Robotics Project Part2

0560011
鄭宇彤

◎Introduce the Interface

Open the program of path_planning.m and run the program with F5 or hit Run(Fig.1). The program will run with two path planning method and show the result(Fig.2).

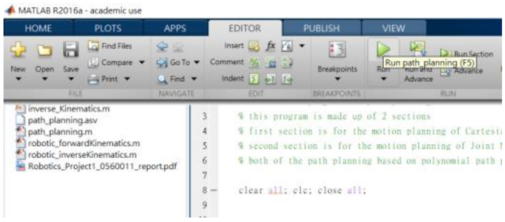


Fig.1

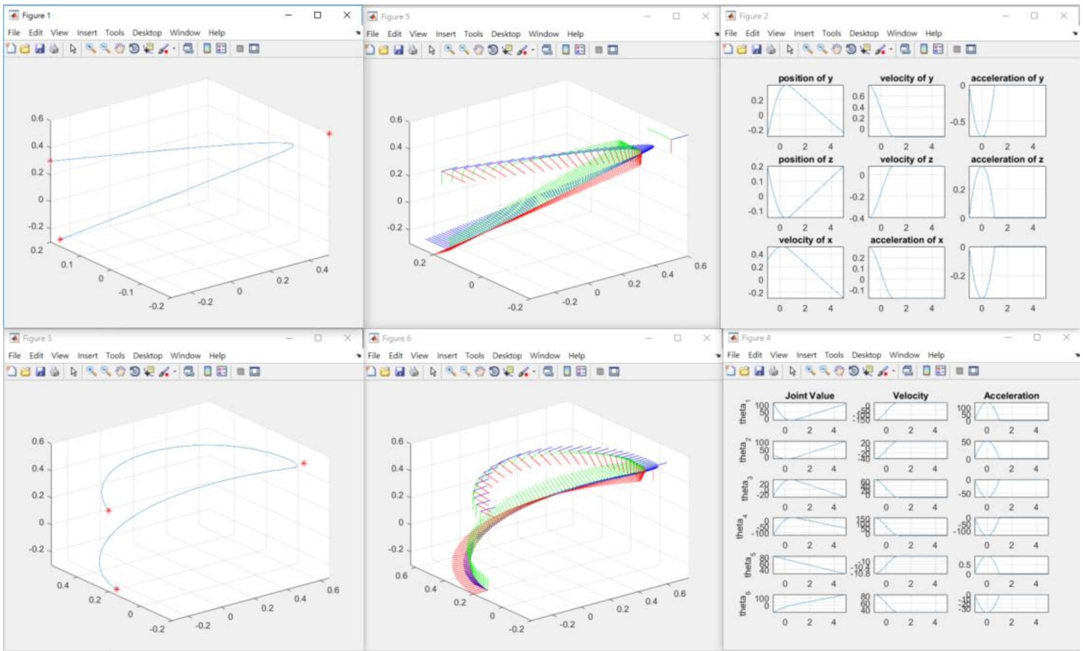
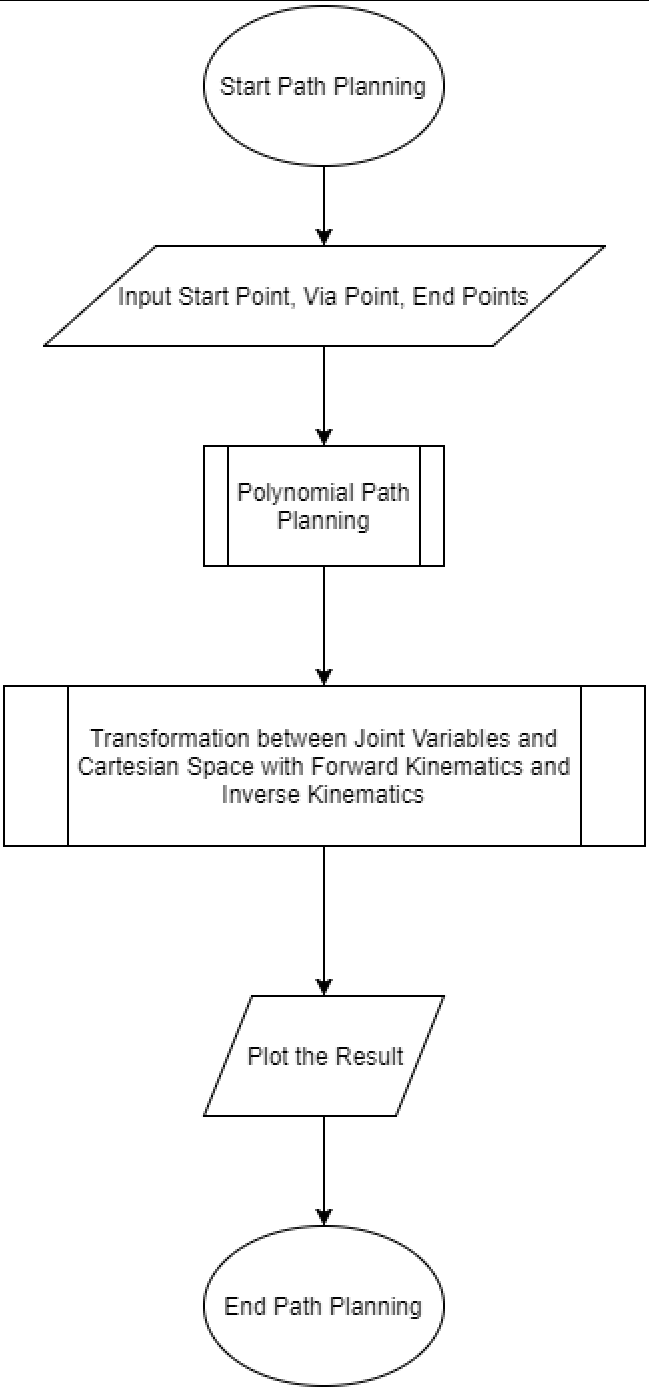


