FOR Notes

GIOCATORI SERI

February 10, 2026

Chapter 1

Introduction

1.1 Syllabus

- Robotica industriale e robotica avanzata
- Descrizione e principi di funzionamento di un robot
- Cinematica diretta
- Calibrazione cinematica
- Cinematica differenziale e Jacobiano
- Ridondanza e singolarità
- Algoritmi per l'inversione cinematica
- Dualità cineto-statica
- Pianificazione di traiettorie nello spazio dei giunti e nello spazio operativo
- Attuatori e sensori
- Unità di governo
- Modello Lagrangiano
- Proprietà notevoli del modello dinamico
- Algoritmo ricorsivo di Newton-Eulero
- Identificazione dei parametri dinamici
- Dinamica diretta e dinamica inversa
- Controllo decentralizzato
- Controllo indipendente ai giunti
- Controllo centralizzato
- Controllo a coppia precalcolata

- Controllo PD con compensazione di gravità
- Controllo a dinamica inversa
- Controllo robusto e adattativo
- Controllo nello spazio operativo

Contents

Chapter 2

Kinematics

2.1 Kinematics

2.1.1 Position Representation

The position of the origin of a mobile coordinate frame O' with respect to a fixed reference frame O is defined by the vector:

$$\underline{O'} = o'_x\underline{x} + o'_y\underline{y} + o'_z\underline{z} = \begin{bmatrix} o'_x \\ o'_y \\ o'_z \end{bmatrix} \tag{2.1}$$

2.1.2 Orientation Representation

The orientation is described by the unit vectors (versors) of the mobile frame axes projected onto the axes of the fixed frame:

$$\underline{x}' = x'_x\underline{x} + x'_y\underline{y} + x'_z\underline{z} \tag{2.2}$$

The rotation matrix R is constructed from these column vectors:

$$R = \begin{bmatrix} \underline{x}' & \underline{y}' & \underline{z}' \end{bmatrix} = \begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{bmatrix} = \begin{bmatrix} \underline{x}'^T\underline{x} & \underline{y}'^T\underline{x} & \underline{z}'^T\underline{x} \\ \underline{x}'^T\underline{y} & \underline{y}'^T\underline{y} & \underline{z}'^T\underline{y} \\ \underline{x}'^T\underline{z} & \underline{y}'^T\underline{z} & \underline{z}'^T\underline{z} \end{bmatrix} \tag{2.3}$$

2.1.3 Properties of the Rotation Matrix $SO(m)$

For a matrix to belong to the Special Orthonormal group $SO(m)$, it must satisfy the following mathematical constraints:

- **Orthonormality:** The basis vectors must be unit length and mutually orthogonal:

$$\underline{x}'^T\underline{y}' = 0, \quad \underline{x}'^T\underline{x}' = 1 \tag{2.4}$$

- **Inverse and Transpose:** For orthogonal matrices, the inverse is equivalent to the transpose:

$$R^TR = I \implies R^T = R^{-1} \tag{2.5}$$

- **Right-Handed Convention:** The determinant of the matrix must be positive unity to preserve the orientation of the space:

$$\det(R) = 1 \tag{2.6}$$

Note: The dot product between two vectors is defined as:

$$x' \cdot x = x'^T x \tag{2.7}$$

2.2 Elementary Rotation Matrices

Elementary rotations are transformations where the rotation occurs around one of the principal axes of the reference frame. These matrices belong to the $SO(3)$ group.

2.2.1 Rotation about the z-axis

A rotation by an angle α around the z-axis is described by:

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

2.2.2 Rotation about the y-axis

A rotation by an angle β around the y-axis is described by:

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.9}$$

Note: The signs of the sine terms are swapped compared to R_z and R_x to maintain the right-handed convention.

2.2.3 Rotation about the x-axis

A rotation by an angle γ around the x-axis is described by:

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{2.10}$$

2.2.4 General Properties

For any elementary rotation matrix $R(\theta)$:

- $R(0) = I$ (Identity matrix)
- $R(-\theta) = R(\theta)^T = R(\theta)^{-1}$

2.3 Applications of the Rotation Matrix

The rotation matrix R serves two primary mathematical purposes in robotics: rotating a vector within the same frame or transforming the coordinates of a point between two different frames.

2.3.1 Rotation of a Vector

A rotation matrix can be used as an operator to rotate a vector $\underline{p}$ by a certain amount within a fixed reference frame, resulting in a new vector $\underline{p}'$:

$$\underline{p}' = R\underline{p} \tag{2.11}$$

In this case, both $\underline{p}$ and $\underline{p}'$ are expressed in the same coordinate system.

2.3.2 Change of Reference Frame (Coordinate Transformation)

When considering a point P in space and two frames, 0 (fixed) and 1 (mobile), the matrix R_1^0 maps the coordinates of P relative to frame 1 ($\underline{p}^1$) to its coordinates relative to frame 0 ($\underline{p}^0$):

$$\underline{p}^0 = R_1^0\underline{p}^1 \tag{2.12}$$

2.3.3 Inverse Transformation

To perform the inverse mapping (from the fixed frame back to the mobile frame), we use the property $R^{-1} = R^T$:

$$\underline{p}^1 = (R_1^0)^{-1}\underline{p}^0 = (R_1^0)^T\underline{p}^0 \tag{2.13}$$

2.3.4 Mapping of Vectors

If we have a vector $\underline{v}$ expressed in the mobile frame as $\underline{v} = v'_x\underline{x}' + v'_y\underline{y}' + v'_z\underline{z}'$, its representation in the fixed frame is:

$$\underline{v} = \begin{bmatrix}\underline{x}' & \underline{y}' & \underline{z}'\end{bmatrix}\begin{bmatrix}v'_x\\ v'_y\\ v'_z\end{bmatrix} = R\underline{v}' \tag{2.14}$$

2.4 Composition of Rotations

To obtain the final orientation of a body after multiple rotations, we multiply the individual rotation matrices. The order of multiplication depends on the reference frame.

2.4.1 Rules for Composition

- **Fixed Frame (Current is stationary):** If the rotation is performed with respect to the fixed (global) frame, we **pre-multiply** the existing matrix:

$$R_{final} = R_{new} \cdot R_{previous} \tag{2.15}$$

- **Current Frame (Relative):** If the rotation is performed with respect to the current (local) moving frame, we **post-multiply**:

$$R_{final} = R_{previous} \cdot R_{new} \tag{2.16}$$

2.5 Euler Angles

2.5.1 Definition

Euler angles are a set of three independent parameters (ϕ, θ, ψ) used to describe any orientation of a rigid body in a 3D Euclidean space. According to Euler's rotation theorem, any rotation can be represented as a sequence of three elementary rotations around the axes of a coordinate system.

2.5.2 Combinations and Sequences

There are a total of **12 possible combinations** of elementary rotations. These are divided into two categories:

- **Proper Euler Angles (Classic):** The first and third rotations are about the same axis (e.g., ZYZ, ZXZ, XYX, XZX, YXY, YZY). There are 6 such sequences.
- **Tait-Bryan Angles:** Rotations are about three distinct axes (e.g., XYZ, ZYX, XZY, YXZ, YZX, ZXY). These are often called Roll-Pitch-Yaw angles. There are 6 such sequences.

2.5.3 Useful Sequences in Robotics

The most relevant sequences in robotics and aerospace are:

- **ZYZ:** Widely used in the description of spherical wrists in industrial manipulators.
- **ZYX (Roll-Pitch-Yaw):** Standard in aerospace and mobile robotics to describe the orientation of vehicles.

2.5.4 ZYZ Convention: Rotation Sequence

The total rotation is obtained by three consecutive rotations relative to the **current** frames:

1. Rotate about the z-axis by ϕ
2. Rotate about the current y'-axis by θ
3. Rotate about the current z''-axis by ψ

$$R(\phi, \theta, \psi) = R_z(\phi) R_y(\theta) R_z(\psi) \tag{2.17}$$

2.5.5 Resulting Matrix

The symbolic multiplication yields:

$$R_{ZYZ} = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{2.18}$$

Note: $c_x = \cos(x)$ *and* $s_x = \sin(x)$.

2.5.6 Inverse Problem: Analytical Derivation

To find the angles (ϕ, θ, ψ) that correspond to a given numerical rotation matrix R, we compare the elements of R with the symbolic matrix R_{ZYZ}:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{2.19}$$

Step 1: Finding θ From the element r_{33}, we have:

$$r_{33} = \cos\theta \tag{2.20}$$

From the elements r_{13} and r_{23}, we can write:

$$r_{13}^2 + r_{23}^2 = (c_\phi s_\theta)^2 + (s_\phi s_\theta)^2 = s_\theta^2(c_\phi^2 + s_\phi^2) = \sin^2\theta \tag{2.21}$$

Therefore:

$$\sin\theta = \pm\sqrt{r_{13}^2 + r_{23}^2} \tag{2.22}$$

By choosing the positive root (consistent with the convention $\theta \in [0, \pi]$), we obtain:

$$\theta = \text{atan2}\left(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}\right) \tag{2.23}$$

Step 2: Finding ϕ Using the elements r_{13} and r_{23} again:

$$r_{13} = \cos\phi \sin\theta \tag{2.24}$$
$$r_{23} = \sin\phi \sin\theta \tag{2.25}$$

Assuming $\sin\theta \neq 0$, we can solve for ϕ:

$$\phi = \text{atan2}(r_{23}, r_{13}) \tag{2.26}$$

Step 3: Finding ψ Finally, using the elements r_{31} and r_{32}:

$$r_{31} = -\sin\theta\cos\psi \tag{2.27}$$
$$r_{32} = \sin\theta\sin\psi \tag{2.28}$$

Again, assuming $\sin\theta \neq 0$:

$$\psi = \text{atan2}(r_{32}, -r_{31}) \tag{2.29}$$

Degenerate Case (Singularity) If $\sin\theta = 0$ (i.e., $r_{13} = r_{23} = 0$), the orientation is in a singularity called *Gimbal Lock*. In this case, $\theta = 0$ or $\theta = \pi$, and only the sum or difference of ϕ and ψ can be identified from the matrix.

2.5.7 Numerical Implementation: atan2 vs. atan

In the inverse kinematic derivation, the function $\text{atan2}(y, x)$ is used instead of the standard $\arctan(y/x)$ for several critical analytical reasons:

- **Quadrant Identification:** The standard arctan function has a range of $(-\pi/2, \pi/2)$, meaning it cannot distinguish between opposite quadrants (e.g., I vs III or II vs IV). $\text{atan2}(y, x)$ uses the signs of both arguments to identify the correct quadrant within $(-\pi, \pi]$.
- **Handling Vertical Slopes:** When $x = 0$, the ratio y/x is undefined, causing a division-by-zero error in a standard calculator. atan2 handles this case internally, returning $\pm\pi/2$ based on the sign of y.
- **Numerical Stability:** In robotic control, where angles are extracted in real-time, atan2 provides the necessary robustness to avoid jumps in the calculated joint positions when the denominator is near zero.

2.6 Angle and Axis (3-Axis) Representation

To address the degeneracies of Euler angles, an orientation can be described by a rotation of an angle θ around a generic axis in space defined by the unit vector $\underline{r} = [r_x, r_y, r_z]^T$, where $r_x^2 + r_y^2 + r_z^2 = 1$.

2.6.1 Geometric Derivation

The rotation matrix $R(\theta, \underline{r})$ is derived by aligning the arbitrary axis $\underline{r}$ with the reference z-axis through two rotations (α and β), performing the rotation θ, and reversing the alignment:

$$R(\theta, \underline{r}) = R_z(\alpha)R_y(\beta)R_z(\theta)R_y(-\beta)R_z(-\alpha) \tag{2.30}$$

The alignment angles are calculated as follows:

$$\sin\alpha = \frac{r_y}{\sqrt{r_x^2 + r_y^2}}, \quad \cos\alpha = \frac{r_x}{\sqrt{r_x^2 + r_y^2}} \tag{2.31}$$

$$\sin\beta = \sqrt{r_x^2 + r_y^2}, \quad \cos\beta = r_z \tag{2.32}$$

2.6.2 Inverse Problem: Extracting Angle and Axis

Given a rotation matrix R, the parameters θ and $\underline{r}$ are determined by:

$$\theta = \arccos\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{2.33}$$

$$\underline{r} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{2.34}$$

Note: As indicated in the notes, for $\theta = 0$ or $\theta = \pi$ there are no physical singularities in the representation, although the specific formula for $\underline{r}$ above is not well-defined at these points (division by zero).

2.7 Unit Quaternions (Euler Parameters)

Since the Axis-Angle representation still requires 4 parameters (θ and the 3 components of $\underline{r}$) and suffers from numerical issues when $\sin\theta = 0$, we transition to **Unit Quaternions**. This representation is more robust and numerically stable.

2.7.1 Definition and Relation to Axis-Angle

A quaternion $\boldsymbol{q} = \{\eta, \underline{\epsilon}\}$ consists of a scalar part η and a vector part $\underline{\epsilon} = [\epsilon_x, \epsilon_y, \epsilon_z]^T$:

$$\eta = \cos\left(\frac{\theta}{2}\right) \quad \text{(Scalar part)} \tag{2.35}$$

$$\underline{\epsilon} = \sin\left(\frac{\theta}{2}\right)\underline{r} \quad \text{(Vector part)} \tag{2.36}$$

The unit norm constraint must be satisfied:

$$\eta^2 + \epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2 = 1 \tag{2.37}$$

2.7.2 Properties

- **Identity:** The quaternion $\boldsymbol{q} = \{1, [0,0,0]^T\}$ represents zero rotation (Identity matrix).
- **Antipodal Map:** Both $\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same rotation matrix R. This is because a rotation of θ about $\underline{r}$ is identical to a rotation of $-\theta$ about $-\underline{r}$.

2.7.3 Inverse Problem: Detailed Derivation

The extraction of the quaternion parameters is performed by comparing the elements of the numerical rotation matrix R with the symbolic matrix $R(\eta, \underline{\epsilon})$.

Derivation of the Scalar Part η The derivation starts by analyzing the sum of the diagonal elements of the rotation matrix (the trace). Using the symbolic form $r_{ii} = 2(\eta^2 + \epsilon_i^2) - 1$:

$$r_{11} + r_{22} + r_{33} = [2(\eta^2 + \epsilon_x^2) - 1] + [2(\eta^2 + \epsilon_y^2) - 1] + [2(\eta^2 + \epsilon_z^2) - 1] \tag{2.38}$$

$$= 6\eta^2 + 2\epsilon_x^2 + 2\epsilon_y^2 + 2\epsilon_z^2 - 3 \tag{2.39}$$

By rearranging the terms to exploit the unit norm constraint $(\eta^2 + \epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2) = 1$:

$$r_{11} + r_{22} + r_{33} = 2(\eta^2 + \epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2) + 4\eta^2 - 3 \tag{2.40}$$

$$= 2(1) + 4\eta^2 - 3 \tag{2.41}$$

$$= 4\eta^2 - 1 \tag{2.42}$$

Solving for η:

$$\eta = \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1} \tag{2.43}$$

Derivation of the Vector Part $\underline{\epsilon}$ The components of the vector part $\underline{\epsilon}$ are obtained by subtracting the off-diagonal symmetric elements of R:

$$r_{32} - r_{23} = (2\epsilon_y\epsilon_z + 2\eta\epsilon_x) - (2\epsilon_y\epsilon_z - 2\eta\epsilon_x) = 4\eta\epsilon_x \tag{2.44}$$
$$r_{13} - r_{31} = (2\epsilon_x\epsilon_z + 2\eta\epsilon_y) - (2\epsilon_x\epsilon_z - 2\eta\epsilon_y) = 4\eta\epsilon_y \tag{2.45}$$
$$r_{21} - r_{12} = (2\epsilon_x\epsilon_y + 2\eta\epsilon_z) - (2\epsilon_x\epsilon_y - 2\eta\epsilon_z) = 4\eta\epsilon_z \tag{2.46}$$

Which yields the extraction formulas:

$$\epsilon_x = \frac{r_{32} - r_{23}}{4\eta}, \quad \epsilon_y = \frac{r_{13} - r_{31}}{4\eta}, \quad \epsilon_z = \frac{r_{21} - r_{12}}{4\eta} \tag{2.47}$$

Note on Numerical Robustness and Sign As shown above, if $\eta \to 0$ (which happens for rotations near π), the division by η becomes unstable. To maintain robustness, the final implementation often extracts the magnitudes from the diagonal elements and uses the sgn function to determine relative signs:

$$\epsilon_x = \frac{1}{2}\text{sgn}(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \tag{2.48}$$
$$\epsilon_y = \frac{1}{2}\text{sgn}(r_{13} - r_{31})\sqrt{r_{22} - r_{11} - r_{33} + 1} \tag{2.49}$$
$$\epsilon_z = \frac{1}{2}\text{sgn}(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \tag{2.50}$$

2.8 Homogeneous Transformations

To describe the full pose (position and orientation) of a rigid body in space, we combine rotation and translation into a single mathematical entity using **homogeneous coordinates**.

2.8.1 Homogeneous Transformation Matrix

Following the professor's notation, the transformation from a mobile frame j to a frame i is described by the 4×4 matrix A_j^i:

$$A_j^i = \begin{bmatrix} R_j^i & \underline{o}_j^i \\ \underline{0}^T & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & o_x \\ n_y & s_y & a_y & o_y \\ n_z & s_z & a_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.51}$$

Where:

- R_j^i is the 3×3 rotation matrix.
- $\underline{o}_j^i$ is the 3×1 position vector of the origin of frame j relative to frame i.
- The unit vectors $\underline{n}, \underline{s}, \underline{a}$ represent the Normal, Sliding, and Approach directions of the mobile frame.

2.8.2 Coordinate Transformation

Given a point P, its coordinates in the reference frame i are computed from its coordinates in the mobile frame j as:

$$\underline{\tilde{p}}^i = A^i_j \underline{\tilde{p}}^j \tag{2.52}$$

Which, in expanded form, corresponds to:

$$\begin{bmatrix} \underline{p}^i \\ 1 \end{bmatrix} = \begin{bmatrix} R^i_j & \underline{o}^i_j \\ \underline{0}^T & 1 \end{bmatrix} \begin{bmatrix} \underline{p}^j \\ 1 \end{bmatrix} = \begin{bmatrix} R^i_j \underline{p}^j + \underline{o}^i_j \\ 1 \end{bmatrix} \tag{2.53}$$

2.8.3 Inverse Transformation

The inverse transformation $A^j_i = (A^i_j)^{-1}$ is calculated using the property of block matrices:

$$(A^i_j)^{-1} = \begin{bmatrix} (R^i_j)^T & -(R^i_j)^T \underline{o}^i_j \\ \underline{0}^T & 1 \end{bmatrix} \tag{2.54}$$

2.8.4 Composition of Transformations

Sequential transformations are obtained through matrix multiplication (Chain Rule). For a sequence of frames $0, 1, \ldots, n$:

$$A^0_n = A^0_1 A^1_2 \ldots A^{n-1}_n \tag{2.55}$$

Note: The order of multiplication follows the same rules as rotation matrices (pre-multiplication for fixed axes, post-multiplication for current axes).

2.9 Direct Kinematics

The goal of direct kinematics is to determine the pose (position and orientation) of the robot's end-effector relative to a fixed base coordinate frame, given the joint variables (angles for revolute joints, displacements for prismatic joints).

2.9.1 Problem Definition

For an n-degree-of-freedom manipulator, let $\underline{q} = [q_1, q_2, \ldots, q_n]^T$ be the vector of joint variables. The direct kinematics function $\mathcal{K}(\cdot)$ maps the joint space to the operational space:

$$\underline{x} = \mathcal{K}(\underline{q}) \tag{2.56}$$

Where $\underline{x}$ represents the transformation matrix A^0_n:

$$A^0_n(\underline{q}) = A^0_1(q_1) A^1_2(q_2) \ldots A^{n-1}_n(q_n) \tag{2.57}$$

2.9.2 The Systematic Approach

While the chain of transformations is conceptually simple, calculating each A^{i-1}_i manually for complex robots is prone to errors. This necessitates a systematic convention to:

- Define a consistent coordinate frame for each link.

- Minimize the number of parameters needed to describe each transformation.

This systematic approach is provided by the **Denavit-Hartenberg (D-H) Convention**.

2.10 Denavit-Hartenberg (D-H) Convention

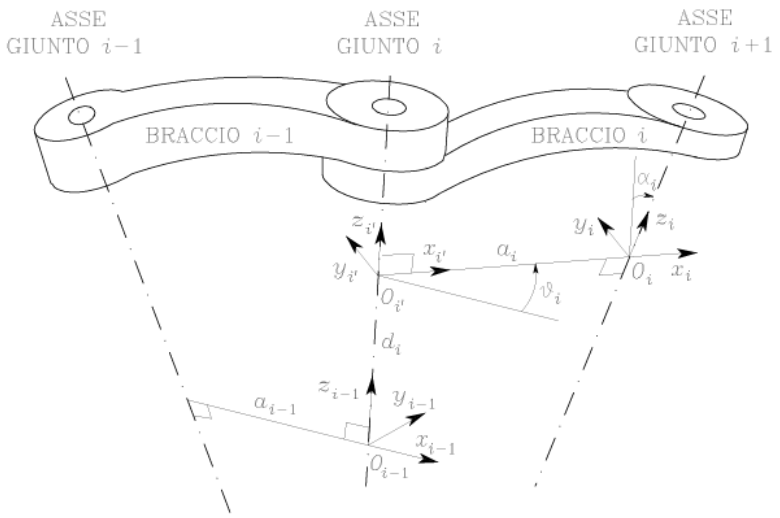


Figure 2.1: D-H Convention.

The D-H convention provides a systematic method to describe the kinematics of an n-link manipulator by assigning a coordinate frame to each link and defining four parameters to describe the transformation between consecutive frames.

2.10.1 Frame Assignment Procedure

To define the frame i relative to the link i, the following steps must be performed:

1. **Axis z_i:** Choose the axis z_i along the axis of joint $i+1$.

2. **Origin O_i:** Locate the origin O_i at the intersection of axis z_i with the **common normal** to axes z_{i-1} and z_i.

3. **Axis x_i:** Choose axis x_i along the common normal to axes z_{i-1} and z_i, directed from joint i to joint $i+1$.

4. **Axis y_i:** Choose axis y_i to complete a right-handed frame (following the cross product $z_i \times x_i$).

Note: The common normal of two skewed axes is the shortest segment connecting them.

2.10.2 The Four Parameters

The transformation A_i^{i-1} is defined by four parameters describing the relation between frame $i-1$ and frame i:

- a_i (Link length): Distance between O_{i-1} and O_i measured along x_i.
- d_i (Link offset): Coordinate of O_i along z_{i-1}.
- α_i (Link twist): Angle between axes z_{i-1} and z_i about x_i (positive counter-clockwise).
- θ_i (Joint angle): Angle between axes x_{i-1} and x_i about axis z_{i-1} (positive counter-clockwise).

2.10.3 Joint Variables

- **Revolute Joint:** θ_i is the variable ($q_i = \theta_i$).
- **Prismatic Joint:** d_i is the variable ($q_i = d_i$).

2.10.4 Decomposition of the D-H Transformation

The transformation from frame $i-1$ to frame i is decomposed into two main steps.

Step 1: From $i-1$ to an intermediate frame i' This step accounts for the joint displacement (θ_i and d_i). We translate the $i-1$ frame by d_i along z_{i-1} and rotate it by θ_i about z_{i-1}:

$$A_{i'}^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.58}$$

Step 2: From i' to i This step accounts for the link geometry (a_i and α_i). We translate the intermediate frame by a_i along x_i and rotate it by α_i about x_i:

$$A_i^{i'} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.59}$$

2.10.5 Resulting Transformation Matrix

The total transformation A_i^{i-1} is the product of the two matrices $A_{i'}^{i-1}A_i^{i'}$:

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.60}$$

2.10.6 Frame Assignment Procedure

To ensure a unique and consistent kinematic model, the following procedure must be followed:

1. **Joint Axis Enumeration:** Identify and enumerate the joint axes from $1, \ldots, n$ and associate each axis i with z_{i-1}.

2. **Base Frame:** Define the base frame (O_0, x_0, y_0, z_0) at joint 1.

3. **Origin O_i Selection:** Locate the origin O_i at the intersection between z_i and the common normal to axes z_{i-1} and z_i.

   - **Parallel Axes Case:** If z_{i-1} and z_i are parallel:
     - If joint i is **revolute**, locate O_i such that $d_i = 0$.
     - If joint i is **prismatic**, locate O_i at a reference position for the joint range.

4. **Axis x_i Selection:** Choose axis x_i along the common normal to axes z_{i-1} and z_i, directed from z_{i-1} to z_i.

