Imperial College London

DE3-ROB1 ROBOTICS 1

Tutorial 3: 3D Kinematics

Marcin Laskowski

supervised by
Dr Petar Kormushev
Dr Thrishantha Nanayakkara

November 21, 2017

1 Abstract

During the third Tutorial classes it was given ...

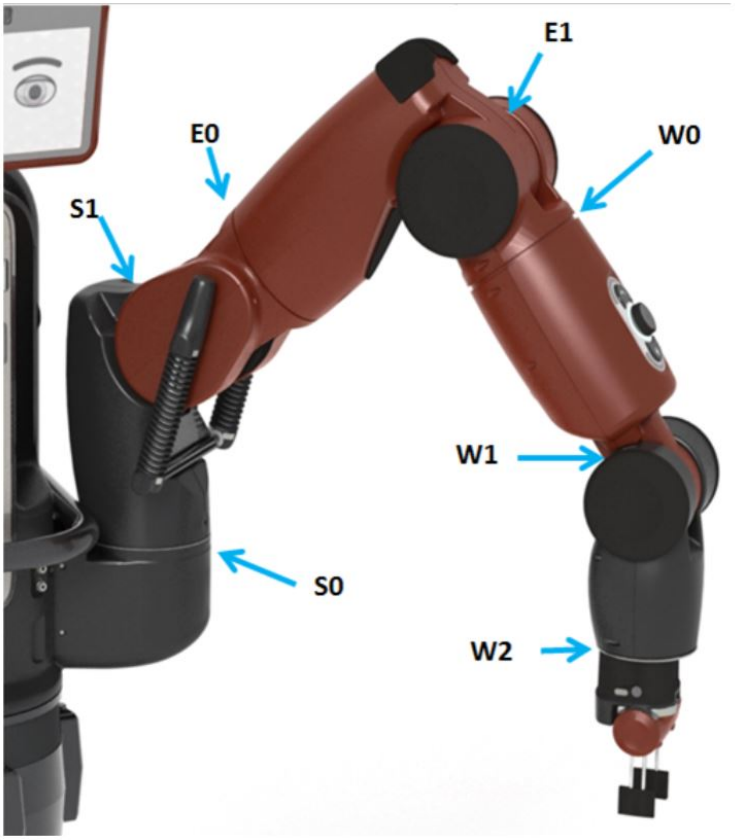

Figure 1: Baxter robot arm with 7 degrees of freedom

The goal of the tutorial was to calculate the 3D Kinematic equations.
Taking into account that each joint has one degree of freedom, the action of each joint can be described by one real number: the angle of rotation in the case of a rotating member or displacement in the case of a prismatic member. An approach based on classical mechanics equations can be used to describe a robotic kinematics, or to use a suitable convention of calculation. In this tutorial Denavit-Hartenberg notation was used. Subordination of this convention makes it possible to determine the equations of kinematics.

2 TASK 1

In the first task of the tutorial classes was determined the dynamics of such a system using the Lagrange formulation.

2.1 Homogeneous transformation matrix for the Baxter arm

To obtain Homogenous Transformation Matrix following code was implemented in matlab software:

Listing 1: Calculating homogoneus Transformation Matrix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Robotic_tut3_1a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Denavit-Hartenberg Notation
%RotZ = [cos(theta) sin(theta) 0 0; sin(theta) cos(theta) 0 0; 0 0 1 0; 0 0 0 1]
%TransZ = [1 0 0 0; 0 1 0 0; 0 0 1 d; 0 0 0 1]
%TransX = [1 0 0 a; 0 1 0 0; 0 0 1 0; 0 0 0 1]
%RotX = [1 0 0 0; 0 cos(alpha) -sin(alpha) 0; 0 sin(alpha) cos(alpha) 0; 0 0 0 1];
% A = RotZ * TransZ * TransX * RotX;

syms theta1 theta2 theta3 theta4 theta5 theta6 theta7
syms alpha1 alpha2 alpha3 alpha4 alpha5 alpha6 alpha7
syms a1 a2 a3 a4 a5 a6 a7
syms d1 d2 d3 d4 d5 d6 d7

% a1 = 0;        alpha1 = 0;        d1 = 0.2703;
% a2 = 0.069;    alpha2 = -1.571;   d2 = 0;
% a3 = 0;        alpha3 = 1.571;    d3 = 0.3644;
% a4 = 0.069;    alpha4 = -1.571;   d4 = 0;
% a5 = 0;        alpha5 = 1.571;    d5 = 0.3743;
% a6 = 0.01;     alpha6 = -1.571;   d6 = 0;
% a7 = 0;        alpha7 = 1.571;    d7 = 0.2295;

A1 = [1 0 0 0; 0 1 0 0; 0 0 1 d1; 0 0 0 1]*[cos(theta1) -sin(theta1) 0 0; sin(theta1) cos(theta1) 0 0; 0 0 1 0; 0 0 0 1];
A2 = [1 0 0 a2; 0 1 0 0; 0 0 1 0; 0 0 0 1]*[1 0 0 0; 0 cos(alpha2) -sin(alpha2) 0; 0 sin(alpha2) cos(alpha2) 0; 0 0 0 1]*[cos(theta2) -sin(theta2) 0 0; sin(theta2) cos(theta2) 0 0; 0 0 1 0; 0 0 0 1];
A3 = [1 0 0 0; 0 cos(alpha3) -sin(alpha3) 0; 0 sin(alpha3) cos(alpha3) 0; 0 0 0 1]*[1 0 0 0; 0 1 0 0; 0 0 1 d3; 0 0 0 1]*[cos(theta3) -sin(theta3) 0 0; sin(theta3) cos(theta3) 0 0; 0 0 1 0; 0 0 0 1];
A4 = [1 0 0 a4; 0 1 0 0; 0 0 1 0; 0 0 0 1]*[1 0 0 0; 0 cos(alpha4) -sin(alpha4) 0; 0 sin(alpha4) cos(alpha4) 0; 0 0 0 1]*[cos(theta4) -sin(theta4) 0 0; sin(theta4) cos(theta4) 0 0; 0 0 1 0; 0 0 0 1];
A5 = [1 0 0 0; 0 cos(alpha5) -sin(alpha5) 0; 0 sin(alpha5) cos(alpha5) 0; 0 0 0 1]*[1 0 0 0; 0 1 0 0; 0 0 1 d5; 0 0 0 1]*[cos(theta5) -sin(theta5) 0 0; sin(theta5) cos(theta5) 0 0; 0 0 1 0; 0 0 0 1];
A6 = [1 0 0 a6; 0 1 0 0; 0 0 1 0; 0 0 0 1]*[1 0 0 0; 0 cos(alpha6) -sin(alpha6) 0; 0 sin(alpha6) cos(alpha6) 0; 0 0 0 1]*[cos(theta6) -sin(theta6) 0 0; sin(theta6) cos(theta6) 0 0; 0 0 1 0; 0 0 0 1];
A7 = [1 0 0 0; 0 cos(alpha7) -sin(alpha7) 0; 0 sin(alpha7) cos(alpha7) 0; 0 0 0 1]*[1 0 0 0; 0 1 0 0; 0 0 1 d7; 0 0 0 1]*[cos(theta7) -sin(theta7) 0 0; sin(theta7) cos(theta7) 0 0; 0 0 1 0; 0 0 0 1];