Fig.2

◎Program Structure

Flow Chart



Source Code

```
% This is the program for path planning of PUMA560
% This program is made up of 2 sections :
% first section is for the path planning of
Cartesian Move
% second section is for the path planning of Joint
```

```
Move
% both of the path planning are based on polynomial
path planning
% last modified by YuTung, Cheng
% last modified Jun 14th,2018


clear all; clc; close all;


%%  Section 1 : Path Planning based on Cartesian
Coordinate

% =========== set up parameters ===========%
% set A, B, C points for the trajectory
% A is the start point, B is the via point, and C
is the end point of this
% path
A = [0 0 1 -0.30; -1 0 0 0.20; 0 -1 0 0.30; 0 0 0
1]; %unit : m
B = [0 0 1 0.50; 0 1 0 -0.20; -1 0 0 0.60; 0 0 0
1]; %unit : m
C = [-1 0 0 -0.25; 0 0 1 0.20; 0 1 0 -0.30; 0 0 0
1]; %unit : m

% changed the position into Cartesian space
position_A = DHconvert2cartesian(A);
position_B = DHconvert2cartesian(B);
position_C = DHconvert2cartesian(C);

% ===============calculation of the path
planning=============%
% Polynomial Path Planning
tacc = 1;
T = 5*tacc; % If the T is more times of tacc, it
makes the trajectory sharper
t = [-tacc:0.001:T];
q = zeros(4,4,length(t)); % trajectory for the
```

```
planned path
q_dot = zeros(4,4,length(t)); % velocity
q_dou_dot = zeros(4,4,length(t)); % acceleration

del_C = C-B;
del_B = A-B;

for i = 1:1:length(t)

    % two time segments is considered
    % first segments is for the time with
acceleration
    % second segments is the linear part to reach
end point after the
    % acceleration time

    if t(i)<=tacc
        % the time between -tacc to +tacc
        h = (t(1,i)+tacc)/2*tacc;
        q(:,:,i) = [ (del_C*(tacc/T)+del_B)*(2-h)*h^2
- 2*del_B ]*h + B + del_B;
        q_dot(:,:,i) = [ (del_C*(tacc/T)+del_B)*(1.5-
h)*2*h^2 - del_B ]*(1/tacc);
        q_dou_dot(:,:,i) = [ ( del_C*(tacc/T) +
del_B )*( 1-h ) ]*(3*h/tacc^2);
    else
        % the time between -tacc to +T
        q(:,:,i) = del_C*t(1,i)/T+B;
        q_dot(:,:,i) = del_C/T;
        q_dou_dot(:,:,i) = 0;
    end
end

%  ============Plotting==================  %
% first figure, plot the trajectory, velocity,
acceleration of x, y, z axis

x = zeros(1,length(t));
```

```
y = zeros(1,length(t));
z = zeros(1,length(t));
x_dot = zeros(1,length(t));
y_dot = zeros(1,length(t));
z_dot = zeros(1,length(t));
x_dou_dot = zeros(1,length(t));
y_dou_dot = zeros(1,length(t));
z_dou_dot = zeros(1,length(t));

for i = 1:1:length(t)
    position = DHconvert2cartesian(q(:,:,i));
    x(1,i) = position(1,1);
    y(1,i) = position(1,2);
    z(1,i) = position(1,3);

