NYU TANDON SCHOOL OF ENGINEERING

# Autonomous Mobile Robots

## Lecture 4: Nonlinear Control Methods

Dr. Aliasghar Arab

NYU Tandon School of Engineering

Fall 2025

PART 01

# Lecture 4

## Nonlinear Control Methods

# Topics

1. Introduction & Physical Constraints
2. Vehicle Kinematics & Equations of Motion
3. Control & Perception Fundamentals
4. **Nonlinear Control Methods**
5. Advanced MPC + Constraints
6. Motion Planning Algorithms
7. Learning-Based Planning & Control
8. Industry Standards & Safety

- We are Modelling a **nonlinear, time-invariant** system:

### System Dynamics

$$\dot{\boldsymbol{X}} = f(\boldsymbol{X}, u)$$

Where:

$$\text{Robot state} = \boldsymbol{X} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

$$u = \text{Control Input } [v, \omega]^T$$

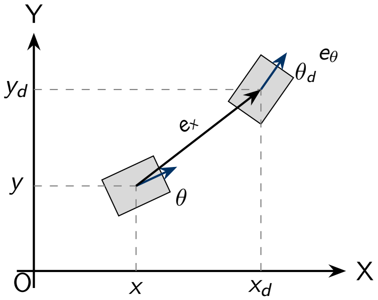



**Fig. 2** Viewpoint of error coordinate

- Define the **error vector** as:

$$\boldsymbol{e} = \boldsymbol{X}_d - \boldsymbol{X}$$

Expanding:

$$\boldsymbol{e} = \begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix} - \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

**Component-wise errors:**

$e_x = x_d - x$ (Position error along $x$-axis)

$e_y = y_d - y$ (Position error along $y$-axis)

$e_\theta = \theta_d - \theta$ (Heading error)

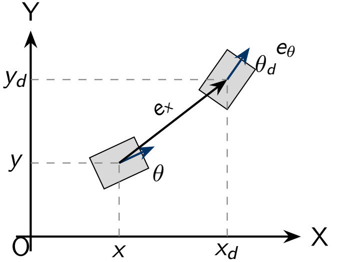
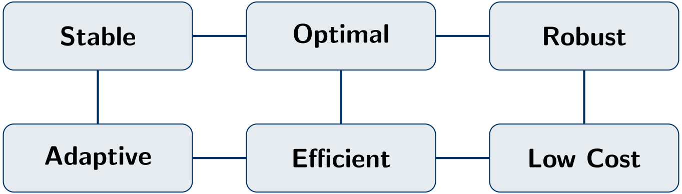



**Fig. 2** Viewpoint of error coordinate




## Key Considerations

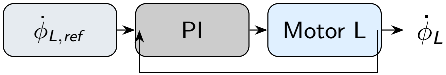
- **Stability**: Converge to desired state without oscillations
- **Robustness**: Handle model uncertainties and disturbances
- **Optimality**: Minimize energy, time, or tracking error

**Pros:**

- Multiple PI controllers (straightforward loop design)
- Works well with ROS kinematic frameworks
- Scalable (can extend to more wheels)
- Simple at the kinematic level

**Cons:**

- Lacks deeper physical/dynamic understanding
- Doesn't capture full robot dynamics
- More sensors required (encoders per wheel)
- Adds dynamic complexity at higher levels



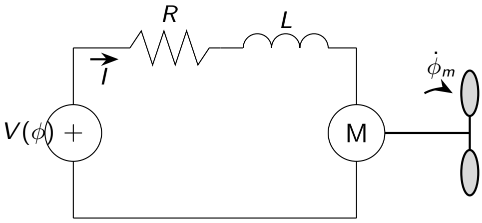

(Similar structure for right wheel)

## DC Motor Electrical Equation

$$L\dot{I} + RI = V - K_b\dot{\phi}_m$$

**Parameters:**

- $V$ = applied motor voltage
- $I$ = current in motor coil
- $R$ = coil resistance
- $L$ = coil inductance
- $K_b\dot{\phi}_m$ = back EMF opposing voltage




DC Motor Circuit Model

## Motor Torque Equation

$$\boxed{\tau = k_m I}$$

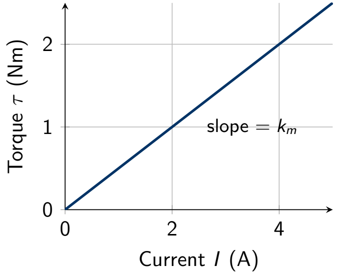
**Parameters:**

- $k_m$ = motor torque constant
- $I$ = motor current
- $\tau$ = generated torque

