MSc Human and Biological Robotics

Author:
Edward McLaughlin

Lecturer:
Dr. Petar *Kormushev*

Robotics: Tutorial 3 - 3D Kinematics

Department of Bioengineering

May 5, 2018

1 Question 1

1.1 Parts a-c

See Matlab Code in the appendix for method.

1.2 Part d

Figures 1.1 and 1.2 show the joint angles and velocity of the end-effector respectively. These plots are for the initial condition angles given by eqn. 1 where j denotes the joint which are numbered $1-7$ from S0 to W2.

$$\theta_{j,initial} = Lowerbound_j + 0.5Range_j \tag{1}$$

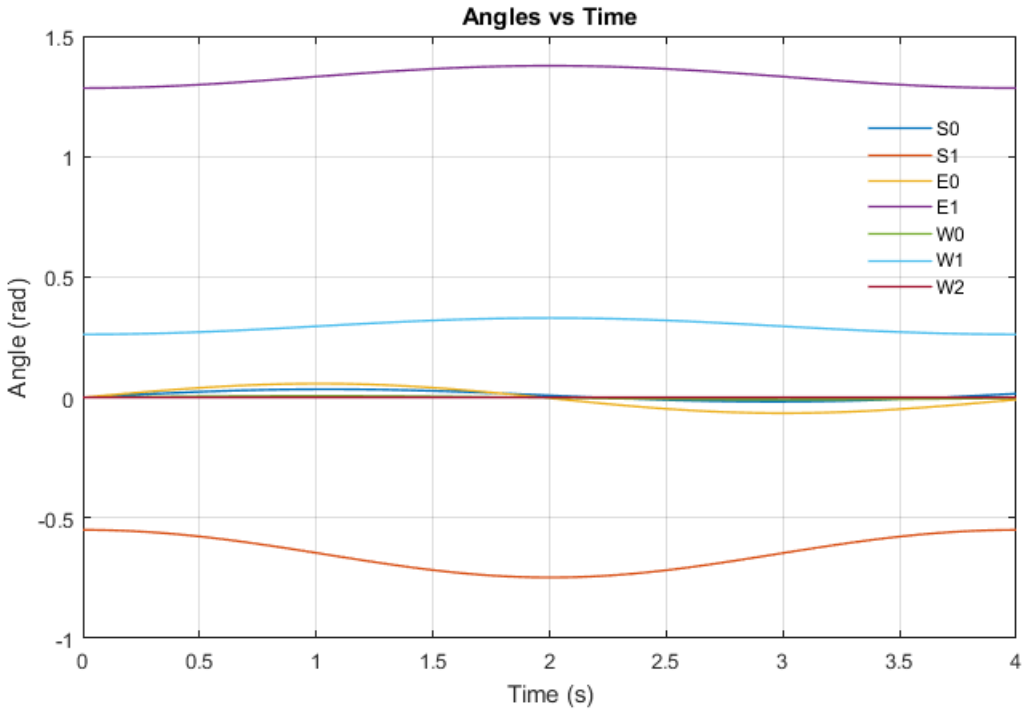


Figure 1.1: System angles over the duration of the simulation.

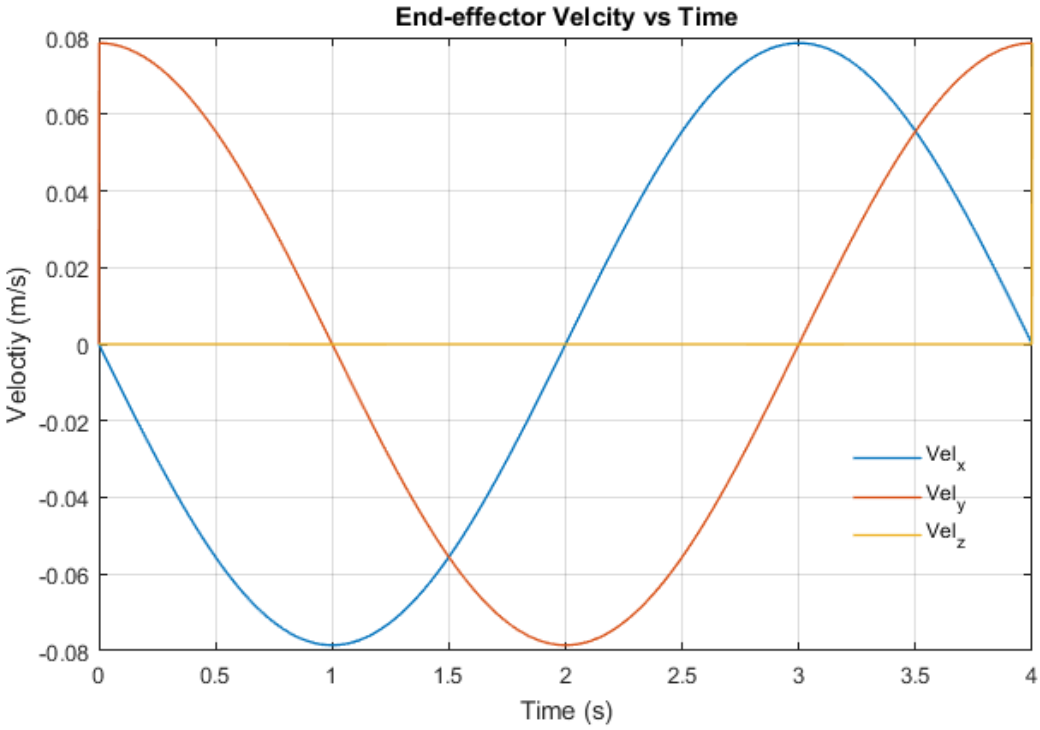


Figure 1.2: End-effector velocity in x, y and z directions over the duration of the simulation.

