Modeling and Analysis of 5 DOF Robotic Manipulator in MATLAB

Indushekhar Prasad Singh

A Project report for the course ENPM 662 Introduction to Robot Modeling
Masters of Engineering Robotics

A. JAMES CLARK
SCHOOL OF ENGINEERING

University of Maryland , College Park
United State of America

17th Dec 2017

Acknowledgments

I would like to thank the University of Maryland, Professor William S Levine and Chad Kessens for providing me an opportunity to work on this project. I would also like to thank my friends and advisors for their technical support.

Contents

1 Introduction **3**

2 Approach **3**

3 Forward Kinematics **3**

4 Inverse Kinematics **5**

5 Workspace Representation **6**

6 Simulink Model and Simulation **7**

6.1 Rigid Body Modeling 7

6.2 Simulation 8

7 Validation **9**

8 Conclusion **10**

9 Future Work **10**

A Appendix A : Simulink Diagram and Model **11**

A.1 Simulink Diagram 11

A.2 Model 11

B Appendix B : MATLAB Programs **11**

B.1 Forward Kinematics MATLAB Program 11

B.2 Workspace MATLAB Program 12

B.3 Simulink Simulation MATLAB Program 13

B.4 Robotics Tooolbox Validation MATLAB Program 16

1 Introduction

Robot manipulators today find application in many manufacturing industries. It is used to complete a designated task as some programmed behaviors or by manual commands. In both the cases the end or end effector is the part that do the meaning work for us. This makes the end effector's, kinematics and workspace analysis critical for any robotic manipulator modeler.

The manipulator under consideration for this project is a having 5 Degree of freedom with all five joints as revolute. Since the model proposed will have all revolute joint, it will provide dexterous configuration. The spherical workspace produced by the arm has got wide range of application in industries. This type of manipulator are typically used in pick and place application so, the simulation of the robot from its initial configuration to the target joint angles is also of foremost importance.

Objective

The objective is :

1. To perform the Forward and Inverse kinematics analysis of the manipulator.
2. Development of MATLAB program to calculate forward kinematics.
3. 3D workspace representation using MATLAB for the manipulator using Monte Carlo method.
4. Simulation of the model in simulink from its intial configuration to the target joints angles.

2 Approach

First step is assigning the coordinate axis to the robot model to get the DH parameter. Parameter obtained will then be used to calculate the forward kinematics of the robotic manipulator by calculating the homogeneous transformation matrix of end effector. A MATLAB program will then compute the same for the given joint angles . The purpose of MATLAB program is to ease the process of forward kinematics calculation, so that a designer can utilize it for the practical purposes thereby getting the end effector position for any joint angle.

Second step is to calculate the inverse kinematics of the manipulator analytically, which will enable us to get the set of joint angles for the desired end effector orientation.

For the workspace Visulation, Monte- Carlo method will be used. This method uses random intervals between the minimum and Maximum joint variable range, In our case it's the joint angles. We can define the number of samples in each interval.

Next, a model of the manipulator will be created in Simscape Multibody and then it will be programmed using Simulink is to simulate the system. A MATLAB code will be developed to pass the desired input to the Simulation. At the validation of the results will be done using Robotic Toolbox and Revolute joint sensors.

3 Forward Kinematics

Theoretical Solution

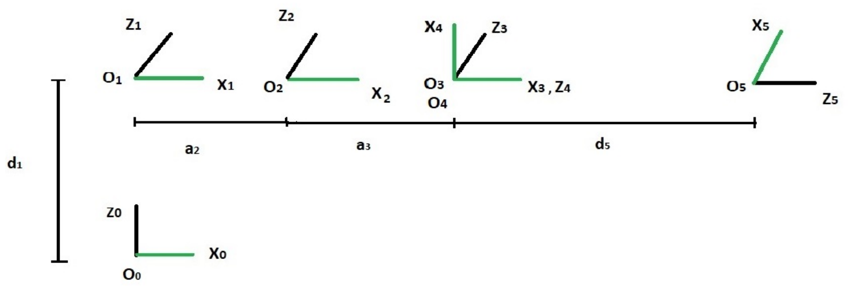
The Forward Kinematics analysis derives the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints. The Denavit-Hartenberg, or DH convention used in this project to perform the Forward Kinematics analysis simplify the analysis considerably. In this convention, each homogeneous transformation A_i is represented as a product of four basic transformations:

$$A_i = Rot_{z,\theta_i} Trans_{Z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \tag{1}$$

$$= \begin{bmatrix} C_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}c_{\alpha_i} & a_{\theta_i}c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_{\theta_i}s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where the four quantities $\theta_i, a_i, d_i, \alpha_i$ are parameters associated with link i and joint i, called as joint angle ,link length,link offset and link twist .

Coordinate frame assignment was done as per DH convention. There was a trick involved in it, because of the joint axis configuration of the manipulator. For the last revolute joint, to assign the DH convention its origin had to be selected at the origin of 4th joint i.e O_3 and O_4 coincides.