**Key relationship:**

$$\tau \propto I$$

Torque is directly proportional to current



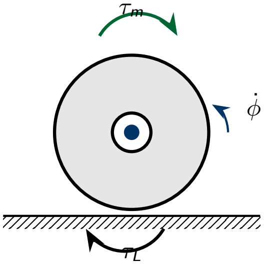

Linear torque-current relationship

## Rotational Equation of Motion

$$J_w \ddot{\phi} + B_w \dot{\phi} = \tau_m - \tau_L$$

**Parameters:**

- $J_w$ = wheel inertia
- $B_w \dot{\phi}$ = viscous damping (friction at bearings)
- $\tau_m$ = torque from motor
- $\tau_L$ = load torque (robot weight, ground resistance)




Wheel force diagram

Combining electrical and mechanical dynamics yields a **first-order transfer function**:

$$\boxed{\frac{\dot{\phi}(s)}{V(s)} = \frac{r_g \cdot \frac{K_m}{R}}{J_w s + \left(B_w + \frac{K_b K_m}{R}\right)}}$$

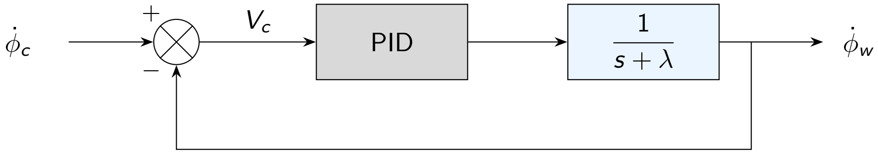
**Parameters:**

- $r_g$ = gear ratio
- $R$ = electrical resistance
- $J_w$ = wheel inertia

**Motor constants:**

- $B_w$ = mechanical damping
- $K_m$ = torque constant
- $K_b$ = back-EMF constant

Note: Often $K_m \approx K_b$ for DC motors




## Control Loop Components

- $\dot{\phi}_c - \dot{\phi}_w =$ Error
- $\dot{\phi}_c =$ Desired wheel speed
- $\dfrac{1}{s+\lambda} =$ Approximated wheel + motor dynamics (first-order response)

**Robot Pose:**

$$\boldsymbol{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

**Wheel angular velocities:**

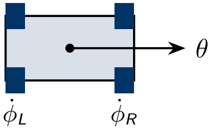
$$\dot{\phi}_w = \begin{bmatrix} \dot{\phi}_L \\ \dot{\phi}_R \end{bmatrix}$$

**Kinematic relation:**

$$\boxed{\dot{\boldsymbol{q}} = \mathbf{J}(\boldsymbol{q})\, \dot{\phi}_w}$$

$\mathbf{J}(\boldsymbol{q})$ = Jacobian mapping wheel speeds to body velocities

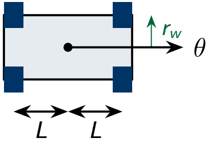
Robot pose $\boldsymbol{q}$




Differential drive robot

**Kinematic Jacobian:**

$$\mathbf{J}(\boldsymbol{q}) = \frac{r_w}{2}\begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix}$$




**Parameters:**

- $r_w$ = wheel radius
- $L$ = half wheelbase (center to wheel)

Robot **cannot move sideways** (no velocity in body-$y$ direction):

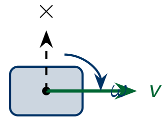
$$\boxed{\dot{x}\sin\theta - \dot{y}\cos\theta = 0}$$

Equivalently: $\dot{x}\sin\theta = \dot{y}\cos\theta$

This **reduces motion** to two DOFs:

$$\boldsymbol{V} = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

- $v$ = Linear velocity
- $\omega$ = Angular velocity




No sideways motion

## General Dynamic Equation of Mobile Robot

$$\boxed{\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\Delta} = \mathbf{B}(\boldsymbol{q})\boldsymbol{\tau}}$$

- $\mathbf{M}(\boldsymbol{q})$ = inertia (mass) matrix
- $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ = Coriolis/centrifugal forces
- $\boldsymbol{\Delta}$ = disturbances (friction, external forces)
- $\mathbf{B}(\boldsymbol{q})$ = input mapping matrix
- $\boldsymbol{\tau}$ = wheel torques

Euler-Lagrange formulation

**Velocity transformation** using the nonholonomic constraint:

$$\dot{\boldsymbol{q}} = \mathbf{T}_v \boldsymbol{V}$$

$$\ddot{\boldsymbol{q}} = \dot{\mathbf{T}}_v \boldsymbol{V} + \mathbf{T}_v \dot{\boldsymbol{V}}$$

where $\boldsymbol{V} = [v, \omega]^T$

**Transformation matrix:**

$$\mathbf{T}_v(\theta) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}$$

This maps the 2D velocity space $\boldsymbol{V}$ to the 3D configuration space derivative $\dot{\boldsymbol{q}}$.

