

Robotic Manipulator Design

By:

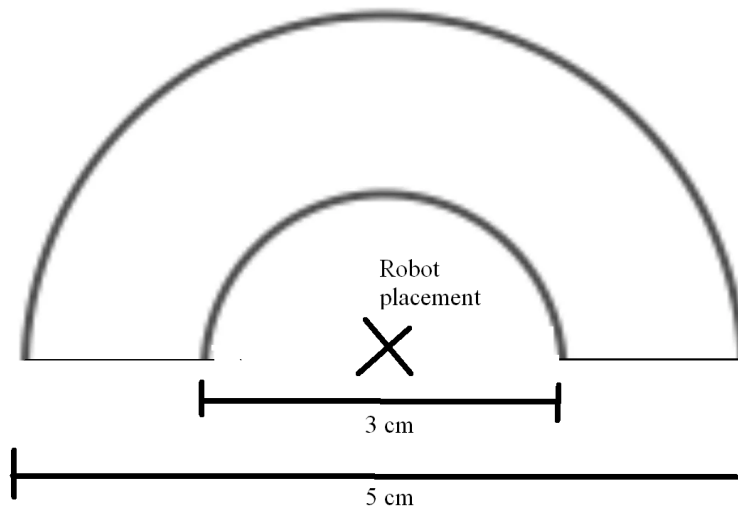
Ahmad Nadeem Saigol

Contents

Problem Statement:	3
Manipulator Design:	3
RP Manipulator:	3
Modified DH parameters:	4
Transformation Matrix:	4
Forward Kinematics:	4
Inverse Kinematics:	5
Jacobian and Singularities:	5
Trajectory Generation:	6
Homing:	6
Working of the Code:	6
Results:	8
From Homing Point to Point 1:	8
Changes w.r.t Time (Quintic Polynomial Profile):	8
From Point 1 to Point 2:	9
Changes w.r.t Time (Trapezoidal Profile):	9
From Point 3 to Final Point:	9
From Point 3 to Final Point (Quintic Polynomial Profile):	10
Robot, Workspace and Paths:	10
Total time:	11
Code:	11

Problem Statement:

The objective is to design a robotic manipulator which has a task space of Half 2D Donut shape with the following dimensions:



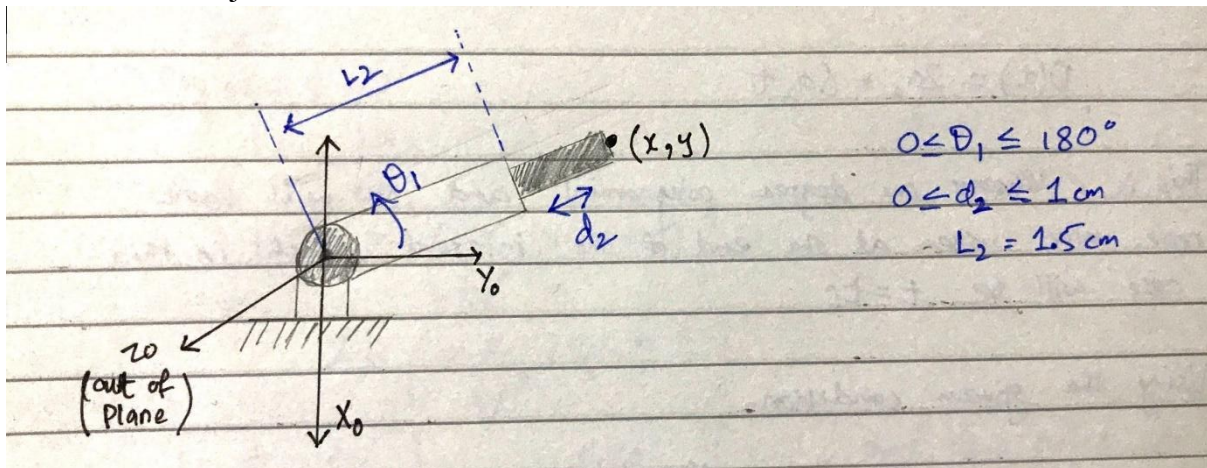
Manipulator Design:

The first step in designing any robotic manipulator is to decide the type and the number of joints it will have, what will be its link lengths and how will they be oriented in space. These variables are decided while keeping the task space in mind. Thus, for the given scenario, several configurations were considered namely, RR, PP etc. but the RP robot was deemed appropriate as it will not only achieve the objective but also there will be a smaller number of singularities in the workspace and its kinematics will be much simpler. Further, it directly relates to the cylindrical coordinate system which is more convenient to use in this case than cartesian coordinate system.

