# 312605015 詹恆瑜 機器人學 Project2

# (一)介面說明

我使用 Matlab 來做這次的 Project2，我將 Joint move、Cartesian move 分別寫在 robotic_p2_joint.m 和 robotic_p2_cartesian.m 中。

### 以 robotic_p2_joint.m 為例:

點選介面上面的 run 鍵來做執行(如左圖所示)

執行之後會呈現出四張圖，Figure 1、Figure 2、Figure 3、Figure 4 分別為位置、速度、加速度和 3D 軌跡圖。

### 以 robotic_p2_cartesian.m 為例:

同樣點選介面上面的 run 鍵來做執行(如左圖所示)

執行之後會呈現出四張圖，Figure 1、Figure 2、Figure 3、Figure 4 分別為位置、速度、加速度和 3D 軌跡圖。

# (二)程式架構說明

### 先以 robotic_p2_joint.m 為例:

主要程式架構就是定義所需參數再數學運算，最後再把圖形給印出來。
(取部分程式碼來做說明)

```
clc;
clear;
%由逆運動矩陣求出，助教說可以寫死角度
A = [-100.4577 70.6108 48.3997 0.0000 60.9895 29.2746];
B = [-52.1158 -1.4358 30.2060 58.6167 11.4781 1.5427 ];
C = [0.0955 65.7969 14.3196 -164.6623 20.1731 -149.9598];

%建立array在迴圈裡，如次可以存放迴圈計算出的值
[x_top,y_top ,z_top, ...
 x_dir, y_dir ,z_dir] = deal([]);

for i = 1:6
    eval(['p' num2str(i) ' = [];']);
    eval(['v' num2str(i) ' = [];']);
    eval(['a' num2str(i) ' = [];']);
end
sampling_time = 0.002;
```

首先前面第一大段，我利用我的 Project1 的逆運動學把題目的 4*4 矩陣給求得其 theta 值，每個計算出來都各有八組解，我都從中挑選一組當作我的輸入角度，經過與助教討論後，確定可以直接寫其值在我的程式中。

```
%路徑前面段-線性
for t = -0.5:sampling_time:-0.2-sampling_time
    h = (t+0.5)/0.5; %將坐標軸定義移到A段的起點
    %依照講義公式求出所需值並代入
    delta_b1= B(1)-A(1);
    delta_b2= B(2)-A(2);
    delta_b3= B(3)-A(3);
    delta_b4= B(4)-A(4);
    delta_b5= B(5)-A(5);
    delta_b6= B(6)-A(6);
    q1 = delta_b1*h + A(1);
    q2 = delta_b2*h + A(2);
    q3 = delta_b3*h + A(3);
    q4 = delta_b4*h + A(4);
    q5 = delta_b5*h + A(5);
    q6 = delta_b6*h + A(6);
    p1 = [p1 q1];
    p2 = [p2 q2];
    p3 = [p3 q3];
    p4 = [p4 q4];
    p5 = [p5 q5];
    p6 = [p6 q6];
```

這部分是數學計算的主要程式碼，主要分為三段，第一段和第三段為線性的，第二段為非線性的，我這邊設了三個迴圈來分別去對他們計算，透過老師的上課講義來做公式的撰寫，如此來得到我所要的參數值，我前面定義的 Array 會在迴圈裡面不斷的存放每個時間點計算出來的值，這樣我就可以畫出他的線圖，我也在每個迴圈後面存該線段的最後點位置，而且我原先是把座標起點訂在 0，也就是最左邊點的位置，但不斷出現問題，在與助教討論之後，我們認為是老師講義中的公式的起點是訂在中間點，改正之後經過嘗試，我需要把座標的起點移至兩線性的起點才不會有圖形不連續的奇怪現象。

```
%畫出角度與時間圖
figure(1);
set(gcf, 'Position', [100, 100, 900, 650]);
subplot(3, 2, 1);
plot(0:0.002:1, p1);
xticks(0:0.05:1);
title('Joint1');
xlabel('time(s)');
ylabel('angle(degree)');

subplot(3, 2, 2);
plot(0:0.002:1, p2);
xticks(0:0.05:1);
yticks(0:20:80)
title('Joint2');
xlabel('time(s)');
ylabel('angle(degree)');

subplot(3, 2, 3);
plot(0:0.002:1, p3);
xticks(0:0.05:1);
yticks(0:20:130)
title('Joint3');
xlabel('time(s)');
ylabel('angle(degree)');

subplot(3, 2, 4);
plot(0:0.002:1, p4);
xticks(0:0.05:1);
title('Joint4');
xlabel('time(s)');
ylabel('angle(degree)');
```

最後一大段就是畫圖的部分了，把各個圖的名稱、x 和 y 軸的名稱給寫上去，必要的話也要把 x 和 y 軸的刻度給寫上去，在最後畫 3D 圖的時候也要記得把點位置給畫上去，最後的 Function 定義是運用我上次作業的 Project1 的程式碼，目的是為了計算出要畫 3D 圖所需要的數值，如此就可以得到我們要的 3D 軌跡圖。

