 7 Robot arm in XY plane.

From the figure above, $\theta 1$ can be found by using the trigonometric function arctan2 as shown in the equation below.

θ1=arctan2(5y,5x)

- **Find $\theta 3$**

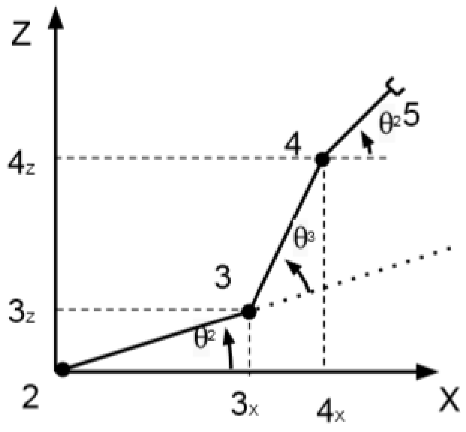

Figure 7 Robot arm in XZ plane.

To find the value for $\theta 2$ and $\theta 3$, the position of the fourth joint on the XZ plane must first be known. Which means that the position of joint4 (X4, Z4) in the XZ plane should be calculated. This is done using the target position and the length of link 4 as well as the target pitch angle φ.

$$4x = 5x - length(4) * cos(\varphi) \qquad \textit{Equation 1}$$

$$4z = 5z - length(4) * sin(\varphi) \qquad \textit{Equation 2}$$

The work-out the value for X, the lengths of the adjacent sides of the two triangles formed by the second link and its angle and the third link and its angle is summed. Also, the value of Z is determined the same way but using the opposite side.

$$4x = Length(3) * cos(\theta 2 + \theta 3) + length(2) * cos(\theta 2) \qquad \textit{Equation 3}$$

$$4z = Length(3) * sin(\theta 2 + \theta 3) + length(2) * sin(\theta 2) \qquad \textit{Equation 4}$$

The value of $\theta 3$ can be found by the summation of squaring both sides of the previous two equations.

$$(4x)^2 = (Length(3) * cos(\theta 2 + \theta 3) + length(2) * cos(\theta 2))^2 \qquad \textit{Equation 5}$$

$$(4z)^2 = (Length(3) * sin(\theta 2 + \theta 3) + length(2) * sin(\theta 2))^2 \qquad \textit{Equation 6}$$

$$(4x)^2 + (4z)^2 = (Length(3) * cos(\theta 2 + \theta 3) + length(2) * cos(\theta 2))^2 + (Length(3) * sin(\theta 2 + \theta 3) + length(2) * sin(\theta 2))^2 \qquad \textit{Equation 7}$$

Which yields to

$$(4x)^2 + (4z)^2 = length(3)^2 + length(2)^2 + 2 * length(3) * length(2) * cos(3)$$

Equation 8

And then rearranged to give,

$$cos(\theta 3) = \frac{-length(2)^2 - length(3)^2 + (4x)^2 + (4z)^2}{2 * length(2) * length(3)} \qquad \textit{Equation 9}$$

- **Find $\boldsymbol{\theta 2}$**

As $\theta 3$ is now known, the next step is to calculate $\theta 2$. Multiply 4x by cos(theta2) and 4z by sin(theta2).

$$cos(\theta 2) * 4x = Length(3) * cos(\theta 2 + \theta 3) * cos(\theta 2) + length(2) * cos(\theta 2)^2 \qquad \textit{Equation 10}$$

$$Sin(\theta 2) * 4z = Length(3) * sin(\theta 2 + \theta 3) * sin(\theta 2) + length(2) * sin(\theta 2)^2 \qquad \textit{Equation 11}$$

By adding these two equations,

$$cos(\theta 2) * 4x + Sin(\theta 2) * 4z \qquad \textit{Equation 12}$$

$$= Length(3) * cos(\theta 2 + \theta 3) * cos\ (\theta 2) + length(2) * cos\ (\theta 2)^2 + Length(3) * sin(\theta 2 + \theta 3) * sin\ (\theta 2) + length(2) * sin\ (\theta 2)^2$$

Which yields to

$$cos(\ \theta 2) * 4x + sin(\theta 2) * 4z = length(3) * cos(\theta 3) + length(2) \qquad \textit{Equation 13}$$

Now, the same approach is repeated but by multiplying those equations by -sin ($\theta 2$) and cos ($\theta 2$).

Which yields to,

$$cos(\theta 2) * ((4x)^2 + (4z)^2) = 4x\big(length(3) * cos(3) + length(2)\big) + length(3) * sin(\theta 3) * 4z \qquad \textit{Equation 14}$$

Then rearrange to give,

$$cos(\theta 2) = \frac{4x\big(length(3)*cos(3)+length(2)\big)+length(3)*sin(3)*4z}{(4x)^2+(4z)^2} \qquad \textit{Equation 15}$$

- **Find $\boldsymbol{\theta 4}$**

As θ2, θ3 and the pitch angle ϕ are all known, θ4 can be found using the following equation.

$$\theta 4 = \ \phi - (\theta 2 + \theta 3) - 90 \qquad \textit{Equation 16}$$

2.2 Part 2

In this part of the report, the method taken to model a parallel 3RRR robot is explored, the robot consists of a mobile platform and three RRR chains that are connected to a fixed base. Each chain consists of three rotational joints.

Let P (x,y) be the position and φ is the orientation of the end-effector. Allow O to be the origin of the fixed frame and let A, B and C be the rotational articulation of each leg of the robot. Let I_A be the length of link A and I_B the length of link B.

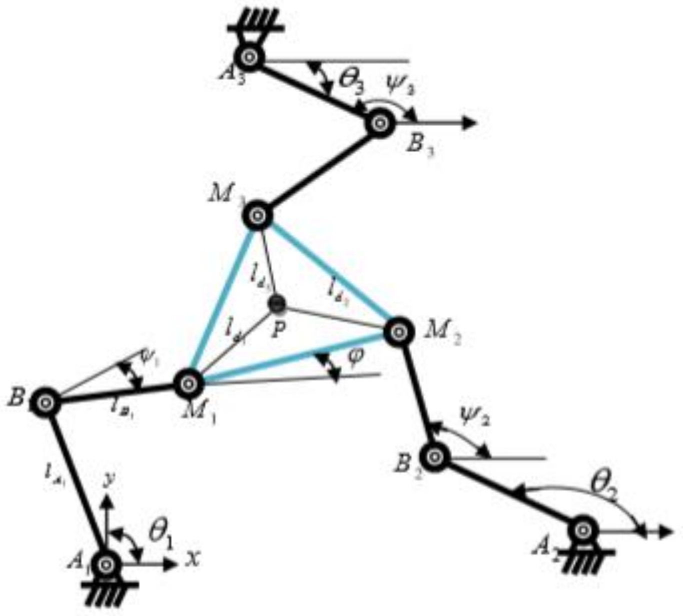

Figure 8 Geometric description of 3RRR parallel robot.[5]

2.2.1 Inverse kinematic model of the parallel 3RRR robot.

To derive the inverse kinematics, the closed loop closure equation of the robot is given as follows;

$$OP = OA_i + A_iB_i + B_iM_i + M_iP$$ *Equation 17*

$$P_x = l_{Ai}\cos(\theta_i) + l_{Bi}\cos(\psi_i) + l_{di}\cos(\sigma_i + \varphi) + x_{Ai}$$ *Equation 18*

$$P_y = l_{Ai}\sin(\theta_i) + l_{Bi}\sin(\psi_i) + l_{di}\sin(\sigma_i + \varphi) + y_{Ai}$$ *Equation 19*

To find the inverse kinematics, these equations are rewritten as follows;

$$K_i \sin\theta + F_i \cos\theta = E_i$$

Equation 20

Where

$$E_i = P_x{}^2 + P_y{}^2 + l_{Ai}{}^2 + l_{Bi}{}^2 + l_{di}{}^2 + x_{Ai}{}^2 + y_{Ai}{}^2 - 2P_x l_{di}\cos(\sigma_i + \varphi) - 2P_x x_{Ai} + 2l_{di} x_{Ai}\cos(\sigma_i + \varphi) - 2P_y l_{di}\sin(\sigma_i + \varphi) - 2P_y y_{Ai} + 2l_{di} y_{Ai}\sin(\sigma_i + \varphi)$$

Equation 21

$$F_i = -2l_{Ai} + 2l_{Ai} l_{di}\cos(\sigma_i + \varphi) + 2l_{Ai} x_{Ai}$$

Equation 22

$$K_i = -\left[-2P_y l_{Ai} + 2l_{Ai} l_{di}\sin(\sigma_i + \varphi) + 2l_{Ai} y_{Ai}\right]$$

Equation 23

Therefore, the inverse kinematic solution is;

$$\theta_i = A\tan 2(K_i, F_i) \pm A\tan 2(\sqrt{(K_i{}^2 + F_i{}^2 - E_i{}^2)}, E_i)$$

Equation 24

The obtained equation above can be used for any number of legs by only varying the subscript in the equation. There are two possible solutions for the inverse kinematics of this robot, which are given by the solution of the above equation. When the position and the orientation of the end-effector is outside the workspace, the solution to this equation will be imaginary and thus has no real solution.

3 Results and Discussion

3.1 Part I

We use previously calculated transformation matrix to find the position of each joint and end effector by plotting lines between the points. The link lengths used are

L1 = 5 cm   L2 = 5 cm   L3 = 5 cm   L4 = 3 cm   L5 = 0cm

3.1.1 A1: Forward kinematics

After designing the robotic arm, random angles were fed into the DH tables to alter the transformation matrices, which result in different orientation of the arm.

