Kinematics Analysis and Simulation of an 6R Industrial Robot using MATLAB

MEKATRONIK MÜHENDISLIĞI

MTM302 ROBOT TEKNIĞI I

Prof. Dr. RAİF BAYIR

Khaled HAMIDI

2110809516

Forward and Inverse Kinematics Analysis and Simulation in MATLAB of an 6R Industrial Robot

Abstract - This study introduces the forward and inverse kinematics of the Collaborative Robot E15, a 6R Industrial Robot created by Han's Robot. Using D-H table of the robot and close loop method to determine the kinematic equations of the manipulator. It also describes an interface created for simulating in MATLAB, and 3D plot of the robot's using Robotics Toolbox will be visualization.

1. Introduction

The Elfin E15 is a Denso type robot consisting of six linked segments connected by six revolute joints. This design features uniform joint types throughout, with no inclusion of prismatic joint variation. Figure 1 presents a diagrammatic representation of the Elfin E15, illustrating the frames attached to its joints and the corresponding axes of joint rotation, which align with the length of the robot's links.

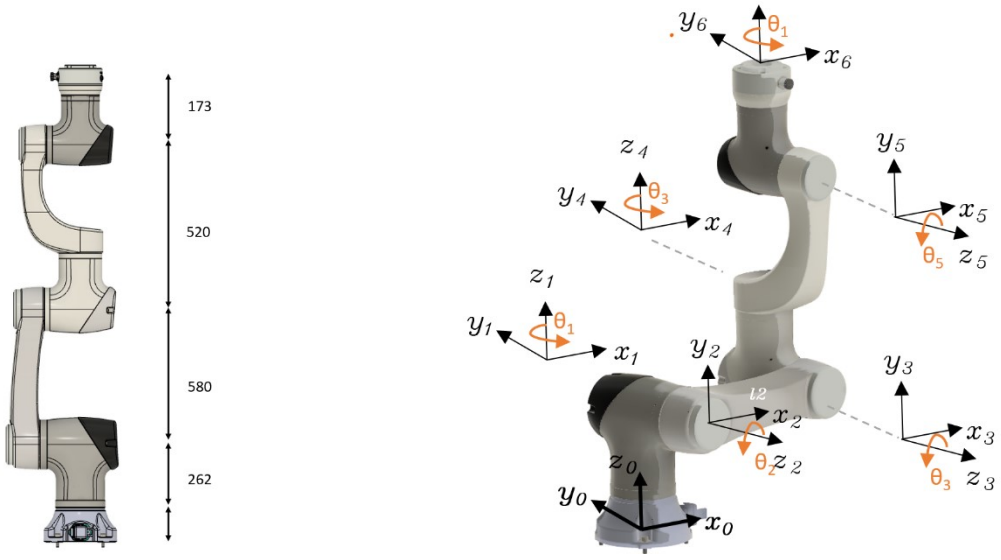


Figure 1: D-H notation for a 6-DOF Elfin E15 robot manipulator.

A pivotal component in the characterization of a robot manipulator is the set of Denavit-Hartenberg parameters, commonly referred to as the D-H parameter table. This table delineates the robot manipulator's specifications, encompassing the lengths of links and the relative orientations of the joints. Table 1 details the D-H parameters specific to the Elfin E15.

Table 1: Elfin E15 robot arm link coordinate parameters

Link	α_{i-1}	a_{i-1}	d_i	θ_i	joint variable
1	0_i	0	h_1	θ_1	θ_1
2	90	0	0	θ_2	θ_2
3	0	a_2	0	θ_3	θ_3
4	-90	0	d_4	θ_4	θ_4
5	$+90$	0	0	θ_5	θ_5
6	-90	0	d_6	θ_6	θ_6

Table 2: Based on the properties of the arm, the values of the lengths are as indicated in.

$\boldsymbol{link}$	$\boldsymbol{Length\ (mm)}$
$\boldsymbol{h_1}$	262
$\boldsymbol{a_2}$	580
$\boldsymbol{d_4}$	520
$\boldsymbol{d_6}$	173

2. Forward Kinematics Analysis

Forward kinematics problem is finding the position and orientation of the end effector of the robot by a given set of joint angles and also having D-H parameters of the robot. This section explains an analytical method for solving the forward kinematics problem of a Elfin E15 . A robot manipulator's forward kinematics problem is solved by attaching a single frame to each joint along with the robot's base. Each frame describes the position and orientation of each joint of the robot relative to the base or any other global coordinate. Attaching these frames to the joints reduces the calculation of the robot's end effector's position and orientation to a coordinate translation problem which is solved by transformation matrices. Therefore, every joint has a position and orientation relative to its previous joint. These relations are described by transformation matrices. A general formulation for calculation of these matrices is as follows:

The D-H transformation matrix for adjacent coordinate frames, i and i-1.

$$\begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_{i-1} \\ \sin(\theta_1)\cos(\alpha_{i-1}) & \cos(\theta_1)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})\, d_i \\ \sin(\theta_1)\sin(\alpha_{i-1}) & \cos(\theta_1)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then the individual transformation matrices for I = 1, 2, 3, 4, 5 and 6 a n d can be easily obtained.

