Step-by-Step Construction of the Linear Velocity Jacobian for a 6DoF Robotic Arm

1 Introduction

The linear velocity Jacobian $\mathbf{J}_v \in \mathbb{R}^{3\times n}$ relates joint velocities to the linear velocity of the end-effector. For a 6DoF manipulator:

$$\mathbf{v} = \mathbf{J}_v \dot{\mathbf{q}} \tag{1}$$

where $\mathbf{v} \in \mathbb{R}^3$ is the end-effector linear velocity, $\dot{\mathbf{q}} \in \mathbb{R}^6$ is the joint velocity vector, and $\mathbf{J}_v$ is the 3×6 linear velocity Jacobian.

2 Prerequisites and Notation

Before constructing the Jacobian, ensure you have:

- Forward kinematics: $\mathbf{T}_0^6(\mathbf{q})$ (4×4 homogeneous transformation from base to end-effector)
- For each joint i: Position $\mathbf{p}_i$ and axis of rotation $\mathbf{z}_i$ (or translation direction for prismatic joints)
- End-effector position: $\mathbf{p}_e$ (typically extracted from $\mathbf{T}_0^6$)

Notation:

- $\mathbf{z}_i$: Unit vector along joint i axis (in base frame)
- $\mathbf{p}_i$: Position of joint i origin (in base frame)
- $\mathbf{p}_e$: Position of end-effector (in base frame)
- Superscript denotes reference frame, subscript denotes which joint/link

3 Step-by-Step Construction

3.1 Step 1: Extract Frame Information

For each joint $i = 1, 2, \ldots, 6$:

1. Compute the homogeneous transformation $\mathbf{T}_0^i$ from the base frame to frame i
2. Extract the rotation matrix: $\mathbf{R}_0^i = \mathbf{T}_0^i(1:3, 1:3)$
3. Extract the position: $\mathbf{p}_i = \mathbf{T}_0^i(1:3, 4)$

4. Extract the z-axis: $\mathbf{z}_i = \mathbf{R}_0^i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{R}_0^i(:,3)$

The end-effector position is:

$$\mathbf{p}_e = \mathbf{T}_0^6(1:3,4) = \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} \tag{2}$$

3.2 Step 2: Determine Joint Types

For each joint, identify whether it is:

- **Revolute**: Rotation about an axis
- **Prismatic**: Translation along an axis

This determines the formula for each Jacobian column.

3.3 Step 3: Construct Each Column

The linear velocity Jacobian is constructed column-by-column:

$$\mathbf{J}_v = \begin{bmatrix} \mathbf{J}_{v,1} & \mathbf{J}_{v,2} & \mathbf{J}_{v,3} & \mathbf{J}_{v,4} & \mathbf{J}_{v,5} & \mathbf{J}_{v,6} \end{bmatrix} \tag{3}$$

For each joint i:

3.3.1 Revolute Joint

The i-th column represents the contribution of joint i's angular velocity to the end-effector's linear velocity:

$$\mathbf{J}_{v,i} = \mathbf{z}_i \times (\mathbf{p}_e - \mathbf{p}_i) \tag{4}$$

Geometric interpretation: When joint i rotates with angular velocity $\dot{q}_i$, points farther from the joint axis have higher linear velocities. The cross product gives the direction and magnitude of this velocity contribution.

3.3.2 Prismatic Joint

The i-th column is simply the direction of translation:

$$\mathbf{J}_{v,i} = \mathbf{z}_i \tag{5}$$

Geometric interpretation: When joint i extends with velocity $\dot{d}_i$, the end-effector moves linearly in the direction of $\mathbf{z}_i$.

3.4 Step 4: Compute Cross Products (for Revolute Joints)

For revolute joint i, compute the cross product using:

$$\mathbf{z}_i \times (\mathbf{p}_e - \mathbf{p}_i) = \begin{bmatrix} z_{i,y}(p_{e,z} - p_{i,z}) - z_{i,z}(p_{e,y} - p_{i,y}) \\ z_{i,z}(p_{e,x} - p_{i,x}) - z_{i,x}(p_{e,z} - p_{i,z}) \\ z_{i,x}(p_{e,y} - p_{i,y}) - z_{i,y}(p_{e,x} - p_{i,x}) \end{bmatrix} \tag{6}$$

