機器人學 project2

電控所 碩一

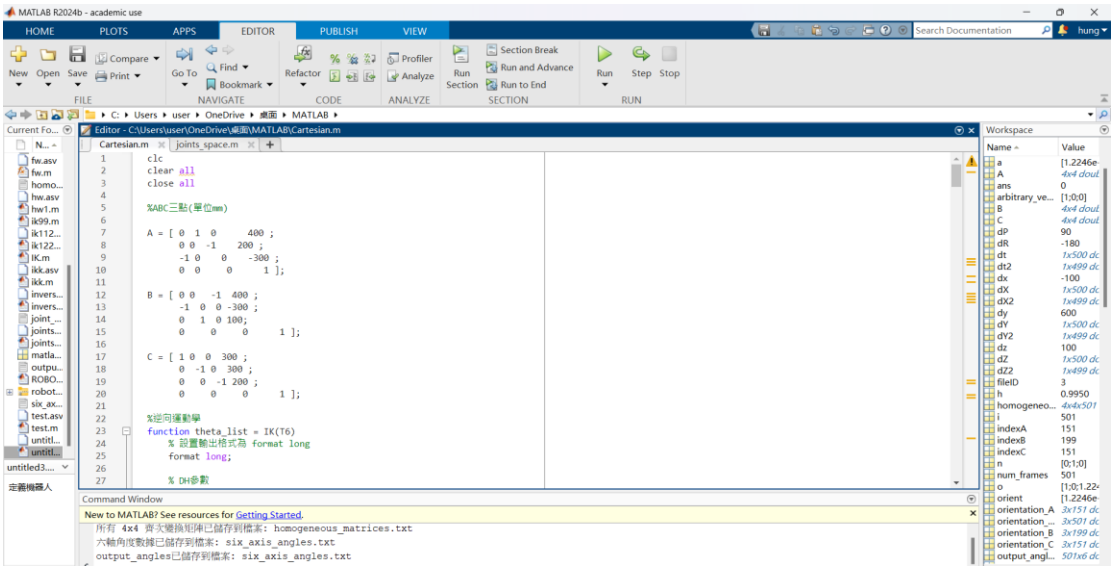
姓名:洪亮

信箱:31morris.abc@gmail.com

學號:313512072

一、 介面說明

開發平台 :MATLAB



開啟 Cartesian.m / joints_space.m 後依照下列程序:

1. 程式編輯視窗

2. 按下 “Run” 執行程式碼

二、 程式架構說明

joint_space.m

定義變數和參數

```
% 變數
init_p = [];end_p = [];
target_p = [];start_point = [];end_point = [];
init_px = [];init_py = [];init_pz = [];
end_px = [];end_py = [];end_pz = [];

%位置
p1 = [];p2 = [];p3 = [];p4 = [];p5 = [];p6 = [];

%速度
v1 = [];v2 = [];v3 = [];v4 = [];v5 = [];v6 = [];

%加速度
acc1 = [];acc2 = [];acc3 = [];acc4 = [];acc5 = [];acc6 = [];

% 參數
sampling_time = 0.002;
```

利用 project 1 的逆向運動學得到 A、B、C 三點關節角度

```
%利用project 1的逆向運動學得到ABC關節角度
A = [26.5651 72.6476 -58.9929 -13.6546 -153.4349   90.0000];
B = [-36.8699 25.7996 -79.2267 53.4272 -126.8699 180.0000];
C = [45.0000 12.2384 -88.9084 -13.33 -90.0000 135.0000];
```

以下程式碼的主要功能是在給定的時間範圍內（-0.5 到 0.5 秒，步長為 sampling_time），根據當前時間段的條件計算機器人關節的即時狀態，包括位置、速度和加速度，根據時間 t 分為三個主要區間進行處理：

區間 1: $t<-0.2t < -0.2t<-0.2$

- **位置計算**:
    - 使用線性插值計算目標位置：

$$target\_p = del\_BA \cdot \left(\frac{t}{0.5}\right) + A$$

- **速度計算**:
    - 速度為常數：

$$joint\_v = \frac{del\_BA}{0.5}$$

- **加速度計算**:
    - 加速度為零向量，因為此區間是線性運動。

```
for t = 0.5*-1:sampling_time:0.5
    if t < 0.2*-1
        t = t + 0.5;
        target_p = del_BA*(t/0.5) + A;
        start_point = target_p;
        [X_i, Y_i, Z_i, X_f, Y_f, Z_f] = FK(target_p);

        % 更新初始和最終位置
        init_px = [init_px X_i];init_py = [init_py Y_i];init_pz = [init_pz Z_i];
        end_px = [end_px X_f];end_py = [end_py Y_f];end_pz = [end_pz Z_f];

        % 記錄每個關節的角度值
        p1 = [p1 target_p(1)];p2 = [p2 target_p(2)];p3 = [p3 target_p(3)];
        p4 = [p4 target_p(4)];p5 = [p5 target_p(5)];p6 = [p6 target_p(6)];

        % 計算關節的速度
        joint_v = del_BA/0.5;
        v1 = [v1 joint_v(1)];v2 = [v2 joint_v(2)];v3 = [v3 joint_v(3)];
        v4 = [v4 joint_v(4)];v5 = [v5 joint_v(5)];v6 = [v6 joint_v(6)];

        % 計算關節的加速度
        joint_a = zeros(6,1);
        acc1 = [acc1 joint_a(1)];acc2 = [acc2 joint_a(2)];acc3 = [acc3 joint_a(3)];
        acc4 = [acc4 joint_a(4)];acc5 = [acc5 joint_a(5)];acc6 = [acc6 joint_a(6)];
```

區間 2: $-0.2 \leq t \leq 0.2$

平滑過渡的三次多項式插值:

- 計算進度參數 h：

$$h = \frac{t + 0.2}{2 \cdot 0.2}$$

- 使用三次多項式插值計算目標位置：

$$target\_p = \left(\left(del\_CB \cdot \frac{0.2}{0.5} + del\_BA \right) \cdot (2 - h) \cdot h^2 - 2 \cdot del\_BA \right) \cdot h + start\_point$$