1.3 Part e: Changing initial conditions

The effect of changing the initial conditions was looked at by incrementing the initial angle of each joint. An example of this is shown in figure 1.3. The first 20 columns of the AnglesInit matrix are shown, the first 10 show θ_1 being incremented, the second 10 column show θ_2 being incremented. By doing this we create a matrix of 7x(7xqSweeps), where in this case, qSweeps $= 10$. In this section, the increment of the angles is governed by eqn. 2; i is incremented from 1 to qSweeps and j represents the joint angle in question.

$$\theta_{j,initial} = Lowerbound_j + \left(0.2 + \frac{i-1}{qSweeps}\right) Range_{\theta_4} \tag{2}$$

Each column is then taken and used as an initial condition. Figure 1.4 shows the results of this graphically. The graphs for each given angle demonstrate the effect of changing its initial angle. Processing the initial conditions of the angles in this manner allows the sensitivity of each of the angles to be assessed independently. The effects of changes to one angle on the end-effector position and speed as well as the system angles over the course of the simulation can be inspected. Figures 1.4 and 1.5 show the system response as the in initial conditions of θ_6 are incremented. Figures 1.6 and 1.7 show the angles of the system and end-effector speed for the 6^{th} increment in θ_6 (majenta trajectory in figure 1.5). These initial condition angles are given explicitly in figure 1.6.

```
>> anglesInit(:,1:20)

ans =

  Columns 1 through 10

   -1.1680   -0.9012   -0.6344   -0.3676   -0.1008    0.1660    0.4328    0.6996    0.9664    1.2332
   -1.6134   -1.6134   -1.6134   -1.6134   -1.6134   -1.6134   -1.6134   -1.6134   -1.6134   -1.6134
   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205
    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836
   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254
   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371
   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254

  Columns 11 through 20

   -1.1680   -1.1680   -1.1680   -1.1680   -1.1680   -1.1680   -1.1680   -1.1680   -1.1680   -1.1680
   -1.6134   -1.3466   -1.0798   -0.8130   -0.5462   -0.2794   -0.0126    0.2542    0.5210    0.7878
   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205   -2.5205
    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836    0.4836
   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254
   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371   -1.0371
   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254   -2.5254
```

Figure 1.3: Example data for first 20 columns of *AnglesInit*

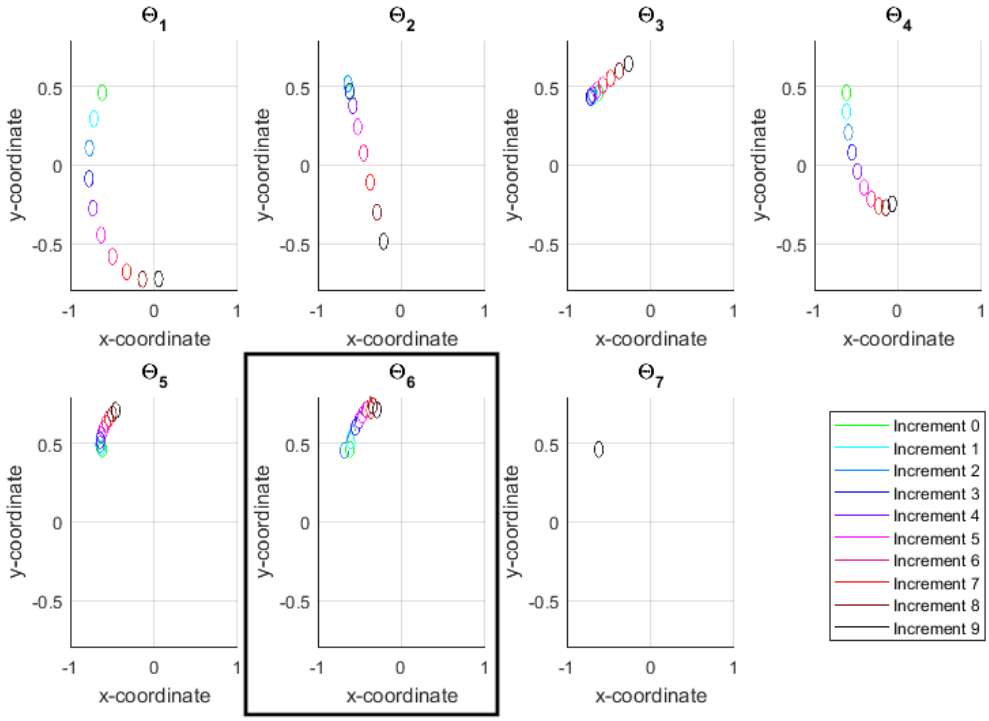


Figure 1.4: Showing the effects of changing initial conditions of system angles on the end-effector. Black boxed output shows a case where a singularity occurs at the 'wrist' of the robot.