% homogeneous transformation matrix
T = A1 * A2 * A3 * A4 * A5 * A6 * A7;

% subs(T,{a1,a2,a3,a4,a5,a6,a7},{0,0.069,0,0.069,0,0.01,0})
% T = ans;
% subs(T,{d1,d2,d3,d4,d5,d6,d7},{0.2703,0,0.3644,0,0.3743,0,0.2295})
% T = ans;
```

```
% subs(T,{alpha1,alpha2,alpha3,alpha4,alpha5,alpha6,alpha7
    },{0,-1.571,1.571,-1.571,1.571,-1.571,1.571})
% T = ans;
% subs(T,{theta1,theta2,theta3,theta4,theta5,theta6,theta7},{0,-0.55,0,1.2840,0,0.2616,0})
% T = ans;
```

Final furmula of the Homogenous Transformation Matrix is very complex.

2.2 The Jacobian matrix

Jacobian Matrix was obtain thank to calculations of the partial derivatives of the angles. Formula for that calculations is presented below:

Listing 2: Calculating homogoneus Transformation Matrix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Robotic_tut3_1b
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% calculating partial derivatives of the Jacobian Matrix
J11 = jacobian(T(1,4),theta1);
J12 = jacobian(T(1,4),theta2);
J13 = jacobian(T(1,4),theta3);
J14 = jacobian(T(1,4),theta4);
J15 = jacobian(T(1,4),theta5);
J16 = jacobian(T(1,4),theta6);
J17 = jacobian(T(1,4),theta7);

J21 = jacobian(T(2,4),theta1);
J22 = jacobian(T(2,4),theta2);
J23 = jacobian(T(2,4),theta3);
J24 = jacobian(T(2,4),theta4);
J25 = jacobian(T(2,4),theta5);
J26 = jacobian(T(2,4),theta6);
J27 = jacobian(T(2,4),theta7);

J31 = jacobian(T(3,4),theta1);
J32 = jacobian(T(3,4),theta2);
J33 = jacobian(T(3,4),theta3);
J34 = jacobian(T(3,4),theta4);
J35 = jacobian(T(3,4),theta5);
J36 = jacobian(T(3,4),theta6);
J37 = jacobian(T(3,4),theta7);

% Jacobian Matrix
J = [J11 J12 J13 J14 J15 J16 J17; J21 J22 J23 J24 J25 J26 J27; J31 J32 J33 J34 J35 J36 J37];
J = simplify(J);

% substituting given values from the D-H table
subs(J,{a1,a2,a3,a4,a5,a6,a7},{0,0.069,0,0.069,0,0.01,0})
J = ans;
subs(J,{d1,d2,d3,d4,d5,d6,d7},{0.2703,0,0.3644,0,0.3743,0,0.2295})
J = ans;
subs(J,{alpha1,alpha2,alpha3,alpha4,alpha5,alpha6,alpha7
    },{0,-1.571,1.571,-1.571,1.571,-1.571,1.571})
J = ans;
```

2.3 Joint angle profiles

Jacobian matrix was used to obtain the joint angle profiles to trace a circle

Listing 3: Calculating Joint Angles Profiles to trace a circle

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Robotic_tut3_1c
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% sampling step and time
Time = 4;
dt = 0.002;
t = 0:dt:Time;

% position
xx = 0.05*cos(0.5*pi*t);
xy = 0.05*sin(0.5*pi*t);
xz = zeros(1,length(t));
X = [xx; xy; xz];

% velocity
vx = 0.05*(-sin(0.5*pi*t))*(0.5*pi);
vy = 0.05*(cos(0.5*pi*t))*(0.5*pi);
vz = zeros(1,length(t));
V = [vx; vy; vz];

% angle ranges
S0 = [-1.7016, 1.7016];
S1 = [-2.147, 1.047];
E0 = [-3.0541, 3.0541];
E1 = [-0.05, 2.618];
W0 = [-3.059, 3.059];
W1 = [-1.5707, 2.094];
W2 = [-3.059, 3.059];

Angle_ranges = [S0; S1; E0; E1; W0; W1; W2];

% Initial Q (Q = [q1; q2; q3; q4; q5; q6; q7])
LowerBound = [Angle_ranges(:,1)];
Range = -Angle_ranges(:,1) + Angle_ranges(:,2);

Q = [];
Q = LowerBound + 0.5*Range;

% Inverse Kinematics of differential motion: dQ = pinv(J)*V

for i = 1:length(t)

    loading = i
    J_temp = subs(J,{theta1,theta2,theta3,theta4,theta5,theta6,theta7},{Q(1,i),Q(2,i),Q(3,i)
        ,Q(4,i),Q(5,i),Q(6,i),Q(7,i)});
    JJ = double(J_temp);

    % VELOCITY - DESIRED dQ
    dQ(:,i) = pinv(JJ) * V(:,i);

    % POSITION - DESIRED Q
    Q(:,i+1) = Q(:,i) + dQ(:,i)*dt;

end



% Calculating the x, y and z for the end effector drawing circle
Txyz = [T(1,4); T(2,4); T(3,4)]

subs(Txyz,{a1,a2,a3,a4,a5,a6,a7},{0,0.069,0,0.069,0,0.01,0})
Txyz = ans;
```

```
subs(Txyz,{d1,d2,d3,d4,d5,d6,d7},{0.2703,0,0.3644,0,0.3743,0,0.2295})
Txyz = ans;
subs(Txyz,{alpha1,alpha2,alpha3,alpha4,alpha5,alpha6,alpha7
    },{0,-1.571,1.571,-1.571,1.571,-1.571,1.571})
Txyz = ans;

