1.介面說明：

開發平台：Matlab Script

如何執行：分為 P2_Joint 和 P2_Cartesian 兩個腳本，在 matlab 介面按下 run

2.程式架構說明：

Cartesian move

1. 找到開始加速的點 Ap，使用 D1 和 rotation 兩個 function 找 B→Ap 和 B→的運動矩陣，以及對應的參數 delta B 和 delta C
2. 用公式計算每個 sampling time 的 position, velocity 和 accleration
3. 呼叫 inverse kinematics 函數來計算每個 joint 應該有的角度
4. Plot，把所有結果繪製成對時間的曲線圖

```
for t = -T:Ts:T
    if t < -Tacc
        %加速段：從A到B起始過渡
        h = (t+0.5)/T;
        Dr = rotation(pose_a); %旋轉矩陣
        Dv = [pose_a(1)/T;pose_a(2)/T;pose_a(3)/T;0];
        joint = inverse_kinematics(A*Dr); %逆向運動學計算角度
        j_set = [j_set,joint']; %儲存關節角度數據
        %更新位置、速度，加速度為0
        p = [p,A(1:3,:)*Dr(:,4)];
        v = [v,A(1:3,:)*Dv];
        a = [a,zeros(3,1)];
        tempX = A*Dr*[0.1 0 0 1]';
        tempZ = A*Dr*[0 0 0.1 1]';
        Xaxis = [Xaxis tempX(1:3,1)];
        Zaxis = [Zaxis tempZ(1:3,1)];
```

```
elseif  Tacc>=t && t>=-Tacc
    % 過渡段：從B到C的過渡
    h = (t+Tacc)/(2*Tacc);
    Dp = [((XC*Tacc/T+XA)*(2-h)*h^2-2*XA)*h + XA;
        ((YC*Tacc/T+YA)*(2-h)*h^2-2*YA)*h + YA;
        ((ZC*Tacc/T+ZA)*(2-h)*h^2-2*ZA)*h + ZA;(psiC-psiA)*h+psiA;
        ((thetaC*Tacc/T+thetaA)*(2-h)*h^2-2*thetaA)*h + thetaA;
        ((phiC*Tacc/T+phiA)*(2-h)*h^2-2*phiA)*h + phiA];
    Dv = [((XC*Tacc/T+XA)*(1.5-h)*2*h^2-XA)/Tacc;
        ((YC*Tacc/T+YA)*(1.5-h)*2*h^2-YA)/Tacc;
        ((ZC*Tacc/T+ZA)*(1.5-h)*2*h^2-ZA)/Tacc;
        0];
    Da = [(XC*Tacc/T+XA)*(1-h)*3*h/Tacc^2;
        (YC*Tacc/T+YA)*(1-h)*3*h/Tacc^2;
        (ZC*Tacc/T+ZA)*(1-h)*3*h/Tacc^2;0];
    Dr = rotation([Dp(1),Dp(2),Dp(3),Dp(4),Dp(5),Dp(6),1]);
    joint = inverse_kinematics(B*Dr);
    j_set = [j_set,joint'];
    p = [p,B(1:3,:)*Dr(:,4)];
    v = [v,B(1:3,:)*Dv];
    a = [a,B(1:3,:)*Da];
    tempZ = B*Dr*[0 0 0.1 1]';
    tempX = B*Dr*[0.1 0 0 1]';
    Zaxis = [Zaxis tempZ(1:3,1)];
    Xaxis = [Xaxis tempX(1:3,1)];

elseif t > Tacc
    % 減速段：完成從B到C的過渡
    h = t/T;
    Dr = rotation([XC,YC,ZC,psiC,thetaC,phiC,h]);
    Dv = [delta_C(1)/T;delta_C(2)/T;delta_C(3)/T;0];
    joint = inverse_kinematics(B*Dr);
    j_set = [j_set,joint'];
    p = [p,B(1:3,:)*Dr(:,4)];
    v = [v,B(1:3,:)*Dv];
    a = [a,zeros(3,1)];
    tempZ = B*Dr*[0 0 0.1 1]';
    tempX = B*Dr*[0.1 0 0 1]';
    Zaxis = [Zaxis tempZ(1:3,1)];
    Xaxis = [Xaxis tempX(1:3,1)];
end
```