5. **Axis y_i Selection:** Choose y_i to obtain a right-handed frame (cross product $z_i \times x_i$).

6. **The n-th (End-Effector) Frame:**

   - If joint n is **revolute**, align z_n with z_{n-1}.
   - Otherwise, if n is **prismatic**, choose z_n arbitrarily (usually following the direction of motion).

2.11 Closed Kinematic Chains

While serial manipulators consist of a single path of links and joints, a **closed kinematic chain** occurs when a sequence of links forms a loop. This structure introduces geometric constraints because the end of two different branches must coincide at a specific joint.

2.11.1 Kinematic Constraints

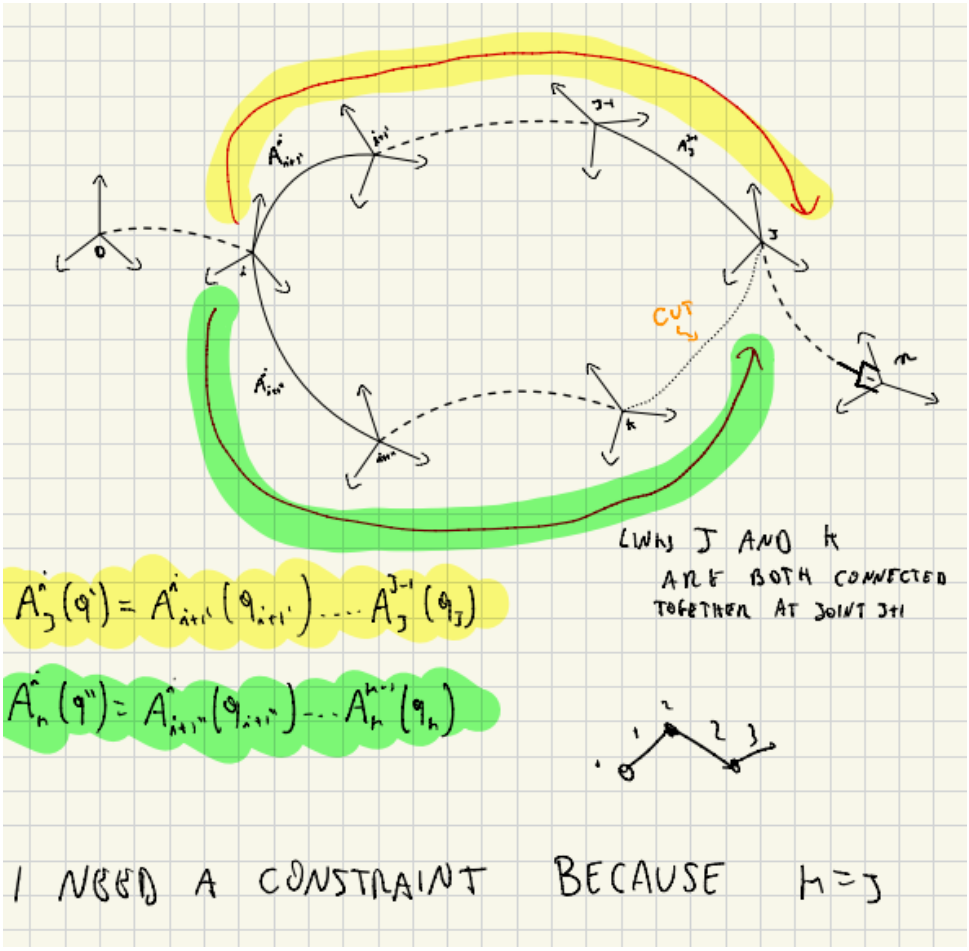


Figure 2.2: Closed chains.

As shown in the figure, if we consider a loop starting from frame i and splitting into two branches that reconnect at joint $j+1$, we can define two different paths for the transformation:

- **Branch 1 (Yellow Path):** The transformation from frame i to frame j is given by:
$$A_j^i(q') = A_{i+1}^i(q_{i+1}) \dots A_j^{j-1}(q_j) \tag{2.61}$$
- **Branch 2 (Green Path):** An alternative path leads to the same connection point:
$$A_k^i(q'') = A_{i+1'}^i(q_{i+1'}) \dots A_k^{k-1}(q_k) \tag{2.62}$$

2.11.2 Closure Equations

Since the two branches are physically connected at the same location, the position and orientation described by both paths must be consistent. This leads to the **closure constraint**:

$$A_j^i(q') = A_k^i(q'') \quad \text{where } k = j \tag{2.63}$$

These equations imply that the joint variables $\underline{q}$ are no longer independent. The number of independent degrees of freedom (DoF) is reduced by the number of constraints imposed by the closed loop.

2.11.3 Specific Kinematic Constraints for Loop Closure

To ensure the physical continuity of the closed chain, the reference frames at the cut (frame j and frame k) must satisfy several geometric constraints.

1^o **Constraint: Axis Alignment** The first requirement is that the z-axes of the two frames must coincide:

$$z_j^i(q') = z_k^i(q'') \quad \text{(Same } z\text{-axis)} \tag{2.64}$$

2^o **Constraint: Orientation Alignment** For a prismatic joint, the relative orientation between the frames is constrained. The projection of the x-axes must account for the joint angle θ_{jk}:

$$x_j^{iT}(q')x_k^i(q'') = \cos\theta_{jk} \tag{2.65}$$

3^o **Constraint: Position Vector** The distance between the origins of the two frames, when projected into the j-th frame, must match the joint's geometric parameters (like the link offset d_{jk}):

$$R_i^j(q')\left(p_j^i(q') - p_k^i(q'')\right) = \begin{bmatrix} 0 \\ 0 \\ d_{jk} \end{bmatrix} \tag{2.66}$$

Note: The rotation matrix R_i^j projects the distance vector into the local j-frame coordinates.

2.11.4 Distinction between Rotoid and Prismatic Joints

The nature of the constraint equations changes depending on the joint type at the cut:

- **Rotoid Joint:** The displacement d_{jk} remains constant.
- **Prismatic Joint:** The displacement d_{jk} becomes the joint variable. In this case, the position constraint simplifies to 2 equations by ensuring the origins lie on the same sliding axis:

$$\begin{bmatrix} x_j^{iT}(q') \\ y_j^{iT}(q') \end{bmatrix}\left(p_j^i(q') - p_k^i(q'')\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.67}$$

2.12 Closed Kinematic Chains: Procedure and Constraints

To analyze a closed kinematic chain, we follow a systematic procedure to define the geometric constraints that keep the loop closed.

2.12.1 General Procedure

1. **Virtual Cut:** Identify a non-actuated joint and "cut" it to transform the closed loop into an open tree structure.

2. **Kinematic Modeling:** Compute the homogeneous transformation matrices using the D-H convention for the two resulting branches.

3. **Constraint Identification:** Define the kinematic constraints for the two frames (let's call them j and n) that are physically connected by the cut joint.

2.12.2 Kinematic Loop Closure Equations

The consistency between the two paths (Branch 1 with variables q' and Branch 2 with variables q'') is enforced by three fundamental constraints.

1° **Constraint: Axis Alignment** The z-axes of the two frames must remain aligned:

$$z_j^i(q') = z_n^i(q'') \quad \text{(Same } z\text{-axis)} \tag{2.68}$$

2° **Constraint: Relative Orientation** Specifically for prismatic joints, the orientation of the x-axes must account for the fixed or variable joint angle θ_{jn}:

$$x_j^{iT}(q')x_n^i(q'') = \cos\theta_{jn} \tag{2.69}$$

3° **Constraint: Position Vector** The vector connecting the origins of the two frames, when projected into the local frame j, must satisfy the joint's translation:

$$R_i^j(q')\left(p_j^i(q') - p_n^i(q'')\right) = \begin{bmatrix} 0 \\ 0 \\ d_{jn} \end{bmatrix} \tag{2.70}$$

Note: The rotation matrix $R_i^j = (R_j^i)^T$ is used to project the distance vector into the local j-frame coordinates.

2.12.3 Joint-Specific Constraints

The mathematical form of the constraints depends on whether the joint at the cut $(j+1)$ is rotational or prismatic.

Case 1: Rotational Joint If the joint is rotational, the offset d_{jn} is constant. The system consists of:

$$\begin{cases} R_i^j(q')\left(p_j^i(q') - p_n^i(q'')\right) = [0 \quad 0 \quad d_{jn}]^T \\ z_j^i(q') = z_n^i(q'') \end{cases} \tag{2.71}$$

Case 2: Prismatic Joint If the joint is prismatic, the displacement d_{jn} is variable. Therefore, the distance constraint only applies to the x and y components (ensuring the origins lie on the same sliding axis), reducing it to 2 equations:

$$\begin{cases} \begin{bmatrix} x_j^{iT}(q') \\ y_j^{iT}(q') \end{bmatrix} \left(p_j^i(q') - p_n^i(q'')\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ z_j^i(q') = z_n^i(q'') \\ x_j^{iT}(q') x_n^i(q'') = \cos\theta_{jn} \end{cases} \tag{2.72}$$

2.13 Inverse Kinematics

The inverse kinematics problem consists of determining the joint coordinates $\underline{q}$ required to obtain a desired end-effector pose (position and orientation).

2.13.1 Difficulties and Characteristics

Unlike direct kinematics, the inverse problem is generally more complex due to:

- **Non-linear Equations:** The relationship between joints and pose involves transcendental functions.
- **Multiple Solutions:** Different joint configurations can lead to the same end-effector pose.
- **Infinite Solutions:** Occurs in redundant manipulators (more degrees of freedom than required).
- **Non-Admissible Solutions:** Some poses may be outside the robot's workspace or lead to joint limit violations.

2.14 Differential Kinematics

Differential kinematics describes the relationship between the joint velocities $\dot{\underline{q}}$ and the corresponding end-effector linear velocity $\dot{\underline{p}}_e$ and angular velocity $\underline{\omega}_e$.

2.14.1 The Geometric Jacobian

The fundamental tool for this mapping is the **Jacobian matrix** $J(\underline{q})$. Given the transformation matrix:

$$T(\underline{q}) = \begin{bmatrix} R(\underline{q}) & \underline{p}(\underline{q}) \\ \underline{0}^T & 1 \end{bmatrix} \tag{2.73}$$

The velocities are related by:

$$\underline{v}_e = \begin{bmatrix} \dot{\underline{p}}_e \\ \underline{\omega}_e \end{bmatrix} = J(\underline{q})\dot{\underline{q}} \tag{2.74}$$

Where J is a $(6 \times n)$ matrix composed of two blocks:

- J_P $(3 \times n)$: Linear velocity Jacobian.
- J_O $(3 \times n)$: Angular velocity Jacobian.

2.14.2 Derivative of the Rotation Matrix

To derive the Jacobian, we analyze the time derivative of the rotation matrix $R(t)$. Since $R(t)R^T(t) = I$:

$$\dot{R}(t)R^T(t) + R(t)\dot{R}^T(t) = 0 \implies S(t) + S^T(t) = 0 \tag{2.75}$$

This defines the **Skew-Symmetric Matrix** $S(\underline{\omega}(t))$, which relates $\dot{R}$ to the angular velocity $\underline{\omega}$:

$$\dot{R}(t) = S(\underline{\omega}(t))R(t) \tag{2.76}$$

Where S is defined as:

$$S = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \implies S(\underline{\omega}) \tag{2.77}$$

2.14.3 Example: Rotation about the z-axis

To verify the relationship $\dot{R} = S(\underline{\omega})R$, let's consider a rotation about the z-axis by an angle $\alpha(t)$:

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.78}$$

Computing the skew-symmetric matrix $S = \dot{R}R^T$:

$$S = \begin{bmatrix} -\dot{\alpha}\sin\alpha & -\dot{\alpha}\cos\alpha & 0 \\ \dot{\alpha}\cos\alpha & -\dot{\alpha}\sin\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -\dot{\alpha} & 0 \\ \dot{\alpha} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.79}$$

This confirms that for a rotation about z, the angular velocity vector is $\underline{\omega} = [0, 0, \dot{\alpha}]^T$, such that $\omega_z = \dot{\alpha}$.

2.14.4 Velocity of a Point and Transportation Term

Consider a point P whose position in the fixed frame 0 is given by $\underline{p}^0 = \underline{o}_1^0 + R_1^0\underline{p}^1$. Differentiating with respect to time:

$$\dot{\underline{p}}^0 = \dot{\underline{o}}_1^0 + R_1^0\dot{\underline{p}}^1 + \dot{R}_1^0\underline{p}^1 \tag{2.80}$$

Substituting $\dot{R}_1^0 = S(\underline{\omega}_1^0)R_1^0$:

$$\dot{\underline{p}}^0 = \dot{\underline{o}}_1^0 + R_1^0\dot{\underline{p}}^1 + S(\underline{\omega}_1^0)R_1^0\underline{p}^1 \tag{2.81}$$

Defining $\underline{r}_1^0 = R_1^0\underline{p}^1$ as the position of P relative to the mobile origin expressed in the fixed frame (representing rotation but not translation), the expression becomes:

$$\dot{\underline{p}}^0 = \dot{\underline{o}}_1^0 + R_1^0\dot{\underline{p}}^1 + \underline{\omega}_1^0 \times \underline{r}_1^0 \tag{2.82}$$

The term $\underline{\omega}_1^0 \times \underline{r}_1^0$ is the **Transportation Term**.

2.14.5 Algebraic Cross Product Representation

The skew-symmetric operator allows computing the cross product $\underline{\omega} \times \underline{b}$ through matrix multiplication:

$$S(\underline{\omega})\underline{b} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} -\omega_z b_y + \omega_y b_z \\ \omega_z b_x - \omega_x b_z \\ -\omega_y b_x + \omega_x b_y \end{bmatrix} \tag{2.83}$$

This matches the formal determinant expansion:

$$\underline{\omega} \times \underline{b} = \det \begin{bmatrix} \underline{i} & \underline{j} & \underline{k} \\ \omega_x & \omega_y & \omega_z \\ b_x & b_y & b_z \end{bmatrix} \tag{2.84}$$

2.15 Velocity of a Rigid Body

2.15.1 Linear Velocity of a Link

Consider a generic link i in a kinematic chain. Its position relative to the base frame is determined by the recursive relation:

$$\underline{p}_i = \underline{p}_{i-1} + R_{i-1}\underline{r}_{i-1,i}^{i-1} \tag{2.85}$$

Where $\underline{r}_{i-1,i}^{i-1}$ is the vector from the origin of frame $i-1$ to frame i, expressed in frame $i-1$. By differentiating this expression:

$$\dot{\underline{p}}_i = \dot{\underline{p}}_{i-1} + R_{i-1}\dot{\underline{r}}_{i-1,i}^{i-1} + \dot{R}_{i-1}\underline{r}_{i-1,i}^{i-1} \tag{2.86}$$

Applying the skew-symmetric operator $\dot{R} = S(\underline{\omega})R$:

$$\dot{\underline{p}}_i = \dot{\underline{p}}_{i-1} + R_{i-1}\dot{\underline{r}}_{i-1,i}^{i-1} + S(\underline{\omega}_{i-1})R_{i-1}\underline{r}_{i-1,i}^{i-1} \tag{2.87}$$

Defining $\underline{r}_{i-1,i} = R_{i-1}\underline{r}_{i-1,i}^{i-1}$, we obtain the recursive linear velocity formula:

$$\dot{\underline{p}}_i = \dot{\underline{p}}_{i-1} + \underline{v}_{i-1,i} + \underline{\omega}_{i-1} \times \underline{r}_{i-1,i} \tag{2.88}$$

Where $\underline{v}_{i-1,i}$ represents the velocity of frame i with respect to frame $i-1$.

2.15.2 Angular Velocity of a Link

The total orientation of link i is the composition of orientations $R_i = R_{i-1}R_i^{i-1}$. Differentiating with respect to time yields:

$$\dot{R}_i = \dot{R}_{i-1}R_i^{i-1} + R_{i-1}\dot{R}_i^{i-1} \tag{2.89}$$

Substituting the skew-symmetric representations $S(\underline{\omega}_i)R_i$ and utilizing the property $RS(\underline{\omega})R^T = S(R\underline{\omega})$:

$$S(\underline{\omega}_i)R_i = S(\underline{\omega}_{i-1})R_i + S(R_{i-1}\underline{\omega}_{i-1,i}^{i-1})R_i \tag{2.90}$$

Extracting the angular velocity vectors, we obtain the recursive form:

$$\underline{\omega}_i = \underline{\omega}_{i-1} + R_{i-1}\underline{\omega}_{i-1,i}^{i-1} = \underline{\omega}_{i-1} + \underline{\omega}_{i-1,i} \tag{2.91}$$

where $\underline{\omega}_{i-1,i}$ is the angular velocity of frame i with respect to frame $i-1$ expressed in the base frame.

2.15.3 In Total

The recursive kinematics for the entire chain can be summarized as:

$$\begin{cases} \underline{\omega}_i = \underline{\omega}_{i-1} + \underline{\omega}_{i-1,i} \\ \underline{\dot{p}}_i = \underline{\dot{p}}_{s-1} + \underline{v}_{i-1,i} + \underline{\omega}_{i-1} \times \underline{r}_{i-1,i} \end{cases} \tag{2.92}$$

2.16 Jacobian Computation

The Jacobian J maps joint velocities to end-effector velocities. It is composed of n columns, where each column refers to a joint and is divided into two 3×1 vectors referring to linear and angular motion:

$$J = \begin{bmatrix} J_{P1} & \dots & J_{Pn} \\ J_{O1} & \dots & J_{On} \end{bmatrix} \tag{2.93}$$

2.16.1 Angular Velocity Contribution

- **Prismatic joint**: Since it does not rotate, $\dot{q}_i \underline{J}_{Oi} = 0 \implies \underline{J}_{Oi} = 0$.
- **Revolute joint**: The rotation occurs about the axis $\underline{z}_{i-1}$, thus $\dot{q}_i \underline{J}_{Oi} = \dot{\theta}_i \underline{z}_{i-1} \implies \underline{J}_{Oi} = \underline{z}_{i-1}$.

2.16.2 Linear Velocity Contribution

- **Prismatic joint**: The translation occurs along the axis, $\dot{q}_i \underline{J}_{Pi} = \dot{d}_i \underline{z}_{i-1} \implies \underline{J}_{Pi} = \underline{z}_{i-1}$.
- **Revolute joint**: We need to find the linear velocity of the end-effector ($\underline{p}_e$) due to the revolute joint i:

$$\dot{q}_i \underline{J}_{Pi} = \underline{\omega}_{i-1,i} \times \underline{r}_{i-1,e} = (\dot{\theta}_i \underline{z}_{i-1}) \times (\underline{p}_e - \underline{p}_{i-1}) \tag{2.94}$$

  Which leads to the column entry:

$$\underline{J}_{Pi} = \underline{z}_{i-1} \times (\underline{p}_e - \underline{p}_{i-1}) \tag{2.95}$$

2.16.3 Final Formulation

The complete geometric Jacobian components for each joint i are:

$$\begin{bmatrix} \underline{J}_{Pi} \\ \underline{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} \underline{z}_{i-1} \\ 0 \end{bmatrix} & \text{Prismatic Joint} \\ \begin{bmatrix} \underline{z}_{i-1} \times (\underline{p}_e - \underline{p}_{i-1}) \\ \underline{z}_{z-1} \end{bmatrix} & \text{Revolute Joint} \end{cases} \tag{2.96}$$

From forward kinematics, we can identify:

- $\underline{z}_{i-1} = R_1^0(q_1) \dots R_{i-1}^{i-2}(q_{i-1}) \underline{z}_0$
- $\underline{p}_e = A_1^0(q_1) \dots A_n^{n-1}(q_n) \underline{p}_0$

As a result, **we can compute the Jacobian without using derivatives**.

2.17 Example: 3 Links Planar Manipulator

Consider a planar manipulator with three revolute joints moving in the xy-plane.

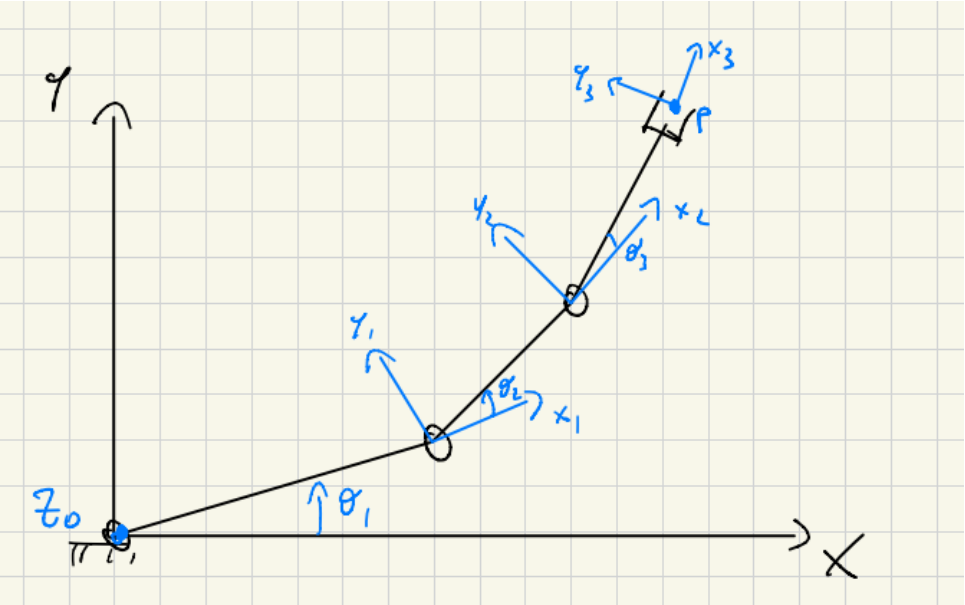


Figure 2.3: 3 Links Planar Manipulator.

The Jacobian matrix $J(q)$ for this 3-DOF robot is defined column by column as:

$$J(q) = \begin{bmatrix} \underline{z}_0 \times (\underline{p}_3 - \underline{p}_0) & \underline{z}_1 \times (\underline{p}_3 - \underline{p}_1) & \underline{z}_2 \times (\underline{p}_3 - \underline{p}_2) \\ \underline{z}_0 & \underline{z}_1 & \underline{z}_2 \end{bmatrix} \tag{2.97}$$

2.17.1 Geometric Definitions

From the manipulator's geometry and forward kinematics, we identify the following vectors:

- **Joint axes**: Since the motion is planar, all rotation axes are parallel to z:

$$\underline{z}_0 = \underline{z}_1 = \underline{z}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.98}$$

- **Joint positions**:
  - $\underline{p}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$
  - $\underline{p}_1 = \begin{bmatrix} a_1 C_1 & a_1 S_1 & 0 \end{bmatrix}^T$
  - $\underline{p}_2 = \begin{bmatrix} a_1 C_1 + a_2 C_{12} & a_1 S_1 + a_2 S_{12} & 0 \end{bmatrix}^T$
  - $\underline{p}_3 = \begin{bmatrix} a_1 C_1 + a_2 C_{12} + a_3 C_{123} & a_1 S_1 + a_2 S_{12} + a_3 S_{123} & 0 \end{bmatrix}^T$

2.17.2 Calculation of Linear Velocity Columns (J_P)

We compute the cross products $\underline{z}_{i-1} \times (\underline{p}_3 - \underline{p}_{i-1})$ using the determinant form:

First Column ($i = 1$):

$$\underline{z}_0 \times (\underline{p}_3 - \underline{p}_0) = \begin{vmatrix} \underline{i} & \underline{j} & \underline{k} \\ 0 & 0 & 1 \\ p_{3x} & p_{3y} & 0 \end{vmatrix} = \begin{bmatrix} -p_{3y} \\ p_{3x} \\ 0 \end{bmatrix} = \begin{bmatrix} -a_1 S_1 - a_2 S_{12} - a_3 S_{123} \\ a_1 C_1 + a_2 C_{12} + a_3 C_{123} \\ 0 \end{bmatrix} \tag{2.99}$$

Second Column ($i = 2$):

$$\underline{z}_1 \times (\underline{p}_3 - \underline{p}_1) = \begin{vmatrix} \underline{i} & \underline{j} & \underline{k} \\ 0 & 0 & 1 \\ p_{3x} - p_{1x} & p_{3y} - p_{1y} & 0 \end{vmatrix} = \begin{bmatrix} -(p_{3y} - p_{1y}) \\ p_{3x} - p_{1x} \\ 0 \end{bmatrix} = \begin{bmatrix} -a_2 S_{12} - a_3 S_{123} \\ a_2 C_{12} + a_3 C_{123} \\ 0 \end{bmatrix} \tag{2.100}$$

Third Column ($i = 3$):

$$\underline{z}_2 \times (\underline{p}_3 - \underline{p}_2) = \begin{vmatrix} \underline{i} & \underline{j} & \underline{k} \\ 0 & 0 & 1 \\ p_{3x} - p_{2x} & p_{3y} - p_{2y} & 0 \end{vmatrix} = \begin{bmatrix} -(p_{3y} - p_{2y}) \\ p_{3x} - p_{2x} \\ 0 \end{bmatrix} = \begin{bmatrix} -a_3 S_{123} \\ a_3 C_{123} \\ 0 \end{bmatrix} \tag{2.101}$$

2.17.3 Final Jacobian Matrix

Assembling the linear (J_P) and angular (J_O) parts, the 6×3 Geometric Jacobian is:

$$J = \begin{bmatrix} -a_1 S_1 - a_2 S_{12} - a_3 S_{123} & -a_2 S_{12} - a_3 S_{123} & -a_3 S_{123} \\ a_1 C_1 + a_2 C_{12} + a_3 C_{123} & a_2 C_{12} + a_3 C_{123} & a_3 C_{123} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.102}$$

2.18 Example: Anthropomorphic Manipulator

Consider a 3-DOF anthropomorphic arm. This configuration consists of a revolute joint rotating about the base z-axis, followed by two revolute joints with parallel axes.