    velocity = DHconvert2cartesian(q_dot(:,:,i));
    x_dot(1,i) = velocity(1,1);
    y_dot(1,i) = velocity(1,2);
    z_dot(1,i) = velocity(1,3);

    acc = DHconvert2cartesian(q_dou_dot(:,:,i));
    x_dou_dot(1,i) = acc(1,1);
    y_dou_dot(1,i) = acc(1,2);
    z_dou_dot(1,i) = acc(1,3);
end

figure(1);
title('3D path of Cartesian Move');
plot3(x,y,z)
axis tight;
grid on;
hold on;
scatter3(position_A(1,1),position_A(1,2),position_A
(1,3),'r*')
hold on;
scatter3(position_B(1,1),position_B(1,2),position_B
(1,3),'r*')
hold on;
```

```
scatter3(position_C(1,1),position_C(1,2),position_C
(1,3),'r*')
hold off;

% second figure, plot the trajectory, velocity,
acceleration of x, y, z
% axis separately

figure(2);
title('position of x');
subplot(3,3,1),plot(t(1,:),x(1,:))
axis tight;
grid on;

title('position of y');
subplot(3,3,4),plot(t(1,:),y(1,:))
axis tight;
grid on;

title('position of z');
subplot(3,3,7),plot(t(1,:),z(1,:))
axis tight;
grid on;

title('velocity of x');
subplot(3,3,2),plot(t(1,:),x_dot(1,:))
axis tight;
grid on;

title('velocity of y');
subplot(3,3,5),plot(t(1,:),y_dot(1,:))
axis tight;
grid on;

title('velocity of z');
subplot(3,3,8),plot(t(1,:),z_dot(1,:))
axis tight;
grid on;
```

```
title('acceleration of x');
subplot(3,3,3),plot(t(1,:),x_dou_dot(1,:))
axis tight;
grid on;

title('acceleration of y');
subplot(3,3,6),plot(t(1,:),y_dou_dot(1,:))
axis tight;
grid on;

title('acceleration of z');
subplot(3,3,9),plot(t(1,:),z_dou_dot(1,:))
axis tight;
grid on;

figure(5)
title('3D path of Cartesian Move')
plot_euler(A);hold on;
plot_euler(B);hold on;
plot_euler(C);hold on;
for i = 1:1:length(t)/50
    plot_euler(q(:,:,50*i));hold on;
end
axis tight;
grid on;
hold off;


%% Section 2 : Path Planning based on Joint
Variables

% =========== set up parameters ===========%
% set A, B, C points for the trajectory
% A is the start point, B is the via point, and C
is the end point of this
% path
```

```
A = [0 0 1 -0.30; -1 0 0 0.20; 0 -1 0 0.30; 0 0 0
1]; %unit : m
B = [0 0 1 0.50; 0 1 0 -0.20; -1 0 0 0.60; 0 0 0
1]; %unit : m
C = [-1 0 0 -0.25; 0 0 1 0.20; 0 1 0 -0.30; 0 0 0
1]; %unit : m

% find joint variables of each points

j_A = inverse_Kinematics(A);
j_B = inverse_Kinematics(B);
j_C = inverse_Kinematics(C);

% ===============calculation of the path
planning=============%
% polynomial path planning
j_tacc = 1;
j_T = 5*tacc;
j_t = [-tacc:0.001:T];
j_q = zeros(6,length(t)); % 6xt vector for 6 join
variables
j_q_dot = zeros(6,length(t)); % 6xt vector for 6
joint variables
j_q_dou_dot = zeros(6,length(t)); %6xt vector for 6
joint variables

j_del_C = (j_C-j_B).'; % 6x1 vector