程式中使用到 inverse_kinematics 是以 project 1 的部分直接拿來用

Joint move 同樣算出 ABC 三個 frame 的角度 theta_A, theta_B, theta_C，並利用公式和 project1 的 forward kinematics 函數計算每個 sampling time 的 position, velocity, accleration，最後繪製結果

```
function [theta_p, theta_v, theta_a] = compute_theta(t, T, Tacc, theta_A, theta_B, theta_C, r)
    if t < -Tacc
        theta_p = (theta_B - theta_A) * (t + 0.5) / T + theta_A;
        theta_v = (theta_B - theta_A) / T;
        theta_a = zeros(size(theta_A));
    elseif t <= Tacc && t >= -Tacc
        k = (t + Tacc) / (2 * Tacc);
        common_term = (theta_C - theta_B) * Tacc / T + ((theta_A + (theta_B - theta_A) * r) - theta_B);
        theta_p = (common_term * (2 - k) * k^2 - 2 * ((theta_A + (theta_B - theta_A) * r) - theta_B)) * k ...
                + ((theta_A + (theta_B - theta_A) * r) - theta_B) + theta_B;
        theta_v = (common_term * (1.5 - k) * 2 * k^2 - ((theta_A + (theta_B - theta_A) * r) - theta_B)) / Tacc;
        theta_a = (common_term * (1 - k) * 3 * k / Tacc^2);
    else
        theta_p = (theta_C - theta_B) * t / T + theta_B;
        theta_v = (theta_C - theta_B) / T;
        theta_a = zeros(size(theta_A));
    end
end
```

3.數學運算說明：

Transition portion:

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

$$\begin{cases} q(t) = a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \\ \dot{q}(t) = 4a_4 t^3 + 3a_3 t^2 + 2a_2 t + a_1 \\ \ddot{q}(t) = 12a_4 t^2 + 6a_3 t + 2a_2 \end{cases}$$

在邊界條件下求多項式 p,v,a 的各個參數

Boundary Conditions:

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

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

$$\Rightarrow a_4 = -\left(\Delta C \frac{t_{acc}}{T} + \Delta B\right)$$

$$a_3 = -2a_4 = 2\left(\Delta B + \Delta C \frac{t_{acc}}{T}\right)$$

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

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

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

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

定義兩個 function，D1 和 rotation，分別對應到兩個點之間的轉移矩陣&運動矩陣，轉移矩陣只包含平移，運動矩陣包含平移以及轉動

由 POS1 移動到 POS2

$$\text{Let}\quad POS1 = \begin{pmatrix} {}^1n & {}^1o & {}^1a & {}^1p \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$POS2 = \begin{pmatrix} {}^2n & {}^2o & {}^2a & {}^2p \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$D(1) = POS1^{-1} * POS2$$

$$= \begin{pmatrix} {}^1n\cdot{}^2n & {}^1n\cdot{}^2o & {}^1n\cdot{}^2a & {}^1n\cdot({}^2p-{}^1p) \\ {}^1o\cdot{}^2n & {}^1o\cdot{}^2o & {}^1o\cdot{}^2a & {}^1o\cdot({}^2p-{}^1p) \\ {}^1a\cdot{}^2n & {}^1a\cdot{}^2o & {}^1a\cdot{}^2a & {}^1a\cdot({}^2p-{}^1p) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Dr 由 Tr, Ra 和 Ro 相乘得到，Tr 代表平移，Ra 是對 a 軸轉 rθ，Ro 是對 o 軸旋轉 ϕ，兩次旋轉和一次平移。

$$D(r)=\begin{pmatrix} n_x & (\ldots) & C\psi S(r\theta) & r_x \\ n_y & (\ldots) & S\psi S(r\theta) & r_y \\ n_z & (\ldots) & C(r\theta) & r_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$D(r)=T_r(r)*Ra(r)*Ro(r)$$

$$T_r(r)=\begin{pmatrix} 1 & 0 & 0 & r_x \\ 0 & 1 & 0 & r_y \\ 0 & 0 & 1 & r_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Where x, y, z are the distances between POS1 and POS2

$$Ra(r)=\begin{pmatrix} S\psi^2 V(r\theta)+C(r\theta) & -S\psi C\psi V(r\theta) & C\psi S(r\theta) & 0 \\ -S\psi C\psi V(r\theta) & C\psi^2 V(r\theta)+C(r\theta) & S\psi S(r\theta) & 0 \\ -C\psi S(r\theta) & -S\psi S(r\theta) & C(r\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $V(r\theta)=Vers(r\theta)=1-\cos(r\theta)$

$$Ro(r)=\begin{pmatrix} C(r\phi) & -S(r\phi) & 0 & 0 \\ S(r\phi) & C(r\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