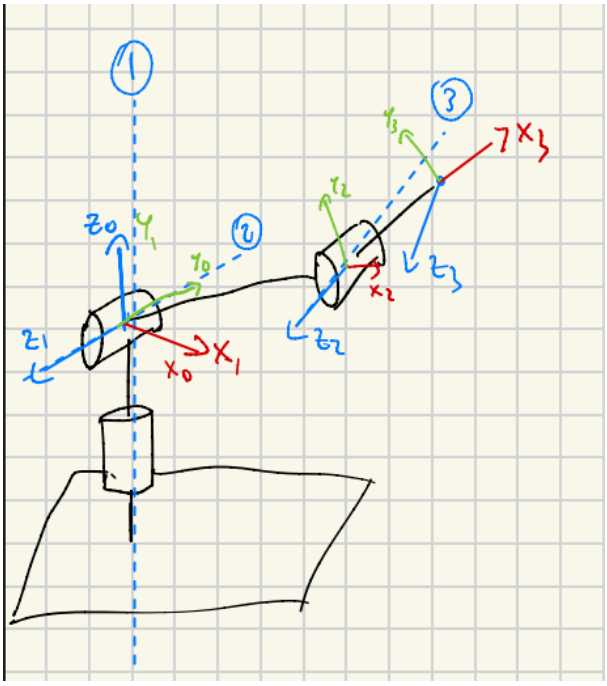


Figure 2.4: Anthropomorphic Manipulator.

2.18.1 Geometric Definitions

From the Denavit-Hartenberg frames or direct inspection of the geometry, we identify the following vectors:

- **Joint axes**:
  - $\underline{z}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ (Rotation about the base)
  - $\underline{z}_1 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix}$ (Horizontal axis dependent on the base rotation)
  - $\underline{z}_2 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix}$ (Axis parallel to $\underline{z}_1$)
- **Joint and End-Effector positions**:
  - $\underline{p}_0 = \underline{p}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
  - $\underline{p}_2 = \begin{bmatrix} a_2 C_1 C_2 \\ a_2 S_1 C_2 \\ a_2 S_2 \end{bmatrix}$
  - $\underline{p}_3 = \begin{bmatrix} C_1(a_2 C_2 + a_3 C_{23}) \\ S_1(a_2 C_2 + a_3 C_{23}) \\ a_2 S_2 + a_3 S_{23} \end{bmatrix}$

2.18.2 Jacobian Matrix

By calculating the cross products $\underline{J}_{Pi} = \underline{z}_{i-1} \times (\underline{p}_e - \underline{p}_{i-1})$ for the linear part and using the joint axes for the angular part, we obtain the 6×3 Geometric Jacobian:

$$J = \begin{bmatrix} -S_1(a_2C_2 + a_3C_{23}) & -C_1(a_2S_2 + a_3S_{23}) & -a_3C_1S_{23} \\ C_1(a_2C_2 + a_3C_{23}) & -S_1(a_2S_2 + a_3S_{23}) & -a_3S_1S_{23} \\ 0 & a_2C_2 + a_3C_{23} & a_3C_{23} \\ 0 & S_1 & S_1 \\ 0 & -C_1 & -C_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.103}$$

2.18.3 Analysis

Only the first 3 rows are linearly independent. This makes sense because we have 3 D.O.F., and the manipulator is being analyzed in terms of its primary positioning capabilities in 3D space.

2.19 Stanford Manipulator

The Stanford manipulator is a 6-DOF robot with a spherical wrist. It is characterized by a specific sequence of joints (RRPRRR).

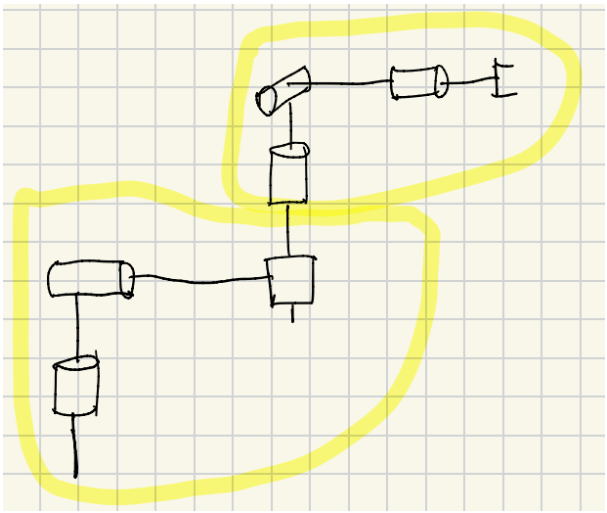


Figure 2.5: Stanford Manipulator.

The Geometric Jacobian J for this 6-degree-of-freedom manipulator is a 6×6 matrix, where each column is calculated based on the type of joint (revolute or prismatic):

$$J = \begin{bmatrix} \underline{z}_0 \times (\underline{p}_6 - \underline{p}_0) & \underline{z}_1 \times (\underline{p}_6 - \underline{p}_1) & \underline{z}_2 & \underline{z}_3 \times (\underline{p}_6 - \underline{p}_3) & \underline{z}_4 \times (\underline{p}_6 - \underline{p}_4) & \underline{z}_5 \times (\underline{p}_6 - \underline{p}_5) \\ \underline{z}_0 & \underline{z}_1 & 0 & \underline{z}_3 & \underline{z}_4 & \underline{z}_5 \end{bmatrix} \tag{2.104}$$

2.19.1 Observations on the Jacobian Structure

- **Joint 3 (Prismatic)**: The third column corresponds to a prismatic joint. Therefore, its angular velocity contribution is null ($J_{O3} = 0$) and its linear velocity contribution is simply the direction of the joint axis ($J_{P3} = \underline{z}_2$).
- **Spherical Wrist**: The last three columns ($\underline{z}_3, \underline{z}_4, \underline{z}_5$) represent the spherical wrist. The axes of these three revolute joints intersect at a single point (the wrist center).

2.20 Kinematic Singularities

The mapping between joint velocities and end-effector velocities is given by $\underline{v}_e = J(\underline{q})\dot{\underline{q}}$. If the rank of the Jacobian matrix $\rho(J)$ decreases, we encounter configurations known as **kinematic singularities**.

The main consequences of being in a singular configuration are:

- **Reduced Mobility**: It is not possible to impose an arbitrary motion to the end-effector in certain directions of the operational space.
- **Infinite Solutions**: There may exist infinite solutions for the inverse kinematics problem.
- **Velocity Discontinuity**: In the proximity of a singularity, even small velocities in the operational space can cause extremely large (theoretically infinite) velocities in the joint space.

2.21 Example: 2 Links Planar Arm

Consider a planar arm with two revolute joints to analyze its singular configurations.

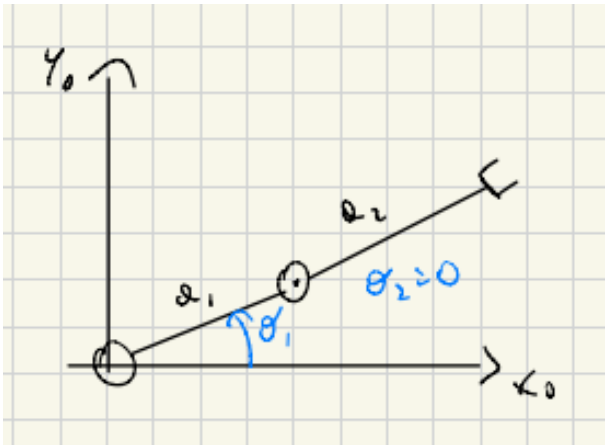


Figure 2.6: 2 Links Planar Arm.

2.21.1 Jacobian Matrix Construction

The geometric Jacobian is defined as:

$$J = \begin{bmatrix} \underline{z}_0 \times (\underline{p}_2 - \underline{p}_0) & \underline{z}_1 \times (\underline{p}_2 - \underline{p}_1) \\ \underline{z}_0 & \underline{z}_1 \end{bmatrix} \tag{2.105}$$

Given the geometry:

- $\underline{z}_0 = \underline{z}_1 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$
- $\underline{p}_0 = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
- $\underline{p}_1 = \begin{bmatrix}a_1C_1 & a_1S_1 & 0\end{bmatrix}^T$
- $\underline{p}_2 = \begin{bmatrix}a_1C_1 + a_2C_{12} & a_1S_1 + a_2S_{12} & 0\end{bmatrix}^T$

Computing the cross products for the linear part:

$$\underline{z}_0 \times (\underline{p}_2 - \underline{p}_0) = \begin{vmatrix} i & j & k \\ 0 & 0 & 1 \\ a_1C_1 + a_2C_{12} & a_1S_1 + a_2S_{12} & 0 \end{vmatrix} = \begin{bmatrix} -a_1S_1 - a_2S_{12} \\ a_1C_1 + a_2C_{12} \\ 0 \end{bmatrix} \tag{2.106}$$

$$\underline{z}_1 \times (\underline{p}_2 - \underline{p}_1) = \begin{vmatrix} i & j & k \\ 0 & 0 & 1 \\ a_2C_{12} & a_2S_{12} & 0 \end{vmatrix} = \begin{bmatrix} -a_2S_{12} \\ a_2C_{12} \\ 0 \end{bmatrix} \tag{2.107}$$

The full 6×2 Jacobian is:

$$J = \begin{bmatrix} -a_1S_1 - a_2S_{12} & -a_2S_{12} \\ a_1C_1 + a_2C_{12} & a_2C_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.108}$$

2.21.2 Determinant and Singularity Analysis

To find the singularities, we consider only the motion in the xy plane $(\dot{p}_x, \dot{p}_y)$, reducing J to a 2×2 matrix:

$$J = \begin{bmatrix} -a_1S_1 - a_2S_{12} & -a_2S_{12} \\ a_1C_1 + a_2C_{12} & a_2C_{12} \end{bmatrix} \tag{2.109}$$

The determinant is calculated as follows:

$$\det(J) = a_2C_{12}(-a_1S_1 - a_2S_{12}) - (-a_2S_{12})(a_1C_1 + a_2C_{12}) \tag{2.110}$$

$$\det(J) = -a_1a_2S_1C_{12} - a_2^2S_{12}C_{12} + a_1a_2C_1S_{12} + a_2^2S_{12}C_{12} \tag{2.111}$$

Simplifying the terms $a_2^2S_{12}C_{12}$:

$$\det(J) = a_1a_2(S_{12}C_1 - C_{12}S_1) \tag{2.112}$$

Using the trigonometric identity $\sin(\alpha - \beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta$:

$$\det(J) = a_1a_2\sin(\theta_{12} - \theta_1) = a_1a_2\sin(\theta_2) \tag{2.113}$$

2.21.3 Conclusion on Singularities

The determinant $\det(J) = 0$ when $\sin(\theta_2) = 0$, which implies:

$$\theta_2 = 0 \quad \text{or} \quad \theta_2 = \pi \tag{2.114}$$

In these configurations, the matrix becomes:

$$J = \begin{bmatrix} -(a_1 + a_2)S_1 & -a_2 S_1 \\ (a_1 + a_2)C_1 & a_2 C_1 \end{bmatrix} \tag{2.115}$$

As we can see, the two column vectors are parallel. Consequently:

$$\rho(J) = 1 < 2 \tag{2.116}$$

The manipulator loses one degree of freedom and cannot move along the direction of the arm itself (radial direction).

2.22 Spherical Arm + Spherical Wrist

In this case, computing the determinant of the full 6×6 Jacobian and studying the singularities directly is algebraically difficult. To simplify the problem, we split it into two separate sub-problems:

1. Computation of **Arm Singularities** (related to the first 3 links).

2. Computation of **Wrist Singularities** (related to the last 3 links).

The Jacobian can be partitioned into 3×3 blocks:

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{2.117}$$

By choosing the origin of the end-effector frame (p_e) at the intersection of the wrist axes (p_w), the vector $p_e - p_i$ becomes parallel to the joint axes z_i for $i = 3, 4, 5$. Consequently, the cross products in J_{12} vanish:

$$J_{12} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.118}$$

This results in a **block lower triangular matrix**:

$$\det(J) = \det(J_{11}) \cdot \det(J_{22}) \tag{2.119}$$

The global singularity condition $\det(J) = 0$ is satisfied if either $\det(J_{11}) = 0$ (Arm Singularities) or $\det(J_{22}) = 0$ (Wrist Singularities).

2.22.1 Wrist Singularities

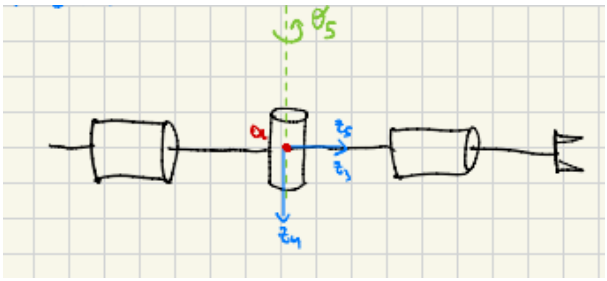


Figure 2.7: Wrist singularity.

The wrist Jacobian block is defined by the rotation axes of the spherical wrist:

$$J_{22} = \begin{bmatrix} \underline{z}_3 & \underline{z}_4 & \underline{z}_5 \end{bmatrix} \tag{2.120}$$

The determinant $\det(J_{22})$ is zero when $\underline{z}_3$ and $\underline{z}_5$ are aligned. This occurs when:

$$\theta_5 = 0 \quad \text{or} \quad \theta_5 = \pi \tag{2.121}$$

Physical interpretation : When $\theta_5 = 0$, a rotation about θ_4 and θ_6 in opposite directions does not produce any rotation of the end-effector. The manipulator loses mobility since it cannot rotate about the axis orthogonal to $\underline{z}_3$ and $\underline{z}_4$. This singularity can be encountered anywhere in the workspace.

2.22.2 Arm Singularities

For an anthropomorphic arm, the determinant of the linear part J_P is:

$$\det(J_{11}) = -a_2 a_3 S_3 (a_2 C_2 + a_3 C_{23}) \tag{2.122}$$

We identify two main cases for $\det(J_{11}) = 0$:

1. **Elbow Singularity**: $S_3 = 0 \implies \theta_3 = 0, \pi$. The arm is either fully extended or folded on itself. Like a 2D manipulator, it loses the radial degree of freedom.

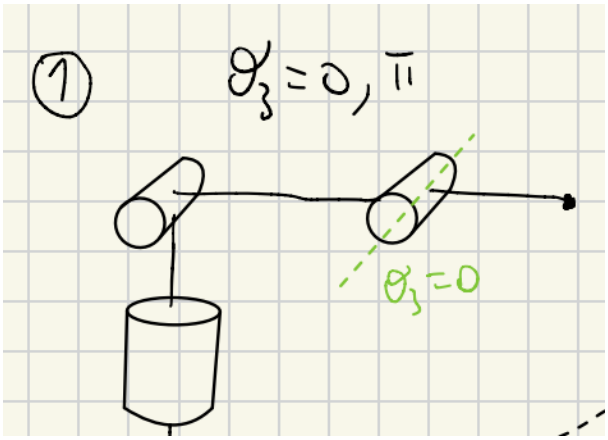


Figure 2.8: Elbow singularity.

2. **Shoulder Singularity**: $a_2C_2+a_3C_{23} = 0$. This corresponds to the condition where the wrist center lies on the z_0 axis ($p_x = p_y = 0$).

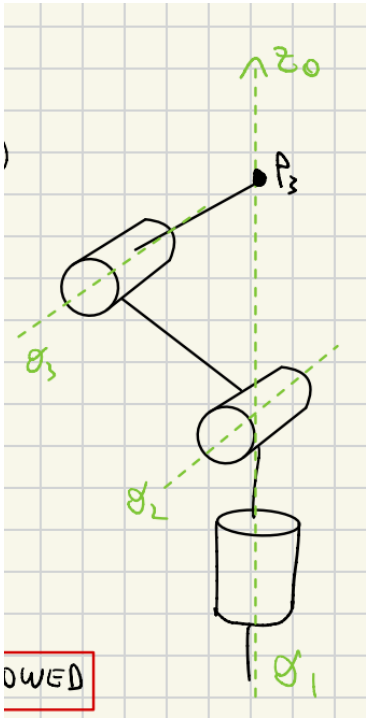


Figure 2.9: Shoulder singularity.

Note : When the wrist center is on the z_0 axis, the axis becomes a continuum of singularities. In this configuration, a motion along the z_0 axis is not allowed, or rather, the joint velocity $\dot{\theta}_1$ becomes indeterminate as it doesn't affect the end-effector position.

2.23 Redundancy in Robotics

2.23.1 Definitions

n**:** Degrees of Freedom (D.O.F.)

m**:** Operational space variables

r**:** Operational space variables necessary for a task

2.23.2 Kinematic Redundancy

A manipulator is **kinematically redundant** if the number of DOF is greater than the number of variables necessary to describe a task:

$$r < n$$

It is **intrinsically redundant** if:

$$n > m$$

Example: 3-link Planar Manipulator ($n = m = 3$)

- If only the end-effector position is considered: $r = 2 \implies$ **Redundant**.
- If the orientation is also considered: $r = 3 \implies$ **Not Redundant**.
- With 4 links: $n = 4 > m = 3 \implies$ **Intrinsically Redundant**.

2.23.3 Advantages

Redundancy provides the manipulator with **dexterity** and **versatility**. Specifically, additional DOFs allow the manipulator to **avoid obstacles** within the workspace.

Chapter 3

Differential Kinematics and Statics

3.0.1 Differential Kinematics

Given $\dot{q} \in \mathbb{R}^n$ and $v_e \in \mathbb{R}^r$, the mapping is defined by the Jacobian:

$$v_e = J(q)\dot{q}$$

In redundant systems, certain joint velocities result in zero end-effector velocity (*internal motions*).

- **If Jacobian has full rank:**

$$\dim(\mathcal{R}(J)) = r, \quad \dim(\mathcal{N}(J)) = n - r$$

- **Always true (Rank-Nullity Theorem):**

$$\dim(\mathcal{R}(J)) + \dim(\mathcal{N}(J)) = n$$

3.0.2 General Solution

If $\dot{q}^*$ is a solution to $v_e = J\dot{q}$, we can choose a matrix $P \in \mathbb{R}^{n\times n}$ such that $\mathcal{R}(P) \equiv \mathcal{N}(J)$. The general solution is:

$$\dot{q} = \dot{q}^* + P\dot{q}_0$$

where $\dot{q}_0$ is an **arbitrary** vector. The matrix P ensures that $\dot{q}_0$ produces only internal motions:

$$J\dot{q} = J\dot{q}^* + \underbrace{JP\dot{q}_0}_{=0} = J\dot{q}^* = v_e$$

Since $P\dot{q}_0 \in \mathcal{N}(J)$, these motions do not affect the end-effector velocity.

3.1 Redundant Case

In the redundant case, the Jacobian matrix J has dimensions $r \times n$ with $r < n$.

$$v_e = J(q)\dot{q}$$

In this scenario, it is not possible to invert $J(q)$ directly. To find $\dot{q}$, a cost function must be minimized:

$$g(\dot{q}) = \frac{1}{2}\dot{q}^T W \dot{q}$$

where W is an $n \times n$ positive definite matrix ($W > 0 \Rightarrow g(\dot{q}) > 0$).

3.1.1 Lagrangian Method

The optimization problem is formulated as:

$$\begin{cases} \min g(\dot{q}) = \frac{1}{2}\dot{q}^T W \dot{q} \\ \text{s.t. } v_e = J(q)\dot{q} \end{cases}$$

Using the Lagrangian:

$$g(\dot{q}, \lambda) = \frac{1}{2}\dot{q}^T W \dot{q} + \lambda^T (v_e - J\dot{q})$$

Setting the partial derivatives to zero:

1. $\left(\frac{\partial g}{\partial \dot{q}}\right)^T = (\dot{q}^T W - \lambda^T J)^T = 0 \Rightarrow W^T \dot{q} - J^T \lambda = 0 \Rightarrow \dot{q} = W^{-1} J^T \lambda$
2. $\left(\frac{\partial g}{\partial \lambda}\right)^T = (v_e - J\dot{q})^T = 0 \Rightarrow v_e^T - \dot{q}^T J^T = 0 \Rightarrow v_e = J\dot{q}$

3.1.2 Solution for Joint Velocities

Substituting the expression for $\dot{q}$ into the constraint:

$$v_e = JW^{-1}J^T \lambda$$

Assuming the Jacobian has **full rank** and we are not in a singularity, the dimensions are $[r \times n][n \times n][n \times r] = [r \times r]$. Solving for λ:

$$\lambda = (JW^{-1}J^T)^{-1} v_e$$

The optimal solution is:

$$\dot{q} = W^{-1}J^T(JW^{-1}J^T)^{-1} v_e$$

This is the only solution among the infinite possible ones that minimizes $g(\dot{q})$.

Case $W = I$ If we weight the joints equally ($W = I$), the solution simplifies to the **right pseudo-inverse** J^+:

$$\dot{q} = J^T(JJ^T)^{-1} v_e = J^+ v_e \quad \text{where } JJ^+ = I$$

3.1.3 Secondary Constraints and Internal Motions

If redundancy exists, we can use an arbitrary vector $\dot{q}_0$ to satisfy secondary constraints. The new optimization problem becomes:

$$\begin{cases} g(\dot{q}) = \frac{1}{2}(\dot{q} - \dot{q}_0)^T(\dot{q} - \dot{q}_0) \\ \text{s.t. } v_e = J\dot{q} \end{cases}$$

The Lagrangian leads to the solution:

$$\dot{q} = \dot{q}_0 + J^T(JJ^T)^{-1}(v_e - J\dot{q}_0)$$

Rewriting using J^+:

$$\dot{q} = \dot{q}_0 + J^+(v_e - J\dot{q}_0) = \dot{q}_0 + J^+ v_e - J^+ J\dot{q}_0$$

$$\dot{q} = J^+ v_e + (I_n - J^+ J)\dot{q}_0$$

The term $P = (I_n - J^+ J)$ is a projection matrix. Verified by multiplying by J:

$$J\dot{q} = \underbrace{JJ^+}_{I} v_e + \underbrace{(J - JJ^+ J)}_{0} \dot{q}_0 = v_e$$

3.1.4 How to choose $\dot{q}_0$

The vector $\dot{q}_0$ is typically chosen as the gradient of a secondary cost function $w(q)$:

$$\dot{q}_0 = k_0 \left(\frac{\partial w(q)}{\partial q} \right)^T$$

Examples of $w(q)$:

- **Manipulability:** $w(q) = \sqrt{\det(J(q)J^T(q))}$. Maximizing this ensures the Jacobian is far from singularity.
- **Obstacle Avoidance:** $w(q) = \min_{p,o} \|p(q) - o\|$ (distance from an obstacle).