## 再以 robotic_p2_cartesian.m 為例:

主要程式架構就是先定義並用正運動學來得到我要的數值，之後就是數學運算，最後再把我們所需要的圖給畫出來。

(取部分程式碼來做說明)

```
    0 0 -1 -0.6;
    0 0 0 1];

B = [0.87 -0.1 0.48 0.5;
    0.29 0.9 -0.34 -0.4;
    -0.4 0.43 0.81 0.4;
    0 0 0 1];

C = [0.41 -0.29 0.87 0.6;
    0.69 0.71 -0.09 0.15;
    -0.6 0.64 0.49 -0.3;
    0 0 0 1];

for i = 1:6
    eval(['p' num2str(i) ' = [];']);
    eval(['v' num2str(i) ' = [];']);
    eval(['a' num2str(i) ' = [];']);
end

[x_top,y_top ,z_top, ...
 x_dir, y_dir ,z_dir] = deal([]);
sampling_time = 0.002;

%對A矩陣做我Project1的正運動學轉換
if A(3,3) == 1 || A(3,3) == -1
    % 如果為奇異點，將把其值設為0
    theta_A = 0;
    psi_A = 0; % 可為任意值
    phi_A = atan2(A(1,2), A(1,1));
else
    theta_A = acos(A(3,3));         % 先利用 Z-axis 旋轉
    psi_A = atan2(A(2,3), A(1,3)); % 再利用 Y-axis 旋轉
    phi_A = atan2(A(3,2), -A(3,1));% 最後再利用 Z-axis 旋轉

end
```

前面第一大段是先把之後會用到的參數給計算出來，我先把題目的三點先給定義好，要注意位置要記得把它改成公尺，之後再利用正運動學來得到三個點的相關參數，這邊的正運動學同樣是使用我 Project1 的正運動學程式碼來做計算，並定義好題目所需要的參數。

```
%路徑前面段-線性
for t = -0.5:sampling_time:-0.2-sampling_time
    h = (t+0.5)/0.5;%將坐標軸定義移到A段的起點
    %依照講義公式求出所需值並代入
    delta_b1 = B(1,4)- A(1,4);
    delta_b2 = B(2,4)- A(2,4);
    delta_b3 = B(3,4)- A(3,4);
    delta_b4 = phi_B - phi_A;
    delta_b5 = theta_B - theta_A;
    delta_b6 = psi_B - psi_A;
    q1 = delta_b1*h + A(1,4);
    q2 = delta_b2*h + A(2,4);
    q3 = delta_b3*h + A(3,4);
    q4 = delta_b4*h + phi_A;
    q5 = delta_b5*h + theta_A;
    q6 = delta_b6*h + psi_A;
    p1 = [p1 q1];
    p2 = [p2 q2];
    p3 = [p3 q3];
    p4 = [p4 q4];
    p5 = [p5 q5];
    p6 = [p6 q6];
```

第二大段和 Joint 一樣是做主要的數學運算，這部分是數學計算的主要程式碼，主要分為三段，第一段和第三段為線性的，第二段為非線性的，我這邊設了三個迴圈來分別去對他們計算，透過老師的上課講義來做公式的撰寫，如此來得到我所要的參數值，我前面定義的 Array 會在迴圈裡面不斷的存放每個時間點計算出來的值，這樣我就可以畫出他的線圖，我也在每個迴圈後面存該線段的最後點位置，而且我原先是把座標起點訂在 0，也就是最左邊點的位置，但不斷出現問題，在與助教討論之後，我們認為是老師講義中的公式的起點是訂在中間點，改正之後經過嘗試，我需要把座標的起點移至兩線性的起點才不會有圖形不連續的奇怪現象。

```
%畫出角度與時間圖
figure(1);
set(gcf, 'Position', [100, 100, 900, 650]);
subplot(3, 1, 1);
plot(0:0.002:1, x_top);
xticks(0:0.05:1);
yticks(0:0.1:1)
xlabel('time(s)');
ylabel('Position(m)');

subplot(3, 1, 2);
plot(0:0.002:1, y_top);
xticks(0:0.05:1);
xlabel('time(s)');
ylabel('Position(m)');


subplot(3, 1, 3);
plot(0:0.002:1, z_top);
xticks(0:0.05:1);
yticks(-2:0.1:1)
xlabel('time(s)');
ylabel('Position(m)');

%畫出角速度與時間圖
figure(2);
set(gcf, 'Position', [100, 100, 900, 650]);
subplot(3, 1, 1);
plot(0:0.002:1, v1);
xticks(0:0.05:1);
xlabel('time(s)');
ylabel('Velocity(m/s)');
```