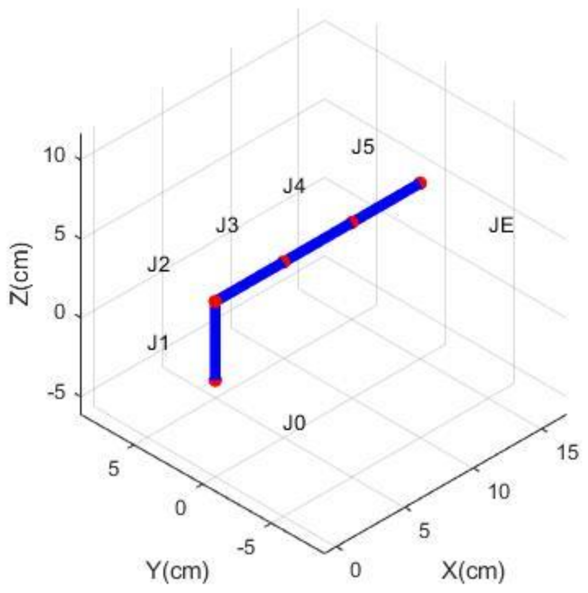

Figure 9 MATLAB Robot arm Model

3.1.2 A2: Workspace Analysis

Workspace describes all possible points that can be accessed by the end effector of the robotic arm with all possible orientations.

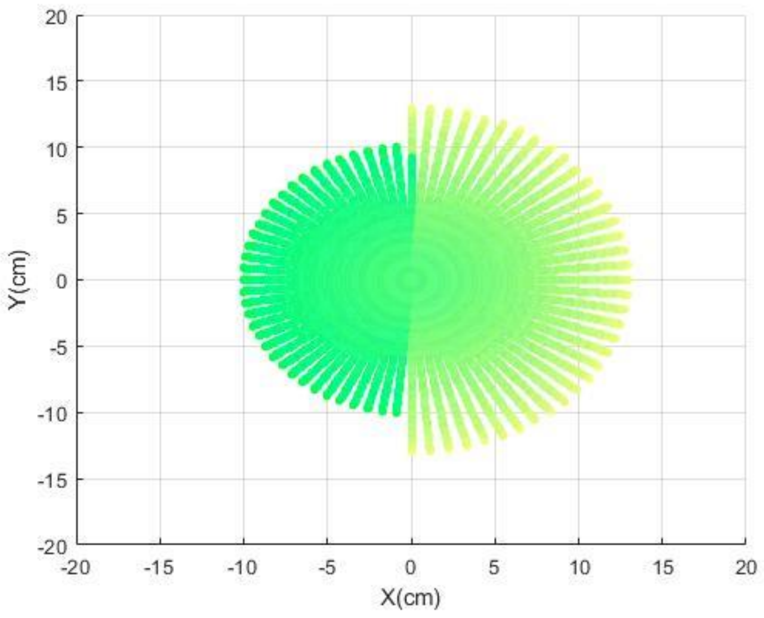

Figure 10 Top view of workspace

In order to plot the workspace, we need to analyse the range of possible angles. They are as follows

***µ*1 = -90 to 90 *µ*2 = 0 to 135 *µ*3 = 0 to -135 *µ*4 = 0 to -180.**

***µ*5** has not been changed as variation in this angle will not result in change in position in the end effector.

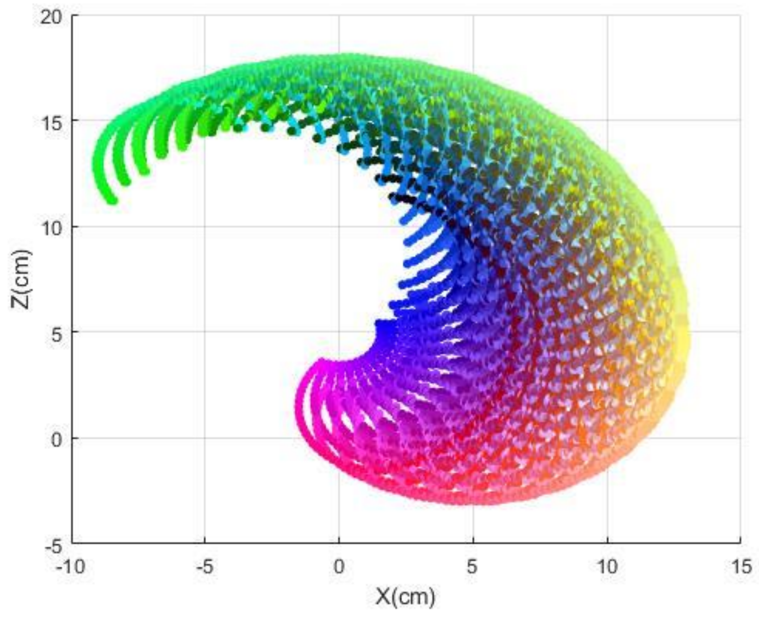

Figure 11 Vertical cross section of workspace

The joint plots are colour coded so that we can analyse the work space for each joint and it is easier for us to interpret the plots.

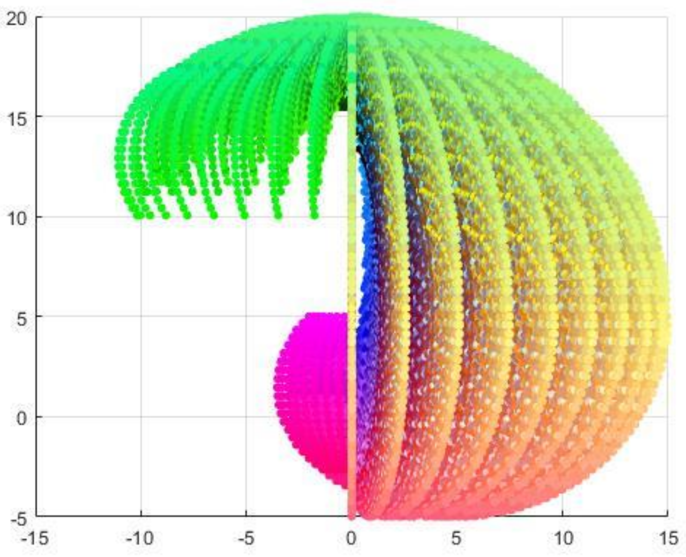

Figure 12 Side view of workspace

Now we need to iterate the forward kinematics model through all possible angles for all joints. The plot for the end effector will constitute as workspace.

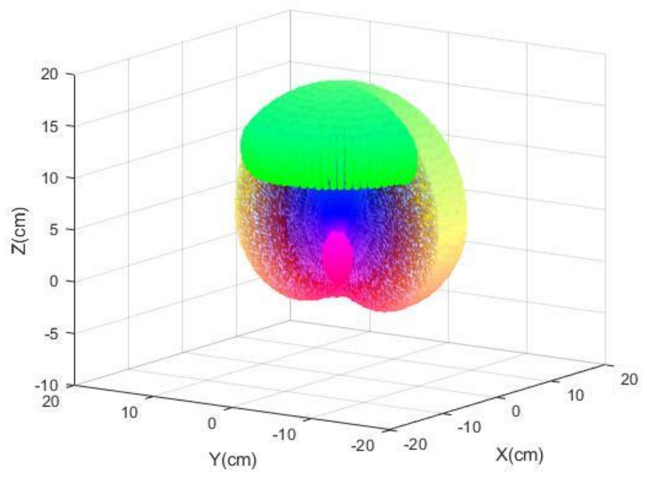

Figure 13 Isometric view of workspace

As it can be understood from the plots as we move away from the origin, the accessibility of points increases. Whereas around the origin, we can see a cavity due to physical constraints for links of the arm.

3.1.3 A3: Inverse Kinematics

The inverse kinematics is solved using geometric model. Each joint corresponds to a theta angle. Random angles are given to joints and fed into the DH table. After computing the position of end effector. The coordinates of the end effector are fed into the inverse kinematics model and it is observed that the angles obtained are the angles that were given in the first place.

Angles fed: $\boldsymbol{\theta 1}$ = 0 $\boldsymbol{\theta 2}$ = 60 $\boldsymbol{\theta 3}$ = -30 $\boldsymbol{\theta 4}$ = 90 $\boldsymbol{\theta 5}$ = 0

Position of End effector: $\boldsymbol{X}$ = 4.23 $\boldsymbol{Y}$ = 1.42 **Z** = 10.33

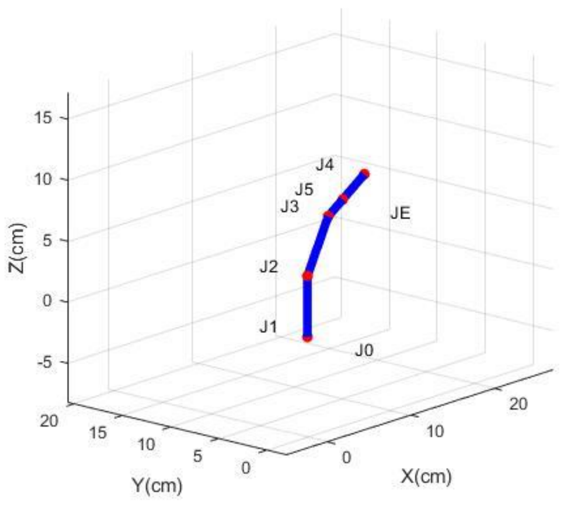

Figure 14

$\boldsymbol{\theta 1}$ = 0 $\boldsymbol{\theta 2}$ = 60 $\boldsymbol{\theta 3}$ = -30 $\boldsymbol{\theta 4}$ = 90 $\boldsymbol{\theta 5}$ = 0

This process was tested using a set of different inputs and the results were similar. Hence, we can confirm the working of Inverse kinematics function.