藉由以下公式可以得到規畫路徑所需的x、y、z、θ、ϕ、ψ

□ Solution of x, y, z

$$r=1,\quad D(1)=T(1)*Ra(1)*Ro(1)$$

$$T(1)=\begin{pmatrix}1 & 0 & 0 & x\\ 0 & 1 & 0 & y\\ 0 & 0 & 1 & z\\ 0 & 0 & 0 & 1\end{pmatrix}=D(1)*Ro(1)^{-1}*Ra(1)^{-1}$$

$$x={}^{1}n\cdot({}^{2}p-{}^{1}p)$$

$$y={}^{1}o\cdot({}^{2}p-{}^{1}p)$$

$$z={}^{1}a\cdot({}^{2}p-{}^{1}p)$$

□ Solution of ψ

$$Ra(1)=T(1)^{-1}*D(1)*Ro(1)^{-1}$$

Third column $C\psi S\theta={}^{1}n\cdot{}^{2}a$

$$S\psi S\theta={}^{1}o\cdot{}^{2}a$$

$$C\theta={}^{1}a\cdot{}^{2}a$$

$$\therefore\psi=\tan^{-1}\left(\frac{{}^{1}o\cdot{}^{2}a}{{}^{1}n\cdot{}^{2}a}\right)$$

Where $sin\theta>0$

i.e. $0°\le\theta\le180°$

□ Solution of θ

$$\tan\theta = \frac{[({}^1n \cdot {}^2a)^2 + ({}^1o \cdot {}^2a)^2]^{\frac{1}{2}}}{{}^1a \cdot {}^2a}, 0^o \le \theta \le 180^o$$

□ Solution of ϕ

$$\begin{pmatrix} C\phi & -S\phi & 0 & 0 \\ S\phi & C\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = Ro(1) = Ra(1)^{-1} * T(1)^{-1} * D(1)$$

$$= \begin{pmatrix} (S\psi)^2 V\theta + C\theta & -S\psi C\psi V\theta & -C\psi S\theta & 0 \\ -S\psi C\psi V\theta & (C\psi)^2 V\theta + C\theta & -S\psi S\theta & 0 \\ C\psi S\theta & S\psi S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} * T(1)^{-1} * D(1)$$

$$\therefore S\phi = -S\psi C\psi V\theta({}^1n \cdot {}^2n) + [(C\psi)^2 V\theta + C\theta]({}^1o \cdot {}^2n) - S\psi S\theta({}^1a \cdot {}^2n)$$

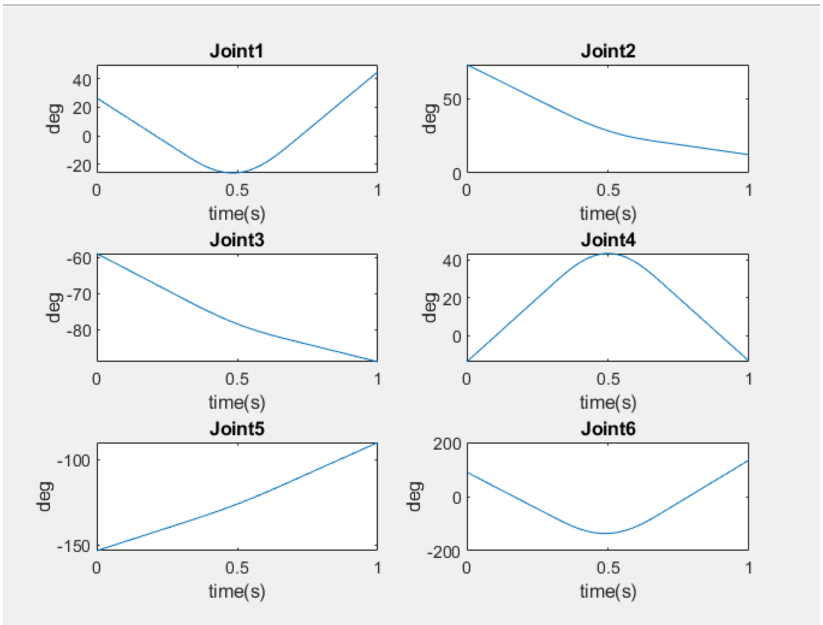
$$C\phi = -S\psi C\psi V\theta({}^1n \cdot {}^2o) + [(C\psi)^2 V\theta + C\theta]({}^1o \cdot {}^2o) - S\psi S\theta({}^1a \cdot {}^2o)$$