3.2 Redundant Case

In the redundant case, the Jacobian matrix J is $r \times n$ with $r < n$.

$$v_e = J(q)\dot{q}$$

Since $J(q)$ cannot be inverted directly, we find $\dot{q}$ by minimizing a cost function:

$$g(\dot{q}) = \frac{1}{2}\dot{q}^T W \dot{q} \quad \text{with } W_{n\times n} > 0 \Rightarrow g(\dot{q}) > 0$$

3.2.1 Lagrangian Method

The optimization problem is:

$$\begin{cases} \min g(\dot{q}) = \frac{1}{2}\dot{q}^T W \dot{q} \\ \text{sbj. } v_e = J(q)\dot{q} \end{cases}$$

Using the Lagrangian:

$$g(\dot{q}, \lambda) = \frac{1}{2}\dot{q}^T W \dot{q} + \lambda^T (v_e - J\dot{q})$$

Setting derivatives to zero:

1. $\left(\frac{\partial g}{\partial \dot{q}}\right)^T = (\dot{q}^T W - \lambda^T J)^T = 0 \Rightarrow W^T \dot{q} - J^T \lambda = 0 \Rightarrow \dot{q} = W^{-1} J^T \lambda$
2. $\left(\frac{\partial g}{\partial \lambda}\right)^T = (v_e - J\dot{q})^T = 0 \Rightarrow v_e = J\dot{q}$

3.2.2 Solution for Joint Velocities

Substituting the previous results:

$$v_e = JW^{-1}J^T\lambda \quad \text{(Assuming full rank, not in a singularity)}$$

$$\lambda = (JW^{-1}J^T)^{-1} v_e \Rightarrow \dot{q} = W^{-1}J^T(JW^{-1}J^T)^{-1}v_e$$

If $W = I$ (equally weighted joints), we obtain the **right pseudo-inverse** J^+:

$$\dot{q} = J^T(JJ^T)^{-1}v_e = J^+ v_e \quad \text{with } JJ^+ = I$$

3.2.3 Secondary Constraints and Internal Motions

If redundancy exists, $\dot{q}_0$ can satisfy secondary constraints:

$$\dot{q} = J^+ v_e + (I_n - J^+ J)\dot{q}_0$$

Where $P = (I_n - J^+ J)$ is the projection matrix into the null space of J.

How to choose $\dot{q}_0$ $\dot{q}_0 = k_0 \left(\frac{\partial w(q)}{\partial q}\right)^T$ where $w(q)$ is a secondary cost function:

- **Manipulability**: $w(q) = \sqrt{\det(J(q)J^T(q))}$ (far from singularity).
- **Obstacle Avoidance**: $w(q) = \min_{p,o} \|p(q) - o\|$.

3.2.4 Singularity Handling (Non-Full Rank J)

When J is not full rank, two possibilities arise:

- $v_e \in \mathcal{R}(J)$: In a singularity, but v_e is **admissible**.
- $v_e \notin \mathcal{R}(J)$: Nothing we can do.

In the neighborhood of a singularity, a small v_e leads to a very large $\dot{q}$ due to the small determinant.

Damped Least-Squares (DLS) Inverse The solution is the DLS inverse J^*:

$$J^* = J^T(JJ^T + k^2 I)^{-1}$$

This comes from minimizing the error $(v_e - J\dot{q})$ without imposing it to be zero, preventing $\dot{q}$ from becoming too big:

$$g''(\dot{q}) = \frac{1}{2}(v_e - J\dot{q})^T(v_e - J\dot{q}) + \frac{1}{2}k^2\dot{q}^T\dot{q}$$

3.3 Singularity Handling

3.3.1 Problem Definition

When the Jacobian J is not full rank, we face two possibilities:

- $v_e \in \mathcal{R}(J)$: We are in a singularity, but v_e is **admissible**.
- $v_e \notin \mathcal{R}(J)$: Nothing we can do.

Even in the neighborhood of a singularity, a small v_e results in a large $\dot{q}$ due to the small determinant.

3.3.2 Damped Least-Squares (DLS) Inverse

The solution to this problem is the DLS inverse J^*:

$$J^* = J^T(JJ^T + k^2 I)^{-1}$$

This is derived from a minimization problem where we minimize the error $(v_e - J\dot{q})$ without forcing it to zero, preventing joint velocities from becoming too large:

$$g''(\dot{q}) = \frac{1}{2}(v_e - J\dot{q})^T(v_e - J\dot{q}) + \frac{1}{2}k^2\dot{q}^T\dot{q}$$

3.4 Analytical Jacobian

Linear and Angular Velocity

The linear velocity is the derivative of the position with respect to the joints:

$$\dot{p}_e = \frac{\partial p}{\partial q}\dot{q} = J_p(q)\dot{q}$$

For the orientation:

$$q \xrightarrow{\text{Direct}} R_e \xrightarrow{\text{Inv.}} \phi_e$$

3.4.1 ZYZ Euler Angles Case

$\dot{\phi} = \begin{bmatrix} \dot{\varphi} \\ \dot{\vartheta} \\ \dot{\psi} \end{bmatrix}$

Let's consider ZYZ:

1. $\omega_z = \dot{\varphi} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

2. Now the frame is rotated (R_z). Now I want to rotate about the y' axis:

$$\omega^0 = \dot{\vartheta} R_z y' = \dot{\vartheta} \begin{bmatrix} c\varphi & -s\varphi & 0 \\ s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \Rightarrow \omega^0 = \dot{\vartheta} \begin{bmatrix} -s\varphi \\ c\varphi \\ 0 \end{bmatrix}$$

   $R_z y'$ is the y' axis respect to the 0-frame.

3. Now the rotation is $R_z R_y$:

$$R_{zy} = \begin{bmatrix} c\varphi c\vartheta & -s\varphi & c\varphi s\vartheta \\ s\varphi c\vartheta & c\varphi & s\varphi s\vartheta \\ -s\vartheta & 0 & c\vartheta \end{bmatrix}$$

$$\omega_0 = \dot{\psi} R_{zy} z'' = \dot{\psi} \begin{bmatrix} c\varphi s\vartheta \\ s\varphi s\vartheta \\ c\vartheta \end{bmatrix}$$

The total angular velocity is:

$$\omega_e = \begin{bmatrix} 0 & -s\varphi & c\varphi s\vartheta \\ 0 & c\varphi & s\varphi s\vartheta \\ 1 & 0 & c\vartheta \end{bmatrix} \dot{\phi} = T(\phi_e)\dot{\phi}$$

$\det T = -s^2\varphi s\vartheta - c^2\varphi s\vartheta = -(s^2\varphi + c^2\varphi)s\vartheta = -\sin\vartheta$ $\det T = 0$ if $\vartheta = 0, \pi \Rightarrow$ **REPRESENTATION SINGULARITY**

3.4.2 Relationship between Jacobians

$$v_e = \begin{bmatrix} I & 0 \\ 0 & T(\phi_e) \end{bmatrix} \dot{x} = T_A(\phi_e)\dot{x} \quad \text{where } \dot{x} = \begin{bmatrix} \dot{p}_e \\ \dot{\phi}_e \end{bmatrix}$$

$J = T_A(\phi_e)J_A$ $J_A = T_A^{-1}(\phi_e)J$

J_A is worse because it suffers of both the singularities of $T_A^{-1}(\phi_e)$ and J. The Geometrical Jacobian is more "physical" (w is the angular velocity). In the Analytical Jacobian you use the derivative of parameters we use to describe the pose (Euler, Quaternion...). $\dot{\phi}$ is the derivative of ϕ used to describe the orientation of the end-effector (not physical).

SOMETIMES $J = J_A$

3.5 Inverse Kinematics Algorithms

The basic numerical approach for inverse kinematics is:

$$q(t_{k+1}) = q(t_k) + J^{-1}(q(t_k))v_e(t_k)\Delta t$$

It is a **numerical integration** $\Longrightarrow$ **DRIFT**. This means that the end-effector pose corresponding to the computed joint variables differs from the desired one. $\Longrightarrow$ We need a **closed-loop control** that accounts for the operational space error.

3.5.1 Closed-Loop Inverse Kinematics (CLIK)

Defining the operational space error:

$$e = x_d - x_e$$

The error dynamics are:

$$\dot{e} = \dot{x}_d - \dot{x}_e = \dot{x}_d - J_A(q)\dot{q}$$

(Note: We used operational space variables $\Longrightarrow$ **Analytical Jacobian**).

We have to find a relationship between $\dot{q}$ and e in order to have a differential equation describing the error over time. If $e \to 0 \implies 0 = \dot{x}_d - J_A(q)\dot{q} \implies \dot{q} = J_A^{-1}(q)\dot{x}_d$ (Open Loop).

3.5.2 Jacobian (Pseudo) Inverse

Assuming J_A is square and non-singular, we choose:

$$\dot{q} = J_A^{-1}(q)(\dot{x}_d + Ke)$$

where K is a feedback term proportional to the error. Replacing this in $\dot{e} = \dot{x}_d - J_A(q)\dot{q}$:

$$\dot{e} = \dot{x}_d - J_A J_A^{-1}(\dot{x}_d + Ke) = -Ke$$

$$\dot{e} + Ke = 0$$

If $K > 0$ is a diagonal matrix $\implies$ the system is **asymptotically stable**:

$$e(t) = e(0)e^{-Kt}$$

For a Redundant Manipulator: The solution incorporates the null-space projection:

$$\dot{q} = J_A^{\dagger}(\dot{x}_d + Ke) + (I - J_A^{\dagger}J_A)\dot{q}_0$$

3.5.3 Control Scheme

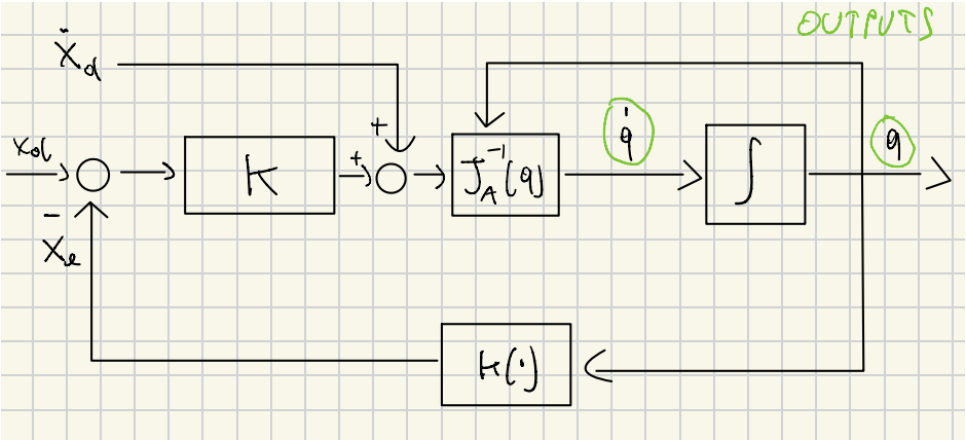


Figure 3.1: Closed-loop inverse kinematics control scheme.

- The larger the eigenvalues of K, the faster the error convergence.
- But since we use discrete-time systems, we cannot make it too large.

3.6 Jacobian Transpose

Instead of inverting the Jacobian, we can use the **Jacobian Transpose**. This approach is computationally lighter and avoids problems related to matrix inversion near singularities.

Consider the Lyapunov function candidate:

$$V(e) = \frac{1}{2}e^T K^{-1} e > 0$$

where K is a positive definite symmetric matrix. To guarantee asymptotic stability, we need $\dot{V}(e) < 0$.

Taking the derivative:

$$\dot{V} = e^T K^{-1}\dot{e} = e^T K^{-1}(\dot{x}_d - J_A\dot{q})$$

Assuming a constant desired pose ($\dot{x}_d = 0$):

$$\dot{V} = -e^T K^{-1} J_A \dot{q}$$

If we choose:

$$\dot{q} = J_A^T K e$$

Then:

$$\dot{V} = -e^T K^{-1} J_A J_A^T K e = -(J_A^T e)^T (J_A^T e) \leq 0$$

The error will converge to zero as long as the system is not in a singularity where $J_A^T e = 0$ with $e \neq 0$.

3.6.1 Comparison: Inverse vs. Transpose

- **Inverse**: Provides a linear error dynamics ($\dot{e} + Ke = 0$), meaning the convergence rate is predictable and constant.
- **Transpose**: The convergence depends on the state of the manipulator (the Jacobian) and is generally slower, but it is much more robust.

3.6.2 Transpose Control Scheme

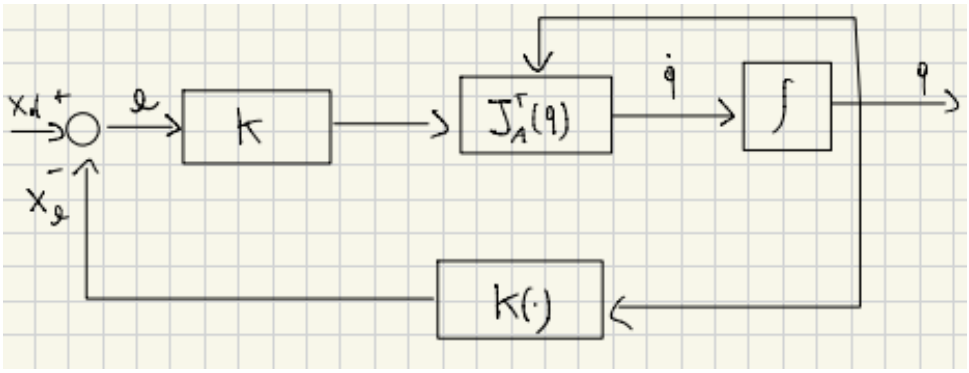


Figure 3.2: Inverse kinematics control scheme using the Jacobian Transpose.

3.7 Anthropomorphic Arm: Shoulder Singularity

Let's consider an anthropomorphic arm in the shoulder singularity.

$$p_e = \begin{bmatrix} 0 \\ 0 \\ p_z \end{bmatrix} \tag{3.1}$$

$$dim(\mathcal{R}(J_p)) = 2 \tag{3.2}$$

I can only move in 2 dimensions.

$$\mathcal{R}(J_p) \equiv \mathcal{N}^{\perp}(J_p^T) \tag{3.3}$$

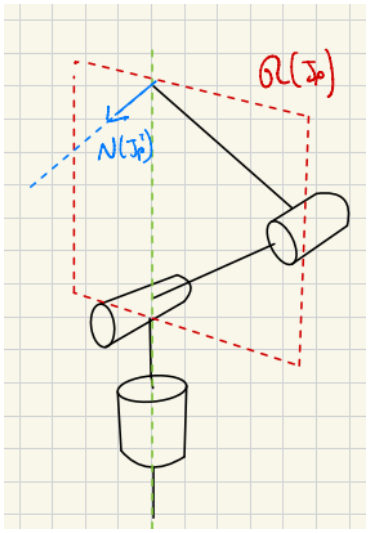


Figure 3.3: Anthropomorphic Shoulder Singularity.

$$J_p(q) = \begin{bmatrix} 0 & -c_1(a_2 s_2 + a_3 s_{23}) & -a_3 c_1 s_{23} \\ 0 & -s_1(a_2 s_2 + a_3 s_{23}) & -a_3 s_1 s_{23} \\ 0 & 0 & a_3 c_{23} \end{bmatrix} \tag{3.4}$$

$$v_e = J_p \begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix} \tag{3.5}$$

3.8 Orientation Error

3.8.1 Position

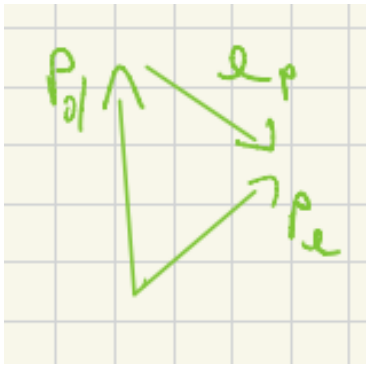


Figure 3.4:

$$e_p = p_d - p_e(q) \tag{3.6}$$

$$\dot{e}_p = \dot{p}_d - \dot{p}_e \tag{3.7}$$

3.8.2 Euler Angles

$$e_o = \phi_d - \phi_e(q) \tag{3.8}$$

$$\dot{e}_o = \dot{\phi}_d - \dot{\phi}_e(q) \tag{3.9}$$

$$\dot{q} = J_A^{-1}(q) \begin{bmatrix} \dot{p}_d + K_p e_p \\ \dot{\phi}_d + K_o e_o \end{bmatrix} \tag{3.10}$$

3.9 Angle and Axis

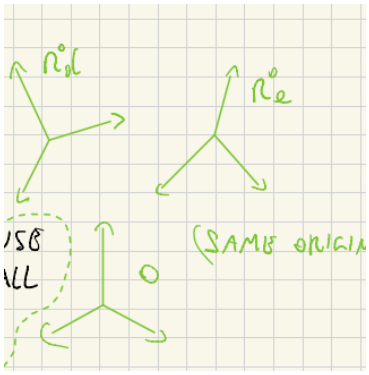


Figure 3.5:

$$R(\theta, r) = R_d R_e^T \tag{3.11}$$

Mutual rotation matrix (same origin).

$$-\frac{\pi}{2} < \theta < \frac{\pi}{2} \quad \text{IT'S OK BECAUSE THE ERROR IS SMALL} \tag{3.12}$$

$e_o = 0$ if $\theta = 0, \theta = \pi$.

$$e_o = r \sin\theta = \frac{1}{2}(n_e \times n_d + s_e \times s_d + a_e \times a_d) \tag{3.13}$$

$$\dot{e}_o = L^T \omega_d - L\omega_e \tag{3.14}$$

$$L = -\frac{1}{2}(S(n_d)S(n_e) + S(s_d)S(s_e) + S(a_d)S(a_e)) \tag{3.15}$$

$$\Rightarrow \dot{e} = \begin{bmatrix} \dot{e}_p \\ \dot{e}_o \end{bmatrix} = \begin{bmatrix} \dot{p}_d - J_p(q)\dot{q} \\ L^T\omega_d - LJ_o(q)\dot{q} \end{bmatrix} = \begin{bmatrix} \dot{p}_d \\ L^T\omega_d \end{bmatrix} - \begin{bmatrix} I & 0 \\ 0 & L \end{bmatrix} J\dot{q} \tag{3.16}$$

WE USED THE GEOMETRIC JACOBIAN

$$\dot{q} = J^{-1}(q) \begin{bmatrix} \dot{p}_d + K_p e_p \\ L^{-1}(L^T\omega_d + K_o e_o) \end{bmatrix} \tag{3.17}$$

3.10 Unit Quaternions

$$\Delta Q = Q_d \cdot Q_e^{-1} = \{\Delta\eta, \Delta\epsilon\} \tag{3.18}$$

I ONLY USE $\Delta\epsilon$

$$e_o = \Delta\epsilon = \eta_e(q)\epsilon_d - \eta_d\epsilon_e(q) - S(\epsilon_d)\epsilon_e(q) \tag{3.19}$$

$$\dot{q} = J^{-1}(q)\begin{bmatrix}\dot{p}_d + K_p e_p \\ \omega_d + K_o e_o\end{bmatrix} \tag{3.20}$$

3.11 Second Order Algo.

We know $\dot{x}_e = J_A(q)\dot{q}$ so:

$$\ddot{x}_e = J_A(q)\ddot{q} + \dot{J}_A(q,\dot{q})\dot{q} \tag{3.21}$$

$$\ddot{q} = J_A^{-1}(q)\left(\ddot{x}_d + K_D\dot{e} + K_P e - \dot{J}_A(q,\dot{q})\dot{q}\right) \tag{3.22}$$

$$\Rightarrow \ddot{e} + K_D\dot{e} + K_P e = 0 \quad K_D, K_P > 0 \Rightarrow e \to 0 \tag{3.23}$$

3.12 Comparison

3 links planar arm.

$$x_e = h(q) \tag{3.24}$$

$$\begin{bmatrix}p_x \\ p_y \\ \phi_e\end{bmatrix} = \begin{bmatrix}a_1c_1 + a_2c_{12} + a_3c_{123} \\ a_1s_1 + a_2s_{12} + a_3s_{123} \\ \theta_1 + \theta_2 + \theta_3\end{bmatrix} \tag{3.25}$$

$$J_A = \begin{bmatrix}-a_1s_1 - a_2s_{12} - a_3s_{123} & -a_2s_{12} - a_3s_{123} & -a_3s_{123} \\ a_1c_1 + a_2c_{12} + a_3c_{123} & a_2c_{12} + a_3c_{123} & a_3c_{123} \\ 1 & 1 & 1\end{bmatrix} \tag{3.26}$$

$$q_i = \begin{bmatrix}\pi & -\frac{\pi}{2} & -\frac{\pi}{2}\end{bmatrix}^T \quad \text{Initial configuration} \tag{3.27}$$

$$p_d(t) = \begin{bmatrix}0.25(1 - \cos\pi t) \\ 0.25(2 + \sin\pi t)\end{bmatrix} \tag{3.28}$$

$$\phi_d = \sin\frac{\pi}{24}t \quad 0 \le t \le 4 \tag{3.29}$$

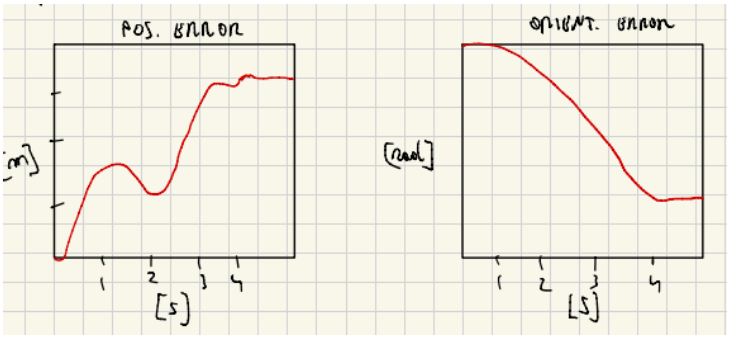
Desired trajectory

3.13 On Matlab

$$q(t_{n+1}) = q(t_n) + \dot{q}(t_n)\Delta t \quad \Delta t = 1ms \tag{3.30}$$

Let's see with the simple inverse Jacobian

$$\dot{q} = J_A^{-1}(q)\dot{x}_d \tag{3.31}$$



⇒ Drift

3.14 Let's use the feedback error

$$\dot{q} = J_A^{-1}(\dot{x}_d + Ke) \quad K = diag\{500, 500, 100\} \tag{3.32}$$

- If we don't care about the orientation:

$$r = 2 \quad n = 3 \tag{3.33}$$

$$\dot{q} = J_p^{\dagger}(\dot{p}_d + K_p e_p) \quad K_p = diag\{500, 500\} \tag{3.34}$$

- We can also use the transpose

$$\dot{q} = J_p^T(q) K_p e_p \quad K_p = diag\{500, 500\} \tag{3.35}$$

  ⇒ Computationally lighter

- If it's redundant:

$$\dot{q} = J_p^{\dagger}(q)(\dot{p}_d + K_p e_p) + (I - J_p^{\dagger} J_p)\dot{q}_0 \tag{3.36}$$

Manipulability measure:

$$w(\theta_1, \theta_2) = \frac{1}{2}(s_2^2 + s_3^2) \quad \text{Avoid singularity} \tag{3.37}$$

$$\dot{q}_0 = k_0 \left(\frac{\partial w}{\partial q}\right)^T \quad k_0 = 50 \tag{3.38}$$

3.15 Statics

Relationship between the forces on the end-effector and the forces on the joints (linear forces for prismatic joints and torques for revolute joints) with the manipulator at the equilibrium.

Elementary work done by torques:

$$dW_\tau = \tau^T dq \quad \tau \in \mathbb{R}^{n\times 1}, \gamma_e = \begin{bmatrix} f_e \\ \mu_e \end{bmatrix} \in \mathbb{R}^{r\times 1} \tag{3.39}$$

Elementary work done by forces:

$$dW_\gamma = f_e^T dp_e + \mu_e^T \omega_e dt = \tag{3.40}$$
$$= f_e^T J_p(q) dq + \mu_e^T J_o(q) dq = \tag{3.41}$$
$$= \gamma_e^T J(q) dq \quad \text{Geometric Jacobian} \tag{3.42}$$

Note: $dp_e = J_p(q)dq$ and $\omega_e = J_o(q)\frac{dq}{dt}$.