Angles fed: $\boldsymbol{\theta 1}$ = 90 $\boldsymbol{\theta 2}$ = 30 $\boldsymbol{\theta 3}$ = -45 $\boldsymbol{\theta 4}$ = 120 $\boldsymbol{\theta 5}$ = 0

Position of End effector: $\boldsymbol{X}$ = 9.30 $\boldsymbol{Y}$ = 11.28 **Z** = 4.08

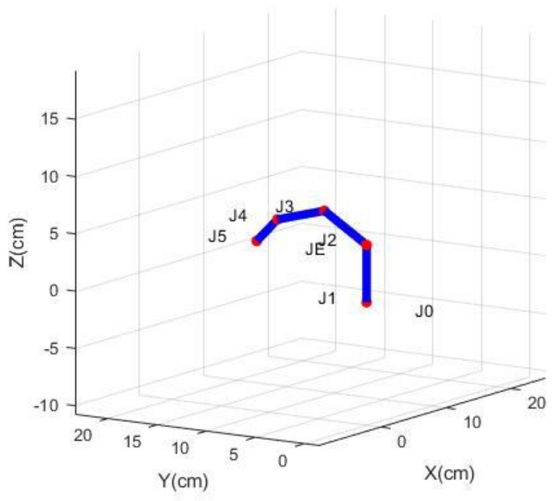

Figure 15

$\boldsymbol{\theta 1}$ = 90 $\boldsymbol{\theta 2}$ = 30 $\boldsymbol{\theta 3}$ = -45 $\boldsymbol{\theta 4}$ = -120 $\boldsymbol{\theta 5}$ = 0

3.1.4 B1: Forward Kinematics Test

Random joint angles were given to the arm to check and verify the position of the end effector. The angles used were simple to make the calculations easier and the position of the end effector was validated in all scenarios.

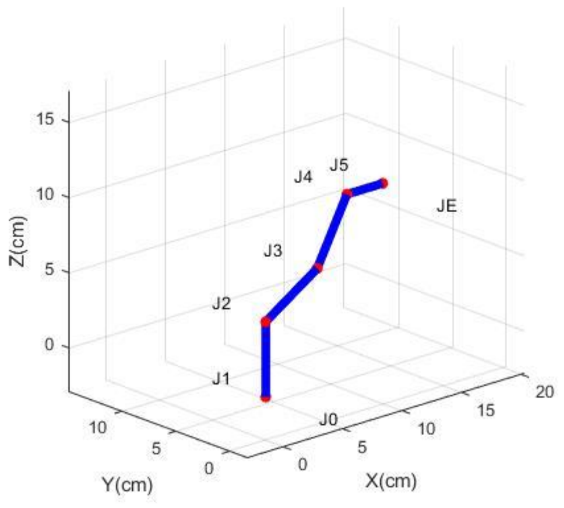

Figure 16 MATLAB Forward Kinematics (1)

$\boldsymbol{\theta 1}$ = 0 $\boldsymbol{\theta 2}$ = 30 $\boldsymbol{\theta 3}$ = 30 $\boldsymbol{\theta 4}$ = -150 $\boldsymbol{\theta 5}$ = 0

Figure below shows the arm orientation for different set of joint angles.

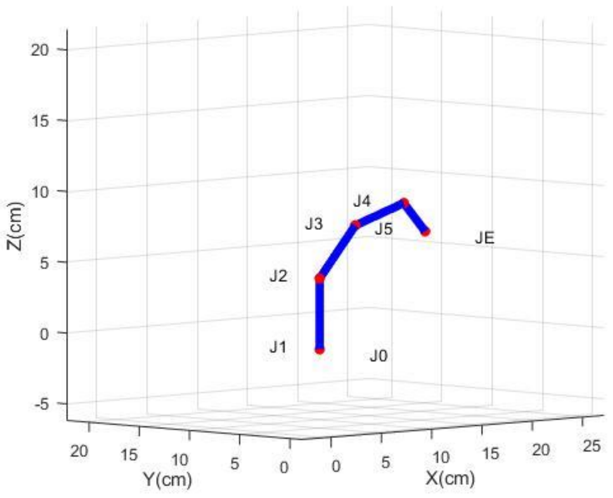

Figure 17 MATLAB Forward Kinematics (2)

$\boldsymbol{\theta 1}$ = 0 $\boldsymbol{\theta 2}$ **=** 45 $\boldsymbol{\theta 3}$ = -30 $\boldsymbol{\theta 4}$ = -150 $\boldsymbol{\theta 5}$ = 0

3.1.5 B2: Inverse Kinematics Test

The task considered is to plot an octagon. A regular octagon was plotted and the points of the vertices were fed the inverse kinematics function. The plots show how the arm changes the joint angles in order to move alone the edges of the octagon.

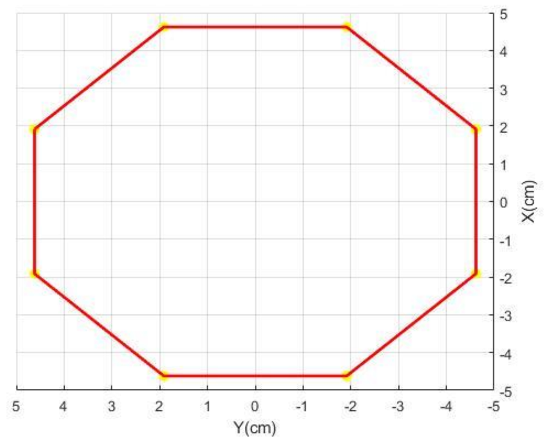

Figure 18 Task Plot Octagon

The vertices of the octagon were fed into the inverse kinematics function to obtain the joint angles. Once we computed the joint angles, they were used to create the DH table and find the transformation matrices. The plots for the joints and links are shown in the figure below.

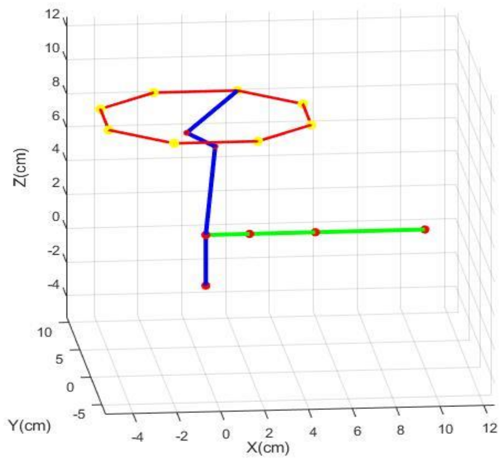

Figure 19

The code snippet for the task was run in a loop to check if the arm continued to move along the trajectory and go back to the first point after the last point.

As expected, the arm moved along the entire octagonal path till it reached the end of the loop intervals.

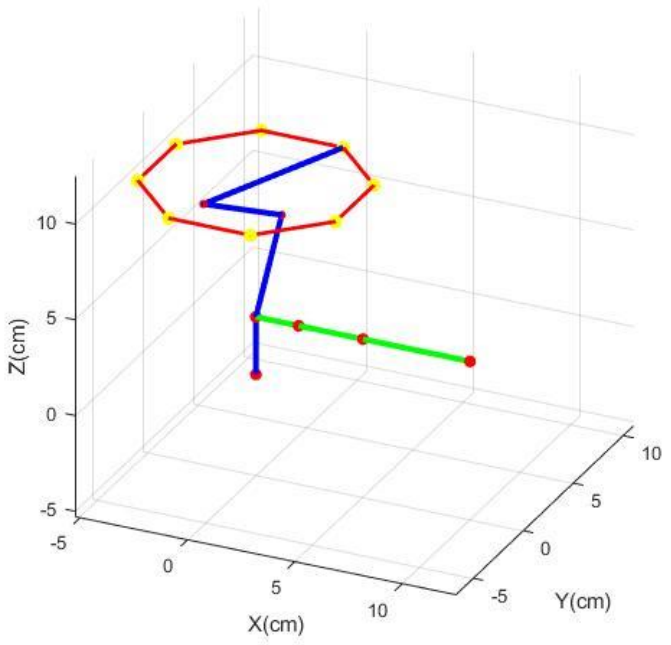

Figure 20

3.1.6 B3: Trajectories

The task taken above does consist of a straight-line trajectory. When the end effector moves along the edges of the octagon it follows a straight line.

For obstacle avoidance, a cube was placed in a straight-line path. The plots suggest how the arm follows a secondary path in order to avoid collision with the cube. The points of the path were fed into the inverse kinematics function of the arm to derive the join angles. These angles were passed in a loop to the see the motion of the arm following the secondary path.

