Model Predictive Control and Safety-Critical Control

Lecture 7: Nonlinear MPC, Control Lyapunov Functions, and Control Barrier Functions

Autonomous Mobile Robots

Fall 2025

The Fundamental Control Problem

How do we guarantee both **performance** and **safety**?

Liveness (Performance)

- Reach target destinations
- Minimize tracking error
- Optimize metrics

Safety

- Collision avoidance
- Respect physical limits
- Maintain stability

Challenge

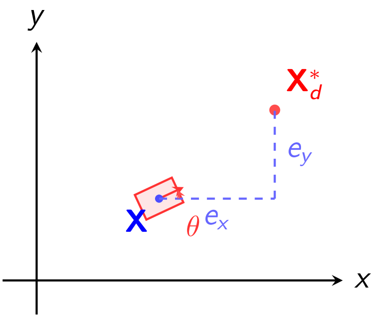
These objectives often **conflict**. Traditional methods struggle when safety and performance contradict.

System Formulation

Control Objective: Navigate to goal $\mathbf{x}_d^*$

State Variables:

$$\begin{aligned} \text{Position:} &\quad x, y \\ \text{Orientation:} &\quad \theta \\ \text{Velocities:} &\quad v, \omega \end{aligned}$$



State Vector:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \end{bmatrix}$$

Mathematical Framework

System Dynamics:

$$\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{u}) \tag{1}$$

- $\mathbf{X}$: State vector (system configuration)
- $\mathbf{u}$: Control input (actuator commands)
- f: Nonlinear dynamics (system physics)

Output Equation:

$$y = h(\mathbf{X})$$

Often only certain states matter (e.g., position: $y = [x, y]^T$)

With Disturbances:

$$\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{u}) + d(t)$$

Assume perfect models initially, address robustness later

From Classical to Optimal Control

Classical Approach:

Design $\mathbf{u} = k(\mathbf{x})$ using:

- Linearization
- Pole placement
- Lyapunov methods

Stable but limited constraint handling

Optimal Control:

Formulate as optimization:

- Explicit performance metric
- Systematic constraints
- Handles nonlinearity

Flexible but computationally intensive

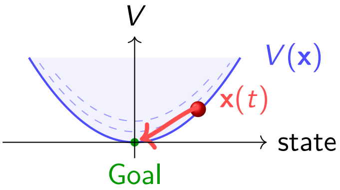
Optimal Control Problem

$$\min_{\mathbf{u}} \quad J(\mathbf{x}, \mathbf{u}) \tag{2}$$

subject to: dynamics, input limits, state constraints

The Cost Function

Cost function J defines optimal behavior



Standard Components:

1. Tracking error: $e^T Q e$ where $e = \mathbf{x} - \mathbf{x}_d$

2. Control effort: $\int_0^T \mathbf{u}^T R \mathbf{u} \, dt$

3. Constraint penalties: Soft penalties for violations

Can construct J to also serve as Lyapunov function

Constraints: Formalizing Safety

Input Constraints: $\mathbf{u} \in \mathcal{U}_a$

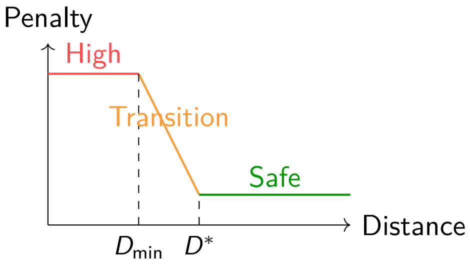
Physical actuator limits:

- Torque bounds
- Velocity limits
- Power constraints

State Constraints: $\mathbf{x} \in \mathcal{X}_s$

Safety requirements:

- Obstacle avoidance
- Lane boundaries
- Stability regions



Implementation:

Hard: Strict inequality constraints

Soft: Penalty functions in cost

NMPC vs MPPI

NMPC

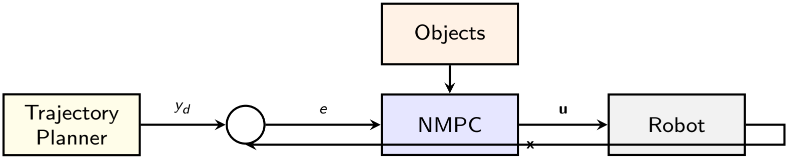
Deterministic optimization

- Explicit system model
- Gradient-based methods
- Local optimum
- Precise when model accurate