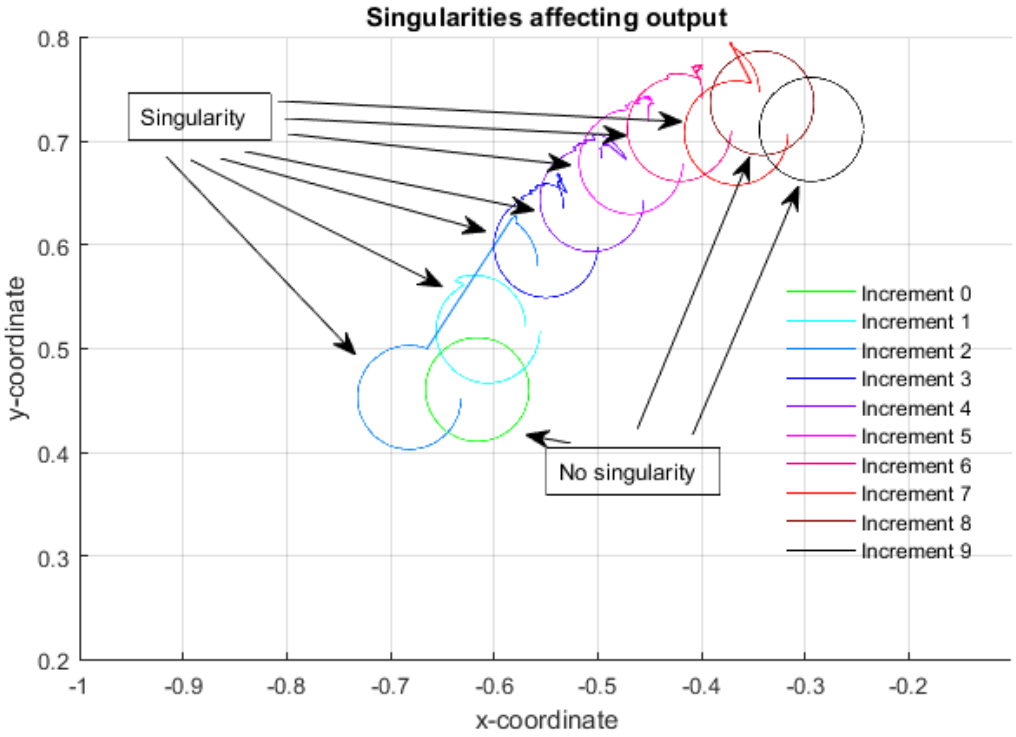


Figure 1.5: A more detailed view of the example of an apparent singularity as seen in figure 1.4.

It can be seen that the system is unable to move the end-effector in the controlled manner and that the system angles and end-effector velocity are not as desired. This is due to a singularity in the system for this configuration of initial angles. More specifically, θ_6 is such that the 'wrist' and 'forearm' align, resulting in null (or close to null) columns in the jacobian matrix. This results in the pseudo-inverse of the jacobian to have large values (greater than π for the row corresponding to the singularity in question - in this case, row 6 of the inverted jacobian is given by [7.3961, -11.2921, 4.6531]. Thus there is insufficient control of the end-effector. Relating the graphs, the first quarter of the shape made by the end-effector (majenta shape in figure 1.5) is the non-circular section. This can also be seen in the first sections of the systems angles and end-effector velocity graphs (figures refimage:AnglesSing and 1.7). As the simulated robot moves, the 'forearm' and 'wrist' become unaligned and allow for proper control of the end-effector.

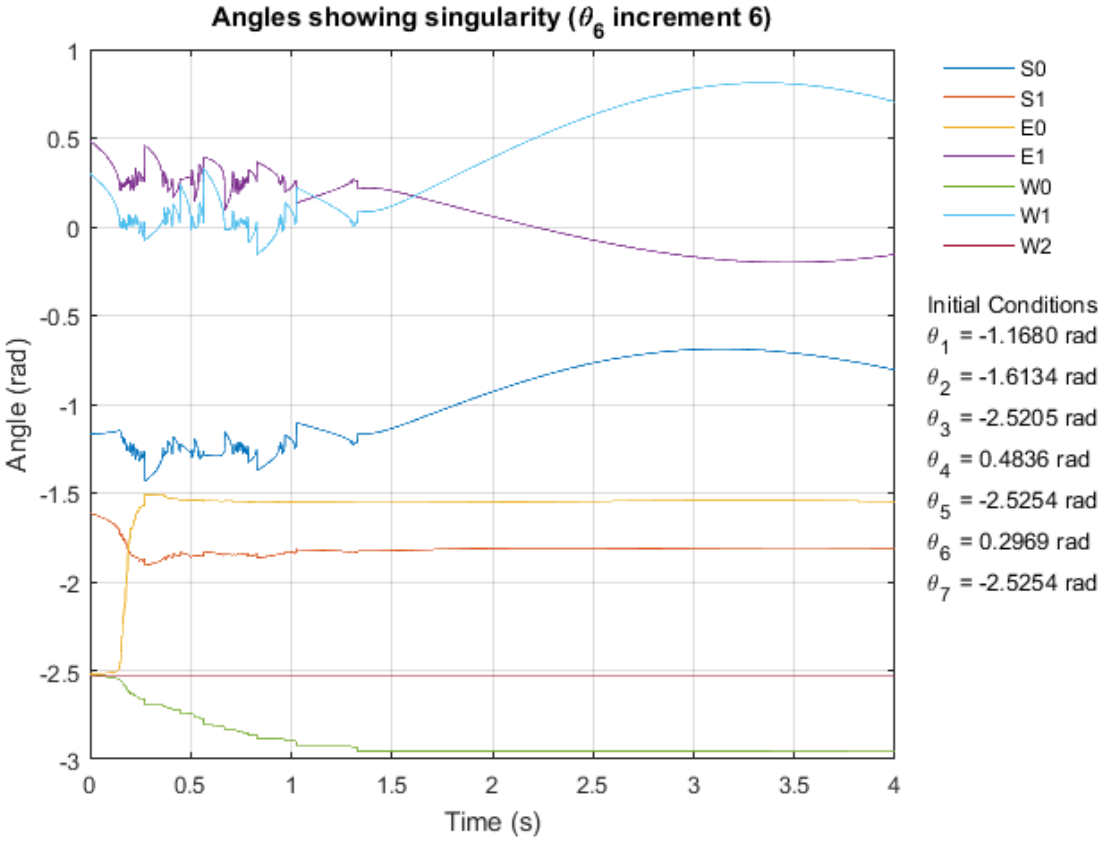


Figure 1.6: System angles over the duration of the simulation in the case of apparent singularity.