$$ {}^{0}_{1}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 262 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 580 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 520 \\ -\sin(\theta_4) & -\cos(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{6}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_6) & -\sin(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & 173 \\ -\sin(\theta_6) & -\cos(\theta_6) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

So, can find the position vector ${}^{0}_{6}\boldsymbol{P}$ by cross matrix:

$$ {}^{0}_{6}\boldsymbol{T} = {}^{0}_{1}\boldsymbol{T} * {}^{1}_{2}\boldsymbol{T} * {}^{2}_{3}\boldsymbol{T} * {}^{3}_{4}\boldsymbol{T} * {}^{4}_{5}\boldsymbol{T} * {}^{5}_{6}\boldsymbol{T} $$

$$ {}^{0}_{6}\boldsymbol{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} $$

$$ \boldsymbol{p_x} = 580 * c\theta_1 * c\theta_2 + 173 * c\theta_5 * \left(c\theta_1 * c\theta_4 + c\theta_4 * (c\theta_1 * c\theta_2 * c\theta_3 - c\theta_1 * c\theta_2 * c\theta_3)\right) - 173 * c\theta_5 * (c\theta_1 * c\theta_2 * c\theta_3 + c\theta_1 * c\theta_3 * c\theta_2) - 520 * c\theta_1 * c\theta_2 * c\theta_3 - 520 * c\theta_1 * c\theta_3 * c\theta_2 $$

$$ \boldsymbol{p_y} = 580 * c\theta_2 * s\theta_1 - 173 * s\theta_5 * \left(c\theta_1 * s\theta_4 - c\theta_4 * (s\theta_1 * s\theta_2 * s\theta_3 - c\theta_2 * c\theta_3 * s\theta_1)\right) - 173 * c\theta_5 * (c\theta_2 * s\theta_1 * s\theta_3 + c\theta_3 * s\theta_1 * s\theta_2) - 520 * c\theta_2 * s\theta_1 * s\theta_3 - 520 * c\theta_3 * s\theta_1 * s\theta_2 $$

$$ \boldsymbol{p_z} = 580 * s\theta_2 + 520 * c\theta_2 * c\theta_3 - 520 * s\theta_2 * s\theta_3 + 173 * c\theta_5 * (c\theta_2 * c\theta_3 - s\theta_2 * s\theta_3) - 173 * c\theta_4 * s\theta_5 * (c\theta_2 * s\theta_3 + c\theta_3 * s\theta_2) + 262 $$

1. Forword kinematics using MATLAB:

To find the transformation matrix using Denavit-Hartenberg parameters; dh2mat function has been created. Which describe the relative position and orientation between links in a robotic arm. The parameters include alpha (twist angle), a (link length), d (link offset), and theta (joint angle). This matrix is fundamental in robotics for modeling and controlling robot motions.

```
function T = dh2mat(alpha, a, d, theta)
        T = [cos(theta), -sin(theta), 0, a;
        sin(theta)*cos(alpha), cos(theta)*cos(alpha), -sin(alpha), -sin(alpha)*d;
        sin(theta)*sin(alpha), cos(theta)*sin(alpha), cos(alpha), cos(alpha)*d;
        0, 0, 0, 1];
end
```

The code used to extract the transfer matrices:

```m
%% setup workspace :
clear;
close all;
clc;




%% define the robot

%  parameters
h0 = 30;
h1 = 262;     % Height of the first link
d2 = 580;     % Length of the third link
d4 =520;   % Offset of the fourth link
d6 = 173;   % Offset of the sixth link

syms theta1 theta2 theta3 theta4 theta5 theta6;


DH = [      0           0        h1        theta1
                  pi/2        0        0          theta2;
                    0          d2       0          theta3
                  -pi/2        0        d4        theta4
                   pi/2        0        0          theta5
                  -pi/2        0        d6        theta6        ];



disp('D-H Tablo');
DH = simplify(DH)


disp('transformation matrices');

T01 = dh2mat(DH(1,1),DH(1,2),DH(1,3),DH(1,4))
T12 = dh2mat(DH(2,1),DH(2,2),DH(2,3),DH(2,4))
T23 =   dh2mat(DH(3,1),DH(3,2),DH(3,3),DH(3,4))
T34 =  dh2mat(DH(4,1),DH(4,2),DH(4,3),DH(4,4))
T45 = dh2mat(DH(5,1),DH(5,2),DH(5,3),DH(5,4))
T56 = dh2mat(DH(6,1),DH(6,2),DH(6,3),DH(6,4))



% Compute cumulative transformations from the base
T06 = T01 * T12 * T23 * T34 * T45 * T56 ;

disp('T06');
 simplify(T06)

 disp('Px Py Pz : ');
simplify(T06(:,4))