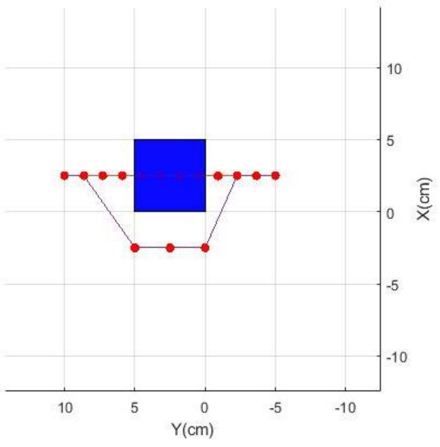

Figure 21

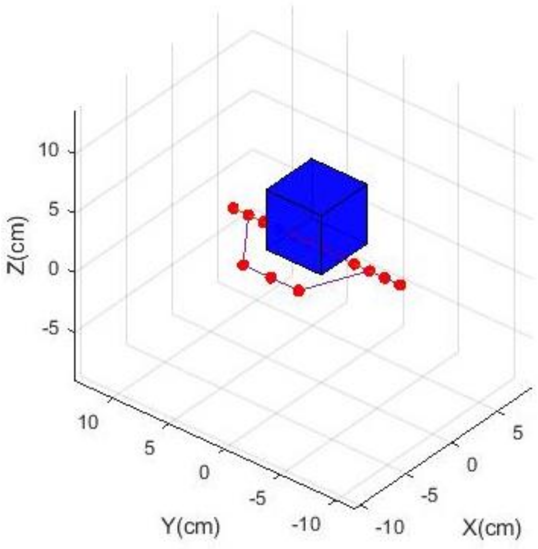

Figure 22

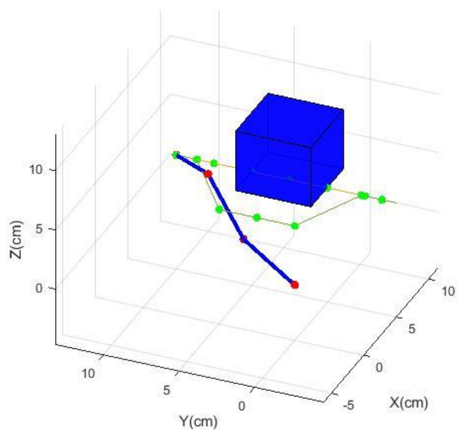

Figure 23

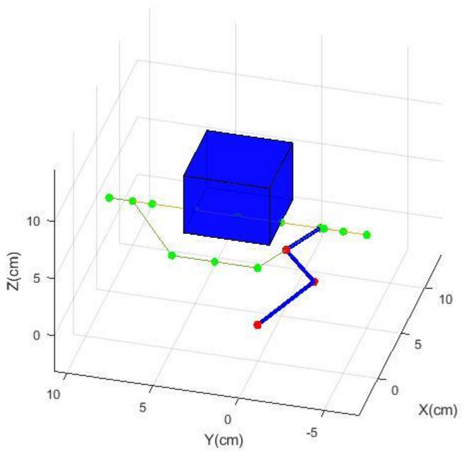

Figure 24

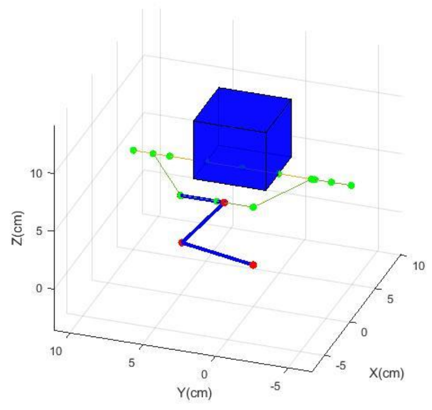

Figure 25

In the plots above you can see the end effector of the arm moving along the secondary path.

3.1.7 Inverse Kinematics

Firstly, a function for workspace was written to determine if a given point will be in the workspace or not. Using the calculations mentioned in the above section. Based on the position of the end effector the end effectors for each arm (1, 2, 3). Based on the position of these three end effectors the problem boils down to a 2-link arm. Using geometric approach, the joint angles are calculated.

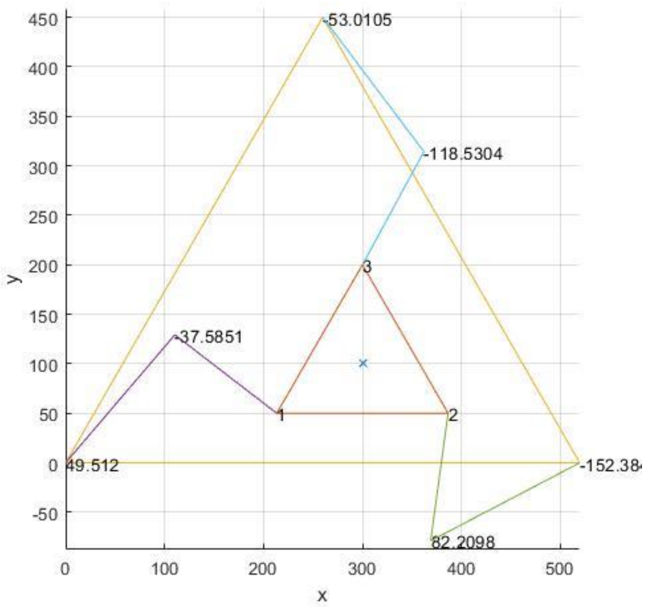

Figure 26

Position of End effector:	$\boldsymbol{X}$ = 300	$\boldsymbol{Y}$ = 100	$\boldsymbol{alpha}$ = 0
Angles fed:	$\boldsymbol{\theta 11}$ = 49.51	$\boldsymbol{\theta 12}$ = 37.58	
	$\boldsymbol{\theta 21}$ = -152.38	$\boldsymbol{\theta 22}$ = 82.20	
	$\boldsymbol{\theta 31}$ = 53.01	$\boldsymbol{\theta 32}$ = -118.53	

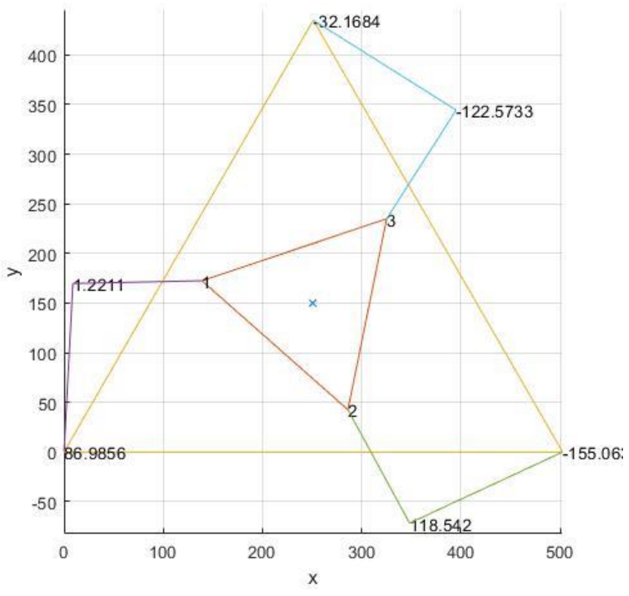

Figure 27

Position of End effector:	$\boldsymbol{X}$ = 250	$\boldsymbol{Y}$ = 150	$\boldsymbol{alpha}$ = 45
Angles fed:	$\boldsymbol{\theta 11}$ = 86.98	$\boldsymbol{\theta 12}$ = 1.22	
	$\boldsymbol{\theta 21}$ = 155.06	$\boldsymbol{\theta 22}$ = 118.54	
	$\boldsymbol{\theta 31}$ = -32.16	$\boldsymbol{\theta 32}$ = -122.57	

3.1.8 Workspace

Work space for each joint end effector is plotted before plotting the workspace if the end effector. As expected, the workspaces of the joint ends appear to be in circular form, where the radius of the circle will be the combined length of the links. The plots for three circles are similar hence they are colour coded to make it easier on the eye.

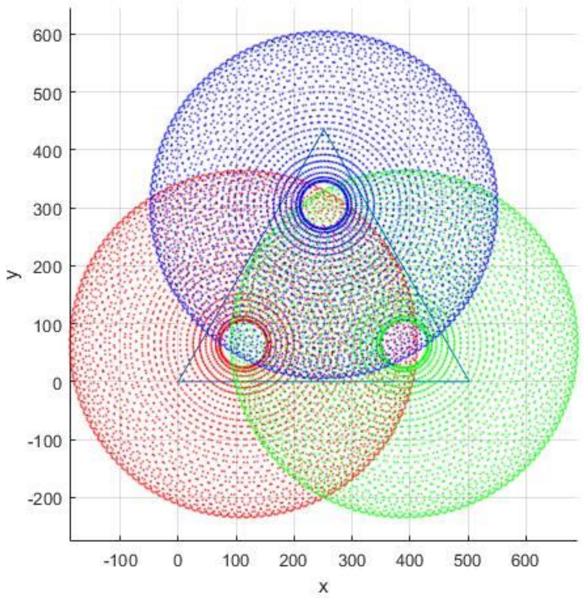

Figure 28

Workspace for the end effector was plotted for two parameters of alpha 0 and 45 respectively. As we can observe when the alpha is 0. There are cavities near the triangle joints caused by physical constraints of the planar robot.

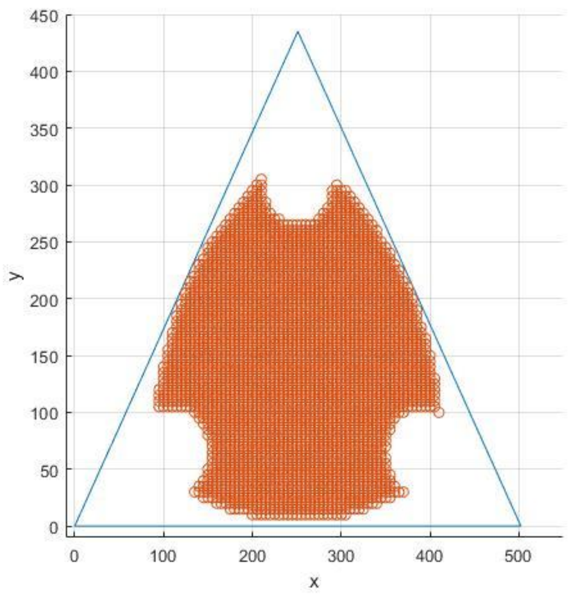

Figure 29

However, this is not observable when the alpha value is 45. Which is expected, given the tilt the end effector will not be able to venture as lose to the triangle vertices and hence only will be able to access the more central region of the platform, which can be observed from the figure.