**Inertia Matrix:**

$$\mathbf{M}(\boldsymbol{q}) = \begin{bmatrix} m & 0 & -md\sin\theta \\ 0 & m & md\cos\theta \\ -md\sin\theta & md\cos\theta & I_{zz} \end{bmatrix}$$

**Coriolis Matrix:**

$$\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{bmatrix} 0 & 0 & -md\dot{\theta}\cos\theta \\ 0 & 0 & -md\dot{\theta}\sin\theta \\ 0 & 0 & 0 \end{bmatrix}$$

$m$ = total mass, $d$ = offset from center to CoM, $I_{zz}$ = moment of inertia

## Skew-Symmetric Property

The matrix $\dot{\mathbf{M}}(\boldsymbol{q}) - 2\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is **skew-symmetric**.

For a skew-symmetric matrix $\mathbf{A}$:

$$\mathbf{A} = -\mathbf{A}^T \qquad \Rightarrow \qquad \boldsymbol{x}^T \mathbf{A} \boldsymbol{x} = 0$$

**Importance:**

- Guarantees **energy conservation** in robot dynamics
- Essential for proving **stability** using Lyapunov methods
- Exploited in passivity-based control design

**Configuration-space dynamics:**

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\Delta} = \mathbf{B}(\boldsymbol{q})\boldsymbol{\tau}$$

**Substitute** transformation $\dot{\boldsymbol{q}} = \mathbf{T}_v \boldsymbol{V}$ and $\ddot{\boldsymbol{q}} = \dot{\mathbf{T}}_v \boldsymbol{V} + \mathbf{T}_v \dot{\boldsymbol{V}}$:

$$\mathbf{M}(\boldsymbol{q})\left(\mathbf{T}_v \dot{\boldsymbol{V}} + \dot{\mathbf{T}}_v \boldsymbol{V}\right) + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\mathbf{T}_v \boldsymbol{V} + \boldsymbol{\Delta} = \mathbf{B}\boldsymbol{\tau}$$

**Velocity-space dynamics:**

$$\boxed{\bar{\mathbf{M}}(\boldsymbol{q})\dot{\boldsymbol{V}} + \bar{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{V} + \bar{\boldsymbol{\Delta}} = \bar{\mathbf{B}}(\boldsymbol{q})\boldsymbol{\tau}}$$

**Transformed matrices:**

$$\bar{\mathbf{M}}(\boldsymbol{q}) = \mathbf{T}_v^T \mathbf{M}(\boldsymbol{q}) \mathbf{T}_v$$

$$\bar{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathbf{T}_v^T \left[ \mathbf{M}(\boldsymbol{q}) \dot{\mathbf{T}}_v + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \mathbf{T}_v \right]$$

$$\bar{\mathbf{B}} = \mathbf{T}_v^T \mathbf{B}(\boldsymbol{q})$$

$$\bar{\boldsymbol{\Delta}} = \mathbf{T}_v^T \boldsymbol{\Delta}$$

These 2×2 matrices are much simpler than the original 3×3 matrices!

**Kinematic control law** for trajectory tracking:

$$\boldsymbol{V}_c = \begin{bmatrix} v_d \cos(e_\theta) + k_x e_x \\ \omega_d + k_y v_d e_y + k_\theta v_d \sin(e_\theta) \end{bmatrix}$$

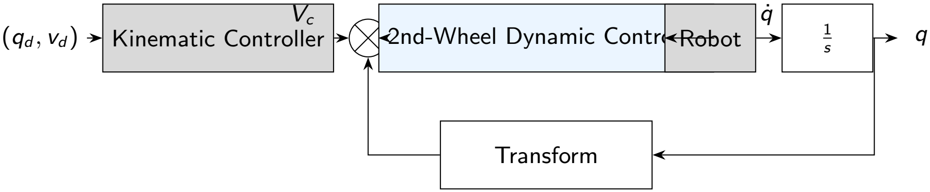
**Inputs:**

- $\boldsymbol{q}_d = [x_d, y_d, \theta_d]^T$ – Desired pose
- $v_d, \omega_d$ – Desired velocities

**Gains:** $k_x, k_y, k_\theta > 0$

**Note**

This controller is for **trajectory tracking**, not set-point regulation.




