311605012 機器人碩士碩一 楊曜瑋

一、 介面說明

開發平台: matlab

如何執行:

1. 選擇相對應的 cartesian_motion_311605012.m 及 joint_motion_311605012.m 程式
2. 按下 run 鍵執行程式

二、 程式架構說明

- joint_motion_311605012.m

I. 程式運行流程

- 先將題目給予的 A、B、C 三點換算成 inch，並代入 project 1 的 inverse kinematic 程式去計算出 thetaA、thetaB、thetaC 各自的角度，從中挑出合適的一組解
- 計算出 deltaB 與 deltaC
- 根據時間切分為三段(-0.5~-0.2s、-0.2~0.2s、0.2~0.5s)，分別為加速、保持速度、以及減速，套用軌跡方程式去計算位置、速度與加速度
- 繪製出計算完的結果

II. 核心程式碼說明

```
%result of calculating inverse kinemetic
thetaA = [-70.166696084446997 -27.204412794324707 13.280020368960283 -107.785254677590729 81.076847119486033 64.194498589472474];
thetaB = [7.004109132849219 72.754870654120566 13.280020368960281 -0.000000000000000 -72.754870654120566 -7.004109132849219];
thetaC = [109.8332 27.2045 -13.2801 -22.0744 64.5293 9.8929];
```

- 透過 inverse kinematic 求出 thetaA、thetaB、thetaC 分別 8 組結果，從中挑選出適合的解

```
for t = -0.5:0.002:0.5
    if t < -0.2
        t = t + 0.5;
        linear_h = t/T;
        theta_p = deltaB*linear_h + thetaA;
        endpointA = theta_p;
        [temp1, temp2, temp3, temp4, temp5, temp6] = forward(theta_p);
        ini_px = [ini_px temp1* 2.54];
        ini_py = [ini_py temp2* 2.54];
        ini_pz = [ini_pz temp3* 2.54];
        end_px = [end_px temp4* 2.54];
        end_py = [end_py temp5* 2.54];
        end_pz = [end_pz temp6* 2.54];

        joint1 = [joint1 theta_p(1)];
        joint2 = [joint2 theta_p(2)];
        joint3 = [joint3 theta_p(3)* 2.54];
        joint4 = [joint4 theta_p(4)];
        joint5 = [joint5 theta_p(5)];
        joint6 = [joint6 theta_p(6)];

        theta_v = deltaB/T;
        v_joint1 = [v_joint1 theta_v(1)];
        v_joint2 = [v_joint2 theta_v(2)];
        v_joint3 = [v_joint3 theta_v(3)* 2.54];
        v_joint4 = [v_joint4 theta_v(4)];
        v_joint5 = [v_joint5 theta_v(5)];
        v_joint6 = [v_joint6 theta_v(6)];

        theta_a = zeros(6,1);
        a_joint1 = [a_joint1 theta_a(1)];
        a_joint2 = [a_joint2 theta_a(2)];
        a_joint3 = [a_joint3 theta_a(3)* 2.54];
        a_joint4 = [a_joint4 theta_a(4)];
        a_joint5 = [a_joint5 theta_a(5)];
        a_joint6 = [a_joint6 theta_a(6)];
```

```
    elseif t<=0.2 && t>=-0.2
        deltaB = endpointA - thetaB;
        h = (t + Tacc) / (2 * Tacc);
        theta_p = ((deltaC * (Tacc/T) + deltaB)*(2-h)*(h^2) - (2 * deltaB)) * h + endpointA;
        endpointB = theta_p;
        [temp1, temp2, temp3, temp4, temp5, temp6] = forward(theta_p);
        ini_px = [ini_px temp1* 2.54];
        ini_py = [ini_py temp2* 2.54];
        ini_pz = [ini_pz temp3* 2.54];
        end_px = [end_px temp4* 2.54];
        end_py = [end_py temp5* 2.54];
        end_pz = [end_pz temp6* 2.54];

        joint1 = [joint1 theta_p(1)];
        joint2 = [joint2 theta_p(2)];
        joint3 = [joint3 theta_p(3)* 2.54];
        joint4 = [joint4 theta_p(4)];
        joint5 = [joint5 theta_p(5)];
        joint6 = [joint6 theta_p(6)];

        theta_v = ((deltaC * (Tacc/T) + deltaB)*(1.5-h)*2*(h^2) - deltaB)/Tacc;
        v_joint1 = [v_joint1 theta_v(1)];
        v_joint2 = [v_joint2 theta_v(2)];
        v_joint3 = [v_joint3 theta_v(3)* 2.54];
        v_joint4 = [v_joint4 theta_v(4)];
        v_joint5 = [v_joint5 theta_v(5)];
        v_joint6 = [v_joint6 theta_v(6)];

        theta_a = ((deltaC * (Tacc/T) + deltaB)*(1-h))*(3*h)/(Tacc^2);
        a_joint1 = [a_joint1 theta_a(1)];
        a_joint2 = [a_joint2 theta_a(2)];
        a_joint3 = [a_joint3 theta_a(3)* 2.54];
        a_joint4 = [a_joint4 theta_a(4)];
        a_joint5 = [a_joint5 theta_a(5)];
        a_joint6 = [a_joint6 theta_a(6)];
```

```
        elseif t > 0.2
            t = (t-0.2);
            linear_h = t/T;
            theta_p = deltaC*linear_h + endpointB;
            [temp1, temp2, temp3, temp4, temp5, temp6] = forward(theta_p);
            ini_px = [ini_px temp1* 2.54];
            ini_py = [ini_py temp2* 2.54];
            ini_pz = [ini_pz temp3* 2.54];
            end_px = [end_px temp4* 2.54];
            end_py = [end_py temp5* 2.54];
            end_pz = [end_pz temp6* 2.54];

            joint1 = [joint1 theta_p(1)];
            joint2 = [joint2 theta_p(2)];
            joint3 = [joint3 theta_p(3)* 2.54];
            joint4 = [joint4 theta_p(4)];
            joint5 = [joint5 theta_p(5)];
            joint6 = [joint6 theta_p(6)];

            theta_v = deltaC/T;
            v_joint1 = [v_joint1 theta_v(1)];
            v_joint2 = [v_joint2 theta_v(2)];
            v_joint3 = [v_joint3 theta_v(3)* 2.54];
            v_joint4 = [v_joint4 theta_v(4)];
            v_joint5 = [v_joint5 theta_v(5)];
            v_joint6 = [v_joint6 theta_v(6)];

            theta_a = zeros(6,1);
            a_joint1 = [a_joint1 theta_a(1)];
            a_joint2 = [a_joint2 theta_a(2)];
            a_joint3 = [a_joint3 theta_a(3)* 2.54];
            a_joint4 = [a_joint4 theta_a(4)];
            a_joint5 = [a_joint5 theta_a(5)];
            a_joint6 = [a_joint6 theta_a(6)];
        end
    end
```

- 依據題目給予的時間，分成-0.5~-0.2、-0.2~0.2、0.2~0.5 三段，分別為加速段、維持速度段、以及減速段
- 加速段及減速段透過線性方程式去進行計算，維持速度段透過多項式方程式去計算，為保持 h 介於 0-1 之間，會將時間進行平移調整
- 每一段會首尾相連，因此在每一段結束位置會記錄，並作為下一段的起始位置
- 透過投影片的軌跡方程式進行計算，並利用 forward kinematic 求出 6 個 joint 的結果

311605012 機器人碩士碩一 楊曜瑋

- cartesian_motion_311605012.m

I. 程式運行流程

1. 從 A、B、C 三個矩陣取得各自的 x、y、z 位置，並計算出ϕ, θ, ψ
2. 計算出 deltaB 與 deltaC
3. 同樣會分成三段，加速、保持速度、以及減速，套用軌跡方程式去計算位置、速度與加速度
4. 繪製出計算完的結果

