BMED436 Medical Robot
2018250064 Sibeen Kim

Inverse Kinematics

1 Workspace Analysis

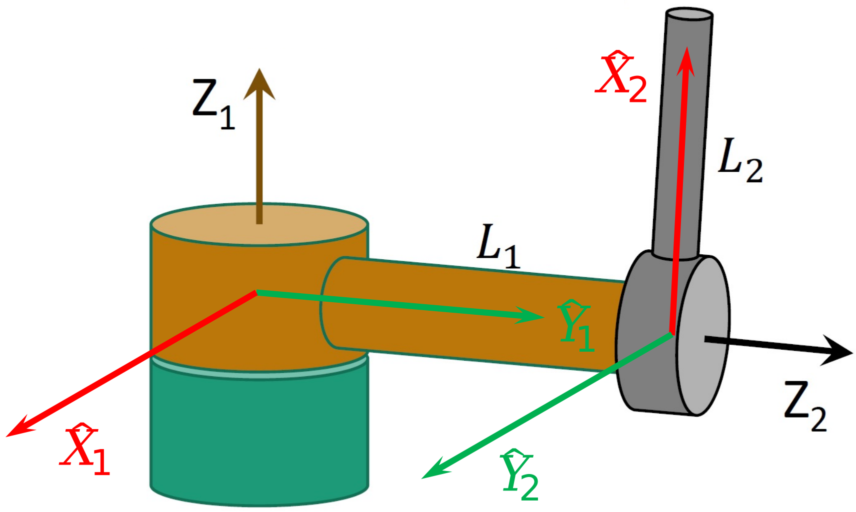

Figure 1: Link frames assigned to the manipulator.

Figure 1 shows the link frames of the manipulator. Listing 1 provides the MATLAB code used to compute the workspace of the end-effector. The workspace is shown in Figure 2, and it is based on the DH parameters from Table 1. .

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$-90°$	0	L_1	θ_2

Table 1: DH Parameters.

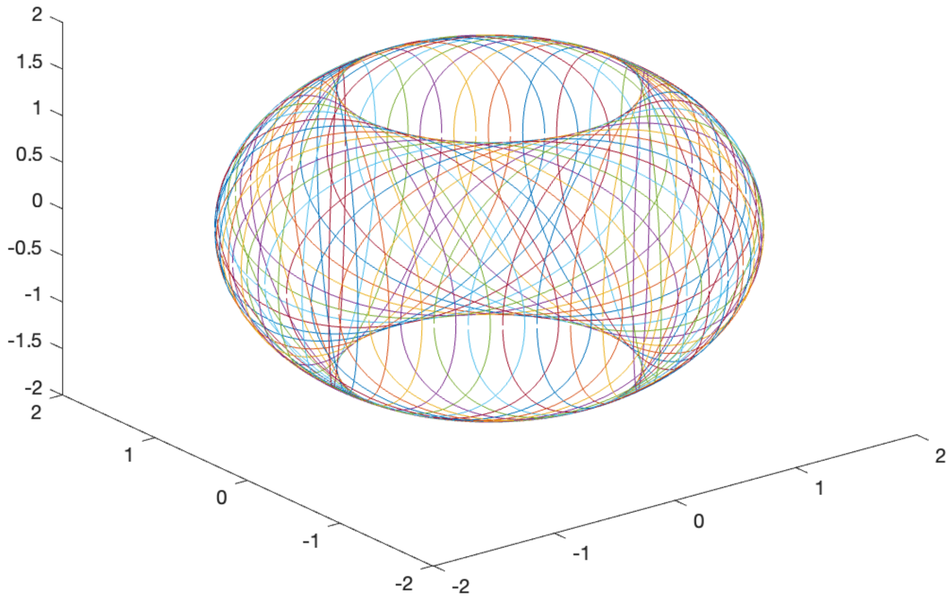

Figure 2: Workspace of the end-effector of the manipulator.

```
k1 = 50; k2 = 200;
L1 = 1; L2 = 1.5;
P = zeros(3, k2);
P2 = [L2 0 0 1]';

for i1 = 1:k1
    theta1 = 2*pi*(i1/k1);
    for i2 = 1:k2
        theta2 = 2*pi*(i2/k2);

        T1 = [
            cos(theta1) -sin(theta1) 0           0;
            sin(theta1)  cos(theta1) 0           0;
            0            0           1           0;
            0            0           0           1
            ];

        T2x = [
            1            0           0           0;
            0            cos(-pi/2) -sin(-pi/2)  0;
            0            sin(-pi/2)  cos(-pi/2)  0;
            0            0           0           1
            ];
        T2z = [
            cos(theta2) -sin(theta2) 0           0;
            sin(theta2)  cos(theta2) 0           0;
            0            0           1           L1;
            0            0           0           1
            ];
        T2 = T2x * T2z;

        P0 = T1 * T2 * P2;

        P(1:3, i2) = P0(1:3);
    end

    plot3(P(1, :), P(2, :), P(3, :))
    grid
    zlim([-2 2])
    hold on
end

hold off;
```

Listing 1: MATLAB code to compute and plot the workspace of the end-effector of the manipulator.