```

Command Window

```
DH =

[    0,   0, 262, theta1]
[ pi/2,   0,   0, theta2]
[    0, 580,   0, theta3]
[-pi/2,   0, 520, theta4]
[ pi/2,   0,   0, theta5]
[-pi/2,   0, 173, theta6]

transformation matrices

T01 =

[cos(theta1), -sin(theta1), 0,   0]
[sin(theta1),  cos(theta1), 0,   0]
[          0,           0, 1, 262]
[          0,           0, 0,   1]


T12 =

[cos(theta2), -sin(theta2),  0, 0]
[          0,           0, -1, 0]
[sin(theta2),  cos(theta2),  0, 0]
[          0,           0,  0, 1]


T23 =

[cos(theta3), -sin(theta3), 0, 580]
[sin(theta3),  cos(theta3), 0,   0]
[          0,           0, 1,   0]
[          0,           0, 0,   1]


T34 =

[ cos(theta4), -sin(theta4), 0,   0]
[           0,           0, 1, 520]
[-sin(theta4), -cos(theta4), 0,   0]
[           0,           0, 0,   1]

T45 =

[cos(theta5), -sin(theta5),  0, 0]
[          0,           0, -1, 0]
[sin(theta5),  cos(theta5),  0, 0]
[          0,           0,  0, 1]


T56 =

[ cos(theta6), -sin(theta6), 0,   0]
[           0,           0, 1, 173]
[-sin(theta6), -cos(theta6), 0,   0]
[           0,           0, 0,   1]
```

T06=

[- sin(theta6)*(cos(theta4)*sin(theta1) - sin(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) - cos(theta6)*(cos(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) + sin(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2))), sin(theta6)*(cos(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) + sin(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) - cos(theta6)*(cos(theta4)*sin(theta1) - sin(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))), sin(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) - cos(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2)), 580*cos(theta1)*cos(theta2) + 173*sin(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) - 173*cos(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) - 520*cos(theta1)*cos(theta2)*sin(theta3) - 520*cos(theta1)*cos(theta3)*sin(theta2)]

[sin(theta6)*(cos(theta1)*cos(theta4) + sin(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) + cos(theta6)*(cos(theta5)*(cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) - sin(theta5)*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2))), cos(theta6)*(cos(theta1)*cos(theta4) + sin(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) - sin(theta6)*(cos(theta5)*(cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) - sin(theta5)*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2))), - sin(theta5)*(cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) - cos(theta5)*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)), 580*cos(theta2)*sin(theta1) - 173*sin(theta5)*(cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) - 173*cos(theta5)*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) - 520*cos(theta2)*sin(theta1)*sin(theta3) - 520*cos(theta3)*sin(theta1)*sin(theta2)]

[cos(theta6)*(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)*cos(theta4)*cos(theta5)) - sin(theta2 + theta3)*sin(theta4)*sin(theta6), - sin(theta6)*(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)*cos(theta4)*cos(theta5)) - sin(theta2 + theta3)*cos(theta6)*sin(theta4), cos(theta2 + theta3)*cos(theta5) - sin(theta2 + theta3)*cos(theta4)*sin(theta5), 520*cos(theta2 + theta3) + 580*sin(theta2) + (173*sin(theta4 - theta5)*sin(theta2 + theta3))/2 - (173*sin(theta2 + theta3)*sin(theta4 + theta5))/2 + 173*cos(theta2 + theta3)*cos(theta5) + 262]

[0, 0, 0, 1]

The P vector:

```
Px Py Pz :

ans =

580*cos(theta1)*cos(theta2) + 173*sin(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) - 173*cos(theta5)*(cos(theta1)*
580*cos(theta2)*sin(theta1) - 173*sin(theta5)*(cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) - 173*cos(theta5)*(cos(theta2)*
                                                                                                                                                     520*cos(theta2 + theta3) + 580*sin(theta2) +

cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) - 520*cos(theta1)*cos(theta2)*sin(theta3) - 520*cos(theta1)*cos(theta3)*sin(theta2)
sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) - 520*cos(theta2)*sin(theta1)*sin(theta3) - 520*cos(theta3)*sin(theta1)*sin(theta2)
(173*sin(theta4 - theta5)*sin(theta2 + theta3))/2 - (173*sin(theta2 + theta3)*sin(theta4 + theta5))/2 + 173*cos(theta2 + theta3)*cos(theta5) + 262
                                                                                                                                                   1
```

2. Matlab simulation:

An application is designed using the built-in App Designer in Matab version 2023a.

A function is created whose input is the angles. and ploting an robot in 3d plan.

