BMED436 Medical Robot
2018250064 Sibeen Kim

Velocity Kinematics

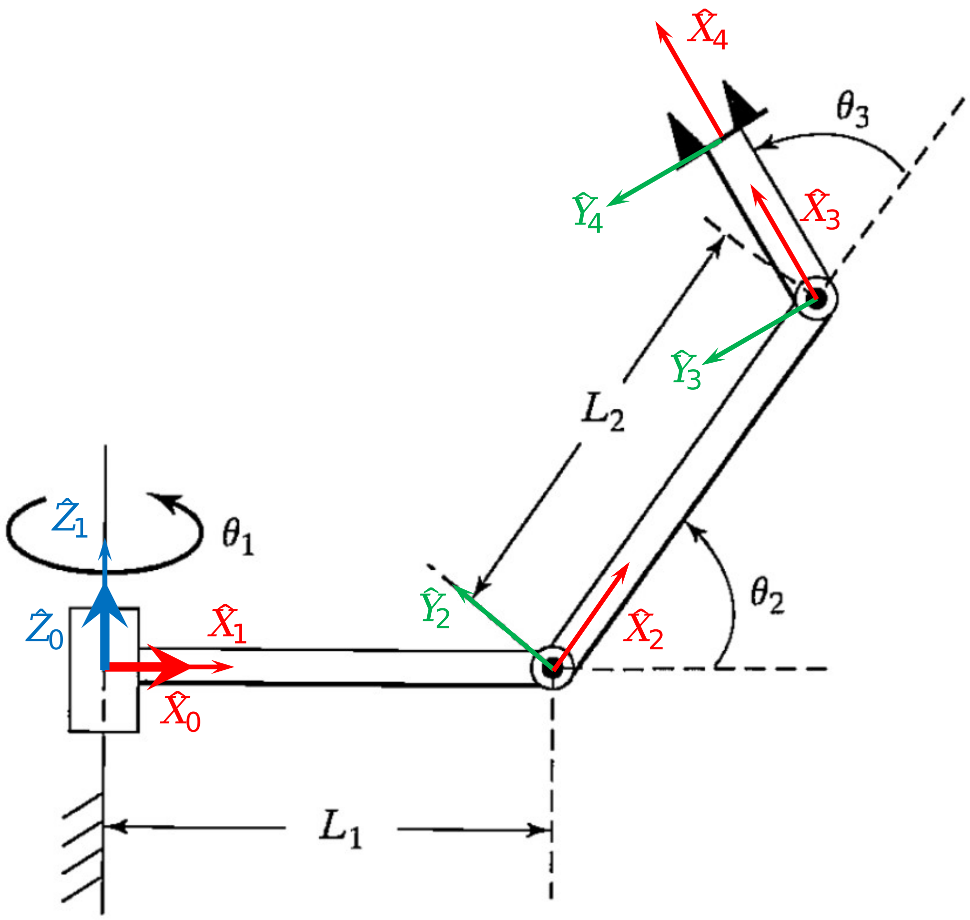

Figure 1: Link frame assignment.

Figure 1 illustrates an RRR manipulator with assigned link frames. Listing 1 provides MATLAB code to derive the transformation matrices from the DH parameters shown in Table 1. We will derive ${}^{4}J(\Theta)$ through three different methods: velocity propagation from the base to the tip, static force propagation from the tip to the base, and direct differentiation of the kinematic equations.

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

Table 1: DH Parameters.

```
syms th1 th2 th3 L1 L2 L3

DH = [
    0    0    0    th1;
    pi/2 L1   0    th2;
    0    L2   0    th3;
    0    L3   0    0
    ];

T = link_transform(DH);
N = size(T, 3);
```

Listing 1: Transform matrices. Source code for `link_transform` is provided in Listing 5 (Supplementary Material).