Coordinate frame assigned to 5 DOF Manipulator

The DH parameters obtained for our manipulator after assigning Coordinate frames as per DH convention is :

Link(i)	α_i	a_i	d_i	θ_i
1	-90	0	d_1	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3
4	-90	0	0	$\theta_4 - 90$
5	0	0	d_5	$\theta_5 - 90$

DH Parameters

Next step is to calculate the A matrices for each link using the equation (2) and then multiply them get the homogeneous transformation matrix of End effector:

$$T_5 = A_1A_2A_3A_4A_5$$

$$T_5 = \begin{bmatrix} c_1c_{234}c_5 + s_1s_5 & -c_1c_{234}s_5 + s_1c_5 & -c_1s_{234} & c_1(-d_5s_{234} + a_3c_{23} + a_2c_2) \\ c_1c_{234}c_5 - s_1s_5 & -s_1c_{234}s_5 - c_1c_5 & -s_1s_{234} & s_1(-d_5s_{234} + a_3c_{23} + a_2c_2) \\ -s_{234}c_5 & s_{234}s_5 & -c_{234} & d_1 - a_2s_2 - a_3s_{23} - d_5c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Where $C_{ijk} = cos(\theta_i + \theta_j + \theta_k), s_{ijk} = sin(\theta_i + \theta_j + \theta_k)$.

From T_5 one can obtain the position of end effector as :

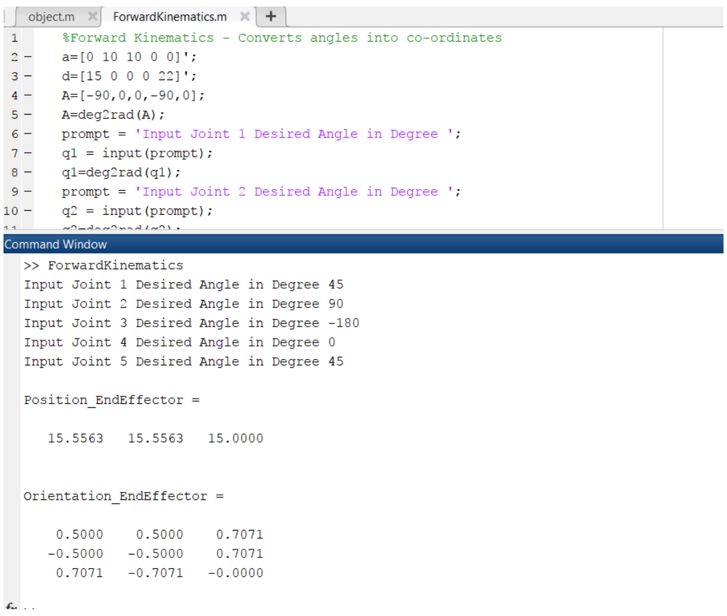
$$P_5 = \begin{bmatrix} c_1(-d_5 s_{234} + a_3 c_{23} + a_2 c_2) \\ s_1(-d_5 s_{234} + a_3 c_{23} + a_2 c_2) \\ d_1 - a_2 s_2 - a_3 s_{23} - d_5 c_{234} \end{bmatrix} \tag{4}$$

and orientation as :

$$O_5 = \begin{bmatrix} c_1 c_{234} c_5 + s_1 s_5 & -c_1 c_{234} s_5 + s_1 c_5 & -c_1 s_{234} \\ c_1 c_{234} c_5 - s_1 s_5 & -s_1 c_{234} s_5 - c_1 c_5 & -s_1 s_{234} \\ -s_{234} c_5 & s_{234} s_5 & -c_{234} \end{bmatrix} \tag{5}$$

MATLAB Implementation

It is hectic to do theoretical calculation and thus it is useful to do computer do your work. The MATLAB program calculate the end effector position and orientation as per input joint angles given by the user in command window :



object.m   ForwardKinematics.m

```
%Forward Kinematics - Converts angles into co-ordinates
a=[0 10 10 0 0]';
d=[15 0 0 0 22]';
A=[-90,0,0,-90,0];
A=deg2rad(A);
prompt = 'Input Joint 1 Desired Angle in Degree ';
q1 = input(prompt);
q1=deg2rad(q1);
prompt = 'Input Joint 2 Desired Angle in Degree ';
q2 = input(prompt);
```

Command Window

```
>> ForwardKinematics
Input Joint 1 Desired Angle in Degree 45
Input Joint 2 Desired Angle in Degree 90
Input Joint 3 Desired Angle in Degree -180
Input Joint 4 Desired Angle in Degree 0
Input Joint 5 Desired Angle in Degree 45

Position_EndEffector =

   15.5563   15.5563   15.0000


Orientation_EndEffector =

    0.5000    0.5000    0.7071
   -0.5000   -0.5000    0.7071
    0.7071   -0.7071   -0.0000
```

MATLAB interface for forward kinematics analysis.

Run the file **ForwardKinematics.m** and then enter joint angles to obtain the end effector position and orientation. The manipulator link parameter used is as follows : $a_2 = a_3 = 10cm$, $d_1 = 15cm, d_2 = 22cm$, which is same as used in simulink model later.

4 Inverse Kinematics

Inverse kinematics analysis is concerned with the inverse problem of finding the joint variables in terms of the end-effector position and orientation. There are various approaches to solve this problem , Analytic method is used in this project.