- **速度計算**:
    - 使用插值公式計算速度：

$$joint\_v = \frac{\left(del\_CB \cdot \frac{0.2}{0.5} + del\_BA\right) \cdot (1.5 - h) \cdot 2 \cdot h^2 - del\_BA}{0.2}$$

- **加速度計算**:
    - 加速度來自二階導數：

$$joint\_a = \frac{\left(del\_CB \cdot \frac{0.2}{0.5} + del\_BA\right) \cdot (1 - h) \cdot 3 \cdot h}{0.2^2}$$

```
elseif t <= 0.2 && t >= 0.2*-1
    del_BA = start_point - B;
    h = (t + 0.2) / (2 * 0.2);
    target_p = ((del_CB * (0.2/0.5) + del_BA)*(2-h)*(h^2) - (2 * del_BA)) * h + start_point;
    end_point = target_p;
    [X_i, Y_i, Z_i, X_f, Y_f, Z_f] = FK(target_p);

    % 更新初始和最終位置
    init_px = [init_px X_i];init_py = [init_py Y_i];init_pz = [init_pz Z_i];
    end_px = [end_px X_f];end_py = [end_py Y_f];end_pz = [end_pz Z_f];

    % 記錄每個關節的角度值
    p1 = [p1 target_p(1)];p2 = [p2 target_p(2)];p3 = [p3 target_p(3)];
    p4 = [p4 target_p(4)];p5 = [p5 target_p(5)];p6 = [p6 target_p(6)];

    % 計算關節的速度
    joint_v = ((del_CB * (0.2/0.5) + del_BA)*(1.5-h)*2*(h^2) - del_BA)/0.2;
    v1 = [v1 joint_v(1)];v2 = [v2 joint_v(2)];v3 = [v3 joint_v(3)];
    v4 = [v4 joint_v(4)];v5 = [v5 joint_v(5)];v6 = [v6 joint_v(6)];

    % 計算關節的加速度
    joint_a = ((del_CB * (0.2/0.5) + del_BA)*(1-h))*(3*h)/(0.2^2);
    acc1 = [acc1 joint_a(1)];acc2 = [acc2 joint_a(2)];acc3 = [acc3 joint_a(3)];
    acc4 = [acc4 joint_a(4)];acc5 = [acc5 joint_a(5)];acc6 = [acc6 joint_a(6)];
```

區間 3: t>0.2

位置計算:

- 使用線性插值：

$$target\_p = del\_CB \cdot \left(\frac{t - 0.2}{0.5}\right) + end\_point$$

- **速度計算**:
    - 速度為常數：

$$joint\_v = \frac{del\_CB}{0.5}$$

- **加速度計算**:
    - 加速度為零向量，因為此區間是線性運動。

```
    elseif t > 0.2
        t = (t-0.2);
        target_p = del_CB*(t/0.5) + end_point;
        [X_i, Y_i, Z_i, X_f, Y_f, Z_f] = FK(target_p);

        % 更新初始和最終位置
        init_px = [init_px X_i];init_py = [init_py Y_i];init_pz = [init_pz Z_i];
        end_px = [end_px X_f];end_py = [end_py Y_f];end_pz = [end_pz Z_f];

        % 記錄每個關節的角度值
        p1 = [p1 target_p(1)];p2 = [p2 target_p(2)];p3 = [p3 target_p(3)];
        p4 = [p4 target_p(4)];p5 = [p5 target_p(5)];p6 = [p6 target_p(6)];

        joint_v = del_CB/0.5;
        v1 = [v1 joint_v(1)];v2 = [v2 joint_v(2)];v3 = [v3 joint_v(3)];
        v4 = [v4 joint_v(4)];v5 = [v5 joint_v(5)];v6 = [v6 joint_v(6)];

        joint_a = zeros(6,1);
        acc1 = [acc1 joint_a(1)];acc2 = [acc2 joint_a(2)];acc3 = [acc3 joint_a(3)];
        acc4 = [acc4 joint_a(4)];acc5 = [acc5 joint_a(5)];acc6 = [acc6 joint_a(6)];
    end
end
```

以下程式繪製了機器人在三維空間中的運動軌跡、標記了三個參考點A、B、C，並用箭頭顯示關節加速度的方向和大小。它以三維曲線表示運動路徑，標示點位置及座標，並用箭頭視覺化加速度。

```
%繪製Joint move
figure
plot3(init_px, init_py, init_pz)
xlabel('x(mm)');ylabel('y(mm)');zlabel('z(mm)');
grid;
hold on;
%A點
plot3(400,200,-300,'r*')
text(400,200,-300,'A (400,200,-300)')
%B點
plot3(400,-300,100,'r*')
text(400,-300,100,'B (400,-300,100)')
%C點
plot3(300,300,200,'r*')
text(300,300,200,'C (300,300,200)')

%繪製加速度方向的箭頭
quiver3(init_px, init_py, init_pz, end_px, end_py, end_pz, 'Color', [0, 1, 1])
```

