

Introduction to Model Predictive Control

From Dynamic Programming and LQR to Receding Horizon Control

Dinesh Krishnamoorthy

Department of Engineering Cybernetics, NTNU

Spring 2026

Formulating control design as an optimization problem

- ▶ The fundamental idea in optimal control is to formulate the goal of control as the long-term optimization of a scalar cost function.
- ▶ In the additive cost formulation, the long-term objective for a trajectory can be written as

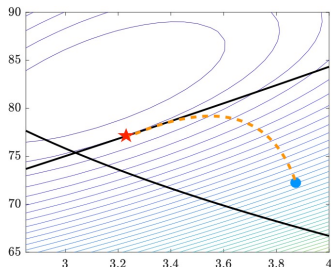
$$\min \sum_{k=0}^{\infty} \ell(\mathbf{x}_k, \mathbf{u}_k)$$

General dynamic optimization problem

$$\begin{aligned} \min_w \quad & \sum_{k=0}^{\infty} \ell(x_k, u_k) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k) \\ & g(x_k, u_k) \leq 0 \\ & x_0 = x(t) \end{aligned}$$

x_0 given, and

$$w := [x_0^\top, u_0^\top, x_1^\top, u_1^\top, x_2^\top, \dots]^\top$$



Open-loop optimal policy

$$\pi : \{0, 1, \dots, T-1\} \rightarrow \mathcal{U}$$

- ▶ Time-indexed sequence of actions calculated offline
- ▶ Independent of the state x , i.e., no feedback

Control policy

Given what I know about the system right now, what action should I apply next to achieve my goal?

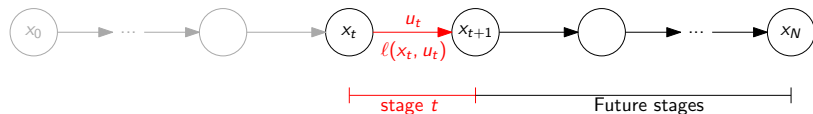
- ▶ A control policy $u = \pi(x)$ is a formal mathematical object that expresses this decision-making logic.

Control policy

A policy $\pi : \mathcal{X} \rightarrow \mathcal{U}$ is a mapping from the states x to the action u that we would apply to the dynamic system.

- ▶ But not all policies are equally good.
- ▶ Finding the best policy is an optimization problem

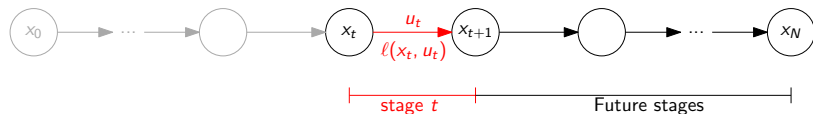
Finding the optimal policy



Decision-making: What do I need to take into account?

cost now + future costs

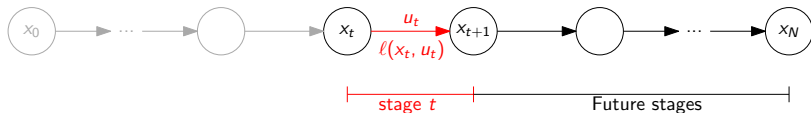
Finding the optimal policy



Decision-making: What do I need to take into account?

$$\min_u (\text{cost now} + \text{future costs})$$

Finding the optimal policy



Decision-making: What do I need to take into account?

$$\min_u (\text{cost now} + \text{future costs})$$

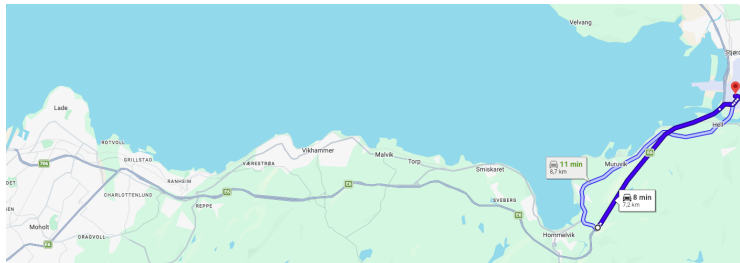
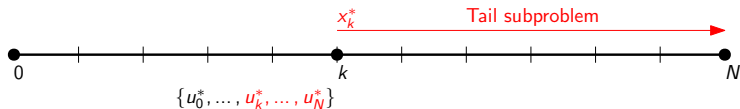
Optimal cost function

$$J^*(x_t) = \min_{u_k \in U_k(x_k)} \ell_t(x_t, u_t) + \sum_{k=t+1}^{N-1} \ell_k(x_k, u_k) + \ell_N(x_N)$$

Optimal policy

$$\pi^*(x_t) = u_t^*$$

Principle of optimality



Every optimal policy consists only of optimal sub policies.