From (4) and (5), we can write the equations of each x,y,z components of position and orientation as :

$$x_{5x} = c_1 c_{234} + s_1 s_5 \tag{6}$$

$$x_{5y} = c_1 c_{234} c_5 - s_1 s_5 \tag{7}$$

$$x_{5z} = -s_{234} c_5 \tag{8}$$

$$y_{5x} = -c_1 c_{234} s_5 + s_1 c_5 \tag{9}$$

$$y_{5y} = -s_1 c_{234} s_5 - c_1 c_5 \tag{10}$$

$$y_{5z} = s_{234} s_5 \tag{11}$$

$$z_{5x} = -c_1 s_{234} \tag{12}$$

$$z_{5y} = -s_1 s_{234} \tag{13}$$

$$z_{5z} = -c_{234} \tag{14}$$

$$p_{5x} = c_1(-d_5 s_{234} + a_3 c_{23} + a_2 c_2) \tag{15}$$

$$p_{5y} = s_1(-d_5 s_{234} + a_3 c_{23} + a_2 c_2) \tag{16}$$

$$p_{5z} = d_1 - a_2 s_2 - a_3 s_{23} - d_5 c_{234} \tag{17}$$

Now, we solve these equations for θ_1, θ_2, θ_3, θ_4, θ_5 :

$$\theta_1 = tan^{-1}\Big(\frac{p_{5y}}{p_{5x}}\Big), where,\ p_{5x} > 0\ or \frac{\pi}{2} < \theta_1 < \frac{\pi}{2} \tag{18}$$

$$\theta_2 = atan2\Big\{a(a_2 + a_3 c_3) - b a_3 s_3,\ a a_3 s_3 + b(a_2 + a_3 c_3)\Big\} \tag{19}$$

$$\theta_3 = arccos\Big(\frac{a^2 + b^2 - {a_2}^2 - {a_3}^2}{2 a_2 a_3}\Big) \tag{20}$$

Where, $a = d_1 - d_5 c_{234} - p_{5z}$ and $b = p_{5x} c_1 + p_{5y} s_1 + d_5 s_{234}$

$$\theta_4 = c_{23} - \theta_2 - \theta_3 \tag{21}$$

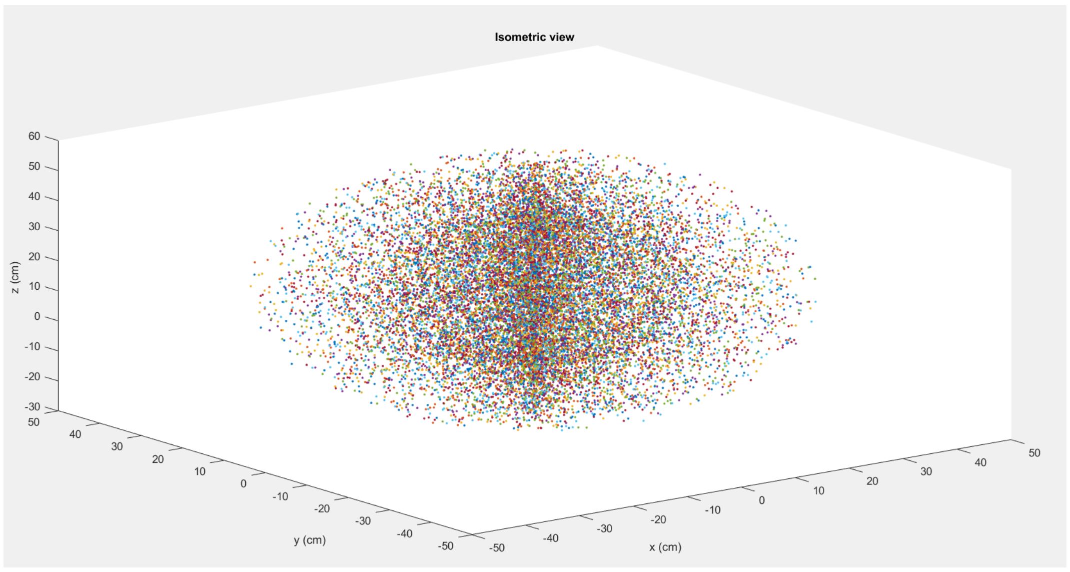
$$\theta_5 = c_{234}\theta_1 - 2atan(x_{5y}, x_{5x}) \tag{22}$$

5 Workspace Representation

The workspace of a manipulator is the total volume swept out by the endeffector as the manipulator executes all possible motions. The workspace is constrained by the geometry of the manipulator as well as mechanical constraints on the joints. It is very useful to visualize the workspace of the manipulator in order to use to for practical applications.

For plotting the workspace, we use the Monte-Carlo method. This method use random intervals between the minimum and maximum joint Angle and plot it against the position of end effector obtained from transformation matrix T_5 obtained in forward kinematics analysis. We can define the number of samples N in each interval. Increasing the number samples improves the resolution of the plot. Link parameter used for manipulator is same as simulink model parameters. Joint constrained used are :

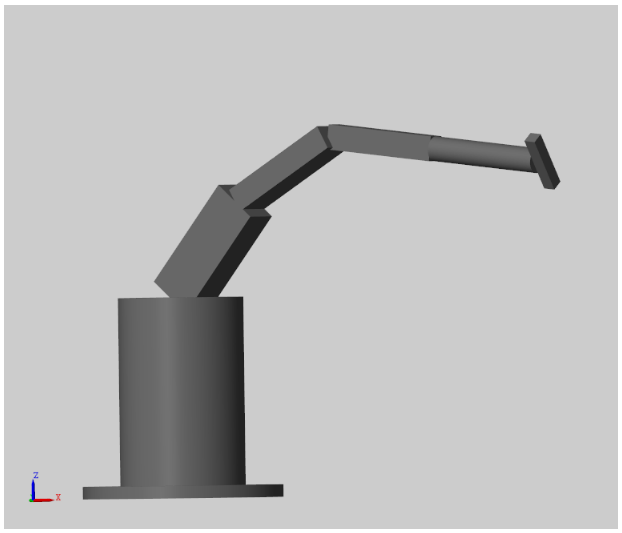
$$-\pi \le \theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5 \le \pi,$$