II. 核心程式碼說明

```
%calculate Φ,θ,ϕ of A,B,C
A_phi = atan2(A(2,3), A(1,3));
A_theta = atan2((cos(A_phi)*A(1,3)+sin(A_phi)*A(2,3)), A(3,3));
A_psi = atan2((-sin(A_phi)*A(1,1)+cos(A_phi)*A(2,1)), (-sin(A_phi)*A(1,2)+cos(A_phi)*A(2,2)));
thetaA = [A(1,4),A(2,4),A(3,4),A_phi,A_theta,A_psi];

B_phi = atan2(B(2,3), B(1,3));
B_theta = atan2((cos(B_phi)*B(1,3)+sin(B_phi)*B(2,3)), B(3,3));
B_psi = atan2((-sin(B_phi)*B(1,1)+cos(B_phi)*B(2,1)), (-sin(B_phi)*B(1,2)+cos(B_phi)*B(2,2)));
thetaB = [B(1,4),B(2,4),B(3,4),B_phi,B_theta,B_psi];

C_phi = atan2(C(2,3), C(1,3));
C_theta = atan2((cos(C_phi)*C(1,3)+sin(C_phi)*C(2,3)), C(3,3));
C_psi = atan2((-sin(C_phi)*C(1,1)+cos(C_phi)*C(2,1)), (-sin(C_phi)*C(1,2)+cos(C_phi)*C(2,2)));
thetaC = [C(1,4),C(2,4),C(3,4),C_phi,C_theta,C_psi];
```

- 取得 A、B、C 各自的$(x, y, z, \phi, \theta, \psi)$

```
%According to time to calculate the position, velocity, and acceleration
for t = -0.5:0.002:0.5
    if t < -0.2
        t = t + 0.5;
        linear_h = t/T;
        theta_p = deltaB*linear_h + thetaA;
        endpointA = theta_p;
        ini_px = [ini_px theta_p(1)];
        ini_py = [ini_py theta_p(2)];
        ini_pz = [ini_pz theta_p(3)];
        end_px = [end_px theta_p(4)+A(1,3)*2];
        end_py = [end_py theta_p(5)+A(2,3)*2];
        end_pz = [end_pz theta_p(6)+A(3,3)*2];

        theta_v = deltaB/T;
        v_px = [v_px theta_v(1)];
        v_py = [v_py theta_v(2)];
        v_pz = [v_pz theta_v(3)];

        theta_a = zeros(6,1);
        a_px = [a_px theta_a(1)];
        a_py = [a_py theta_a(2)];
        a_pz = [a_pz theta_a(3)];
```

```
    elseif t<=0.2 && t>=-0.2
        deltaB = endpointA - thetaB;
        h = (t + Tacc) / (2 * Tacc);
        theta_p = ((deltaC * (Tacc/T) + deltaB)*(2-h)*(h^2) - (2 * deltaB)) * h + endpointA;
        endpointB = theta_p;
        ini_px = [ini_px theta_p(1)];
        ini_py = [ini_py theta_p(2)];
        ini_pz = [ini_pz theta_p(3)];
        end_px = [end_px theta_p(4)+A(1,3)*2];
        end_py = [end_py theta_p(5)+A(2,3)*2];
        end_pz = [end_pz theta_p(6)+A(3,3)*2];

        theta_v = ((deltaC * (Tacc/T) + deltaB)*(1.5-h)*2*(h^2) - deltaB)/Tacc;
        v_px = [v_px theta_v(1)];
        v_py = [v_py theta_v(2)];
        v_pz = [v_pz theta_v(3)];

        theta_a = ((deltaC * (Tacc/T) + deltaB)*(1-h))*(3*h)/(Tacc^2);
        a_px = [a_px theta_a(1)];
        a_py = [a_py theta_a(2)];
        a_pz = [a_pz theta_a(3)];
```

```
    elseif t > 0.2
        t = (t-0.2);
        linear_h = t/T;
        theta_p = deltaC*linear_h + endpointB;
        ini_px = [ini_px theta_p(1)];
        ini_py = [ini_py theta_p(2)];
        ini_pz = [ini_pz theta_p(3)];
        end_px = [end_px theta_p(4)+A(1,3)*2];
        end_py = [end_py theta_p(5)+A(2,3)*2];
        end_pz = [end_pz theta_p(6)+A(3,3)*2];

        theta_v = deltaC/T;
        v_px = [v_px theta_v(1)];
        v_py = [v_py theta_v(2)];
        v_pz = [v_pz theta_v(3)];

        theta_a = zeros(6,1);
        a_px = [a_px theta_a(1)];
        a_py = [a_py theta_a(2)];
        a_pz = [a_pz theta_a(3)];
    end
end
```

- 依據題目給予的時間，分成-0.5~-0.2、-0.2~0.2、0.2~0.5 三段，分別為加速段、維持速度段、以及減速段
- 加速段及減速段透過線性方程式去進行計算，維持速度段透過多項式方程式去計算，為保持 h 介於 0-1 之間，會將時間進行平移調整
- 每一段會首尾相連，因此在每一段結束位置會記錄，並作為下一段的起始位置
- 透過投影片的軌跡方程式進行計算，得到最終結果

三、 數學運算說明

- 各段軌跡方程式

參考投影片提供的軌跡方程式去進行運算

- 加速、減速段

□ For linear portion

$$\begin{cases} q = \Delta C \cdot h + B \\ \dot{q} = \dfrac{\Delta C}{T} \\ \ddot{q} = 0 \end{cases} \qquad h = \frac{t}{T}, t_{acc} \le t \le T - t_{acc}$$

- 保持速度

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

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

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

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

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

311605012 機器人碩士碩一 楊曜瑋

- cartesian_motion 計算ϕ, θ, ψ的方式

  透過 Euler angle 推得結果

  - For Euler z-y-z
    a) Solution of ϕ:

       If $\theta \neq 0 \quad \Rightarrow \phi = \tan^{-1}\left[\frac{a_y}{a_x}\right]$ or $\Rightarrow \tan^{-1}\left[\frac{a_y}{a_x}\right] + 180^o$

    b) Solution of θ:

       If $c\theta = a_z \quad \Rightarrow s\theta = c\phi a_x + s\phi a_y = c\phi(c\phi s\theta) + s\phi(s\phi s\theta)$

       $$\theta = \tan^{-1}\left(\frac{s\theta}{c\theta}\right) = \tan^{-1}\left[\frac{c\phi a_x + s\phi a_y}{a_z}\right]$$

    c) Solution of ψ:

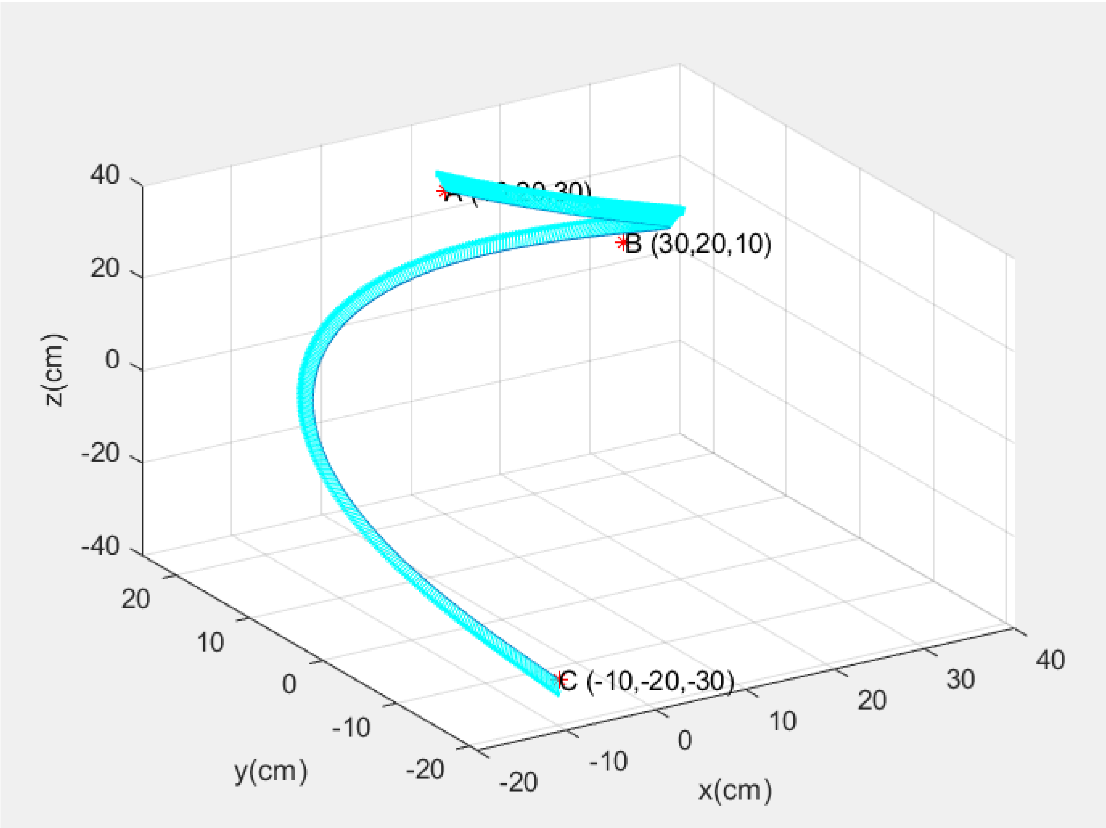
$$s\psi = -s\phi n_x + c\phi n_y = s\phi^2 s\psi + c\phi^2 s\psi$$
$$c\psi = -s\phi o_x + c\phi o_y$$

$$\psi = \tan^{-1}\frac{s\psi}{c\psi} = \tan^{-1}\left\{\frac{-s\phi n_x + c\phi n_y}{-s\phi o_x + c\phi o_y}\right\}$$