$$\therefore \tan\phi = \frac{S\phi}{C\phi}, -\pi \le \phi \le \pi$$

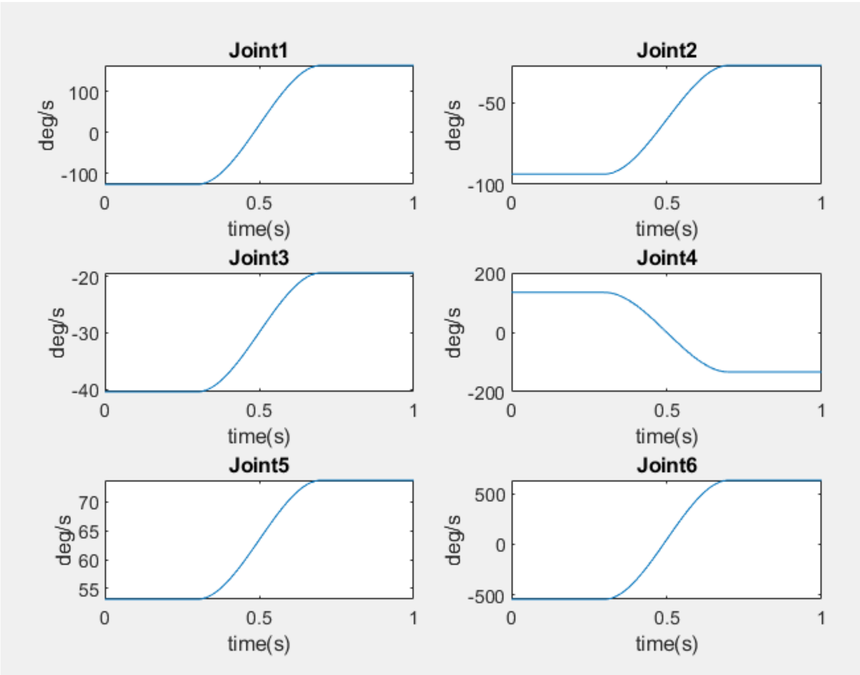
4.軌跡規劃曲線圖結果：