MPPI

Probabilistic sampling

- Monte Carlo evaluation
- Tests many trajectories
- Global search capability
- Robust to model errors



NMPC: Prediction-Based Control

Core Concept: Control the present by optimizing the predicted future

NMPC Algorithm

At each time step:

1. Measure current state $\mathbf{x}(t)$
2. Predict evolution over horizon P (1-2 seconds)
3. Optimize control sequence $\{\mathbf{u}_t, \mathbf{u}_{t+1}, \ldots, \mathbf{u}_{t+P}\}$
4. Apply only first control $\mathbf{u}_t$
5. Repeat at next time step

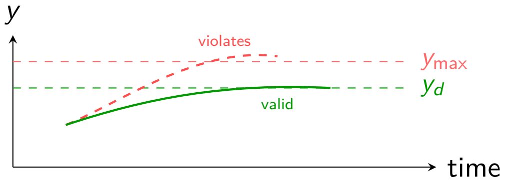
Forward Simulation:

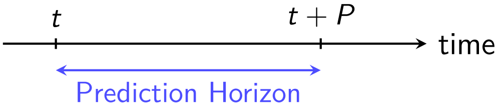
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \cdot f(\mathbf{x}_k, \mathbf{u}_k)$$

Objective:

$$\min_{\{\mathbf{u}_k\}} J = \sum_{k=t}^{t+P} \ell(\mathbf{x}_k, \mathbf{u}_k)$$

Understanding Prediction Horizon





At time t:

- Current state: $\mathbf{x}(t)$
- Horizon: P time steps
- Test control sequences
- Generate predictions

Selection:

- Minimizes cost J
- Satisfies constraints
- Approaches y_d
- Smooth dynamics

NMPC Design Parameters

Critical Choices

1. **Problem Formulation** Deterministic vs probabilistic, time discretization
2. **Objective Function** Quadratic error? Time-optimal? CLF-based?
3. **Constraints** Hard (inequality) vs soft (penalty)
4. **Optimization Solver** Gradient descent, SQP, interior point
5. **Derivative Computation** Analytical (fast) vs numerical (general)

Example: Autonomous Racing

High-Speed Vehicle Control

Objectives:

- Minimize lap time
- Track centerline
- Allow controlled drift

Safety Constraint:

- Prevent spinout
- Maintain stability

Cost Function:

$$J = e_{\text{lat}}^2 + \dot{e}_{\text{lat}}^2 + e_{\theta}^2 + \text{penalties} \tag{3}$$

Implementation:

- Nonlinear conjugate gradient
- Numerical derivatives
- Real-time embedded
- Stability as barrier function

Note

Commercial cruise control uses similar formulations

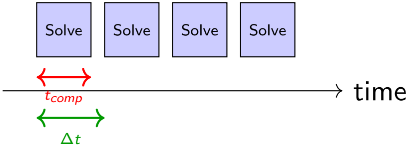
NMPC Challenges

Strengths:

- Handles nonlinearity
- Explicit constraints
- Performance optimization
- Future prediction

Limitations:

- High computational cost
- Non-convex optimization
- No real-time guarantees
- Local optima



Timing Constraint:

Must satisfy:

$$t_{\text{compute}} < \Delta t_{\text{control}}$$

Solution: Use NMPC for performance (slower), safety filter (fast)

The Stability Question

Problem: How do we *guarantee* the system reaches its goal?