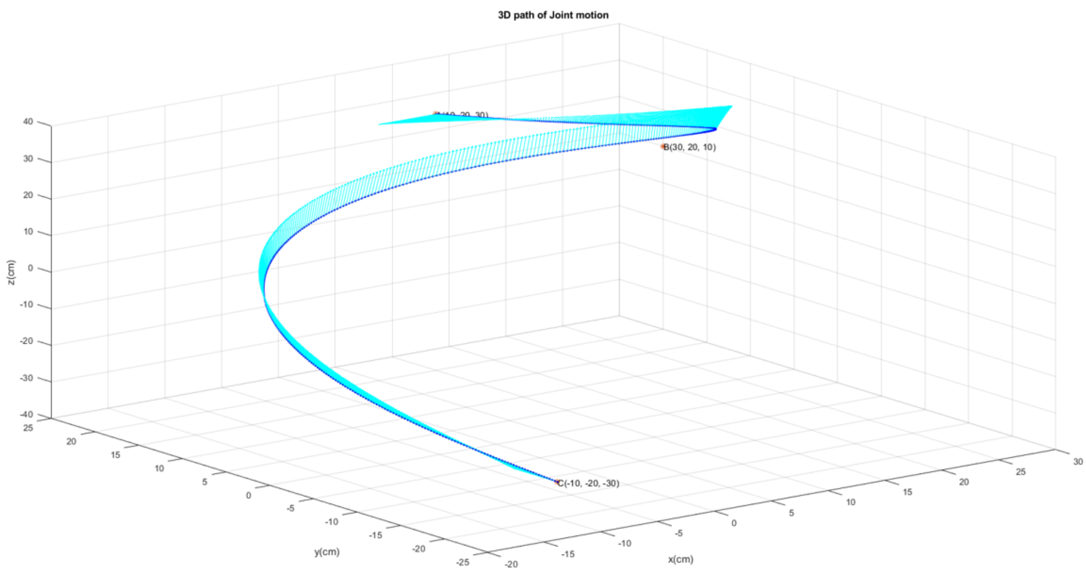
311605012 機器人碩士碩一 楊曜瑋

四、 軌跡規劃曲線圖結果

- **Joint move**
  - 3D plot



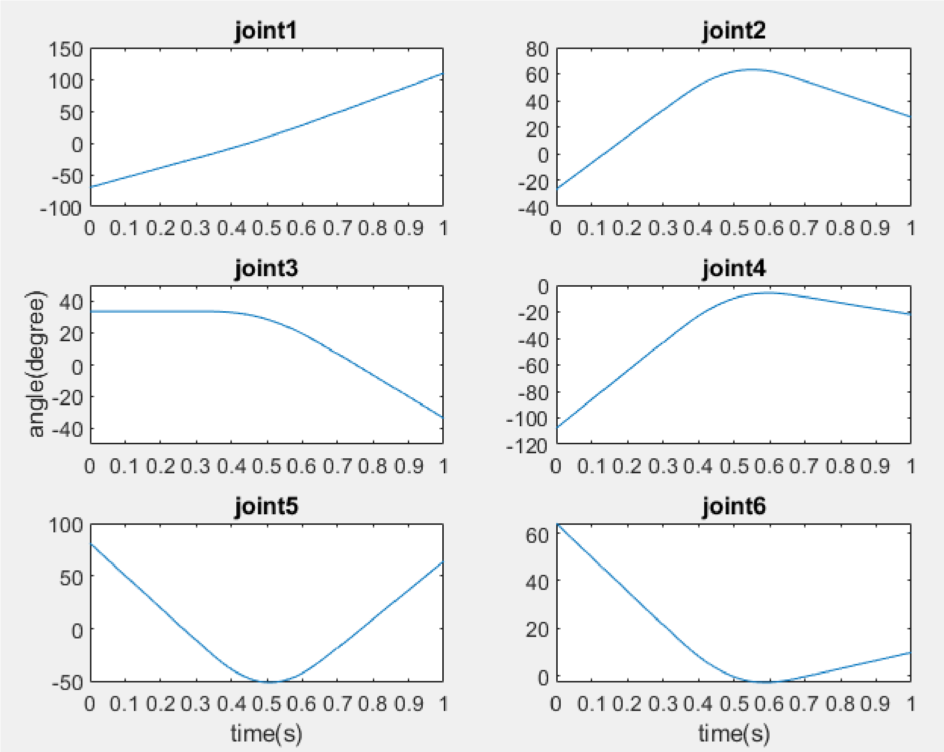
我的結果



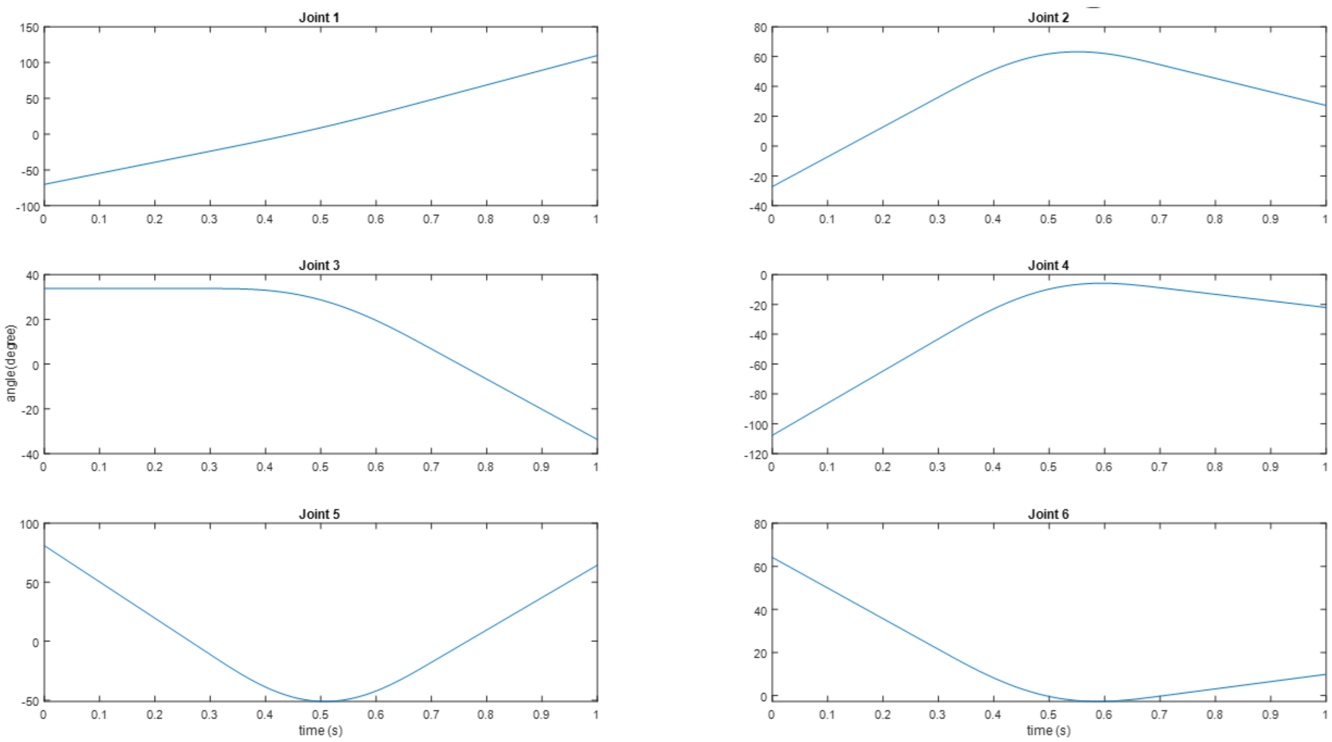
助教結果

311605012 機器人碩士碩一 楊曜瑋

- Angle



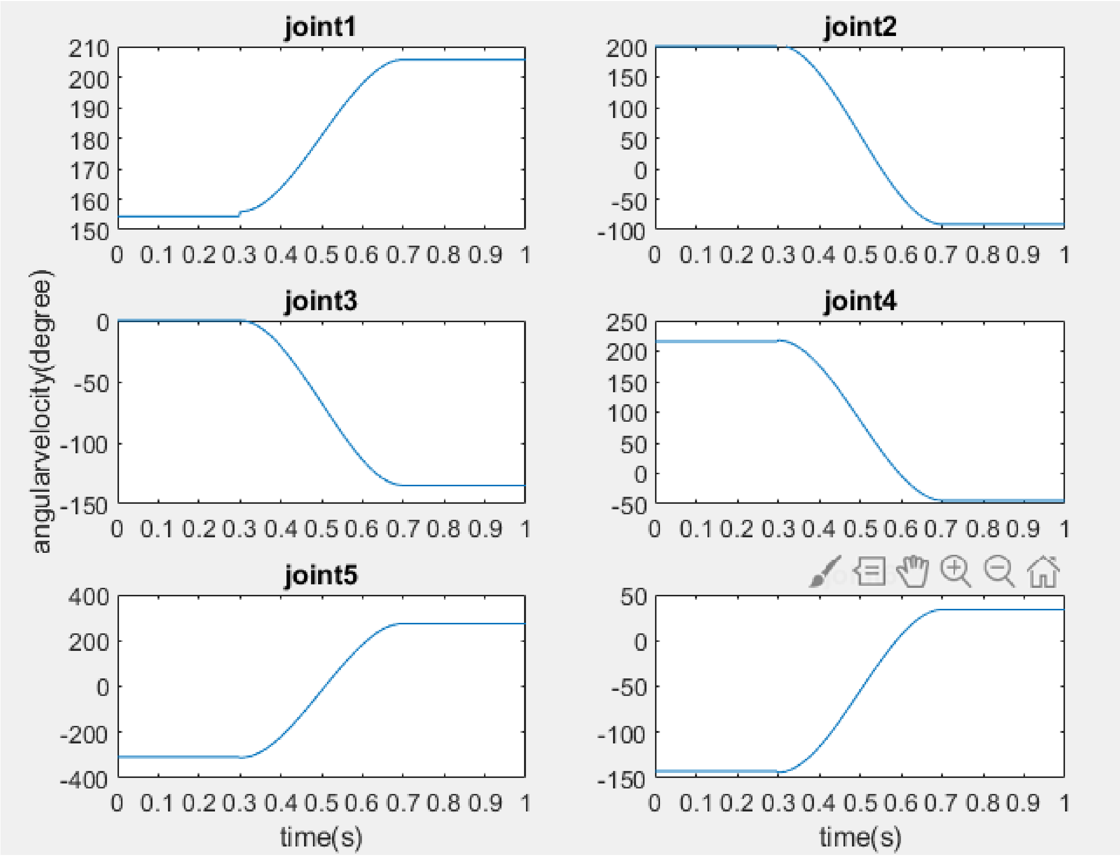
我的結果



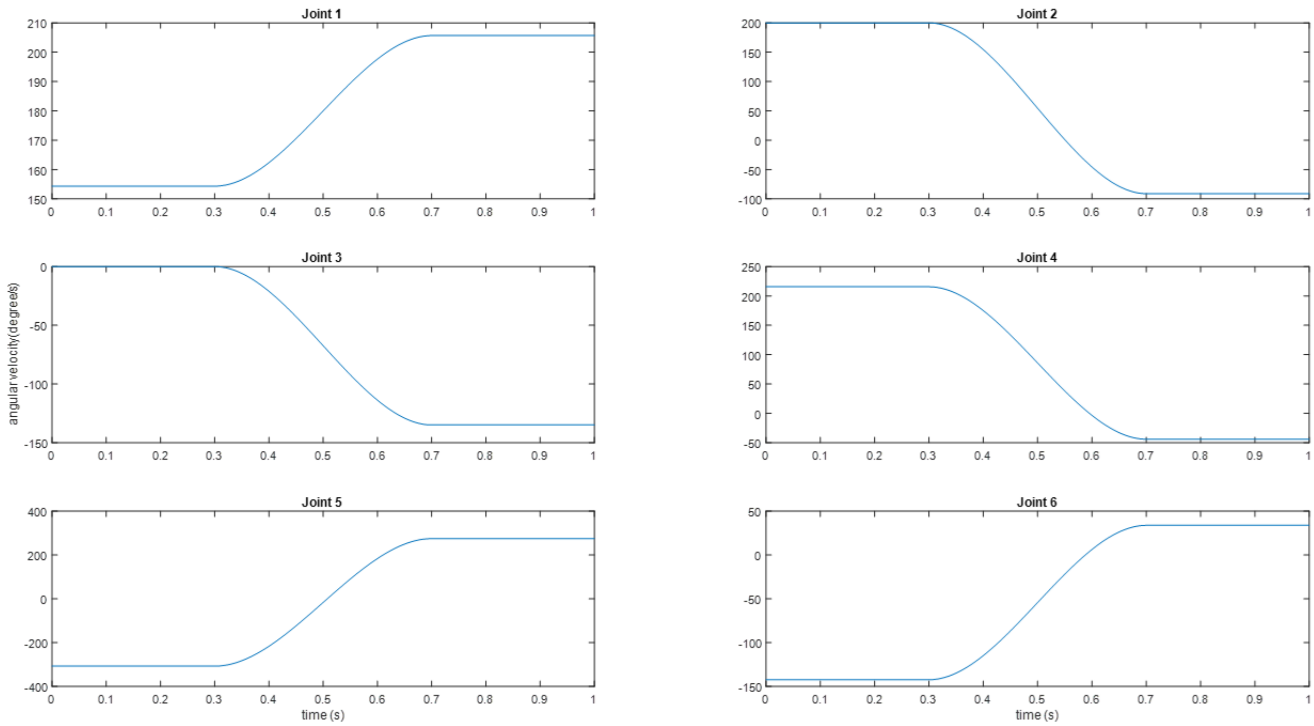
助教結果