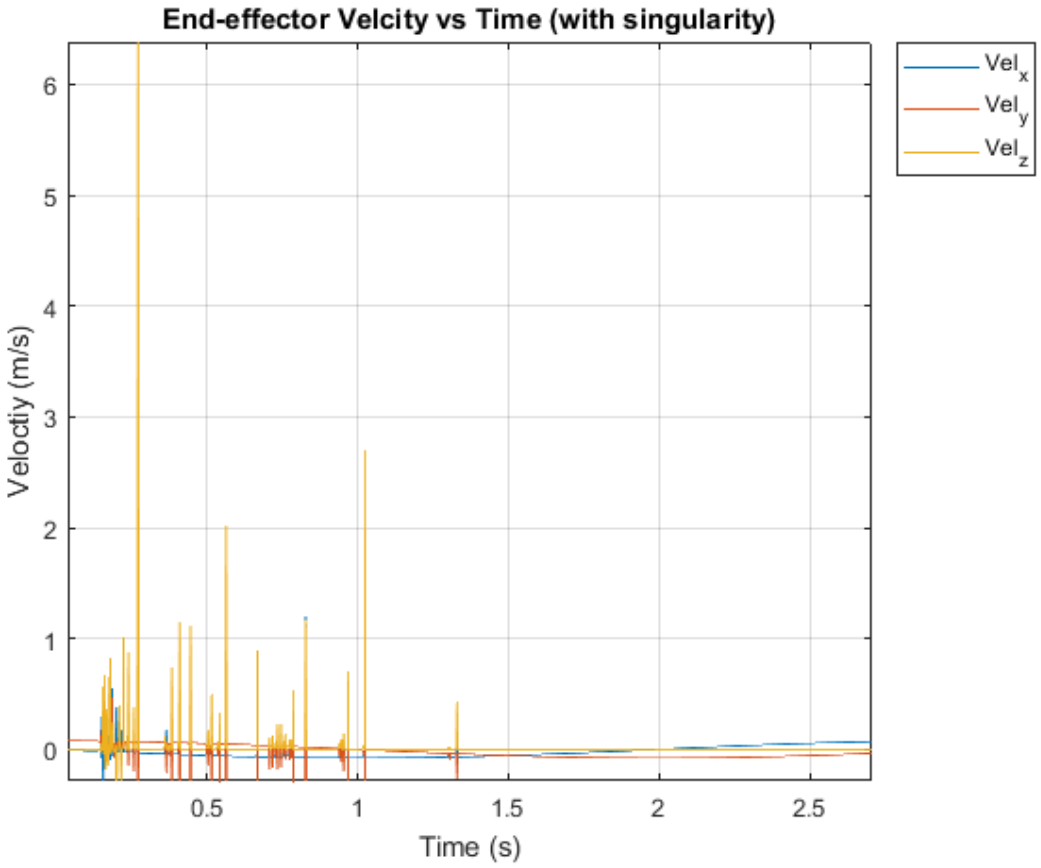


Figure 1.7: End-effector velocity over the course of the simulation in the case of apparent singularity.

2 Question 2

The 2.1 has been checked and is the same as those shown in the tutorial hand out sheet.