TT_NEW = [];

for i = 1:length(t)

    loading_XYZ = i
    Txyz_temp = subs(Txyz,{theta1,theta2,theta3,theta4,theta5,theta6,theta7},{Q(1,i),Q(2,i),
        Q(3,i),Q(4,i),Q(5,i),Q(6,i),Q(7,i)});
    TT = double(Txyz_temp);
    TT_NEW(:,i) = TT;

    x(i) = TT(1,1);
    y(i) = TT(2,1);
    z(i) = TT(3,1);

end
```

End-effector of the arm has drown the cirlce.

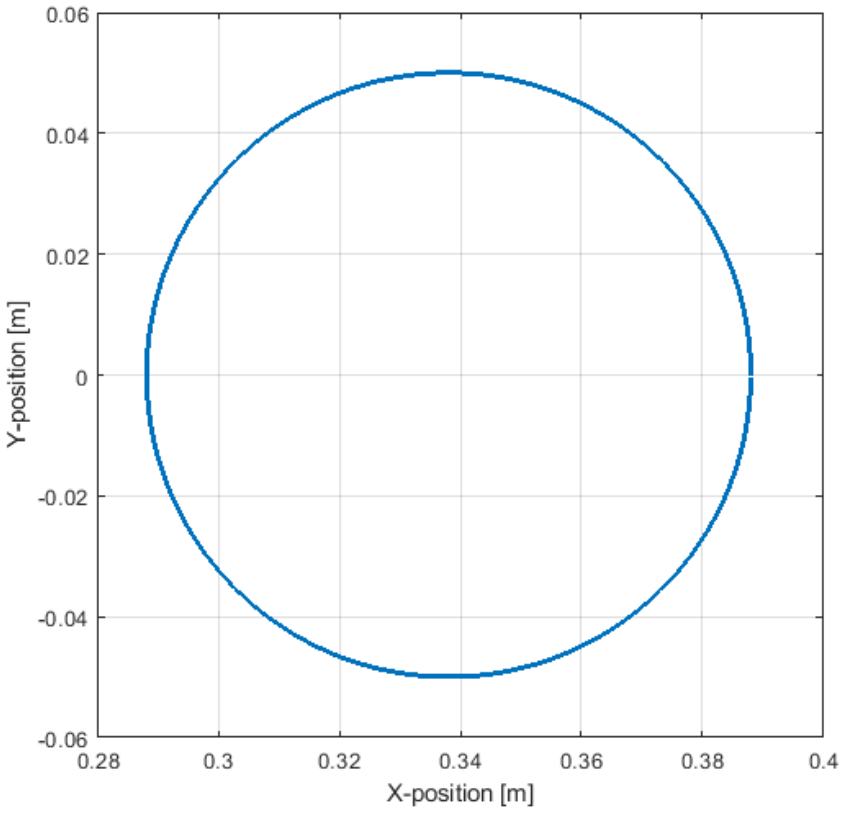

Figure 2: Plot of the circle traced by the end effector

2.4 Plot of the joint angle profiles and end-effector speed profile

Joint angles profiles and end-effector profiles looks as follows.

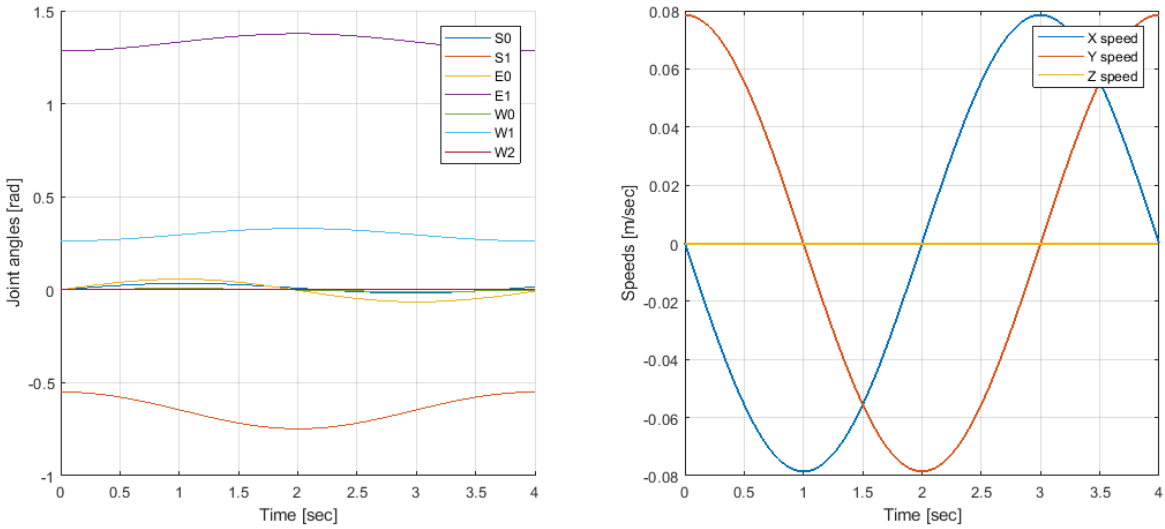

Figure 3: Plot of the Joint angles and end-effector speed

2.5 Different initial joint configuration

