

Lecture 9 - Mixed-integer Model Predictive Control

Dinesh Krishnamoorthy

Department of Engineering Cybernetics, NTNU

Fall 2026

Why hybrid / mixed-integer MPC?

Standard MPC assumes **continuous** inputs $u \in \mathcal{U} \subseteq \mathbb{R}^m$. But many real decisions are **discrete**:

- ▶ on/off equipment: pumps, compressors, heaters, valves;
- ▶ how many units to run (parallel chillers, boilers);
- ▶ mode selection, switches (e.g., gear changes, PWA models);
- ▶ Disjoint constraint sets;
- ▶ Logics and rules, e.g., “if temperature $>$ limit, then ...” type conditions.

How do we bring discrete decisions and logic directly into the MPC problem?

Hybrid MPC: optimal control of systems mixing continuous dynamics with discrete / logic decisions $\Rightarrow$ **mixed-integer** optimization.

Where do the integer variables come from?

- ▶ **Discrete actuators**: on/off units, unit counts $\Rightarrow$ disconnected input set $\mathcal{U}$.
- ▶ **Logic & rules**: if-then-else, AND / OR conditions, constraint switching.
- ▶ **Piecewise-affine (PWA) dynamics**: approximation of nonlinear or switched systems.
- ▶ **Disjoint (non-convex) constraint sets**: obstacle / collision avoidance.

All four introduce **binary variables** $\delta \in \{0, 1\}$ and turn the MPC problem into a **mixed-integer program** (MILP / MIQP).

How much does the MPC theory change?

Surprisingly little. Recall the assumptions from Lecture 4:

A2. $\mathcal{Z} = \mathcal{X} \times \mathcal{U}$ is closed; $\mathcal{U}$ compact.

We **never** assumed that $\mathcal{U}$ has an interior, or that the equilibrium lies in the interior of $\mathcal{U}$.

A **discrete-valued** $\mathcal{U}$ (e.g. $\{0, 1, 2\}$) is still **closed and compact**.

$\Rightarrow$ the nominal closed-loop **stability results carry over unchanged** (existence of the minimizer, Lyapunov descent, recursive feasibility).

What does change?

- ▶ The feasible set $\mathcal{X}_{feas}$ may be **disconnected** – impossible for a linear system with convex $\mathcal{U}$.
- ▶ The online problem becomes a **mixed-integer program** (MILP / MIQP), solved by branch-and-bound.
- ▶ Terminal design: discrete actuators are usually **fixed** near the steady state; fine regulation is done by the continuous actuators (as with saturated actuators).

Bottom line: the *theory* needs little change; the *modelling and computation* is where the work moves.

MPC with discrete actuators

Split the input into continuous and discrete parts: $u = (u_c, u_\ell)$, $u_\ell \in \{0, 1, \dots\}$.

Typical example: a bank of n identical parallel units (chillers, compressors, boilers). Decide *how many* and *which* to run as the load varies.

- ▶ In practice these decisions are made by **heuristics** outside the MPC layer.
- ▶ Choosing them **optimally** inside MPC can greatly improve performance.

The price: $\mathcal{U}$ becomes a **disconnected** set.

Discrete actuators: on/off units

A single unit delivers a duty Q that is either **off** ($Q = 0$) or **on** ($Q \in [Q_{\min}, Q_{\max}]$), with $Q_{\min} > 0$ – a **semicontinuous** variable. Introduce a binary $\delta \in \{0, 1\}$ ($\delta = 1$: on):

$$\delta Q_{\min} \leq Q \leq \delta Q_{\max}$$

- ▶ $\delta = 0 \Rightarrow Q = 0$ (off),
- ▶ $\delta = 1 \Rightarrow Q \in [Q_{\min}, Q_{\max}]$ (on).

For n units: total duty $Q = \sum_i Q_i$, number on $= \sum_i \delta_i$. When

$Q_{\min} > 0$ the reachable duties $\{0\} \cup [Q_{\min}, Q_{\max}]$ form a **disconnected** set.

Resulting MPC problem (discrete actuators)

$$\begin{aligned} \min \quad & \sum_{k=0}^{N-1} \|x_k - x_s\|_Q^2 + \|u_k - u_s\|_R^2 \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k \\ & x_k \in \mathcal{X}, \quad u_k \in \mathcal{U} \\ & \delta_{k,i} Q_{\min} \leq Q_{k,i} \leq \delta_{k,i} Q_{\max} \\ & \delta_{k,i} \in \{0, 1\} \end{aligned}$$

- ▶ Quadratic cost + linear constraints + binary variables $\Rightarrow$ **MIQP**.
- ▶ Solved online by branch-and-bound.