最後一大段就是畫圖的部分了，把各個圖的名稱、x 和 y 軸的名稱給寫上去，必要的話也要把 x 和 y 軸的刻度給寫上去，在最後畫 3D 圖的時候也要記得把點位置給畫上去，最後的 Function 定義是運用我上次作業的 Project1 的程式碼，目的是為了計算出要畫 3D 圖所需要的數值，如此就可以得到我們要的 3D 軌跡圖。

# (三)數學運算說明

手寫運算老師講義的數學公式，以及繪畫出其運動簡易示意圖。

透過對位置資訊做一次微分和兩次微分來得到速度和加速度的公式，之後再對各個時間做迴圈來記錄各個時間點的值，之後就可以畫出位置、速度、加速度和 3D 的軌跡圖，主要分為線性和非線性兩種公式，對他們做微分即可得到速度和加速度公式。

# 機器人學 312605015 詹恆瑜 Project2 數學推導

$$q(t) = a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0$$

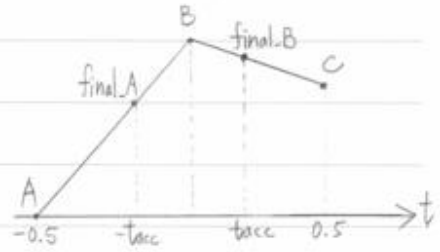
$$\Rightarrow q(0) = a_0$$

$$\dot{q}(t) = 4a_4t^3 + 3a_3t^2 + 2a_2t + a_1$$

$$\Rightarrow \dot{q}(0) = a_1$$

$$\ddot{q}(t) = 12a_4t^2 + 6a_3t + 2a_2$$

$$\Rightarrow \ddot{q}(0) = 2a_2$$




中間加減速段：$h = \frac{t + t_{acc}}{2t_{acc}}$ $-t_{acc} \le t \le t_{acc} \Rightarrow \frac{-t_{acc}+t_{acc}}{2t_{acc}} \le h \le \frac{2t_{acc}}{2t_{acc}} = 1$

$$q(h) = \left[\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right)(2-h)h^2 - 2\Delta B\right]h + B + \Delta B \quad , \Delta B = A - B,\ \Delta C = C - B$$

$$\Rightarrow q(0) = A$$

$$\dot{q}(h) = \left[\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right)(1.5-h)2h^2 - \Delta B\right]\frac{1}{t_{acc}}$$

$$\Rightarrow \dot{q}(0) = -\frac{\Delta B}{t_{acc}}$$

$$\ddot{q}(h) = \left[\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right)(1-h)\right]\frac{3h}{t_{acc}^2}$$

$$\Rightarrow \ddot{q}(0) = 0$$

linear：$h = \frac{t}{T}$ , $t_{acc} \le t \le T - t_{acc}$

$$q = \Delta C \cdot h + B$$

$$= \Delta C \frac{t}{T} + B$$

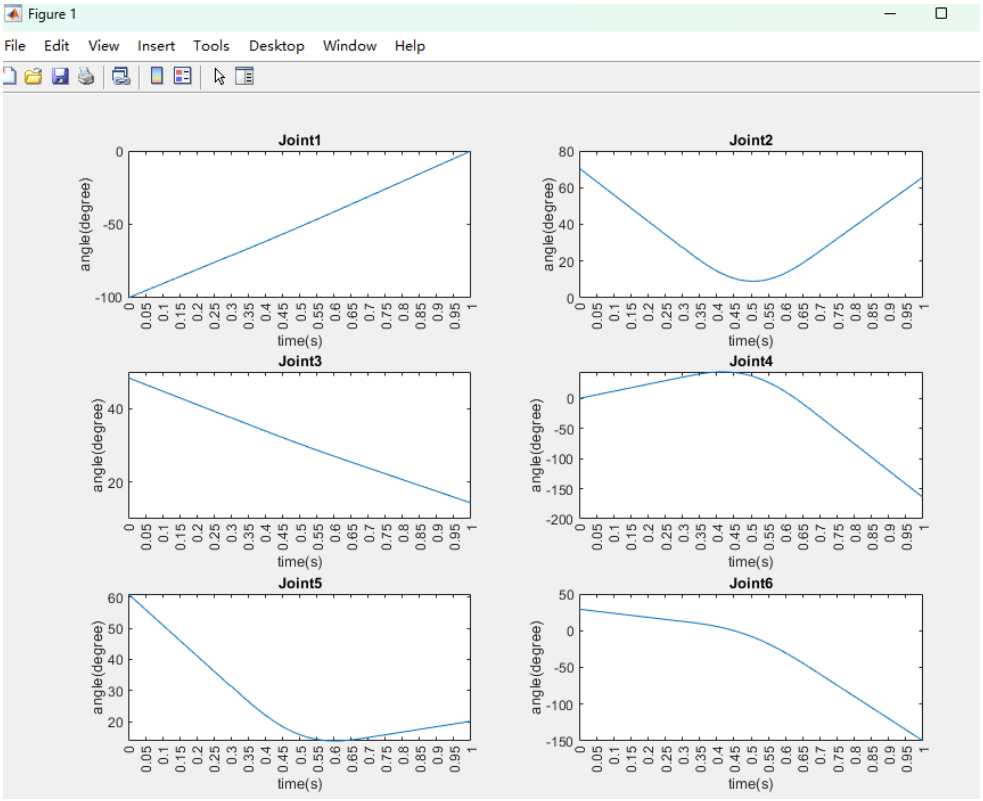
$$\dot{q}(t) = \Delta C \cdot \frac{1}{T}$$