2 Inverse Kinematics Simulation

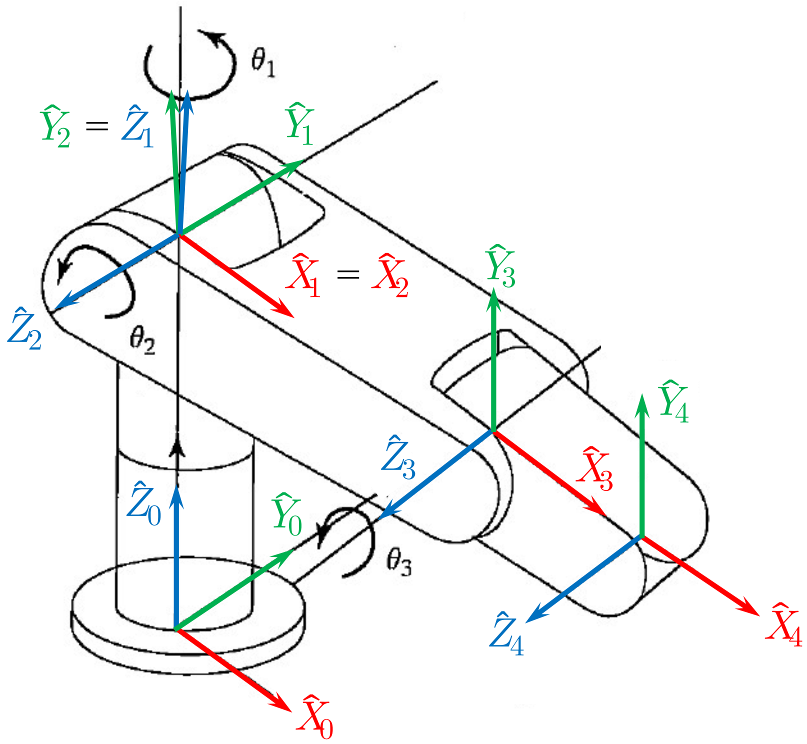

Figure 3: Link frame assignment.

Figure 3 illustrates the link frames assigned to the robotic arm, which has three degrees of freedom. Figure 4 shows the trajectory of the end-effector, plotted using the MATLAB code provided in Listing 2.

```
% End-effector trajectory parameters
x0 = 0.0; y0 = 0.0; z0 = L12; r = 0.4; N = 10;

for i = 0: 2*pi/N: 2*pi
    % End-effector position
    xd = x0 + cos(-pi/4)*(r*cos(i));
    yd = y0 + r*sin(i);
    zd = z0 - sin(-pi/4)*(r*cos(i));
```

Listing 2: End-effector trajectory.