3.16 Virtual Works Principle

Virtual work of the joints:

$$\delta W_\tau = \tau^T \delta q \tag{3.43}$$

Virtual work of the end-effector:

$$\delta W_\gamma = \gamma_e^T \delta x = \gamma_e^T J(q) \delta q \tag{3.44}$$

Static Equilibrium $\Rightarrow$ The sum of the virtual works must be zero.

$$\Rightarrow \delta W_\tau = \delta W_\gamma \quad \forall \delta q \tag{3.45}$$

So:

$$\tau^T \delta q = \gamma_e^T J(q) \delta q \quad \forall q \tag{3.46}$$

$$\Rightarrow \tau = J^T(q)\gamma_e \tag{3.47}$$

3.17 Kineto-Statics Duality

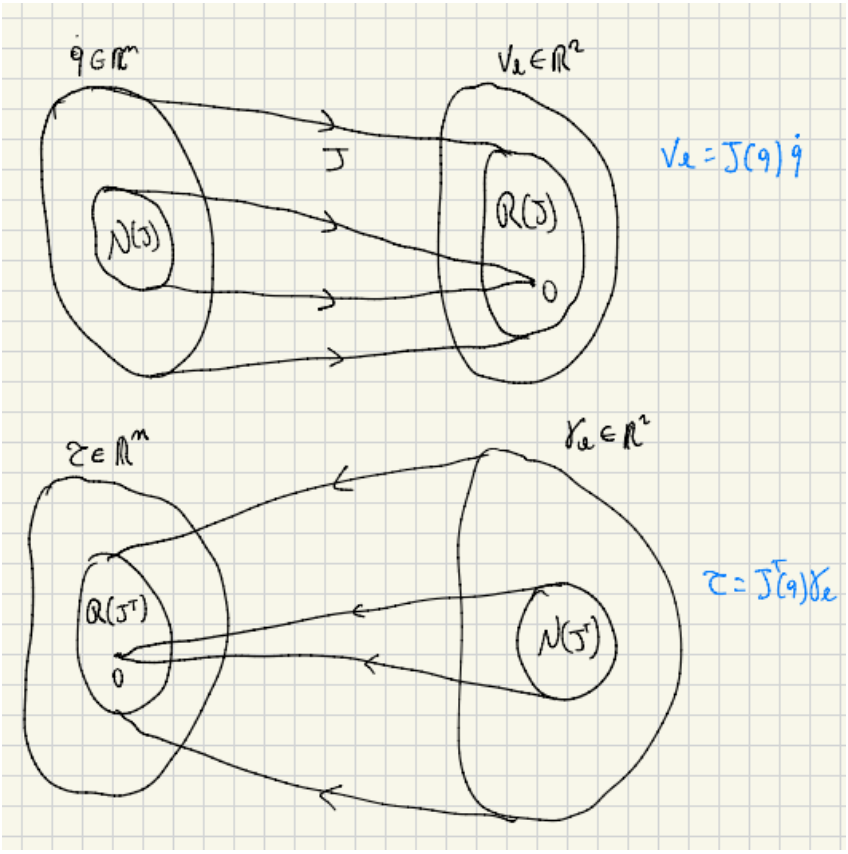
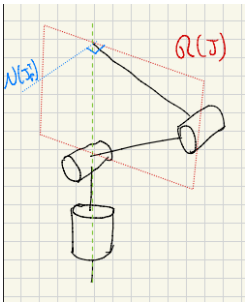


Figure 3.6: Mapping between joint space and operational space for velocities and forces.

The relationship $v_e = J(q)\dot{q}$ maps joint velocities to end-effector velocities. The relationship $\tau = J^T(q)\gamma_e$ maps end-effector forces to joint torques.

If $\mathcal{N}(J^T) \neq \emptyset$ means that you can find some forces γ_e that don't require a τ in the joints to stay in equilibrium.

3.17.1 Singularity Analysis



- $\mathcal{R}(J) = \mathcal{N}^{\perp}(J^T)$
- $\mathcal{N}(J) = \mathcal{R}^{\perp}(J^T)$

$\gamma_e \in \mathcal{N}(J^T)$ is totally absorbed by the structure.

3.18 Manipulability Ellipsoid

$\dot{q}^T\dot{q} = 1$ is the set of unit joint velocities. It describes the points on a surface of a sphere in the joint velocity space. Now we want to see which velocities in the operational space are generated.

Knowing: $\dot{q} = J^\dagger(q)v_e$

$$v_e^T(J^{\dagger T}J^\dagger)v_e = 1 \tag{3.48}$$

Where $J^\dagger = J^T(JJ^T)^{-1}$.

$$v_e^T\left((JJ^T)^{-1}JJ^T(JJ^T)^{-1}\right)v_e = 1 \tag{3.49}$$

$$v_e^T(JJ^T)^{-1}v_e = 1 \tag{3.50}$$

(I don't use $\dot{q}_0$ because it doesn't affect the end effector).

3.18.1 Properties of the Ellipsoid

If I compute the eigenvectors of JJ^T, I find the principal axis of the ellipsoid. The dimensions are given by the singular values of J: $\sigma_i = \sqrt{\lambda_i(JJ^T)}$ for $i = 1, \ldots, r$, where $\lambda_i(JJ^T)$ is an eigenvalue of JJ^T.

The volume of this ellipsoid is proportional to:

$$w(q) = \sqrt{\det(JJ^T)} \tag{3.51}$$

In the case of a non-redundant manipulator:

$$w(q) = |\det(J(q))| \tag{3.52}$$

3.19 Example: 2 Link Planar Arm

$$w(q) = |\det(J)| = a_1a_2|s_2| \tag{3.53}$$

Maximum for $\theta_2 = \pm\frac{\pi}{2}$.

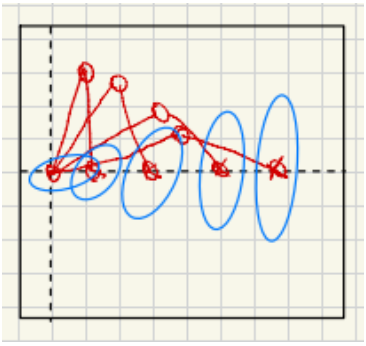


Figure 3.7: Evolution of the manipulability ellipsoid along the workspace.

Meaning: the last configuration is telling us that for a given $\dot{q}$ I will move profoundly in the vertical direction. If it's fully stretched we lose a dimension.

3.20 Force Ellipsoid

We can do the same for the forces.

$$\tau^T \tau = 1 \Rightarrow \gamma_e^T (J(q) J^T(q)) \gamma_e = 1 \tag{3.54}$$

The eigenvectors are the same but the eigenvalues are the inverse. A direction with a high velocity manipulability has a low force manipulability.

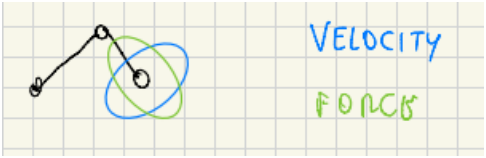


Figure 3.8: Duality between velocity (blue) and force (green) ellipsoids.

Chapter 4

Trajectory Planning

4.1 Trajectory Planning

A trajectory is defined as the union of a path and a time law:

- **Path**: Geometric description (set of points).
- **Time Law**: Specifies velocities and accelerations along the path.

The trajectory planner receives the path, time law, limits, and obstacles as input to generate the reference variables for the control system: $q_d, \dot{q}_d, \ddot{q}_d$ or $x_d, \dot{x}_d, \ddot{x}_d$.

4.1.1 Path and Time Law Description

- **Path Description**: Includes initial and final points, intermediate points of interest, and the primitive of motion (how to connect the points).
- **Time Law**: Includes total time, maximum velocities/accelerations, and specific velocity/acceleration constraints at certain points.

Trajectory planning can occur in the **joint space** or in the **operational space**.

4.2 Point-to-Point Motion

We must choose a trajectory that is easy to compute and continuous in position, velocity, and acceleration.

4.2.1 Polynomial Trajectories

For a single rigid body with model $I\dot{\omega} = \tau$ and $\omega = \dot{q}$, we want to minimize the control effort:

$$\min J = \int_0^{t_f} \tau^2(t)dt \tag{4.1}$$

The type of solution for the velocity profile is $w(t) = at^2 + bt + c$. Consequently, the position is described by a 3rd grade polynomial:

$$q(t) = a_3t^3 + a_2t^2 + a_1t + a_0 \tag{4.2}$$

To find the 4 unknown coefficients, we use 4 boundary conditions:

$$q(0) = q_i = a_0 \tag{4.3}$$
$$q(t_f) = q_f = a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0 \tag{4.4}$$
$$\dot{q}(0) = \dot{q}_i = a_1 \tag{4.5}$$
$$\dot{q}(t_f) = \dot{q}_f = 3a_3 t_f^2 + 2a_2 t_f + a_1 \tag{4.6}$$

Note: If we want to impose a specific acceleration at the boundaries, a 5th grade polynomial is required.

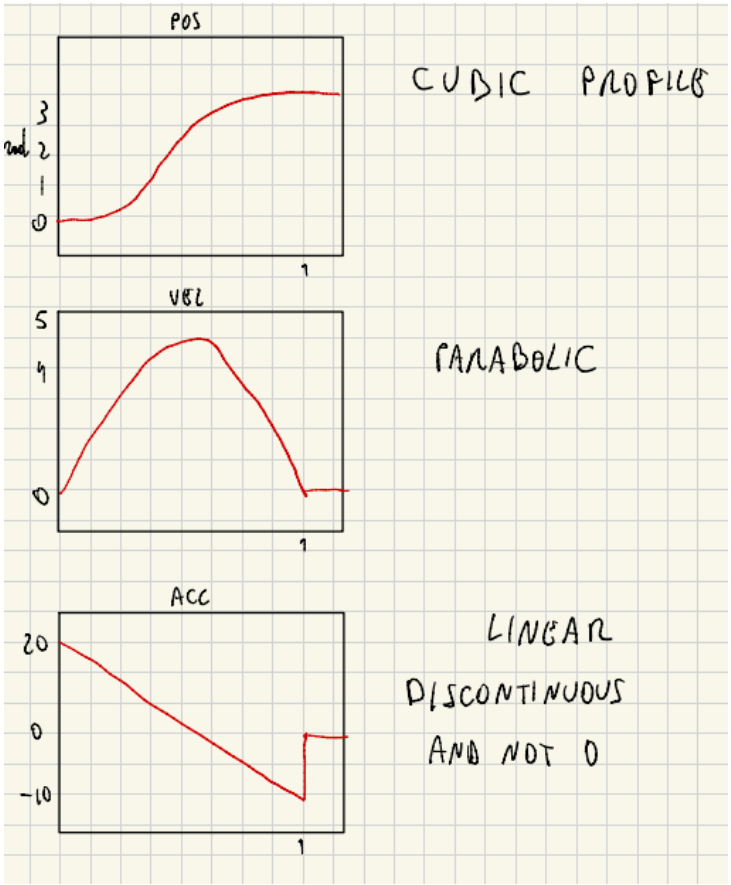


Figure 4.1: Graphics.

4.2.2 Trapezoidal Velocity Trajectories (LSPB)

Usually, we want a trapezoid velocity profile. This approach results in a trajectory composed of a constant acceleration phase, a constant velocity phase, and a constant deceleration phase.

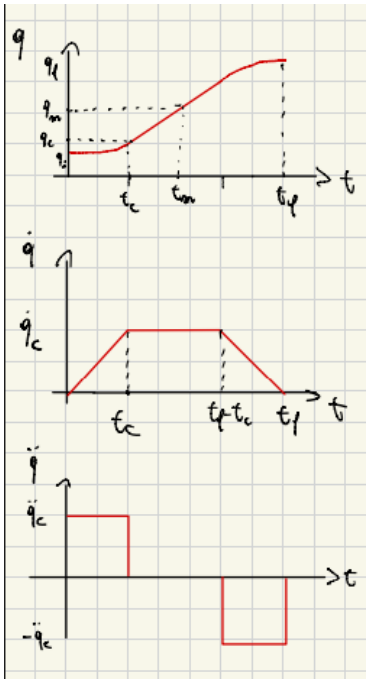


Figure 4.2: Position $q(t)$, velocity $\dot{q}(t)$, and acceleration $\ddot{q}(t)$ profiles for LSPB.

The symmetry of the trajectory implies:

$$t_m = \frac{t_f}{2}, \quad q_m = \frac{q_i + q_f}{2} \tag{4.7}$$

For the linear segment (constant velocity $\dot{q}_c$) starting at t_c, the position is derived from the line equation:

$$q = q_0 + m(t - x_0) + q_i \tag{4.8}$$

$$q_0 = \frac{1}{2}\dot{q}_c t_c, \quad m = \dot{q}_c, \quad x_0 = t_c$$

$$\Rightarrow \frac{1}{2}\dot{q}_c t_c + \dot{q}_c(t - t_c) + q_i \tag{4.9}$$

Given that $\dot{q}_c = \ddot{q}_c t_c$, the position during the linear phase can be expressed as:

$$q(t) = \frac{1}{2}\ddot{q}_c t_c^2 - \ddot{q}_c t_c^2 + \ddot{q}_c t_c t + q_i \tag{4.10}$$

The complete mathematical description of the position trajectory $q(t)$ is:

$$q(t) = \begin{cases} q_i + \frac{1}{2}\ddot{q}_c t^2 & 0 \le t \le t_c \\ q_i + \ddot{q}_c t_c \left(t - \frac{1}{2}t_c\right) & t_c < t \le t_f - t_c \\ q_f - \frac{1}{2}\ddot{q}_c (t_f - t)^2 & t_f - t_c < t \le t_f \end{cases} \tag{4.11}$$

Pros: Constant acceleration and velocity segments.
Con: Not the optimal solution: J increases by 12.5%.
If $\ddot{q}_c$ is specified, we can find the acceleration time t_c. The base constraint is that the motion lasts t_f and moves from q_i to q_f.

The displacement is equivalent to the area under the velocity profile:

$$q_i + t_c\dot{q}_c + (t_f - 2t_c)\dot{q}_c = q_f \tag{4.12}$$

Knowing that $\dot{q}_c = \ddot{q}_c t_c$, we can substitute to obtain the quadratic equation:

$$\ddot{q}_c t_c^2 - \ddot{q}_c t_f t_c + q_f - q_i = 0 \tag{4.13}$$

Solving for t_c:

$$t_c = \frac{t_f}{2} - \frac{1}{2}\sqrt{\frac{t_f^2\ddot{q}_c - 4(q_f - q_i)}{\ddot{q}_c}} \tag{4.14}$$

To ensure a real solution, the term inside the square root (the numerator) must be ≥ 0. This leads to the following constraint on the acceleration:

$$|\ddot{q}_c| \geq \frac{4|q_f - q_i|}{t_f^2} \tag{4.15}$$

If $\ddot{q}_c$ reaches the lower limit of this inequality, the velocity profile is no longer trapezoidal but triangular, with $t_c = t_m = \frac{t_f}{2}$.

4.3 Multiple Points Trajectories

If N points must be connected, a single polynomial of degree $N-1$ would be required. This approach is computationally intensive and susceptible to oscillations (Runge's phenomenon).

4.3.1 Cubic Splines

To avoid these issues, we use $N-1$ polynomials of 3rd degree (Π_k) with continuity constraints. Each segment Π_k is defined between t_k and t_{k+1}.

$$\Pi_k(t) \text{ with } t \in [t_k, t_{k+1}] \tag{4.16}$$

For each segment, we impose the following boundary conditions:

$$\Pi_k(t_k) = q_k \tag{4.17}$$
$$\Pi_k(t_{k+1}) = q_{k+1} \tag{4.18}$$
$$\dot{\Pi}_k(t_k) = \dot{q}_k \tag{4.19}$$
$$\dot{\Pi}_k(t_{k+1}) = \dot{q}_{k+1} \tag{4.20}$$

To ensure velocity continuity at the internal waypoints, we require:

$$\dot{\Pi}_k(t_{k+1}) = \dot{\Pi}_{k+1}(t_{k+1}) \tag{4.21}$$

For N waypoints, we have $N-1$ segments and $4(N-1)$ coefficients to determine.

4.3.2 Velocity Assignment

If the velocities $\dot{q}_k$ are not specified, they can be calculated using the average velocity of adjacent segments $V_k = \frac{q_k - q_{k-1}}{t_k - t_{k-1}}$:

$$\begin{cases} \dot{q}_1 = 0 \\ \dot{q}_n = \begin{cases} 0 & \text{if } \operatorname{sgn}(V_n) \neq \operatorname{sgn}(V_{n+1}) \\ \frac{1}{2}(V_n + V_{n+1}) & \text{if } \operatorname{sgn}(V_n) = \operatorname{sgn}(V_{n+1}) \end{cases} \\ \dot{q}_N = 0 \end{cases} \tag{4.22}$$

Note: This ensures velocity continuity (C^1), but there will be jumps in the acceleration profile at the waypoints.

4.3.3 Acceleration Continuity (C^2)

To achieve C^2 continuity, we add acceleration constraints at each internal node t_n:

- $\Pi_{n-1}(t_n) = q_n$ (Connect position)
- $\Pi_{n-1}(t_n) = \Pi_n(t_n)$ (Continuous position)
- $\dot{\Pi}_{n-1}(t_n) = \dot{\Pi}_n(t_n)$ (Continuous velocity)
- $\ddot{\Pi}_{n-1}(t_n) = \ddot{\Pi}_n(t_n)$ (Continuous acceleration)

However, for $k = 1, \ldots, N$, the terms $\Pi_0(t_1)$ and $\Pi_N(t_N)$ do not exist. This creates a mathematical inconsistency in the boundary segments that must be resolved with additional conditions.

4.3.4 Virtual Points Approach

To satisfy all continuity constraints, we analyze the balance between equations and unknowns. For $N-2$ intermediate waypoints, we have 4 equations each, totaling $4(N-2)$ equations. Adding the 3 equations for the initial point and 3 for the final point (+6 equations), we obtain:

$$4(N-2) + 6 = 4N - 2 \text{ equations} \tag{4.23}$$

However, with $N-1$ segments, there are $4(N-1)$ unknowns. Since $4N - 2 > 4(N-1)$, the system is overdetermined. To solve this, we introduce 2 virtual points, resulting in $N+2$ total points and $N+1$ polynomials.

Consider $N+2$ time instants t_k, where t_2 and t_{N+1} refer to the virtual points.

For the $N-2$ intermediate points ($k = 3, \ldots, N$), we define 4 equations each:

$$\Pi_{k-1}(t_k) = q_k \tag{4.24}$$
$$\Pi_{k-1}(t_k) = \Pi_k(t_k) \tag{4.25}$$
$$\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k) \tag{4.26}$$
$$\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k) \tag{4.27}$$

For the initial and final points, we impose position, velocity, and acceleration (6 equations total):

$$\Pi_1(t_1) = q_i \qquad \Pi_{N+1}(t_{N+2}) = q_f \tag{4.28}$$

$$\dot{\Pi}_1(t_1) = \dot{q}_i \qquad \dot{\Pi}_{N+1}(t_{N+2}) = \dot{q}_f \tag{4.29}$$

$$\ddot{\Pi}_1(t_1) = \ddot{q}_i \qquad \ddot{\Pi}_{N+1}(t_{N+2}) = \ddot{q}_f \tag{4.30}$$

Finally, for the virtual points ($k = 2$ and $k = N + 1$), we impose position, velocity, and acceleration continuity (6 equations total):

$$\Pi_{k-1}(t_k) = \Pi_k(t_k) \tag{4.31}$$

$$\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k) \tag{4.32}$$

$$\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k) \tag{4.33}$$

In total, the system now has:

$$4(N-2) + 6 + 6 = 4(N+1) \text{ equations} \tag{4.34}$$

With $N+1$ polynomials, we have exactly $4(N+1)$ unknowns, making the system solvable via standard algorithms.

4.4 Interpolating Linear Polynomials with Parabolic Blends

In this approach, instead of interpolating exactly through all points with high-order curves, we use linear segments and "pass by" the points using parabolic blends.

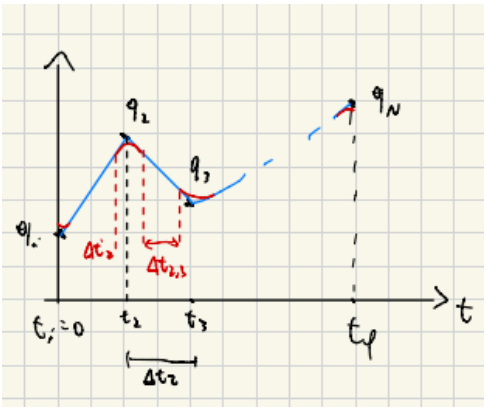


Figure 4.3: Trajectory with linear segments and parabolic blends.

The velocity of the linear segment and the acceleration during the blend are calculated as:

$$\dot{q}_{n-1,n} = \frac{q_n - q_{n-1}}{\Delta t_{n-1}} \tag{4.35}$$

$$\ddot{q}_n = \frac{\dot{q}_{n,n+1} - \dot{q}_{n-1,n}}{\Delta t'_n} \tag{4.36}$$

For example, at the first internal waypoint ($n = 2$):

- $\dot{q}_{1,2} = \frac{q_2 - q_1}{\Delta t_1}$ is the average velocity from q_1 to q_2.
- $\ddot{q}_2 = \frac{\dot{q}_{2,3} - \dot{q}_{1,2}}{\Delta t_2'}$ represents the difference in velocity divided by the time allowed to change.

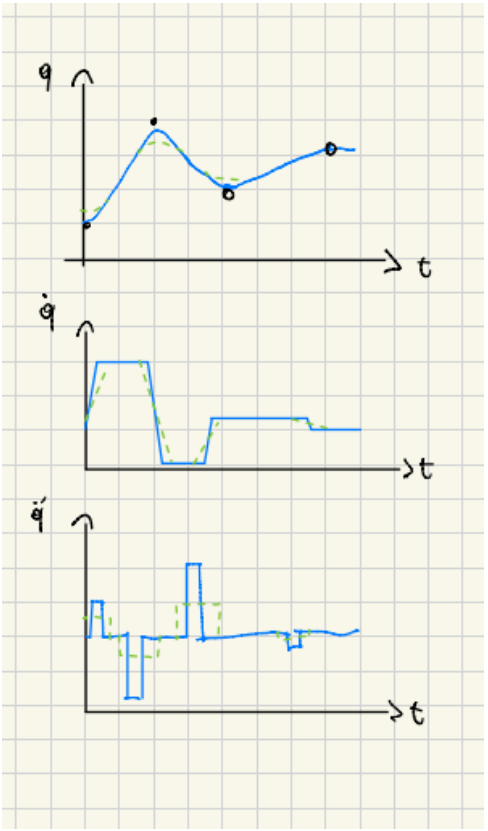


Figure 4.4: Trajectory with linear segments and parabolic blends.

Using trapezoidal profiles allows the trajectory to touch the points exactly, but requires stopping at each one. By summing shifted trapezoidal profiles, we avoid exact point intersection but gain a computationally easier and smoother transition.

4.5 Operational Space Trajectories

In operational space, we define the end-effector position $x_e(t)$ (e.g., a 2×1 or 3×1 vector). Given a sequence of points $P_1, P_2, \ldots, P_N$, we can plan the trajectory $x_e(t_n)$ for $t_n \in [t_1, t_N]$ using various methods:

- Polynomials
- Linear Polynomials with Parabolic Blends
- Splines

The generated operational space trajectory is then passed through **Inverse Kinematics** to obtain the joint space references.

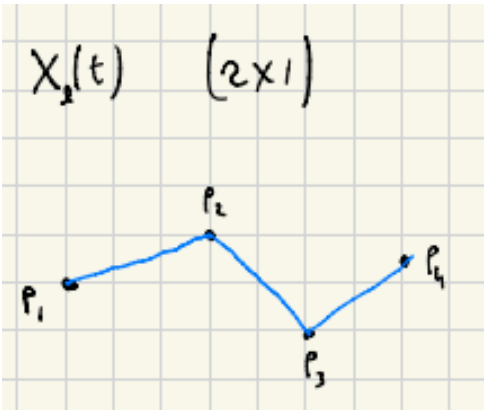


Figure 4.5: Path primitives and inverse kinematics mapping.

4.5.1 Path Primitives

A path Γ is a locus of points in $\mathbb{R}^3$ described by a parameter σ:

$$p = f(\sigma), \quad \sigma \in [\sigma_i, \sigma_f] \tag{4.37}$$

Arc Length as Parameter

By choosing the arc length s as the parameter ($\sigma \equiv s$), we define the path in terms of the distance traveled along it:

$$p = f(s), \quad s \in [0, s_f] \tag{4.38}$$

At any point on the path, we define a **Right Handed Frame** (t, n, b):

- **Tangent**: $t = \frac{dp}{ds}$
- **Normal**: $n = \frac{1}{\|\frac{d^2p}{ds^2}\|}\frac{d^2p}{ds^2}$
- **Binormal**: $b = t \times n$

4.5.2 Rectilinear Path

For a straight line between p_i and p_f:

$$f(s) = p_i + \frac{s}{\|p_f - p_i\|}(p_f - p_i), \quad s \in [0, \|p_f - p_i\|] \tag{4.39}$$

Boundary checks:

- $f(0) = p_i$
- $f(s_f) = p_i + \frac{\|p_f - p_i\|}{\|p_f - p_i\|}(p_f - p_i) = p_f$

The tangent is constant, and the normal is zero:

$$t = \frac{dp}{ds} = \frac{p_f - p_i}{\|p_f - p_i\|}, \quad n = 0 \tag{4.40}$$

4.5.3 Circular Path

To define a circular path, we need an axis $\hat{r}$, a point on the axis d, and a point on the circumference p_i.