311605012 機器人碩士碩一 楊曜瑋

- Angular velocity



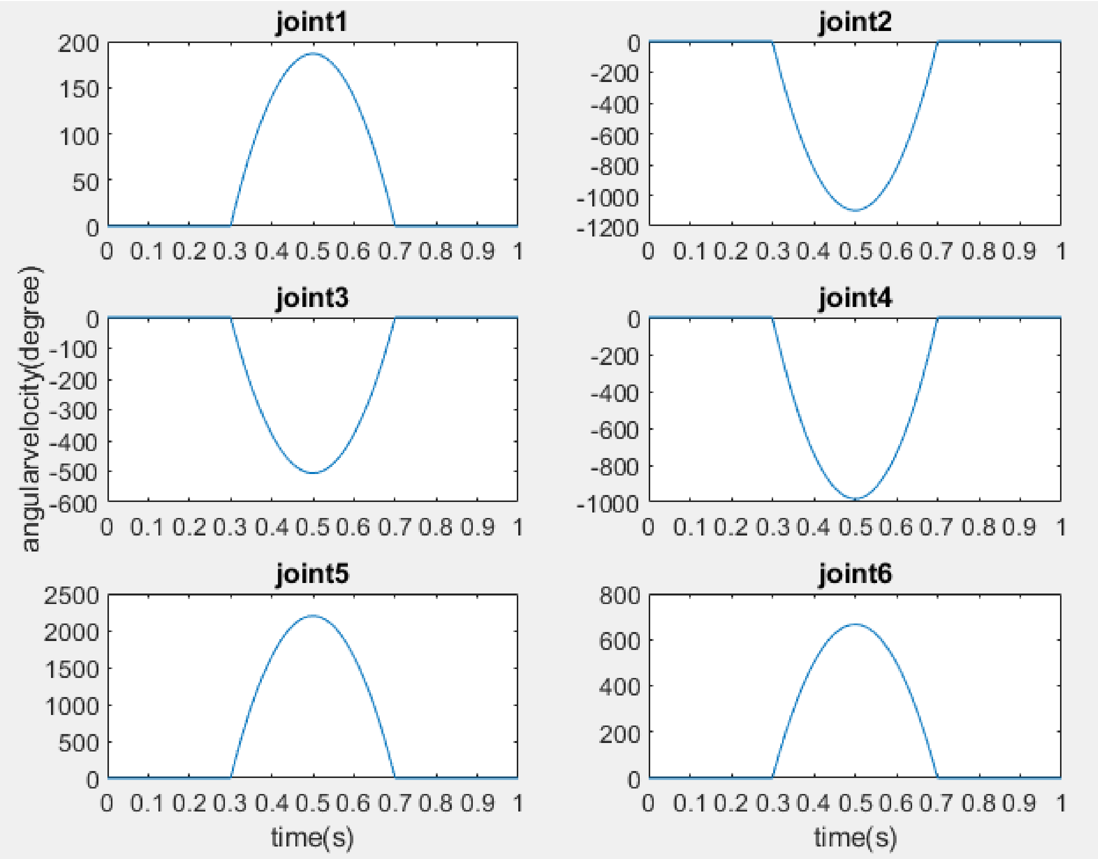
我的結果



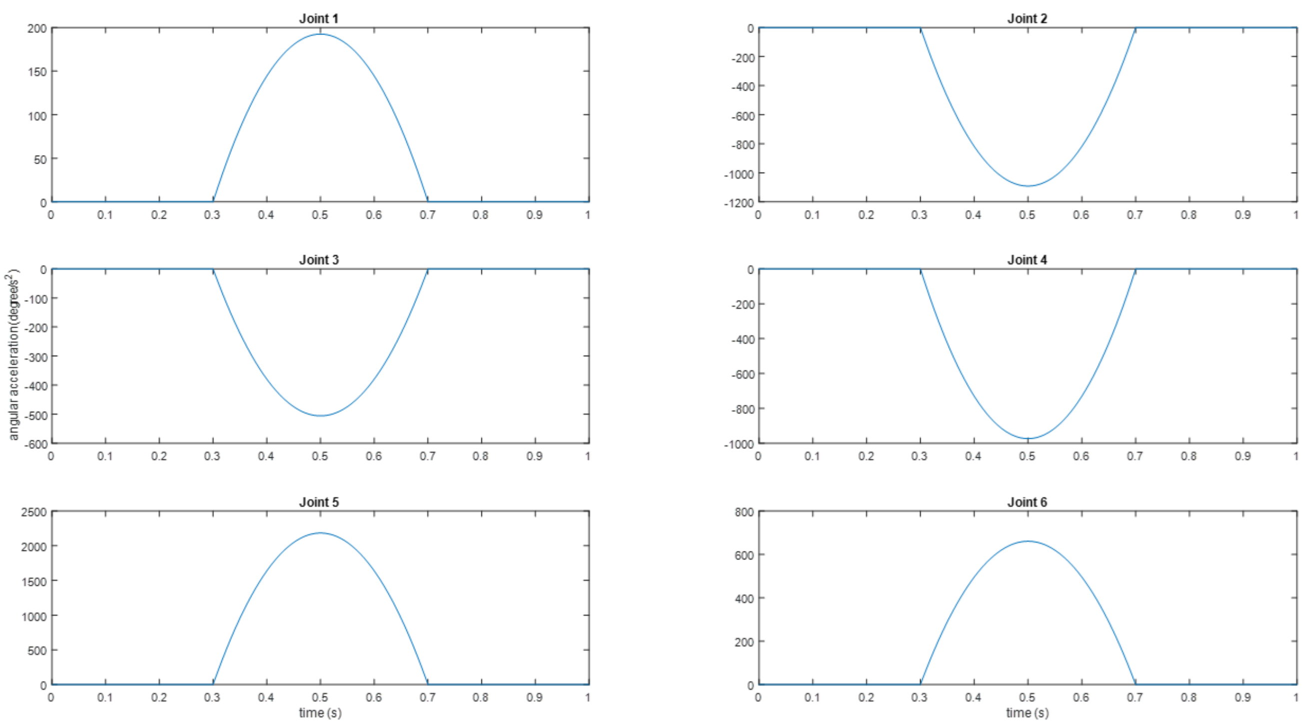
助教結果

311605012 機器人碩士碩一 楊曜瑋

- Angular acceleration



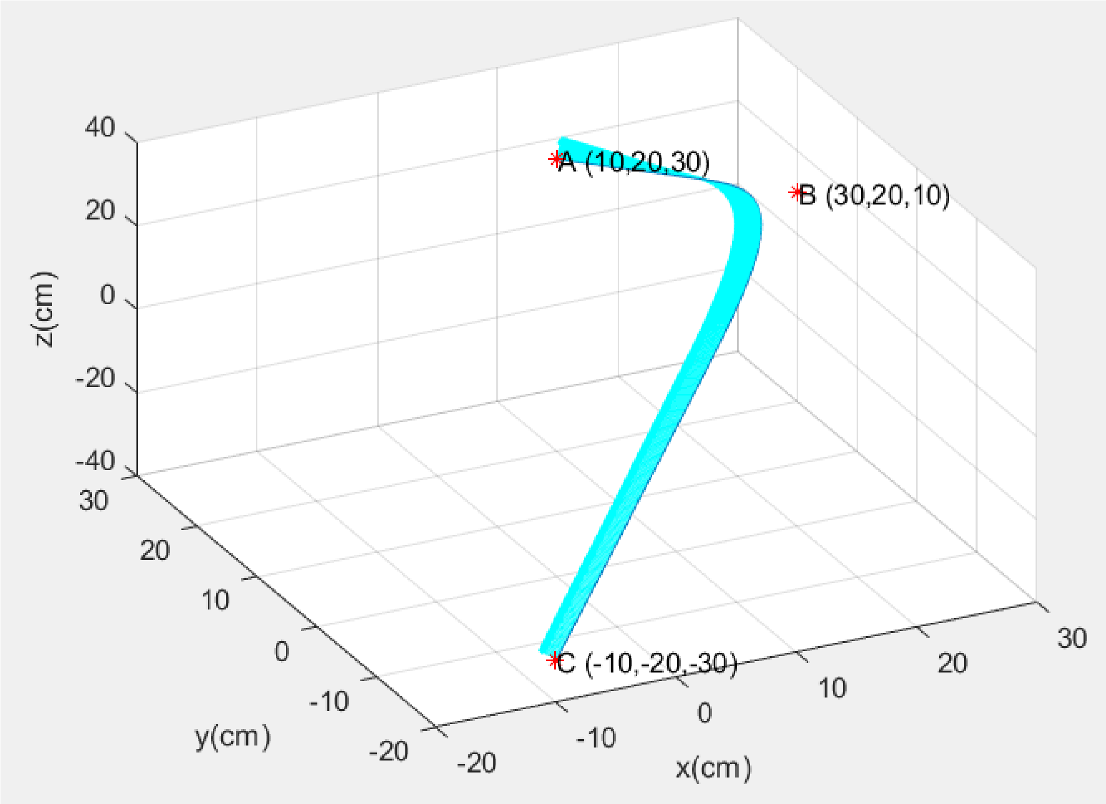
我的結果



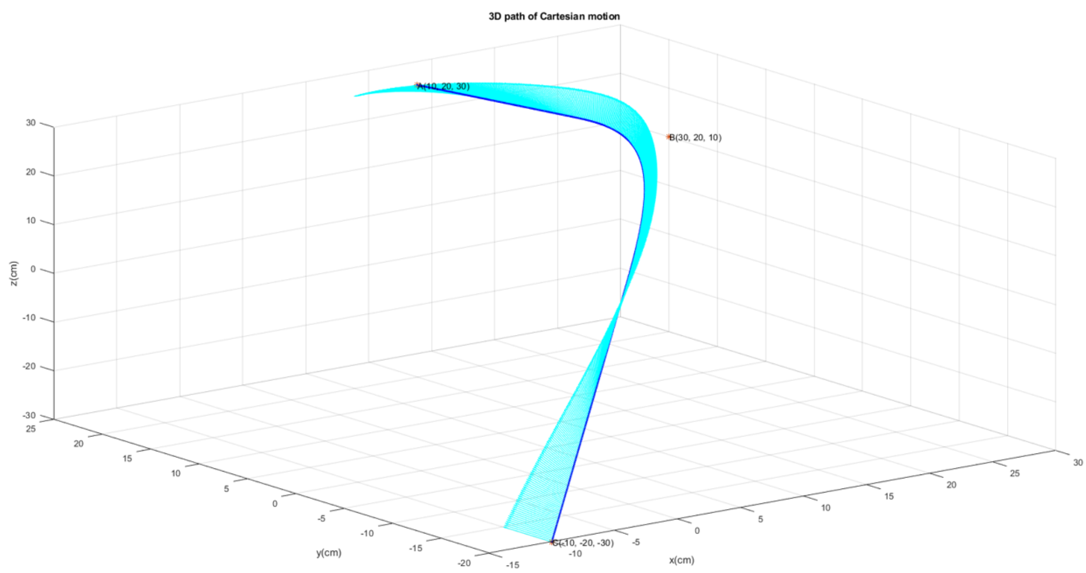
助教結果

311605012 機器人碩士碩一 楊曜瑋

- **Cartesian move**
  - 3D plot