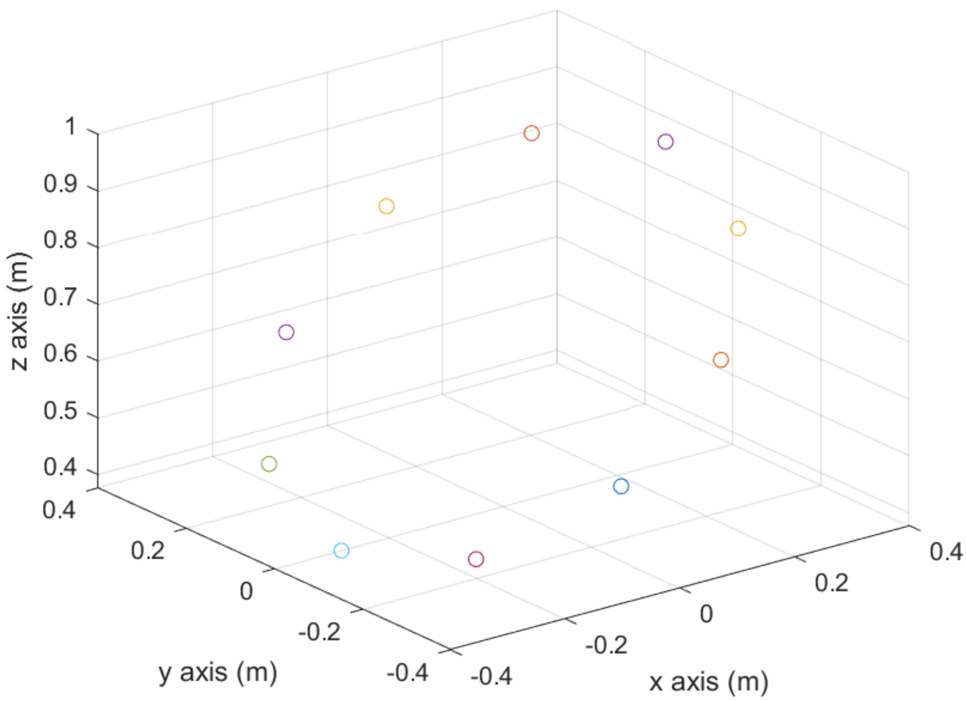

Figure 4: End-effector trajectory.

We aim to derive the joint angles required to reach the desired position of the end-effector through inverse kinematics. Figure 5 depicts the two possible configurations, from which the joint angles can be calculated as:

$$d = \sqrt{x_d^2 + y_d^2 + (z_d - (L_1 + L_2))^2} \tag{1}$$

$$\theta_3 = \pm \arccos\left(\frac{d^2 - L_3^2 - L_4^2}{2L_3L_4}\right) \tag{2}$$

$$\beta = \arctan\left(\frac{z_d - L_{12}}{\sqrt{x_d^2 + y_d^2}}\right) \tag{3}$$

$$\psi = \arccos\left(\frac{L_3^2 + d^2 - L_4^2}{2L_3 d}\right) \tag{4}$$

$$\theta_2 = \beta \mp \psi \tag{5}$$

$$\theta_1 = \arctan\left(\frac{y_d}{x_d}\right) \tag{6}$$

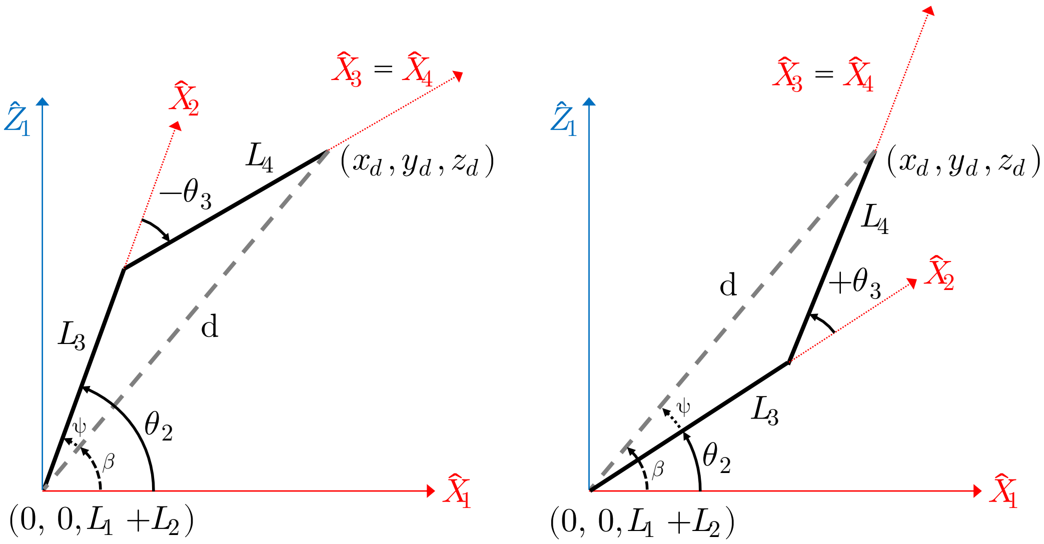

Figure 5: Inverse kinematics in the $\hat{X}_1$-$\hat{Z}_1$ plane. (L) $\theta_3 < 0$ (R) $\theta_3 > 0$.

Implementation of this inverse kinematics process is provided in Listing 3.

```
% Link parameters
L12 = 0.66; L3 = 0.43; L4 = 0.43;

% Multiple solutions
theta3_sign = -1; % -1 or 1

for i = 0: 2*pi/N: 2*pi
    ...
    d = sqrt(xd^2 + yd^2 + (zd-L12)^2);
    theta3 = theta3_sign * acos((d^2-L3^2-L4^2) / (2*L3*L4));

    beta   = atan2(zd-L12, sqrt(xd^2 + yd^2));
    psi    = acos((L3^2+d^2-L4^2) / (2*L3*d));
    theta2 = beta - theta3_sign * psi;

    theta1 = atan2(yd, xd);
```

Listing 3: Inverse kinematics.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$0°$	0	$L_1 + L_2$	θ_1
2	$90°$	0	0	θ_2
3	$0°$	L_3	0	θ_3
4	$0°$	L_4	0	$0°$

Table 2: DH Parameters.