Forword kinematics function:

```
function results = robot_fkine(app,q)

cla (app.UIAxes,'reset')


%% define the robot

%  parameters
h1 = 262;
d2 = 580;
d4 =520;
d6 = 173;


DH = [      0           0       h1      q(1)
            pi/2        0       0       q(2)
            0           d2      0       q(3)
            -pi/2       0       d4      q(4)
            pi/2        0       0       q(5)
            -pi/2       0       d6      q(6)
];


T01 = dh2mat(DH(1,1),DH(1,2),DH(1,3),DH(1,4));
T12 = dh2mat(DH(2,1),DH(2,2),DH(2,3),DH(2,4));
T23 = dh2mat(DH(3,1),DH(3,2),DH(3,3),DH(3,4));
T34 = dh2mat(DH(4,1),DH(4,2),DH(4,3),DH(4,4));
T45 = dh2mat(DH(5,1),DH(5,2),DH(5,3),DH(5,4));
T56 = dh2mat(DH(6,1),DH(6,2),DH(6,3),DH(6,4));


%%  Compute cumulative transformations from
the base
T02 = T01 * T12;
T03 = T02 * T23;
T04 = T03 * T34;
T05 = T04 * T45;
T06 = T05 * T56;

% Transformation matrices for each joint
Ts = {T01, T02, T03, T04, T05, T06};

%$ Visualization
figure(app.UIFigure)

axis (app.UIAxes,'equal');
grid (app.UIAxes,'on');
xlabel(app.UIAxes,'X');
ylabel(app.UIAxes,'Y');
zlabel(app.UIAxes,'Z');
title(app.UIAxes,'Robot Visualization');

% Plot the base
plot3(app.UIAxes, 0, 0, 0, 'ko',
'MarkerFaceColor', 'k');
hold(app.UIAxes,"on");
% Draw each joint and link
for i = 1:length(Ts)
    x = Ts{i}(1,4);
    y = Ts{i}(2,4);
    z = Ts{i}(3,4);

    plot3(app.UIAxes,x, y, z, 'ro',
'MarkerFaceColor', 'r'); % Joints

    if i > 1
        x_prev = Ts{i-1}(1,4);
        y_prev = Ts{i-1}(2,4);
        z_prev = Ts{i-1}(3,4);
        plot3(app.UIAxes,[x_prev, x], [y_prev,
y], [z_prev, z], 'b-', 'LineWidth', 2);
    else
        plot3(app.UIAxes,[0, x], [0, y], [0,
z], 'b-', 'LineWidth', 2);
    end
end
hold(app.UIAxes,"on");
grid (app.UIAxes,'on');
 axis (app.UIAxes,[-1000 1000 -1000 1000 -0
1000]);

% End effector

plot3(app.UIAxes,Ts{end}(1,4), Ts{end}(2,4),
Ts{end}(3,4), 'go','MarkerFaceColor', 'g');

hold (app.UIAxes,"off");
end
```

when the button is pressed:

```
function ForwardButtonPushed(app, event)
  q(1) =     str2double(app.theta1EditField.Value)*pi/180;
  q(2) =     str2double(app.theta2EditField.Value)*pi/180;
  q(3) =     str2double(app.theta3EditField.Value)*pi/180;
  q(4) =     str2double(app.theta4EditField.Value)*pi/180;
  q(5) =     str2double(app.theta5EditField.Value)*pi/180;
  q(6) =     str2double(app.theta6EditField.Value)*pi/180;

   app.robot_fkine(q);
        end
    end
```

Simulation result

The positioning of the robot is shown at different angles.:

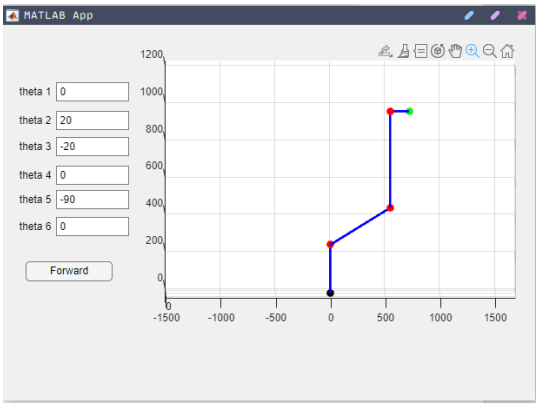


Figure 2

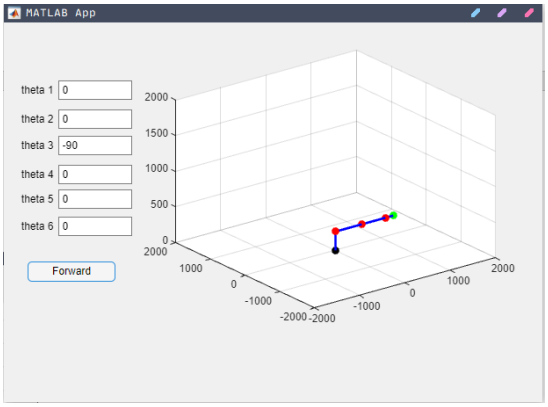


Figure 3

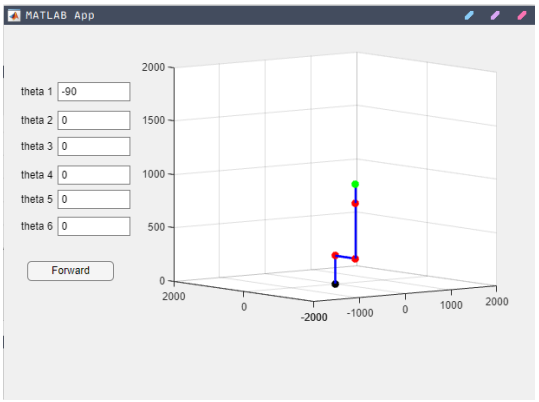


Figure 5

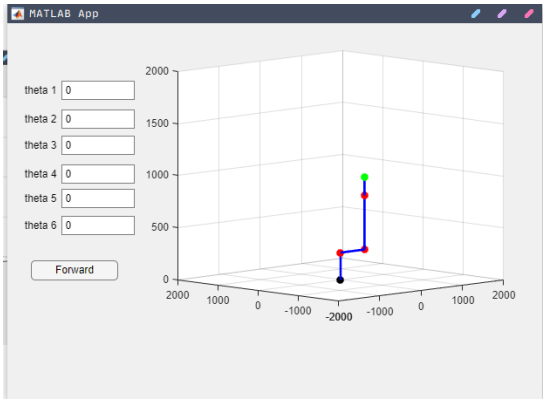


Figure 4

3. Inverse Kinematics Analysis

$$[{}^0_1T]^{-1} * {}^0_6T = {}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T$$