## Control Architecture

- **Kinematic Controller** $\rightarrow$ computes command velocity $V_c$ from tracking errors
- **Dynamic Controller** $\rightarrow$ converts $V_c$ into wheel torques $\tau$
- **Robot + EOM** $\rightarrow$ updates motion states $(q, \dot{q})$
- **Feedback transform** $\rightarrow$ converts states to error signals

**Robot velocity dynamics:**

$$\bar{\mathbf{M}}(\boldsymbol{q})\dot{\boldsymbol{V}} + \bar{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{V} + \bar{\boldsymbol{\Delta}} = \bar{\mathbf{B}}\boldsymbol{\tau}$$

**Goal:** Design $\boldsymbol{\tau}$ such that robot velocity $\boldsymbol{V}$ tracks desired $\boldsymbol{V}_c(t)$.

### General Nonlinear System

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$$

Feedback linearization transforms this into a **linear** closed-loop system.

**Nonlinear system:** $\dot{x} = f(x) + b(x)u$

**Choose control law:**

$$u = b^{-1}\left[-f(x) + \dot{x}_d + k(x_d - x)\right]$$

**Substituting into the system:**

$$\dot{x} = \dot{x}_d + k(x_d - x)$$

**Error dynamics:** Define $e = x_d - x$

$$\boxed{\dot{e} + ke = 0}$$

Linear, exponentially stable error dynamics with rate $k > 0$.

**Apply feedback linearization** to velocity-space dynamics.

**Velocity-space EOM:**

$$\bar{\mathbf{M}}(\boldsymbol{q})\dot{\boldsymbol{V}} + \bar{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{V} + \bar{\boldsymbol{\Delta}} = \bar{\mathbf{B}}\boldsymbol{\tau} \quad \hookrightarrow (1)$$

**Chosen input torque:**

$$\boldsymbol{\tau} = \bar{\mathbf{B}}^{-1}\left[\bar{\mathbf{M}}(\boldsymbol{q})\dot{\boldsymbol{V}}_c + \bar{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{V} + \tilde{\boldsymbol{\Delta}} + \mathbf{K}(\boldsymbol{V}_c - \boldsymbol{V})\right] \quad \hookrightarrow (2)$$

**Substituting (2) into (1):**

$$\bar{\mathbf{M}}(\boldsymbol{q})(\dot{\boldsymbol{V}}_c - \dot{\boldsymbol{V}}) + \mathbf{K}(\boldsymbol{V}_c - \boldsymbol{V}) = \tilde{\boldsymbol{\Delta}} - \bar{\boldsymbol{\Delta}}$$

$$\bar{\mathbf{M}}(\boldsymbol{q})(\dot{\boldsymbol{V}}_c - \dot{\boldsymbol{V}}) + \mathbf{K}(\boldsymbol{V}_c - \boldsymbol{V}) = \tilde{\boldsymbol{\Delta}} - \bar{\boldsymbol{\Delta}}$$

**Define velocity error:** $\boldsymbol{e}_v = \boldsymbol{V}_c - \boldsymbol{V}$

**Error dynamics:**

$$\boxed{\dot{\boldsymbol{e}}_v + \mathbf{K}'\boldsymbol{e}_v = \delta(t)}$$

where:

$$\mathbf{K}' = \bar{\mathbf{M}}(\boldsymbol{q})^{-1}\mathbf{K}$$
$$\delta(t) = \bar{\mathbf{M}}(\boldsymbol{q})^{-1}(\tilde{\boldsymbol{\Delta}} - \bar{\boldsymbol{\Delta}})$$

If $\tilde{\boldsymbol{\Delta}} = \bar{\boldsymbol{\Delta}}$ (perfect estimation), error converges exponentially to zero.