Table 2 presents the Denavit-Hartenberg (DH) parameters, incorporating the derived joint angles. The corresponding transformation matrices are derived using these DH parameters:

$$
{}^0_1T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^1_2T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^2_3T = \begin{bmatrix} c_3 & -s_3 & 0 & L_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^3_4T = \begin{bmatrix} 1 & 0 & 0 & L_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0_2T = {}^0_1T\,{}^1_2T = \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & 0 \\ s_1c_2 & -s_1s_2 & -c_1 & 0 \\ s_2 & c_2 & 0 & L_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0_3T = {}^0_2T\,{}^2_3T = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & L_3c_1c_2 \\ s_1c_{23} & -s_1s_{23} & -c_1 & L_3s_1c_2 \\ s_{23} & c_{23} & 0 & L_{12} + L_3s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0_4T = {}^0_3T\,{}^3_4T = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(L_3c_2 + L_4c_{23}) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(L_3c_2 + L_4c_{23}) \\ s_{23} & c_{23} & 0 & L_{12} + L_3s_2 + L_4s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Joint positions:

$$r_1 = {}^0_1T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ L_{12} \\ 1 \end{bmatrix} \qquad r_2 = {}^0_2T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ L_{12} \\ 1 \end{bmatrix}$$

$$r_3 = {}^0_3T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} L_3c_1c_2 \\ L_3s_1c_2 \\ L_{12} + L_3s_2 \\ 1 \end{bmatrix} \qquad r_4 = {}^0_4T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1(L_3c_2 + L_4c_{23}) \\ s_1(L_3c_2 + L_4c_{23}) \\ L_{12} + L_3s_2 + L_4s_{23} \\ 1 \end{bmatrix}$$

In Listing 4, the positions of each joint of the robotic arm are computed via forward kinematics using the transformation matrices. Additional details, including the function `link_transform` (Listing 5) and the complete source code (Listing 6), are provided in the Supplementary Material.

```
for i = 0: 2*pi/N: 2*pi
    ...
    DH = [
        0    0  L12 theta1;
        pi/2 0  0   theta2;
        0    L3 0   theta3;
        0    L4 0   0
        ];
    T = link_transform(DH); % T(:, :, i): Frame i-1 to Frame i

    % T0i: Frame 0 to Frame i
    T01 = T(:, :, 1);
    T02 = T01 * T(:, :, 2);
    T03 = T02 * T(:, :, 3);
    T04 = T03 * T(:, :, 4);

    % ri(1:3): Origin of Frame i; ri(4): Dummy 1
    r0 = [0 0 0 1]';
    r2 = T02 * [0 0 0 1]'; % r1(1:3) == r2(1:3)
    r3 = T03 * [0 0 0 1]';
    r4 = T04 * [0 0 0 1]'; % r4(1:3) == [xd, yd, zd]

    % Robot body
    rx = [r0(1), r2(1), r3(1), r4(1)];
    ry = [r0(2), r2(2), r3(2), r4(2)];
    rz = [r0(3), r2(3), r3(3), r4(3)];
```

Listing 4: Forward kinematics.

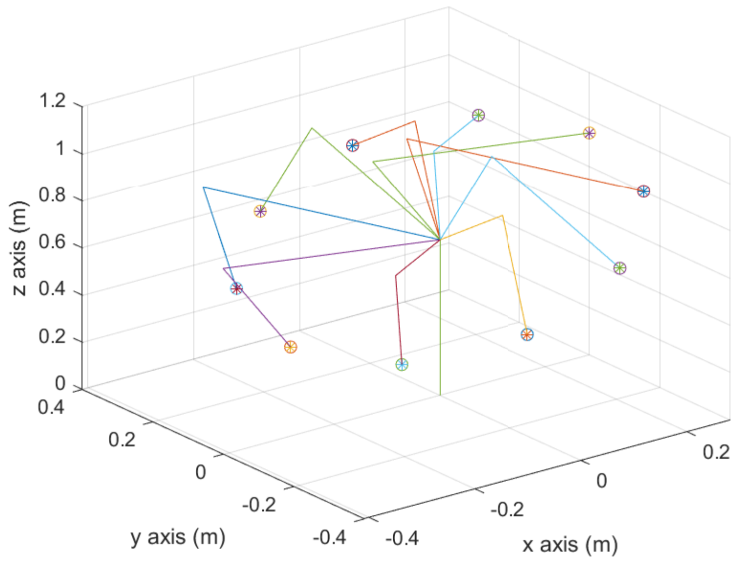

Figure 6: Three-link arm trajectory ($\theta_3 < 0$).