Piecewise-affine (PWA) approximation

A nonlinear model $x^+ = f(x, u)$ can be approximated by a PWA model. Partition the (x, u) -space into s polyhedral regions

$$\mathcal{R}_i = \{\xi = (x, u) : H_i \xi \leq K_i\}, \quad i = 1, \dots, s,$$

and fit an affine model on each:

$$x^+ = A_i x + B_i u + c_i \quad \text{if } (x, u) \in \mathcal{R}_i.$$

Goal: write the resulting MPC problem as a mixed-integer program.

PWA $\rightarrow$ MIQP: mode selection

Introduce, at each time k , a binary per region $\delta_{k,i} \in \{0, 1\}$:
 $\delta_{k,i} = 1 \Leftrightarrow (x_k, u_k) \in \mathcal{R}_i$.

(a) Exactly one region active: $\sum_{i=1}^s \delta_{k,i} = 1$.

(b) Region membership (guard): $H_i[x_k; u_k] \leq K_i + \bar{M}_i(1 - \delta_{k,i})$.

The guard is essential: without it the optimizer would apply whichever affine model is cheapest, regardless of where the state actually is.

PWA $\rightarrow$ MIQP: dynamics via auxiliary variables

The active dynamics are a **product** of a binary and an affine term.

Introduce $z_{k,i} \in \mathbb{R}^n$ with $z_{k,i} = \delta_{k,i}(A_i x_k + B_i u_k + c_i)$ and

$$x_{k+1} = \sum_{i=1}^s z_{k,i}.$$

Encode the product exactly with big-M (componentwise), where

$$m_i \leq A_i x_k + B_i u_k + c_i \leq M_i$$

$$z_{k,i} \leq M_i \delta_{k,i},$$

$$z_{k,i} \geq m_i \delta_{k,i},$$

$$z_{k,i} \leq w_{k,i} - m_i(1 - \delta_{k,i}),$$

$$z_{k,i} \geq w_{k,i} - M_i(1 - \delta_{k,i}).$$

$$\delta_{k,i} = 1 \Rightarrow z_{k,i} = w_{k,i}; \quad \delta_{k,i} = 0 \Rightarrow z_{k,i} = 0.$$

PWA $\rightarrow$ MIQP: the full problem

$$\begin{aligned} \min \quad & \sum_{k=0}^{N-1} \|x_k - x_s\|_Q^2 + \|u_k - u_s\|_R^2 + \|x_N - x_s\|_P^2 \\ \text{s.t.} \quad & x_{k+1} = \sum_i z_{k,i} && \text{(dynamics)} \\ & z_{k,i} \leq M_i \delta_{k,i}, \\ & z_{k,i} \geq m_i \delta_{k,i}, \\ & z_{k,i} \leq w_{k,i} - m_i(1 - \delta_{k,i}), \\ & z_{k,i} \geq w_{k,i} - M_i(1 - \delta_{k,i}) && \text{(big-M, 4 ineq.)} \\ & H_i[x_k; u_k] \leq K_i + \bar{M}_i(1 - \delta_{k,i}) && \text{(guard)} \\ & \sum_i \delta_{k,i} = 1 && \text{(one region)} \\ & x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \\ & x_0 = x(t) \\ & \delta_{k,i} \in \{0, 1\} \end{aligned}$$

Quadratic cost + linear constraints + integrality $\Rightarrow$ MIQP.

Disjoint constraint sets

Sometimes the feasible set is a **union** of polyhedra (non-convex):

$$\xi \in \bigcup_{i=1}^s \mathcal{P}_i, \quad \mathcal{P}_i = \{\xi : A_i \xi \leq b_i\}.$$

We want ξ to satisfy **at least one** of the sets:

$$\bigvee_{i=1}^s [A_i \xi \leq b_i].$$

Encode with binaries: $\delta_i \in \{0, 1\}$, $\delta_i = 1 \Rightarrow$ set i is enforced.