$$\ddot{q}(t) = 0$$

$$q(1) = a_4 + a_3 + a_2 + a_1 + a_0$$
$$= \left[ (\Delta C \frac{t_{acc}}{T} + \Delta B) - 2\Delta B \right] + A$$
$$\dot{q}(1) = 4a_4 + 3a_3 + 2a_2 + a_1$$
$$= \left[ (\Delta C \frac{t_{acc}}{T} + \Delta B) \times \frac{1}{2} \times 2 - \Delta B \right] \frac{1}{t_{acc}}$$
$$\ddot{q}(1) = 12a_4 + 6a_3 + 2a_2$$
$$= 0$$

# (四)軌跡規劃曲線圖結果

先以 robotic_p2_joint.m 為例: (Joint move)



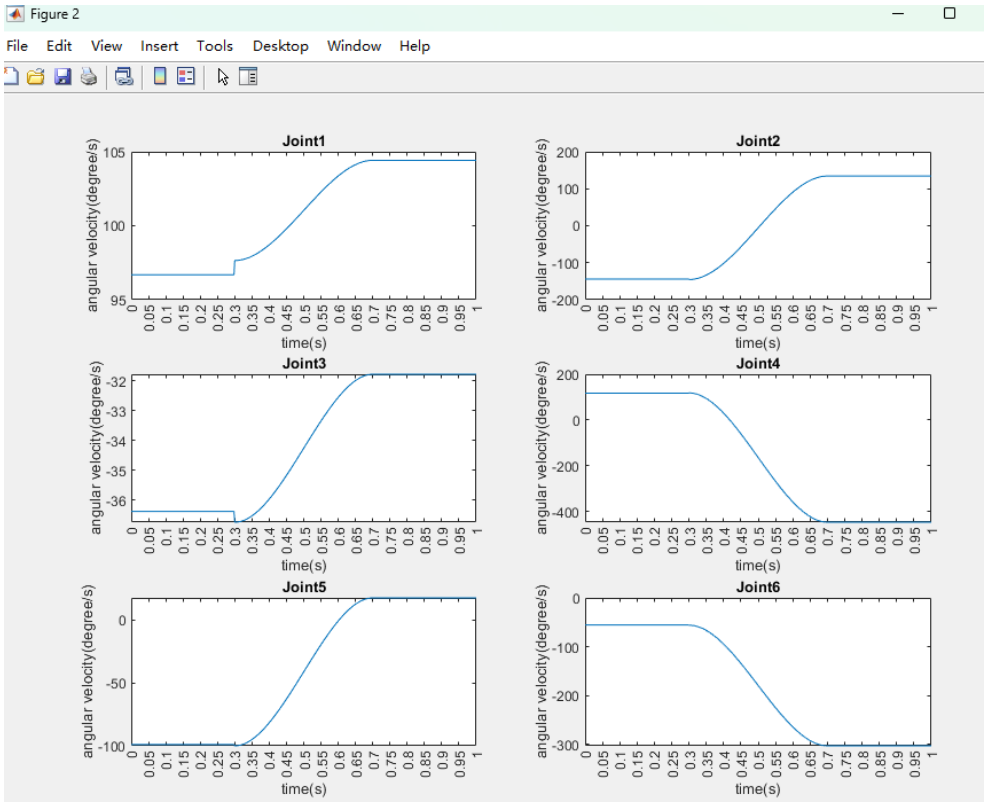


Figure 2
File Edit View Insert Tools Desktop Window Help
Joint1
Joint2
Joint3
Joint4
Joint5
Joint6
angular velocity(degree/s)
time(s)