Part1.Joint move

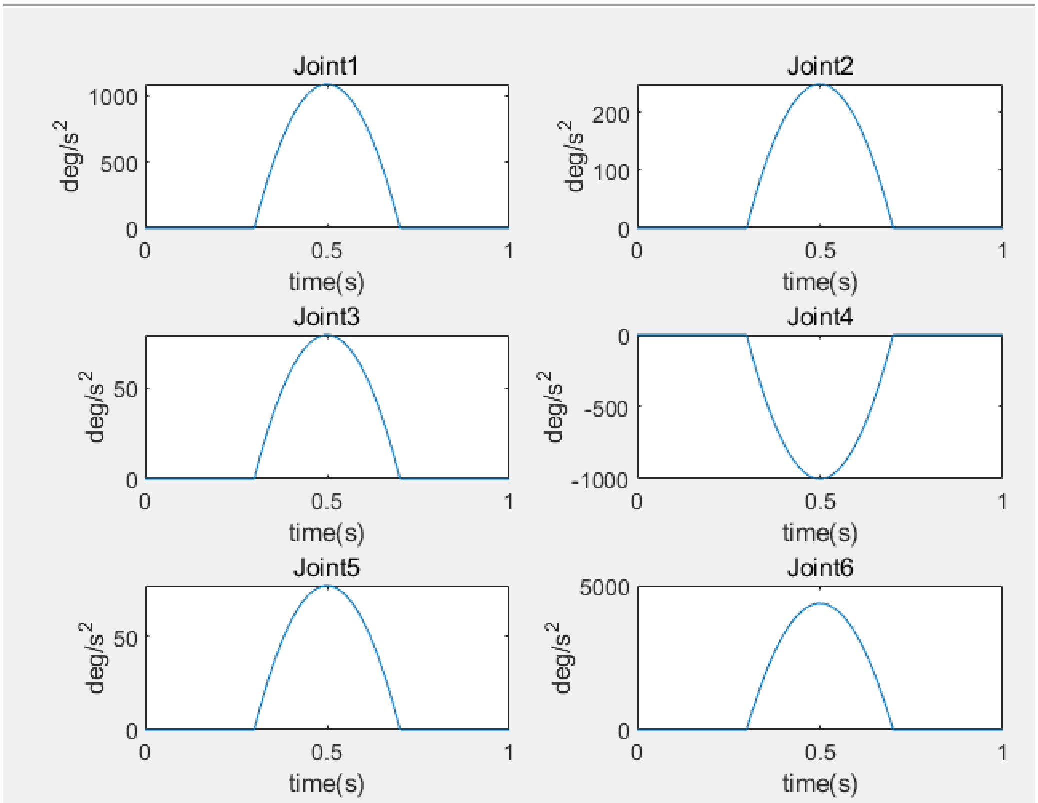
Angle



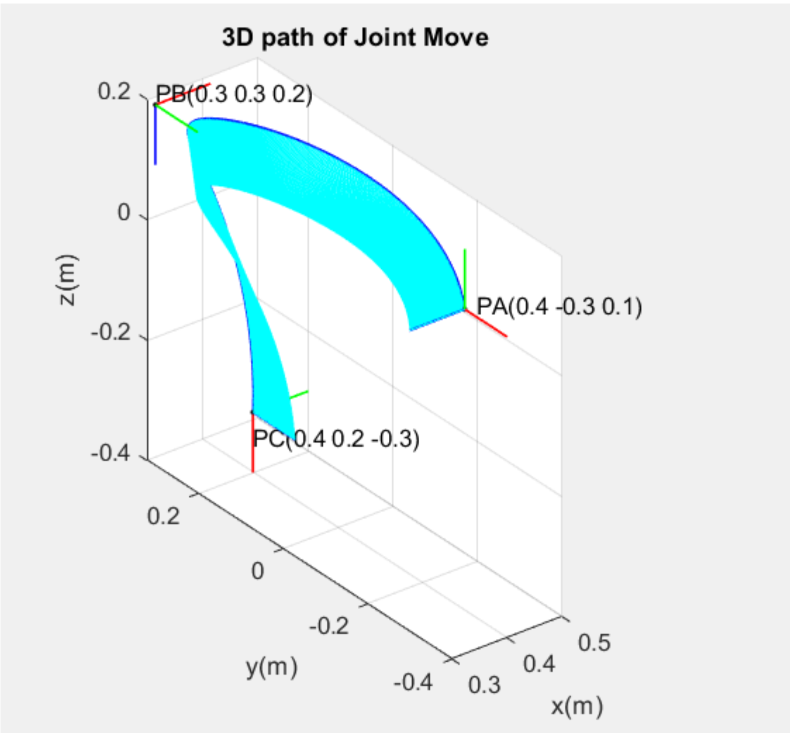
Angular Velocity



