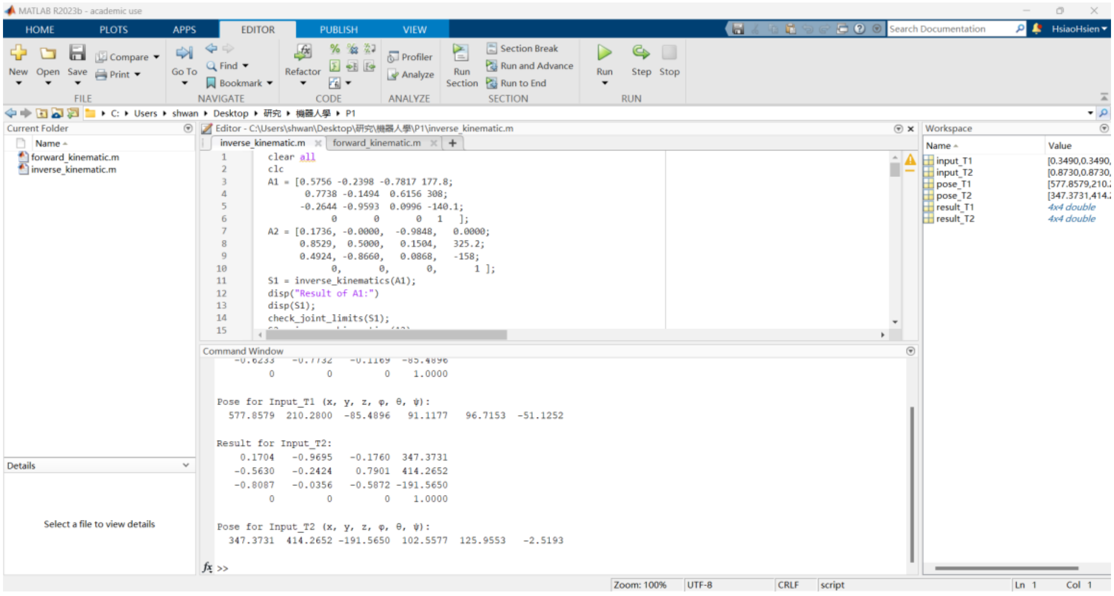
李效賢　學號 312512011 shwan255143@gmail.com

1.介面說明：

開發平台：Matlab 2023b release

執行方式：按下工具列右上的 Run 按鈕一鍵執行 .m 腳本



Results:

Part I Forward Kinematic

1. input_T1 = [0.349, 0.349, -0.349, 0.349, 0.349, 0.349];

```
Result for Input_T1:
    0.7798   -0.6257   -0.0194  577.8579
   -0.0582   -0.1033    0.9930  210.2800
   -0.6233   -0.7732   -0.1169  -85.4896
         0         0         0    1.0000

Pose for Input_T1 (x, y, z, φ, θ, ψ):
  577.8579  210.2800  -85.4896   91.1177   96.7153  -51.1252
```

2. input_T2 = [0.873, 0.873, -0.873, 0.873, 0.873, 0.873];

```
Result for Input_T2:
    0.1704   -0.9695   -0.1760  347.3731
   -0.5630   -0.2424    0.7901  414.2652
   -0.8087   -0.0356   -0.5872 -191.5650
         0         0         0    1.0000

Pose for Input_T2 (x, y, z, φ, θ, ψ):
  347.3731  414.2652 -191.5650  102.5577  125.9553   -2.5193
```

核心程式碼架構：

Part I Forward Kinematic

```
clear all
clc
% Test joint angles (degrees)
input_T1 = [0.349, 0.349, -0.349, 0.349, 0.349, 0.349]; % rad
input_T2 = [0.873, 0.873, -0.873, 0.873, 0.873, 0.873]; % rad

% call the forward kinematic function
[result_T1, pose_T1] = forward_kinematics(input_T1);
[result_T2, pose_T2] = forward_kinematics(input_T2);

% Show the Result
fprintf('Result for Input_T1:\n');
disp(result_T1);
fprintf('Pose for Input_T1 (x, y, z, ϕ, θ, ψ):\n');
disp(pose_T1);

fprintf('Result for Input_T2:\n');
disp(result_T2);
fprintf('Pose for Input_T2 (x, y, z, ϕ, θ, ψ):\n');
disp(pose_T2);
```