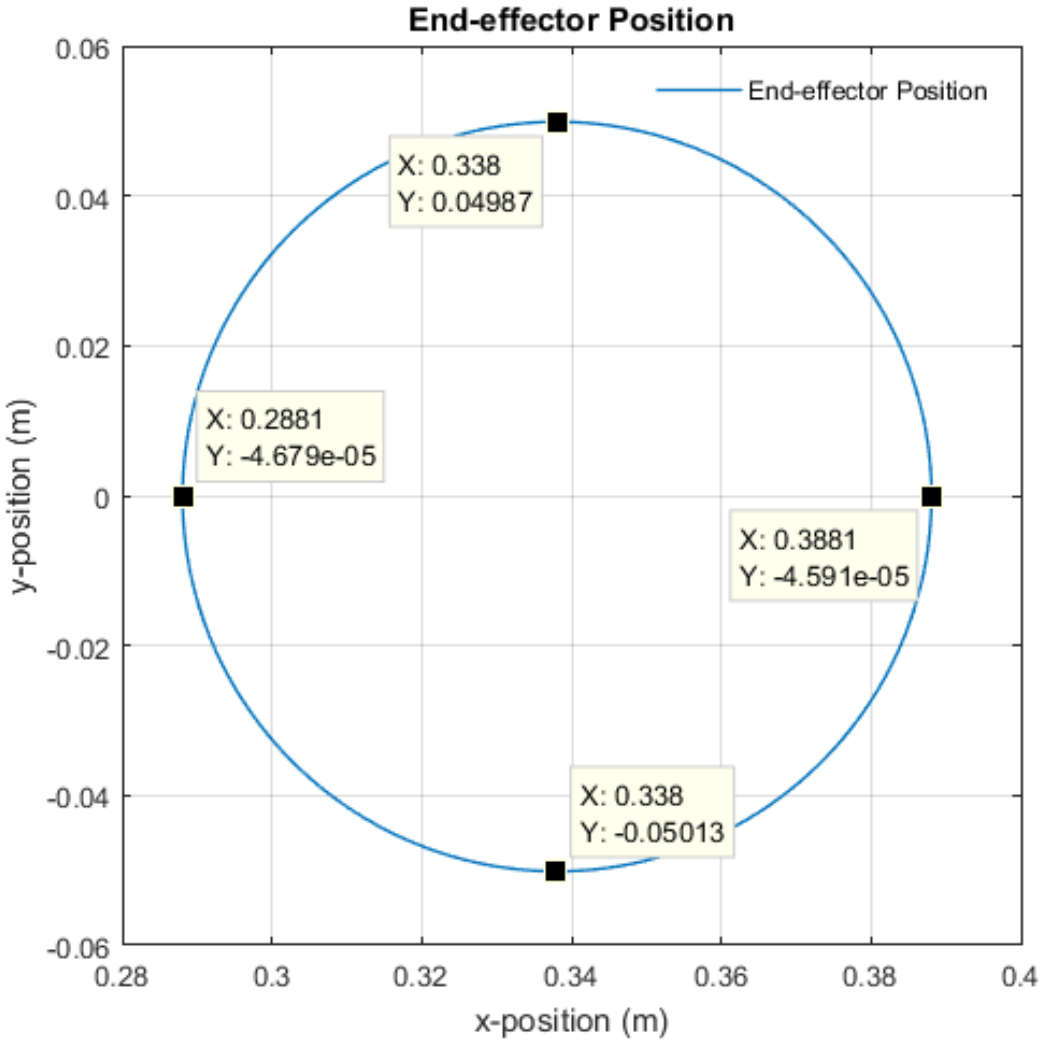


Figure 2.1: Plot of the end-effector trajectory in x- and y- directions for desired initial condition angels.

3 Question 3

The graphs shown in the answer to question 1 part d have been checked and are the same as those shown in the tutorial hand out sheet.

Figures 3.1(a) and 3.1(b) show data from a simulation where the initial conditions are swept through for each joint angle as described in the answer to question 1 part e, but where the initial angles are given by eqn. 3 to avoid singularities.

$$\theta_{j,initial} = Lowerbound_j + \left(0.5 + \frac{i-1}{qSweeps}\right) Range_{\theta_j} \tag{3}$$

Figure 3.1(a) shows the sensitivity of the position of the end-effector to changes in the initial angle of any given θ. In this case, the incrementation of all of the angles' initial conditions are ***not*** *identical*, instead the incrementation is a percentage of the given angle's total possible range. This effectively shows the workspace of the end-effector if all the other joints were locked and the robot had one degree of freedom. Figure 3.1(b) shows the euclidean distance between the centres of consecutive outputs. It can be seen that *in general*, the greater the change in the initial angle, the greater the change to the end-effector position.

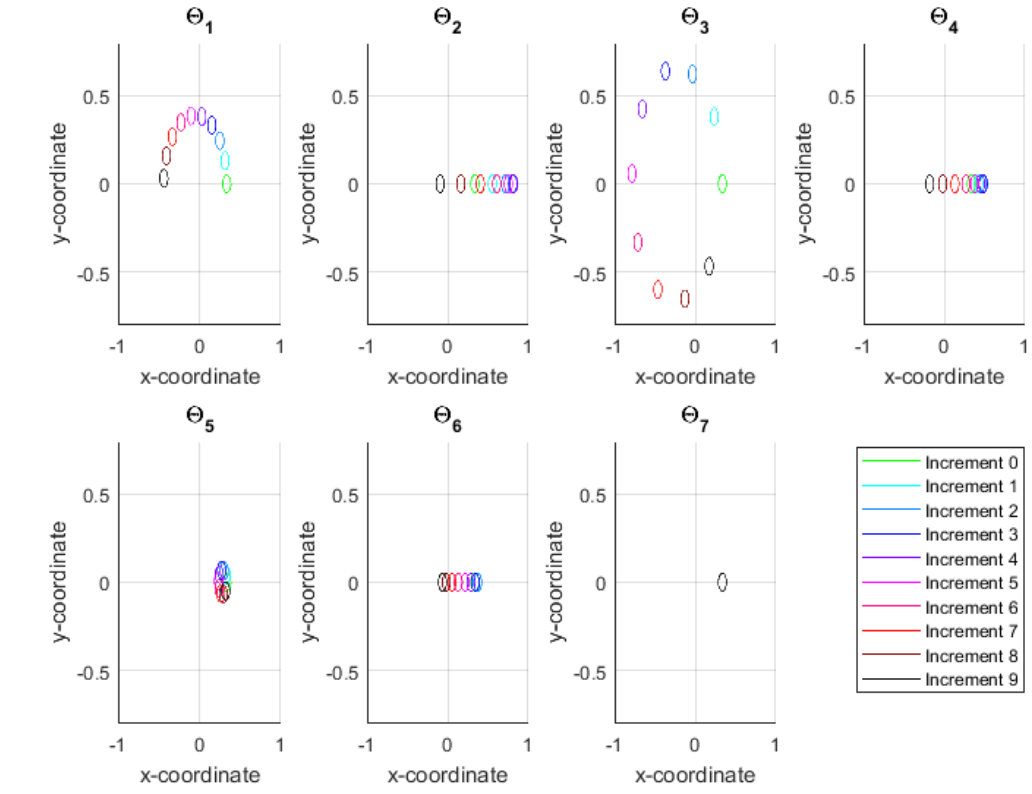
Figures 3.2(a) and 3.2(b) show the same data as in figures 3.1(a) and 3.1(b) but where the initial angles are given by eqn. 4, again, to avoid singularities.