6 Simulink Model and Simulation

6.1 Rigid Body Modeling

Simscape Multibody (formerly SimMechanic) used for modeling in this project, provides a multibody simulation environment for 3D mechanical systems, such as robot. The model created for this project uses mainly three blocks :
Solid: Represents a solid combining a geometry, an inertia and mass, a graphics component, and rigidly attached frames into a single unit. A solid is the common building block of rigid bodies.
Revolute Joints: This block Represents a revolute joint acting between two frames and has one rotational degree of freedom.
Rigid transform: Defines a fixed 3-D rigid transformation between two frames.



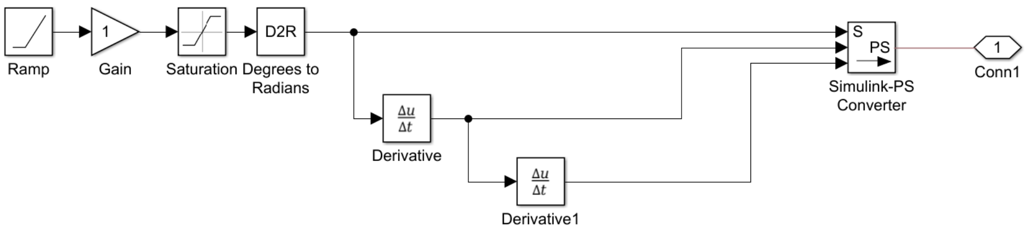
5 DOF Manipulator Model

The manipulator under consideration has 5 revolute joints which are modeled using the revolute joint block described. Manipulator links are represented by solid blocks , a total of 7 solid block is used for our model. Position and orientation of solid blocks and joints are assigned using rigid transformation, which gives us option to translate and rotate the blocks.

6.2 Simulation

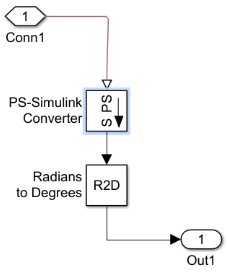
The simulation is done as per desired input provided by the user in the MATLAB command window by executing the file ENPM667.m. User is prompted to input the desired joint angles and joint velocity they want to achieve, then this input is passed to the simulink revoulte joint block of each joints. Joint sensor attached to each revolute joint measures the rotation in degree. Video of the simulation can be seen in the youtube link provided [4].

Joint Actuation in Simulink
"Ramp" block is used to supply input to joints, which provides a ramp signal, with a slope equal to the joint velocity in deg/sec supplied by user. The joint angle is equal to the value provided by the ramp input at that time. Since the ramp input does has not any final value, one may wonder how the joints are going to stop revolving. To solve this problem, a "Saturation" block is added after ramp input. Limit of this block is nothing but the input provided by the user in MATLAB command window or the desired joint angles. One other problem associated with this architecture is the case of negative joint angles, since ramp input provides only positive input. To resolve this issue a "Gain" block is added in series in between "Ramp" and "Saturation" block , which is programmed to achieve value of -1 or +1 according to the sign of input joint angles.



Joint Actuation System Diagram

It is important to note here that ramp input being time varying requires velocity and acceleration signal in addition to position. "Derivative" blocks in the diagram provides these two additional signals.
Joint Sensor
Joint sensor which gives the output as the rotation in degree is simply obtained by actuating the position sensing option of revolute joint and then adding the degree output to scope.



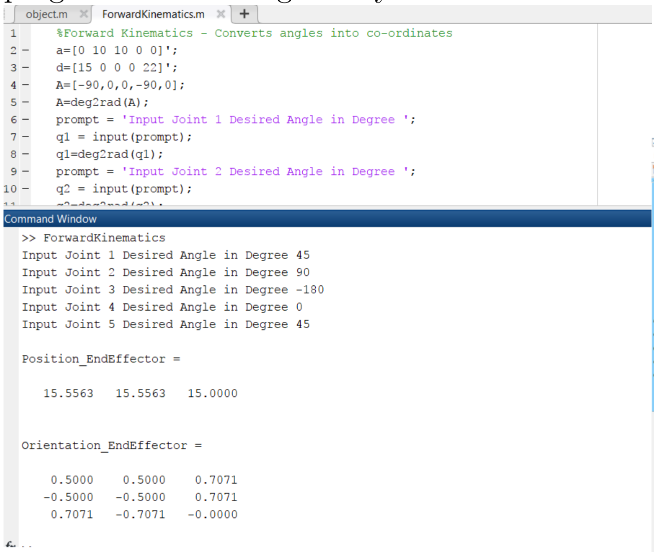
Sensor Diagram

7 Validation

To verify the results validation was done for Forward kinematics analysis and Simulink simulation output.