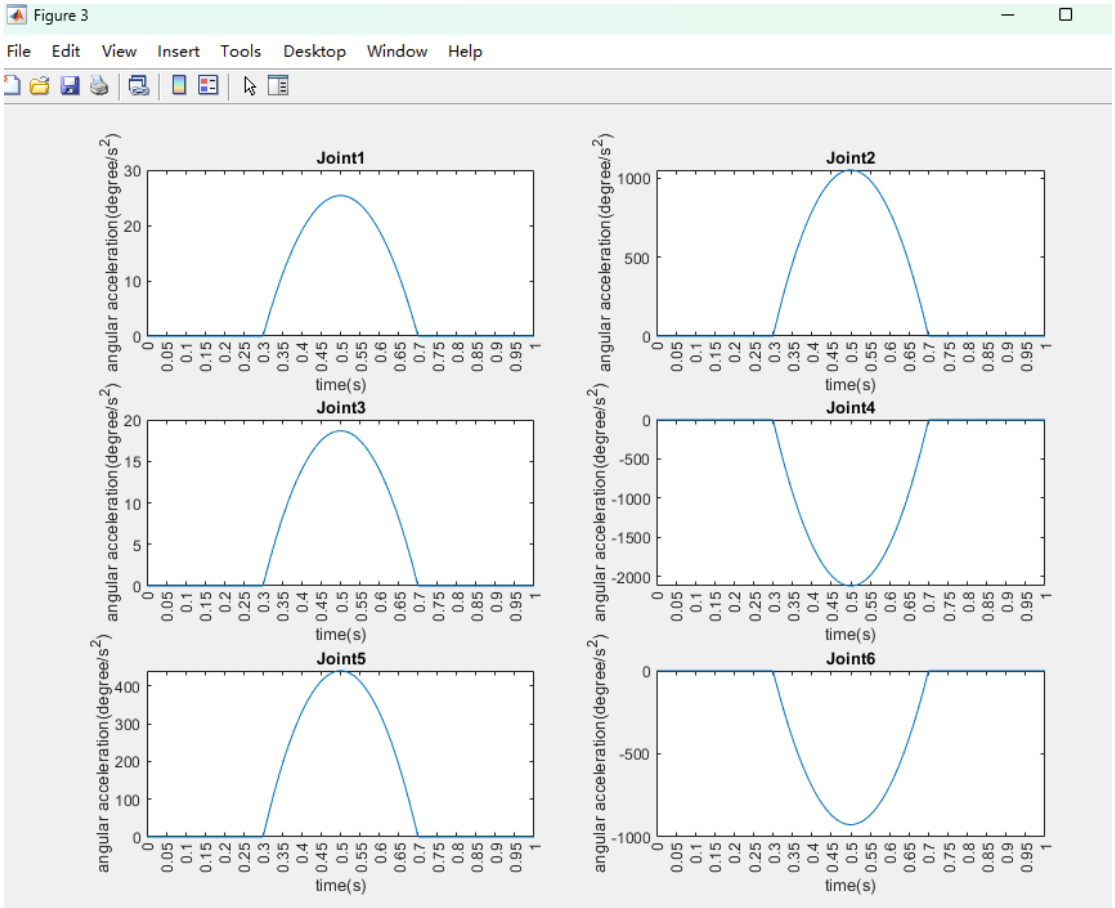
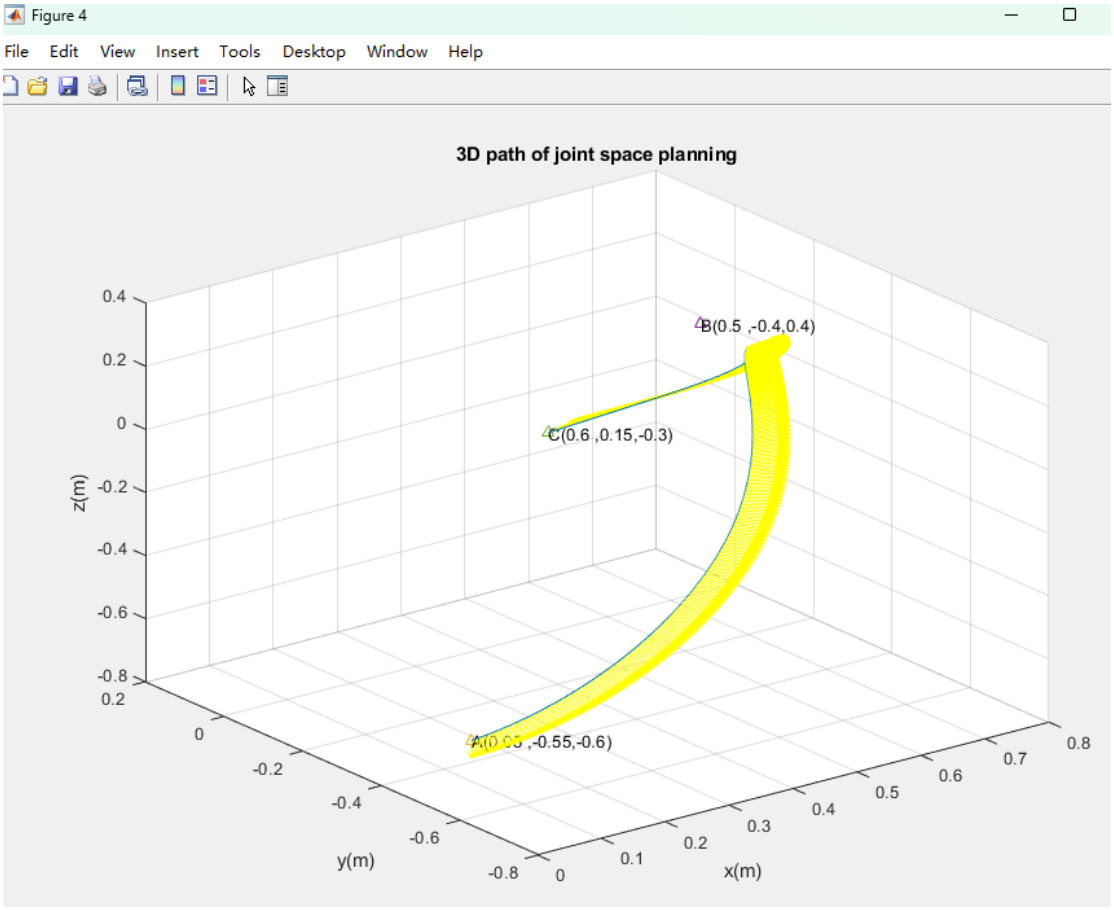