定義 Input (每個 joint 角度)，Forward kinematic 計算過程包成 function 型式，r 回傳計算結果(noap 矩陣)和 pose(x, y, z, φ, θ, ψ)，這部分在下面會解釋：

計算 forward kinematic 的函式首先依照 P1 上的說明，訂出每個關節的活動範圍限制，確認是否超過活動範圍，若超過則顯示錯誤訊息：標示第幾個 joint 超出範圍，允許的活動範圍是幾度到幾度。

```
function [T, pose] = forward_kinematics(joint_angles)
    % Mitsubishi RV-M2 six-axis robotic arm DH parameters
    % joint_angles: six joint angles (rad)
    % Output:
    % T: Homogeneous transformation matrix for the entire manipulator
    % pose: End-effector position and orientation (x, y, z, ϕ, θ, ψ)

    % Joint limits (rad)
    joint_limits_deg = [
        -150, 150;
        -30, 100;
        -120, 0;
        -110, 110;
        -180, 180;
        -180, 180
    ];
    joint_limits_rad = deg2rad(joint_limits_deg); % Convert to radians

    % Check if the number of joint angles is correct
    if length(joint_angles) ~= 6
        error('6 joint angles are required.');
    end

    % Check if each joint angle is within range
    for i = 1:6
        if joint_angles(i) < joint_limits_rad(i, 1) || joint_angles(i) > joint_limits_rad(i, 2)
            error('Joint %d is out of range: angle %.2f degrees (allowed range: %.2f ~ %.2f degrees).', ...
                i, rad2deg(joint_angles(i)), joint_limits_deg(i, 1), joint_limits_deg(i, 2));
        end
    end
```

接著定義 DH parameter table，也就是 θ, d, a , α 四個參數。

```
% DH parameter table [theta d a alpha]
dh_table = [
    joint_angles(1) 0 120 -pi/2;
    joint_angles(2) 0 250 0;
    joint_angles(3) 0 260 0;
    joint_angles(4) 0 0 -pi/2;
    joint_angles(5) 0 0 pi/2;
    joint_angles(6) 0 0 0
];
```

\\

然後計算初始化一個 4*4 的 identity 矩陣，定義 Ai(=1~6)，在迴圈內相乘最後得到結果。

```
% Initialize the homogeneous transformation matrix
T = eye(4); % 4x4 identity matrix
for i = 1:size(dh_table, 1)
    theta = dh_table(i, 1);
    d = dh_table(i, 2);
    a = dh_table(i, 3);
    alpha = dh_table(i, 4);

    % Compute the transformation matrix for each joint
    A_i = [
        cos(theta), -sin(theta)*cos(alpha), sin(theta)*sin(alpha), a*cos(theta);
        sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta);
        0, sin(alpha), cos(alpha), d;
        0, 0, 0, 1
    ];

    % Accumulate the product
    T = T * A_i;
end
```

最後是計算 Pose，px, py, pz 分別是最後一個 column 的 1, 2, 3 個元素。

$$\Phi = \tan^{-1}(\frac{a_y}{a_x})$$

$$\theta = \tan^{-1}(\frac{c(\Phi)a_x+s(\Phi)a_y}{a_z})$$

$$\psi = \tan^{-1}(\frac{-s(\Phi)n_x+c(\Phi)n_y}{-s(\Phi)o_x+c(\Phi)o_y})$$

```
% Extract position (x, y, z)
x = T(1, 4);
y = T(2, 4);
z = T(3, 4);

% Extract rotation matrix
R = T(1:3, 1:3);
% Calculate Euler angles (ϕ, θ, ψ)


phi = atan2(R(2, 3), R(1, 3)); % Roll
theta = atan2((cos(phi)*T(1, 3)+sin(phi)*T(2, 3)) , T(3, 3)); % Pitch
psi = atan2((-sin(phi)*T(1, 1)+cos(phi)*T(2,1)),(-sin(phi)*T(1, 2)+cos(phi)*T(2,2))); % Yaw

% Final pose
pose = [x, y, z, phi*180/pi, theta*180/pi, psi*180/pi];
```