j_del_B = (j_A-j_B).'; % 6x1 vector

for i = 1:1:length(t)

    % two time segments is considered
    % first segments is for the time with
acceleration
    % second segments is the linear part to reach
end point after the
    % acceleration time
```

```
    if j_t(i)<=j_tacc
        % the time between -tacc to +tacc
        h = (j_t(1,i)+j_tacc)/2*j_tacc; % time grid
        j_q(:,i) =
[ (j_del_C*(j_tacc/j_T)+j_del_B)*(2-h)*h^2 -
2*j_del_B ]*h + j_B.' + j_del_B;
        j_q_dot(:,i) =
[ (j_del_C*(j_tacc/j_T)+j_del_B)*(1.5-h)*2*h^2 -
j_del_B ]*(1/j_tacc);
        j_q_dou_dot(:,i) = [ ( j_del_C*(j_tacc/j_T) +
j_del_B )*( 1-h ) ]*(3*h/j_tacc^2);
    else
        % the time between -tacc to +T
        j_q(:,i) = j_del_C*t(1,i)/j_T+ j_B.';
        j_q_dot(:,i) = j_del_C/j_T;
        j_q_dou_dot(:,i) = [0 0 0 0 0 0].';
    end
end

%  ============Plotting=================  %
% use forward kinematics to turn joint variables
into cartesian space
j_q_cartesian = zeros(6,length(j_t));
for i = 1:1:length(j_t)
    j_q_cartesian(:,i) =
foward_Kinematics(j_q(:,i)*(pi/180)).';
end

% figure 3 is for the 3D path of Joint Move
% this plot will first use the function of forward
Kinematics to obtain the
% Cartesian space location of end effector of the
robot manipulator; and it
% will plot the Cartesian space of each point after
that

figure(3);
title('3D path of Joint Move');
```

```
plot3(j_q_cartesian(1,:),j_q_cartesian(2,:),j_q_car
tesian(3,:));
axis tight;
grid on;
hold on;
A_plot = DHconvert2cartesian(A);
B_plot = DHconvert2cartesian(B);
C_plot = DHconvert2cartesian(C);
scatter3(A_plot(1,1),A_plot(1,2),A_plot(1,3),'r*');
hold on;
scatter3(B_plot(1,1),B_plot(1,2),B_plot(1,3),'r*');
hold on;
scatter3(C_plot(1,1),C_plot(1,2),C_plot(1,3),'r*');
hold off;


% figure 4 is the angular position, angular
velocity, and angular
% acceleration of each joint
figure(4);

for i = 1:1:6
    subplot(6,3,(i-1)*3+1),plot(j_t(1,:),j_q(i,:))
    axis tight;
    grid on;
    if i == 1
        title('Joint Value');
    end
    ylabel(['theta_' num2str(i)]);
end

for i = 1:1:6
    subplot(6,3,(i-
1)*3+2),plot(j_t(1,:),j_q_dot(i,:))
    axis tight;
    grid on;

    if i == 1
```

```
        title('Velocity');
    end
end

for i = 1:1:6
    subplot(6,3,(i-
1)*3+3),plot(j_t(1,:),j_q_dou_dot(i,:))
    axis tight;
    grid on;
    if i == 1
        title('Acceleration');
    end
end


figure(6)
title('3D path of Joint Move')
plot_euler(A);hold on;
plot_euler(B);hold on;
plot_euler(C);hold on;
for i = 1:1:length(j_t)/50

plot_euler( forward_Kinematics_T( j_q(1:6,50*i)*(pi
/180) ));hold on;
end
axis tight;
grid on;
hold off;
```