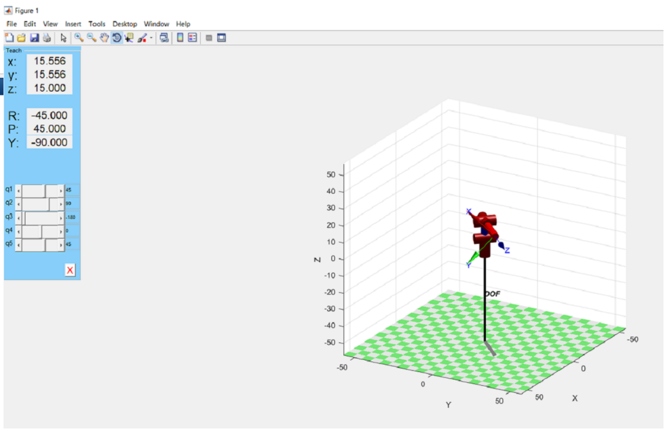
Forward Kinematics

Result obtained from the MATLAB program of Forward kinematics was verified using robotic toolbox [2]. It can be seen that the x,y,z coordinates obtained from MATLAB program is same as given by toolbox.



```
%Forward Kinematics - Converts angles into co-ordinates
a=[0 10 10 0 0]';
d=[15 0 0 0 22]';
A=[-90,0,0,-90,0];
A=deg2rad(A);
prompt = 'Input Joint 1 Desired Angle in Degree ';
q1 = input(prompt);
q1=deg2rad(q1);
prompt = 'Input Joint 2 Desired Angle in Degree ';
q2 = input(prompt);
```

```
>> ForwardKinematics
Input Joint 1 Desired Angle in Degree 45
Input Joint 2 Desired Angle in Degree 90
Input Joint 3 Desired Angle in Degree -180
Input Joint 4 Desired Angle in Degree 0
Input Joint 5 Desired Angle in Degree 45

Position_EndEffector =

   15.5563   15.5563   15.0000


Orientation_EndEffector =

    0.5000    0.5000    0.7071
   -0.5000   -0.5000    0.7071
    0.7071   -0.7071   -0.0000
```

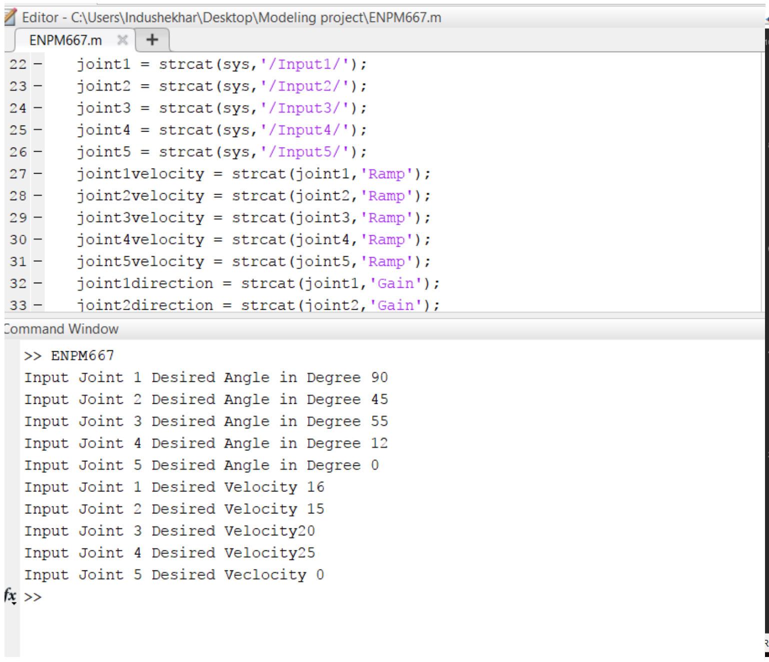


```
x: 15.556
y: 15.556
z: 15.000

R: -45.000
P: 45.000
Y: -90.000

q1 45
q2 90
q3 -180
q4 0
q5 45
```

Robotic Toolbox

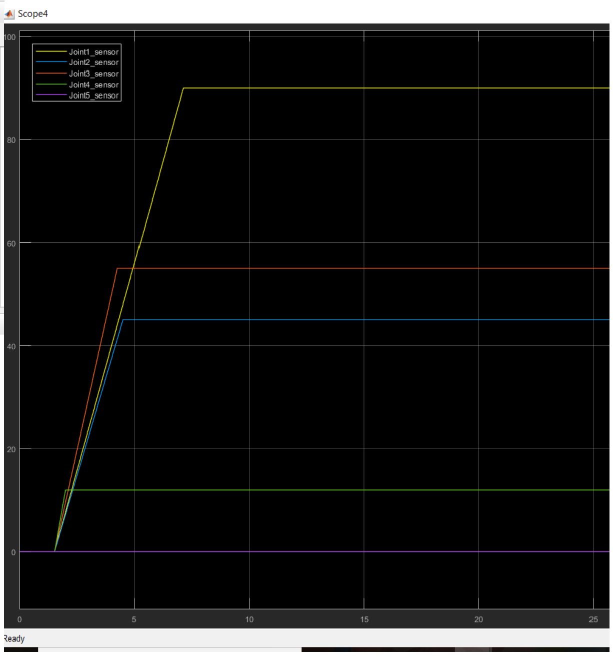
Simulation Validation

The output from joint sensors can be used verify the input given from MATLAB command window. Joint angles of all the five joints matches with the actual data obtained from Simulink joint sensor.