$$\theta_{j,initial} = Lowerbound_j + \left(0.5 + \frac{i-1}{qSweeps}\right) Range_{\theta_4} \tag{4}$$

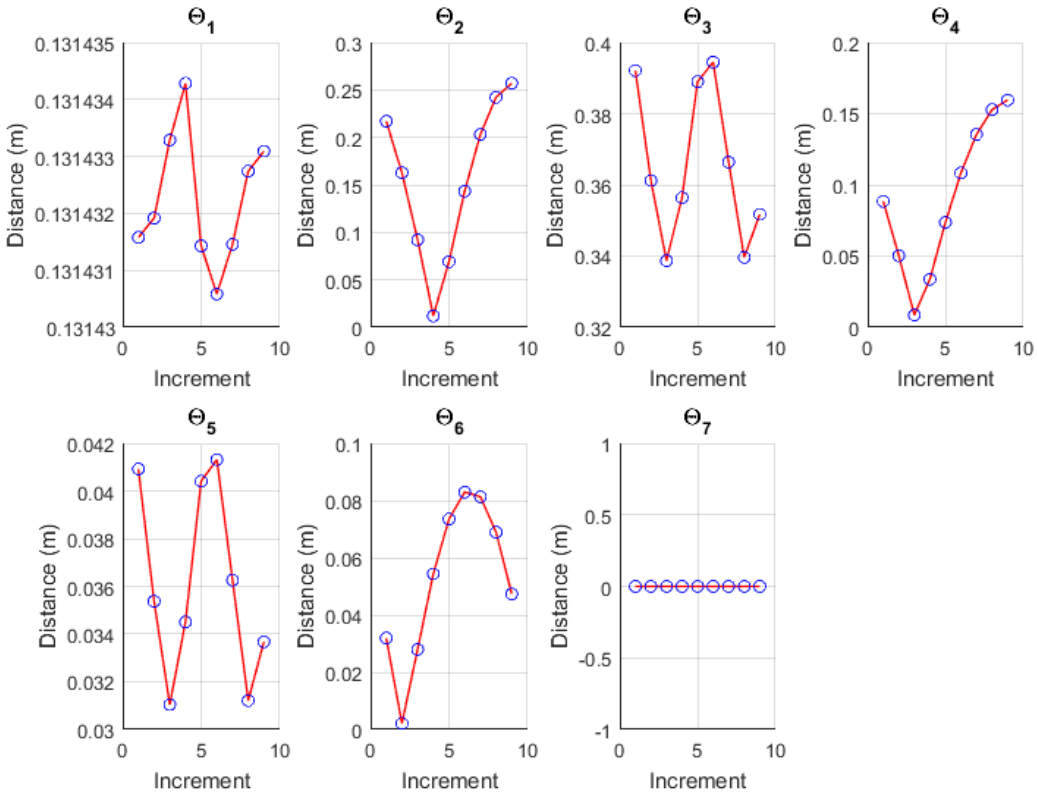
Figure 3.2(a) shows the sensitivity of the position of the end-effector to changes in the initial angle of any given θ. In this case, the incrementation of all of the angles' initial conditions is *identical*. Figure 3.2(b) shows the euclidean distance between the centre of consecutive outputs. It can be seen that *in general*, the further away from the end-effector the angle being changed is, the greater the change to the end-effector position.

From these two sets of figures 3.1 and 3.2, it is evident that sensitivity of the end-effector position is a function of both the starting angle of each joint and the distance of the joint from the end-effector i.e. dependent on subsequent link lengths and their angles. This being said, there is no effect on the end-effector position for changes in θ_7 as the axis of rotation, Z_7 is orthogonal to the axis of rotation of the previous joint, Z_6, i.e. lies on the previous joints' x-axis, X_6.

It is noted that the distance between the centres of consecutive outputs (see figures 3.1(b) and 3.2(b)) can likely be fitted with a sinusoidal curve. This is expected as the position of an objection subject to simple harmonic motion plotted against time is a sine wave.