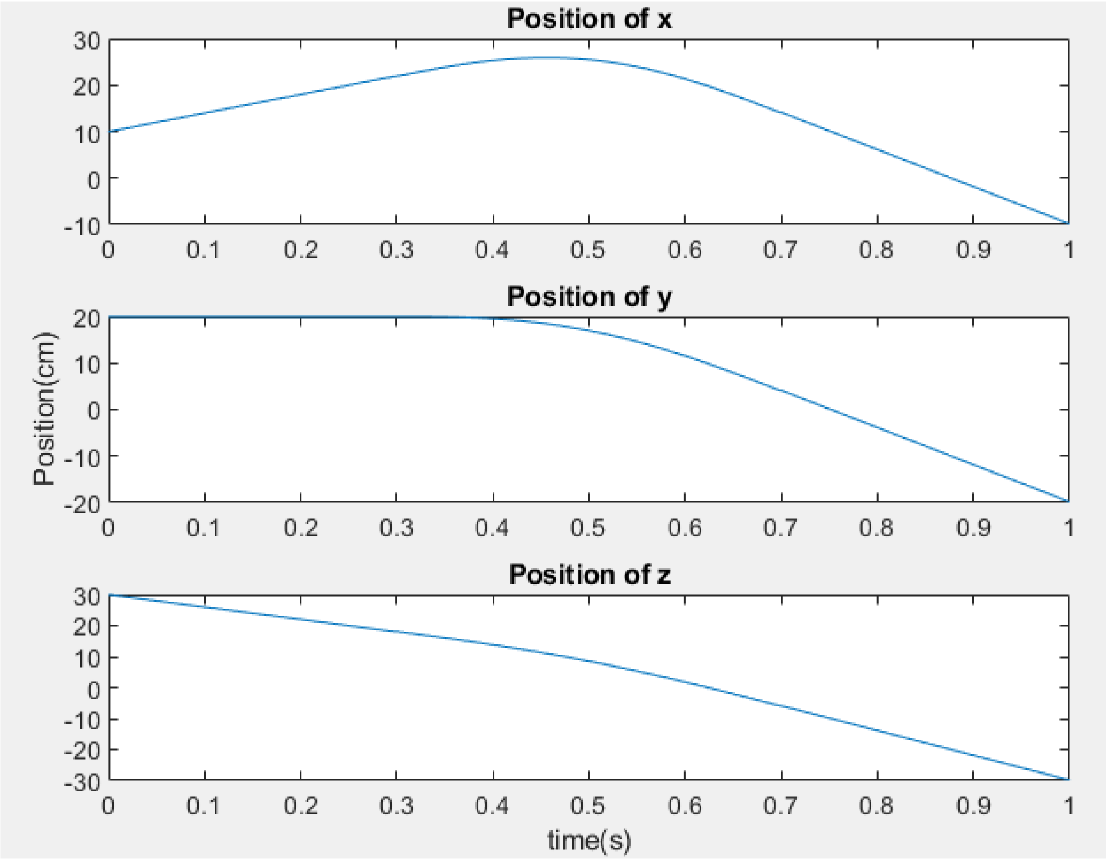
我的結果



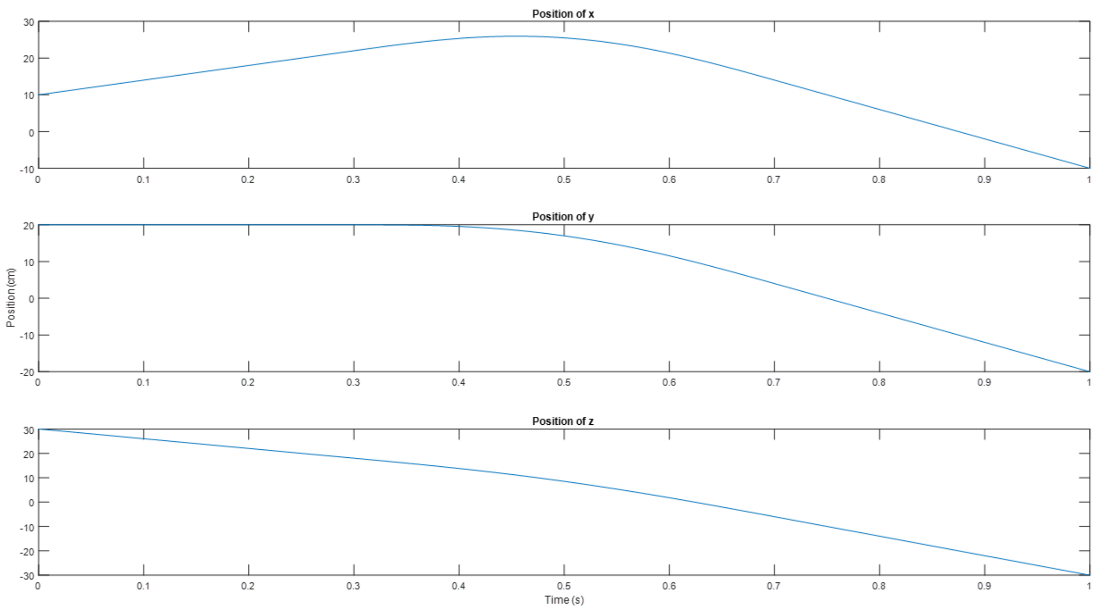
助教結果

311605012 機器人碩士碩一 楊曜瑋

- Position of x,y,z



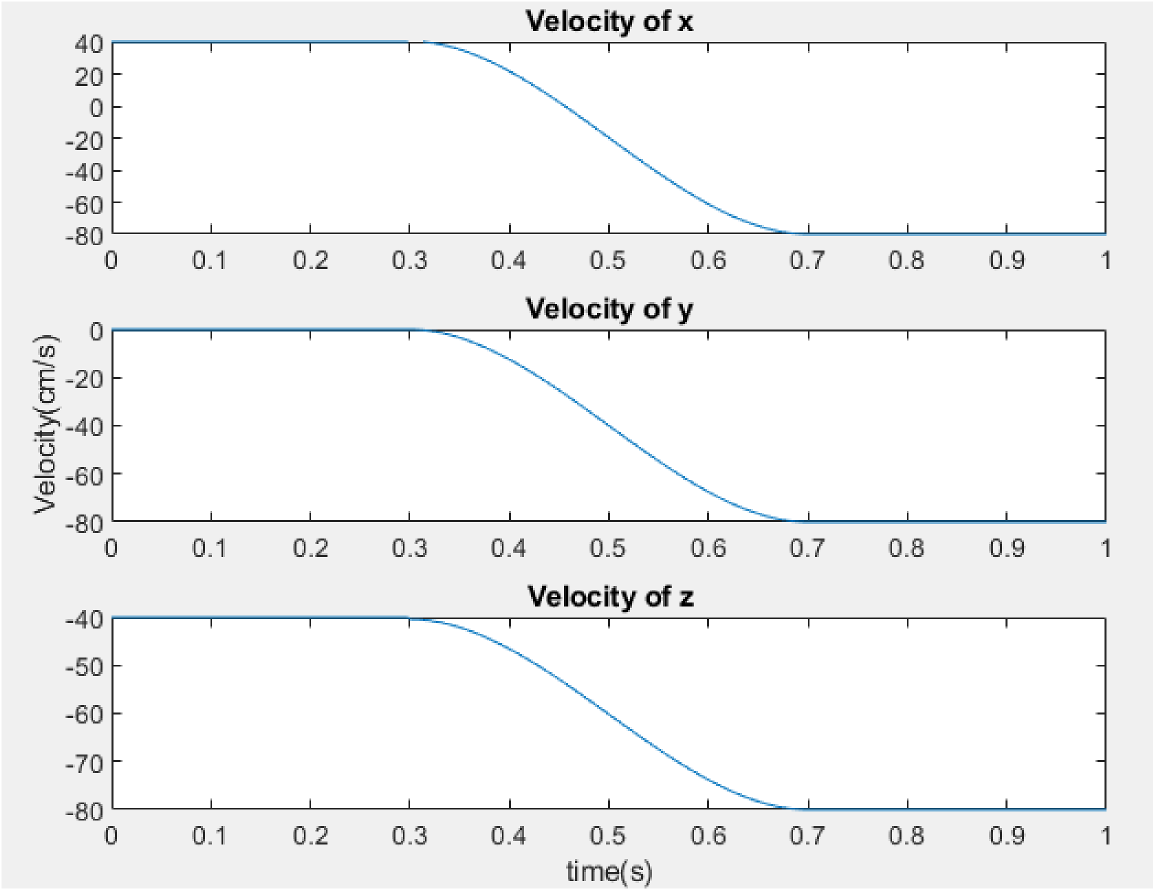
我的結果



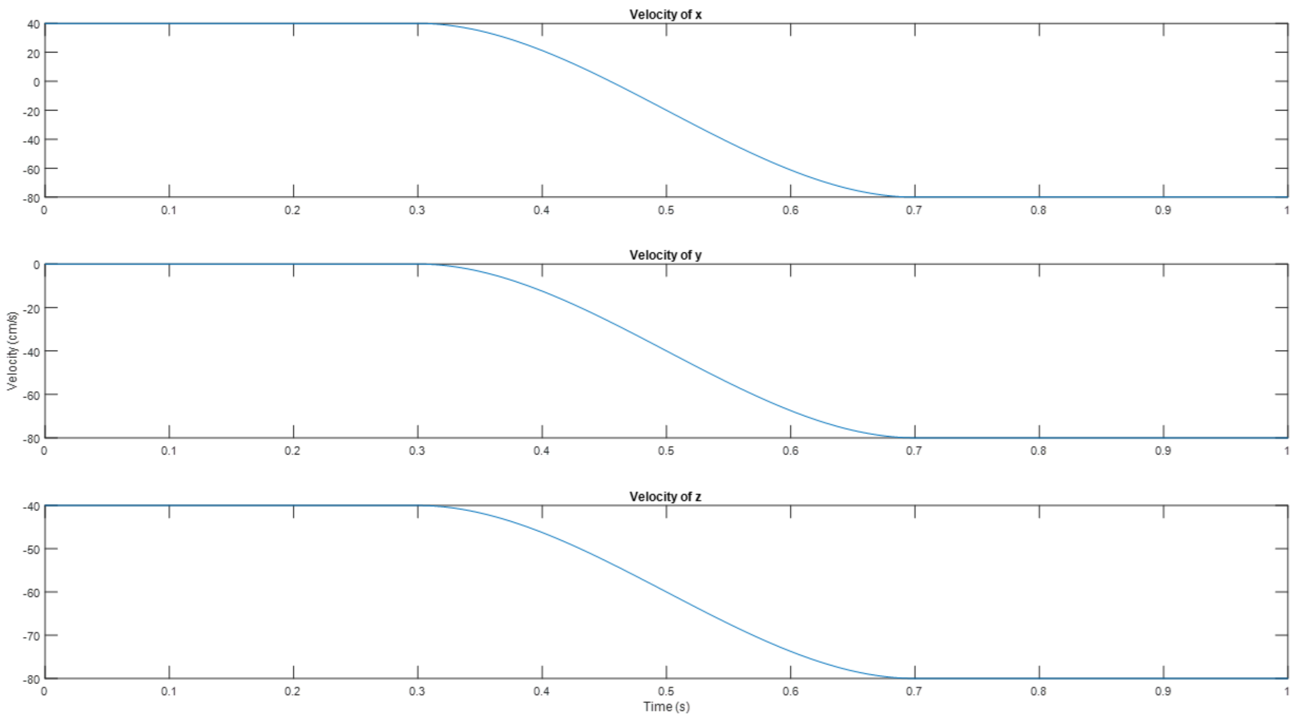
助教結果

311605012 機器人碩士碩一 楊曜瑋

- Velocity of x,y,z



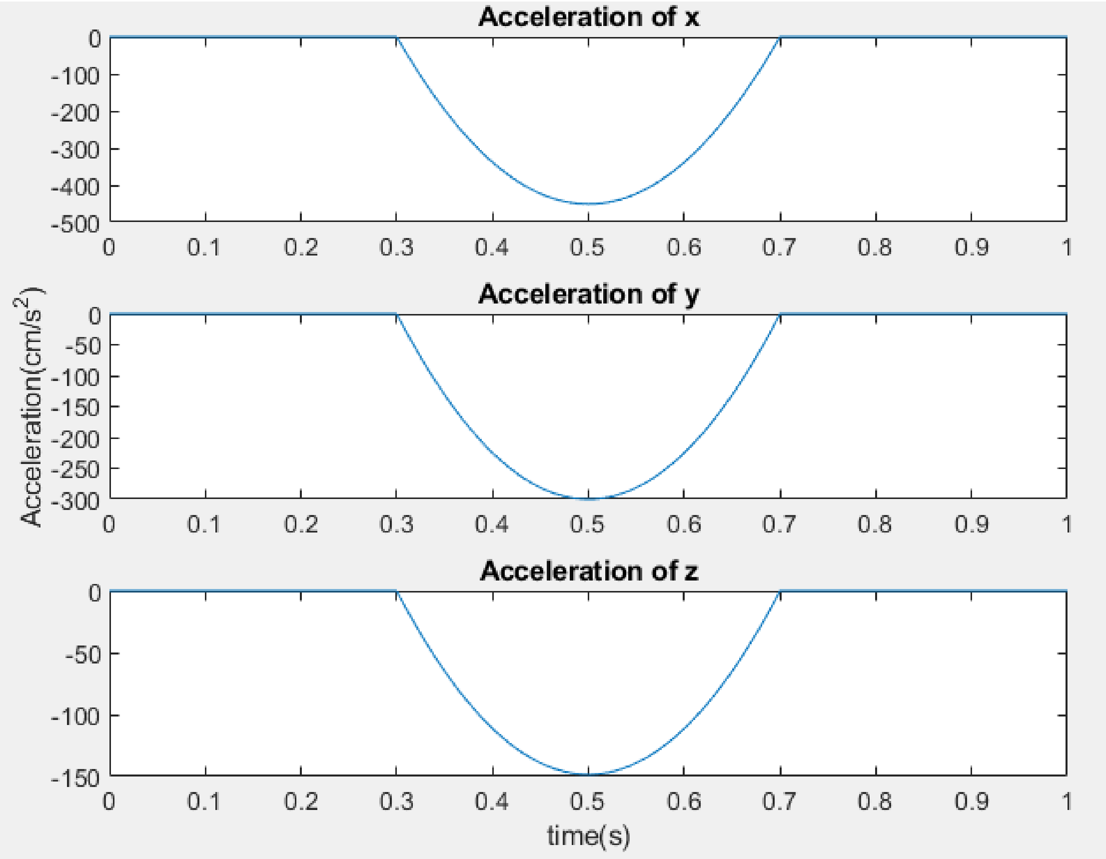
我的結果