$$[{}^0_1T]^{-1} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & \text{Px} \\ r_{21} & r_{22} & r_{23} & \text{Py} \\ r_{31} & r_{33} & r_{33} & \text{Pz} \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T \qquad (13)$$

where $[{}^0_1T]^{-1} * [{}^0_1T] =$ I is identity matrix. In this case, the above equation is resulted in Eq. (14).

$$\begin{bmatrix} r11 * c\theta_1 + r21 * s\theta_1 & r12 * c\theta_1 + r22 * s\theta_1 & r13 * c\theta_1 + r23 * s\theta_1 & px * c\theta_1 + py * s\theta_1 \\ r21 * c\theta_1 - r11 * s\theta_1 & r22 * c\theta_1 - r12 * s\theta_1 & r23 * c\theta_1 - r13 * s\theta_1 & py * c\theta_1 - px * s\theta_1 \\ r31 & r32 & r33 & pz - 262 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

The required multiplication in Eq. (14) is carried out and it yields as Eq. (15)

$$\begin{bmatrix} . & . & . & 580c\theta_2 - 520c\theta_2 s\theta_3 - 520c\theta_3 s\theta_2 - 173c\theta_5(c\theta_2 s\theta_3 + c\theta_3 s\theta_2) - 173c\theta_4 s\theta_5(c\theta_2 c\theta_3 - s\theta_2 s\theta_3) \\ . & . & -s\theta_4 s\theta_5 & -173 s\theta_4 s\theta_5 \\ . & . & . & 580s\theta_2 + 520c\theta_2 c\theta_3 - 520s\theta_2 s\theta_3 + 173c\theta_5(c\theta_2 c\theta_3 - s\theta_2 s\theta_3) - 173c\theta_4 s\theta_5(c\theta_2 s\theta_3 + c\theta_3 s\theta_2) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To solve for θ_i, we focus on elements that exclusively or dominantly depend on this angle in the equations.

$$p_x c\theta_1 + p_y s\theta_1 = 580c\theta_2 - 520c\theta_2 s\theta_3 - 520c\theta_3 s\theta_2 - 173c\theta_5(c\theta_2 s\theta_3 + c\theta_3 s\theta_2) - 173c\theta_4 s\theta_5(c\theta_2 c\theta_3 - s\theta_2 s\theta_3) \qquad (16)$$

$$p_y c\theta_1 - p_x s\theta_1 = -173 s\theta_4 s\theta_5 \qquad (17)$$

$$p_z - 262 = 0580s\theta_2 + 520c\theta_2 c\theta_3 - 520s\theta_2 s\theta_3 + 173c\theta_5(c\theta_2 c\theta_3 - s\theta_2 s\theta_3) - 173c\theta_4 s\theta_5(c\theta_2 s\theta_3 + c\theta_3 s\theta_2) \qquad (18)$$

Now the orientation matrix 03T can be computed once we have $\theta_1\theta_2\theta_3$ and use it later to find 36 T from equation (19).

$$\begin{bmatrix} c\theta_4 c\theta_5 c\theta_6 - s\theta_4 s\theta_6 & -c\theta_6 s\theta_4 - c\theta_4 c\theta_5 s\theta_6 & r - c\theta_4 s\theta_5 & -173c\theta_4 s\theta_5 \\ c\theta_6 s\theta_5 & -s\theta_5 s\theta_6 & c\theta_5 & 173c\theta_5 + 520 \\ -c\theta_4 s\theta_6 - c\theta_5 * c\theta_6 s\theta_4 & c\theta_5 s\theta_4 s\theta_6 - c\theta_4 c\theta_6 & s\theta_4 s\theta_5 & 173 s\theta_4 s\theta_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (20)$$

$$[{}^0_1T\,{}^2_3T\,{}^1_2T]^{-1} * {}^0_6T = {}^3_4T\,{}^4_5T\,{}^5_6T \quad (19)$$

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ r11 s\theta_1 - r21 c\theta_1 & r12 s\theta_1 - r22 c\theta_1 & r13 s\theta_1 - r23 * c\theta_1 & p_x s\theta_1 - p_y c\theta_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} (21)$$

$$\theta_1 = \arctan\left(\frac{p_x(580cos(\theta_2) + 262)}{{p_x}^2 + {p_y}^2}\right)$$

$$\theta_3 = \arctan(r_{33}, -r_{13}) - \theta_2$$

$$\theta_2 + \theta_3 = \tan^{-1}(p_z - 262, \sqrt{{p_x}^2 + {p_y}^2})$$

$$\theta_4 = \left(\frac{r_{13}\ \sin\theta_1\text{-}a y \cos\theta_1}{r_{13}\ cos(\theta_1)cos(\theta_3) + r_{23}sin\theta_1 + r_{33}\ \sin\theta_2 sin\theta_3}\right)$$

$$\theta_5 = \cos\theta_5 \sin\theta_4 \sin\theta_6\text{-}\cos\theta_4 \cos\theta_6$$

$$\theta_6 = atan\left(-\left(cos(\theta_6)sin(\theta_4) + cos(\theta 4)cos(\theta_5)sin(\theta_6)\right), cos(\theta_4)cos(\theta_5)cos(\theta_6) - sin(\theta_4)sin(\theta_6)\right)$$