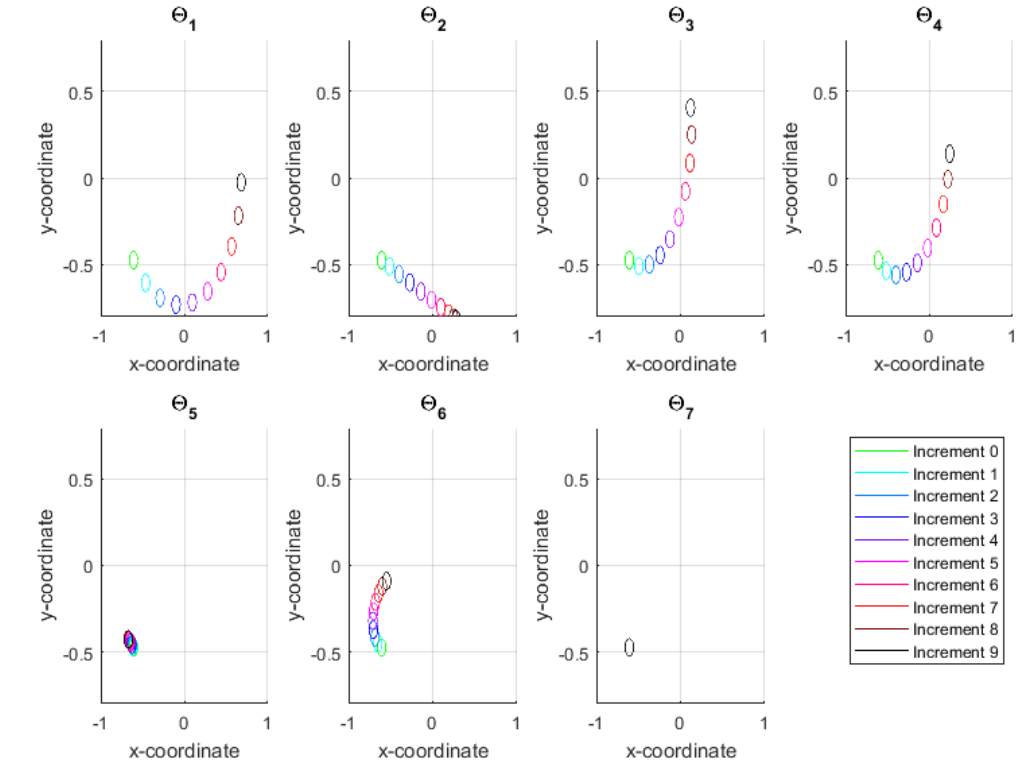


(a) Effects on end-effector trajectory.

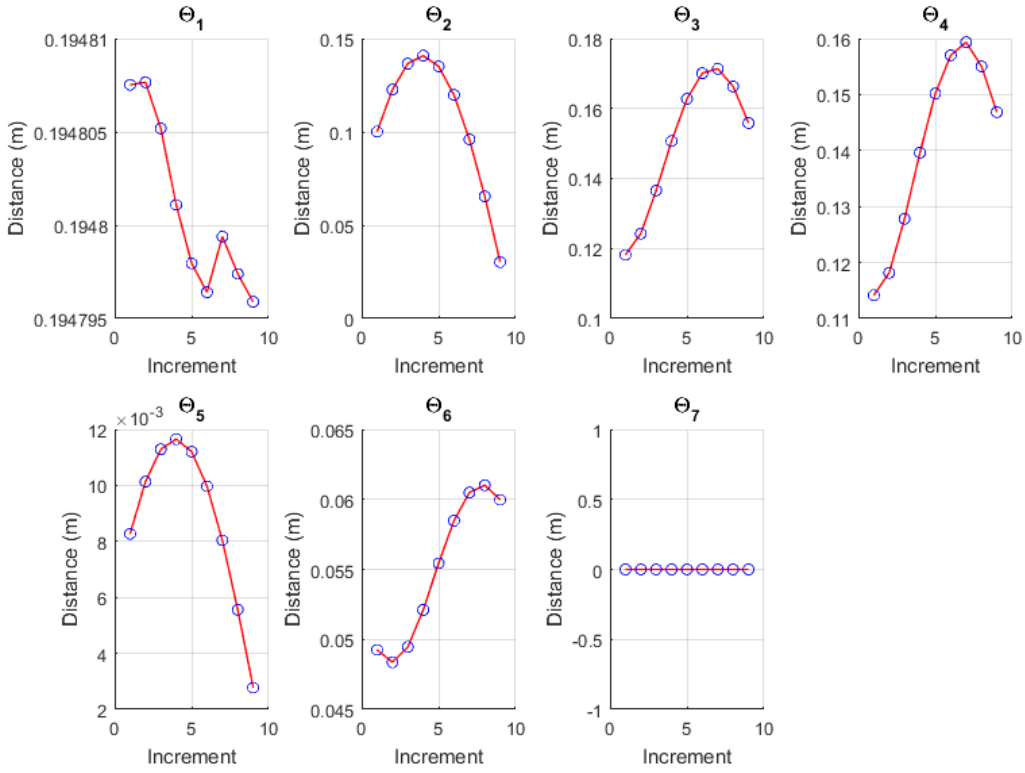


(b) Effects on the distance between the centres of consecutive end-effector trajectories.

Figure 3.1: Effects of varying each angles' initial condition by increments proportional to their full range.



(a) Effects on end-effector trajectory.



(b) Effects on the distance between the centres of consecutive end-effector trajectories.

Figure 3.2: Effects of varying each angles' initial condition by the same increment: proportional to θ_4's full range.