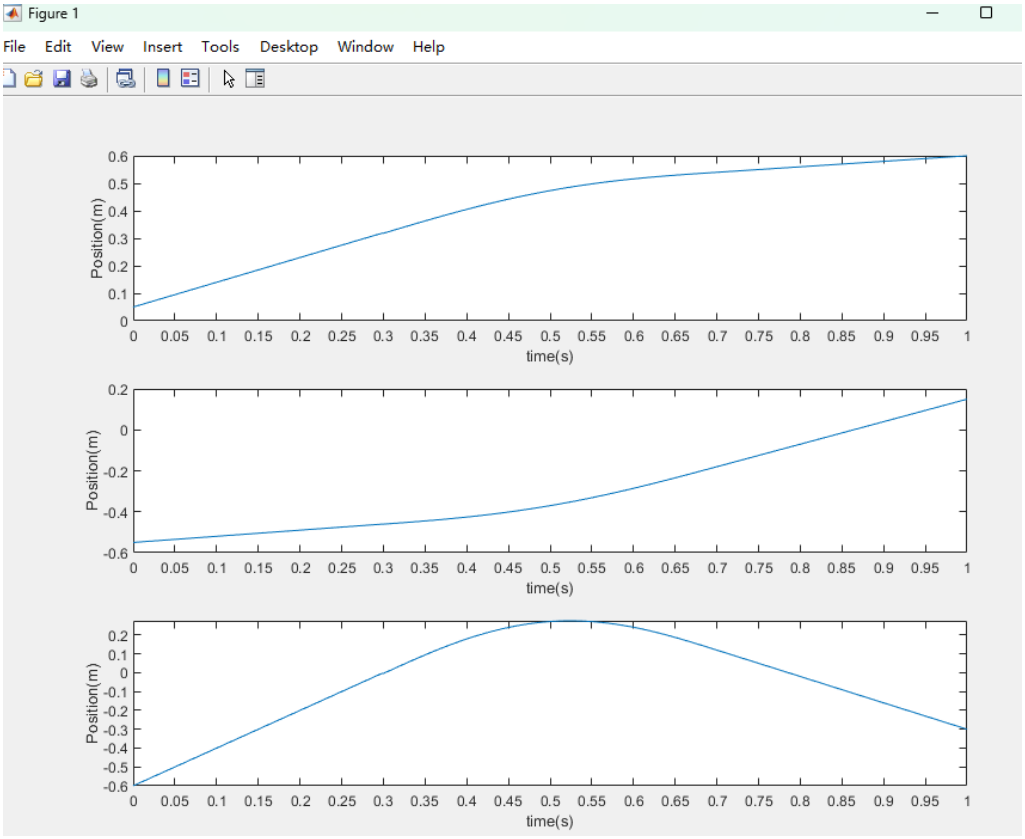