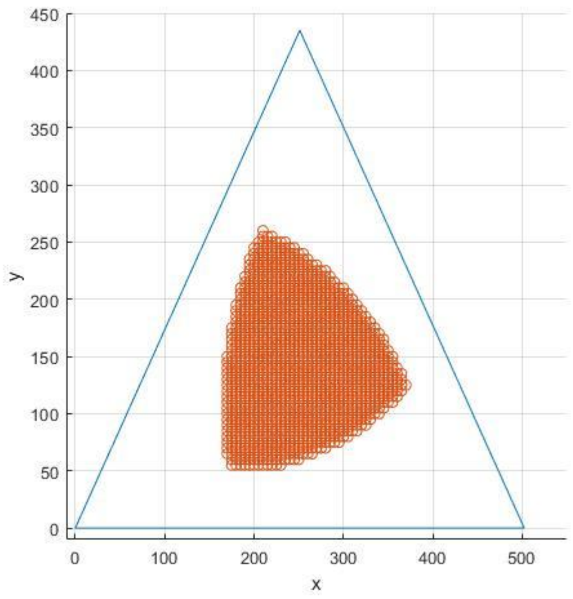

Figure 30

4 Conclusion

In conclusion the forward kinematics for the Lynxmotion robot arm was successfully derived using the Proximal method. We have also produced plots for the model of the robot arm and used this to test and verify that the forward kinematics are correct. The forward kinematics have then been used to accurately plot the robot's workspace, showing what areas are within reach and also indicating required joint angels to reach the target.

Also, the inverse kinematics was successfully derived using the geometric method. The geometric method is simpler for smaller robot arms but can become difficult when working with larger robot arms. The Inverse kinematics was then modelled and tested in a variety of positions using the Forwards kinematics to check each set of results. In addition, we have used this model to plan a task. The output of each stage of the task have been checked using the forward kinematics. Overall, we have produced a stable and reliable model of the robot arm. Finally, for the second part of the assessment, we present an approach for determining the inverse kinematics applied on 3-RRR planar parallel, which have important potential for different applications. Thus, the analysis of the workspace of the robot lead us to a great performance and a higher precision in positioning of the platform in order to do a trajectory planning of the manipulator.

5 References

[1] *Garden, H., HowStuffWorks, Science, Engineering and Robotics (2019). How Robots Work. [online] HowStuffWorks. Available at: https://science.howstuffworks.com/robot2.htm*

[2] *Anon, (2019). [online] Available at: https://www.robotshop.com/en/lynxmotion-al5b-robot-arm-hardware-4dof.html.*

[3] Owen-Hill, A. (2019). *How to Calculate a Robot's Forward Kinematics in 5 Easy Steps. [online]* Blog.robotiq.com. *Available at: https://blog.robotiq.com/how-to-calculate-a-robots-forward-kinematics-in-5-easy-steps.*

[4] *Mohammed Reyad AbuQassem, (2019). [online] Available at: https://www.academia.edu/27412828/Simulation_and_Interfacing_of_5_DOF_Educational_Robot_Arm*

[5] Robotpark ACADEMY. *(2019). PARALLEL ROBOTS - Robotpark ACADEMY. [online] Available at: http://www.robotpark.com/academy/all-types-of-robots/stationary-robots/parallel-robots/.*