最後*180/pi 是把 rad 轉換成 degree，用 Matlab 的 rad2deg 指令也可以。

Part II. Inverse Kinematic

1.A1

```
(x,y,z,ψ,θ,φ) =
    0.1778    0.3080   -0.1401    2.4745    1.4710    1.8398

Result of A1:
   60.0033  -28.5307  115.0036   58.5440  -10.0045   49.9807
   60.0033  -28.5307  115.0036 -121.4560   10.0045 -130.0193
   60.0033   89.9989 -115.0036   -9.9785   10.0045   49.9807
   60.0033   89.9989 -115.0036  170.0215  -10.0045 -130.0193
 -119.9967  149.7815   27.0712   38.1305  169.9955 -130.0193
 -119.9967  149.7815   27.0712 -141.8695 -169.9955   49.9807
 -119.9967  177.3935  -27.0712   64.6608  169.9955 -130.0193
 -119.9967  177.3935  -27.0712 -115.3392 -169.9955   49.9807
```

```
Checking joint limits:
Solution 1:
θ3 out of range! (115.00)
Solution 2:
θ3 out of range! (115.00)
θ4 out of range! (-121.46)
Solution 3:
All joint angles within range.
Solution 4:
θ4 out of range! (170.02)
Solution 5:
θ2 out of range! (149.78)
θ3 out of range! (27.07)
Solution 6:
θ2 out of range! (149.78)
θ3 out of range! (27.07)
θ4 out of range! (-141.87)
Solution 7:
θ2 out of range! (177.39)
Solution 8:
θ2 out of range! (177.39)
θ4 out of range! (-115.34)
```

2.A2

```
Result of A2:
   90.0000  -23.8120  119.0019   54.8197  -10.0002   -0.0083
   90.0000  -23.8120  119.0019 -125.1803   10.0002  179.9917
   90.0000   99.0031 -119.0019   -9.9917   10.0002   -0.0083
   90.0000   99.0031 -119.0019  170.0083  -10.0002  179.9917
  -90.0000  137.8618   44.2830   27.8457  169.9998  179.9917
  -90.0000  137.8618   44.2830 -152.1543 -169.9998   -0.0083
  -90.0000 -176.9410  -44.2830   71.2144  169.9998  179.9917
  -90.0000 -176.9410  -44.2830 -108.7856 -169.9998   -0.0083


Checking joint limits:
Solution 1:
θ3 out of range! (119.00)
Solution 2:
θ3 out of range! (119.00)
θ4 out of range! (-125.18)
Solution 3:
All joint angles within range.
Solution 4:
θ4 out of range! (170.01)
Solution 5:
θ2 out of range! (137.86)
θ3 out of range! (44.28)
Solution 6:
θ2 out of range! (137.86)
θ3 out of range! (44.28)
θ4 out of range! (-152.15)
Solution 7:
θ2 out of range! (-176.94)
Solution 8:
θ2 out of range! (-176.94)
```

程式架構說明

首先清除所有計算的過程以及命令列的文字方便觀測結果，接下來定義輸入

A1A2 兩個 noap 矩陣。

```
clear all
clc
A1 = [0.5756 -0.2398 -0.7817 177.8;
       0.7738 -0.1494  0.6156 308;
      -0.2644 -0.9593  0.0996 -140.1;
            0       0       0    1    ];
A2 = [0.1736, -0.0000,  -0.9848,   0.0000;
      0.8529,  0.5000,   0.1504,   325.2;
      0.4924, -0.8660,   0.0868,  -158;
           0,       0,       0,       1 ];
```

由於計算 inverse kinematic 要先算出對應的關節角度才能確認是否碰到 limit，

故我把這兩件事分開來處理，之後會詳述。

```
S1 = inverse_kinematics(A1);
disp("Result of A1:")
disp(S1);
check_joint_limits(S1);
S2 = inverse_kinematics(A2);
disp("Result of A2:");
disp(S2);
check_joint_limits(S2);
```