RP Manipulator:

This robot has two joints – Revolute and Prismatic – and both are at 90° with each other. Revolute joint can rotate from 0 to 180 degrees while the prismatic joint has a range from 0 to 1 cm. Further, the prismatic joint has an offset and is at 1.5cm along its Z axis

from the revolute joint.



Modified DH parameters:

The DH parameters were calculated using modified approach:

Modified DH Parameters Table for RP robot

r =

noname:: 2 axis, RP, modDH, fastRNE

j	theta	d	a	alpha	offset
1	q1	0	0	0	0
2	0	q2	0	-1.5708	1.5

Transformation Matrix:

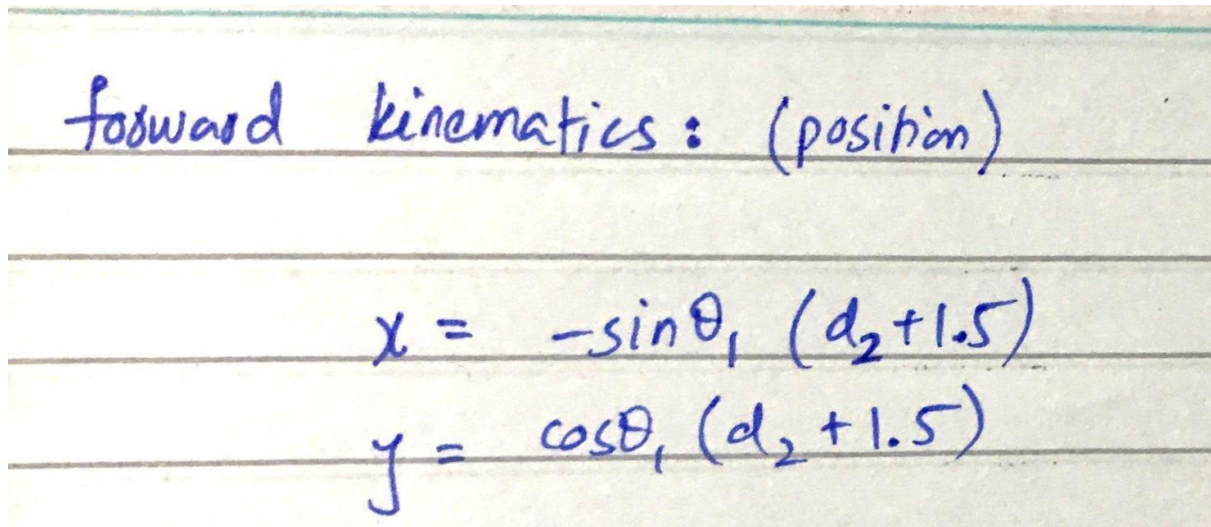
Transformation matrix of the end effector frame {2} with respect to the base frame {0} is as follows:

Transformation Matrix :

$${}^0_2T = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & -\sin\theta_1 (d_2 + 1.5) \\ \sin\theta_1 & 0 & \cos\theta_1 & \cos\theta_1 (d_2 + 1.5) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Forward Kinematics:

For any given joint angles, the corresponding values of the end effector of RP robot in the Cartesian Coordinate system is calculated using these formulas:

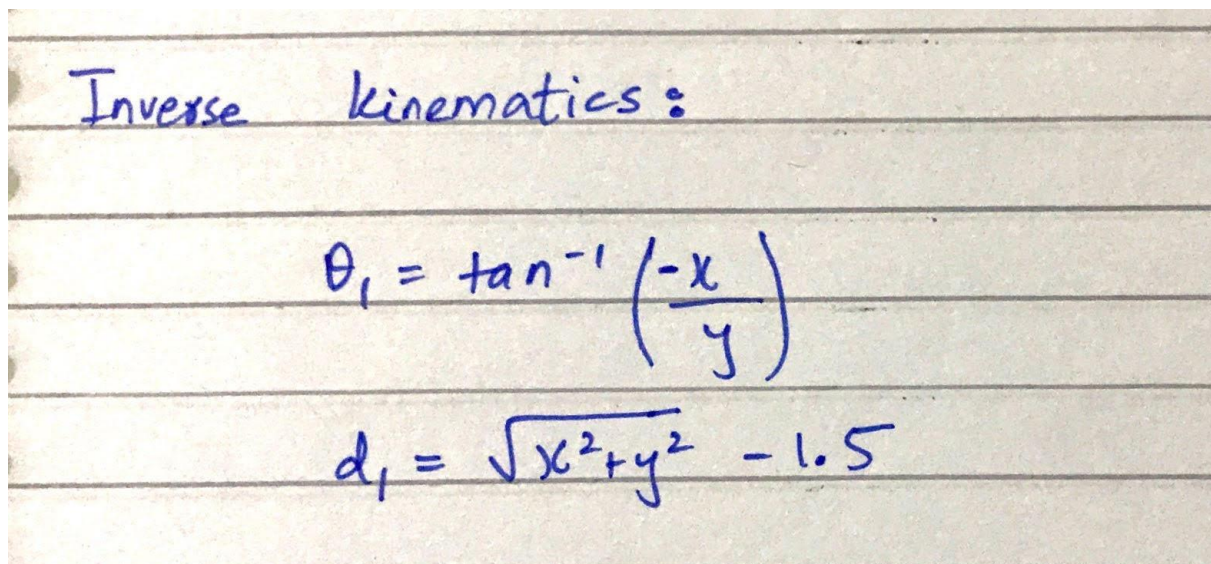


forward kinematics: (position)

$$x = -\sin\theta_1 (d_2 + 1.5)$$
$$y = \cos\theta_1 (d_2 + 1.5)$$

Inverse Kinematics:

For any point given in Cartesian Coordinate system, the corresponding joint angles is calculated using inverse kinematics. For RP robot, it is calculated using algebraic approach and is follows:



Inverse kinematics:

$$\theta_1 = \tan^{-1}\left(\frac{-x}{y}\right)$$
$$d_1 = \sqrt{x^2 + y^2} - 1.5$$

Note: It is d_2 in the figure not d_1 .

Jacobian and Singularities:

Jacobian maps the joint angle velocities to the velocities in the Cartesian coordinate system. Furthermore, it can also be used for mapping Cartesian velocities to Joint velocities using its inverse. However, in the Cartesian workspace of a manipulator, there often exists certain points where the manipulator loses its degree

of freedom. These points are known as Singularities and they can be estimated using the determinant of the Jacobian. In case of RP manipulator, as it can also be seen in the figure, its singularity will exist when the second joint will have a value of -1.5. But the value of this joint can only be from 0 to 1, thus there will be no singularity in its workspace.

Jacobian and Singularity:

$${}^0J = \begin{bmatrix} -\cos\theta_1 (1.5+d_2) & -\sin\theta_1 \\ -\sin\theta_1 (1.5+d_2) & \cos\theta_1 \end{bmatrix}$$

$$|J| = -(1.5+d_2)$$

$$0 = -1.5-d_2$$

$$\boxed{d_2 = -1.5}$$

Trajectory Generation:

To move the robot from one configuration to another, one must decide what path or geometry it must follow to reach its destination. The next question comes how fast or slow it should move. Also, there should not be any jerks which could cause vibration and stresses in the structure. Furthermore, often when moving a robot in the real world, it encounters different obstacles and hurdles in its way which needs to be avoided. Keeping all this in view, the trajectory generation becomes a key aspect when working with manipulators.

Trajectory generation can be carried out in both Cartesian space and Joint Space. Each approach has its own pros and cons but for RP robot in consideration, the trajectory has been generated in the joint space. Moreover, out of many profiles that could have been used for the creation of the path, two commonly used profiles have been considered: a quintic (5th order) polynomial and trapezoidal path. Zero boundary conditions for the velocity and acceleration have been used for the calculation of the coefficients of the function.