◎Math Calculation

Polynomial Path Planning is applied in this project. For a 3-dimensional space, 6 degree of freedom is needed for 3-dimension of location and 3-dimension of velocity for each direction. As a result, 5 order polynomial is required. However, it's known that an order can be reduced when the acceleration profile of the planned path is symmetrical. In conclusion, a 4^{th} order polynomial will be applied to the path planning for this project.

Path planning can be seen as a point-to-point path planning for the points in a smooth route. In the following, the path planning will be simplified as a route with 3 points. For a more complex route, it can be break down with sets of each three points and solved as following methods, too.

The 4th order polynomial path planning will be shown as Eq.1.

$$\begin{cases} q(t) = a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \\ \dot{q}(t) = 4a_4t^3 + 3a_3t^2 + 2a_2t + a_1 \quad for - t_{acc} \le t \le t_{acc} \\ \ddot{q}(t) = 12a_4t^2 + 6a_3t + 2a_2 \end{cases} \quad \text{Eq.1}$$

For path planning problem, at least 3 points should be known : the Start Point, Via Point, and End Points. The relations of these points and time are shown with Fig.3.

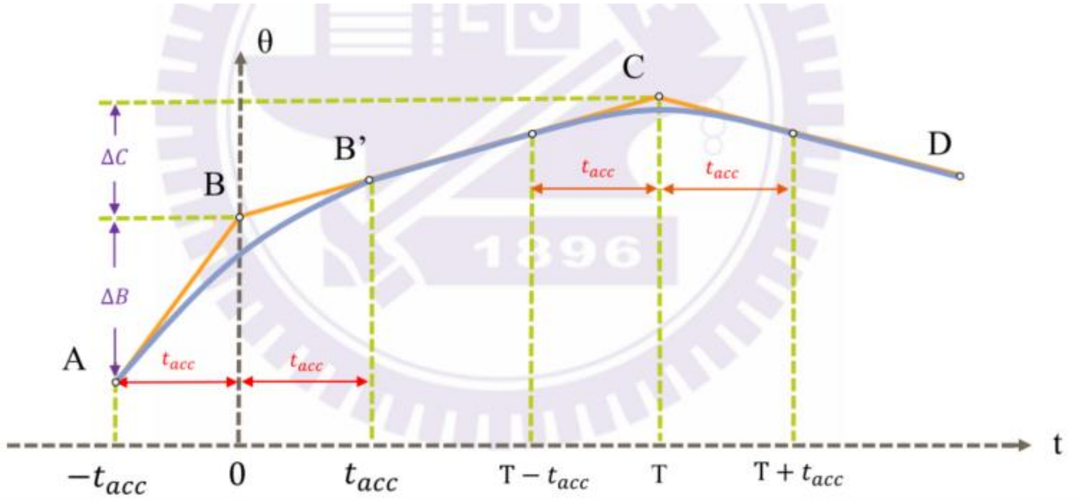


Fig.3

Now point A is defined as start point, point B is defined as via points, and point C is defined as end point.

Applied A, B, C points to the 4th order polynomial and substitute time factor with tacc for the acceleration time for this robot manipulator, Eq.4 is obtained.

Let $\begin{cases} \Delta C = C - B \\ \Delta B = A - B \end{cases}$

$$\begin{cases} q(h) = \left[\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right)(2-h)h^2 - 2\Delta B\right]h + B + \Delta B \\ \dot{q}(h) = \left[\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right)(1.5-h)2h^2 - \Delta B\right]\frac{1}{t_{acc}} \\ \ddot{q}(h) = \left[\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right)(1-h)\right]\frac{3h}{t_{acc}^2} \end{cases} \quad \text{Eq.4}$$

$$where \;\; h = \frac{t + t_{acc}}{2t_{acc}} for - t_{acc} \le t \le t_{acc}$$

Moreover, here will be a segment of linear portion of planned path. The linear portion will be planned as Eq.5.

$$\begin{cases} q(h) = \Delta C h + B \\ \dot{q}(h) = \frac{\Delta C}{T} \\ \ddot{q}(h) = 0 \end{cases} \qquad \text{Eq.5}$$

$$where \;\; h = \frac{t}{T} for \;\; t_{acc} \leq t \leq T - t_{acc}$$

Notice that the path may not pass via point. The route can be adjust by apply more points known as pseudo points if needed. Shown as Fig.4.