計算 inverse kinematics 的 function 輸入為 noap 矩陣，首先把 nx, ny, nz....等相對應位置的變數傳遞到相對應的符號，px, py, pz 由於我 a1, a2, a3 選擇使用 m 為單位所以要乘以 0.001。然後計算姿態(x, y, z, φ, θ, ψ)

```
function [Solutions] = inverse_kinematics(Cartesian_point)
    nx = Cartesian_point(1,1);
    ny = Cartesian_point(2,1);
    nz = Cartesian_point(3,1);
    ox = Cartesian_point(1,2);
    oy = Cartesian_point(2,2);
    oz = Cartesian_point(3,2);
    ax = Cartesian_point(1,3);
    ay = Cartesian_point(2,3);
    az = Cartesian_point(3,3);
    px = Cartesian_point(1,4)*0.001;
    py = Cartesian_point(2,4)*0.001;
    pz = Cartesian_point(3,4)*0.001;
    a1 = 0.12;
    a2 = 0.25;
    a3 = 0.26;
    phi = atan2(ay,ax);
    thetaa = atan2((cos(phi)*ax + sin(phi)*ay),az);
    psi = atan2((-sin(phi)*nx + cos(phi)*ny),(-sin(phi)*ox + cos(phi)*oy));
    if(psi < 0)
        psi = psi +pi;
    end
    disp('(x,y,z,ψ,θ,ϕ) =')
    disp([px py pz phi thetaa psi])
```

開始計算 theta1~theta6 的過程，不過每一次計算完都要確保角度落在-180~180 degree 之間，所以每一次計算完都會先經過我自己寫的 wrapToPi function 確保輸出的角度會落在此範圍之內。

```
function theta = wrapToPi(the)
    if the>pi
        theta = the-2*pi;
    elseif the<-pi
        theta = the+2*pi;
    else
        theta = the;
    end
end
```

以下為 theta1~theta6 的計算過程：

```
%Solve theta1
theta1_1 = atan2(py,px);
theta1_2 = atan2(py,px)-pi;
%Solve theta2
gama1 = atan2(-pz,(cos(theta1_1)*px+sin(theta1_1)*py)-a1);
gama2 = atan2(-pz,(cos(theta1_2)*px+sin(theta1_2)*py)-a1);
R1 = (((cos(theta1_1)*px+sin(theta1_1)*py)-a1)^2 + pz^2)^(0.5);
R2 = (((cos(theta1_2)*px+sin(theta1_2)*py)-a1)^2 + pz^2)^(0.5);
aphar1 = acos((R1^2+a2^2-a3^2)/(2*R1*a2));
aphar2 = acos((R2^2+a2^2-a3^2)/(2*R2*a2));
theta2_1 = (gama1 - aphar1);
theta2_2 = (gama1 + aphar1);
theta2_3 = (gama2 - aphar2);
theta2_4 = (gama2 + aphar2);
theta2_1 = wrapToPi(theta2_1);
theta2_2 = wrapToPi(theta2_2);
theta2_3 = wrapToPi(theta2_3);
theta2_4 = wrapToPi(theta2_4);
%Solve theta3
beta1 = acos((R1^2+a3^2-a2^2)/(2*R1*a3));
beta2 = acos((R2^2+a3^2-a2^2)/(2*R2*a3));
theta3_1 = aphar1 + beta1;
theta3_2 = -aphar1 - beta1;
theta3_3 = aphar2 + beta2;
theta3_4 = -aphar2 - beta2;
theta3_1 = wrapToPi(theta3_1);
theta3_2 = wrapToPi(theta3_2);
theta3_3 = wrapToPi(theta3_3);
theta3_4 = wrapToPi(theta3_4);
```