Lyapunov Stability Theory

For $\dot{\mathbf{x}} = f(\mathbf{x})$, equilibrium $\mathbf{x}^*$ is stable if $\exists V(\mathbf{x})$:

1. $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{x}^*$ (positive definite)
2. $V(\mathbf{x}^*) = 0$ (zero at equilibrium)
3. $\dot{V}(\mathbf{x}) \leq 0$ (non-increasing)

Interpretation: $V(\mathbf{x})$ is an "energy function" that monotonically decreases, guaranteeing convergence

Control Lyapunov Functions

Extension to Controlled Systems: $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$

CLF Definition

Function $V(\mathbf{x})$ is a CLF if:

1. V is positive definite with $V(\mathbf{x}_d) = 0$
2. For any $\mathbf{x} \neq \mathbf{x}_d$, $\exists \mathbf{u}$ such that:

$$\dot{V}(\mathbf{x}, \mathbf{u}) \leq -\gamma V(\mathbf{x}), \quad \gamma > 0 \tag{4}$$

Transforms stability analysis into control design tool

Condition $\dot{V} \leq -\gamma V$ guarantees exponential convergence

CLF Example: Position Control

Problem:
Robot at $(x, y) \to$ reach (x_d, y_d)

Define error:

$$e = \mathbf{x} - \mathbf{x}_d = \begin{bmatrix} x - x_d \\ y - y_d \end{bmatrix}$$

Choose CLF:

$$V(\mathbf{x}) = \frac{1}{2}\|e\|^2 = \frac{1}{2}(e_x^2 + e_y^2)$$

Squared Euclidean distance to goal

Verify:

- $V > 0$ when $\mathbf{x} \neq \mathbf{x}_d$
- $V = 0$ when $\mathbf{x} = \mathbf{x}_d$
- V differentiable

Time derivative:

$$\dot{V} = e^T \dot{e} = e_x v_x + e_y v_y$$

Control law:

$$v_x = -ke_x, \quad v_y = -ke_y$$

For $k > \gamma/2$:

The CLF Constraint

CLF condition defines stabilizing controls:

$$\mathcal{K}_{\mathsf{CLF}}(\mathbf{x}) = \{\mathbf{u} \mid \dot{V}(\mathbf{x}, \mathbf{u}) \leq -\gamma V(\mathbf{x})\} \tag{5}$$

Any $\mathbf{u} \in \mathcal{K}_{\mathsf{CLF}}$ guarantees progress toward goal

Control Lyapunov Function Constraint

$$\dot{V}(\mathbf{x}, \mathbf{u}) \leq -\gamma V(\mathbf{x})$$

This constraint will be a *soft constraint* in final optimization

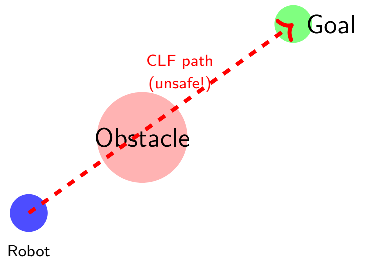
The Safety Problem

CLF Provides:

- Goal convergence
- Error minimization
- System stability

CLF Does NOT Provide:

- Obstacle avoidance
- State constraints
- Safety guarantees



Critical Gap

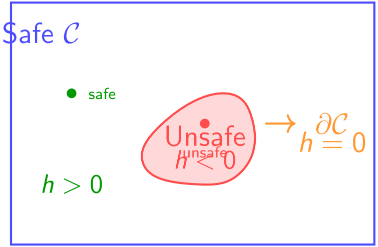
CLF controller chooses shortest path to goal, even through obstacles. Need additional framework for safety.

Control Barrier Functions

Approach: Define barrier function $h(\mathbf{x})$ encoding safe set

Interpretation:

$$h(\mathbf{x}) > 0 \Rightarrow \text{SAFE}$$
$$h(\mathbf{x}) = 0 \Rightarrow \text{BOUNDARY}$$
$$h(\mathbf{x}) < 0 \Rightarrow \text{UNSAFE}$$



Safe Set:

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}$$

Objective: keep $\mathbf{x}(t) \in \mathcal{C}$ for all t

Forward invariance

CBF Requirements

Control Barrier Function

1. **Safe Set:** $\mathcal{C} = \{\mathbf{x} \in D : h(\mathbf{x}) \geq 0\}$
2. **Boundary:** $\partial\mathcal{C} = \{\mathbf{x} : h(\mathbf{x}) = 0\}$
3. **Interior:** $\text{Int}(\mathcal{C}) = \{\mathbf{x} : h(\mathbf{x}) > 0\}$

Function must equal zero *only* on boundary

Forward Invariance

Set $\mathcal{C}$ is **forward invariant** if:

For all $\mathbf{x}(t_0) \in \mathcal{C}$ and all $t \geq t_0$: $\mathbf{x}(t) \in \mathcal{C}$

CBF Safety Condition

To guarantee forward invariance:

$$\dot{h}(\mathbf{x}, \mathbf{u}) \geq -\alpha(h(\mathbf{x})) \tag{6}$$

where $\alpha : \mathbb{R} \to \mathbb{R}$ is extended class-$\mathcal{K}$ (commonly $\alpha(h) = \gamma h$, $\gamma > 0$)

$$\boxed{\dot{h}(\mathbf{x}, \mathbf{u}) \geq -\alpha(h(\mathbf{x}))}$$

CBF safety condition

$$\boxed{\text{At boundary: } \dot{h} \geq 0}$$

Boundary condition

This inequality, when satisfied, mathematically guarantees safety

Understanding the CBF Condition

Condition: $\dot{h}(\mathbf{x}, \mathbf{u}) \geq -\alpha(h(\mathbf{x}))$

Case 1: At Boundary ($h = 0$)

$$\dot{h} \geq -\alpha(0) = 0$$

Time derivative must be non-negative. Robot can:

- Move tangent to boundary ($\dot{h} = 0$)
- Move away from danger ($\dot{h} > 0$)

Movement into unsafe region ($\dot{h} < 0$) is prohibited

Case 2: In Interior ($h > 0$)

$$\dot{h} \geq -\alpha(h) < 0$$

CBF Design Examples

Example 1: Adaptive Cruise

Maintain safe following distance:

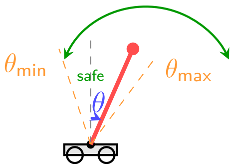
$$h_1 = D - \tau V_{\text{rel}}$$

- D: inter-vehicle distance
- τ: time headway
- V_{rel}: relative velocity

Example 2: Lane Keeping

$$h_2 = d - \frac{\sin(\theta) y_{\text{ref}}}{V^2}$$

Example 3: Pendulum Angle



$$h_1 = \theta - \theta_{\text{min}} \geq 0$$
$$h_2 = -\theta + \theta_{\text{max}} \geq 0$$

Homework: Circular Obstacle

Problem: Avoid circular obstacle

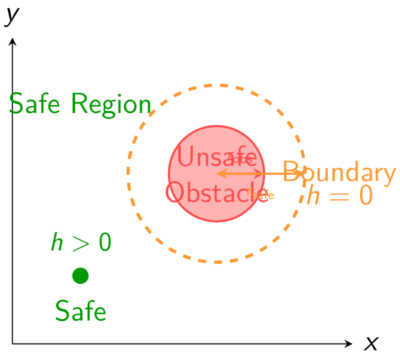
Setup:

- Obstacle center: (x_{obs}, y_{obs})
- Obstacle radius: r_{obs}
- Robot radius: r_{robot}
- Safety radius:

$$r_{safe} = r_{obs} + r_{robot}$$

Barrier Function:

$$h = (x - x_{obs})^2 + (y - y_{obs})^2 - r_{safe}^2$$



The Tractability Challenge

Two constraints established:

$$\text{Stability (CLF):} \quad \dot{V}(\mathbf{x}, \mathbf{u}) \leq -\gamma V(\mathbf{x}) \tag{7}$$

$$\text{Safety (CBF):} \quad \dot{h}(\mathbf{x}, \mathbf{u}) \geq -\alpha(h(\mathbf{x})) \tag{8}$$

Challenge: Both $\dot{V}$ and $\dot{h}$ are complex nonlinear functions of $\mathbf{x}$ and $\mathbf{u}$

Key Question

How to compute and solve efficiently in real-time (100+ Hz)?