以下是程式為順向運動學，主要負責根據給定的機器人關節角度，計算其末端執行器在三維空間中的位置與方向，並將這些結果應用於運動軌

跡的規劃與可視化。

```
%順向運動學
function [init_x,init_y,init_z,end_x,end_y,end_z] = FK(theta)
    theta1 = theta(1);
    theta2 = theta(2);
    theta3 = theta(3);
    theta4 = theta(4);
    theta5 = theta(5);
    theta6 = theta(6);

    A1 = [  cosd(theta1),              0,   -sind(theta1),    120*cosd(theta1);
        sind(theta1),              0,    cosd(theta1),    120*sind(theta1);
                     0,             -1,               0,               0;
                     0,              0,               0,               1];

    A2 = [  cosd(theta2),   -sind(theta2),               0,    250*cosd(theta2);
            sind(theta2),    cosd(theta2),               0,    250*sind(theta2);
                         0,              0,               1,               0;
                         0,              0,               0,               1];

    A3 = [  cosd(theta3),   -sind(theta3),               0,    260*cosd(theta3);
            sind(theta3),    cosd(theta3),               0,    260*sind(theta3);
                         0,              0,               1,               0;
                         0,              0,               0,               1];

    A4 = [  cosd(theta4),              0,   -sind(theta4),               0;
            sind(theta4),              0,    cosd(theta4),               0;
                         0,             -1,               0,               0;
                         0,              0,               0,               1];


    A5 = [  cosd(theta5),              0,    sind(theta5),               0;
            sind(theta5),              0,   -cosd(theta5),               0;
                         0,              1,               0,               0;
                         0,              0,               0,               1];

    A6 = [  cosd(theta6),   -sind(theta6),               0,               0;
            sind(theta6),    cosd(theta6),               0,               0;
                         0,              0,               1,               0;
                         0,              0,               0,               1];

    T6 = A1*A2*A3*A4*A5*A6;
    init_x = T6(13);
    init_y = T6(14);
    init_z = T6(15);
    end_x = init_x + T6(9) * 2;
    end_y = init_y + T6(10) * 2;
    end_z = init_z + T6(11) * 2;
end
```

以下是將 6 軸角度儲存成 txt 的程式

```
% 儲存joint_angles.txt
fileID = fopen('joint_angles.txt', 'w');
fprintf(fileID, '\tjoint1\t\tjoint2\t\tjoint3\t\tjoint4\t\tjoint5\t\tjoint6\t\t if out of range\n');
for i = 1:length(p1)
    fprintf(fileID, '%d\t', i);
    fprintf(fileID, '%f\t', p1(i), p2(i), p3(i), p4(i), p5(i), p6(i));
    fprintf(fileID, '\n');
end
fclose(fileID);
```

Cartesian.m

定義三個目標點 A,B,C，包含位置（平移向量）和姿態（旋轉矩陣）

```
%ABC三點(單位mm)

A = [ 0  1  0      400 ;
      0 0  -1    200 ;
      -1 0    0    -300 ;
      0  0     0      1 ];

B = [ 0 0   -1  400 ;
      -1  0  0 -300 ;
      0   1  0 100;
      0     0     0      1 ];

C = [ 1 0  0  300 ;
      0  -1 0  300 ;
      0   0  -1 200 ;
      0     0     0      1 ];
```

使用 project1 的逆向運動學

```
%逆向運動學
function theta_list = IK(T6)
```

使用 find_6D_poses 函數從目標點齊次矩陣 A, B, C 中分解出位姿參數，設定插值的時間步長為 0.002

```
% 取得6D位姿的參數（位置和姿態）
[xA ,yA ,zA ,RA ,PA, YA] = find_6D_poses(A);
[xB ,yB ,zB ,RB ,PB, YB] = find_6D_poses(B);
[xC ,yC ,zC ,RC ,PC, YC] = find_6D_poses(C);

% 採樣時間
sampling_time = 0.002;
```

以下程式碼計算了從點 A 到點 B1 以及從 B2 到點 C 的位移和姿態變化，並使用插值平滑過渡。在每個時間點，程式根據位移差值和姿態差值計算位置和姿態，並存儲到相應的矩陣中。對於每個時間點，使用 Orientation 函數計算方向。最後利用二次插值計算曲線 B 的位置和姿態，並輸出平滑過渡的路徑、位置、姿態和方向。以下分別說明這三部分:

A 到點 B1:

計算從點 **A** 到點 **B1** 的位移差值 x, y, z 和姿態差值 Roll、Pitch、Yaw。並初始化 indexA 為 0，用於記錄生成的中間點數量。t：從 -0.5 到 -0.2，以 sampling_time 為步長進行迭代。h = (t+0.5)/0.5：將時間 t 映射為比例因子 h（從 0 到 1）。根據比例因子 h，計算每個中間點的位移差值 dx, dy, dz 和姿態差值 dR, dP, dY。然後更新中間點的座標和姿態，並存儲到矩陣 x_A, y_A, z_A（位置）和 R_A, P_A, Y_A（姿態）中。最後對於生成的每個中間點，使用 Orientation 函數，根據姿態角（Roll, Pitch, Yaw）計算方向，並存儲到矩陣 orientation_A 中。

```
%計算從A到B1的位移與姿態變化
indexA = 0;
x = xB - xA; y = yB - yA; z = zB - zA;
R = RB - RA; P = PB - PA; Y = YB - YA;

%使用採樣時間計算A到B1之間每個點的座標與姿態
for t=-0.5:sampling_time:-0.2
    indexA = indexA+1;
    h = (t+0.5)/0.5;
    dx = x*h; dy = y*h; dz = z*h;
    dR = R*h; dP = P*h; dY = Y*h;
    x_A(:,indexA) = xA+dx; y_A(:,indexA) = yA+dy; z_A(:,indexA) =zA+dz;
    R_A(:,indexA) = RA+dR; P_A(:,indexA) = PA+dP; Y_A(:,indexA) =YA+dY;
end

% 計算A到B1之間每個點的方向
for i = 1:indexA
    orientation_A(:,i) = Orientation(R_A(:,i),P_A(:,i),Y_A(:,i));
end
```

B2 到點 C:

計算從點 B2 到點 C 的位移差值 x, y, z 和姿態差值 Roll、Pitch、Yaw。並初始化 indexC 為 0，用於記錄生成的中間點數量。

t：從 0.2 到 0.5，以 sampling_time 為步長進行迭代。

h= h = t/0.5：將時間 t 映射為比例因子 hh（從 0.4 到 1）。

根據比例因子 h，計算每個中間點的位移差值 dx,dy,dz 和姿態差值 dR,dP,dY。

然後更新中間點的座標和姿態，並存儲到矩陣 xC,yC,zC （位置）和 RC,PC,YC（姿態）中。最後對於生成的每個中間點，使用 Orientation 函數，根據姿態角（Roll, Pitch, Yaw）計算方向，並存儲到矩陣 orientation_C 中。

```
%計算B2到C之間每個點的座標與姿態
indexC = 0;
x = xC - xB; y = yC - yB; z = zC - zB;
R = RC - RB; P = PC - PB; Y = YC - YB;

% 使用採樣時間計算B2到C之間每個點的座標與姿態
for t=0.2:sampling_time:0.5
    indexC = indexC+1;
    h = (t)/0.5;
    dx = x*h; dy = y*h; dz = z*h;
    dR = R*h; dP = P*h; dY = Y*h;
    x_C(:,indexC) = xB+dx; y_C(:,indexC) = yB+dy; z_C(:,indexC) = zB+dz;
    R_C(:,indexC) = RB+dR; P_C(:,indexC) = PB+dP; Y_C(:,indexC) = YB+dY;
end

%計算B2到C之間每個點的方向
for i = 1:indexC
    orientation_C(:,i) = Orientation(R_C(:,i),P_C(:,i),Y_C(:,i));
end
```