Appendix

MatLab code

The functions 'updatePos˙ee' and 'updatejacobian' are left out here as they are extremely long formulas. Available upon request!

```
1
2  clear all
3  clc
4  close all
5
6  tic
7
8  %Set-up time properties
9  T=4;
10 t=0.002;
11 time=0:t:(T);
12
13 %Calculate position and velocity profiles
14 x_velocity = -0.05*0.5*pi*sin(0.5*pi*time);
15 y_velocity = 0.05*0.5*pi*cos(0.5*pi*time);
16 velocity_desired = zeros(3,(T/t+1));
17 velocity_desired(1,:) = x_velocity(1,:);
18 velocity_desired(2,:) = y_velocity(1,:);
19
20 %Initialise angle and velocitt matricies
21 qSweeps = 10;
22 angles = zeros(7,(T/t+1),(7*qSweeps));
23 angles_dot = zeros(7,(T/t+1),(7*qSweeps));
24 pos_ee = zeros(3,(T/t+1),(7*qSweeps));
25 vel_ee = zeros(3,(T/t+1),(7*qSweeps));
26 anglesInit = zeros(7,(7*qSweeps));
27
28 %Set-up predefined angle ranges
29 angleRanges = [-1.7016, 1.7016;
30                -2.147,  1.047;
31                -3.0541, 3.0541;
32                -0.05,   2.618;
33                -3.059,  3.059;
34                -1.5707, 2.094;
35                -3.059,  3.059];
36
```

```
%Populate the anglesStart matrix with the varied initial conditions: pick
%one depending on desired changing on initial conditions
for i=1:1:qSweeps
%       sweeping the angles form lower bound + (0.5 + steps)* total ranges
 anglesStart(:,i) = angleRanges(:,1) + (0.5+1/qSweeps*(i-1))*( angleRanges(:,2) - angleRanges(:,1));
%       sweeping the angles form lower bound + (0.5 + steps)* smallest range
%       smallest (theta 4)
% anglesStart(:,i) = angleRanges(:,1) + (0.5+1/qSweeps*(i-1)) *(angleRanges(4,2) - angleRanges(4,1));
%       sweeping angles from lower bound + (0.2 + steps)* smallest range
% anglesStart(:,i) = angleRanges(:,1) + (0.2+1/qSweeps*(i-1)) *(angleRanges(4,2) - angleRanges(4,1));
end

%Define the angles as real symbols, and set-up initial angles vector
syms theta1 theta2 theta3 theta4 theta5 theta6 theta7 real
initialAngles = [theta1;
                 theta2;
                 theta3;
                 theta4;
                 theta5;
                 theta6;
                 theta7];

%Set-up predefined DH Table
DHTable = [0,              0, 0.2703, initialAngles(1,1);...
           0.069,     -1.571,      0, initialAngles(2,1);...
           0,          1.571, 0.3644, initialAngles(3,1);...
           0.069,     -1.571,      0, initialAngles(4,1);...
           0,          1.571, 0.3743, initialAngles(5,1);...
           0.01,      -1.571,      0,initialAngles(6,1);...
           0,          1.571, 0.2295, initialAngles(7,1)];

%Calculate the individual transformation matrices for the joints
s0_T_s1 = transformationMatrix(DHTable, 1);
```

```
70 s1_T_e0 = transformationMatrix(DHTable, 2);
71 e0_T_e1 = transformationMatrix(DHTable, 3);
72 e1_T_w0 = transformationMatrix(DHTable, 4);
73 w0_T_w1 = transformationMatrix(DHTable, 5);
74 w1_T_w2 = transformationMatrix(DHTable, 6);
75 w2_T_ee = transformationMatrix(DHTable, 7);
76
77 % Calculate the total transformation matrix from S0 to end-
     effector and pull out the position transformations into a
     position matrix
78 s0_T_ee = s0_T_s1*s1_T_e0*e0_T_e1*e1_T_w0*w0_T_w1*w1_T_w2*
     w2_T_ee;
79 P_s0_T_ee = s0_T_ee(1:3,4);
80
81 % Calculate the S0 to end-effector jacobian
82 s0eeJacobian = jacobian(P_s0_T_ee, [theta1, theta2, theta3,
     theta4, theta5, theta6, theta7]);
83
84 % Create a matrix 7x(7*qSweep) which maintain the initial
     angles of LowerAngle + 0.5*Range while varying each angle (
     theta1, theta2, etc..) in turn as defined by anglesStart (or
      otherwise depending on the calculation chosen in lines
     39-47).
85 n = 1;
86 for j=1:1:7
87     k = 1;
88     while k<=qSweeps
89         anglesInit(:,n) = anglesStart(:,1);
90         anglesInit(j,n) = anglesStart(j,k);
91         k = k + 1;
92         n = n + 1;
93     end
94 end
95
96 % Main loop calculating angles, updating position and jacobian
     on each
97 % iteration.
98 for j=1:1:(7*qSweeps)
99     angles(:,1,j) = anglesInit(:,j);
100    pos_ee(:,1,j) = updatePos_ee(angles(:,1,j));
101    for i=2:1:(T/t+1)
```

```
102             angles(:,i,j) = angles(:,i-1,j) + angles_dot(:,i-1,j)
                    * t;
103             pos_ee(:,i,j) = updatePos_ee(angles(:,i,j));
104             vel_ee(:,i-1,j) = (pos_ee(:,i,j) - pos_ee(:,i-1,j))/t;
105             jacobian = updatejacobian(angles(:,i,j));
106             InvJacobian = pinv(jacobian);
107             angles_dot(:,i,j) = InvJacobian*velocity_desired(:,i);
108         end
109         vel_ee(:,end,j) = y_velocity(:,end);
110     end
111
112             function [aTb] = transformationMatrix(DHTable, i)
113
114             aTb = [ cos(DHTable(i,4)),-sin(DHTable(i,4)),0, DHTable
                    (i,1) ;...
115                     sin(DHTable(i,4)) * cos(DHTable(i,2)), cos(
                        DHTable(i,4)) * cos(DHTable(i,2)),-sin(DHTable
                        (i,2)),-sin(DHTable(i,2)) * DHTable(i,3);...
116                     sin(DHTable(i,4))*sin(DHTable(i,2)), cos(DHTable
                        (i,4)) * sin(DHTable(i,2)),cos(DHTable(i,2)),
                        cos(DHTable(i,2)) * DHTable(i,3) ;...
117                     0,          0,          0,          1];
118
119             end
```