Solution: Exploit **control-affine** structure using **Lie derivatives**

Control-Affine Systems

Most robotic systems have control-affine form:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \tag{9}$$

Components:

$f(\mathbf{x})$: Drift vector field (evolution with $\mathbf{u} = 0$)

$g(\mathbf{x})$: Control influence matrix (maps control to state velocity)

Differential Drive:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Driftless ($f = 0$)

$$\boxed{\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}}$$

Deriving Time Derivatives

Chain Rule for $\dot{h}$

$$\dot{h} = \frac{dh}{dt} = \frac{\partial h}{\partial \mathbf{x}}\dot{\mathbf{x}}$$

$$\dot{h} = \frac{\partial h}{\partial \mathbf{x}}[f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}]$$

$$\dot{h} = \underbrace{\frac{\partial h}{\partial \mathbf{x}} f(\mathbf{x})}_{\text{drift term}} + \underbrace{\frac{\partial h}{\partial \mathbf{x}} g(\mathbf{x})}_{\text{control term}} \mathbf{u}$$

Key Result: Control $\mathbf{u}$ now appears **linearly**!

Lie Derivative Notation

Standard names from geometric control:

Lie Derivative Definitions

Lie derivative of h along f:

$$\mathrm{L_f h}(\mathbf{x}) := \frac{\partial h}{\partial \mathbf{x}} f(\mathbf{x})$$

Rate of change due to natural drift

Lie derivative of h along g:

$$\mathrm{L_g h}(\mathbf{x}) := \frac{\partial h}{\partial \mathbf{x}} g(\mathbf{x})$$

How control $\mathbf{u}$ influences rate of change of h

The Power of Lie Derivatives

At any fixed state $\mathbf{x}$, Lie derivatives are constants:

- $L_f h(\mathbf{x})$ is a **scalar**
- $L_g h(\mathbf{x})$ is a **row vector**
- $h(\mathbf{x})$ is a **scalar**

Linear Constraints

Original (nonlinear):

$$\dot{h}(\mathbf{x}, \mathbf{u}) \geq -\alpha(h(\mathbf{x}))$$

With Lie derivatives (linear in u):

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x}))$$

Rearranged:

$$L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x})) - L_f h(\mathbf{x})$$

Computation Example

System: Differential drive, $\mathbf{x} = [x, y, \theta]^T$, $\mathbf{u} = [v, \omega]^T$

Dynamics: $f(\mathbf{x}) = 0$, $g(\mathbf{x}) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}$

Barrier: Circular obstacle

$$h = (x - x_{\text{obs}})^2 + (y - y_{\text{obs}})^2 - r_{\text{safe}}^2$$

Gradient:

$$\nabla h = \begin{bmatrix} 2(x - x_{\text{obs}}) & 2(y - y_{\text{obs}}) & 0 \end{bmatrix}$$

Lie Derivatives:

$$\mathrm{L_f h} = \nabla h \cdot f = 0$$

$$\mathrm{L_g h} = \begin{bmatrix} 2(x - x_{\text{obs}})\cos\theta + 2(y - y_{\text{obs}})\sin\theta & 0 \end{bmatrix}$$

Note: Only v appears; ω has no direct effect

The Control Hierarchy

At time t, we have:

- Current state $\mathbf{x}(t)$
- Desired control $\mathbf{u}_{\text{des}}$ from NMPC
- CLF constraint: $\mathrm{L_f V} + \mathrm{L_g V} \cdot \mathbf{u} \leq -\gamma V$
- CBF constraint: $\mathrm{L_f h} + \mathrm{L_g h} \cdot \mathbf{u} \geq -\alpha h$

Potential Conflicts

1. What if $\mathbf{u}_{\text{des}}$ violates safety?
2. What if safety and stability incompatible?
3. How to prioritize?

Solution: Real-time optimization with:

- Safety (hard constraint)
- Stability (soft constraint)
- Stay close to $\mathbf{u}_{\text{des}}$ (objective)

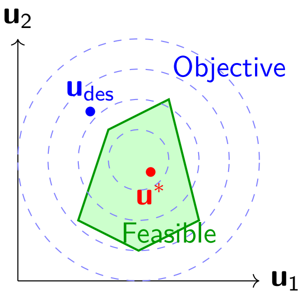
Why Quadratic Programming?