theta1 ~ theta3 使用 geometric solutional 計算，theta4~theta6 使用 gebraic solution 計算

```
tan_theta4_1 = (-cos(theta1_1)*sin(theta2_1+theta3_1)*ax - ...
sin(theta1_1)*sin(theta2_1+theta3_1)*ay - cos(theta2_1+theta3_1)*az) /...
(cos(theta1_1)*cos(theta2_1+theta3_1)*ax + sin(theta1_1)*...
cos(theta2_1+theta3_1)*ay - sin(theta2_1+theta3_1)*az);
theta4_1_1 = atan(tan_theta4_1);
theta4_1_1 = wrapToPi(theta4_1_1);
theta4_1_2 = theta4_1_1 - pi;
theta4_1_2 = wrapToPi(theta4_1_2);

tan_theta4_2 = (-cos(theta1_1)*sin(theta2_2+theta3_2)*ax ...
- sin(theta1_1)*sin(theta2_2+theta3_2)*ay - cos(theta2_2+theta3_2)*az)...
/ (cos(theta1_1)*cos(theta2_2+theta3_2)*ax + sin(theta1_1)*cos(theta2_2...
+theta3_2)*ay - sin(theta2_2+theta3_2)*az);
theta4_2_1 = atan(tan_theta4_2);
theta4_2_1 = wrapToPi(theta4_2_1);
theta4_2_2 = theta4_2_1 - pi;
theta4_2_2 = wrapToPi(theta4_2_2);

tan_theta4_3 = (-cos(theta1_2)*sin(theta2_3+theta3_3)*ax - sin(theta1_2)...
*sin(theta2_3+theta3_3)*ay - cos(theta2_3+theta3_3)*az) / (cos(theta1_2)...
*cos(theta2_3+theta3_3)*ax + sin(theta1_2)*cos(theta2_3+theta3_3)*ay - ...
sin(theta2_3+theta3_3)*az);
theta4_3_1 = atan(tan_theta4_3);
theta4_3_1 = wrapToPi(theta4_3_1);
theta4_3_2 = theta4_3_1 - pi;
theta4_3_2 = wrapToPi(theta4_3_2);
```

```
theta5_1 = atan2(ax*cos(theta1_1)*cos(theta2_1+theta3_1+theta4_1_1) +...
ay*sin(theta1_1)*cos(theta2_1+theta3_1+theta4_1_1) - ...
az*sin(theta2_1+theta3_1+theta4_1_1) , -ax*sin(theta1_1)+ay*cos(theta1_1));
theta5_2 = atan2(ax*cos(theta1_1)*cos(theta2_1+theta3_1+theta4_1_2) +...
ay*sin(theta1_1)*cos(theta2_1+theta3_1+theta4_1_2) - ...
az*sin(theta2_1+theta3_1+theta4_1_2) , -ax*sin(theta1_1)+ay*cos(theta1_1));
theta5_3 = atan2(ax*cos(theta1_1)*cos(theta2_2+theta3_2+theta4_2_1) +...
ay*sin(theta1_1)*cos(theta2_2+theta3_2+theta4_2_1) - ...
az*sin(theta2_2+theta3_2+theta4_2_1) , -ax*sin(theta1_1)+ay*cos(theta1_1));
theta5_4 = atan2(ax*cos(theta1_1)*cos(theta2_2+theta3_2+theta4_2_2) + ...
ay*sin(theta1_1)*cos(theta2_2+theta3_2+theta4_2_2) - ...
az*sin(theta2_2+theta3_2+theta4_2_2) , -ax*sin(theta1_1)+ay*cos(theta1_1));
theta5_5 = atan2(ax*cos(theta1_2)*cos(theta2_3+theta3_3+theta4_3_1) + ...
ay*sin(theta1_2)*cos(theta2_3+theta3_3+theta4_3_1) - ...
az*sin(theta2_3+theta3_3+theta4_3_1) , -ax*sin(theta1_2)+ay*cos(theta1_2));
theta5_6 = atan2(ax*cos(theta1_2)*cos(theta2_3+theta3_3+theta4_3_2) + ...
ay*sin(theta1_2)*cos(theta2_3+theta3_3+theta4_3_2) - ...
az*sin(theta2_3+theta3_3+theta4_3_2) , -ax*sin(theta1_2)+ay*cos(theta1_2));
theta5_7 = atan2(ax*cos(theta1_2)*cos(theta2_4+theta3_4+theta4_4_1) + ...
ay*sin(theta1_2)*cos(theta2_4+theta3_4+theta4_4_1) - ...
az*sin(theta2_4+theta3_4+theta4_4_1) , -ax*sin(theta1_2)+ay*cos(theta1_2));