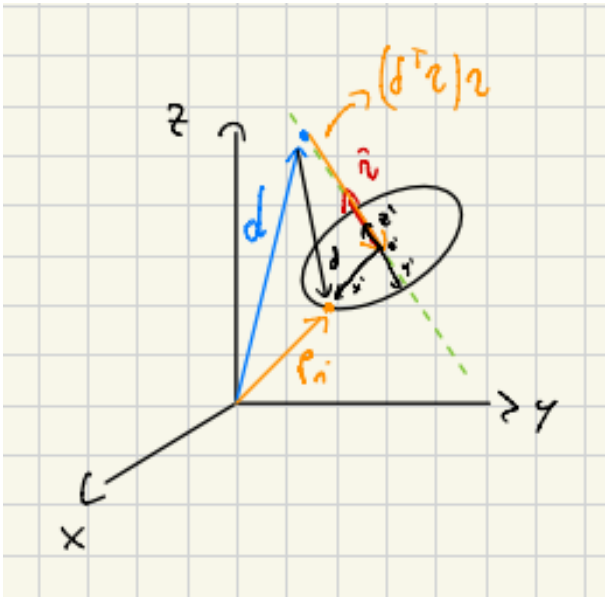


Figure 4.6: Geometry of a circular path in 3D space.

The center of the circle c and the radius ρ are:

$$\delta = p_i - d \tag{4.41}$$

$$c = d + (\delta^T r)r \tag{4.42}$$

$$\rho = \|p_i - c\| \tag{4.43}$$

In a new reference frame $O'x'y'z'$, the path coordinates $p'(s)$ for $s \in [0, 2\pi\rho)$ are:

$$p'(s) = \begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix} \tag{4.44}$$

Transforming back to the base frame using rotation matrix $R = [x'\ y'\ z']$:

$$p(s) = c + Rp'(s) \tag{4.45}$$

The tangent and normal vectors are derived as:

$$t = \frac{dp}{ds} = R\frac{dp'(s)}{ds} = R\begin{bmatrix} -\sin(s/\rho) \\ \cos(s/\rho) \\ 0 \end{bmatrix} \tag{4.46}$$

$$n = \frac{d^2p}{ds^2} = R\begin{bmatrix} -\cos(s/\rho)/\rho \\ -\sin(s/\rho)/\rho \\ 0 \end{bmatrix} \tag{4.47}$$

4.6 Time Law along a Path

To complete the trajectory description, we need to define how the path parameter s evolves over time, denoted as $s(t)$. This is the **Time Law**.

The end-effector position is defined as:

$$p_e(t) = f(s(t)) \tag{4.48}$$

The velocity and acceleration profiles are derived using the chain rule:

$$\dot{p}_e(t) = \frac{df(s)}{ds}\dot{s} = \dot{s}\hat{t} \tag{4.49}$$

$$\ddot{p}_e(t) = \ddot{s}\hat{t} + \dot{s}^2\hat{n} \tag{4.50}$$

Where the second term in the acceleration equation represents the **Centripetal Acceleration**.

4.6.1 Analytical Derivation of Acceleration

The acceleration can be derived step-by-step from the velocity:

$$\frac{d}{dt}\left(\frac{df}{ds}\dot{s}\right) = \left(\frac{d}{dt}\left(\frac{df}{ds}\right)\right)\dot{s} + \frac{df}{ds}\ddot{s} \tag{4.51}$$

$$= \left(\frac{d}{ds}\left(\frac{df}{dt}\right)\right)\dot{s} + \frac{df}{ds}\ddot{s} \tag{4.52}$$

$$= \frac{d}{ds}(\dot{s}\hat{t})\dot{s} + \hat{t}\ddot{s} \tag{4.53}$$

$$= \left(\frac{d\dot{s}}{ds}\hat{t} + \dot{s}\frac{d\hat{t}}{ds}\right)\dot{s} + \hat{t}\ddot{s} \tag{4.54}$$

Given that $\frac{d\hat{t}}{ds} = \frac{1}{\rho}\hat{n} = \kappa\hat{n}$, where κ is the curvature, we obtain:

$$\ddot{p}_e(t) = \ddot{s}\hat{t} + \dot{s}^2\hat{n} \tag{4.55}$$

4.6.2 Kinematics of Path Primitives

Rectilinear Path

For a straight line, the end-effector position and its derivatives are:

$$p_e(t) = p_i + \frac{s(t)}{\|p_f - p_i\|}(p_f - p_i) \tag{4.56}$$

$$\dot{p}_e(t) = \frac{\dot{s}}{\|p_f - p_i\|}(p_f - p_i) = \dot{s}\hat{t} \tag{4.57}$$

$$\ddot{p}_e(t) = \frac{\ddot{s}}{\|p_f - p_i\|}(p_f - p_i) = \ddot{s}\hat{t} \tag{4.58}$$

Note: A rectilinear path involves **only linear acceleration**, as $\hat{n} = 0$.

Circular Path

For a circular path, using the rotation matrix R:

$$p_e = c + R \begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix} \tag{4.59}$$

$$\dot{p}_e = R \begin{bmatrix} -\dot{s}\sin(s/\rho) \\ \dot{s}\cos(s/\rho) \\ 0 \end{bmatrix} = \dot{s}\hat{t} \tag{4.60}$$

$$\ddot{p}_e = R \begin{bmatrix} -\frac{\dot{s}^2}{\rho}\cos(s/\rho) - \ddot{s}\sin(s/\rho) \\ -\frac{\dot{s}^2}{\rho}\sin(s/\rho) + \ddot{s}\cos(s/\rho) \\ 0 \end{bmatrix} = \ddot{s}\hat{t} + \dot{s}^2\hat{n} \tag{4.61}$$

4.7 Multi-point Trajectories with Anticipated Timing

Consider $N+1$ points in the operational space. The path consists of N segments connected linearly. The position of the end-effector is described by:

$$p_e(t) = p_0 + \sum_{j=1}^{N} \frac{s_j(t)}{\|p_j - p_{j-1}\|}(p_j - p_{j-1}) \tag{4.62}$$

The curvilinear abscissa $s_j(t)$ for the j-th segment is defined as:

$$s_j(t) = \begin{cases} 0 & 0 \le t \le t_{j-1} \\ s_j(t) & t_{j-1} < t \le t_j \\ \|p_j - p_{j-1}\| & t_j < t \le t_N \end{cases} \tag{4.63}$$

Velocity and acceleration profiles are obtained by differentiation:

$$\dot{p}_e(t) = \sum_{j=1}^{N} \frac{\dot{s}_j(t)}{\|p_j - p_{j-1}\|}(p_j - p_{j-1}) = \sum_{j=1}^{N} \dot{s}_j\hat{t}_j \tag{4.64}$$

$$\ddot{p}_e(t) = \sum_{j=1}^{N} \ddot{s}_j\hat{t}_j \tag{4.65}$$

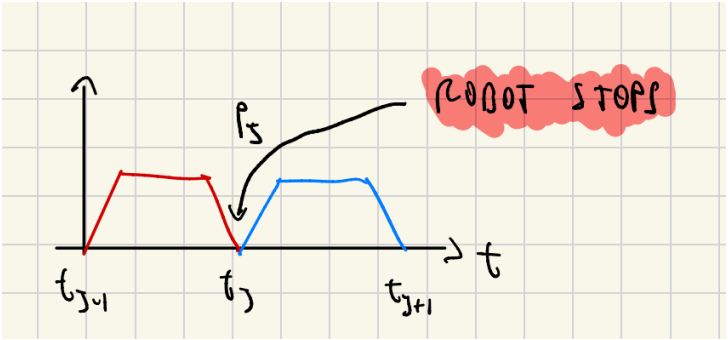


Figure 4.7: Velocity profile where the robot stops at each waypoint p_j.

To avoid stops at waypoints and ensure a smooth transition, the second trapezoidal profile is anticipated by a time interval δt_j.

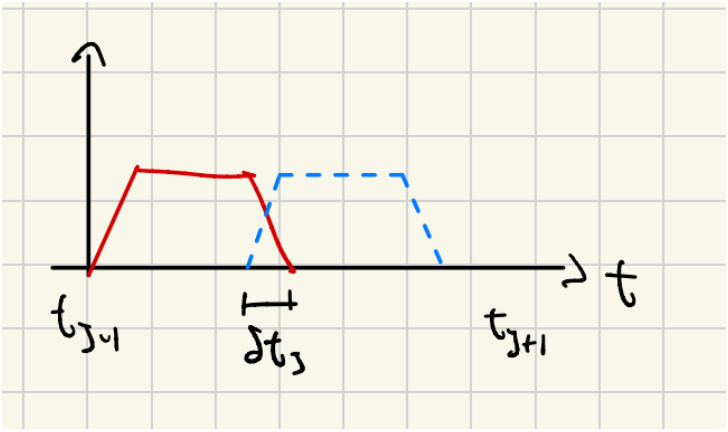


Figure 4.8: Anticipated velocity profile for trajectory blending.

The shifted curvilinear abscissa $s_j(t)$ is defined by:

$$s_j(t) = \begin{cases} 0 & 0 \leq t \leq t_{j-1} - \Delta t_j \\ s_j(t + \Delta t_j) & t_{j-1} - \Delta t_j < t \leq t_j - \Delta t_j \\ \|p_j - p_{j-1}\| & t_j - \Delta t_j < t \leq t_N - \Delta t_N \end{cases} \tag{4.66}$$

The cumulative time shift Δt_j follows the recursive relation:

$$\Delta t_j = \Delta t_{j-1} + \delta t_j \tag{4.67}$$

$$\Delta t_0 = 0 \tag{4.68}$$

4.8 Orientation Trajectories

The orientation of the end-effector is represented by the rotation matrix $R = \begin{bmatrix} n_e(t) & s_e(t) & a_e(t) \end{bmatrix}$, describing the transition from an initial orientation R_i to a final orientation R_f. Since it is numerically difficult to ensure that R remains orthonormal throughout the motion, alternative representations are used.

4.8.1 Euler Angles

Using Euler angles ϕ, the trajectory can be interpolated linearly in the angle space:

$$\phi_e = \phi_i + \frac{s(t)}{\|\phi_f - \phi_i\|}(\phi_f - \phi_i) \tag{4.69}$$

The corresponding angular velocity and acceleration in the Euler space are:

$$\dot{\phi}_e = \frac{\dot{s}}{\|\phi_f - \phi_i\|}(\phi_f - \phi_i) \tag{4.70}$$

$$\ddot{\phi}_e = \frac{\ddot{s}}{\|\phi_f - \phi_i\|}(\phi_f - \phi_i) \tag{4.71}$$

The physical angular velocity ω_e can then be derived using the transformation matrix $T(\phi_e)$:

$$\omega_e = T(\phi_e)\dot{\phi}_e \tag{4.72}$$

4.8.2 Axis-Angle Representation

To avoid the singularities associated with Euler angles, the Axis-Angle representation can be employed. We define the relative rotation matrix R_f^i such that $R_f = R_i R_f^i$:

$$R_f^i = R_i^T R_f = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.73}$$

From R_f^i, we derive the total rotation angle θ_f and the unit rotation axis r:

$$\theta_f = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{4.74}$$

$$r = \frac{1}{2\sin(\theta_f)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \quad \text{for } \sin(\theta_f) \neq 0 \tag{4.75}$$

The time-varying orientation is then $R(t) = R_i R^i(t)$, with boundary conditions $R^i(0) = I$ and $R^i(t_f) = R_f^i$. Since the axis r is fixed in space, we can apply polynomials to generate a time law for $\theta(t)$. The angular velocity and acceleration are:

$$\omega_e^i = \dot{\theta} r \tag{4.76}$$

$$\dot{\omega}_e^i = \ddot{\theta} r \tag{4.77}$$

4.9 Dynamics: Equations of Motion of the Manipulator

The study of manipulator dynamics establishes the relationship between the forces and moments applied to the structure and the resulting motion over time.

4.9.1 Lagrangian Formulation

The dynamic model can be derived using the Lagrangian approach, where the Lagrangian function L is defined as the difference between kinetic energy T and potential energy U:

$$L = T - U \tag{4.78}$$

The equations of motion are obtained through the Euler-Lagrange equations:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \xi_i \tag{4.79}$$

where q_i represents the generalized coordinates and ξ_i represents the non-conservative generalized forces (torques and frictions).

4.9.2 Case Study: Single Joint System

Considering a system with a motor and a load (link) connected via a transmission with gear ratio $K_r \gg 1$. Let I_m be the motor inertia and I the load inertia.

The external forces and torques are related by:

$$\theta_m = K_r\theta, \quad \tau = K_r\tau_m \tag{4.80}$$

The kinetic energy T and potential energy U are:

$$T = \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}I_m\dot{\theta}_m^2 = \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}I_mK_r^2\dot{\theta}^2 \tag{4.81}$$

$$U = mgl(1 - \cos\theta) \tag{4.82}$$

The resulting Lagrangian is:

$$L = \frac{1}{2}(I + I_mK_r^2)\dot{\theta}^2 - mgl(1 - \cos\theta) \tag{4.83}$$

Including friction terms F (link) and F_m (motor), the equation of motion becomes:

$$(I + I_mK_r^2)\ddot{\theta} + mgl\sin\theta = \tau - F\dot{\theta} - F_mK_r^2\dot{\theta} \tag{4.84}$$

Rearranging into the final equation of motion:

$$(I + I_mK_r^2)\ddot{\theta} + (F + F_mK_r^2)\dot{\theta} + mgl\sin\theta = \tau \tag{4.85}$$

4.10 Computation of Kinetic Energy

For a manipulator with n links, the total kinetic energy is the sum of the kinetic energies of the links and the motors:

$$T = \sum_{i=1}^{n}(T_{li} + T_{mi}) \tag{4.86}$$

where T_{li} is the kinetic energy of link i and T_{mi} is the kinetic energy of the motor actuating joint i.

4.10.1 Kinetic Energy of Link i

The kinetic energy of a rigid body (link i) is calculated by integrating the velocity of its elementary masses:

$$T_{li} = \frac{1}{2}\int_{V_{li}} \dot{P}_i^{*T}\dot{P}_i^*\rho dV \tag{4.87}$$

where $\dot{P}_i^*$ is the linear velocity of a generic point on the link.

Velocity of a Generic Point

Defining P_{li} as the position of the center of mass:

$$P_{li} = \frac{1}{m_{li}}\int_{V_{li}} P_i^*\rho dV \tag{4.88}$$

The velocity of a generic point P_i^* can be expressed as:

$$P_i^* = P_{li} + \omega_i \times z_i = \dot{P}_{li} + S(\omega_i)z_i \tag{4.89}$$

where $S(\omega_i)$ is the skew-symmetric matrix and $z_i = P_i^* - P_{li}$ is the distance from the center of mass.

Energy Terms Decomposition

Substituting the velocity into the integral yields three terms:

1. **Translation Term:**
$$\frac{1}{2}\int_{V_{li}} \dot{P}_{li}^T \dot{P}_{li}\rho dV = \frac{1}{2} m_{li} \dot{P}_{li}^T \dot{P}_{li} \tag{4.90}$$

2. **Mutual Term:**
$$2\left(\frac{1}{2}\int_{V_{li}} \dot{P}_{li}^T S(\omega_i) z_i \rho dV\right) = \dot{P}_{li}^T S(\omega_i) \int_{V_{li}} (P_i^* - P_{li})\rho dV = 0 \tag{4.91}$$

   This term vanishes because the integral of the position relative to the center of mass is zero.

3. **Rotational Term:** To be detailed in the following section, involving the inertia tensor.

Rotational Term and Inertia Tensor

The rotational component of the kinetic energy for link i is derived from the velocity distribution relative to the center of mass:

$$\frac{1}{2}\int_{V_{li}} z_i^T S^T(\omega_i) S(\omega_i) z_i \rho dV \tag{4.92}$$

Using the property $S(\omega_i)z_i = -S(z_i)\omega_i$, we can extract the angular velocity ω_i from the integral:

$$T_{rot,i} = \frac{1}{2}\omega_i^T \left(\int_{V_{li}} S^T(z_i) S(z_i) \rho dV\right) \omega_i = \frac{1}{2}\omega_i^T I_{li} \omega_i \tag{4.93}$$

where I_{li} is the inertia tensor of link i expressed in the base frame. The skew-symmetric matrix $S(a)$ is defined as:

$$S(a) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{4.94}$$

4.10.2 Frame Transformation for Inertia Tensor

Since I_{li} depends on the manipulator's configuration when expressed in the base frame, we use the rotation matrix R_i to express it relative to a constant tensor $\bar{I}_{li}^i$ in the local link frame:

$$I_{li} = R_i \bar{I}_{li}^i R_i^T \tag{4.95}$$

The angular velocity in the base frame is $\omega_i = R_i \omega_i^i$. Thus, the kinetic energy of the link is:

$$T_{li} = \frac{1}{2} m_{li} \dot{P}_{li}^T \dot{P}_{li} + \frac{1}{2}\omega_i^T R_i \bar{I}_{li}^i R_i^T \omega_i \tag{4.96}$$

4.10.3 Velocity Representation via Jacobians

To express T_{li} as a function of the joint variables q and their derivatives $\dot{q}$, we utilize the geometric Jacobians for linear and angular velocities.

Linear Velocity Jacobian

The velocity of the center of mass of link i is:

$$\dot{P}_{li} = J_P^{(li)}\dot{q} = \sum_{j=1}^{i} j_{Pj}^{(li)}\dot{q}_j \tag{4.97}$$

where the columns of the Jacobian are defined based on the joint type:

$$j_{Pj}^{(li)} = \begin{cases} z_{j-1} & \text{if joint } j \text{ is prismatic} \\ z_{j-1} \times (P_{li} - P_{j-1}) & \text{if joint } j \text{ is revolute} \end{cases} \tag{4.98}$$

For $j > i$, $j_{Pj}^{(li)} = 0$.

Angular Velocity Jacobian

Similarly, the angular velocity is expressed as:

$$\omega_i = J_O^{(li)}\dot{q} = \sum_{j=1}^{i} j_{Oj}^{(li)}\dot{q}_j \tag{4.99}$$

with the columns defined as:

$$j_{Oj}^{(li)} = \begin{cases} 0 & \text{if joint } j \text{ is prismatic} \\ z_{j-1} & \text{if joint } j \text{ is revolute} \end{cases} \tag{4.100}$$

Link Kinetic Energy in Joint Space

Combining the above, the final expression for the kinetic energy of link i is:

$$T_{li} = \frac{1}{2}\dot{q}^T \left[m_{li} J_P^{(li)T} J_P^{(li)} + J_O^{(li)T} R_i \bar{I}_{li}^i R_i^T J_O^{(li)} \right] \dot{q} \tag{4.101}$$

4.11 Kinetic Energy of the Motors

The motor for joint i is typically mounted on link $i-1$. Its motion depends on the motion of the preceding links plus its own rotation.

4.11.1 Motor Velocity Kinematics

The angular velocity of the i-th motor, ω_{mi}, is:

$$\omega_{mi} = \omega_{i-1} + k_{ri}\dot{q}_i z_{mi} \tag{4.102}$$

where k_{ri} is the gear ratio and z_{mi} is the motor's axis of rotation.

4.11.2 Total Motor Kinetic Energy

The kinetic energy T_{mi} includes translational and rotational terms:

$$T_{mi} = \frac{1}{2} m_{mi} \dot{P}_{mi}^T \dot{P}_{mi} + \frac{1}{2} \omega_{mi}^T I_{mi} \omega_{mi} \tag{4.103}$$

Expressing this in terms of the joint space velocities:

$$T_{mi} = \frac{1}{2} \dot{q}^T \left[m_{mi} J_P^{(mi)T} J_P^{(mi)} + J_O^{(mi)T} R_{mi} \bar{I}_{mi}^m R_{mi}^T J_O^{(mi)} \right] \dot{q} \tag{4.104}$$

The motor Jacobian $J_O^{(mi)}$ accounts for the transmission ratio k_{ri} at the i-th entry.

4.12 Total Kinetic Energy and Inertia Matrix

The total kinetic energy of the manipulator is obtained by summing the contributions of all links and motors. It can be expressed in a quadratic form with respect to the joint velocities:

$$T = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij}(q) \dot{q}_i \dot{q}_j = \frac{1}{2} \dot{q}^T B(q) \dot{q} \tag{4.105}$$

where $B(q)$ is the $(n \times n)$ symmetric and positive definite inertia matrix.

4.12.1 Structure of the Inertia Matrix

The elements b_{ij} of the inertia matrix $B(q)$ are calculated by summing the translational and rotational components of both links and motors:

$$B(q) = \sum_{i=1}^{n} \left(m_{li} J_P^{(li)T} J_P^{(li)} + J_O^{(li)T} R_i \bar{I}_{li}^i R_i^T J_O^{(li)} + m_{mi} J_P^{(mi)T} J_P^{(mi)} + J_O^{(mi)T} R_{mi} \bar{I}_{mi}^m R_{mi}^T J_O^{(mi)} \right) \tag{4.106}$$

4.13 Potential Energy

The total potential energy U is the sum of the potential energies of each link U_{li} and motor U_{mi}:

$$U = \sum_{i=1}^{n} (U_{li} + U_{mi}) \tag{4.107}$$

Assuming a gravity vector g in the base frame:

- For the link: $U_{li} = -m_{li} g^T P_{li}$
- For the motor: $U_{mi} = -m_{mi} g^T P_{mi}$

The total potential energy is a function of the configuration q:

$$U(q) = -\sum_{i=1}^{n} (m_{li} g^T P_{li} + m_{mi} g^T P_{mi}) \tag{4.108}$$

4.14 Equations of Motion

Applying the Lagrange equations to the Lagrangian $L(q, \dot{q}) = T(q, \dot{q}) - U(q)$:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} = \tau_i \tag{4.109}$$

4.14.1 Detailed Derivation

Expanding the kinetic energy term:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) = \sum_{j=1}^{n} b_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n} \frac{db_{ij}}{dt}\dot{q}_j = \sum_{j=1}^{n} b_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial b_{ij}}{\partial q_k}\dot{q}_k\dot{q}_j \tag{4.110}$$

The derivative of the kinetic energy with respect to the coordinates is:

$$\frac{\partial T}{\partial q_i} = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial b_{jk}}{\partial q_i}\dot{q}_j\dot{q}_k \tag{4.111}$$

Defining $g_i(q) = \frac{\partial U}{\partial q_i}$ as the gravity torque, the equations of motion for each joint i are:

$$\sum_{j=1}^{n} b_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}(q)\dot{q}_j\dot{q}_k + g_i(q) = \tau_i \tag{4.112}$$

4.15 Components of the Torques and Non-Conservative Forces

The equations of motion for each joint i can be expanded to account for various physical effects:

- **Inertia terms:** $b_{ii}(q)$ is the moment of inertia at joint i. $b_{ij}(q)$ represents the effect of the acceleration of joint j on joint i.
- **Coriolis and Centrifugal terms:** $c_{ijj}(q)\dot{q}_j^2$ is the centrifugal effect on joint i by the velocity of joint j. $c_{ijk}(q)\dot{q}_j\dot{q}_k$ is the Coriolis effect on joint i by the velocities of joints j and k.
- **Gravity terms:** $g_i(q)$ is the torque generated at joint i axis by gravity.

4.15.1 Non-Conservative Forces

The generalized forces τ include several non-conservative components:

- **Actuation torques:** u.
- **Viscous friction torques:** $-F_v\dot{q}$.
- **Static friction torques:** $-F_s\text{sgn}(\dot{q})$.
- **Contact forces:** $-J^T(q)h_e$, representing the balancing of contact forces and torques through the Jacobian $J(q)$ and the wrench h_e.

The dynamic model in compact form is:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\text{sgn}(\dot{q}) + g(q) = u - J^T(q)h_e \tag{4.113}$$

where $C(q,\dot{q})$ is an $n \times n$ matrix whose elements are $c_{ij} = \sum_{k=1}^{n} c_{ijk}(q)\dot{q}_k$.

4.16 Notable Properties

The dynamic model possesses significant properties used in the design of control laws.