Angular Acceleration

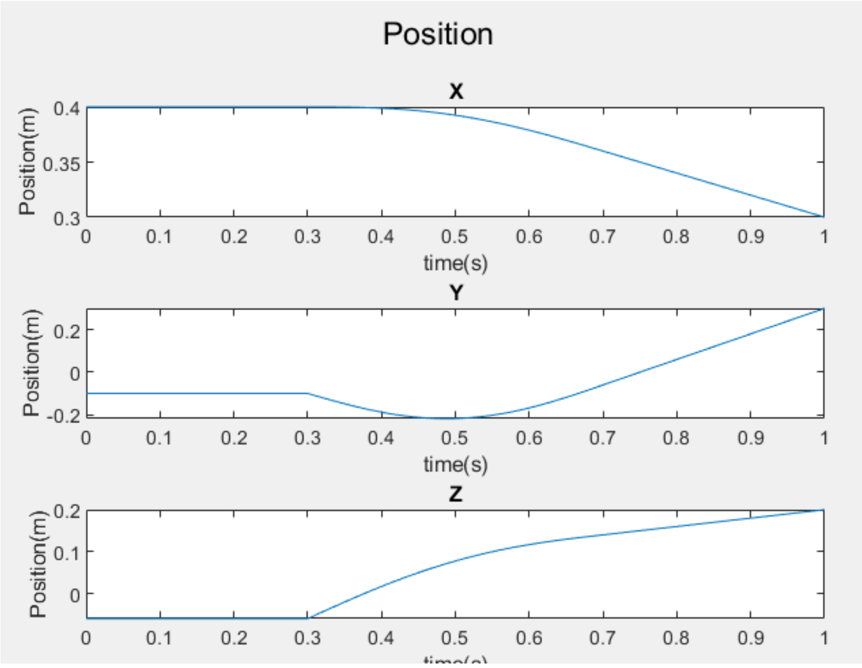


3D path

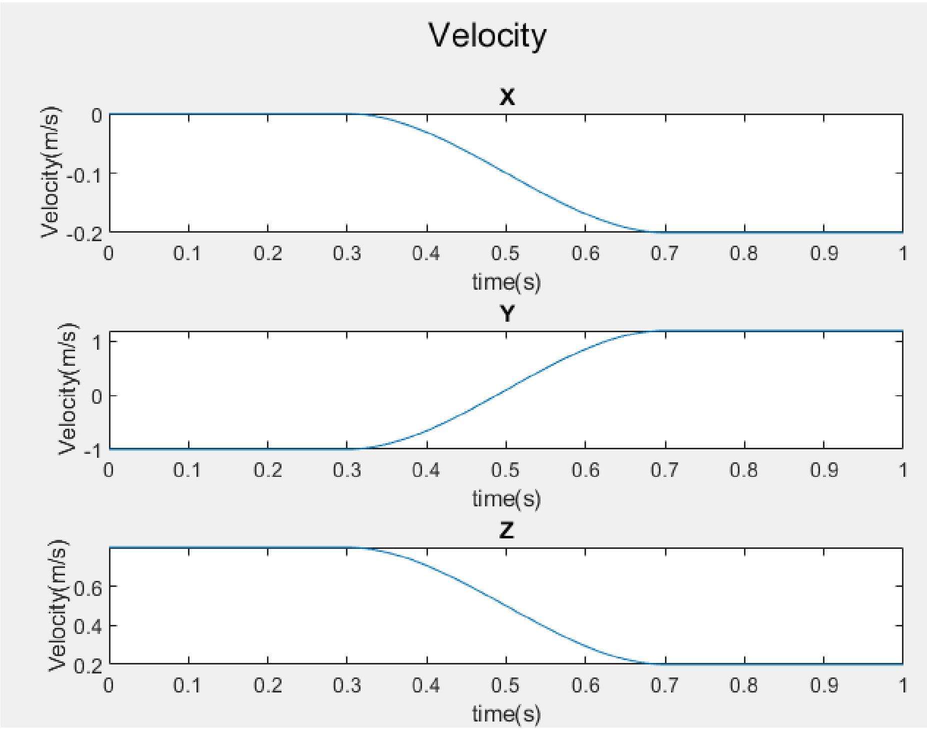


Part2.Cartesian move:

Position

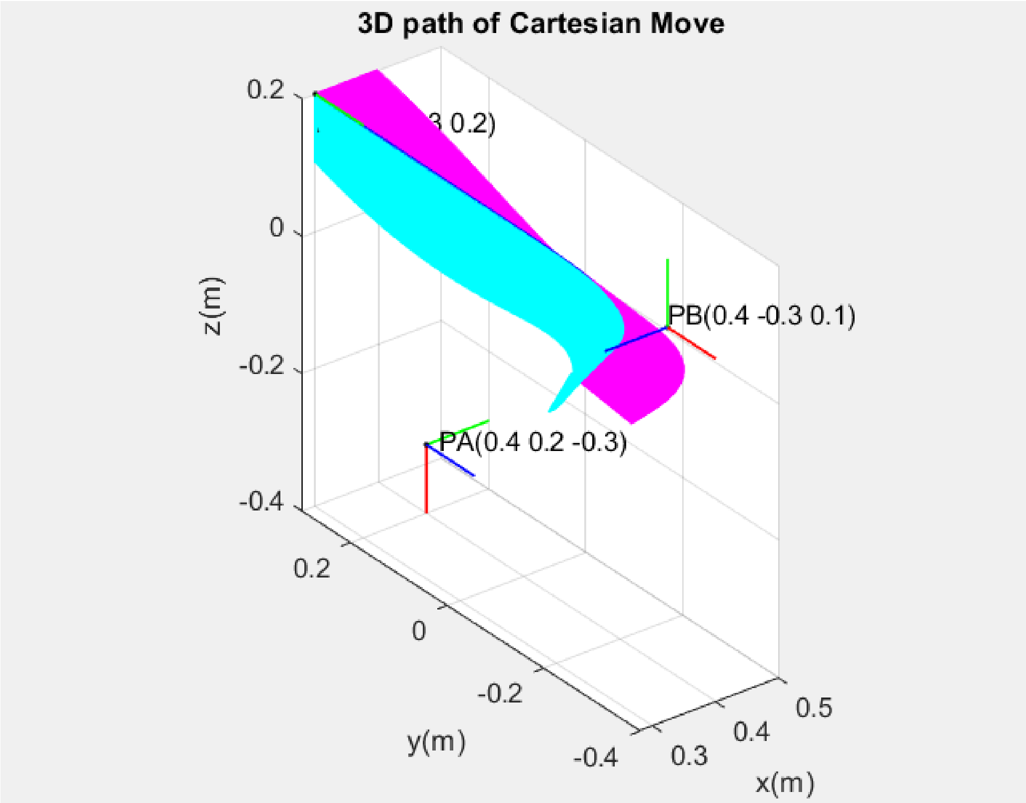


Velocity



較為直觀：基於末端位置與方向進行路徑規劃，與操作空間需求直接相關。

3D path



5.加分題

(1) 討論兩種軌跡規劃的優缺點

Cartesian Move:

優點：

精確性高：能直接規劃 end effector 在三維空間的運動軌跡，適合需要精確追

跡的應用場合。

較為直觀：基於末端位置與方向進行路徑規劃，與操作空間需求直接相關。

容易實現避障：在 Cartesian 空間可以直接設定不可進入之區域，限制機械手

臂末端不進入或經過特定位置。

缺點：

較高的計算成本：需解 inverse kinematic 來將 Cartesian 軌跡轉換為 Joint angle，計算上較複雜。

受 Singular point 影響較大：當機器人遇到 singular point(e.g.特定關節排列導致的自由度喪失)，可能無法計算求解。

依賴模型準確性：對運動學模型精確度要求較高，模型誤差較易使軌跡偏差。

Joint Move

優點：

計算較簡便：只須規劃每一個關節角度的變化，不需要使用逆向運動學，運算較簡單迅速。

穩定性高：因直接規劃 joint angle，在通常情況下不受 singular point 影響。

缺點：

末端軌跡不平滑：在 joint space 中設計的關節軌跡可能讓末端(end effector)的軌跡不連續、平滑。

避障性差：無法直接規劃末端的避障路徑，需間接透過關節調整實現。

(2)奇異點處理：

Cartesian move:

1.在規劃軌跡時避免直接穿過奇異點，透過插植多項式或增加中間過渡點進行奇異點的迴避。

2.速度限制：在接近奇異點時限制關節速度，降低奇異點帶來的不穩定性。

3.冗餘自由度：對於具有 redundant 的機器人，可以利用多個解的特性，選擇避免解出奇異點的組合。

Joint move:

1.直接避開：在 joint 軌跡規劃中通常奇異點會在關節運動範圍外，可以調整運動範圍避開奇異點。

2.增加關節限制：在奇異點附近設定運動範圍限制，避免進入不穩定區域。