曲線 B:

這段程式碼透過三次貝茲曲線公式生成了從點 A 到點 C 經過點 B 的曲線，並計算曲線上每個中間點的座標和姿態。首先，程式初始化 A 和 C 相對於 B 的位移差值（xA,yA,zA 和 xC,yC,zC）及姿態差值（RA,PA,YA 和 RC,PC,YC），並將中間點計數器 indexB 初始化為 0。接著，通過迭代 t（範圍為 −0.2+sampling_time 至 0.2−sampling_time，步長為 sampling_time），計算比例因子 h，並利用貝茲曲線公式更新中間點的座標（xB,yB,zB）與姿態（RB,PB,YB）。最後，根據每個中間點的姿態角（Roll, Pitch, Yaw），使用 Orientation 函數計算其方向，並存儲到矩陣 orientation_B 中。

```
% 曲線B
xA = x_A(:,indexA)-xB; yA = y_A(:,indexA)-yB; zA = z_A(:,indexA)-zB;
RA = R_A(:,indexA)-RB; PA = P_A(:,indexA)-RB; YA = Y_A(:,indexA)-RB;
xC = x_C(:,1)-xB; yC = y_C(:,1)-yB; zC = z_C(:,1)-zB;
RC = R_C(:,1)-RB; PC = P_C(:,1)-PB; YC = Y_C(:,1)-YB;
indexB = 0;

% 使用採樣時間計算曲線B的每個點
for t=(-0.2+sampling_time):sampling_time:(0.2-sampling_time)
    indexB = indexB+1;
    h = (t+0.2)/(2*0.2);
    x_B(:,indexB) = xB + ((xC+xA)*(2-h)*h^2-2*xA)*h+xA;
    y_B(:,indexB) = yB + ((yC+yA)*(2-h)*h^2-2*yA)*h+yA;
    z_B(:,indexB) = zB + ((zC+zA)*(2-h)*h^2-2*zA)*h+zA;
    R_B(:,indexB) = RB + ((RC+RA)*(2-h)*h^2-2*RA)*h+RA;
    P_B(:,indexB) = PB + ((PC+PA)*(2-h)*h^2-2*PA)*h+PA;
    Y_B(:,indexB) = YB + ((YC+YA)*(2-h)*h^2-2*YA)*h+YA;
end
% 計算曲線B的每個點的方向
for i = 1:indexB
    orientation_B(:,i) = Orientation(R_B(:,i),P_B(:,i),Y_B(:,i));
end
```

以下是存 txt 檔的程式，首先，程式從給定的每一幀位置和方向數據中提取出位置向量和歐拉角，進而計算旋轉矩陣。然後，將位置和旋轉矩陣組合成齊次變換矩陣，並將這些矩陣儲存起來。接著，通過逆運動學函數計算出每一幀對應的關節角度，並將這些角度存入 txt 檔案。另外也會將每一幀的齊次變換矩陣和六軸角度數據儲存為單獨的 txt 檔案，方便檢查過程是否出錯。

```
% 將位置和方向數據組合成 4x4 齊次變換矩陣
num_frames = length(x_all);
homogeneous_matrices = zeros(4, 4, num_frames);

% 儲存輸出角度
output_angles = [];

for i = 1:num_frames
    % 取得當前的 X, Y, Z 和 Orientation
    pos = [x_all(i); y_all(i); z_all(i)];
    orient = [orientation_all(1,i), orientation_all(2,i), orientation_all(3,i)];

    % 將方向向量正規化為旋轉矩陣的一部分
    z_axis = orient(:) / norm(orient); % 確保方向向量正規化
    % 假設 X 軸與 Z 軸正交，並計算其他軸 (簡化計算)
    arbitrary_vector = [1; 0; 0]; % 任意參考向量
    if abs(dot(z_axis, arbitrary_vector)) > 0.9
        arbitrary_vector = [0; 1; 0]; % 避免平行
    end
    x_axis = cross(arbitrary_vector, z_axis);
    x_axis = x_axis / norm(x_axis);
    y_axis = cross(z_axis, x_axis); % 確保正交

    % 組成旋轉矩陣
    R = [x_axis, y_axis, z_axis];
```

```
    % 組成齊次變換矩陣
    T = eye(4);
    T(1:3, 1:3) = R;
    T(1:3, 4) = pos;
    homogeneous_matrices(:,:,i) = T;


    % 提取 n, o, a, p 作為逆運動學輸入
    n = T(1:3, 1); % 第一列
    o = T(1:3, 2); % 第二列
    a = T(1:3, 3); % 第三列
    p = T(1:3, 4); % 第四列

    % 將 n, o, a, p 組成 4x4 齊次變換矩陣
    T6 = [n, o, a, p; 0, 0, 0, 1]; % 將位置和方向組合成 4x4 齊次變換矩陣

    % 執行逆運動學計算
    theta = IK(T6);  % 傳遞 T6

    % 將結果存儲
    output_angles = [output_angles; theta]; % 去掉轉置操作，確保橫向追加
    %disp(size(output_angles)); % 顯示 output_angles 的行和列數
end
```