Problem Structure:

Objective: Quadratic $\|\mathbf{u} - \mathbf{u}_{\text{des}}\|^2$

Constraints: Linear (via Lie derivatives)

Result: Convex QP



Properties:

- Global optimum
- Polynomial-time
- Microsecond-scale
- Well-studied algorithms

Naive Formulation

Initial attempt:

$$\min_{\mathbf{u}} \quad \frac{1}{2}\|\mathbf{u} - \mathbf{u}_{\text{des}}\|^2 \tag{10}$$

subject to:

$$L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h) - L_f h \qquad \text{(Safety: CBF)} \tag{11}$$

$$L_g V(\mathbf{x})\mathbf{u} \leq -\gamma V - L_f V \qquad \text{(Stability: CLF)} \tag{12}$$

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \qquad \text{(Actuator limits)} \tag{13}$$

Feasibility Problem

This can be **infeasible**. If goal behind obstacle, no control can simultaneously maintain safety and ensure stability. QP fails!

Slack Variable Technique

Solution: Make CLF constraint "soft" with relaxation variable

Control Hierarchy

1. **Safety (highest):** Hard constraint - always holds
2. **Stability (secondary):** Soft constraint - relaxed when necessary
3. **Performance:** Objective - minimize deviation from $\mathbf{u}_{\mathrm{des}}$

Implementation: Introduce slack $\delta \geq 0$

Modified CLF:

$$\mathrm{L_g V} \cdot \mathbf{u} \leq -\gamma V - \mathrm{L_f V} + \delta$$

- $\delta = 0$: Original CLF satisfied
- $\delta > 0$: Stability relaxed (safety priority)

Final CLF-CBF-QP

Complete Optimization

Find optimal $(\mathbf{u}^*, \delta^*)$:

$$\min_{\mathbf{u},\delta} \quad \frac{1}{2}\|\mathbf{u} - \mathbf{u}_{\mathrm{des}}\|^2 + \frac{p}{2}\delta^2 \tag{14}$$

subject to:

$$\begin{aligned} \mathrm{L_g h} \cdot \mathbf{u} &\geq -\alpha(h) - \mathrm{L_f h} && \textbf{(HARD: Safety)} \\ \mathrm{L_g V} \cdot \mathbf{u} &\leq -\gamma V - \mathrm{L_f V} + \delta && \text{(Soft: Stability)} \\ \mathbf{u}_{\min} \leq \mathbf{u} &\leq \mathbf{u}_{\max}, \quad \delta \geq 0 \end{aligned}$$

where $p \gg 1$ (typically 10^3 to 10^4)

Large p ensures δ minimized. Stability violated *only* when necessary for safety.

Layered Strategy

Philosophy: Separate *performance* from *safety*

Layer 1: NMPC

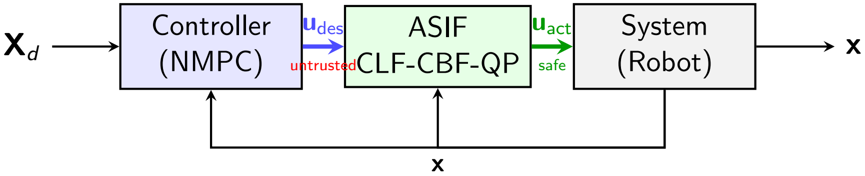
- Rate: 10-50 Hz
- Complex, non-convex
- Horizon: 1-2 sec
- Output: $\mathbf{u}_{\text{des}}$
- **Untrusted**

Layer 2: ASIF

- Rate: 100-1000 Hz
- Fast QP (convex)
- Instantaneous
- Output: $\mathbf{u}_{\text{act}}$
- **Trusted**

Advantage: Use sophisticated (unreliable) planning while maintaining provable safety

ASIF Safety Filter



ASIF Cycle:

1. Receive $\mathbf{u}_{\text{des}}$ and $\mathbf{x}$
2. Compute Lie derivatives
3. Formulate CLF-CBF-QP
4. Solve QP (< 1 ms)
5. Output safe $\mathbf{u}_{\text{act}}$

ASIF: Benefits and Limitations

Advantages:

- Microsecond computation
- Formal safety guarantee
- Modular architecture
- Minimal intervention
- Works with any planner