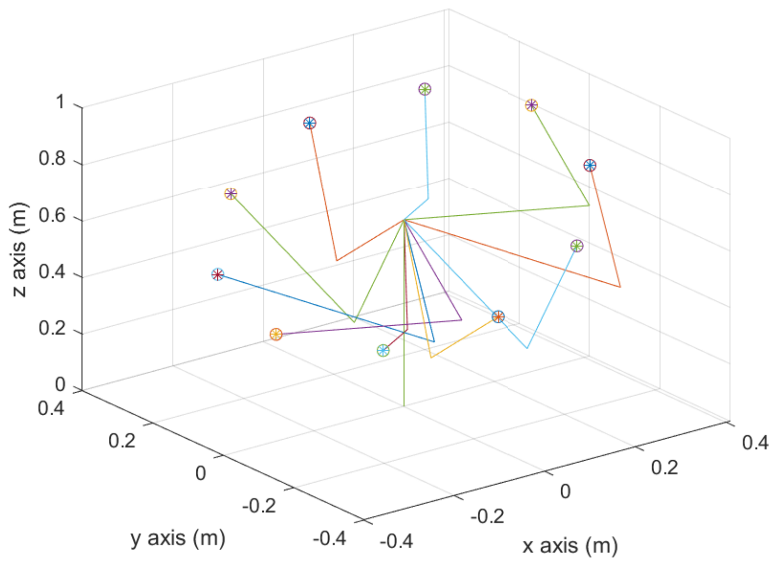

Figure 7: Three-link arm trajectory ($\theta_3 > 0$).

Supplementary Material

```
function T = link_transform(DH)
    N = size(DH, 1);
    T = zeros(4, 4, N);

    for i = 1: N
        screw_X = [
            1 0 0 DH(i,2);
            0 cos(DH(i,1)) -sin(DH(i,1)) 0;
            0 sin(DH(i,1)) cos(DH(i,1)) 0;
            0 0 0 1
            ];

        screw_Z = [
            cos(DH(i,4)) -sin(DH(i,4)) 0 0;
            sin(DH(i,4)) cos(DH(i,4)) 0 0;
            0 0 1 DH(i,3);
            0 0 0 1
            ];

        T(:, :, i) = screw_X * screw_Z;
    end
end
```

Listing 5: Computes transformation matrices using DH parameters.

```
clear; clf; clc; hold; view(3);

L12 = 0.66; L3 = 0.43; L4 = 0.43;
x0 = 0.0; y0 = 0.0; z0 = L12; r = 0.4; N = 10;
theta3_sign = -1; % -1 or 1

for i = 0: 2*pi/N: 2*pi
    xd = x0 + cos(-pi/4)*(r*cos(i));
    yd = y0 + r*sin(i);
    zd = z0 - sin(-pi/4)*(r*cos(i));
    plot3(xd, yd, zd, 'o');

    d = sqrt(xd^2 + yd^2 + (zd-L12)^2);
    theta3 = theta3_sign * acos((d^2-L3^2-L4^2) / (2*L3*L4));
    beta   = atan2(zd-L12, sqrt(xd^2 + yd^2));
    psi    = acos((L3^2+d^2-L4^2) / (2*L3*d));
    theta2 = beta - theta3_sign * psi;
    theta1 = atan2(yd, xd);

    DH = [
        0    0  L12 theta1;
        pi/2 0  0   theta2;
        0    L3 0   theta3;
        0    L4 0   0
        ];
    T = link_transform(DH);

    T01 = T(:, :, 1);
    T02 = T01 * T(:, :, 2);
    T03 = T02 * T(:, :, 3);
    T04 = T03 * T(:, :, 4);

    r0 = [0 0 0 1]';
    r2 = T02 * [0 0 0 1]';
    r3 = T03 * [0 0 0 1]';
    r4 = T04 * [0 0 0 1]';
    plot3(r4(1), r4(2), r4(3), '*');

    rx = [r0(1), r2(1), r3(1), r4(1)];
    ry = [r0(2), r2(2), r3(2), r4(2)];
    rz = [r0(3), r2(3), r3(3), r4(3)];
    plot3(rx, ry, rz, '-');
end

xlabel('x axis (m)'); ylabel('y axis (m)'); zlabel('z axis (m)')
grid;
```

Listing 6: Complete source code.