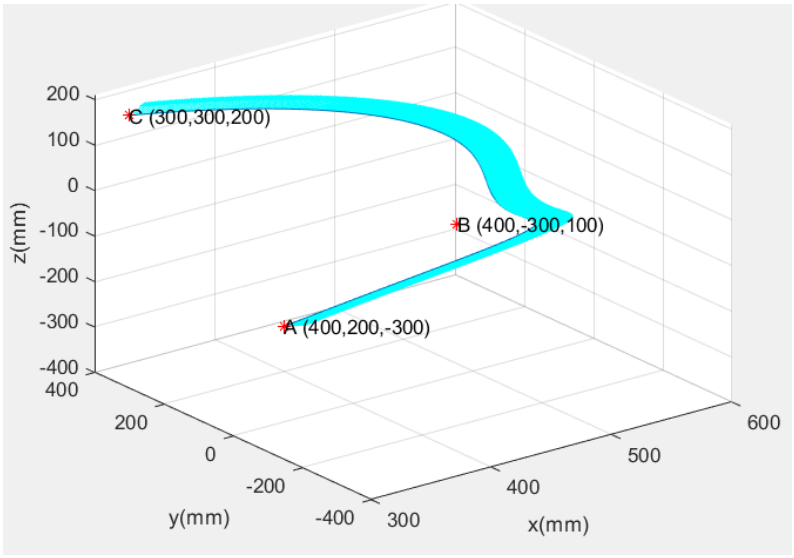
```
% 將角度結果存成txt檔案
output_filename = 'cartesian_angles.txt';
fileID = fopen('cartesian_angles.txt', 'w');

% 寫入標題
fprintf(fileID, '     joint1    joint2     joint3     joint4     joint5     joint6\n');

% 按行寫入，每行添加行號
for i = 1:size(output_angles, 1)
    fprintf(fileID, '%-4d %.6f %.6f %.6f %.6f %.6f %.6f\n', i, output_angles(i, :));
end

disp(['cartesian_angles已儲存到檔案: ', output_filename]);

% 關閉文件
fclose(fileID);
```

```
% 儲存為txt檔案
output_filename = 'homogeneous_matrices.txt';
fileID = fopen(output_filename, 'w');
for i = 1:num_frames
    fprintf(fileID, 'Frame %d:\n', i);
    fprintf(fileID, '%f\t%f\t%f\t%f\n', homogeneous_matrices(:,:,i)');
    fprintf(fileID, '\n');
end
fclose(fileID);

disp(['所有 4x4 齊次變換矩陣已儲存到檔案: ', output_filename]);

% find_6D_poses: find x,y,z,roll,pitch,yaw from a T6 matrix
function [x,y,z,phi,theta,psi] = find_6D_poses(T6)
    % Degree and Radius Transformation
    D_to_R = pi / 180;
    R_to_D = 180 / pi;

    nx = T6(1,1); ny = T6(2,1); nz = T6(3,1);
    ox = T6(1,2); oy = T6(2,2); oz = T6(3,2);
    ax = T6(1,3); ay = T6(2,3); az = T6(3,3);
    px = T6(1,4); py = T6(2,4); pz = T6(3,4);
    x = px; y = py; z = pz;
    phi = atan2(ay, ax) * R_to_D;
    theta = atan2(sqrt(ax^2 + ay^2), az) * R_to_D;
    psi = atan2(oz, -nz) * R_to_D;
    % p = [ x, y, z, phi,theta, psi];
end
```

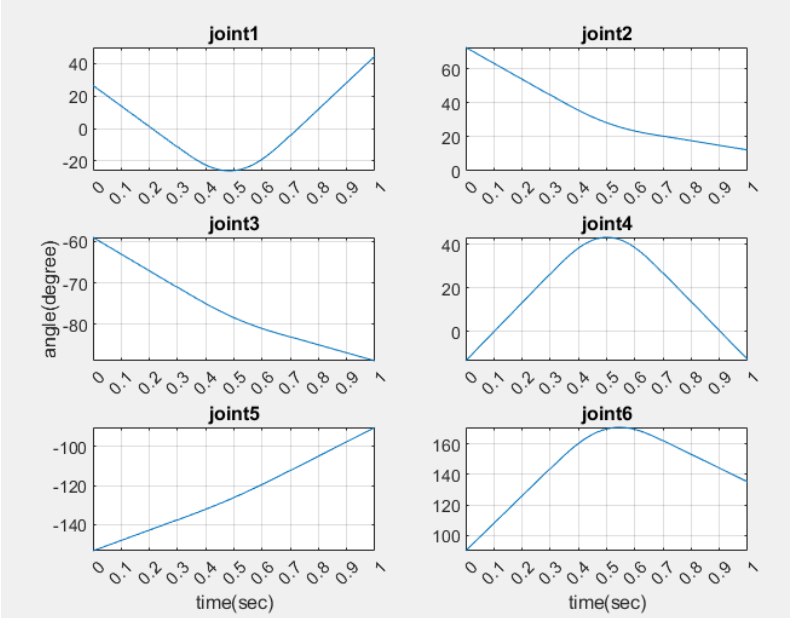
```
% Orientation: find the orientation of Z axis from raw pitch yaw
function orientation = Orientation(R,P,Y)
    % Degree and Radius Transformation
    D_to_R = pi / 180;
    R_to_D = 180 / pi;
    R = R * D_to_R;
    P = P * D_to_R;
    Y = Y * D_to_R;
    orientation = [cos(R)*sin(P) sin(R)*sin(P) cos(P)];
end

% 儲存六軸角度數據到txt檔案
output_data = [x_all', y_all', z_all', orientation_all(1,:)', orientation_all(2,:)', orientation_all(3,:)'];
output_filename = 'six_axis_angles.txt';  % 設定儲存的檔案名稱
writematrix(output_data, output_filename, 'Delimiter', 'tab');  % 儲存為Tab分隔的txt檔案
disp(['六軸角度數據已儲存到檔案: ', output_filename]);
```

Result:

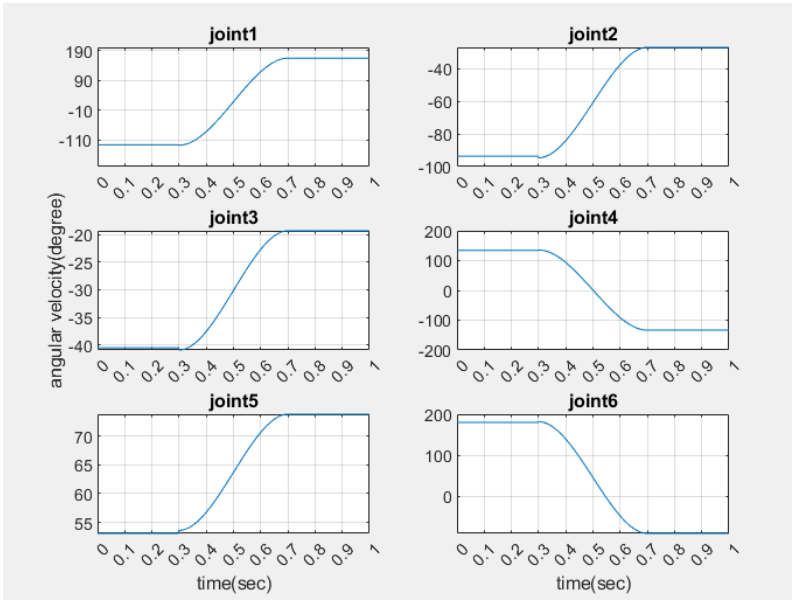
Joint space:



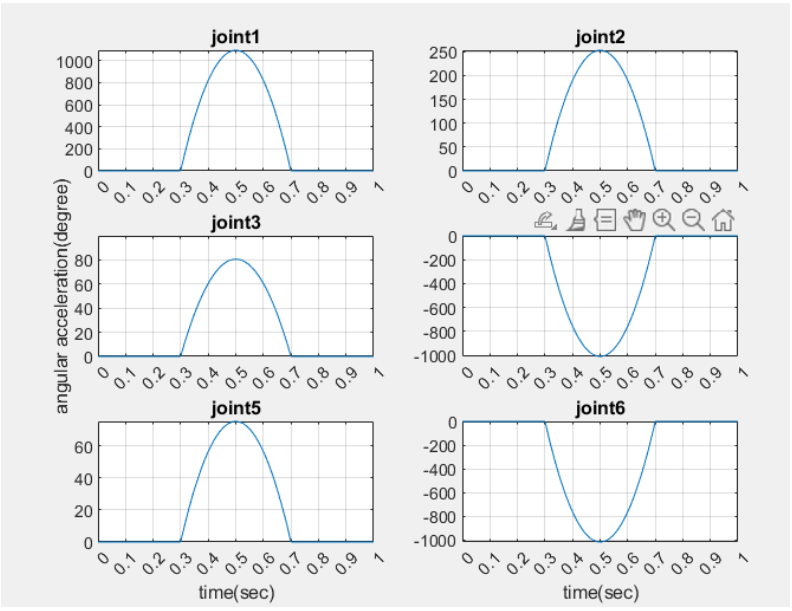
3D path



Angle



Angle velocity

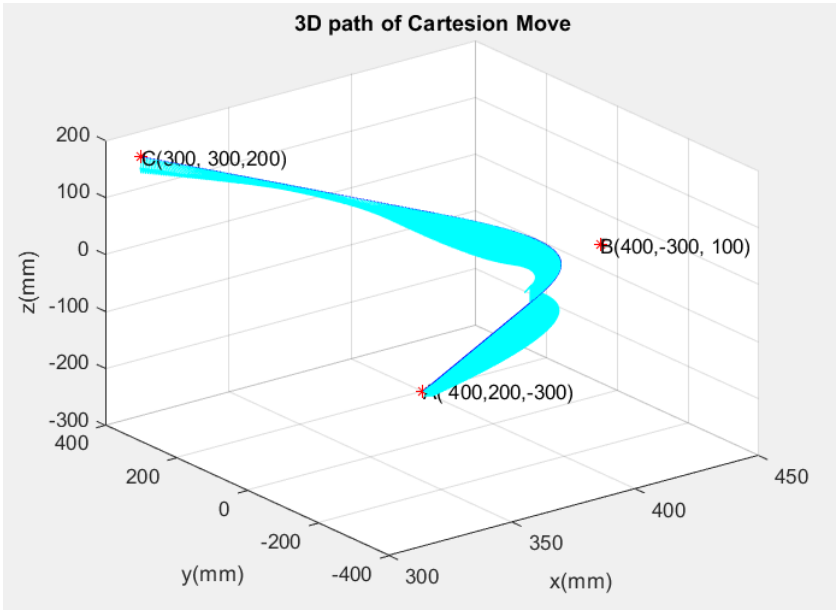


Angle acceleration

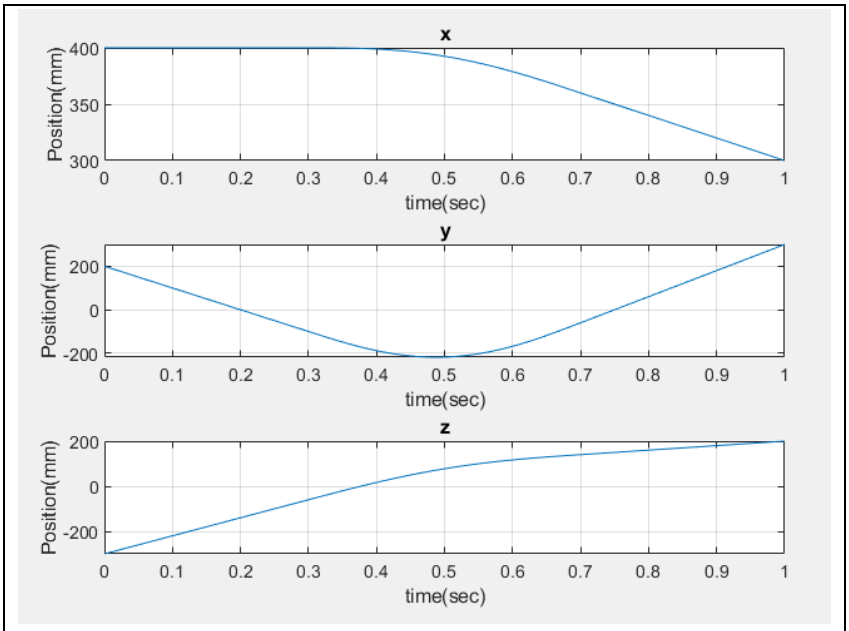
	joint1	joint2	joint3	joint4	joint5	joint6	if out of range
1	26.565100	72.647600	-58.992900	-13.654600	-153.434900	90.000000	
2	26.311360	72.460208	-59.073835	-13.386273	-153.328640	90.360000	
3	26.057620	72.272816	-59.154770	-13.117946	-153.222380	90.720000	
4	25.803880	72.085424	-59.235706	-12.849618	-153.116120	91.080000	
5	25.550140	71.898032	-59.316641	-12.581291	-153.009860	91.440000	
6	25.296400	71.710640	-59.397576	-12.312964	-152.903600	91.800000	
7	25.042660	71.523248	-59.478511	-12.044637	-152.797340	92.160000	
8	24.788920	71.335856	-59.559446	-11.776310	-152.691080	92.520000	
9	24.535180	71.148464	-59.640382	-11.507982	-152.584820	92.880000	
10	24.281440	70.961072	-59.721317	-11.239655	-152.478560	93.240000	
11	24.027700	70.773680	-59.802252	-10.971328	-152.372300	93.600000	
12	23.773960	70.586288	-59.883187	-10.703001	-152.266040	93.960000	
13	23.520220	70.398896	-59.964122	-10.434674	-152.159780	94.320000	
14	23.266480	70.211504	-60.045058	-10.166346	-152.053520	94.680000	
15	23.012740	70.024112	-60.125993	-9.898019	-151.947260	95.040000	
16	22.759000	69.836720	-60.206928	-9.629692	-151.841000	95.400000	

Txt(Joint)

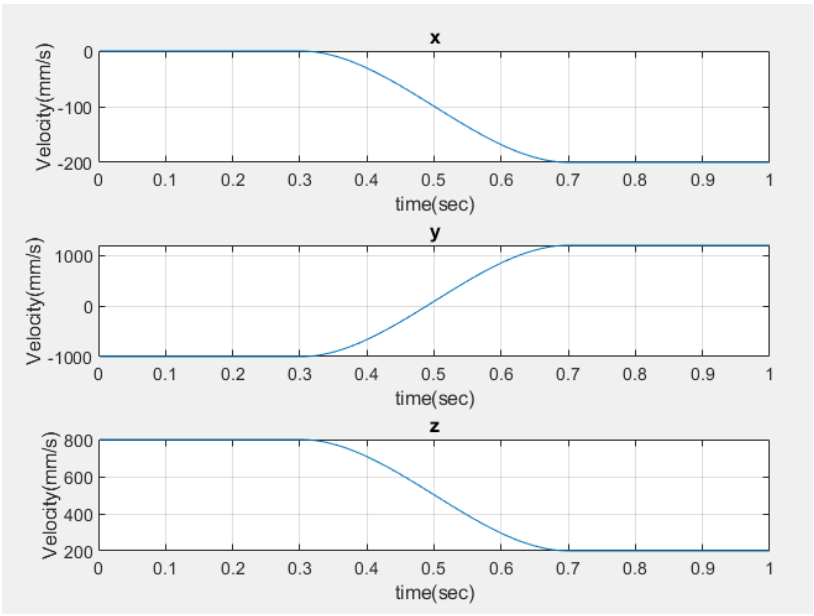
Cartesian space:



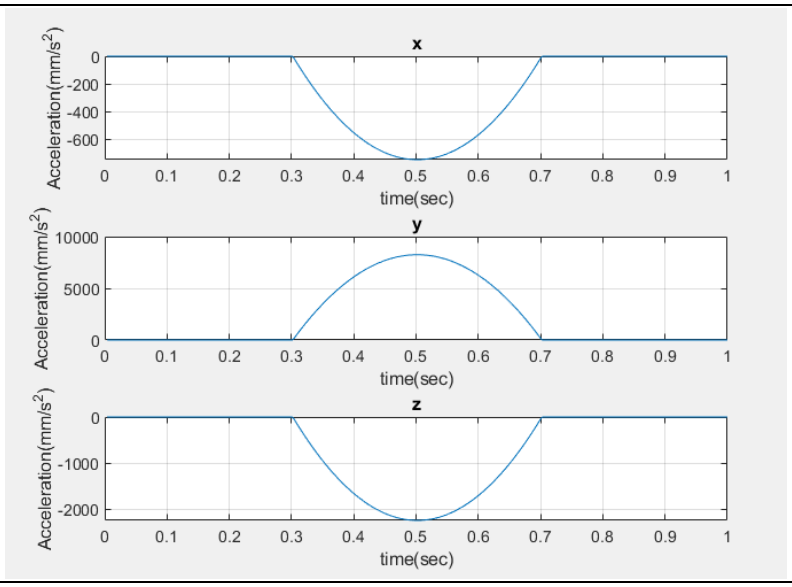
3D path



Position



Velocity



Acceleration

	joint1	joint2	joint3	joint4	joint5	joint6
1	26.565051	72.647557	-58.992911	-13.654647	-153.434949	90.000000
2	26.335410	72.977229	-59.781382	-13.195847	-154.744590	90.000000
3	26.104854	73.299585	-60.557817	-12.741768	-156.055146	90.000000
4	25.873385	73.614826	-61.322641	-12.292185	-157.366615	90.000000
5	25.641006	73.923137	-62.076248	-11.846889	-158.678994	90.000000
6	25.407718	74.224691	-62.819008	-11.405683	-159.992282	90.000000
7	25.173525	74.519650	-63.551268	-10.968382	-161.306475	90.000000
8	24.938428	74.808163	-64.273354	-10.534809	-162.621572	90.000000

txt(Cartesian)