Alternatively, use the skew-symmetric matrix:

$$\mathbf{z}_i \times (\mathbf{p}_e - \mathbf{p}_i) = [\mathbf{z}_i]_\times (\mathbf{p}_e - \mathbf{p}_i) \tag{7}$$

where

$$[\mathbf{z}_i]_\times = \begin{bmatrix} 0 & -z_{i,z} & z_{i,y} \\ z_{i,z} & 0 & -z_{i,x} \\ -z_{i,y} & z_{i,x} & 0 \end{bmatrix} \tag{8}$$

3.5 Step 5: Assemble the Complete Jacobian

Combine all columns into the final 3×6 matrix:

$$\mathbf{J}_v = \begin{bmatrix} | & | & | & | & | & | \\ \mathbf{J}_{v,1} & \mathbf{J}_{v,2} & \mathbf{J}_{v,3} & \mathbf{J}_{v,4} & \mathbf{J}_{v,5} & \mathbf{J}_{v,6} \\ | & | & | & | & | & | \end{bmatrix} \tag{9}$$

4 Algorithm Summary

```
function J_v = compute_linear_velocity_jacobian(T_0_to_6, joint_types)
    % Input: T_0_to_6 - forward kinematics at current configuration
    %        joint_types - array indicating 'R' or 'P' for each joint

    % Extract end-effector position
    p_e = T_0_to_6(1:3, 4)

    % Initialize Jacobian
    J_v = zeros(3, 6)

    % For each joint
    for i = 1:6
        % Compute T_0_to_i
        T_0_to_i = forward_kinematics_to_joint(i)

        % Extract z-axis and position
        z_i = T_0_to_i(1:3, 3)
        p_i = T_0_to_i(1:3, 4)

        % Compute Jacobian column
        if joint_types(i) == 'R'  % Revolute
            J_v(:, i) = cross(z_i, p_e - p_i)
        else  % Prismatic
```

```
            J_v(:, i) = z_i
        end
    end

    return J_v
end
```

5 Worked Example: 6R Manipulator

Consider a 6DoF manipulator with all revolute joints at configuration $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^T$.

Suppose the forward kinematics yields:

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{z}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{10}$$

$$\mathbf{p}_2 = \begin{bmatrix} 0 \\ 0 \\ L_1 \end{bmatrix}, \quad \mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{11}$$

$$\vdots \tag{12}$$

$$\mathbf{p}_e = \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} \tag{13}$$

The first column (joint 1, revolute):

$$\mathbf{J}_{v,1} = \mathbf{z}_1 \times (\mathbf{p}_e - \mathbf{p}_1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} -y_e \\ x_e \\ 0 \end{bmatrix} \tag{14}$$

Continue for all 6 joints to complete $\mathbf{J}_v$.

6 Practical Considerations

6.1 Frame Consistency

All vectors must be expressed in the same reference frame (typically base frame). If DH parameters are used, ensure proper frame transformations.

6.2 Singularities

At singular configurations, $\text{rank}(\mathbf{J}_v) < 3$, meaning certain end-effector velocities cannot be achieved.

6.3 Numerical Computation

When implementing numerically:

- Use robust cross product implementations
- Consider numerical differentiation as a verification: $\mathbf{J}_v \approx \frac{\partial \mathbf{p}_e}{\partial \mathbf{q}}$
- Check condition number to detect near-singular configurations

6.4 Verification

Verify your Jacobian by:

1. Numerical differentiation: Perturb each joint and compare $\Delta\mathbf{p}_e$ with $\mathbf{J}_v\Delta\mathbf{q}$
2. Check dimensions: Should be 3×6 for a 6DoF arm
3. Test at known configurations (e.g., zero configuration)
4. Ensure consistency with full Jacobian if computing separately

7 Relationship to Full Geometric Jacobian

The full geometric Jacobian includes both linear and angular velocity:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_v \\ \mathbf{J}_\omega \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{15}$$

where the angular velocity Jacobian for revolute joints is:

$$\mathbf{J}_{\omega,i} = \mathbf{z}_i \tag{16}$$

and for prismatic joints:

$$\mathbf{J}_{\omega,i} = \mathbf{0} \tag{17}$$

8 References and Further Reading

- Craig, J.J., *Introduction to Robotics: Mechanics and Control*
- Siciliano, B., et al., *Robotics: Modelling, Planning and Control*
- Murray, R.M., et al., *A Mathematical Introduction to Robotic Manipulation*