Disjoint sets: big-M formulation

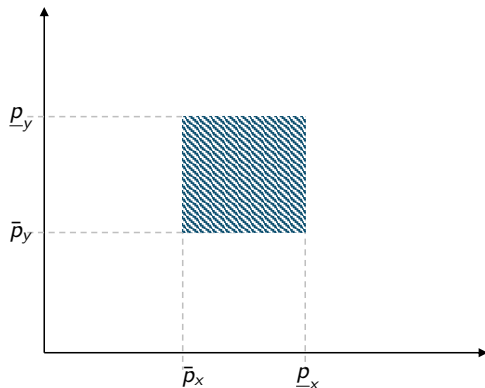
$$\begin{aligned} A_i \xi &\leq b_i + M_i(1 - \delta_i), & i = 1, \dots, s \\ \sum_{i=1}^s \delta_i &\geq 1, & \delta_i \in \{0, 1\} \end{aligned}$$

Change only the count constraint for different logic:

at least one (union)	$\sum_i \delta_i \geq 1$
exactly one (partition)	$\sum_i \delta_i = 1$
at most one	$\sum_i \delta_i \leq 1$
at least k of s	$\sum_i \delta_i \geq k$

Example: obstacle / collision avoidance

Keep the position $p = (p_x, p_y)$ **outside** a box $[\underline{p}_x, \bar{p}_x] \times [\underline{p}_y, \bar{p}_y]$



Example: obstacle / collision avoidance

To keep a position $p = (p_x, p_y)$ **outside** a box $[\underline{p}_x, \bar{p}_x] \times [\underline{p}_y, \bar{p}_y]$, it must violate **at least one** face:

$$p_x \leq \underline{p}_x \vee p_x \geq \bar{p}_x \vee p_y \leq \underline{p}_y \vee p_y \geq \bar{p}_y.$$

$$p_x \leq \underline{p}_x + M(1 - \delta_1), \quad -p_x \leq -\bar{p}_x + M(1 - \delta_2),$$

$$p_y \leq \underline{p}_y + M(1 - \delta_3), \quad -p_y \leq -\bar{p}_y + M(1 - \delta_4).$$

$$\delta_1 + \delta_2 + \delta_3 + \delta_4 \geq 1, \quad \delta_j \in \{0, 1\}.$$

The complement of a convex set is a disjunction over its faces $\Rightarrow$ non-convex.

Encoding logic with binary variables

Boolean variables $\delta_i \in \{0, 1\}$ (1 = true, 0 = false). Logic relations become **linear inequalities**:

relation	Boolean	linear constraint
AND	$\delta_1 \wedge \delta_2$	$\delta_1 + \delta_2 \geq 2$
OR	$\delta_1 \vee \delta_2$	$\delta_1 + \delta_2 \geq 1$
XOR	$\delta_1 \oplus \delta_2$	$\delta_1 + \delta_2 = 1$
IMPLY	$\delta_1 \rightarrow \delta_2$	$\delta_1 - \delta_2 \leq 0$
IFF	$\delta_1 \leftrightarrow \delta_2$	$\delta_1 - \delta_2 = 0$

Any Boolean formula can be rewritten as a set of such linear inequalities.

Linking logic to continuous variables: threshold events

Associate a binary with a **linear threshold** on a continuous variable:

$$[\delta = 1] \Leftrightarrow [f(x) \leq 0],$$

with bounds $m \leq f(x) \leq M$ over the domain. Encode with big-M:

$$\begin{aligned} f(x) &\leq M(1 - \delta) \\ f(x) &\geq \epsilon + (m - \epsilon)\delta \end{aligned}$$

(ϵ = small tolerance, avoids a strict inequality)

- ▶ $\delta = 1 \Rightarrow f(x) \leq 0,$
- ▶ $\delta = 0 \Rightarrow f(x) > 0.$

Products: if-then-else via auxiliary variables

A product of a binary and a continuous term, $z = \delta f(x)$ ($z = f(x)$ if $\delta = 1$, else $z = 0$), is encoded exactly by

$$\begin{aligned} z &\leq M\delta, & z &\geq m\delta, \\ z &\leq f(x) - m(1 - \delta), & z &\geq f(x) - M(1 - \delta). \end{aligned}$$

This is the building block behind:

- ▶ switched / PWA dynamics: $z_{k,i} = \delta_{k,i}(A_i x + B_i u + c_i)$,
- ▶ if-then-else rules and mixed logical dynamical (MLD) models.

Summary: hybrid / mixed-integer MPC

- ▶ Discrete decisions, logic, PWA dynamics and disjoint constraints all introduce **binary variables**.
- ▶ The stability *theory* carries over ($\mathcal{U}$ only needs to be closed / compact); the *computation* becomes an **MILP / MIQP**.
- ▶ Key encodings: **semicontinuous** inputs, **big-M** guards and products, **disjunctions**, and **logic-to-inequality** rules.