Homing:

The initial point at which the robot will be (at $t=0$) is set as:

The Homing point of the Manipulator is:
 In Joint Space: $[\pi/2, 0.5]$
 In Cartesian Space: $[-0.2, 0]$

Working of the Code:

A brief explanation on the working of the code is provided below:

- 1) The RP manipulator is modeled using modified DH parameters with the help of the commands 'Link()' and 'SerialLink()'. The limits on the joints is also provided using 'qlim' key in 'Link()'.
- 2) The workspace is calculated using forward kinematic equations and is plotted on figure 1.
- 3) The singularities of the workspace are calculated using Jacobian Matrix. The matrix (in base frame) is obtained through the method 'jacob0' on 'SerialLink' and after taking its determinant, it is solved for the joint angles.
- 4) There are two nested infinite loops which perform different functions in the code. They are described below:
 - a) Input Loop: It is inside the other loop and performs several functions. Firstly, it takes the Cartesian Coordinates from the user. Then it carries out the inverse kinematics of this point and then finally it checks whether the point is in workspace or not. The loop continues until the user enters either the program terminating key or the correct coordinates.

```
PLease Enter Desired Cartesian Co-ordinates as [x, y]: [3 3]
```

```
Point does not lie in the workspace  
Please try another point
```

- b) Outer Loop: This loop also carries out different functionalities. The code allows the trajectory to be generated between initial and final point in joint space using two different functions. The user decides which function to be used. The trajectory is generated using 'mtraj' command. It takes function, initial point, final point and steps as arguments. The number of steps is set to 50. Furthermore, with the help of 'subplot' command, how joint angles and Cartesian coordinates change with time as the robot moves, is plotted on figure 2. Afterwards, the trajectory calculated in joint space is mapped to cartesian space using 'fkine()' and is plotted on the figure 1. Initial and final points and the animation of the robot is also plotted on the same figure 1. On the next iteration, initial point is replaced with the previous final point and the same procedure is repeated. However, the figure 1 is overwritten on each iteration and the previous trajectories followed by the manipulator remains on the figure. (see figure in the results). The loop terminates when the user enters the terminating key.

- 5) Since the loops are infinite, a program terminating key is used and checks for it have been set at different parts of the code. This key is 'E' (with single quotes) and whenever, the user wants to stop the program execution, he/she can enter this key as input. for example:

NOTE: At any point if you want to exit the program, enter E (in single quotes).

Please Enter Desired Cartesian Co-ordinates as [x, y]: 'E'

- 6) Outside the loops, the last bit of code calculates the total time taken by the manipulator from Homing Point to the last user defined point.

NOTE: If you want the end effector of the robot to reach the boundary points of the workspace where x-axis is zero, then you would have to enter the input as one of these values: [0, y] or [-0, -y] (correspond to the two ends of the workspace on x axis)

Results:

For the demonstration of the working of the code, three different points were provided to the code on the single run.

From Homing Point to Point 1:

Homing Point = [-0.2 0]

Point 1 = [-0.53 -2.31]

The Homing point of the Manipulator is:
In Joint Space: [pi/2, 0.5]
In Cartesian Space: [-0.2, 0]

Please Enter Desired Cartesian Co-ordinates as [x, y]: [-0.53 -2.31]

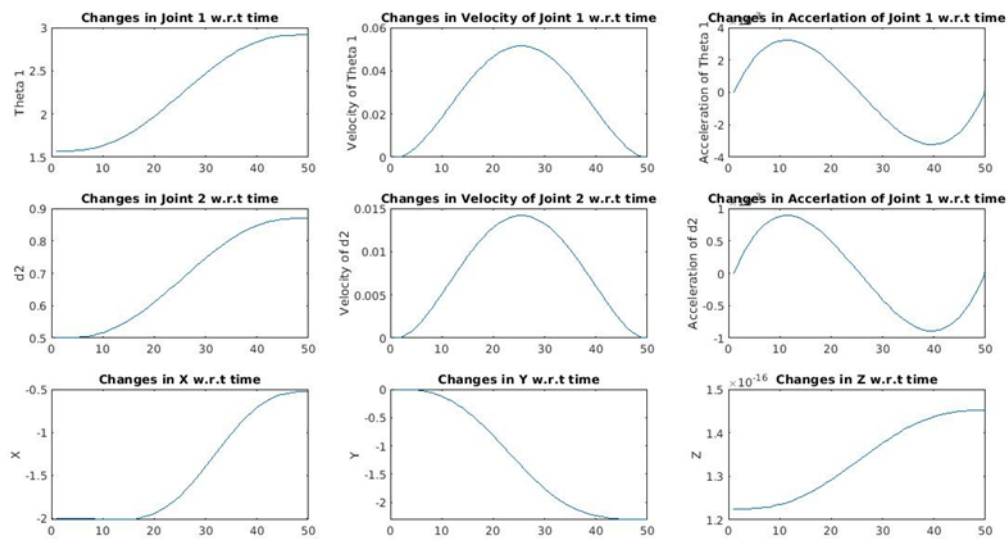
Goal Point Joint Values:

Q1 =

2.9161 0.8700

Which Profile would you like to use for the generation of trajectory?
Enter 0 for polynomial of degree 5 and 1 for tapezoidal: 0

Changes w.r.t Time (Quintic Polynomial Profile):



From Point 1 to Point 2:

Point 1 = [-0.53 -2.31]

Point 2 = [-0.063 -1.79]

Initial Point Joint Values:

Q0 =

2.9161 0.8700

PLease Enter Desired Cartesian Co-ordinates as [x, y]: [-0.063 -1.79]

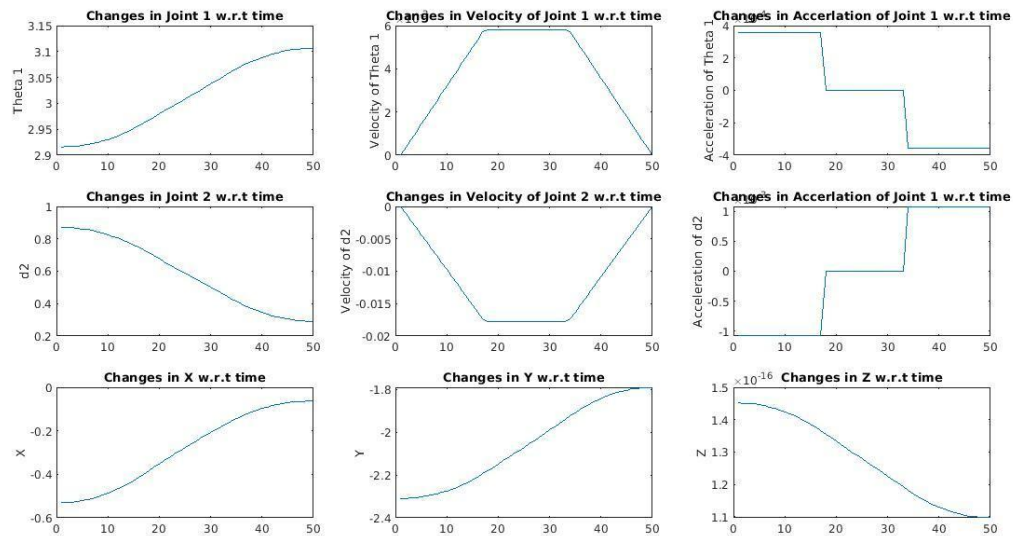
Goal Point Joint Values:

Q1 =

3.1064 0.2911

Which Profile would you like to use for the generation of trajectory?
Enter 0 for polynomial of degree 5 and 1 for tapezoidal: 1

Changes w.r.t Time (Trapezoidal Profile):



From Point 3 to Final Point:

Point 3 = [-0.063 -1.79]

Point 4 = [-1.15 2]

Initial Point Joint Values:

Q0 =

3.1064 0.2911

PLease Enter Desired Cartesian Co-ordinates as [x, y]: [-1.15 2]

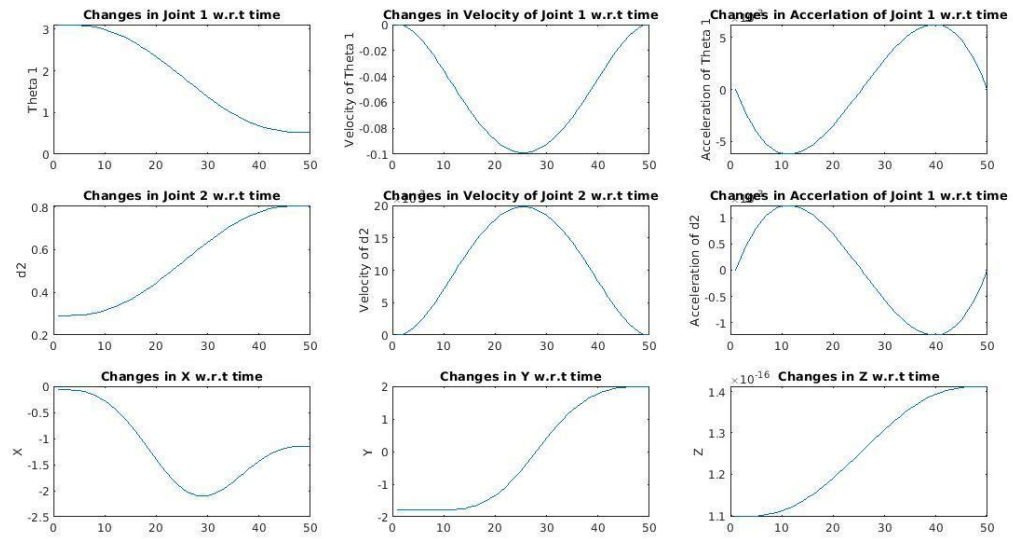
Goal Point Joint Values:

Q1 =

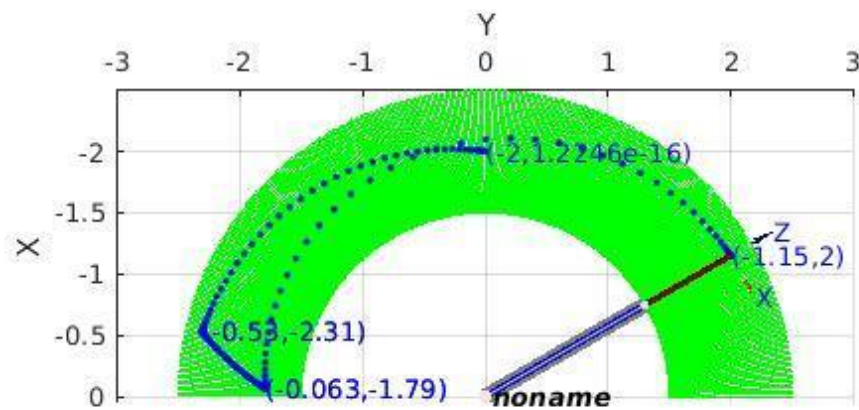
0.5218 0.8071

Which Profile would you like to use for the generation of trajectory?
Enter 0 for polynomial of degree 5 and 1 for tapezoidal: 0

From Point 3 to Final Point (Quintic Polynomial Profile):



Robot, Workspace and Paths:



Total time:

Time taken by manipulator to move from one interpolated point to another is 0.02s.
As number of interpolated points are 50, so for any two user defined points, total time is 1s.
And the total time taken (in seconds) by manipulator from Homing point to the last user defined point is:
3

Code:

```
clear all
close all

disp(newline)
disp('2D HALF DONUT MANIPULATOR')
disp(newline)
disp('NOTE: At any point if you want to exit the program, enter E (in single quotes).')
disp(newline)

f1=figure; %for changes in variables w.r.t time
f2=figure; %for robot and path animation

%Serial Link(RP) using modified DH parameters

L2=1.5; %prismatic joint offset (in cm)

L(1)=Link('d', 0, 'a', 0, 'alpha', 0, 'qlim', [0, pi], 'modified');
L(2)=Link('theta', 0, 'a', 0, 'alpha', -pi/2, 'qlim', [0 1], 'offset', L2, 'modified');
disp('Modified DH Parameters Table for RP robot')
r=SerialLink(L)

%forward kinematics symbolically
syms TH1 D2
disp('The Transformation matrix is given by: ')
T=r.fkine([TH1 D2])

%Calculation of the workspace
th1=[0:0.01:pi]; %resolution=0.01 (angle is in radians)
d2=[0:1/314:1]; %resolution=0.0032 (in cm)
[TH1_,D2_]=meshgrid(th1,d2);
%Not using 'subs' command since it increases execution time significantly
W_X=-sin(TH1_).*(L2+D2_);
W_Y=cos(TH1_).*(L2+D2_);
W_Z=zeros(size(TH1_));
%plotting workspace
figure(f2);
plot3(W_X(:),W_Y(:),W_Z(:), '.', 'Color', 'g', 'MarkerSize', 1);

%Calculation of Singularities
disp('The Jacobian in world frame (base frame) is given by:')
J0=r.jacob0([TH1 D2], 'trans')
disp('its Determinant is given by:')
det_J0 = det(J0(1:2, 1:2))
disp('Singularity will exist at ')
```

```

D2 = solve (det_J0 == 0, D2)
disp('This value does not lie in the range of d2 and thus, there will be no singularities in the workspace')
disp(newline)

%Homing point
disp('The Homing point of the Manipulator is:')
disp('In Joint Space: [pi/2, 0.5]')
disp('In Cartesian Space: [-0.2, 0]')
disp(newline)
Q0=[pi/2 0.5];

countTime=0;
while 1

    %make the last point as initial point for this path
    if ~(countTime==0)
        disp(newline)
        disp('Initial Point Joint Values:')
        Q0=Q1
    end

    %Input loop
    while 1

        %Input from the user
        in=input('Please Enter Desired Cartesian Co-ordinates as [x, y]: ');

        %Check if the user wants to exit the program
        if in == 'E'
            break
        end

        %Inverse Kinematics of entered point
        q1=atan2(-in(1),in(2));
        q2=(sqrt(in(1)^2 + in(2)^2))-1.5;

        %Check if the entered point lies in the workspace
        if ~((q1>=0 && q1<=pi)&&(q2>=0 && q2 <=1))
            disp(newline)
            disp('Point does not lie in the workspace')
            disp('Please try another point')
            disp(newline)
        else
            break
        end
    end

    %check if user wants to exit the program
    if in == 'E'
        break;
    end

    disp(newline)

    % Goal point
    disp('Goal Point Joint Values:')