Previous step has been repeated for different initial joint configuration: a) Initial $Q = LowerBound$ b) Initial $Q = LowerBound + Range$ c) Initial $Q = LowerBound + 0.2 * Range$

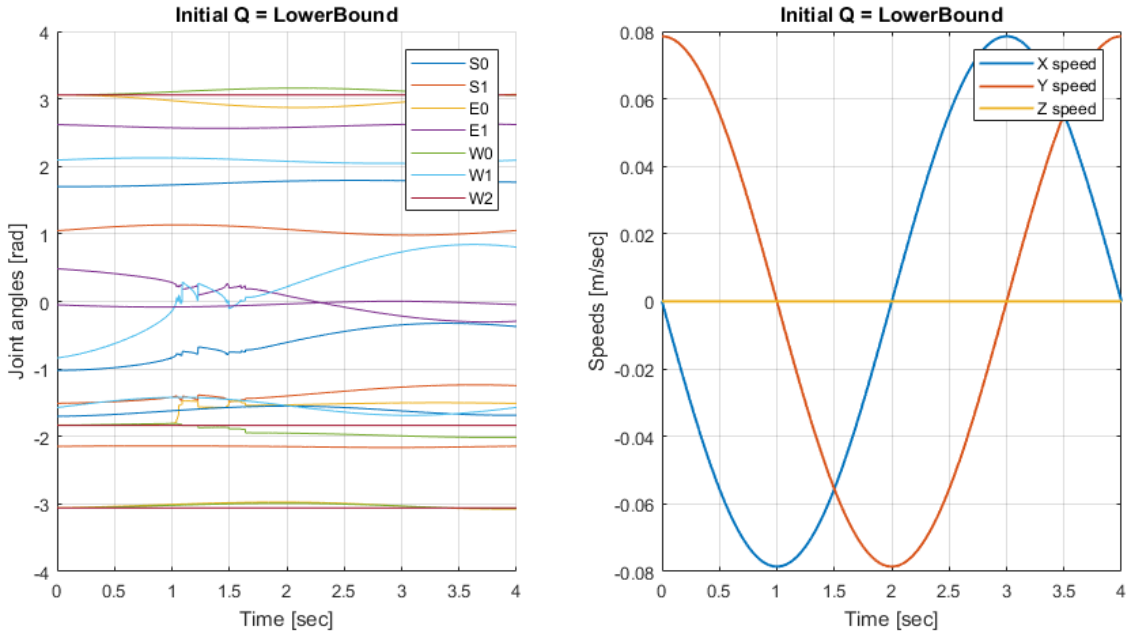

Figure 4: Plot of the Joint angles and end-effector speed when Q = LowerBound

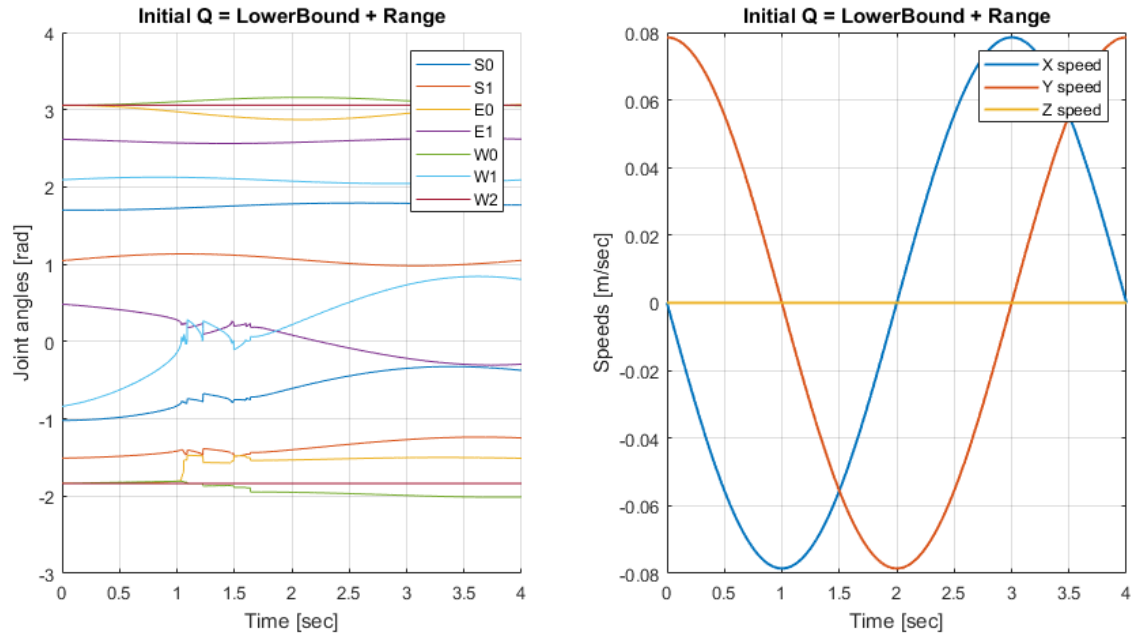

Figure 5: Plot of the Joint angles and end-effector speed when Q = LowerBound + Range

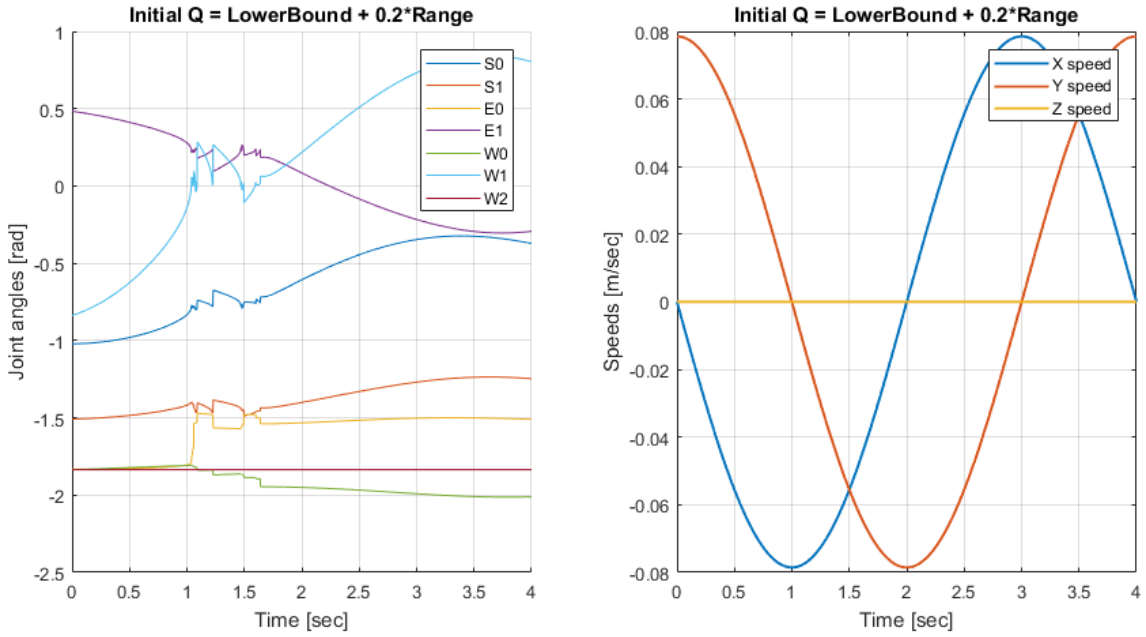

Figure 6: Plot of the Joint angles and end-effector speed when Q = LowerBound + 0.2*Range

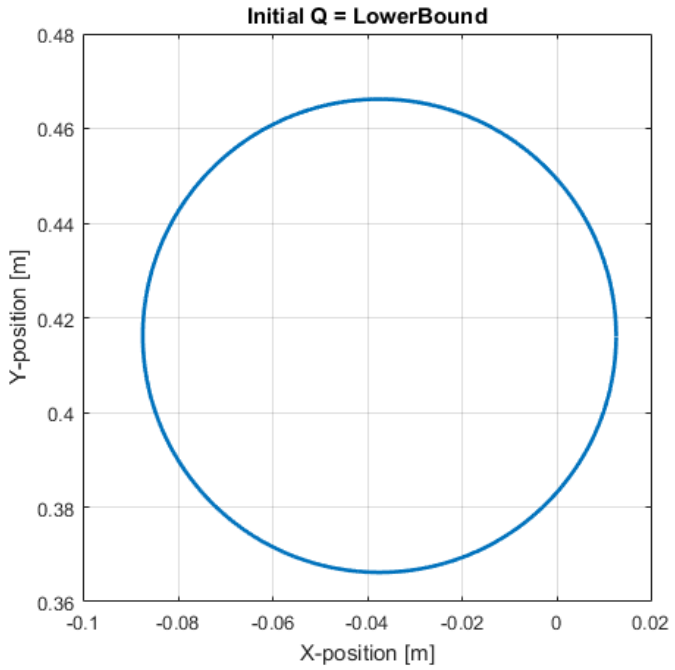

Figure 7: Plot of the circle traced by the end effector when Q = LowerBound

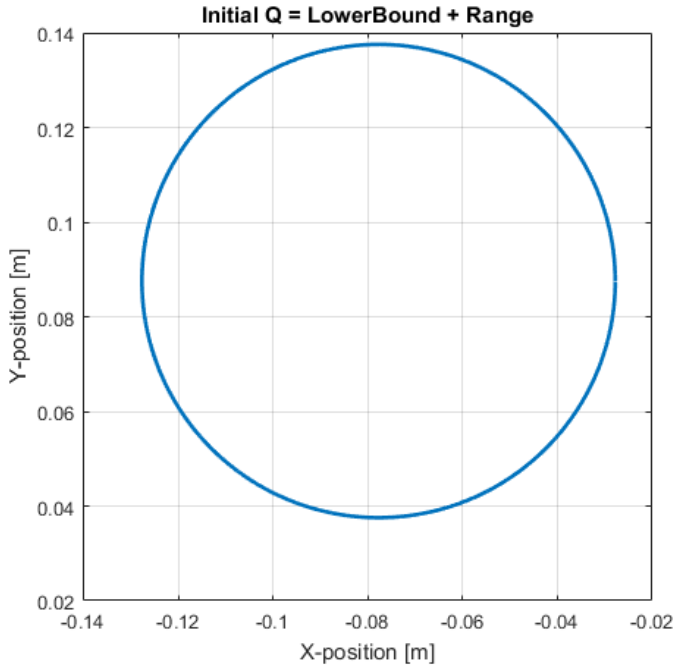

Figure 8: Plot of the circle traced by the end effector when Q = LowerBound + Range

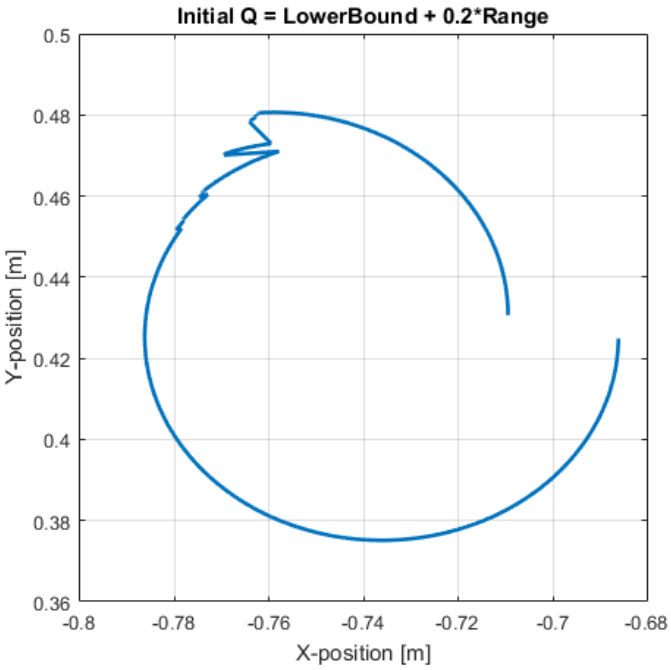

Figure 9: Plot of the circle traced by the end effector when Q = LowerBound + 0.2*Range