theta5_8 = atan2(ax*cos(theta1_2)*cos(theta2_4+theta3_4+theta4_4_2) + ...
ay*sin(theta1_2)*cos(theta2_4+theta3_4+theta4_4_2) - ...
az*sin(theta2_4+theta3_4+theta4_4_2) , -ax*sin(theta1_2)+ay*cos(theta1_2));
```

```
theta6_1 = atan2(-nx*cos(theta1_1)*sin(theta2_1+theta3_1+theta4_1_1) - ...
ny*sin(theta1_1)*sin(theta2_1+theta3_1+theta4_1_1) - nz*cos(theta2_1+theta3_1+theta4_1_1) , ...
-ox*cos(theta1_1)*sin(theta2_1+theta3_1+theta4_1_1) ...
-oy*sin(theta1_1)*sin(theta2_1+theta3_1+theta4_1_1)-oz*cos(theta2_1+theta3_1+theta4_1_1));
theta6_2 = atan2(-nx*cos(theta1_1)*sin(theta2_1+theta3_1+theta4_1_2) - ...
ny*sin(theta1_1)*sin(theta2_1+theta3_1+theta4_1_2)-nz*cos(theta2_1+theta3_1+theta4_1_2) , ...
-ox*cos(theta1_1)*sin(theta2_1+theta3_1+theta4_1_2) ...
-oy*sin(theta1_1)*sin(theta2_1+theta3_1+theta4_1_2)-oz*cos(theta2_1+theta3_1+theta4_1_2));
theta6_3 = atan2(-nx*cos(theta1_1)*sin(theta2_2+theta3_2+theta4_2_1) - ...
ny*sin(theta1_1)*sin(theta2_2+theta3_2+theta4_2_1)-nz*cos(theta2_2+theta3_2+theta4_2_1) , ...
-ox*cos(theta1_1)*sin(theta2_2+theta3_2+theta4_2_1)- ...
oy*sin(theta1_1)*sin(theta2_2+theta3_2+theta4_2_1)-oz*cos(theta2_2+theta3_2+theta4_2_1));
theta6_4 = atan2(-nx*cos(theta1_1)*sin(theta2_2+theta3_2+theta4_2_2) ...
-ny*sin(theta1_1)*sin(theta2_2+theta3_2+theta4_2_2)-nz*cos(theta2_2+theta3_2+theta4_2_2) , ...
-ox*cos(theta1_1)*sin(theta2_2+theta3_2+theta4_2_2)- ...
oy*sin(theta1_1)*sin(theta2_2+theta3_2+theta4_2_2)-oz*cos(theta2_2+theta3_2+theta4_2_2));
theta6_5 = atan2(-nx*cos(theta1_2)*sin(theta2_3+theta3_3+theta4_3_1)- ...
ny*sin(theta1_2)*sin(theta2_3+theta3_3+theta4_3_1)-nz*cos(theta2_3+theta3_3+theta4_3_1) , ...
-ox*cos(theta1_2)*sin(theta2_3+theta3_3+theta4_3_1)- ...
oy*sin(theta1_2)*sin(theta2_3+theta3_3+theta4_3_1)-oz*cos(theta2_3+theta3_3+theta4_3_1));
theta6_6 = atan2(-nx*cos(theta1_2)*sin(theta2_3+theta3_3+theta4_3_2)- ...
ny*sin(theta1_2)*sin(theta2_3+theta3_3+theta4_3_2)-nz*cos(theta2_3+theta3_3+theta4_3_2) , ...
-ox*cos(theta1_2)*sin(theta2_3+theta3_3+theta4_3_2)- ...
oy*sin(theta1_2)*sin(theta2_3+theta3_3+theta4_3_2)-oz*cos(theta2_3+theta3_3+theta4_3_2));
theta6_7 = atan2(-nx*cos(theta1_2)*sin(theta2_4+theta3_4+theta4_4_1)- ...
ny*sin(theta1_2)*sin(theta2_4+theta3_4+theta4_4_1)-nz*cos(theta2_4+theta3_4+theta4_4_1) , ...
-ox*cos(theta1_2)*sin(theta2_4+theta3_4+theta4_4_1)- ...
oy*sin(theta1_2)*sin(theta2_4+theta3_4+theta4_4_1)-oz*cos(theta2_4+theta3_4+theta4_4_1));
theta6_8 = atan2(-nx*cos(theta1_2)*sin(theta2_4+theta3_4+theta4_4_2)- ...
ny*sin(theta1_2)*sin(theta2_4+theta3_4+theta4_4_2)-nz*cos(theta2_4+theta3_4+theta4_4_2) , ...
-ox*cos(theta1_2)*sin(theta2_4+theta3_4+theta4_4_2)- ...
oy*sin(theta1_2)*sin(theta2_4+theta3_4+theta4_4_2)-oz*cos(theta2_4+theta3_4+theta4_4_2));
```