6 Appendix

```
%% PART I --- SERIAL_ROBOT.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% SERIAL ROBOT %%%%%%%%%%%%%%
%%%%%%%%%% Robotics Fundamentals %%%%%%%%
%%%%%%%%%%%% December 2019 %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc
close all

%% static varables and globals
global L1 L2 L3 L4 L5
global IKtheta1 IKtheta2a IKtheta3a IKtheta4a IKtheta2b IKtheta3b IKtheta4b

%% variables such as link length and theta’s
%link lengths
L1 = 5 ;
L2 = 5 ;
L3 = 5;
L4 = 3 ;
L5 = 0;

%Joint angles
theta1 = 0 *(pi/180) ; %¡90 to 90
theta2 = 45 *(pi/180) ; % around ¡10 to 90
theta3 = -30 *(pi/180) ;
theta4 = -150 *(pi/180) ;
theta5 = 0 *(pi/180) ;


%% DH table

DH1 = [ 0 , 0 , L1 , theta1 ] ;
DH2 = [ 0 , ( pi /2) , 0 , theta2 ] ;
DH3 = [ L2 , 0 , 0 , theta3 ] ;
DH4 = [ L3 , 0 , 0 , theta4 ] ;
DH5 = [ 0 , -(pi /2) , L4 , theta5 ] ;
DHE = [ 0 , 0 , L5 , 0 ] ;

%% transformation matrix ’ s
% calculate the transformation matrix for each link
BL = get_dh_matrix ( [ 0 ,0 ,0 ,0] ) ; %Base Link

T01 = get_dh_matrix (DH1) ;
T12 = get_dh_matrix (DH2) ;
T23 = get_dh_matrix (DH3) ;
T34 = get_dh_matrix (DH4) ;
T45 = get_dh_matrix (DH5) ;
T5E = get_dh_matrix (DHE) ;

% calculate the tot a l transformation matrix
T02 = T01 * T12 ;
```

```
T03 = T02 * T23 ;
T04 = T03 * T34 ;
T05 = T04 * T45 ;
T0E = T05 * T5E;

hold on
grid on;
ylabel ( 'Y(cm)' ) , xlabel ( 'X(cm)' ) , zlabel ( 'Z(cm)' )
daspect ( [1 1 1] ) %keep aspect rat io the same
view( [0 , -1 ,0] )%side
axis ([-10 25 -0 25 -5 25] ) ; %manualy set axis


X_joints = [ 0 ; T01(1 ,4) ; T02(1 ,4) ; T03(1 ,4) ; T04(1 ,4) ; T05(1 ,4)
; T0E(1 ,4) ] ;
Y_joints = [ 0 ; T01(2 ,4) ; T02(2 ,4) ; T03(2 ,4) ; T04(2 ,4) ; T05(2 ,4)
; T0E(2 ,4) ] ;
Z_joints = [ 0 ; T01(3 ,4) ; T02(3 ,4) ; T03(3 ,4) ; T04(3 ,4) ; T05(3 ,4)
; T0E(3 ,4) ] ;


%% Plotlinks

figure(1);

original_x = X_joints;
original_y = Y_joints;
original_z = Z_joints;

plot3(X_joints,Y_joints,Z_joints, 'b' , 'LineWidth' , 5 )
scatter3(X_joints,Y_joints,Z_joints,40,'filled','r')
text (BL (1 ,4) ,BL (2 ,4) - 5 ,BL (3 ,4) , 'J0' ) ;
text (BL (1 ,4) ,BL (2 ,4) + 5 ,BL (3 ,4) , 'J1' ) ;
text (T02(1 ,4) ,T02(2 ,4) + 5 ,T02(3 ,4) , 'J2' ) ;
text (T03(1 ,4) ,T03(2 ,4) + 5 ,T03(3 ,4) , 'J3' ) ;
text (T04(1 ,4) ,T04(2 ,4) + 5 ,T04(3 ,4) , 'J4' ) ;
text (T05(1 ,4) ,T05(2 ,4) + 5 ,T05(3 ,4) , 'J5' ) ;
text (T0E(1 ,4) ,T0E(2 ,4) - 5 ,T0E(3 ,4) , 'JE' ) ;

disp(T0E(1 ,4));
disp(T0E(2 ,4));
disp(T0E(3 ,4));


%% Plot workspace


X = [] ;
Y = [] ;
Z = [] ;
C = [],[];

for i =-10:10:10
    fprintf ( 'atiteration : %d \n ' , i ) %progress report 4 workspace
plot
    for j =0:5:135
        for k=-135:5:0
            for l =-180:5:0
```

```
                theta1= i *(pi/180) ;
                theta2= j *(pi/180) ;
                theta3=k*(pi/180) ;
                theta4= l *(pi/180) ;

                DH1 = [ 0 , 0 , L1 , theta1 ] ;
                DH2 = [ 0 , ( pi /2) , 0 , theta2 ] ;
                DH3 = [ L2 , 0 , 0 , theta3 ] ;
                DH4 = [ L3 , 0 , 0 , theta4 ] ;
                DH5 = [ 0 , -(pi /2) , L4 , theta5 ] ;
                DHE = [ 0 , 0 , L5 , 0 ] ;


                T01 = get_dh_matrix (DH1);
                T02 = T01 * get_dh_matrix (DH2);
                T03 = T02 * get_dh_matrix (DH3);
                T04 = T03 * get_dh_matrix (DH4);
                T05 = T04 * get_dh_matrix (DH5);
                T0E = T05 * get_dh_matrix (DHE) ;

                C(end+1,1) = 1-(j/135); %set red value from joint 2
                C(end,2) =((k+135)/135);%set green value from joint 3
                C(end,3) = (l/-180); %set blue value from joint 4
                X(end +1) = T0E(1,4);
                Y(end+1) = T0E(2,4);
                Z(end+1) = T0E(3,4);

            end
        end
    end
end

    figure(2)
    scatter3 (X, Y,Z,20 ,C, 'filled' ) ;
    view([0 , -1 ,0] )%side
    ylabel ( 'Y(cm)' ) , xlabel ( 'X(cm)' ) , zlabel ( 'Z(cm)' )
    figure(3)
    scatter3 (X, Y,Z,20 ,C, 'filled' ) ;
    ylabel ( 'Y(cm)' ) , xlabel ( 'X(cm)' ) , zlabel ( 'Z(cm)' )
    view([-1 ,-1 ,1])%isometric
    axis ([-20 20 -20 20 -10 20] )


%--------------------------------PART B-------------------------------------
--------
%% static varables and globals
global IKtheta1 IKtheta2a IKtheta3a IKtheta4a IKtheta2b IKtheta3b IKtheta4b

%% Program options
PlotAxis = 1;
PlotLinks = 3;
PlotWorkspace = 0;
InverseKinematics = 0;

%% Plot hexagon points
```

```
scale = 5;
N_sides = 8;
t=(1/(N_sides*2):1/N_sides:1)'*2*pi;
x_hex=sin(t);
y_hex=cos(t);
x_hex=scale*[x_hex; x_hex(1)];
y_hex=scale*[y_hex; y_hex(1)];
z_hex = ones(9,1);

end_point_X = x_hex(1:8);
end_point_Y = y_hex(1:8);
end_point_Z = z_hex(1:8);


end_point_X = [1.9134,4.6194,4.6194,1.9134,-1.9134,-4.6194,-4.6194,-
1.9134,1.9134];
end_point_Y = [4.6194,1.9134,-1.9134,-4.6194,-4.6194,-
1.9134,1.9134,4.6194,4.6194];
end_point_Z = [10,10,10,10,10,10,10,10,10];

%% DH table
DH1 = [ 0 , 0 , L1 , theta1 ] ;
DH2 = [ 0 , ( pi /2) , 0 , theta2 ] ;
DH3 = [ L2 , 0 , 0 , theta3 ] ;
DH4 = [ L3 , 0 , 0 , theta4 ] ;
DH5 = [ 0 , -(pi /2) , L4 , theta5 ] ;
DHE = [ 0 , 0 , L5 , 0 ] ;

%% transformation matrix's
% calculate the transformation matrix for each l ink
BL = get_dh_matrix ([0,0,0,0]); %Base Link

T01 = get_dh_matrix (DH1) ;
T12 = get_dh_matrix (DH2) ;
T23 = get_dh_matrix (DH3) ;
T34 = get_dh_matrix (DH4) ;
T45 = get_dh_matrix (DH5) ;
T5E = get_dh_matrix (DHE) ;

% calculate the total transformation matrix
T02 = T01 * T12 ;
T03 = T02 * T23 ;
T04 = T03 * T34 ;
T05 = T04 * T45 ;
T0E = T05 * T5E;

grid on;
ylabel ( 'Y(cm)' ) , xlabel ( 'X(cm)' ) , zlabel ( 'Z(cm)' )
daspect ( [1 1 1] ) %keep aspect ratio the same


%% Plotlinks

Link_X = [ 0 ; T01(1 ,4) ; T02(1 ,4) ; T03(1 ,4) ; T04(1 ,4) ; T05(1 ,4) ;
T0E(1 ,4) ] ;
Link_Y = [ 0 ; T01(2 ,4) ; T02(2 ,4) ; T03(2 ,4) ; T04(2 ,4) ; T05(2 ,4) ;
T0E(2 ,4) ] ;
Link_Z = [ 0 ; T01(3 ,4) ; T02(3 ,4) ; T03(3 ,4) ; T04(3 ,4) ; T05(3 ,4) ;
T0E(3 ,4) ] ;
```

```
flag = 1;
while flag < 10
    for i = 4:1:12
        Number_of_points = 3;

        try
            x_points = linspace(end_point_X(1,i-3), end_point_X(1,i-2),
Number_of_points+2);
            y_points = linspace(end_point_Y(1,i-3), end_point_Y(1,i-2),
Number_of_points+2);
            z_points = linspace(end_point_Z(1,i-3), end_point_Z(1,i-2),
Number_of_points+2);
        catch
            x_points = linspace(end_point_X(1,i-3), end_point_X(1,1),
Number_of_points+2);
            y_points = linspace(end_point_Y(1,i-3), end_point_Y(1,1),
Number_of_points+2);
            z_points = linspace(end_point_Z(1,i-3), end_point_Z(1,1),
Number_of_points+2);
        end

        for point_number = 1:1:5
            [IKtheta1 , IKtheta2a , IKtheta3a , IKtheta4a , IKtheta2b ,
IKtheta3b , IKtheta4b ] = solveIK (x_points(1,point_number),
y_points(1,point_number), z_points(1,point_number), 0 );

            DH1 = [ 0 , 0 , L1 , IKtheta1*(pi/180)];
            DH2 = [ 0 , ( pi /2) , 0 , IKtheta2a*(pi/180)];
            DH3 = [ L2 , 0 , 0 , IKtheta3a*(pi/180)];
            DH4 = [ L3 , 0 , 0 , IKtheta4a*(pi/180)];
            DH5 = [ 0 , -(pi /2) , L4 , 45 *(pi/180)];
            DHE = [ 0 , 0 , L5 , 0 ];


            %% transformation matrix's
            % calculate the transformation matrix for each l ink
            BL = get_dh_matrix ([0,0,0,0]); %Base Link

            T01 = get_dh_matrix (DH1) ;
            T12 = get_dh_matrix (DH2) ;
            T23 = get_dh_matrix (DH3) ;
            T34 = get_dh_matrix (DH4) ;
            T45 = get_dh_matrix (DH5) ;
            T5E = get_dh_matrix (DHE) ;

            % calculate the total transformation matrix
            T02 = T01 * T12 ;
            T03 = T02 * T23 ;
            T04 = T03 * T34 ;
            T05 = T04 * T45 ;
            T0E = T05 * T5E ;

            figure(4);

            ylabel ( 'Y(cm)' ) , xlabel ( 'X(cm)' ) , zlabel ( 'Z(cm)' );
            X_joints = [ 0 ; T01(1 ,4) ; T02(1 ,4) ; T03(1 ,4) ; T04(1 ,4)
; T05(1 ,4) ; T0E(1 ,4) ] ;
            Y_joints = [ 0 ; T01(2 ,4) ; T02(2 ,4) ; T03(2 ,4) ; T04(2 ,4)