4. Visualization using Robotics toolbox:

Create robot using SerialLinks:

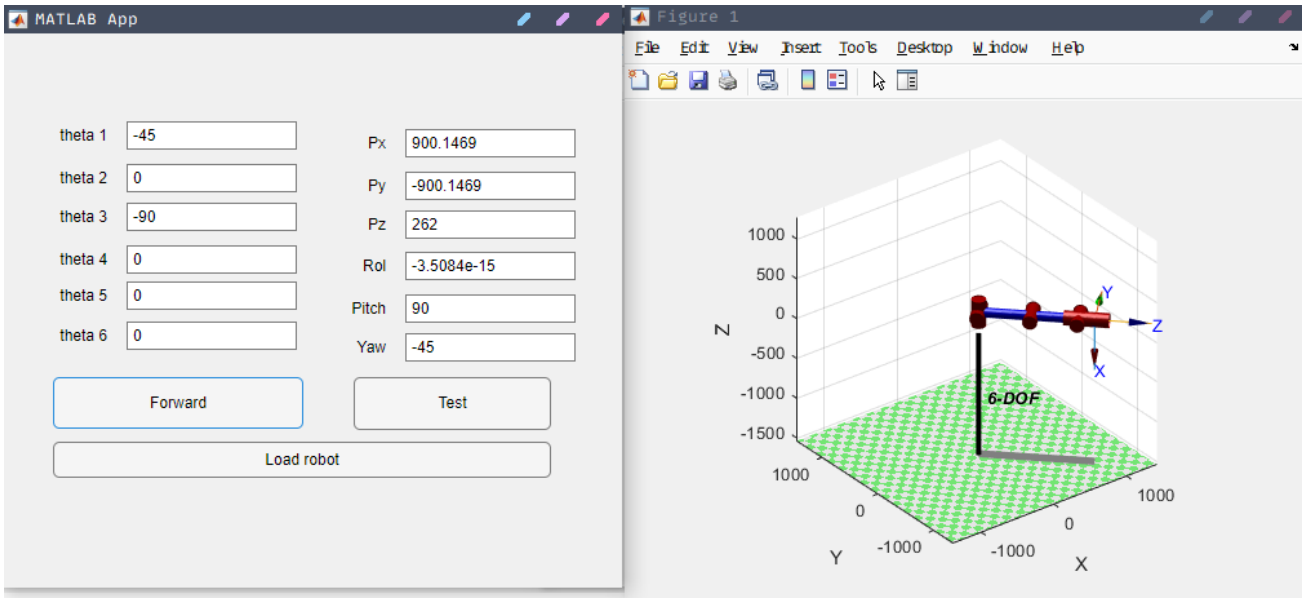
```
%% Using  Robotics toolbox,
function results = CreateRobot(app)
 % Build robot:

 %  parameters
 h1 = 262;
 d2 = 580;
 d4 =520;
 d6 = 173;

 % my robot
 DH = [0 0 h1;pi/2 0 0; 0 d2 0; -pi/2 0 d4;pi/2 0 0;-pi/2 0 d6];

 %% define links
 for i=1:length(DH)
 L(i) = Link('alpha', DH(i,1),'a', DH(i,2), 'd', DH(i,3),'modified');
 end
 % create robot
 app.robot = SerialLink(L,'name','6-DOF');

end
```

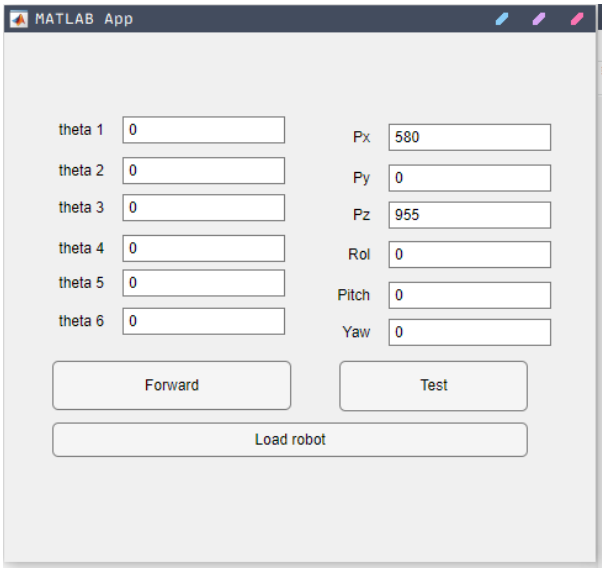
Forword kinematics

```
        % Button pushed function: ForwardButton
        function ForwardButtonPushed(app, event)
      q(1) =     str2double(app.theta1EditField.Value)*pi/180;
      q(2) =     str2double(app.theta2EditField.Value)*pi/180;
      q(3) =     str2double(app.theta3EditField.Value)*pi/180;
      q(4) =     str2double(app.theta4EditField.Value)*pi/180;
      q(5) =     str2double(app.theta5EditField.Value)*pi/180;
      q(6) =     str2double(app.theta6EditField.Value)*pi/180;

      T = fkine(app.robot,q);
      P  = transl(T)
      rol_pitch_yaw = tr2eul(T);

    app.PxEditField.Value = num2str(P(1))
    app.PyEditField.Value = num2str(P(2))
    app.PzEditField.Value = num2str(P(3))

    app.RolEditField.Value = num2str(rol_pitch_yaw(3)*180/pi);
    app.PitchEditField.Value = num2str(rol_pitch_yaw(2)*180/pi);
    app.YawEditField.Value = num2str(rol_pitch_yaw(1)*180/pi);


  app.robot.plot(q);
        end
```