When the robot reaches the maximum point (the limit of the workspace) the determinant of the Jacobian becomes zero and arm cannot reaches further points. However it also important to notice that when such a situation occurs it is very hard for the robot to come back to previous positions because the joints are blocked and there is very big amount of possible solutions when it comes to position. Such a situation is called Singularity. Singularity occurs when the arm of the robot is fully extended. During such a situation joint becomes further extended, higher joint speeds are required to establish constant cartesian speed. However robot cannot extend more due to limit of the workspace. Singularity occurs also when the axes of the joints are aligned.

3 TASK 2

In the second task it was necessary to check if the shape of the circle $x(t) = 0.05cos(0.5\pi t);\ y(t) = 0.05sin(0.5\pi t);\ z(t) = 0$ traced by the end-effector looks like the following:

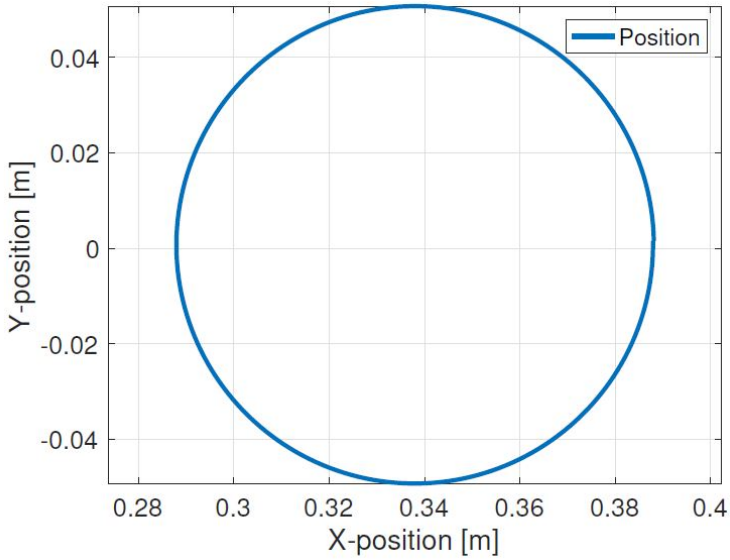

Figure 10: Plot of the circle traced by the end effector given in the Tutorial

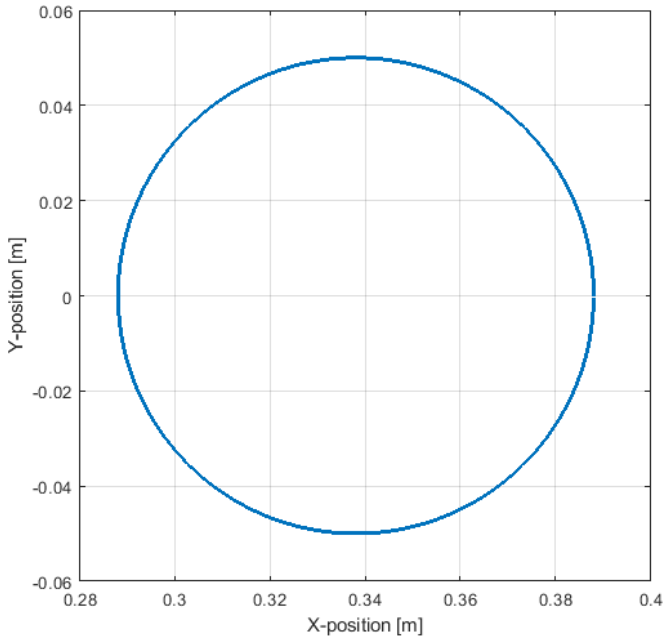

Figure 11: Plot of the circle traced by the end effector given by matlab

Both plots looks exactly the same.

4 TASK 3

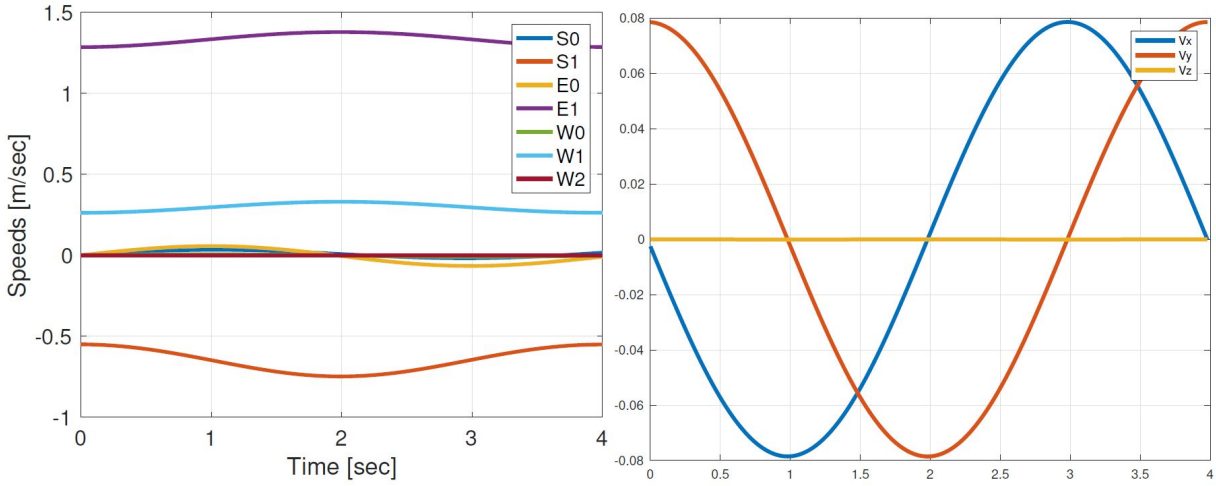

Figure 12: Plot of the Joint angles and end-effector speed given in the Tutorial

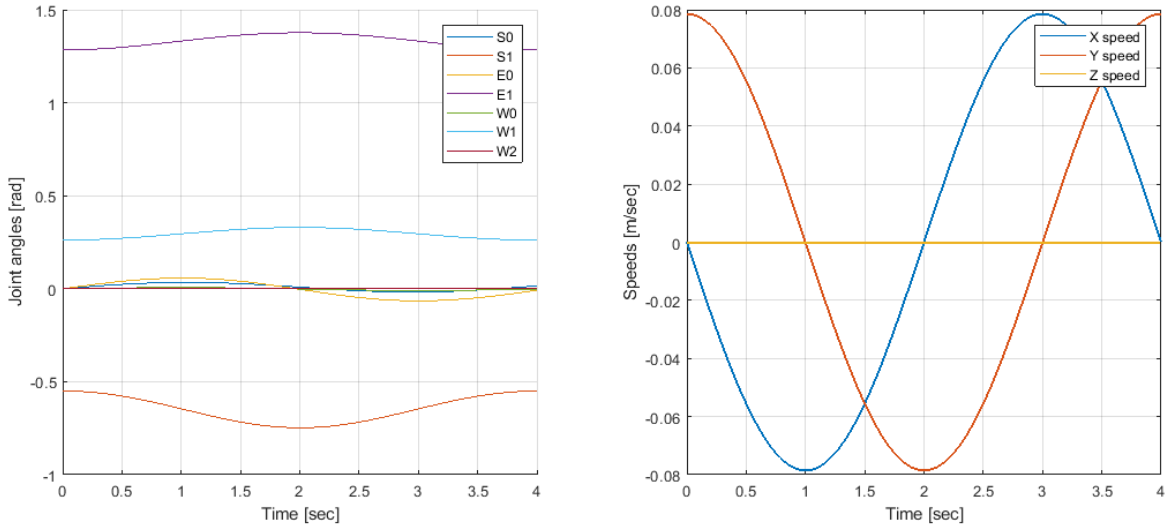

Figure 13: Plot of the Joint angles and end-effector speed given by matlab

Both plots looks exactly the same.

5 TASK 4

5.1 Additional plots