4.16.1 Skew-Symmetry

The matrix $N(q,\dot{q}) = \dot{B}(q) - 2C(q,\dot{q})$ is skew-symmetric. This means:

$$N(q,\dot{q}) = -N^T(q,\dot{q}) \tag{4.114}$$

and consequently, for any vector w:

$$w^T(\dot{B}(q) - 2C(q,\dot{q}))w = 0 \tag{4.115}$$

This property follows from the principle of conservation of energy. The time derivative of the kinetic energy is balanced by the power generated by the forces on the joints:

$$\frac{1}{2}\frac{d}{dt}(\dot{q}^T B(q)\dot{q}) = \dot{q}^T(u - F_v\dot{q} - F_s\text{sgn}(\dot{q}) - g(q) - J^T(q)h_e) \tag{4.116}$$

Since $\frac{d}{dt}(\dot{q}^T B\dot{q}) = \dot{q}^T B\ddot{q} + \frac{1}{2}\dot{q}^T\dot{B}\dot{q}$, substituting the equations of motion leads to the condition $\dot{q}^T(\dot{B} - 2C)\dot{q} = 0$, which must be skew-symmetrical for any $q, \dot{q}$.

4.17 Linearity in the Dynamic Parameters

The dynamic model of a robot is linear with respect to a specific set of dynamic parameters. To demonstrate this, a pivot point P_{p_i} is selected for each link:

- $P_{p_i} = P_{i-1}$ for revolute joints.
- $P_{p_i} = P_i$ for prismatic joints.

Let P_{c_i} be the center of mass of the link. The velocity of the generic point is defined as $v_i = v_{0i} + \omega_i \times z_i$. Defining the distance from the pivot as $r_i = P_{c_i} - P_{p_i}$, we observe that the dynamic parameters do not depend on the choice of the reference frame.

4.17.1 Augmented Kinetic Energy and Potential Energy

The kinetic energy can be rewritten as a function of the mass m_i, the first moment of inertia $m_i r_i$, and the inertia tensor I_i. Similarly, the potential energy V_i is expressed as:

$$V_i = -m_i g^T P_{c_i} = -g^T(m_i P_{p_i} + m_i r_i) \tag{4.117}$$

This confirms that the potential energy is a linear function of the mass and the first moment of inertia.

4.17.2 The Lagrangian and the Parameter Vector

The Lagrangian $L = T - U$ is linear with respect to the vector π. For each link, this vector includes:

- The mass m_i.
- The first moments of inertia: $m_i r_{ix}, m_i r_{iy}, m_i r_{iz}$.
- The six independent components of the inertia tensor.

Under the assumption that the rotors are modeled as uniform bodies with their center of mass on the rotation axis, and that the angular velocity of each rotor is dominated by its own spin, the equations of motion maintain this linearity:

$$\tau = Y(q, \dot{q}, \ddot{q})\pi \tag{4.118}$$

where Y is the regressor matrix. This model is triangular because the torque on a joint does not depend on the parameters of the preceding links.

4.18 Dynamics of a 2-Link Planar Manipulator

4.18.1 Kinematic Characterization

Considering a planar manipulator with two links, we define the positions of the centers of mass:

- $P_{l1} = (l_{c1} \cos q_1, l_{c1} \sin q_1)$
- $P_{l2} = (l_1 \cos q_1 + l_{c2} \cos(q_1 + q_2), l_1 \sin q_1 + l_{c2} \sin(q_1 + q_2))$

The position of the motor mass m_{m2} (mounted on link 1) is:

- $P_{m2} = (l_1 \cos q_1, l_1 \sin q_1)$

4.18.2 Jacobian Matrices

The geometric Jacobians for the links and the motor are derived as follows:

Link 1 Jacobians

Only q_1 contributes to the motion of Link 1:

$$J_P^{(l1)} = \begin{bmatrix} -l_{c1} \sin q_1 & 0 \\ l_{c1} \cos q_1 & 0 \\ 0 & 0 \end{bmatrix}, \quad J_O^{(l1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{4.119}$$

Link 2 Jacobians

Both q_1 and q_2 contribute to the rotation and translation of Link 2:

$$J_P^{(l2)} = \begin{bmatrix} -l_1 \sin q_1 - l_{c2} \sin(q_1 + q_2) & -l_{c2} \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_{c2} \cos(q_1 + q_2) & l_{c2} \cos(q_1 + q_2) \\ 0 & 0 \end{bmatrix}, \quad J_O^{(l2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{4.120}$$

Motor 2 Jacobians

The motor for the second joint is mounted on Link 1:

$$J_P^{(m2)} = \begin{bmatrix} -l_1 \sin q_1 & 0 \\ l_1 \cos q_1 & 0 \\ 0 & 0 \end{bmatrix}, \quad J_O^{(m2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & k_{r2} \end{bmatrix} \tag{4.121}$$

where k_{r2} is the gear ratio of the second motor.

4.18.3 Inertia Matrix $B(q)$

The inertia matrix is computed by summing the contributions:

$$B(q) = \sum_{i=1}^{2} \left(m_{li} J_P^{(li)T} J_P^{(li)} + J_O^{(li)T} R_i \bar{I}_{li}^i R_i^T J_O^{(li)} + m_{mi} J_P^{(mi)T} J_P^{(mi)} + J_O^{(mi)T} R_{mi} \bar{I}_{mi}^m R_{mi}^T J_O^{(mi)} \right) \tag{4.122}$$

The specific elements of $B(q)$ for the 2-link planar robot are:

- $b_{11} = I_{l1} + m_{l1} l_{c1}^2 + k_{r1}^2 I_m + I_{l2} + m_{l2}(l_1^2 + l_{c2}^2 + 2 l_1 l_{c2} \cos q_2) + I_{m2} + m_{m2} l_1^2$
- $b_{12} = b_{21} = I_{l2} + m_{l2}(l_{c2}^2 + l_1 l_{c2} \cos q_2) + k_{r2} I_{m2}$
- $b_{22} = I_{l2} + m_{l2} l_{c2}^2 + k_{r2}^2 I_{m2}$

4.19 Identification of Dynamic Parameters

Identification of dynamic parameters is a process similar to kinematic calibration, but applied to the inertial and friction properties of the manipulator. This process is essential for high-performance simulation and the implementation of advanced control algorithms.

4.19.1 Identification Strategy

The procedure relies on the linearity of the dynamic model with respect to the parameter vector π:

$$\tau = Y(q, \dot{q}, \ddot{q}) \pi \tag{4.123}$$

To identify the unknown parameters, the robot is moved along "exciting" trajectories while measuring the following quantities:

- Joint positions q (typically measured via encoders).
- Joint velocities $\dot{q}$ and accelerations $\ddot{q}$ (often obtained through numerical differentiation and filtering of position data).
- Joint torques τ (measured via torque sensors or estimated from motor currents u using the torque constant K_t).

4.19.2 Least-Squares Estimation

By collecting N sets of measurements at different time instants, we can construct an overdetermined linear system:

$$\bar{\tau} = \bar{Y}\pi \tag{4.124}$$

where $\bar{\tau}$ is the vector of all measured torques ($Nn \times 1$) and $\bar{Y}$ is the global regressor matrix ($Nn \times p$).

The parameter vector π is estimated using the Least-Squares technique:

$$\hat{\pi} = (\bar{Y}^T\bar{Y})^{-1}\bar{Y}^T\bar{\tau} \tag{4.125}$$

The matrix $(\bar{Y}^T\bar{Y})^{-1}\bar{Y}^T$ is known as the left pseudo-inverse of $\bar{Y}$.

4.19.3 Numerical Robustness

In practice, the matrix $\bar{Y}^T\bar{Y}$ might be ill-conditioned if the chosen trajectories do not sufficiently excite all the dynamic degrees of freedom. This occurs when the rank of the regressor matrix is lower than the number of parameters ($rank(\bar{Y}) < p$). In such cases, the Damped Least Squares (DLS) method is employed to improve numerical stability:

$$\hat{\pi} = (\bar{Y}^T\bar{Y} + \lambda^2 I)^{-1}\bar{Y}^T\bar{\tau} \tag{4.126}$$

where λ is a damping factor that prevents the inversion of singular or near-singular values.

4.20 Trajectory Requirements for Identification

The trajectory used for the identification of dynamic parameters must be "rich". This means that during the sampling process, it must ensure the presence of contributions from all dynamic terms so they can be mathematically isolated and identified.

4.20.1 Limitations of Simple Trajectories

The choice of an inadequate trajectory leads to poor identification results:

- **Trapezoidal Velocity Profile:** This type of profile includes segments of constant velocity where acceleration is zero ($\ddot{q} = 0$). This lack of acceleration makes it impossible to identify the inertial terms of the model.
- **High-Degree Polynomials:** While more complex, increasing the degree of the trajectory polynomial can lead to a decrease in numerical accuracy. Furthermore, high-frequency components might excite the elastic dynamics of the structure, which contradicts the fundamental assumption of link rigidity.

4.21 Newton-Euler Formulation

The Newton-Euler formulation is based on a balance of all the forces acting on the generic link i within a robotic structure. We consider the interaction between adjacent links:

- f_i: Force exerted by link $i-1$ on link i.

- $-f_{i+1}$: Force exerted by link $i+1$ on link i.
- μ_i: Moment exerted by link $i-1$ on link i with respect to the origin of frame $i-1$.
- $-\mu_{i+1}$: Moment exerted by link $i+1$ on link i with respect to the origin of frame i.

4.21.1 Fundamental Laws of Motion

To describe the dynamic behavior, we apply Newton's and Euler's laws to each link.

Newton Law

The balance of linear forces is expressed as:

$$f_i - f_{i+1} + m_i g_0 = m_i \ddot{p}_{ci} \tag{4.127}$$

where m_i is the mass of the link and $\ddot{p}_{ci}$ is the acceleration of the center of mass.

Euler Law

The balance of moments is given by:

$$\mu_i + f_i \times r_{i-1,c_i} - \mu_{i+1} - f_{i+1} \times r_{i,c_i} = \frac{d}{dt}\left(\bar{I}_i \omega_i + k_{m,i+1}\dot{q}_{i+1} I_{m,i+1} z_{m,i+1}\right) \tag{4.128}$$

Assuming $\bar{I}_i$ is constant in the local frame, the derivative of the angular momentum $\frac{d}{dt}(I_i\omega_i)$ can be expanded as:

$$\frac{d}{dt}(I_i\omega_i) = \frac{d}{dt}(R_i \bar{I}_i^i R_i^T \omega_i) \tag{4.129}$$

Applying the derivative chain rule and the properties of the skew-symmetric matrix $S(\omega_i)$:

$$= S(\omega_i) R_i \bar{I}_i^i R_i^T \omega_i + R_i \bar{I}_i^i R_i^T \dot{\omega}_i = \bar{I}_i \dot{\omega}_i + \omega_i \times (\bar{I}_i \omega_i) \tag{4.130}$$

Considering the rotor contribution (where $k_{m,i+1}\dot{q}_{i+1} I_{m,i+1} z_{m,i+1}$ represents the motor's angular momentum), the complete derivative for the moment equation becomes:

$$\frac{d}{dt}(\dot{q}_{i+1} I_{m,i+1} z_{m,i+1}) = \ddot{q}_{i+1} I_{m,i+1} z_{m,i+1} + \dot{q}_{i+1} I_{m,i+1} \omega_i \times z_{m,i+1} \tag{4.131}$$

Resulting in the final moment balance:

$$\mu_i + f_i \times r_{i-1,c_i} - \mu_{i+1} - f_{i+1} \times r_{i,c_i} = \bar{I}_i \dot{\omega}_i + \omega_i \times (\bar{I}_i \omega_i) + k_{m,i+1}\ddot{q}_{i+1} I_{m,i+1} z_{m,i+1} + k_{m,i+1}\dot{q}_{i+1} I_{m,i+1} \omega_i \times z_{m,i+1} \tag{4.132}$$

4.21.2 Joint Generalized Force

To find the torque (or force) τ_i applied by the motor at joint i, we project the force and moment onto the motion axis z_{i-1}.

For a **Prismatic** joint:

$$\tau_i = f_i^T z_{i-1} + k_{m,i}\ddot{q}_i I_{m,i} \tag{4.133}$$

For a **Revolute** joint:

$$\tau_i = \mu_i^T z_{i-1} + k_{m,i}\ddot{q}_i I_{m,i} \tag{4.134}$$

4.21.3 Kinematic Propagation

To solve the Newton-Euler equations, we first need to compute the velocities and accelerations for each link. The angular velocity is propagated as:

$$\omega_i = \begin{cases} \omega_{i-1} & \text{Prismatic} \\ \omega_{i-1} + \dot{q}_i z_{i-1} & \text{Revolute} \end{cases} \tag{4.135}$$

The linear velocity of the origin of frame i is:

$$v_i = \begin{cases} v_{i-1} + \dot{d}_i z_{i-1} + \omega_i \times r_{i-1,i} & \text{Prismatic} \\ v_{i-1} + \omega_i \times r_{i-1,i} & \text{Revolute} \end{cases} \tag{4.136}$$

By differentiating these expressions, we obtain the acceleration of the center of mass and the angular acceleration, which are necessary for the Newton-Euler equations.

4.21.4 Recursive Algorithm

The algorithm is structured in two main passes:

1. **Forward Recursion**: From the base to the end-effector ($i = 1 \dots n$). Initial conditions: $\omega_0, \dot{\omega}_0, p_0, \dot{p}_0$. Compute: $\omega_i, \dot{\omega}_i, \ddot{p}_i, \ddot{p}_{ci}$.
2. **Backward Recursion**: From the end-effector to the base ($i = n \dots 1$). Knowing the end-effector force $h_e = [f_{n+1}^T \mu_{n+1}^T]^T$, we compute the internal forces f_i and moments μ_i to finally find τ_i.

Note: All quantities should be converted into their own local frames so that the inertia tensors I_i remain constant.

4.22 Comparison: Lagrange vs. Newton-Euler

The choice between the two main formulations depends on the specific application and the required analytical depth.

4.22.1 Lagrange Formulation

- It is generally easier to understand and more intuitive for complex systems.
- Provides a compact form of the equations of motion.
- Allows for easy identification of the inertia matrix, centrifugal and Coriolis terms, and gravity components.
- It is particularly suitable when accounting for more complex mechanical effects, such as elastic deformations.

4.22.2 Newton-Euler Formulation

- It is a recursive method that is computationally very efficient.
- Highly effective for real-time applications and high-degree-of-freedom systems.

4.23 Direct and Inverse Dynamics

The relationship between joint torques and the resulting motion is categorized into two problems.

4.23.1 Direct Dynamics

In the direct dynamics problem, we determine the joint accelerations $\ddot{q}$ given the applied torques τ and the current state $(q, \dot{q})$.

- It requires the inversion of the inertia matrix B^{-1}.
- It is primarily useful for simulation purposes, where we want to predict the robot's motion.

4.23.2 Inverse Dynamics

In the inverse dynamics problem, we calculate the torques τ required to achieve a desired acceleration $\ddot{q}$, given the current state $(q, \dot{q})$.

- It is useful for control and trajectory planning.
- Newton-Euler is the preferred method to solve the inverse dynamics efficiently.

4.23.3 Computational Complexity

If we have n joints, the number of operations required is:

- $O(n^3)$ for Direct Dynamics (due to matrix inversion).
- $O(n)$ for Inverse Dynamics (due to the recursive nature of Newton-Euler).

4.24 Implementation Details

4.24.1 Direct Dynamics with Lagrange

The equation of motion can be written as:

$$\ddot{q} = B^{-1}(q)(\tau - \tau') \tag{4.137}$$

where τ' accounts for several components:

$$C'(q, \dot{q}) = C(q, \dot{q})\dot{q} + F_v\dot{q} + f(q) + g(q) + J^T(q)h_e \tag{4.138}$$

This includes Coriolis/centrifugal forces, viscous and static friction, gravity, and end-effector forces mapped into torques. Given $q(t_k)$, $\dot{q}(t_k)$, and $\tau(t_k)$, we can find $\ddot{q}(t_k)$ and then use numerical integration methods to find the state at the next time step $q(t_{k+1}), \dot{q}(t_{k+1})$.

4.24.2 Using Newton-Euler for Direct Dynamics

While Newton-Euler (NE) is designed for inverse dynamics (calculating τ from $q, \dot{q}, \ddot{q}$), it can be used to build the components needed for direct dynamics.

To compute the inertia matrix $B(q)$ column by column using Newton-Euler:

1. Set gravity $g_0 = 0$.
2. Set velocities $\dot{q} = 0$.
3. For each column i, set the accelerations such that $\ddot{q}_i = 1$ and $\ddot{q}_j = 0$ for $j \neq i$.
4. The resulting torque vector τ calculated by NE will be the i-th column of $B(q)$:

$$\tau = B(q)\ddot{q} = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \vdots \\ \ddot{q}_n \end{bmatrix} \tag{4.139}$$

4.25 Dynamic Scaling of Trajectories

When a planned trajectory requires torques that exceed the physical limits of the motors, we must scale the trajectory in time. We introduce a time warping function $s(t)$, with $t_i \leq t \leq t_f$, such that the new time scale is $\tau = s(t)$.

The relationship between the original trajectory $q(t)$ and the scaled trajectory $\tilde{q}(t)$ is defined as:

- $\tilde{q}(t) = q(s(t))$
- $\dot{\tilde{q}}(t) = \dot{q}(s(t)) \cdot \dot{s}(t)$
- $\ddot{\tilde{q}}(t) = \ddot{q}(s(t)) \cdot \dot{s}^2(t) + \dot{q}(s(t))\ddot{s}(t)$

4.25.1 Scaling the Dynamic Equation

The standard dynamic equation is:

$$\tau(t) = B(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + g(q(t)) \tag{4.140}$$

Substituting the scaled expressions into the dynamic model, we obtain the scaled torque $\tilde{\tau}(t)$:

$$\tilde{\tau}(t) = B(q(s))[\ddot{q}(s)\dot{s}^2 + \dot{q}(s)\ddot{s}] + C(q(s), \dot{q}(s)\dot{s})\dot{q}(s)\dot{s} + g(q(s)) \tag{4.141}$$

Using the property that the Coriolis/centrifugal terms are quadratic in the velocities ($C(q, \dot{q}\dot{s}) = \dot{s}C(q, \dot{q})$), we can simplify the expression:

$$\tilde{\tau}(t) = \dot{s}^2[B(q)\ddot{q} + C(q, \dot{q})\dot{q}] + \ddot{s}B(q)\dot{q} + g(q) \tag{4.142}$$

Defining $m(s) = B(q)\ddot{q} + C(q, \dot{q})\dot{q}$, the equation becomes:

$$\tilde{\tau}(t) = \dot{s}^2 m(s) + \ddot{s}B(q)\dot{q} + g(q) \tag{4.143}$$

To maintain the path while respecting torque limits, we typically set $\ddot{s} = 0$ (constant velocity scaling), which leads to:

$$\tilde{\tau}_i(t) = \dot{s}^2[\tau_i(s) - g_i(q(s))] + g_i(q(s)) \tag{4.144}$$

From this, we can find the scaling factor $\dot{s}$ for the joint i that has exceeded the limit $\tau_{max,i}$.

4.26 Dynamics in Operational Space

This section describes the relationship between the generalized forces acting on the manipulator and the minimal variables used to describe position and orientation in the operational space.

4.26.1 Operational Space Equations

The dynamic model in joint space is:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau - J^T(q)h_e = J^T\gamma \tag{4.145}$$

To project this into the operational space, we use the relationship between joint acceleration and operational acceleration $\ddot{x}_e$:

$$\ddot{x}_e = J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} \tag{4.146}$$

From the joint space dynamics, we solve for $\ddot{q}$:

$$\ddot{q} = B^{-1}(q)[\tau - C\dot{q} - g - J^T h_e] \tag{4.147}$$

Substituting $\ddot{q}$ into the operational acceleration equation:

$$\ddot{x}_e = JB^{-1}\tau - JB^{-1}(C\dot{q} + g) - JB^{-1}J^T h_e + \dot{J}\dot{q} \tag{4.148}$$

4.26.2 Operational Space Inertia and Forces

We seek an expression in the form:

$$B_a(q)\ddot{x}_e + C_a(q,\dot{q})\dot{x}_e + g_a(q) = F_a - h_e \tag{4.149}$$

By manipulating the terms and using the Analytic Jacobian J_A (if we are in the operational space, we must use J_A instead of J), we define the operational space inertia matrix:

$$B_a = (J_A B^{-1} J_A^T)^{-1} \tag{4.150}$$

This matrix is invertible if and only if J_A has full rank (i.e., the robot is not in a singular configuration).

The complete projected equation becomes:

$$B_a\ddot{x}_e + B_a(J_A B^{-1}C - \dot{J}_A)\dot{q} + B_a J_A B^{-1} g = F_a - h_e \tag{4.151}$$

4.27 Direct and Inverse Dynamics in Operational Space

The relationship between forces and motion can also be expressed directly in the operational space, following a logic similar to the joint space.

4.27.1 Direct Dynamics

In the operational space, the goal is to determine the acceleration of the end-effector $\ddot{x}_e$ given the operational forces F_a and the current state $x, \dot{x}$. The solution can be obtained through two steps:

1. Solve the direct dynamics in the joint space to find $\ddot{q}$.
2. Apply direct kinematics to find $\ddot{x}_e$ using the relation $\ddot{x}_e = J\ddot{q} + \dot{J}\dot{q}$.

4.27.2 Inverse Dynamics

Given a desired trajectory in the operational space $(x, \dot{x}, \ddot{x})$, we calculate the forces F_a required.

$$F_a - h_e = B_a(q)\ddot{x}_e + C_a(q, \dot{q})\dot{x}_e + g_a(q) \tag{4.152}$$

Alternatively, this can be solved by:

1. Performing inverse kinematics to find the joint state.
2. Solving the inverse dynamics at the joint level.

4.27.3 Relationship between Joint and Operational Torques

The relationship between the operational space force F_a and the joint torques τ is expressed as:

$$\tau = J^T(q)F_a + (I - J^T J^{T\#})\tau_0 \tag{4.153}$$

where $J^{T\#}$ is a right pseudo-inverse of J^T. The second term represents torques that do not contribute to the end-effector forces (null space), which is relevant for redundant manipulators.

4.28 Dynamic Manipulability Ellipsoid

The dynamic manipulability ellipsoid represents the ability of the robot to accelerate the end-effector in different directions of the operational space, considering the joint torque limits.

Assuming the joint torques are normalized such that $\|\tau\| \leq 1$, and considering a static condition ($\dot{q} = 0$) and neglecting gravity ($g = 0$):

$$\tau = B(q)\ddot{q} \tag{4.154}$$

Given the relation $\ddot{x}_e = J\ddot{q}$ (for $\dot{q} = 0$), we have $\ddot{q} = J^{\#}\ddot{x}_e$. Substituting this into the torque equation:

$$\tau = BJ^{\#}\ddot{x}_e \tag{4.155}$$

The unit sphere in the torque space $\tau^T\tau \leq 1$ maps to an ellipsoid in the acceleration space:

$$\ddot{x}_e^T(BJ^{\#})^T(BJ^{\#})\ddot{x}_e \leq 1 \tag{4.156}$$

4.28.1 Inclusion of Gravity

When gravity is included, the ellipsoid undergoes a translation. The mapping becomes:

$$\tau = BJ^{\#}(\ddot{x}_e + JB^{-1}g) \tag{4.157}$$

The vector $-JB^{-1}g$ describes the contribution of gravity to the operational acceleration. As observed in the diagrams, this translation indicates that it is generally easier for the manipulator to accelerate downwards (in the direction of gravity) than upwards, as the motor torques are partially aided or hindered by the gravitational field.

Chapter 5

Actuators and Sensors

5.1 Actuators

The general architecture of a robotic actuation system follows a specific power flow. The energy is managed and transformed through several stages:

1. **Power Supply**: Provides the primary power P_R.
2. **Power Amplifier**: Modulates the portion of power (P_a) using a control power signal P_c to generate the input power P_e for the motor.
3. **Servomotor**: Converts input power (typically electric, hydraulic, or pneumatic) into mechanical power P_m.
4. **Transmission**: Adapts the mechanical power to the load, providing the useful power P_u to the joint.

The power variables involved in the system are defined as follows:

- P_R: Primary power.
- P_c: Control power (modulates the portion of P_a).
- P_e: Input power to the motor.
- P_m: Mechanical power.
- P_u: Useful power.
- P_{diss}: Dissipated power (losses in the system).

5.2 Transmission

The transmission system is used to adapt the high-velocity, low-torque output of the motor to the low-velocity, high-torque requirements of the robotic joint. The mechanical power is defined by the product of torque and angular velocity:

$$P_m = \tau\omega \tag{5.1}$$

Different types of transmissions are employed based on the mechanical requirements:

1. **Gears**: Used to change the rotation axis and/or translate the application point of the force.

2. **Lead Screws**: Used to convert rotational motion into translational motion.

3. **Belts**: Allow the motor to be located far from the joint axis. They are suitable for high speed and low forces, but they are prone to deformation (elasticity).

4. **Chains**: Similar to belts but designed for low speed and high forces.