Optimizing Multistage functions

Because of the special structure, we can solve by optimizing a sequence of single-variable problems

$$\min_x \left[f(w, x) + \min_y \left[g(x, y) + \min_z h(y, z) \right] \right]$$

we solve the inner problem over z first. The optimal solution will be a function of y .

$$h^*(y) = \min_z h(y, z)$$

and move on to the next stage

$$g^*(x) = \min_y g(x, y) + h^*(y)$$

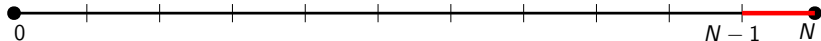
and so on,

$$\min_x = f(w, x) + g^*(x)$$

This nested approach is an example of a class of techniques known as [Dynamic Programming](#) (DP).

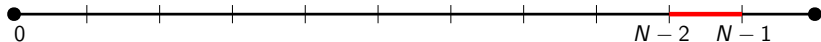
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



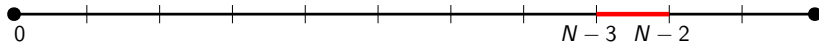
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



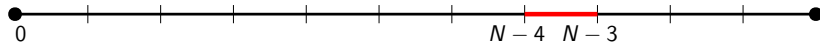
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



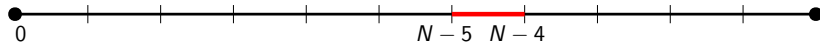
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



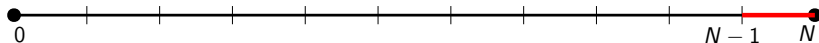
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



DP recursion - Produces the optimal costs $J_k^*(x_k)$ of the x_k -tail subproblems

Start with

$$J_N^*(x_N) = \ell_N(x_N), \quad \text{for all } x_N$$

and for all $k = N - 1, N - 2, \dots, 0$

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \underbrace{\ell_k(x_k, u_k)}_{\text{cost now}} + \underbrace{J_{k+1}^*(f(x_k, u_k))}_{\text{cost-to-go}}, \quad \text{for all } x_k$$

Then the optimal cost $J_0^*(x_0)$ is obtained at the last step

Exact DP algorithm

Go backwards to compute the optimal costs $J_k^*(x_k)$

Start with

$$J_N^*(x_N) = \ell_N(x_N), \quad \text{for all } x_N$$

and for all $k = N - 1, N - 2, \dots, 0$

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \underbrace{\ell_k(x_k, u_k)}_{\text{cost now}} + \underbrace{J_{k+1}^*(f(x_k, u_k))}_{\text{opt. cost-to-go}}, \quad \text{for all } x_k$$

Then the optimal cost $J^*(x_0)$ is obtained at the last step

Go forwards to construct the optimal sequence $\{u_0^*, \dots, u_{N-1}^*\}$

Given x_0 , start with

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} [\ell_0(x_0, u_0) + J_1^*(f_0(x_0, u_0))], \quad x_1^* = f_0(x_0, u_0^*).$$

Going forward sequentially $k = 1, 2, \dots, N - 1$, we get

$$u_k^* \in \arg \min_{u_k \in U_k(x_k)} [\ell_k(x_k^*, u_k) + J_{k+1}^*(f_k(x_k^*, u_k))], \quad x_{k+1}^* = f_k(x_k^*, u_k^*).$$

Exact DP - Illustrative Example

- Consider a system

$$x_{k+1} = x_k + u_k$$

with $x_k \in \mathbb{R}$ and $u_k \in \mathbb{R}$

- OCP:

$$\min \sum_{k=0}^{N-1} (x_k^2 + u_k^2) + x_N^2$$

with $N = 3$

Exact DP - Illustrative Example

- ▶ Consider a system $x_{k+1} = x_k + u_k$ with $x_k \in \mathbb{R}$, $u_k \in \mathbb{R}$
- ▶ OCP: $\min \sum_{k=0}^{N-1} (x_k^2 + u_k^2) + x_N^2$, with $N = 3$

$$J_k^*(x_k) = \min_{u_k} \ell(x_k, u_k) + J_{k+1}^*(x_{k+1})$$

at stage $k = 2$, begin with $J(3) = x_3^2$

$$\begin{aligned} J_2(x_2) &= x_2^2 + u_2^2 + x_3^2 \\ &= x_2^2 + u_2^2 + (x_2 + u_2)^2 \\ \nabla_{u_2} J_2(x_2) &= 2u_2 + 2(x_2 + u_2) = 0 \\ \Rightarrow u_2^* &= -0.5x_2, \quad J_2^*(x_2) = 1.5x_2^2 \end{aligned}$$