Invars kinematics:

```
100         % Button pushed function: TestButton
101         function TestButtonPushed(app, event)
102                 q0(1) =     str2double(app.theta1EditField.Value)*pi/180;
103                 q0(2) =     str2double(app.theta2EditField.Value)*pi/180;
104                 q0(3) =     str2double(app.theta3EditField.Value)*pi/180;
105                 q0(4) =     str2double(app.theta4EditField.Value)*pi/180;
106                 q0(5) =     str2double(app.theta5EditField.Value)*pi/180;
107                 q0(6) =     str2double(app.theta6EditField.Value)*pi/180;
108
109                 Px =     str2double(app.PxEditField.Value);
110                 Py =     str2double(app.PyEditField.Value);
111                 Pz =     str2double(app.PzEditField.Value);
112
113                 T = transl(Px, Py, Pz);
114
115                 q = app.robot.ikine(T,q0, 'mask', [1 1 1 1 1 1]);
116                 app.robot.plot(q);
117
118                 app.theta1EditField.Value = int2str(q(1)*180/pi);
119                 app.theta2EditField.Value = int2str(q(2)*180/pi);
120                 app.theta3EditField.Value = int2str(q(3)*180/pi);
121                 app.theta4EditField.Value = int2str(q(4)*180/pi);
122                 app.theta5EditField.Value = int2str(q(5)*180/pi);
123                 app.theta6EditField.Value = int2str(q(6)*180/pi);
124
125         end
```




MATLAB App
theta 1
0
theta 2
0
theta 3
0
theta 4
0
theta 5
0
theta 6
0
Px
580
Py
0
Pz
955
Rol
0
Pitch
0
Yaw
0
Forward
Test
Load robot

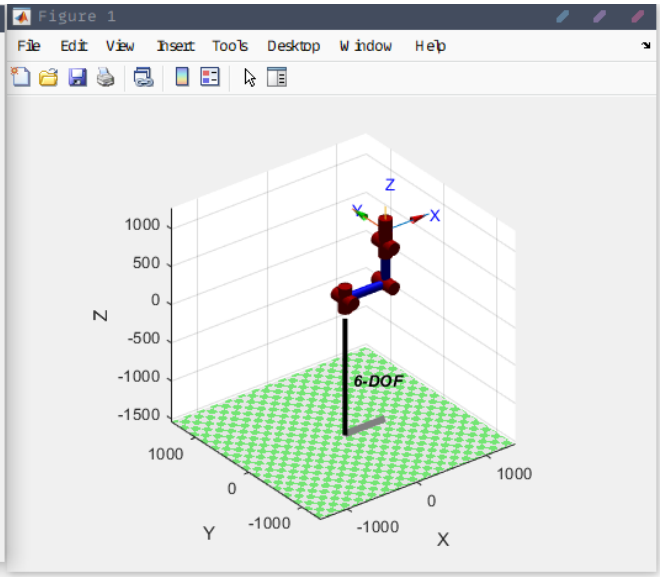
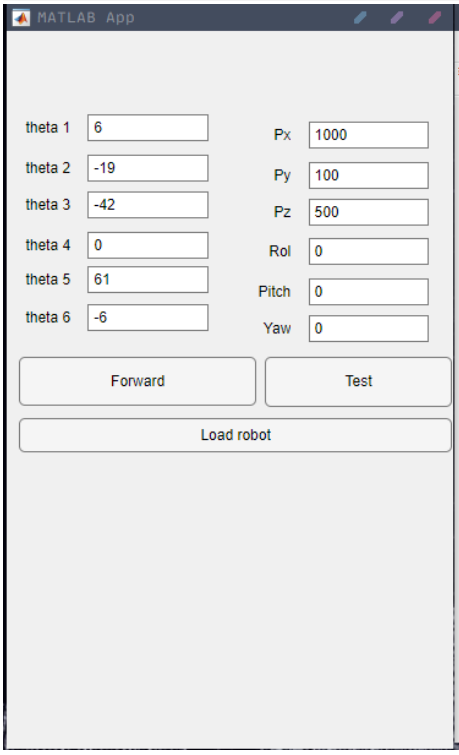