5. **Direct Drive**: A rigid transmission where the motor is connected directly to the load, meaning the reduction ratio $K_r = 1$.

5.3 Servomotors

Servomotors can be classified by the type of energy they utilize:

1. **Electric Motors**: The most common type; energy is easy to find and distribute.

2. **Pneumatic Motors**: Energy is supplied by a compressor and transformed via pistons or turbines.

3. **Hydraulic Motors**: Energy is stored in a tank; they provide large torques at low speeds.

5.4 Design Requirements

To achieve high performance in robotics, the following characteristics are desired in an actuation system:

- Low inertia and high power-to-weight ratio.
- Possibility to handle overloads.
- High acceleration capabilities.
- Large range of velocities.
- High precision in positioning.
- Low torque ripple, especially at low speeds.

5.5 Power Amplifier

The power amplifier is responsible for the **modulation of the power flow** from the alimentation (power supply) to the actuator. This is often achieved through components like transformers or electronic switching systems to ensure the motor receives the exact amount of energy dictated by the control signal.

5.6 Electric Drives

The analysis of an electric drive requires considering both mechanical and electric equilibrium.

5.6.1 Mechanical Equilibrium

The driving torque C_m produced by the motor is proportional to the armature current I_a through the torque constant K_t:

$$C_m = K_t I_a \tag{5.2}$$

The mechanical balance of the system, considering the motor's shaft, is expressed as:

$$C_m = (sI_m + f_m)\Omega + C_l \tag{5.3}$$

Where:

- I_m: Moment of inertia of the motor.
- f_m: Viscous friction coefficient.
- Ω: Angular velocity.
- C_l: Load reaction torque (disturbing torque).

5.6.2 Electric Equilibrium

The electrical behavior of the armature circuit is governed by the following equations:

$$V_j = K_v \Omega \tag{5.4}$$

$$V_a = (R_a + sL_a)I_a + V_j \tag{5.5}$$

Where:

- V_j: Back electromotive force (EMF).
- K_v: Voltage constant.
- V_a: Armature voltage.
- R_a: Armature resistance.
- L_a: Armature inductance.
- I_a: Armature current.

5.7 System Representation and Control

5.7.1 Power Amplifier Modeling

The power amplifier provides the voltage V_a based on a control voltage V_c. Its transfer function is:

$$\frac{V_a}{V_c} = \frac{G_v}{1 + sT_v} \tag{5.6}$$

In many practical robotic applications, the time constant T_v is small enough to be neglected ($T_v \approx 0$), leading to $V_a \approx G_v V_c$.

5.7.2 Simplified Model (Velocity Controlled)

Assuming the armature inductance L_a is very small and neglecting the viscous friction compared to the "electrical friction" ($f_m \ll K_v K_t / R_a$), we can analyze the system under velocity control.

Using the superposition of effects to analyze the influence of the control signal V_c' and the load torque C_l separately:

1. **Effect of Load Torque** ($V_c' = 0$): The velocity variation due to the load Ω^1 is:

$$\Omega^1 = -\frac{C_l}{sI_m + f_m + \frac{K_v K_t}{R_a}} \tag{5.7}$$

Which can be rewritten to highlight the steady-state gain:

$$\Omega^1 \approx -\frac{\frac{R_a}{K_v K_t}}{1 + s\frac{R_a I_m}{K_v K_t}} C_l \tag{5.8}$$

2. **Total Velocity Equation**: Combining the effects of the input voltage and the load torque, the angular velocity Ω is:

$$\Omega = \frac{\frac{1}{K_v}}{1 + s\frac{R_a I_m}{K_v K_t}} G_v V_c' - \frac{\frac{R_a}{K_v K_t}}{1 + s\frac{R_a I_m}{K_v K_t}} C_l \tag{5.9}$$

For further analysis of the nominal performance, we often assume $C_l = 0$.

5.8 Control Strategies: Velocity vs Torque Control

Based on the previous derivations, we can distinguish two main control configurations for the electric drive:

5.8.1 Velocity Control (VCG)

By choosing the gain G_v such that $V_a = G_v V_c$, the system behaves as a velocity-controlled drive.

- The control signal V_c directly influences the angular velocity Ω.
- This configuration is particularly suitable for **independent joint control** schemes.

5.8.2 Torque Control (TCG)

If we choose a very high gain G_v, the system can be configured to control the current I_a, and consequently the torque C_m, since $C_m = K_t I_a$.

- In this mode, the output torque does not depend on the velocity Ω.
- This is often referred to as an "Amperomotor" configuration where I_a is proportional to the control signal V_c'.
- Torque control is preferred for **centralized control schemes** where dynamic interaction between joints must be managed.

5.9 Current Feedback and Protection

When using an "Amperomotor" (torque control), the control signal V_c' is typically proportional to the error between the desired and actual joint angle.

- A significant risk arises: if the error increases, V_c' increases, which causes the current I_a to rise, potentially burning the motor.
- To prevent this, a **Dead-zone nonlinearity** or a saturation block is implemented as a protection mechanism.

5.9.1 Nonlinearity Characteristics

The protection behaves as follows:

- **Flat Range**: If the command is within the flat (dead) range, the loop is effectively open or inactive.
- **Saturation/Limitation**: If the command is too large, it acts as a limitation to keep the current within safe operating bounds.

5.10 Hydraulic Drives

Hydraulic drives utilize fluid transport for power transmission. The flow rate Q is supplied by a distributor.

5.10.1 Flow and Pressure Equations

We consider the relationship between flow (Q), pressure (P), and the displacement of the distributor (x):

$$Q = K_q x - K_c P \tag{5.10}$$

Where:

- K_q: Flow gain.
- K_c: Flow-pressure coefficient (representing leakage and compressibility).

In rotary motors, the displacement volume is constant, whereas for pistons, the volume changes during operation. The useful flow Q_u is related to the velocity:

$$Q_u = K_q \Omega_m \tag{5.11}$$

5.11 Hydraulic Drives: Continued

Continuing the analysis of hydraulic systems, the pressure feedback is an intrinsic part of the physical structure, unlike electric drives where current feedback is optional. The mechanical equilibrium for a hydraulic motor is given by:

$$C_m = (sI_m + f_m)\Omega_m + C_l \tag{5.12}$$

Where the torque produced is related to the pressure P by the constant k_t:

$$C_m = k_t P \tag{5.13}$$

In terms of the flow rate Q, considering the distributor displacement x:

$$P = k_x x - k_v Q \tag{5.14}$$

This confirms that in hydraulic drives, the "pressure feedback" (equivalent to the back-EMF in electric motors) is built into the physics of the distributor and the fluid dynamics.

5.12 Transmission and Load Dynamics

When considering the motor coupled to a load through a transmission, we must account for the inertia and friction of both elements.

5.12.1 System Equations

Let I_m and f_m be the motor's inertia and friction, and I_w, f_w those of the load (joint). The reduction ratio is defined as $k_r = \frac{\theta_m}{\theta_l}$. The total driving torque C_m required to move the system is:

$$C_m = I_m \ddot{\theta}_m + f_m \dot{\theta}_m + \frac{1}{k_r}(I_w \ddot{\theta}_l + f_w \dot{\theta}_l + C_l) \tag{5.15}$$

Substituting $\theta_l = \frac{\theta_m}{k_r}$, we obtain the equivalent dynamics referred to the motor shaft:

$$C_m = \left(I_m + \frac{I_w}{k_r^2}\right) \ddot{\theta}_m + \left(f_m + \frac{f_w}{k_r^2}\right) \dot{\theta}_m + \frac{C_l}{k_r} \tag{5.16}$$

5.12.2 Equivalent Parameters

We can define equivalent inertia I_{eq} and equivalent friction f_{eq}:

- $I_{eq} = I_m + \frac{I_w}{k_r^2}$
- $f_{eq} = f_m + \frac{f_w}{k_r^2}$

The transmission reduces the impact of the load inertia and friction by a factor of k_r^2, making the motor's own inertia dominant when k_r is high.

5.13 Position Control and Nested Loops

To achieve high performance in robotic positioning, control schemes are often structured with nested loops.

5.13.1 VCG Position Control

In a Velocity Controlled (VCG) drive, the position control is typically implemented using a P.I. (Proportional-Integral) controller.

- The integral action is crucial for disturbance rejection (such as rejecting the load torque C_l).

5.13.2 Secondary Loop for Performance

To improve the transitory response, tracking capability, and bandwidth while reducing oscillations, a second loop (typically a velocity loop inside the position loop) can be closed.

- **Wide Bandwidth**: Essential for fast tracking.
- **Oscillation Damping**: Achieved through the derivative action or the secondary velocity feedback.

5.14 Sensors

Sensors are fundamental components in robotics to perceive both the internal state of the robot and the surrounding environment. They can be classified into two main categories:

5.14.1 Proprioceptive Sensors

These sensors measure the internal state of the robot (Robot State). Key measures include:

- Joint positions
- Joint velocities
- Joint torques

5.14.2 Exteroceptive Sensors

These sensors are used to acquire information about the external environment:

- Force
- Proximity
- Vision
- Other measures

5.15 Proprioceptive Sensors: Angular Position

To measure the angular position of joints, encoders are commonly used.

5.15.1 Absolute Encoder

An absolute encoder provides a unique digital code for each angular position. The physical structure consists of a disk with transparencies, a light beam, and a photo diode to detect the pattern.

- **Structure:** Usually, we have 16 tracks, which determines the resolution.

- **Gray Code:** To avoid interferences when moving from one position to the next, Gray Code is used. In this encoding, only one bit changes at a time.

The following table shows an example of the mapping between position and Gray Code:

Position (#)	Code
0	0000
1	0001
2	0011
3	0010
4	0110
...	...
15	1000

5.15.2 Incremental Encoders

Unlike absolute encoders, incremental encoders detect changes in position relative to a home position.

- **Structure:** They typically use 2 tracks.
- **Working Principle:** The sensor detects a pulse when the disk moves from matte to transparent.
- **Direction:** Since we have 2 tracks (often in quadrature), we can determine the direction of rotation.
- **Data Reconstruction:** By counting the pulses, it is possible to reconstruct both velocities and positions.

5.16 Resolver

A resolver is another type of rotary electrical transformer used for measuring degrees of rotation.

- The tension induced on the stator depends on the angle θ.
- The system utilizes a Sinusoidal signal $V \sin(\omega t)$.
- Components in the processing loop include:
  - Sine and Cosine multipliers.
  - Synchronous Demodulator (Synch. Demod.).
  - Correction Network and Integrator.
  - Voltage Controlled Oscillator (VCO) and a Counter.
- The output provides information about the velocity (VEL) and position.

5.17 Velocity Transducers

These sensors are specifically designed to measure the speed of the robot's joints.

5.17.1 Dynamo

Uses a permanent magnet. The rotor spins and generates a voltage directly proportional to the velocity.

5.17.2 AC Tachymeter

- Consists of 2 stator coils and a low inertia rotor.
- The rotation of the rotor induces a current on the coils that is proportional to the velocity.

5.18 Force Sensors

Force measurement is often achieved through the use of strain gauges.

5.18.1 Strain Gauge and Wheatstone Bridge

A strain gauge is a deformable element whose electrical resistance changes when subjected to mechanical stress. To accurately measure these small changes in resistance, a Wheatstone Bridge circuit is utilized.

[Image of Wheatstone bridge circuit for strain gauge]

- **Bridge Configuration**: The bridge consists of four resistors (R_1, R_2, R_3, R_5).
- **Output Voltage**: The output voltage V_o is derived from the input voltage V_i using the following formula:

$$V_o = \left(\frac{R_2}{R_1 + R_2} - \frac{R_5}{R_3 + R_5} \right) V_i$$

- **Accuracy Improvement**: To improve accuracy, another gauge can be used in the circuit. In this configuration, one gauge will undergo compression while the other undergoes extension.

5.19 Torque and Wrist Sensors

These sensors are specialized for measuring rotational forces at the joints or the robot's end-effector.

5.19.1 Shaft Torque Sensor

This sensor consists of strain gauges mounted on a deformable apparatus placed between the motor and the joint.

- **Mechanical Properties**: The apparatus must be designed with low torsional stiffness but high bending stiffness.
- **Measurement**: The strain gauges measure the deformation torque.
- **Notice**: It is important to note that this measures the torque on the joint after accounting for inertial and friction torques, rather than the actual driving torque C_m.

5.19.2 Wrist Force Sensor

By attaching strain sensors to the robot's wrist, it is possible to derive the full state of forces and torques acting on the end-effector.

- **Matrix Formulation**: The relationship between the sensor readings $(w_1, \ldots, w_8)$ and the force/torque components $(l_x, l_y, l_z, M_x, M_y, M_z)$ is expressed through a decoupling matrix.
- **Structural Decoupling**: The design aims for a single force component to induce the least number of deformations possible, ensuring a good structural decoupling of the force components.

5.20 Range Sensors

Range sensors are used to measure the distance between the robot and objects in the environment.

5.20.1 Sonar (Sound Navigation and Ranging)

Sonar sensors use acoustic pulses and their echoes to determine the range to an object.

- **Frequency Range**: Typically operates between 20 kHz and 200 kHz.
- **Beamwidth**: Usually around 15 degrees.
- **Distance Calculation**: The distance d_0 is calculated based on the time of flight t_v and the speed of sound c_s:
$$d_0 = \frac{c_s t_v}{2}$$
- **Speed of Sound**: The velocity c_s is temperature-dependent and can be estimated as:
$$c_s = 20.05\sqrt{T + 273.16} \text{ m/s}$$

5.21 Transducer Technologies and Sonar Limitations

The physical implementation of sonar sensors relies on specific transducer types:

- **Piezoelectric Transducer**: Vibrates under the action of an electric field.
- **Capacitive Transducer**: Consists of a capacitor whose armature deforms due to sound pressure, resulting in a change in voltage.

Both transducers can function as both receivers and transmitters.

5.22 Lasers

Laser sensors offer several advantages for robotic perception:

- **Infrared**: Non-intrusive.
- **Narrow Beams**: High resolution.
- **Single Frequency**: Do not disperse.

5.22.1 Time-of-Flight Laser Sensor

This sensor measures the time interval for a pulse to return.

- **Minimum Distance**: A limitation is the minimum measurable time interval, which determines the minimum measurable distance.
- **Ambiguity Problem**: The sensor emits pulses and accepts returning signals within a specific time window.
- **Signal Interpretation**: If an old signal returns after the window, it can be misinterpreted as a new signal.
- **Maximum Distance**: The maximum measurable distance without ambiguity is limited by this timing.

5.22.2 Triangulation Laser Sensor

This method uses geometry to compute distance:

- The laser hits an object and the light diffuses once reflected.
- The reflected light hits a CCD sensor.
- The CCD determines the exact point where the light hit; thanks to this information, the distance can be computed.

5.23 Vision Sensors

Vision systems generally use two types of solid-state technologies.

5.23.1 CCD (Charged Coupled Device)

- Consists of an array of photosites.
- When a photon hits a photosite, free electrons are created.
- As long as the shutter is open, the electrons keep accumulating.
- Subsequently, the electrons are shifted to be amplified in order to create the image.

5.23.2 CMOS

- Also an array, but each pixel has its own amplifier, so it does not need shifting.
- It does not have electrons accumulate in the same way because each pixel generates a current depending on the quantity of photons.

5.23.3 Camera

A camera system is composed of several functional blocks that process incoming light into a video signal.

Camera Structure

The internal components include:

- **Optics**: Shutter, lens, and a sensor (CCD or CMOS).
- **Timing and Sync**: Control the acquisition process.
- **Analog Electronics**: Process the raw sensor data before the video output.

5.23.4 Camera Model and Coordinate Mapping

To relate a point in the 3D world to a 2D image plane, we use a geometric model based on the lens center.

Perspective Projection

Let $\tilde{P}$ be a point in the base frame. Its coordinates in the camera frame $\tilde{P}^c$ are obtained via a transformation matrix:

$$\tilde{P}^c = T_b^c \tilde{P}$$

Where:

- $P^c = [P_x^c, P_y^c, P_z^c]^T$ represents the coordinates in the camera reference frame.
- The optical axis is aligned with Z_c, and f is the focal distance from the lens center to the image plane.

The coordinates (X, Y) on the image plane are derived through perspective projection:

$$X = -f\frac{P_x^c}{P_z^c}, \quad Y = -f\frac{P_y^c}{P_z^c}$$

Transformation to Pixel Coordinates

To express these coordinates in terms of pixels (X_i, Y_i), we apply scaling and offset factors (intrinsic parameters):

- **Scaling factors**: a_x and a_y represent the transformation from physical units to pixels.
- **Principal Point**: (X_0, Y_0) is the coordinate of the optical center on the pixel grid.

The final pixel coordinates are calculated as:

$$X_i = a_x X + X_0$$

$$Y_i = a_y Y + Y_0$$

Chapter 6

Control Architecture

6.1 Control Architecture

The control architecture is considered the brain of the robot. It is organised in multiple levels, where each level consists of three fundamental modules:

1. **Sensor Module**: Reads data from sensors in order to know the state of the robot and the environment.
2. **Model Module**: Contains the model and the knowledge of the robot and the environment.
3. **Decision Module**: It translates high-level tasks into simple actions and manages temporal sequences.

6.2 Control Levels

The hierarchy of robot control is divided into functional levels that increase in specificity:

1. **Task Level**: The user tells the robot what to do; the robot translates the task into specific actions.
2. **Action Level**: The actions are translated into a series of configurations. This level also decides the workspace.
3. **Primitives Level**: It computes the feasible trajectory and defines the control strategy.
4. **Servo Level**: It translates the trajectory into signals for the joint motion.

6.3 Hardware Components

The physical implementation of the control architecture relies on dedicated boards:

- **System Board**: Contains the microprocessor, math coprocessor, RAM, ROM, I/O, and counters.

- **Kinematic Board**: It calculates direct and inverse kinematics and handles singularities.
- **Dynamic Board**: Dedicated to dynamic model computations.
- **Servo Board**: It handles the servo motors, Digital-to-Analog Converters (DAC), amplifiers, and sensors.
- **Force Board**: It reads and processes data from force sensors.
- **Vision Board**: Analyzes images from the camera and extracts relevant information.

6.4 Control Motion in Joint Space

To achieve precise motion, we utilize Joint Space Control. Given a desired trajectory x_d, the inverse kinematics (Inv. Kin.) provides the reference for the controller.

6.4.1 Dynamic Model

The dynamic behavior of the manipulator is described by the following equation:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q) = \tau \tag{6.1}$$

The objective of the control design is to determine τ so that the actual joint positions track the desired ones: $q(t) = q_d(t)$.

6.4.2 Transmissions and Drives

The relationship between joint variables (q, τ) and motor variables (q_m, τ_m) is governed by the transmission ratio K_r:

$$K_r q = q_m \implies \tau_m = K_r^{-1}\tau \tag{6.2}$$

The electrical drive system (neglecting inductance) is modeled as:

$$v_a = R_a i_a + K_v \dot{q}_m \tag{6.3}$$

Where the armature voltage v_a is related to the control signal V_c by the amplifier gain G_v:

$$v_a = G_v V_c \tag{6.4}$$

6.4.3 Control Signal Derivation

Combining the above, the torque generated is:

$$\tau = K_r K_t i_a = K_r K_t R_a^{-1} v_a - K_r K_t R_a^{-1} K_v \dot{q}_m \tag{6.5}$$

Substituting the amplifier gain:

$$\tau = K_r K_t R_a^{-1} G_v V_c - K_r K_t R_a^{-1} K_v \dot{q}_m \tag{6.6}$$

6.5 Velocity Controlled Systems

The dynamic equation of the manipulator can be rewritten considering the system is velocity controlled:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F\dot{q} + g(q) = \mu \tag{6.7}$$

Where the total friction term F and the control input μ are defined as:

- $F = F_v + K_r K_t R_a^{-1} K_v K_r$
- $\mu = K_r K_t R_a^{-1} G_v V_c$

6.5.1 Design Assumptions for Voltage Control

To simplify the control design, we impose the following assumptions:

1. K_r elements are much greater than 1 ($K_r \gg 1$).
2. R_a elements are very small.
3. The terms are not too large.

Under these conditions, we can assume that $G_v V_c \approx K_v K_r \dot{q}$. This leads to the definition of the control voltage:

$$V_c = G_v^{-1} K_v K_r \dot{q} \tag{6.8}$$

In this configuration, the system is diagonal, meaning the joint velocity depends only on the applied voltage.

6.6 Control Architectures: Decentralised vs Centralised

6.6.1 Decentralised Control

In a decentralised scheme, each joint is controlled independently. This is typical in commercial robots like Kuka. However, we must accurately know parameters such as K_t, R_a, and K_v. Since these are physical parameters, they can vary over time or due to operating conditions.

6.6.2 Centralised Control

To overcome the limitations of parameter variation, we can use a current feedback loop:

$$i_a = G_c V_c \tag{6.9}$$

In this case, the torque is directly proportional to the control effort:

$$\tau = \mu = K_r K_t i_a \tag{6.10}$$

6.7 Current and Torque Control Design

6.7.1 Decentralised Approach and Disturbance Analysis

We design V_c to be proportional to τ. Starting from the joint level dynamic equation:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v(q)\dot{q} + g(q) = \tau \tag{6.11}$$

We transform the equation in terms of motor quantities using the relation $\dot{q} \to K_r^{-1}\dot{q}_m$:

$$K_r^{-1}BK_r^{-1}\ddot{q}_m + K_r^{-1}CK_r^{-1}\dot{q}_m + K_r^{-1}F_vK_r^{-1}\dot{q}_m + K_r^{-1}g = K_r^{-1}\tau = \tau_m \tag{6.12}$$

By splitting the inertia matrix into a constant diagonal part and a configuration-dependent part $B(q) = \bar{B} + \Delta B(q)$, we can rewrite the system as linear and decoupled:

$$K_r^{-1}\bar{B}K_r^{-1}\ddot{q}_m + F_m\dot{q}_m + d = \tau_m \tag{6.13}$$

Where d represents the disturbance term:

$$d = K_r^{-1}\Delta B(q)K_r^{-1}\ddot{q}_m + K_r^{-1}C(q,\dot{q})K_r^{-1}\dot{q}_m + K_r^{-1}g(q) \tag{6.14}$$

The goal is to design the controller so that it effectively rejects this disturbance d. It is important to notice that the term $1/K_r^2$ is very small, which helps in reducing the coupling effects.

6.8 Independent Joint Control

In this approach, the manipulator is treated as n independent systems, where the coupling effects and gravity are considered as a disturbance d. The objective is to reduce the effect of d on the motor position q_m.

To achieve this, the control system requires:

- A large amplifier gain before the disturbance to suppress its influence.
- An integral action to remove the steady-state effect of gravity.

6.8.1 Controller Structure

The chosen controller $C(s)$ is typically a PI (Proportional-Integral) or PID. A purely integral controller would be too slow. The transfer function is:

$$C(s) = K_p\frac{1 + sT_i}{sT_i} \tag{6.15}$$

6.8.2 System Transfer Function

We define the forward part of the transfer function. Considering the motor dynamics $M(s)$, we want to relate the control voltage to the output.

- $M(s) = \frac{1}{Is+F}$ (where I is the inertia and F is friction).
- The electrical relation is $V = iR_a$ (neglecting inductance).

The open-loop transfer function $C(s)M(s)$ is:

$$C(s)M(s) = \frac{K_p(1 + sT_i)}{sT_i} \cdot \frac{A}{1 + sT_m} \tag{6.16}$$

Where we need to design K_p and T_i to ensure stability and performance.

6.9 Root Locus Analysis

To evaluate the stability and transient response, we analyze the Root Locus of the system. We consider the poles and zeros of the transfer function:

- Poles at $s = 0$ (from the integrator) and $s = -1/T_m$ (from the motor).
- Zero at $s = -1/T_i$.

6.9.1 Case 1: $T_i < T_m$

If T_i is smaller than T_m, the zero is further to the left than the motor pole. This configuration is generally avoided because the branches of the root locus can move toward the right half-plane, degrading stability.

6.9.2 Case 2: $T_i > T_m$

This is the preferred configuration. By placing the zero closer to the origin than the motor pole, the root locus stays in the left half-plane, ensuring better damping and stability margins.

6.10 Closed-Loop Transfer Function

The final closed-loop transfer function for the motor position, including the feedback gains, can be expressed as:

$$\frac{q_m}{q_{m,d}} = \frac{sT_i + 1}{s^2T_i(\dots) + sT_i(\dots) + 1} \tag{6.17}$$

In more compact terms, it can be approximated or expressed in the standard second-order form:

$$W(s) = \frac{1 + sT_i}{1 + sT_i + s^2\frac{T_i}{K}} \tag{6.18}$$