Figure 3
File Edit View Insert Tools Desktop Window Help
Joint1
Joint2
Joint3
Joint4
Joint5
Joint6
angular acceleration(degree/s^2)
time(s)




再以 robotic_p2_cartesian.m 為例: (Cartesian move)



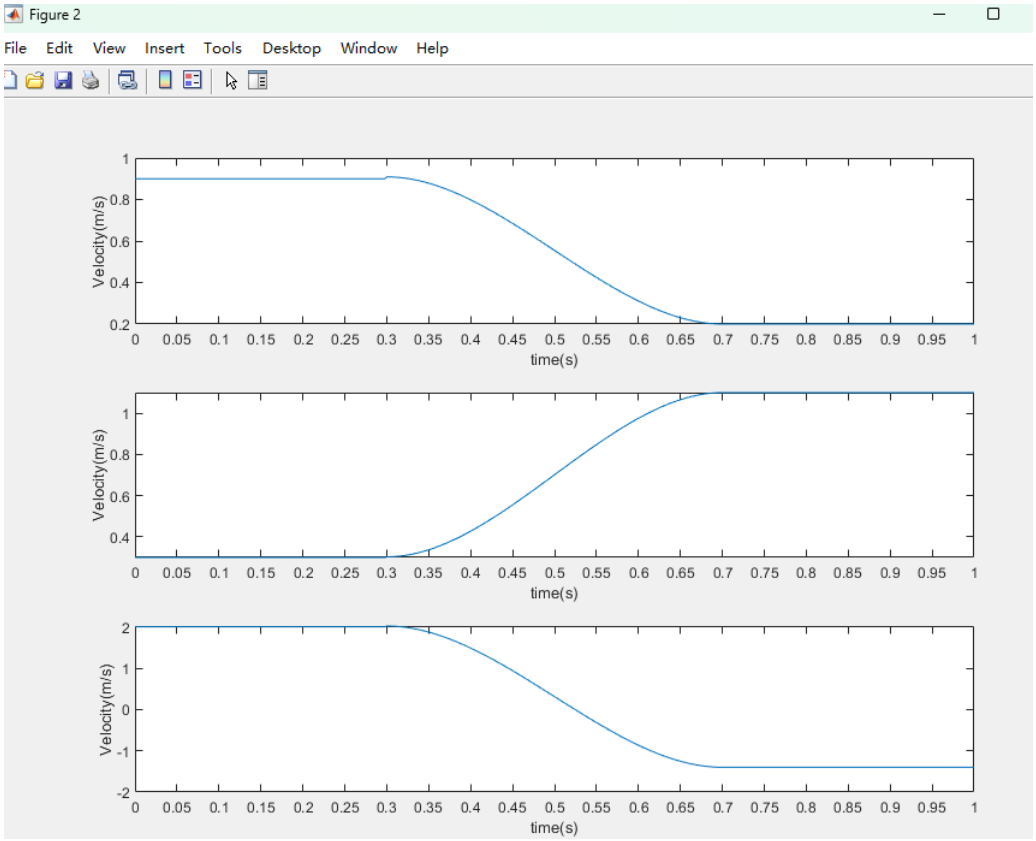


Figure 2
File Edit View Insert Tools Desktop Window Help
Velocity(m/s)
time(s)
Velocity(m/s)
time(s)
Velocity(m/s)
time(s)

