```
clear, clc

syms px py pz az ax ay
syms q1 q2 q3

L1 = 12
L2 = L1
L3 = L1

DHParam = [q1 L1 0 pi/2; q2 0 L2 0; q3 0 L3 0];

[A, T, Q, Rot, Tra] = DH(DHParam, px, py, pz, az);

%assume(q1>-pi & q1<pi & q2>-pi & q2<pi & q3>-pi & q3<pi)
assume(pz>0 & q1<0 & q1 > -pi)% & q2>0 & q3>0)

    EqX = px == simplify(Tra(1));
    EqY = py == simplify(Tra(2));
    EqZ = pz == simplify(Tra(3));

    jointVar = symvar(T);

%     Q3 = solve([EqZ], jointVar(3), 'real',true) % Se puede jugar con las
 variables a resolver y a partir de ahí sacar la última solución. Usando la
 primera variable como parámetro. Dado el caso de que las ecuaciones no son
 linealmente indemendientes entre sí
%     Q3b = Q3(1) %== q3
%     %q3 = Q3(1)
%     EqX = simplify(subs(EqX, q3, Q3b))
%     EqY = simplify(subs(EqY, q3, Q3b))
%     Q12 = solve([EqX EqY], jointVar(1:2)) % Se puede jugar con las
 variables a resolver y a partir de ahí sacar la última solución. Usando la
 primera variable como parámetro. Dado el caso de que las ecuaciones no son
 linealmente indemendientes entre sí

%     Q1 = solve([EqX], jointVar(1), 'real',true) % Se puede jugar con las
 variables a resolver y a partir de ahí sacar la última solución. Usando la
 primera variable como parámetro. Dado el caso de que las ecuaciones no son
 linealmente indemendientes entre sí
%     Q1b = Q1(2) %== q3
%     %q3 = Q3(1)
%     EqY = simplify(subs(EqY, jointVar(1), Q1b))
%     EqZ = simplify(subs(EqZ, jointVar(1), Q1b))
%     Q23 = solve([EqX EqY], jointVar(2:3)) % Se puede jugar con las
 variables a resolver y a partir de ahí sacar la última solución. Usando la
 primera variable como parámetro. Dado el caso de que las ecuaciones no son
 linealmente indemendientes entre sí


    %%%%%%%%%55

% A es el arreglo conteniendo las matrices de transformación de i = 1 a i =
```

```
% n. T es la matriz de transformación simplificada, la matriz de cinemática
% directa. Q es un vector conteniendo los resultados para las variables de
% q1 a qn. Rot es la matriz de rotación extraída de T. Y Tra es el ventor
% posición extraído de T.

% syms n_x s_x a_x p_x n_y s_y a_y p_y n_z s_z a_z p_z
% T_syms = [n_x s_x a_x p_x;n_y s_y a_y p_y;n_z s_z a_z p_z;0 0 0 1];
%
% T1 = (A(:,:,1))\T_syms == A(:,:,2)*A(:,:,3)




%inverseKinematics()

    ia = inv(A(:, :, 1));
    ter2 = A(:, :, 2) * A(:, :, 3);
    eq1 = ter2(3,4) == ia (3,1)*px + ia(3,2) *py;
    %q1 = solve(eq1,q1, 'real',true);

    ib = inv(A(:, :, 2))*ia;
    ter1 = A(:, :, 3);

    eq2 = ter1(2,4) == ib(2,1)*px + ib(2,2)*py + ib(2,3)*pz + ib(2,4)*1;
    eq3 = ter1(1,4) == ib(1,1)*px + ib(1,2)*py + ib(1,3)*pz + ib(1,4)*1;

    [q2,q3] = solve([eq2,eq3],[q2 q3], 'real', true);

    q1 = atan2(py,px);

    q2 = eval(q2(1));
    q3 = eval(q3(1));


    th = [q1,q2,q3];
    th2 = rad2deg(th);

    th2(1);
    th2(2);
    th2(3);


% fun = @(q) norm([Tra(1);Tra(2);Tra(3)]);


L1 =

    12


L2 =

    12
```

```
L3 =

    12

     3

[q1, 12, 0, pi/2]

[q2, 0, 12, 0]

[q3, 0, 12, 0]


(:,:,1) =

[cos(q1),        0,  sin(q1),          0]
[sin(q1),        0, -cos(q1),          0]
[      0,        1,        0,         12]
[      0,        0,        0,          1]

(:,:,2) =

[cos(q2), -sin(q2),        0, 12*cos(q2)]
[sin(q2),  cos(q2),        0, 12*sin(q2)]
[      0,        0,        1,          0]
[      0,        0,        0,          1]

(:,:,3) =

[cos(q3), -sin(q3),        0, 12*cos(q3)]
[sin(q3),  cos(q3),        0, 12*sin(q3)]
[      0,        0,        1,          0]
[      0,        0,        0,          1]

[cos(q2 + q3)*cos(q1), -sin(q2 + q3)*cos(q1),  sin(q1), 12*cos(q1)*(cos(q2 +
 q3) + cos(q2))]
[cos(q2 + q3)*sin(q1), -sin(q2 + q3)*sin(q1), -cos(q1), 12*sin(q1)*(cos(q2 +
 q3) + cos(q2))]
[        sin(q2 + q3),          cos(q2 + q3),        0,   12*sin(q2 + q3) +
 12*sin(q2) + 12]
[                   0,                     0,        0,
            1]

Warning: Solutions are only valid under certain conditions. To include
parameters and conditions in the solution, specify the 'ReturnConditions'
 value
as 'true'.
```

Published with MATLAB® R2022b