Thank to all calculations it is possible to extract other plots. For example plot of the position of the end-effector x and y cooridnate with respect to time, plot of the end-effector velocity on the x and y axis with respect to time,and acceleration of the end effector.

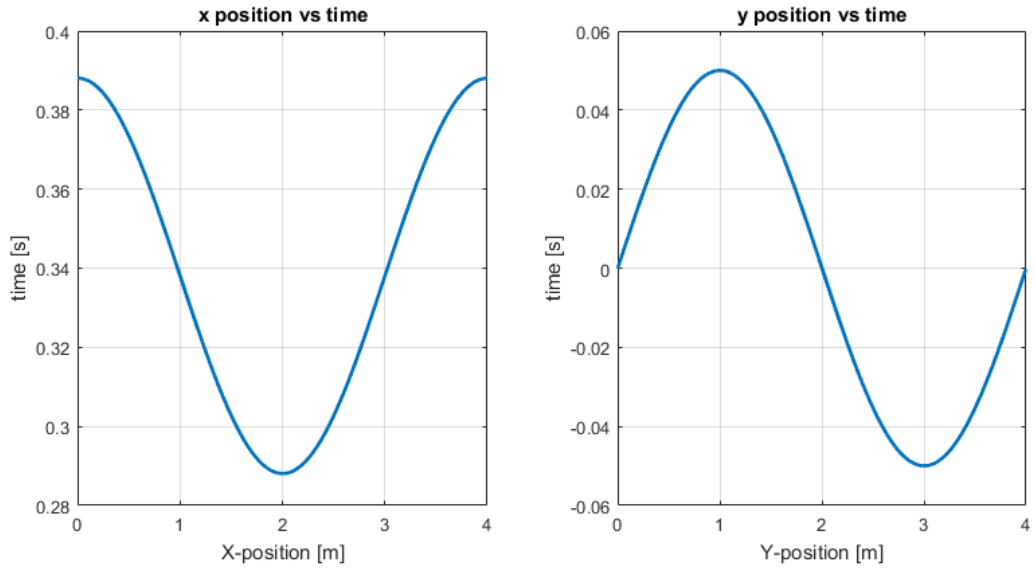

Figure 14: Plot of the Position of the end-effector x axis and y axis with rescpect in time

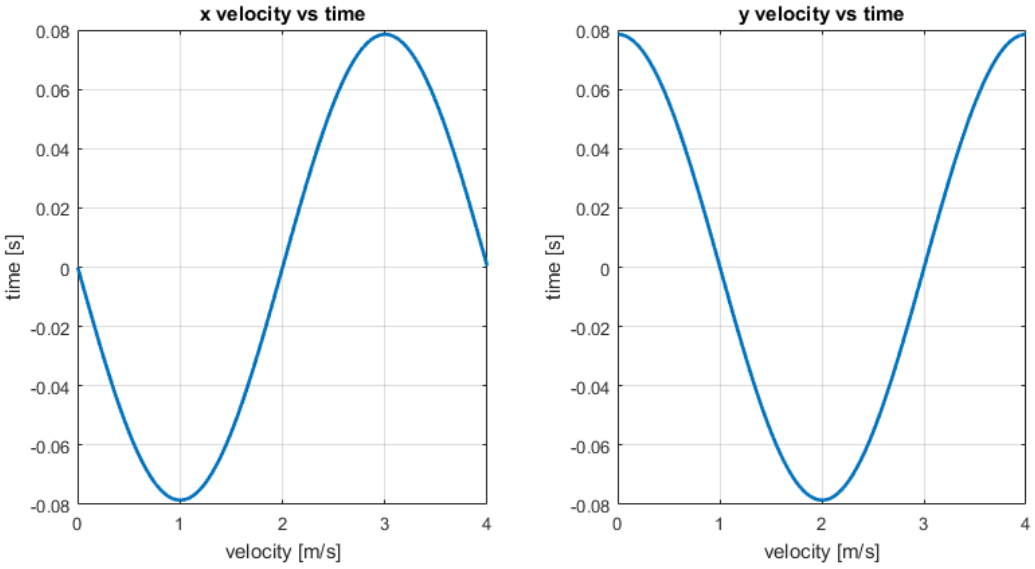

Figure 15: Plot of the Velocity of the end-effector x axis and y axis with rescpect in time

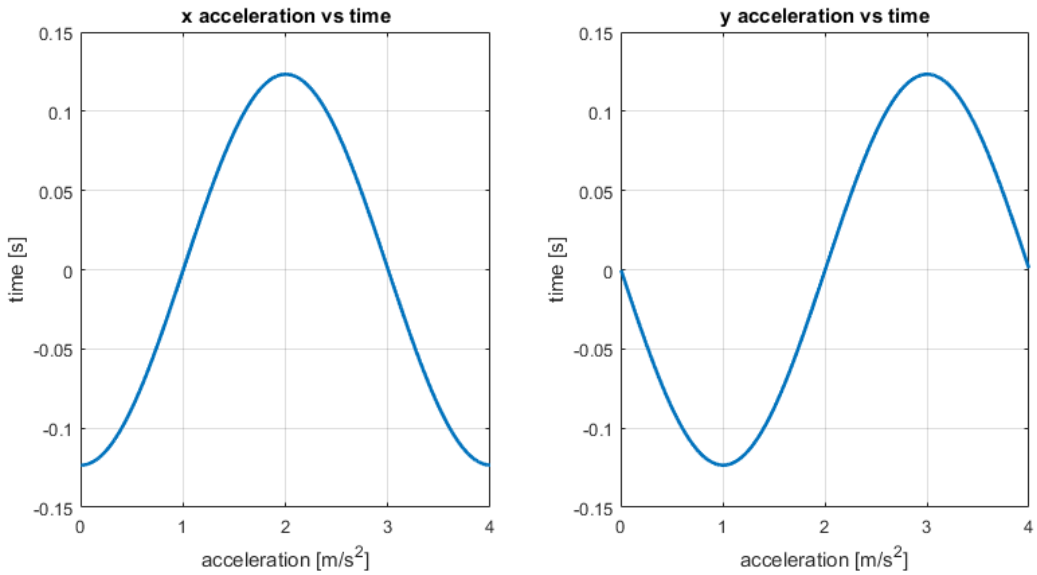

Figure 16: Plot of the Acceleration of the end-effector x axis and y axis with rescpect in time

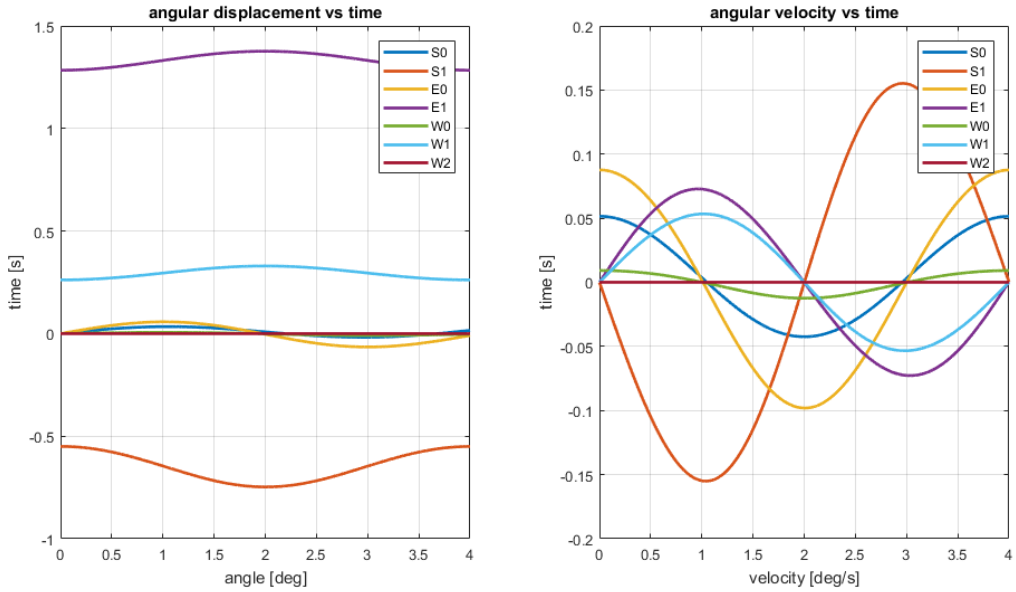

Figure 17: Plot of the Angular position and Angular velocity of all joints with respect to time

5.2 Converting path from SolidWorks to Robot Arm