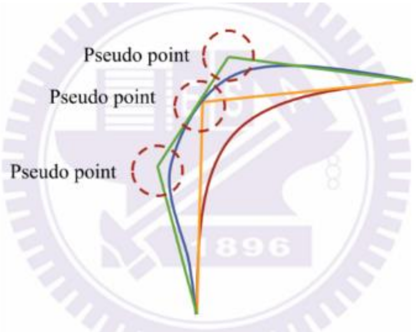
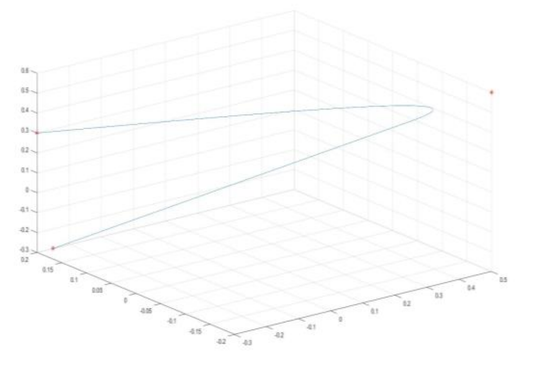
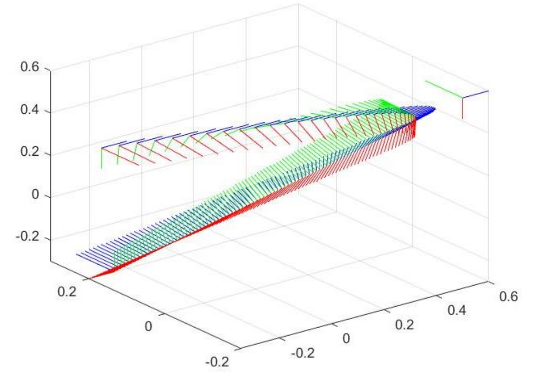
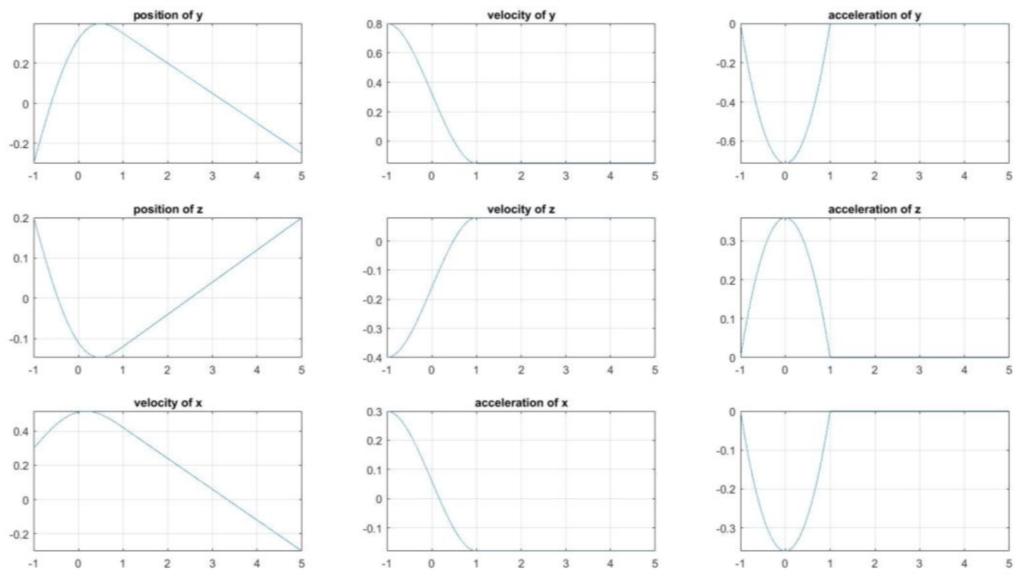
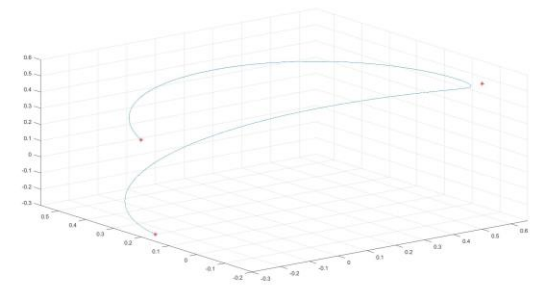
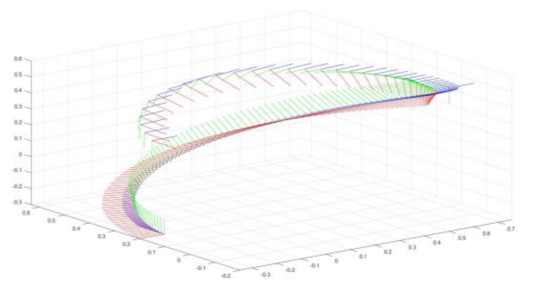
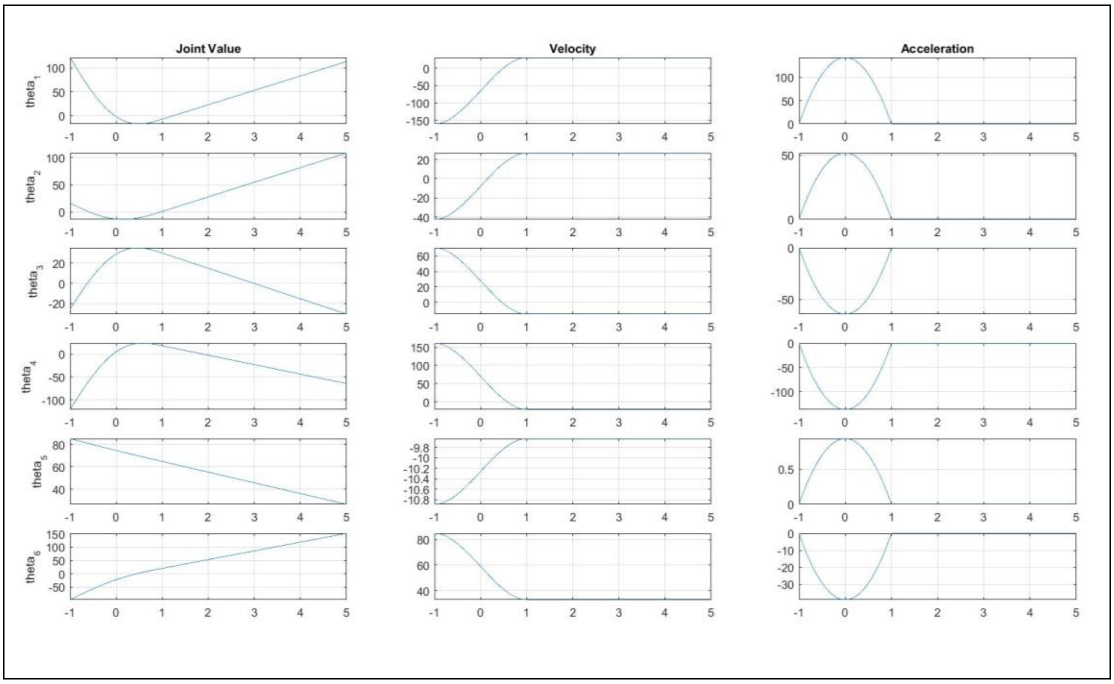


Fig.4

◎Result

<table>
<tr><td>3D Path Planning of Cartesian Motion</td><td>3D Path Planning of Cartesian Motion with Directions</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Position, Velocity, Acceleration of the End Effector</td></tr>
<tr><td colspan="2">
</td></tr>
<tr><td>3D Path of Joint Motion</td><td>3D Path of Joint Motion with Directions</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Angle, Velocity, Acceleration of 6 joint angles</td></tr>
</table>



◎Comparing the Two Path Planning

Path Planning in Cartesian space can be modified simply when asking to adjust the route. However, it may results in some points which are singular to the robot manipulator. It's highly recommended to check if the joint variables of each point is in available range of the robot manipulator. Path Planning in Joint Space avoid the conditions of singularity; however, it's more difficult to adjust the route when the route is asked to change. The Path is harder to design for shorten the path in Cartesian space since a shorter path in joint space doesn't mean a shorter space in Cartesian space. Moreover, Path Planning in Joint Space acquires both forward and inverse kinematics for the robot manipulator. As a result, complex calculation is required. Also, the calculation of forward and inverse kinematics may be effected by the non-linearity of trigonometric functions involved.