; T05(2 ,4) ; T0E(2 ,4) ] ;
            Z_joints = [ 0 ; T01(3 ,4) ; T02(3 ,4) ; T03(3 ,4) ; T04(3 ,4)
; T05(3 ,4) ; T0E(3 ,4) ] ;

            set(4,'position',[700 100 500 500]);
            hold on;
            scatter3(X_joints,Y_joints,Z_joints,20,'filled','r');
            hold on;

scatter3(end_point_X,end_point_Y,end_point_Z,50,'o','y','filled');
            hold on;
            plot3(end_point_X,end_point_Y,end_point_Z,'r' , 'LineWidth' , 2
);

            hold on;
            %scatter3 (Link_X,Link_Y,Link_Z, 'r' , 'LineWidth' , PlotLinks
);
            scatter3(Link_X,Link_Y,Link_Z,40,'filled','r')
            hold on;
            plot3 (X_joints,Y_joints,Z_joints, 'b' , 'LineWidth' ,
PlotLinks ); % this plots the p vector

            pause(.1);

            view([-1 ,-1 ,1]);%isometric;
            axis ([-20 20 -20 20 -20 20] );
            grid on;
            cla;

            flag = flag+1;
        end
    end
end



%% Functions


% get_dh_matrix -- to obtain transformation matrix from dh table
function T = get_dh_matrix ( parameters )
T = [(cos(parameters(1,4))), -(sin(parameters(1,4))), 0, (parameters(1,1));
(sin(parameters(1,4))) * (cos(parameters(1,2))), (cos(parameters(1,4))) *
(cos(parameters(1,2))), -(sin(parameters(1,2))), -(sin(parameters(1,2))) *
(parameters(1,3)) ;
(sin(parameters(1,4))) * (sin(parameters(1,2))), (cos(parameters(1,4))) *
(sin(parameters(1,2))), (cos(parameters(1,2))), (cos(parameters(1,2))) *
(parameters(1 ,3) ) ;
0 ,0 ,0 ,1];
end

% SolveIk -- computes joint angles for given target points
function [ IKtheta1 , IKtheta2a , IKtheta3a , IKtheta4a , IKtheta2b ,
IKtheta3b , IKtheta4b ] =solveIK ( xTarget , yTarget , zTarget ,
pitchTarget )
```

```
global L1 L2 L3 L4 IKtheta1 IKtheta2a IKtheta3a IKtheta4a IKtheta2b
IKtheta3b IKtheta4b

IKtheta1 = atan2 ( yTarget , xTarget ) * (180/pi) ;
pitch = pitchTarget ;
z = zTarget-(L4* sin (pitch*(pi/180) ) )-L1 ;
XYhyp = sqrt ( ( xTarget ^2) +( yTarget ^2) )-(L4* cos (pitch*(pi/180) ) )
;

IKcostheta3 = ( (XYhyp^2) +( z^2)-(L2^2)-(L3^2) ) /(2*L2*L3) ;


IKtheta3a = atan2(+ sqrt (1-( IKcostheta3 ^2) ) , IKcostheta3 ) * (180/pi)
;
IKtheta3b = atan2(-sqrt (1-( IKcostheta3 ^2) ) , IKcostheta3 ) * (180/pi) ;

IKcostheta2a = ( (XYhyp* ( L2+(L3* cos ( IKtheta3a *(pi/180) ) ) ) ) +( z
*L3* sin ( IKtheta3a *(pi/180)) ) ) / ( (XYhyp^2) +( z^2) ) ;
IKcostheta2b = ( (XYhyp* ( L2+(L3* cos ( IKtheta3b *(pi/180) ) ) ) ) +( z
*L3* sin ( IKtheta3b *(pi/180)) ) ) / ( (XYhyp^2) +( z^2) ) ;

IKtheta2a = atan2(+ sqrt (1-( IKcostheta2a ^2) ) , IKcostheta2a ) *
(180/pi) ;
IKtheta2b = atan2(- sqrt (1-( IKcostheta2b^2) ) , IKcostheta2b ) * (180/pi)
;

IKtheta4a = pitch-( IKtheta2a+IKtheta3a ) -90;
IKtheta4b = pitch-( IKtheta2b+IKtheta3b ) -90;
end






%% PART I --- Trajectories.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% SERIAL ROBOT %%%%%%%%%%%%%%
%%%%%%%%%% Robotics Fundamentals %%%%%%%%
%%%%%%%%%%%% December 2019 %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc
close all

%% static varables and globals
global L1 L2 L3 L4 L5
global IKtheta1 IKtheta2a IKtheta3a IKtheta4a IKtheta2b IKtheta3b IKtheta4b

L1 = 0 ;
L2 = 5 ;
L3 = 5;
L4 = 3 ;
L5 = 0;

%% Trajectories
```

```
grid on;
axis on;
ylabel ( 'Y(cm)' ) , xlabel ( 'X(cm)' ) , zlabel ( 'Z(cm)' );
view([-1 ,-1 ,1]);%isometric;
axis ([-20 20 -20 20 -20 20] );
hold on;

trajectory_x_path =  [5;5;0;0;0;5;5;5];
trajectory_y_path =  [10.0000; 8.6364; 5; 2.5; 0 ; -2.2727 ;-3.6364; -
5.0000];
trajectory_z_path =  [5;5;5;5;5;5;5;5];


Number_of_points = 5;
trajectory_x_points = linspace(5, 5, Number_of_points+2);
trajectory_y_points = linspace(10, -5, Number_of_points+2);
trajectory_z_points = linspace(5, 5, Number_of_points+2);

flag_traj = 1;
while flag_traj < 10

    for point_number = 1:1:8

        [IKtheta1 , IKtheta2a , IKtheta3a , IKtheta4a , IKtheta2b ,
IKtheta3b , IKtheta4b ] = solveIK (trajectory_x_path(point_number,1),
trajectory_y_path(point_number,1), trajectory_z_path(point_number,1), 0 );


        DH1 = [ 0 , 0 , L1 , IKtheta1*(pi/180)];
        DH2 = [ 0 , ( pi /2) , 0 , IKtheta2a*(pi/180)];
        DH3 = [ L2 , 0 , 0 , IKtheta3a*(pi/180)];
        DH4 = [ L3 , 0 , 0 , IKtheta4a*(pi/180)];
        DH5 = [ 0 , -(pi /2) , L4 , 45 *(pi/180)];
        DHE = [ 0 , 0 , L5 , 0 ];


        %% transformation matrix’s
        % calculate the transformation matrix for each l ink
        BL = get_dh_matrix ([0,0,0,0]); %Base Link

        T01 = get_dh_matrix (DH1) ;
        T12 = get_dh_matrix (DH2) ;
        T23 = get_dh_matrix (DH3) ;
        T34 = get_dh_matrix (DH4) ;
        T45 = get_dh_matrix (DH5) ;
        T5E = get_dh_matrix (DHE) ;

        % calculate the total transformation matrix
        T02 = T01 * T12 ;
        T03 = T02 * T23 ;
        T04 = T03 * T34 ;
        T05 = T04 * T45 ;
        T0E = T05 * T5E ;

        figure(5);

        ylabel ( 'Y(cm)' ) , xlabel ( 'X(cm)' ) , zlabel ( 'Z(cm)' );
        %daspect ( [1 1 1] ) %keep aspect ratio the same
        %% Plotlinks

        X_joints = [ 0 ; T01(1 ,4) ; T02(1 ,4) ; T03(1 ,4) ; T04(1 ,4) ;
T05(1 ,4) ; T0E(1 ,4) ] ;
        Y_joints = [ 0 ; T01(2 ,4) ; T02(2 ,4) ; T03(2 ,4) ; T04(2 ,4) ;
T05(2 ,4) ; T0E(2 ,4) ] ;
        Z_joints = [ 0 ; T01(3 ,4) ; T02(3 ,4) ; T03(3 ,4) ; T04(3 ,4) ;
T05(3 ,4) ; T0E(3 ,4) ] ;
        disp(T0E(3 ,4));
        set(5,'position',[700 100 500 500]);
        hold on;
        scatter3(X_joints,Y_joints,Z_joints,40,'filled','r');
        hold on;
        plotcube([5 5 5],[2.5 0 5],.8,[0 0 1]);
        hold on;

scatter3(trajectory_x_points,trajectory_y_points,trajectory_z_points,'fille
d','o','g');
        hold on;
        plot3(trajectory_x_points,trajectory_y_points,trajectory_z_points);
        hold on;

scatter3(trajectory_x_path,trajectory_y_path,trajectory_z_path,'filled','o'
,'g')
        hold on;
        plot3(trajectory_x_path,trajectory_y_path,trajectory_z_path);
        hold on;
        plot3 (X_joints,Y_joints,Z_joints, 'b' , 'LineWidth' , PlotLinks );
% this plots the p vector

        pause(.5);

        view([-1 ,-1 ,1]);%isometric;
        axis ([-20 20 -20 20 -20 20] );
        grid on;
        cla;
        flag_traj = flag_traj +1;


    end
end





%% Functions




function T = get_dh_matrix ( parameters )
T = [(cos(parameters(1,4))), -(sin(parameters(1,4))), 0, (parameters(1,1));
(sin(parameters(1,4))) * (cos(parameters(1,2))), (cos(parameters(1,4))) *
(cos(parameters(1,2))), -(sin(parameters(1,2))), -(sin(parameters(1,2))) *
(parameters(1,3)) ;
(sin(parameters(1,4))) * (sin(parameters(1,2))), (cos(parameters(1,4))) *
(sin(parameters(1,2))), (cos(parameters(1,2))), (cos(parameters(1,2))) *
(parameters(1 ,3) ) ;
0 ,0 ,0 ,1];
```

```
end

function [ IKtheta1 , IKtheta2a , IKtheta3a , IKtheta4a , IKtheta2b ,
IKtheta3b , IKtheta4b ] =solveIK ( xTarget , yTarget , zTarget ,
pitchTarget )
global L1 L2 L3 L4 IKtheta1 IKtheta2a IKtheta3a IKtheta4a IKtheta2b
IKtheta3b IKtheta4b

IKtheta1 = atan2 ( yTarget , xTarget ) * (180/pi) ;
%pitch = 90+theta2+theta3+theta4 ;
pitch = pitchTarget ;
z = zTarget-(L4* sin (pitch*(pi/180) ) )-L1 ;
XYhyp = sqrt ( ( xTarget ^2) +( yTarget ^2) )-(L4* cos (pitch*(pi/180) ) )
;

IKcostheta3 = ( (XYhyp^2) +( z^2)-(L2^2)-(L3^2) ) /(2*L2*L3) ;


IKtheta3a = atan2(+ sqrt (1-( IKcostheta3 ^2) ) , IKcostheta3 ) * (180/pi)
;
IKtheta3b = atan2(-sqrt (1-( IKcostheta3 ^2) ) , IKcostheta3 ) * (180/pi) ;

IKcostheta2a = ( (XYhyp* ( L2+(L3* cos ( IKtheta3a *(pi/180) ) ) ) ) +( z
*L3* sin ( IKtheta3a *(pi/180)) ) ) / ( (XYhyp^2) +( z^2) ) ;
IKcostheta2b = ( (XYhyp* ( L2+(L3* cos ( IKtheta3b *(pi/180) ) ) ) ) +( z
*L3* sin ( IKtheta3b *(pi/180)) ) ) / ( (XYhyp^2) +( z^2) ) ;

IKtheta2a = atan2(+ sqrt (1-( IKcostheta2a ^2) ) , IKcostheta2a ) *
(180/pi) ;
IKtheta2b = atan2(- sqrt (1-( IKcostheta2b^2) ) , IKcostheta2b ) * (180/pi)
;

IKtheta4a = pitch-( IKtheta2a+IKtheta3a ) -90;
IKtheta4b = pitch-( IKtheta2b+IKtheta3b ) -90;
end
```

```
%% PART II --- PARALLEL_ROBOT.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% PARALLEL ROBOT %%%%%%%%%%%%%%