Having calculated forward and Inverse Kinematics for the robot is it possible to draw the desire shape in the CAD software and than convert to the path for the robot. For that purpose SolidWorks software was used.

First step was to create the shape in the CAD software.

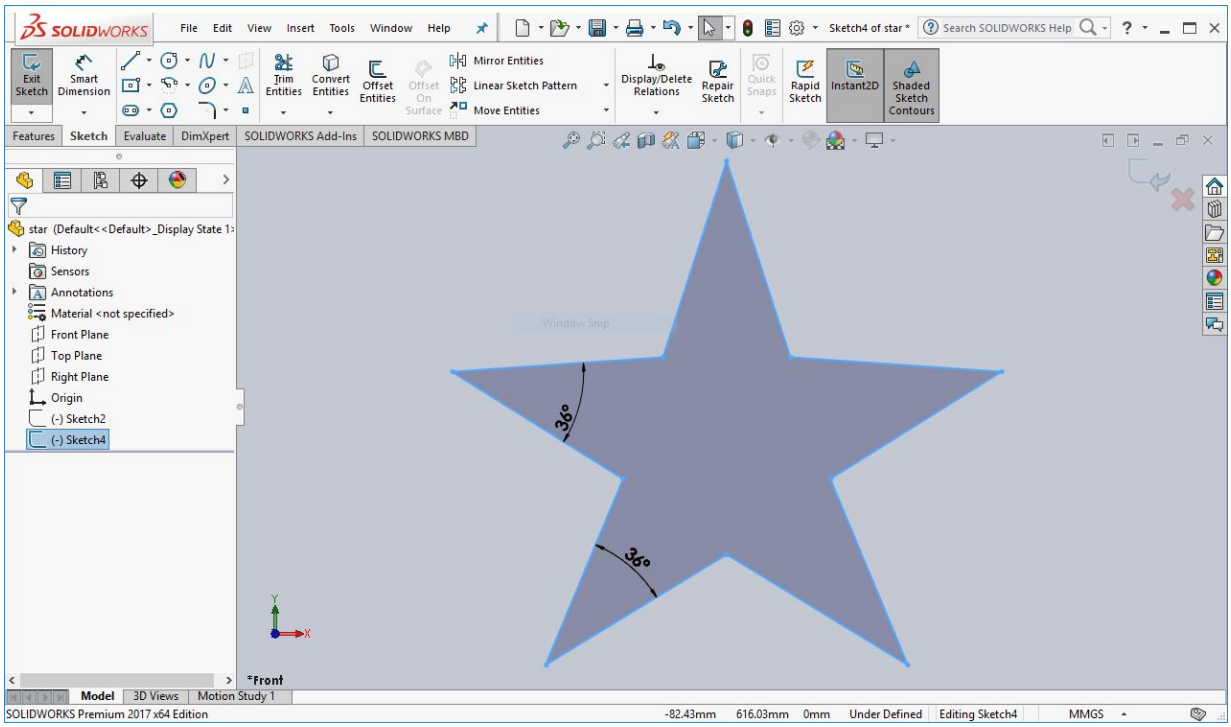

Figure 18: Shape of the star create in the SolidWorks

The next step was to convert that shape into points and save them as the separate file.

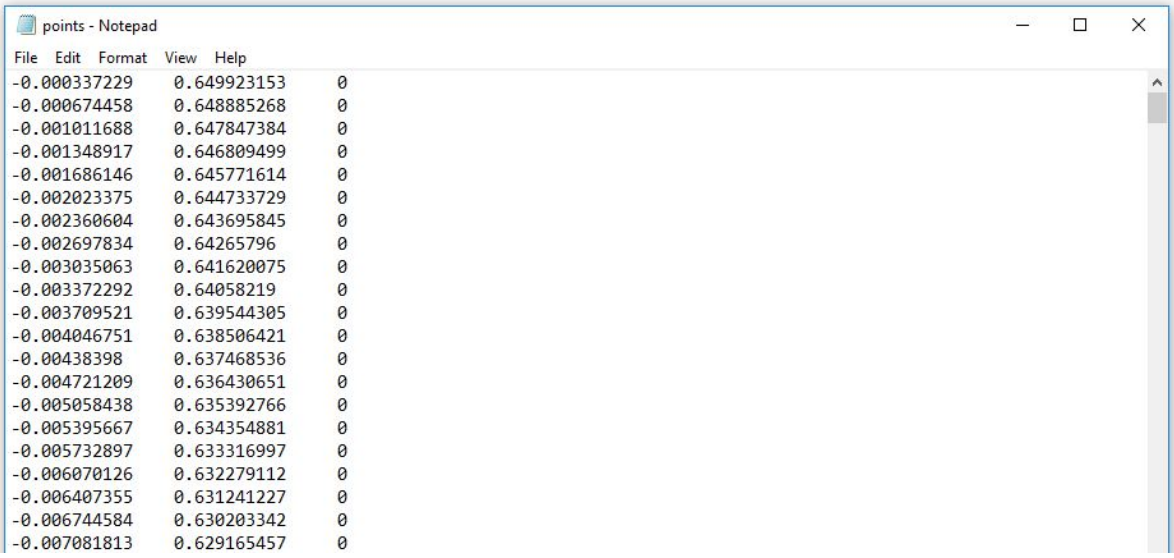

Figure 19: Data base of the points

Last step was to implement the file with the points as the path to the robot and calculate the joint angles and using end-effector draw the star shape.

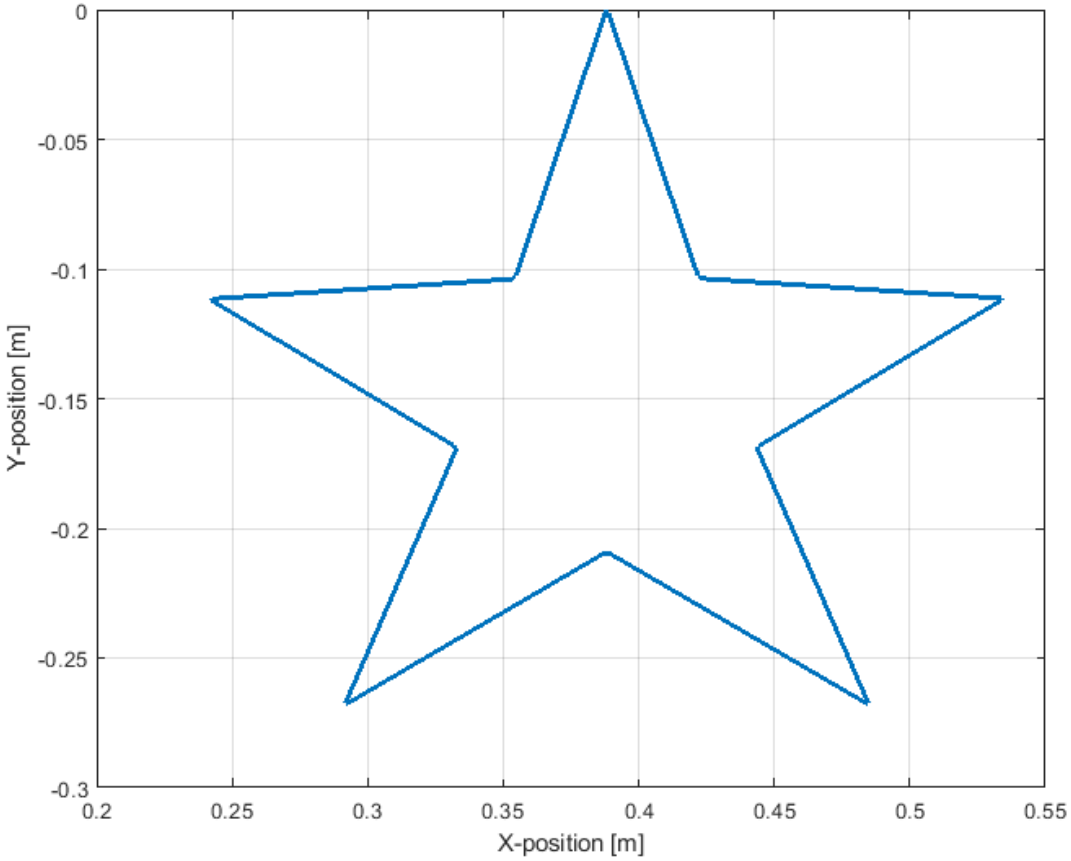

Figure 20: Star shape drawn in matlab

Star was drawn perfectly by the end-effector of the robot.