Editor - C:\Users\Indushekhar\Desktop\Modeling project\ENPM667.m
ENPM667.m
Command Window

```
joint1 = strcat(sys,'/Input1/');
joint2 = strcat(sys,'/Input2/');
joint3 = strcat(sys,'/Input3/');
joint4 = strcat(sys,'/Input4/');
joint5 = strcat(sys,'/Input5/');
joint1velocity = strcat(joint1,'Ramp');
joint2velocity = strcat(joint2,'Ramp');
joint3velocity = strcat(joint3,'Ramp');
joint4velocity = strcat(joint4,'Ramp');
joint5velocity = strcat(joint5,'Ramp');
joint1direction = strcat(joint1,'Gain');
joint2direction = strcat(joint2,'Gain');
```

```
>> ENPM667
Input Joint 1 Desired Angle in Degree 90
Input Joint 2 Desired Angle in Degree 45
Input Joint 3 Desired Angle in Degree 55
Input Joint 4 Desired Angle in Degree 12
Input Joint 5 Desired Angle in Degree 0
Input Joint 1 Desired Velocity 16
Input Joint 2 Desired Velocity 15
Input Joint 3 Desired Velocity20
Input Joint 4 Desired Velocity25
Input Joint 5 Desired Veclocity 0
>>
```



8 Conclusion

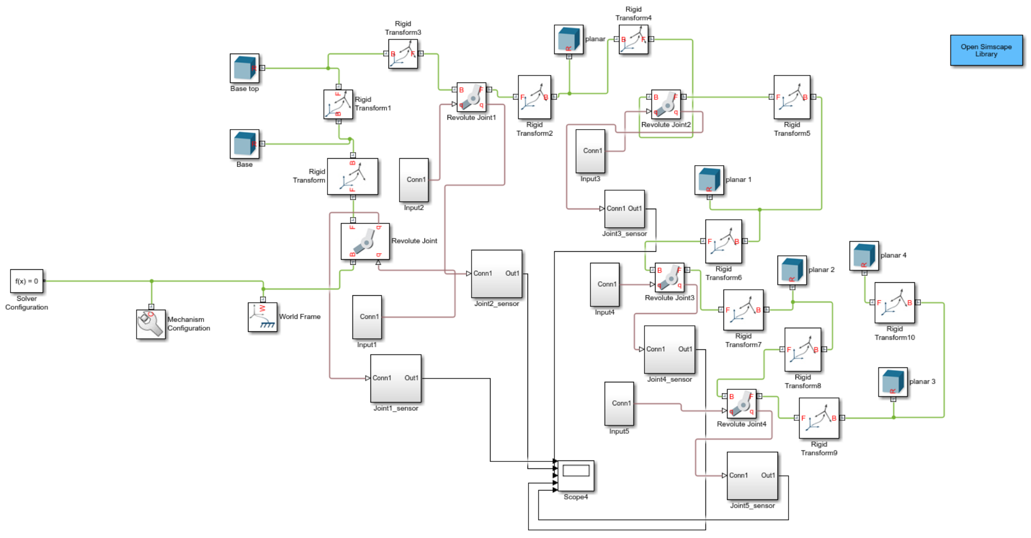
The project presented a foundation approach for modeling a Robotic Manipulator. It started with basic of the manipulator modeling i.e kinematics analysis, by calculating the theoretical results of both Forward and Inverse kinematics. Then a interactive MATLAB program was developed to ease the human effort required in calculation. The program can be used to calculate the Forward Kinematics of any manipulator given its DH parameter and link parameter. The project also developed a novel approach for representing workspace of the manipulator. With no effort the modeler can see the workspace according to the joint angle constraint and link length. The project also presented technique to control the Simulink block parameter from MATLAB command window. Validation at the end leaves no doubt on the work done and strengthen the accuracy of the work.

9 Future Work

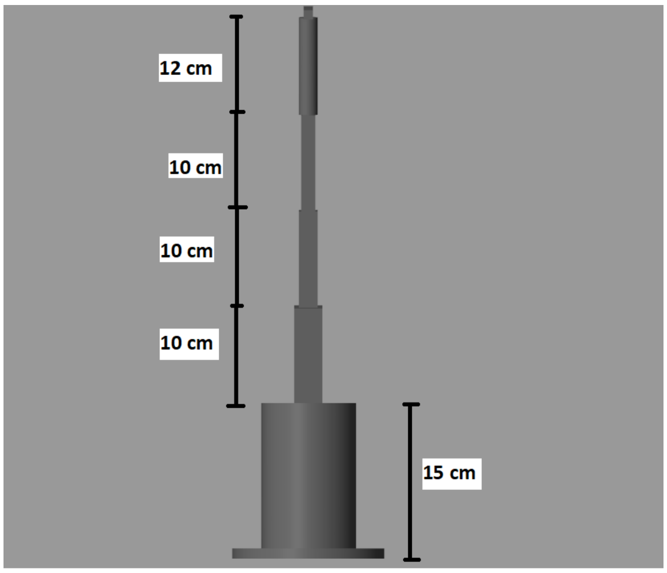
After the Kinematics analysis, it is important to consider the dynamics of the model. Robot joint control technique like Independent joint control can be applied to this manipulator . Other works may include path planning and obstacle detection algorithms.