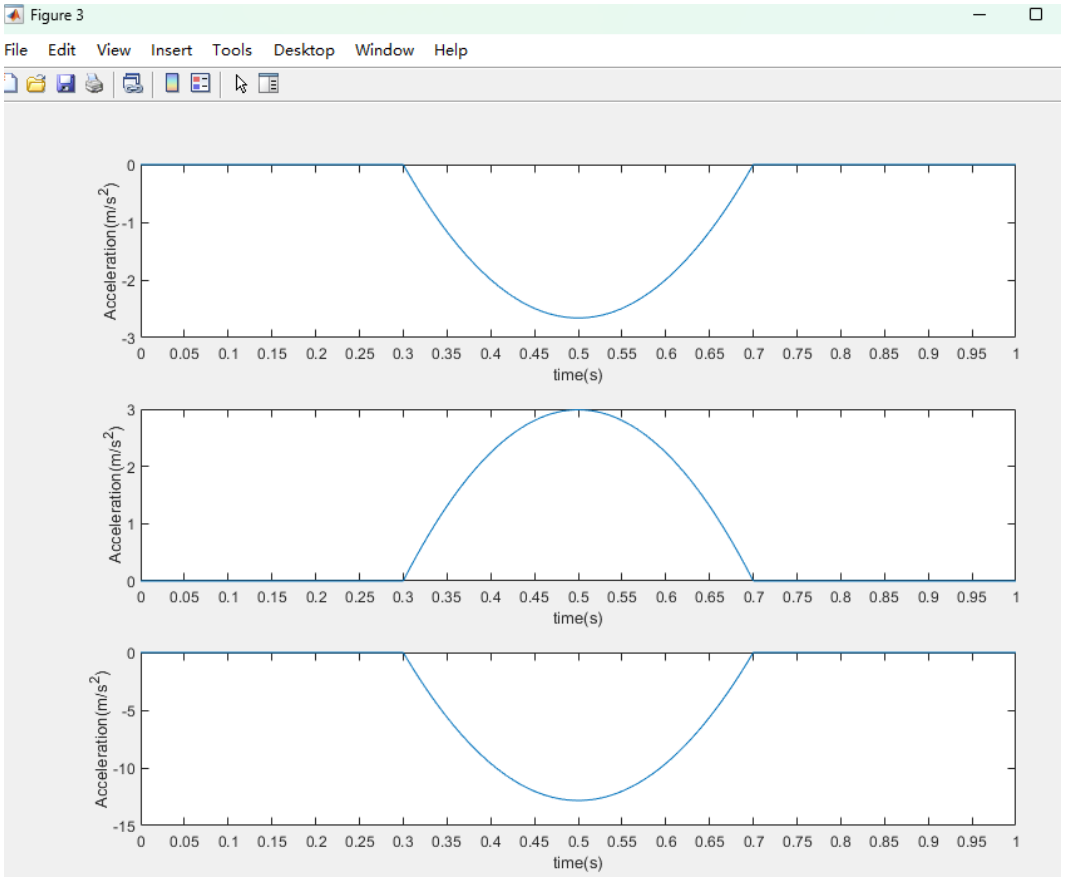
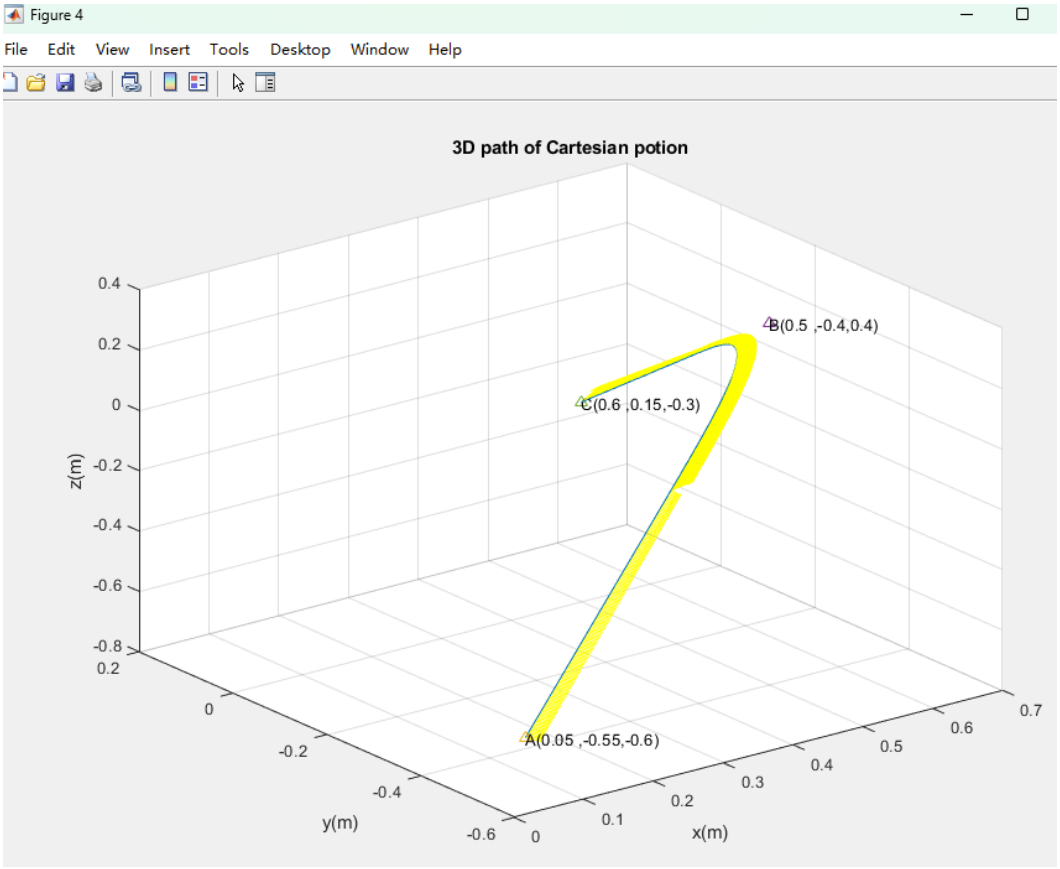


Figure 3
File Edit View Insert Tools Desktop Window Help
Acceleration(m/s2)
time(s)
Acceleration(m/s2)
time(s)
Acceleration(m/s2)
time(s)




# (四)加分題-討論兩種軌跡規劃的優缺點、遇到奇異點如何處理

## Joint move

優點:

1. 可以直接對各關節去做控制，尤其是對多軸機器人來說，更能夠精準的控制機器手臂可以達到我們想要的他到的位置。
2. 在運動受限的區域較適合使用此方法，因為關節控制可以在運動中擁有更高的靈活性。
3. 在能源消耗方面相對節能。

缺點:

1. 某些關節本身的限制會使一些位置無法到達。
2. 各關節的複雜移動中可能會互相干擾，設計起來就會比較麻煩。
3. 在設計移動時相對比較沒那麼直觀，對於設計者的門檻會相對較高。

## Cartesian move

優點:

1. 擁有高的運動穩定性及精度。
2. 在做修正時由於相對直觀，所以我們調整時相對簡單許多。

3. 相對更加簡單去讓機器手臂在複雜的路徑上運動。

缺點:

1. 在複雜和精度高的運動之中，有機會比關節移動還遲鈍。
2. 計算成本高需要複雜的逆運動學，所以相對容易有錯誤發生。

## 遇到奇異點如何處理?

透過運動學分析，我們知道當到達奇異點會產生不符合預期的運動，所以我會規畫其他路線避免機器人到達奇異點，可以透過增加維度等方式去避開奇異點。