最後是判斷 angle limit 的 function， 若所有的角度皆在範圍內會顯示 All joint angles within range.

```
function check_joint_limits(Solutions)
    joint_limits = [
        -150, 150;  % θ1
        -30, 100;   % θ2
        -120, 0;    % θ3
        -110, 110;  % θ4
        -180, 180;  % θ5
        -180, 180   % θ6
    ];

    fprintf("Checking joint limits:\n");
    for i = 1:size(Solutions, 1)
        fprintf("Solution %d: \n", i);
        solution = Solutions(i, :);
        valid = true;
        for j = 1:size(joint_limits, 1)
            if solution(j) < joint_limits(j, 1) || solution(j) > joint_limits(j, 2)
                fprintf("θ%d out of range! (%.2f)\n", j, solution(j));
                valid = false;
            end
        end
        if valid
            fprintf("All joint angles within range.\n");
        end
    end
end
```

數學方程式推導：

Part 1. Forward Kinematic

Forward Kinematic:

$$A_1 = \begin{bmatrix} C1 & 0 & -S1 & a_1C_1 \\ S1 & 0 & C1 & a_1S_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C2 & -S2 & 0 & a_2C_2 \\ S2 & C2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1T_6 = A_1 A_2 A_3 A_4 A_5 A_6 = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pose:

$$P_x = a_1C_1 + a_2C_1C_2 + a_3C_1C_{23} = {}^1T_6(1,4)$$

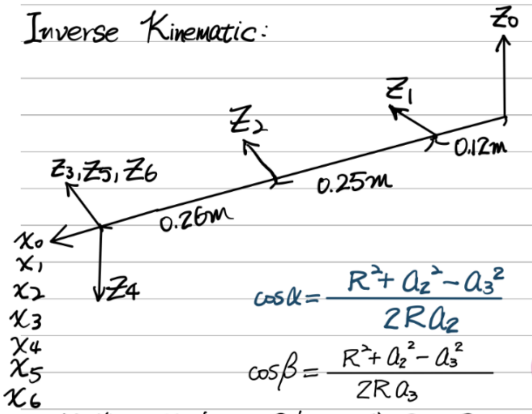
$$P_y = a_1S_1 + a_2S_1C_2 + a_3S_1C_{23} = {}^1T_6(2,4)$$

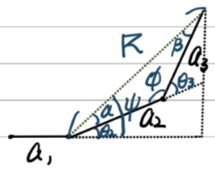
$$P_z = -a_2S_2 - a_3S_{23} = {}^1T_6(3,4)$$

$$\phi = \tan^{-1}\left(\frac{a_y}{a_x}\right), \quad \theta = \tan^{-1}\left[\frac{\cos\psi\, a_x + \sin\psi\, a_y}{a_z}\right], \quad \psi = \tan^{-1}\left[\frac{-\sin\psi\, n_x + \cos\psi\, n_y}{-\sin\psi\, o_x + \cos\psi\, o_y}\right]$$

Part II. Inverse Kinematic

Inverse Kinematic:





Geometric Solution:

$P_x = (a_1 + a_2 C_2 + a_3 C_{23}) C_1$

$P_y = (a_1 + a_2 C_2 + a_3 C_{23}) S_1$

$\Rightarrow \frac{P_y}{P_x} = \tan\theta_1$

$\theta_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right)$

$\tan\psi = \frac{P_z}{\sqrt{P_x^2+P_y^2} - a_1}$, $R = \sqrt{\left(\sqrt{P_x^2+P_y^2} - a_1\right)^2 + P_z^2}$