A Appendix A : Simulink Diagram and Model

A.1 Simulink Diagram



A.2 Model



Density of the solid Used : $8g/cm^3$

B Appendix B : MATLAB Programs

B.1 Forward Kinematics MATLAB Program

MATLAB program for calculating the forward kinematics is given below. Program file name is **ForwardKinematics.m**, which can be found in ENPM667 zip folder. User have to enter the respective joint angles in the command prompt.

```
%Forward Kinematics - Converts angles into co-ordinates
```

```
a=[0 10 10 0 0]';
d=[15 0 0 0 22]';
A=[-90,0,0,-90,0];
A=deg2rad(A);
prompt = 'Input Joint 1 Desired Angle in Degree ';
q1 = input(prompt);
q1=deg2rad(q1);
prompt = 'Input Joint 2 Desired Angle in Degree ';
q2 = input(prompt);
q2=deg2rad(q2);
prompt = 'Input Joint 3 Desired Angle in Degree ';
q3 = input(prompt);
q3=deg2rad(q3);
prompt = 'Input Joint 4 Desired Angle in Degree ';
q4 = input(prompt);
q4=deg2rad(q4);
prompt = 'Input Joint 5 Desired Angle in Degree ';
q5 = input(prompt);
q5=deg2rad(q5);
q=[q1,q2,q3,q4,q5];
n=5;
T1_n = eye(4);
T{n} = zeros(4);
for i=1:n
T_i=[cos(q(i)), -cos(A(i))*sin(q(i)), sin(A(i))*sin(q(i)),  a(i)*cos(q(i));
    sin(q(i)),  cos(A(i))*cos(q(i)),-sin(A(i))*cos(q(i)),  a(i)*sin(q(i));
            0,           sin(A(i)),          cos(A(i)),            d(i);

           0,                    0,                    0,              1];


T1_n = T1_n * T_i;

end
Tot=T1_n;
Position_EndEffector= T1_n((1:3),4)'
Orientation_EndEffector = [T1_n((1:3),1)  T1_n((1:3),2)  T1_n((1:3),3)]
```

B.2 Workspace MATLAB Program

Workspace MATLAB program can be found as **workspace.m** file name.

```
syms a1 a2 a3 a4 a5 a6 a7 d1 d2 d3 d4 d5 d6 d7 alpha1 alpha2 alpha3 alpha4 alpha5 alpha

a1 = 0; alpha1 = -pi/2; d1 = 15;
a2 = 10; alpha2 = 0; d2 = 0;
a3 =10; alpha3 = 0; d3 = 0;
a4 = 0; alpha4 = -pi/2; d4= 0;
a5 = 0; alpha5 = 0; d5 = 22;
```

```
% Inserting joint limits for Arms
t1_min=-3.14;t1_max=3.14;
t2_min = -3.14; t2_max = 3.14;
t3_min =-3.14; t3_max =3.14;
t4_min = -3.14; t4_max =3.14;
t5_min = -3.14; t5_max = 3.14;

% Monte Carlo method
% sampling size
N = 20000;
t1 = t1_min + (t1_max-t1_min)*rand(N,1);
t2 = t2_min + (t2_max-t2_min)*rand(N,1);
t3 = t3_min + (t3_max-t3_min)*rand(N,1);
t4 = t4_min + (t4_max-t4_min)*rand(N,1);
t5 = t5_min + (t5_max-t5_min)*rand(N,1);
for i = 1:N
A1 = TransMat(a1,alpha1,d1,t1(i));
A2 = TransMat(a2,alpha2,d2,t2(i));
A3 = TransMat(a3,alpha3,d3,t3(i));
A4 = TransMat(a4,alpha4,d4,t4(i));
A5 = TransMat(a5,alpha5,d5,t5(i));

T = A1*A2*A3*A4*A5;

X=T(1,4);
Y=T(2,4);
Z=T(3,4);
plot3(X,Y,Z,'.')
hold on;
end
view(3);
title('Isometric view');
xlabel('x (cm)');
ylabel('y (cm)');
zlabel('z (cm) ');

function T = TransMat( a,b,c,d )
T = [ cos(d) -sin(d)*cos(b) sin(d)*sin(b) a*cos(d); sin(d) cos(d)*cos(b) -cos(d)*sin(b
0 sin(b) cos(b) c;
0 0 0 1
];
end
```