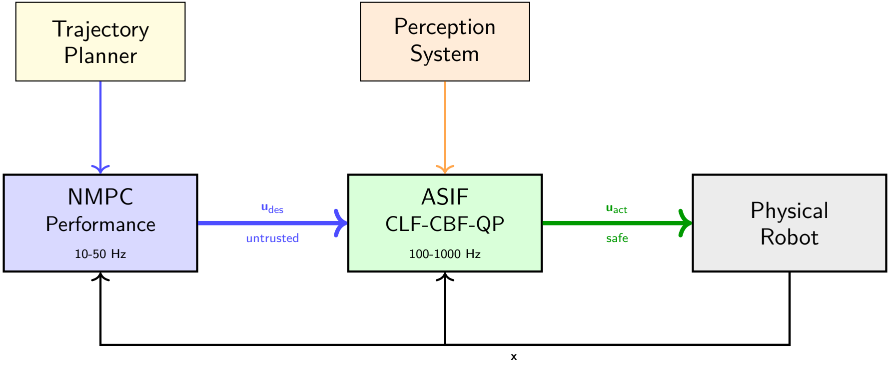
Can use learning-based policies, game theory, or human teleoperation with safety enforced

Limitations:

- Performance layer safety-ignorant
- May generate unsafe commands
- Frequent intervention reduces efficiency
- No conflict resolution

Alternative:
Safety-Aware NMPC embeds CBF directly in optimization

Complete System



State feedback $\mathbf{x}$ to both layers. ASIF has final authority.

The Critical Assumption

Assumption So Far

Perfect system model:

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

Reality: All models are wrong

$$\dot{\mathbf{x}}_{\text{true}} = f(\mathbf{x}, \mathbf{u}) + d(\mathbf{x}, t)$$

Sources of d:

- Wheel slip
- Unknown mass
- Wind disturbances
- Actuator delays
- Sensor noise
- Friction models

Why Model Error Breaks Safety

CBF condition:

$$\dot{h} = \mathrm{L_f h} + \mathrm{L_g h} \cdot \mathbf{u} \geq -\alpha(h)$$

With model error:

$$\dot{h}_{\text{true}} = \mathrm{L_f h} + \mathrm{L_g h} \cdot \mathbf{u} + \underbrace{\frac{\partial h}{\partial \mathbf{x}} d}_{\substack{\text{unknown} \\ \text{disturbance}}}$$

Safety Guarantee Voided

- We compute $\mathrm{L_f h}$, $\mathrm{L_g h}$ using incorrect model f
- True system evolves according to $f + d$
- QP solution may not satisfy true safety condition
- **Mathematical guarantee no longer holds**

Robust Safety Approaches

Three Research Directions:

1. Robust CBF (Worst-Case)

If $\|d\| \leq d_{\max}$, modify safety:

$$\mathrm{L_g h} \cdot \mathbf{u} \geq -\alpha(h) - \mathrm{L_f h} + \|\nabla h\| d_{\max}$$

Trade-off: Conservative (reduces performance)

2. Adaptive CBF (Learning)

Estimate $d(\mathbf{x}, t)$ in real-time using:

- Extended Kalman Filters
- Gaussian Process Regression
- Neural network observers

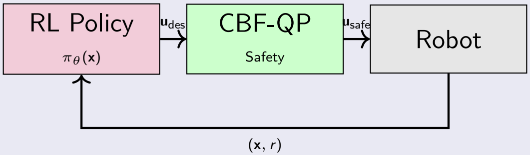
Learning-Based Safe Control

3. Reinforcement Learning + CBF

Concept: Combine learning flexibility with formal safety

- **RL Agent:** Learns optimal policies from experience
- **CBF-QP Filter:** Ensures safety during exploration
- **Result:** Safe learning without crashes

Architecture:



Homework

Overview

Apply Control Barrier Functions to ensure safe robot navigation around a pedestrian.

Requirements:

1. **Scenario:** Robot must reach goal while avoiding pedestrian obstacle
2. **Define geometric constraint:**

$$h() = \|_{\text{robot}} -_{\text{pedestrian}} \| - d_{\text{safe}} \geq 0$$

3. **Apply CBF approach:** Use Control Barrier Function to guarantee safety
4. **Write 5 equations:** System dynamics, barrier function, CBF condition, control bounds, distance formula
5. **Show solution:** Demonstrate robot reaches goal while maintaining safe distance from pedestrian

Questions?

End of Lecture 7