Figure 1
File Edit View Insert Tools Desktop Window Help
Z
X
Y
1000
500
0
-500
-1000
-1500
6-DOF


MATLAB App
theta 1
6
theta 2
-19
theta 3
-42
theta 4
0
theta 5
61
theta 6
-6
Px
1000
Py
100
Pz
500
Rol
0
Pitch
0
Yaw
0
Forward
Test
Load robot

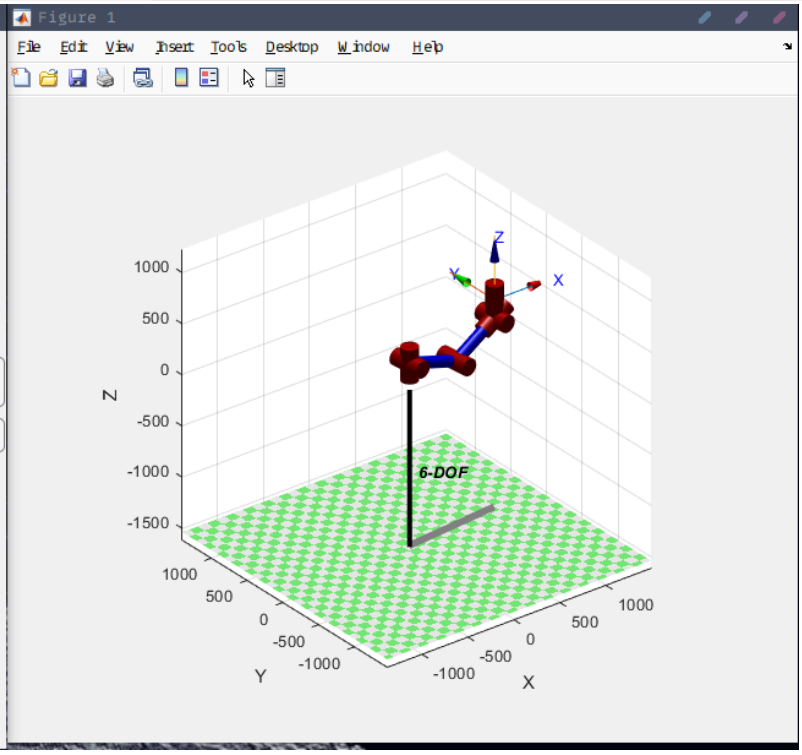
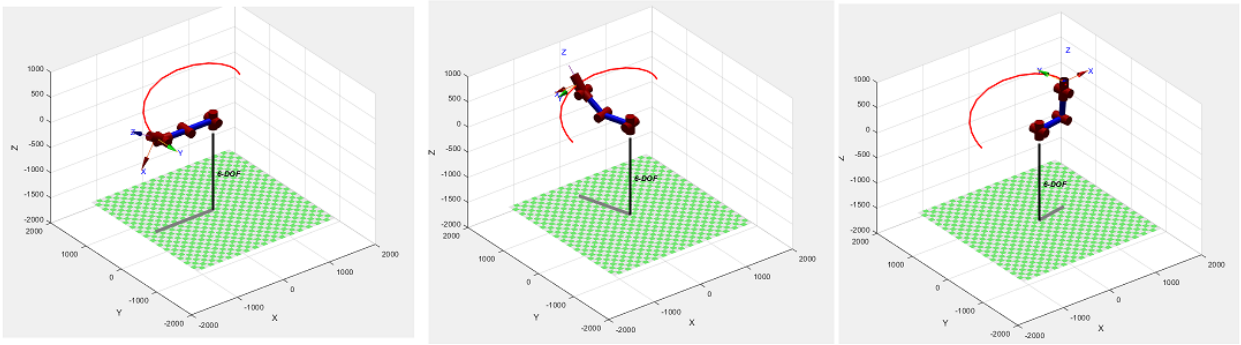


Figure 1
File Edit View Insert Tools Desktop Window Help
Z
X
Y
1000
500
0
-500
-1000
-1500
6-DOF

5. Motion in 3d plan using forward and inverse kinematics.

the robot trajectory planning is conducted via a combination of. Initially, the target joint configuration is specified (q_f), Subsequently, starting from an initial zero configuration (q_0). Forward kinematics are recalculated for each interpolated set of joint angles to ascertain the end-effector's spatial trajectory, which is extracted and visualized in a three-dimensional plot. The script concludes by animating the manipulator's motion along this trajectory, facilitating a comprehensive visualization of dynamic robotic manipulation.

```
%% Invarse kinematics WAY 1

qf = [pi 0 -pi/2 0 pi/6 0];

Tf= robot.fkine(qf);

q0 = [0 0 0 0 0 0];
q = robot.ikine(Tf,q0, 'mask',[1 1 1 1 1 1]);

t=0:0.15:3;
Q = jtraj(q0,qf,t);
Tr= fkine(robot,Q);

for( i =1:1:length(t))
    T = Tr(i);
    trs  = transl(T);
        xx(i) = trs(1);
        yy(i) = trs(2);
        zz(i) = trs(3);
end
plot3(xx,yy,zz,'Color',[1 0 0],'LineWidth',2);
hold on;

plot(robot,Q);

```



The previous figures show the movement of the robot during its movement