For **position tracking**, consider position error $\boldsymbol{e}$:

$$\boxed{\ddot{\boldsymbol{e}} + k_d \dot{\boldsymbol{e}} + k_p \boldsymbol{e} = 0}$$

**Characteristic equation:**

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

**Standard form:**

- $k_p = \omega_n^2$ (natural frequency)
- $k_d = 2\zeta\omega_n$ (damping)

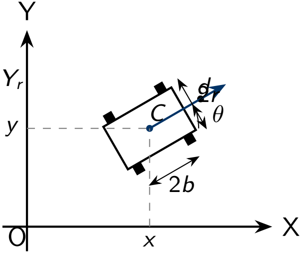
**Design choice:**

- $\zeta = 1$ for critical damping
- $\omega_n$ sets convergence rate

**Goal:** Stabilize robot at desired position $(x_d, y_d)$
Instead of directly controlling $(x, y, \theta)$, define a **virtual output**:

$$\boldsymbol{Y} = \begin{bmatrix} x + \alpha \cos\theta \\ y + \alpha \sin\theta \end{bmatrix}$$

Point at distance $\alpha$ ahead of robot center.




$$\mathbf{M}\ddot{\boldsymbol{q}} + \mathbf{C}\dot{\boldsymbol{q}} + \boldsymbol{\Delta} = \mathbf{B}\boldsymbol{\tau}$$

$$\bar{\mathbf{M}}\dot{\boldsymbol{V}} + \bar{\mathbf{C}}\boldsymbol{V} + \bar{\boldsymbol{\Delta}} = \mathbf{B}\boldsymbol{\tau}$$

**Time derivatives of virtual output:**

$$\dot{\boldsymbol{Y}} = \boldsymbol{\Lambda} \boldsymbol{V} \qquad \ddot{\boldsymbol{Y}} = \dot{\boldsymbol{\Lambda}} \boldsymbol{V} + \boldsymbol{\Lambda} \dot{\boldsymbol{V}}$$

**Decoupling matrix:**

$$\boldsymbol{\Lambda} = \begin{bmatrix} \cos\theta & -\alpha \sin\theta \\ \sin\theta & \alpha \cos\theta \end{bmatrix}$$

Important

$\boldsymbol{\Lambda}$ is invertible when $\alpha \neq 0$, enabling full control of the virtual output.

**Transformed dynamics** in terms of virtual output:

$$\tilde{\mathbf{M}}\ddot{\boldsymbol{Y}} + \tilde{\mathbf{C}}\dot{\boldsymbol{Y}} + \tilde{\boldsymbol{\Delta}} = \tilde{\mathbf{B}}\boldsymbol{\tau}$$

where $\tilde{\mathbf{M}}, \tilde{\mathbf{C}}, \tilde{\boldsymbol{\Delta}}, \tilde{\mathbf{B}}$ are transformed using $\boldsymbol{\Lambda}$.

**PD control law with feedforward:**

$$\boxed{\boldsymbol{\tau} = \tilde{\mathbf{B}}^{-1}\left[\tilde{\mathbf{M}}\ddot{\boldsymbol{Y}}_d + \tilde{\mathbf{C}}\dot{\boldsymbol{Y}} + \tilde{\boldsymbol{\Delta}} + k_d(\dot{\boldsymbol{Y}}_d - \dot{\boldsymbol{Y}}) + k_p(\boldsymbol{Y}_d - \boldsymbol{Y})\right]}$$

For set-point control: $\boldsymbol{Y}_d = \text{const}$, so $\dot{\boldsymbol{Y}}_d = \ddot{\boldsymbol{Y}}_d = 0$.

# Closed-Loop Position Error Dynamics

NYU

With the control law applied, the **position error dynamics** become:

$$\boxed{\ddot{\boldsymbol{e}} + k_d \dot{\boldsymbol{e}} + k_p \boldsymbol{e} = 0}$$

where $\boldsymbol{e} = \boldsymbol{Y}_d - \boldsymbol{Y}$.

**Stability:** For $k_p, k_d > 0$, error converges exponentially to zero.

**Gain selection:**

- $k_p = \omega_n^2$ controls stiffness (steady-state)
- $k_d = 2\zeta\omega_n$ controls damping (transient)

# Summary: Nonlinear Control Methods

NYU

**Key Concepts:**

- Motor and wheel dynamics
- Kinematic Jacobian mapping
- Nonholonomic constraints
- Robot dynamic model $(\mathbf{M}, \mathbf{C}, \mathbf{B})$
- Velocity-space transformation

**Control Approaches:**

- Independent wheel control (PID)
- Feedback linearization
- Trajectory tracking control
- Set-point stabilization
- Virtual output method

Kinematics → Dynamics → Control

# Questions?

End of Lecture 4

*Next: Advanced Model Predictive Control (NMPC)*