```



```

Q1=[q1 q2]

%Trajectory selection by the user
disp('Which Profile would you like to use for the generation of trajectory?')
func_check=input('Enter 0 for polynomial of degree 5 and 1 for trapezoidal: ');

if func_check == 'E' %check if user wants to exit the program
    break
elseif func_check == 0
    func=@tpoly;
elseif func_check == 1
    func=@lspb;
end

%trajectory generation (in joint space) between the points.
[q v a]=mtraj(func, Q0, Q1, 50); %number of steps=50

%Plots how the joint's position, velocity and acceleration changes with time
%in joint space
figure(f1);
subplot(3,3,1)
plot(q(:,1))
title('Changes in Joint 1 w.r.t time')
ylabel('Theta 1')

subplot(3,3,2)
plot(v(:,1))
title('Changes in Velocity of Joint 1 w.r.t time')
ylabel('Velocity of Theta 1')

subplot(3,3,3)
plot(a(:,1))
title('Changes in Acceleration of Joint 1 w.r.t time')
ylabel('Acceleration of Theta 1')

subplot(3,3,4)
plot(q(:,2))
title('Changes in Joint 2 w.r.t time')
ylabel('d2')

subplot(3,3,5)
plot(v(:,2))
title('Changes in Velocity of Joint 2 w.r.t time')
ylabel('Velocity of d2')

subplot(3,3,6)
plot(a(:,2))
title('Changes in Acceleration of Joint 2 w.r.t time')
ylabel('Acceleration of d2')

%Calculation of path in Cartesian Space
fk=r.fkine(q);
for i=1:1:50
    X(i)=fk(1,i).t(1);
    Y(i)=fk(1,i).t(2);
    Z(i)=fk(1,i).t(3);
end

```

```

%plots how X,Y,Z changes w.r.t time as the manipulator moves from one
%configuration to the other
subplot(3,3,7)
plot(X)
title('Changes in X w.r.t time')
ylabel('X')

subplot(3,3,8)
plot(Y)
title('Changes in Y w.r.t time')
ylabel('Y')

subplot(3,3,9)
plot(Z)
title('Changes in Z w.r.t time')
ylabel('Z')

figure(f2)
hold on

%plots the trajectory followed by the manipulator in Cartesian Space.
plot3(X,Y,Z, 'l', 'Color', 'b')

%displays the coordinates (Cartesian Space) on the graph
text(X(1),Y(1), ['(' num2str(X(1)) ',' num2str(Y(1)) ')'], 'Color','b')
text(X(end),Y(end), ['(' num2str(X(end)) ',' num2str(Y(end)) ')'], 'Color','b')

%produces an animation of how robot moves from one configuration to another
%by changing its joint angles
r.plot(q, 'workspace', [-3 0 -3 3 0 1], 'basecolor', 'w', 'notiles')

countTime=countTime+1;

hold off

end

%Total Time Calculation
if countTime ~=0
    disp(newline)
    disp('Time taken by manipulator to move from one interpolated point to another is 0.02s.')
    disp('As number of interpolated points are 50, so for any two user defined points, total time is 1s.')
    disp('And the total time taken (in seconds) by manipulator from Homing point to the last user
defined point is: ')
    disp(countTime)
end

```