$\cos\alpha = \frac{R^2 + a_2^2 - a_3^2}{2R a_2}$ $\quad \theta_2 = \psi - \alpha$

$\cos\beta = \frac{R^2 + a_2^2 - a_3^2}{2R a_3}$ $\quad \theta_3 = \alpha + \beta$

接著用 Algebric Solution 求 $\theta_4, \theta_5, \theta_6$

$$ {}^1T_3^{-1} = A_3^{-1} A_2^{-1} A_1^{-1} = \begin{bmatrix} C_1C_{23} & S_1C_{23} & -S_{23} & -a_3 - a_2C_3 - a_1C_{23} \\ -C_1S_{23} & -S_1S_{23} & -C_{23} & a_2S_3 + a_1S_{23} \\ -S_1 & C_1 & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ A_4A_5A_6 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5C_6 - S_4C_6 & C_4S_5 & 0 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5C_6 + C_4C_6 & S_4C_5 & 0 \\ -S_5C_6 & S_5S_6 & C_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^1T_3^{-1} \times {}^1T_6 $$

$$ \begin{cases} C_1 C_{23} a_x + S_1 C_{23} a_y - S_{23} a_z = C_4 S_5 \\ -C_1 S_{23} a_x - S_1 S_{23} a_y - C_{23} a_z = S_4 S_5 \end{cases} $$

if $S_5 \neq 0$ 則 $\tan\theta_4 = \frac{S_4S_5}{C_4S_5} = \frac{-C_1C_{23}a_x - S_1S_{23}a_y - C_{23}a_z}{C_1C_{23}a_x + S_1C_{23}a_y - S_{23}a_z}$

$\theta_4 = \tan^{-1}\left[\frac{-C_1S_{23}a_x - S_1S_{23}a_y - C_{23}a_z}{C_1C_{23}a_x + S_1C_{23}a_y - S_{23}a_z}\right]$

$$ {}^1T_4^{-1} = A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1} = \begin{bmatrix} C_1C_{234} & S_1C_{234} & -S_{234} & -a_3C_4 - a_2C_{34} - a_1C_{234} \\ S_1 & -C_1 & 0 & 0 \\ -C_1S_{234} & -S_1S_{234} & -C_{234} & a_3S_4 + a_2S_{34} + a_1S_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ A_5A_6 = \begin{bmatrix} C_5C_6 & -C_5S_6 & S_5 & 0 \\ S_5C_6 & -S_5S_6 & -C_5 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^1T_4^{-1} \times {}^1T_6 \Rightarrow \begin{cases} S_5 = a_xC_1C_{234} + a_yS_1C_{234} - a_zS_{234} \\ -C_5 = a_xS_1 - a_yC_1 \end{cases} $$

$\theta_5 = \tan^{-1}\left(\frac{S_5}{C_5}\right) = \tan^{-1}\left[\frac{a_xC_1C_{234} + a_yS_1C_{234} - a_zS_{234}}{-a_xS_1 + a_yC_1}\right]$

$S_6 = -n_xC_1S_{234} - n_yS_1S_{234} - n_zS_{234}$

$C_6 = -o_xC_1S_{234} - o_yS_1S_{234} - o_zC_{234}$

$\Rightarrow \theta_6 = \tan^{-1}\left(\frac{S_6}{C_6}\right) = \tan^{-1}\left[\frac{-n_xC_1S_{234} - n_yS_1S_{234} - n_zC_{234}}{-o_xC_1S_{234} - o_yS_1S_{234} - o_zC_{234}}\right]$

加分題：討論兩種逆向運動學(代數法，幾何法)的優缺點 (10%)

代數法：優點是計算過程簡單，缺點為求解過程不直觀，且容易出現 singular solution 或是超出機器手臂工作範圍

幾何法：優點為求解過程直觀好理解，且較不易出現 singular solution 或超出手臂工作範圍的解。缺點為計算過程複雜需要繪圖來解決問題。