Same MPC principles – richer models – solved as a mixed-integer program.

Mixed-integer MPC

Many control problems involve **discrete decisions** alongside continuous inputs:

- ▶ on/off actuators, valves, compressors; gear or mode selection; logical (if-then) rules.
- ▶ These are modelled with **integer / binary** variables $\delta \in \{0, 1\}$.

With a quadratic cost and linear dynamics/constraints, the MPC problem becomes a **Mixed-Integer Quadratic Program (MIQP)**:

$$\begin{aligned} \min_{u_c, \delta} \quad & \sum_{k=0}^{N-1} x_k^\top Q x_k + u_k^\top R u_k + V_f(x_N) \\ \text{s.t.} \quad & x_{k+1} = A x_k + B u_k, \quad x_k \in \mathcal{X}, \quad u_k \in \mathcal{U} \\ & u_k = \left(\underbrace{u_{c,k}}_{\text{continuous}}, \underbrace{\delta_k}_{\in \{0,1\}} \right) \end{aligned}$$

The continuous part alone is an easy convex QP — the integers are what make it hard.

Solving mixed-integer problems is NP-hard

What does “NP-hard” mean? (informally)

- ▶ **P**: problems solvable in *polynomial time* — “efficient” (e.g. a convex QP).
- ▶ **NP**: problems where a *proposed* solution can be *checked* quickly, even if finding one is hard. *Easy to verify, hard to solve* (like a filled-in Sudoku).
- ▶ **NP-hard**: at least as hard as the hardest problems in NP. **No known algorithm solves them in polynomial time**; worst-case effort grows **exponentially** with problem size.

With m binary variables there are 2^m possible combinations. Checking a horizon of N steps with several binaries each $\Rightarrow$ the search space *explodes*.

Consequence for MPC: solve time has *no polynomial guarantee* — a real concern when we must solve within a fixed sampling interval. (Many instances *do* solve fast in practice — NP-hardness is a *worst-case* statement.)

How MIQPs are solved: branch & bound (high level)

We never enumerate all 2^m combinations. Branch & bound explores a **tree** of subproblems and prunes most of it.

- ▶ **Relax.** Drop integrality ($\delta \in \{0, 1\} \rightarrow \delta \in [0, 1]$) and solve the **convex QP relaxation**. Its optimum is a **lower bound** on the true cost.
- ▶ **Branch.** If a δ comes out fractional (say 0.4), split into two subproblems: fix $\delta = 0$ and fix $\delta = 1$. Recurse.
- ▶ **Bound.** The best integer-feasible solution found so far is the **incumbent** (an upper bound).
- ▶ **Prune.** If a node's relaxation is already worse than the incumbent, discard that *entire* subtree — no need to explore it.

Pruning is the whole trick: it lets us find the *global* optimum while touching only a small fraction of the 2^m leaves — but the worst case is still exponential.

Approximations I: round-off

For real-time use we often *avoid* the full MIP and settle for a good-enough integer solution.

Rounding heuristic:

- ▶ Solve the **continuous relaxation** once (a single convex QP).
- ▶ **Round** each relaxed binary δ_i^* to the nearest integer (0 or 1).
- ▶ Optionally re-solve for the continuous inputs u_c with the integers now fixed.

Pros

- ▶ Very fast: essentially one QP.
- ▶ Trivial to implement.

Cons

- ▶ **Suboptimal**; no optimality guarantee.
- ▶ Rounding can **violate constraints**.

Approximations II: penalty-term homotopy

Idea: replace the discrete constraint by a **smooth penalty** and solve a sequence of *continuous* problems.

- ▶ Key identity: $\delta \in \{0, 1\} \iff \delta(1 - \delta) = 0, \delta \in [0, 1]$.
- ▶ Add a penalty pushing variables toward integers:

$$\min_{u_c, \delta \in [0, 1]} J(u_c, \delta) + \mu \sum_i \delta_i(1 - \delta_i).$$

- ▶ **Homotopy / continuation:** start at $\mu = 0$ (the convex relaxation), then *gradually increase* μ , warm-starting each solve from the previous one. As $\mu \rightarrow \infty$ the solution is driven to $\{0, 1\}$.

Pros

- ▶ Trades a combinatorial search for a sequence of smooth NLPs
- ▶ Cheap and often effective

Cons

- ▶ Penalty is **nonconvex**
- ▶ While it solves suboptimality issue, it still does not fully solve the constraint feasibility issue.