三、 數學運算說明、推導

Six Boundary Conditions

$q_A(t^+) = q_A(t^-)$, $q_B(t^+) = q_B(t^-)$

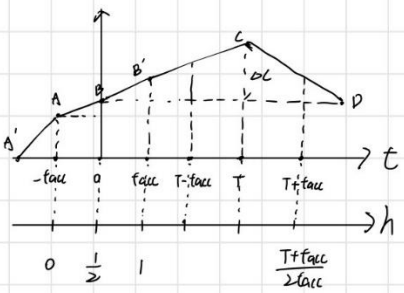
$\dot{q}_A(t^+) = \dot{q}_A(t^-)$, $\dot{q}_B(t^+) = \dot{q}_B(t^-)$

$\ddot{q}_A(t^+) = \ddot{q}_A(t^-)$, $\ddot{q}_B(t^+) = \ddot{q}_B(t^-)$

Symmetry in acceleration can reduce one restriction to a 4^{th} order which is enough.

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

令 $h(t) = \frac{t + t_{acc}}{2t_{acc}}$, $-t_{acc} \le t \le t_{acc}$



$$\begin{cases} q(h) = a_4h^4 + a_3h^3 + a_2h^2 + a_1h + a_0 \\ \dot{q}(h) = 4a_4h^3 + 3a_3h^2 + 2a_2h + a_1 \\ \ddot{q}(h) = 12a_4h + 6a_3h + 2a_2 \end{cases}$$

$$\begin{cases} q(0) = a_0 = A & a_0 = B + \Delta B \\ \dot{q}(0) = a_1 = -\Delta B / \frac{1}{2} \Rightarrow & a_1 = -2\Delta B \\ \ddot{q}(0) = 2a_2 = 0 & a_2 = 0 \end{cases}$$

$$\begin{cases} \dot{q}(1) = 4a_4 + 3a_3 + 2a_2 + a_1 = \frac{\Delta C}{\frac{T+t_{acc}}{2t_{acc}} - \frac{1}{2}} \\ \ddot{q}(1) = 12a_4 + 6a_3 = 0 \end{cases} \Rightarrow \begin{cases} 4a_4 + 3a_3 + a_1 = \frac{2\Delta C t_{acc}}{T} \\ 12a_4 + 6a_3 = 0 \end{cases} \Rightarrow \begin{cases} a_3 = 2\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right) \\ a_4 = -\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right) \end{cases}$$

position:

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

velocity:

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

acceleration:

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

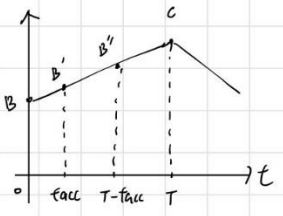
acceleration profile:

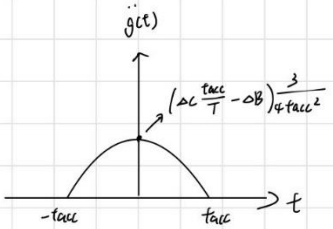
From B' to B'', set $h = \frac{t}{T}$

then $g(t) = \Delta C \cdot h + B$ ... position

$\dot{g}(t) = \frac{\Delta C}{T}$ ... velocity

$\ddot{g}(t) = 0$ ... acceleration





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 Motion 和 Cartesian Motion 的優缺點

1. Joint Motion

定義:

- 控制機器人的每個關節以其獨立的角速度運動。
- 通常基於機器人的關節空間（Joint Space）進行規劃。

優點:

1. 計算簡單
    - 關節運動只需要在機器人的關節空間內規劃，避免了逆運動學的複雜性。
    - 對於大多數機器人控制器，這是一種高效的運動模式。
2. 不受奇異點影響
    - 因為運動只涉及關節角的變化，奇異點不會直接影響軌跡規劃。
3. 穩定性高
    - 在關節空間內，運動軌跡通常較為穩定。

缺點:

1. 路徑不直觀
    - 關節空間的運動可能導致在笛卡爾空間中的軌跡變得不平滑或不可預測。
2. 路徑非最短
    - 由於不考慮工作空間中的直線運動，可能導致不必要的運動。

2. Cartesian Motion

定義:

- 控制機器人的末端執行器（End-Effector）在笛卡爾空間中按指定的路徑運動。

- 通常需要解逆運動學（Inverse Kinematics）來實現。

優點:

1. 路徑直觀
    - 可以直接控制末端執行器的運動軌跡，例如直線或特定形狀，便於規劃和監控。
2. 精確性高
    - 適合需要高精度的操作，比如焊接、塗膠或裝配。

缺點:

1. 受奇異點影響
    - 奇異點可能導致逆運動學無解或速度變得無窮大，運動不穩定。
2. 計算量大
    - 涉及逆運動學和雅可比矩陣的計算，對控制器性能要求高。
3. 潛在的穩定性問題
    - 在接近奇異點時，運動可能變得不平滑。

遇到奇異點的處理方法:

奇異點的概念:

奇異點是機器人在某些特定的關節配置下出現的問題，例如自由度的喪失、運動方向的無法控制，或操作空間中的無限速率需求（運動學雅可比矩陣的行列式為零）。可能導致機械臂無法正常運行或控制失效。

處理奇異點的方法

1.避免奇異點:

規劃路徑時避開奇異點：在運動規劃階段，通過分析工作空間，設計不經過奇異點的軌跡。或是在接近奇異點時，切換到 Joint Motion。

2.運動學優化：

使用運動學優化算法，使機械手臂在接近奇異點時更加平

滑地轉移，減少機械手臂運動的不連續性。

3.教導性運動：

使用教導的方式來規劃路徑，即手動操作機械手臂使其遠離奇異點，然後再切換到自動運動模式。

4.增加自由度：

在某些應用中，可以考慮使用具有更多自由度的機械手臂或是採用虛擬關節的方式，來減少奇異點的出現機會。通常具有冗餘自由度的機械手臂可以更靈活地避免奇異點。