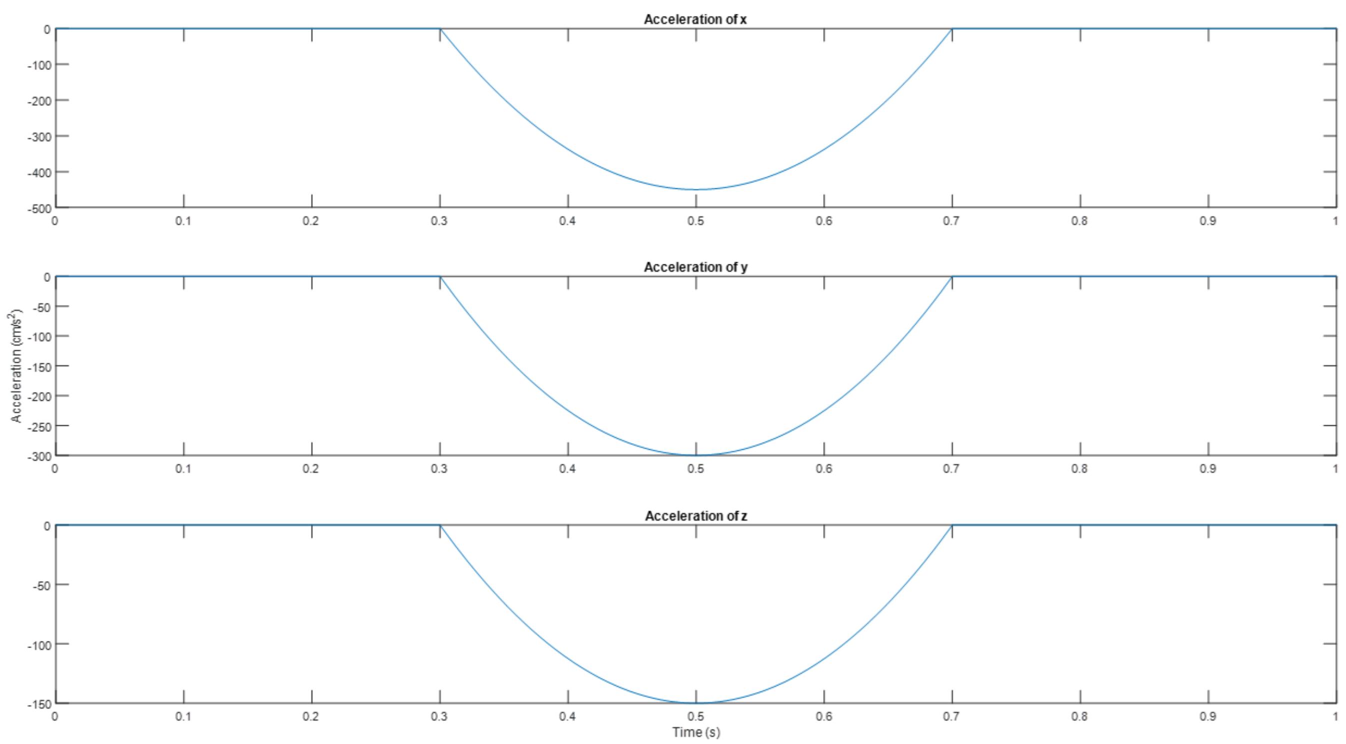
助教結果

311605012 機器人碩士碩一 楊曜瑋

- Acceleration of x,y,z



我的結果



助教結果

311605012 機器人碩士碩一 楊曜瑋

五、 加分題

- Joint Motion:
  - 優點: efficient in computation, no singularity problem, no configuration problem, minimum time planning.
  - 缺點: the corresponding Cartesian locations may be complicated.

- Cartesian Motion:
  - 優點: motion between path segments and points is well defined. Different constraints, such as smoothness and shortest path, etc., can be imposed upon.
  - 缺點:

1. Computational load is high
2. The motion breaks down when singularity occurs i.e. $\dot{x} = J\theta$, J is not invertible

- 遇到奇異點的解決方式

建立一些目標點在奇異點周圍，將奇異點當成障礙物，在規劃路徑時，避開這些障礙物