1 Velocity propagation from base to tip

To compute the velocity propagation through the manipulator, we start by defining the conditions at the base:

$$
{}^0v_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{1}
$$

$$
{}^0\omega_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{2}
$$

The velocity propagation from frame i to $i+1$ is defined by:

$$
{}^{i+1}v_{i+1} = {}^{i+1}_{i}R \left({}^iv_i + {}^i\,\omega_i \times {}^iP_{i+1} \right) \tag{3}
$$

$$
{}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\, {}^i\omega_i + \dot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1} \tag{4}
$$

These equations are iteratively applied from the base to the end-effector. Finally, the Jacobian, ${}^4J(\Theta)$, maps the joint velocities into the Cartesian velocity of the end-effector:

$$^4v_4 = {}^4J(\Theta)\dot{\Theta} \tag{5}$$

Thus, the Jacobian matrix $^4J(\Theta)$ is obtained as follows:

$$^4v_4 = \begin{bmatrix} L_2s_3\dot{\theta}_2 \\ L_2c_3\dot{\theta}_2 + L_3(\dot{\theta}_2 + \dot{\theta}_3) \\ -(L_1 + L_2c_2 + L_3c_{23})\dot{\theta}_1 \end{bmatrix} = {}^4J(\Theta)\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

$$^4J(\Theta) = \begin{bmatrix} 0 & L_2s_3 & 0 \\ 0 & L_2c_3 + L_3 & L_3 \\ -(L_1 + L_2c_2 + L_3c_{23}) & 0 & 0 \end{bmatrix}$$

The MATLAB code for this process is provided in Listing 2.

```
syms dth1 dth2 dth3

dth = [dth1; dth2; dth3; 0];

v = [0; 0; 0]; w = [0; 0; 0]; Z = [0; 0; 1];

for i = 1:N
    R = T(1:3, 1:3, i).';
    P = T(1:3, 4, i);
    v = R * (v + cross(w, P));
    w = R * w + dth(i) * Z;
end

J1 = jacobian(v, dth(1:3));
J1 = simplify(J1)
```

Listing 2: Velocity propagation from base to tip.

2 Static force propagation from tip to base

To compute the static force propagation through the manipulator, we start by defining the conditions at the tip:

$$^4f_4 = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T \tag{6}$$

$$^4n_4 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{7}$$

The static force propagation from frame $i+1$ to i is defined by:

$$ {}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} \tag{8}$$

$$ {}^{i}n_i = {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i \tag{9}$$

$$ \tau_i = {}^{i}n_i^T\,{}^{i}\hat{Z}_i \tag{10}$$

These equations are iteratively applied from the end-effector to the base. Finally, the transpose of the Jacobian matrix, ${}^{4}J^T(\Theta)$, maps the Cartesian forces acting at the end-effector into the equivalent joint torques:

$$ \tau = {}^{4}J^T(\Theta)\,{}^{4}f_4 \tag{11}$$

Thus, the Jacobian matrix ${}^{4}J(\Theta)$ is obtained as follows:

$$ \tau = \begin{bmatrix} -(L_1 + L_2c_2 + L_3c_{23})f_z \\ L_2(s_3f_x + c_3f_y) + L_3f_y \\ L_3f_y \end{bmatrix} = {}^{4}J^T(\Theta) \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} $$

$$ {}^{4}J(\Theta) = \begin{bmatrix} 0 & L_2s_3 & 0 \\ 0 & L_2c_3 + L_3 & L_3 \\ -(L_1 + L_2c_2 + L_3c_{23}) & 0 & 0 \end{bmatrix} $$

The MATLAB code for this process is provided in Listing 3.

```
syms fx fy fz

f = [fx; fy; fz];

t = sym(zeros(N, 1)); n = [0; 0; 0]; Z = [0; 0; 1];

for i = N:-1:1
    R = T(1:3, 1:3, i);
    P = T(1:3, 4, i);
    f = R * f;
    n = R * n + cross(P, f);
    t(i) = n.' * Z;
end

J2 = jacobian(t(N-2:N), [fx; fy; fz]).';
J2 = simplify(J2)
```

Listing 3: Static force propagation from tip to base.