B.3 Simulink Simulation MATLAB Program

This MATLAB program is used to link the user input to the Simulink model and then start the simulation. User have to run the file **ENPM667** and then input the desired joint angles and velocity for the simulation. The simulation will start automatically after this.

```
%Simulation%
```

```
prompt = 'Input Joint 1 Desired Angle in Degree ';
j1 = input(prompt);
prompt = 'Input Joint 2 Desired Angle in Degree ';
j2 = input(prompt);
prompt = 'Input Joint 3 Desired Angle in Degree ';
j3 = input(prompt);
prompt = 'Input Joint 4 Desired Angle in Degree ';
j4 = input(prompt);
prompt = 'Input Joint 5 Desired Angle in Degree ';
j5 = input(prompt);
prompt = 'Input Joint 1 Desired Velocity ';
j1vel = input(prompt);
prompt = 'Input Joint 2 Desired Velocity ';
j2vel = input(prompt);
prompt = 'Input Joint 3 Desired Velocity';
j3vel = input(prompt);
prompt = 'Input Joint 4 Desired Velocity';
j4vel = input(prompt);
prompt = 'Input Joint 5 Desired Veclocity ';
j5vel = input(prompt);
sys = 'ENPM667_FinalProject_Edit';
joint1 = strcat(sys,'/Input1/');
joint2 = strcat(sys,'/Input2/');
joint3 = strcat(sys,'/Input3/');
joint4 = strcat(sys,'/Input4/');
joint5 = strcat(sys,'/Input5/');
joint1velocity = strcat(joint1,'Ramp');
joint2velocity = strcat(joint2,'Ramp');
joint3velocity = strcat(joint3,'Ramp');
joint4velocity = strcat(joint4,'Ramp');
joint5velocity = strcat(joint5,'Ramp');
joint1direction = strcat(joint1,'Gain');
joint2direction = strcat(joint2,'Gain');
joint3direction = strcat(joint3,'Gain');
joint4direction = strcat(joint4,'Gain');
joint5direction = strcat(joint5,'Gain');

joint1angle = strcat(joint1,'Saturation');
joint2angle = strcat(joint2,'Saturation');
joint3angle = strcat(joint3,'Saturation');
joint4angle = strcat(joint4,'Saturation');
joint5angle = strcat(joint5,'Saturation');
open_system(sys);

%joint velocities by modifying slope of ramp input%
set_param(joint1velocity,'Slope',num2str(j1vel));
set_param(joint2velocity,'Slope',num2str(j2vel));
set_param(joint3velocity,'Slope',num2str(j3vel));
set_param(joint4velocity,'Slope',num2str(j4vel));
set_param(joint5velocity,'Slope',num2str(j5vel));

if((strcmp(j1,'NaN')==0)&&(strcmp(j2,'NaN')==0)&&(strcmp(j3,'NaN')==0)&&(strcmp(j4,'Na
     if(sign(str2double(j1))==1)
        set_param(joint1direction,'Gain','1');
        set_param(joint1angle,'UpperLimit',num2str(j1));
    elseif(sign(str2double(j1))==-1)
        set_param(joint1direction,'Gain','-1');
        set_param(joint1angle,'LowerLimit',num2str(j1));
    else
        set_param(joint1direction,'Gain','1');
        set_param(joint1angle,'UpperLimit',num2str(j1));
    end
    if(sign(str2double(j2))==1)
        set_param(joint2direction,'Gain','1');
        set_param(joint2angle,'UpperLimit',num2str(j2));
    elseif(sign(str2double(j2))==-1)
        set_param(joint2direction,'Gain','-1');
        set_param(joint2angle,'LowerLimit',num2str(j2));
    else
        set_param(joint2direction,'Gain','1');
        set_param(joint2angle,'UpperLimit',num2str(j2));
    end
    if(sign(str2double(j3))==1)
        set_param(joint3direction,'Gain','1');
        set_param(joint3angle,'UpperLimit',num2str(j3));
    elseif(sign(str2double(j3))==-1)
        set_param(joint3direction,'Gain','-1');
        set_param(joint3angle,'LowerLimit',num2str(j3));
    else
        set_param(joint3direction,'Gain','1');
        set_param(joint3angle,'UpperLimit',num2str(j3));
    end
    if(sign(str2double(j4))==1)
        set_param(joint4direction,'Gain','1');
        set_param(joint4angle,'UpperLimit',num2str(j4));
    elseif(sign(str2double(j4))==-1)
        set_param(joint4direction,'Gain','-1');
        set_param(joint4angle,'LowerLimit',num2str(j4));
    else
        set_param(joint4direction,'Gain','1');
        set_param(joint4angle,'UpperLimit',num2str(j4));
    end
    if(sign(str2double(j5))==1)
        set_param(joint5direction,'Gain','1');
        set_param(joint5angle,'UpperLimit',num2str(j5));
    elseif(sign(str2double(j5))==-1)
        set_param(joint5direction,'Gain','-1');
        set_param(joint5angle,'LowerLimit',num2str(j5));
    else
        set_param(joint5direction,'Gain','1');
        set_param(joint5angle,'UpperLimit',num2str(j5));
    end
end

  sim(sys);
save_system(sys);
status = 1;
```

B.4 Robotics Tooolbox Validation MATLAB Program

This program is used to validate the theoretical result obtained from forward kinematics analysis. The file name for this program is **toolbox.m**. However, this will require the MATLAB toolbox file [3] to be installed in your device.

```
%validation program for Forward Kinematics%
l1= Link([ 0 15 0 -pi/2]);
l2= Link([ 0 0 10 0]);
l3 =Link([ 0 0 10 0]);
l4 = Link([ 0 0 0 -pi/2]);
l5 =Link([ 0 22 0 0]);
r= SerialLink([l1 l2 l3 l4 l5],'name' ,'DOF')
r.teach([0 0 0 0 0 ]);
```

References

[1] Spong, Mark W., Seth. Hutchinson, and M. Vidyasagar. Robot Modeling and Control. Hoboken, NJ: John Wiley, 2006. Print.

[2] https://www.mathworks.com/help/physmod/smlink/index.html

[3] http://petercorke.com/wordpress/toolboxes/robotics-toolbox

[4] Simulink Simulation : https://www.youtube.com/watch?v=hBRRDCmwHwE