%%%%%%%%%% Robotics Fundamentals %%%%%%%%%
%%%%%%%%%%%% December 2019 %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
close all

% Specify design parameters
base_length = 170; % lower section
platform_length = 130; % upper section
r_p = 130; % platform joint circle radius
r_b = 290; % base joint circle radius

%% Inverse Kinematics

% Input parameters
X_c = 250;
Y_c = 150;
a = 0;

% Calculate global variables
BC = [X_c-r_b*cosd(30);Y_c-r_b*sind(30);0]; % Define vector BC
R_BC = [cosd(a), -sind(a), 0; sind(a), cosd(a), 0; 0, 0, 1]; % Define 2D
rotation matrix for platform with rotation a

% Create arm objects
alpha1 = arm(1,R_BC,BC,base_length,r_p,platform_length,r_b,a,X_c,Y_c);
alpha2 = arm(2,R_BC,BC,base_length,r_p,platform_length,r_b,a,X_c,Y_c);
alpha3 = arm(3,R_BC,BC,base_length,r_p,platform_length,r_b,a,X_c,Y_c);

% determine if target position is within workspace
[x,y,ii,singularity_flag] =
singularity(alpha1,alpha2,alpha3,X_c,Y_c,1,[],[]);

if singularity_flag == 0
    % Get joint coordinates
    posArm1 = getJointCo(base_length,[alpha1(1),alpha1(2)],
[alpha1(4),alpha1(5)],platform_length,alpha1(3));
    posArm2 = getJointCo(base_length,[alpha2(1),alpha2(2)],
[alpha2(4),alpha2(5)],platform_length,alpha2(3));
    posArm3 = getJointCo(base_length,[alpha3(1),alpha3(2)],
[alpha3(4),alpha3(5)],platform_length,alpha3(3));

    % Create lists of arm parameters for looping
    arms = {posArm1,posArm2,posArm3};
    theta =
{[alpha1(4),alpha1(5)],[alpha2(4),alpha2(5)],[alpha3(4),alpha3(5)]};
```

```
    psi = {alpha1(3), alpha2(3), alpha3(3)};

    figure(1)
    set(1,'position',[0 100 500 500])

    hold on; grid on
    xlabel('x'); ylabel('y'); % axis labels
    axis equal;
    plot(X_c,Y_c,'X') % target position (for center of platform)
    tri_b = [0 2*r_b*cosd(30) r_b*cosd(30) 0; 0 0 (r_b*sind(30) + r_b) 0];
% base triangle
    tri_p = [posArm1(1,3) posArm2(1,3) posArm3(1,3) posArm1(1,3);
posArm1(2,3) posArm2(2,3) posArm3(2,3) posArm1(2,3)]; % platform triangle
    plot(tri_p(1,:), tri_p(2,:)) % plot base
    plot(tri_b(1,:), tri_b(2,:)) % plot platform

    for i = 1:3
        text(arms{i}(1,3),arms{i}(2,3),num2str(i)); % add PP label
        text(arms{i}(1,1),arms{i}(2,1),num2str(theta{i}(1))); % add joint
theta values
        text(arms{i}(1,2),arms{i}(2,2),num2str(psi{i}(1))); % add joint psi
values
        plot(arms{i}(1,:),arms{i}(2,:)); % plot arm
    end
else
    fprintf('Target position outside of workspace (singularity)\n')
end

%% Workspace

% Input parameters
x = zeros(1,120);
y = zeros(1,120);
a = 0;

% Calculate workspace by neglecting singularities
ii = 1;
for X_c = 5:5:600
    for Y_c = 5:5:600
        % Calculate global variables
        BC = [X_c-r_b*cosd(30);Y_c-r_b*sind(30);0]; % Define vector BC
        R_BC = [cosd(a), -sind(a), 0; sind(a), cosd(a), 0; 0, 0, 1]; %
Define 2D rotation matrix for platform with rotation a

        % Create arm objects
        alpha1 =
arm(1,R_BC,BC,base_length,r_p,platform_length,r_b,a,X_c,Y_c);
        alpha2 =
arm(2,R_BC,BC,base_length,r_p,platform_length,r_b,a,X_c,Y_c);
        alpha3 =
arm(3,R_BC,BC,base_length,r_p,platform_length,r_b,a,X_c,Y_c);

        % Determine which points lie within the workspace
        [x,y,ii,singularity_flag] =
singularity(alpha1,alpha2,alpha3,X_c,Y_c,ii,x,y);

    end
end
```

```
% Determine workspace by plotting range of each joint
jj = 1;
for theta = 1:5:360
    for psi = 1:5:360

        % x,y positions for each joint
        x1(jj) = base_length*cosd(theta) + platform_length*cosd(psi) +
r_p*cosd(30-a);
        x2(jj) = base_length*cosd(theta+120) +
platform_length*cosd(psi+120) + r_p*cosd(30+120-a) + 2*r_b*cosd(30);
        x3(jj) = base_length*cosd(theta+240) +
platform_length*cosd(psi+240) + r_p*cosd(30+240-a) + r_b*cosd(30);
        y1(jj) = base_length*sind(theta) + platform_length*sind(psi) +
r_p*sind(30-a);
        y2(jj) = base_length*sind(theta+120) +
platform_length*sind(psi+120) + r_p*sind(30+120-a);
        y3(jj) = base_length*sind(theta+240) +
platform_length*sind(psi+240) + r_p*sind(30+240-a) + (r_b*sind(30) + r_b);

        jj = jj + 1;
    end
end

figure(2)
set(2,'position',[500 100 500 500])

hold off; hold on; grid on
xlabel('x'); ylabel('y'); % axis labels
axis([-10 550 -10 450])
tri_b = [0 2*r_b*cosd(30) r_b*cosd(30) 0; 0 0 (r_b*sind(30) + r_b) 0]; %
base triangle
plot(tri_b(1,:), tri_b(2,:)) % plot base
plot(x,y,'o') % target position (for center of platform)

figure(3)
set(3,'position',[900 100 500 500])

hold off; hold on; grid on
xlabel('x'); ylabel('y'); % axis labels
axis equal
tri_b = [0 2*r_b*cosd(30) r_b*cosd(30) 0; 0 0 (r_b*sind(30) + r_b) 0]; %
base triangle
plot(tri_b(1,:), tri_b(2,:)) % plot base
plot(x1,y1,'.r','MarkerSize',1) % target position (for center of platform)
plot(x2,y2,'.g','MarkerSize',1) % target position (for center of platform)
plot(x3,y3,'.b','MarkerSize',1) % target position (for center of platform)

%% Functions needed for the code to run

%Singularity function - To know if the point belongs in the workspace

function [x,y,ii,singularity_flag] =
singularity(alpha1,alpha2,alpha3,X_c,Y_c,ii,x,y)

    singularity_flag = 1;

    theta1 = [alpha1(4),alpha1(5)];
    theta2 = [alpha2(4),alpha2(5)];
    theta3 = [alpha3(4),alpha3(5)];
    psi1 = alpha1(3);
    psi2 = alpha2(3);
    psi3 = alpha3(3);
    if (isnan(theta1(1)) == 0) && (isnan(theta2(1)) == 0) &&
(isnan(theta3(1)) == 0) && ...
            (isnan(psi1) == 0) && (isnan(psi2) == 0) && (isnan(psi3) == 0)
        x(ii) = X_c;
        y(ii) = Y_c;
        ii = ii + 1;
        singularity_flag = 0;

    end
    if (isnan(theta1(2)) == 0) && (isnan(theta2(2)) == 0) &&
(isnan(theta3(2)) == 0) && ...
            (isnan(psi1) == 0) && (isnan(psi2) == 0) && (isnan(psi3) == 0)
        x(ii) = X_c;
        y(ii) = Y_c;
        ii = ii + 1;
        singularity_flag = 0;

    end
end

%arm function - to find joint parameters

function alpha  =
arm(i,R_BC,BC,base_length,r_p,platform_length,r_b,a,X_c,Y_c)

    if nargin > 0
        % set a and PB for joint i
        if i == 1
            a = a + 30;
            PB(1) = 0;
            PB(2) = 0;
        elseif i == 2
            a = a + 120 + 30;
            PB(1) = 2*r_b*cosd(30);
            PB(2) = 0;
        elseif i == 3
            a = a + 120*2 + 30;
            PB(1) = r_b*cosd(30);
            PB(2) = r_b*sind(30) + r_b;
        end
        BPB = [-r_b*cosd(a); -r_b*sind(a); 0];
        CPP = [-r_p*cosd(a); -r_p*sind(a); 0];
        PBPP = R_BC * CPP + BC - BPB;

        e_1 = -2 * PBPP(2) * base_length;
        e_2 = -2 * PBPP(1) * base_length;
        e_3 = (PBPP(1))^2 + (PBPP(2))^2 + base_length^2 -
platform_length^2;

        t_1 = ( -e_1 + sqrt(e_1^2 + e_2^2 - e_3^2) ) / (e_3 - e_2);
        t_2 = ( -e_1 - sqrt(e_1^2 + e_2^2 - e_3^2) ) / (e_3 - e_2);

        T = [R_BC(1,1), R_BC(1,2), R_BC(1,3), X_c;
             R_BC(2,1), R_BC(2,2), R_BC(2,3), Y_c;
             R_BC(3,1), R_BC(3,2), R_BC(3,3), 0;
```

```
                 0,          0,          0,          1];

        if t_1 == real(t_1)
            theta(1) = 2*atand(t_1);
            theta(2) = 2*atand(t_2);
            Theta = theta;
            c_psi = ( PBPP(1) - base_length*cosd(theta(1)) ) /
platform_length;
            s_psi = ( PBPP(2) - base_length*sind(theta(1)) ) /
platform_length;

            link_2_angle = atan2d(s_psi,c_psi);
        else
            theta(1) = NaN;
            theta(2) = NaN;
            link_2_angle = NaN;
            Theta = theta;
        end

    end
    alpha = [PB,link_2_angle,Theta];

end
% GetJoint co-ordinates -- To compute joint positions
function points =
getJointCo(base_length,PB,Theta,platform_length,link_2_angle)
    points = [[PB(1), PB(1) + base_length*cosd(Theta(1)), PB(1) +
base_length*cosd(Theta(1)) + platform_length*cosd(link_2_angle)]; [PB(2),
PB(2) + base_length*sind(Theta(1)), PB(2) + base_length*sind(Theta(1)) +
platform_length*sind(link_2_angle)]];
end
```