3 Direct differentiation of kinematic equations

To derive the Jacobian through direct differentiation, we start by calculating the partial derivatives of the end-effector's position with respect to the joint angles:

$$ {}^{0}J(\Theta) = \frac{\partial({}^{0}P_4)}{\partial\Theta} \tag{12} $$

This results in the Jacobian expressed in the base frame. To transform this Jacobian into the end-effector frame, we apply the rotation matrix:

$$ {}^{4}J(\Theta) = {}^{4}_{0}R\,{}^{0}J(\Theta) \tag{13} $$

Thus, the Jacobian matrix ${}^{4}J(\Theta)$ is obtained as follows:

$$ {}^{4}_{0}R = \begin{bmatrix} c_1c_{23} & s_1c_{23} & s_{23} \\ -c_1s_{23} & -s_1s_{23} & c_{23} \\ s_1 & -c_1 & 0 \end{bmatrix} $$

$$ {}^{0}J(\Theta) = \begin{bmatrix} -s_1(L_1 + L_2c_2 + L_3c_{23}) & -c_1(L_2s_2 + L_3s_{23}) & -L_3c_1s_{23} \\ c_1(L_1 + L_2c_2 + L_3c_{23}) & -s_1(L_2s_2 + L_3s_{23}) & -L_3s_1s_{23} \\ 0 & L_2c_2 + L_3c_{23} & L_3c_{23} \end{bmatrix} $$

$$ {}^{4}J(\Theta) = \begin{bmatrix} 0 & L_2s_3 & 0 \\ 0 & L_2c_3 + L_3 & L_3 \\ -(L_1 + L_2c_2 + L_3c_{23}) & 0 & 0 \end{bmatrix} $$

The MATLAB code for this process is provided in Listing 4.

```
R = eye(3); P = [0; 0; 0; 1];

for i = N:-1:1
    R = R * T(1:3, 1:3, i).';
    P = T(:, :, i) * P;
end

J3 = R * jacobian(P(1:3), [th1 th2 th3]);
J3 = simplify(J3)
```

Listing 4: Direct differentiation of kinematic equations.

Supplementary Material

```matlab
function T = link_transform(DH)
    N = size(DH, 1);
    T = sym(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: Derives transformation matrices using DH parameters.

```matlab
clear; clc;

syms th1 th2 th3 L1 L2 L3

DH = [
    0     0     0     th1;
    pi/2 L1    0     th2;
    0     L2    0     th3;
    0     L3    0     0
    ];

T = link_transform(DH);
N = size(T, 3);

%% Velocity Propagation from Base to Tip

syms dth1 dth2 dth3

dth = [dth1; dth2; dth3; 0];

```

```
v = [0; 0; 0]; w = [0; 0; 0]; Z = [0; 0; 1];

for i = 1:N
    R = T(1:3, 1:3, i).';
    P = T(1:3, 4, i);
    v = R * (v + cross(w, P));
    w = R * w + dth(i) * Z;
end

J1 = jacobian(v, dth(1:3));
J1 = simplify(J1)

%% Static Force Propagation from Tip to Base

syms fx fy fz

f = [fx; fy; fz];

t = sym(zeros(N, 1)); n = [0; 0; 0]; Z = [0; 0; 1];

for i = N:-1:1
    R = T(1:3, 1:3, i);
    P = T(1:3, 4, i);
    f = R * f;
    n = R * n + cross(P, f);
    t(i) = n.' * Z;
end

J2 = jacobian(t(N-2:N), [fx; fy; fz]).';
J2 = simplify(J2)

%% Direct Differentiation of the Kinematic Equations

R = eye(3); P = [0; 0; 0; 1];

for i = N:-1:1
    R = R * T(1:3, 1:3, i).';
    P = T(:, :, i) * P;
end

J3 = R * jacobian(P(1:3), [th1 th2 th3]);
J3 = simplify(J3)
```

Listing 6: Complete source code.