Exact DP - Illustrative Example

- ▶ Consider a system $x_{k+1} = x_k + u_k$ with $x_k \in \mathbb{R}$, $u_k \in \mathbb{R}$
- ▶ OCP: $\min \sum_{k=0}^{N-1} (x_k^2 + u_k^2) + x_N^2$, with $N = 3$

$$J_k^*(x_k) = \min_{u_k} \ell(x_k, u_k) + J_{k+1}^*(x_{k+1})$$

$$\text{at stage } k = 2 \Rightarrow u_2^* = -0.5x_2, \quad J_2^*(x_2) = 1.5x_2^2$$

at stage $k = 1$

$$\begin{aligned} J_1(x_1) &= x_1^2 + u_1^2 + J_2^*(x_2) \Rightarrow x_1^2 + u_1^2 + 1.5x_2^2 \\ &= x_1^2 + u_1^2 + 1.5(x_1 + u_1)^2 \\ \nabla_{u_1} J_1(x_1) &= 2u_1 + 3(x_1 + u_1) = 0 \\ &\Rightarrow u_1^* = -3/5x_1, \quad J_1^*(x_1) = 8/5x_1^2 \end{aligned}$$

Exact DP - Illustrative Example

- ▶ Consider a system $x_{k+1} = x_k + u_k$ with $x_k \in \mathbb{R}$, $u_k \in \mathbb{R}$
- ▶ OCP: $\min \sum_{k=0}^{N-1} (x_k^2 + u_k^2) + x_N^2$, with $N = 3$

$$J_k^*(x_k) = \min_{u_k} \ell(x_k, u_k) + J_{k+1}^*(x_{k+1})$$

$$\text{at stage } k = 2 \Rightarrow u_2^* = -0.5x_2, \quad J_2^*(x_2) = 1.5x_2^2$$

$$\text{at stage } k = 1 \Rightarrow u_1^* = -3/5x_1, \quad J_1^*(x_1) = 8/5x_1^2$$

at stage $k = 0$

$$\begin{aligned} J_0(x_0) &= x_0^2 + u_0^2 + J_1^*(x_1) \Rightarrow x_0^2 + u_0^2 + 8/5x_1^2 \\ &= x_0^2 + u_0^2 + 8/5(x_0 + u_0)^2 \\ \nabla_{u_0} J_0(x_0) &= 2u_0 + 16/5(x_0 + u_0) = 0 \\ &\Rightarrow u_0^* = -8/13x_0, \quad J_0^*(x_0) = 21/26x_0^2 \end{aligned}$$

Exact DP - Illustrative Example

- ▶ Consider a system $x_{k+1} = x_k + u_k$ with $x_k \in \mathbb{R}$, $u_k \in \mathbb{R}$
- ▶ OCP: $\min \sum_{k=0}^{N-1} (x_k^2 + u_k^2) + x_N^2$, with $N = 3$

$$J_k^*(x_k) = \min_{u_k} \ell(x_k, u_k) + J_{k+1}^*(x_{k+1})$$

$$\text{at stage } k = 2 \Rightarrow u_2^* = -0.5x_2, \quad J_2^*(x_2) = 1.5x_2^2$$

$$\text{at stage } k = 1 \Rightarrow u_1^* = -3/5x_1, \quad J_1^*(x_1) = 8/5x_1^2$$

$$\text{at stage } k = 0 \Rightarrow u_0^* = -8/13x_0, \quad J_0^*(x_0) = 21/26x_0^2$$

Go **backward** to compute the optimal cost, go **forward** to construct the optimal sequence

Exact DP - curse of dimensionality

Exact DP computes the optimal cost-to-go $J_k^*(x)$ for **every** state x and stage k .

- ▶ Discretize each of the n states into m points $\Rightarrow m^n$ grid points *per stage*.
- ▶ Cost grows **exponentially** with the state dimension n – Bellman's “curse of dimensionality”.
- ▶ Exact DP is therefore intractable beyond very small systems.

Control policy

- ▶ LQR gives the linear policy $u = -Kx$ that optimizes

$$\begin{aligned} \min \quad & \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \\ \text{s.t.} \quad & x_{k+1} = A x_k + B u_k \end{aligned}$$

- ▶ MPC policy is obtained by repeatedly solving an optimization problem
- ▶ RL policy - learn what action to take at any given state based on rewards from the environment

From LQR to MPC: what is missing?

LQR gives an elegant **offline** feedback law $u = -Kx$, optimal for the *unconstrained* linear–quadratic problem.

- Real systems have **constraints**: actuator limits $u \in \mathcal{U}$, safety / quality limits $x \in \mathcal{X}$.
- Real systems are often **nonlinear**.
- Exact DP is **intractable** for such problems (curse of dimensionality); LQR simply ignores constraints.

How can we keep the DP idea of optimal feedback, while respecting constraints?

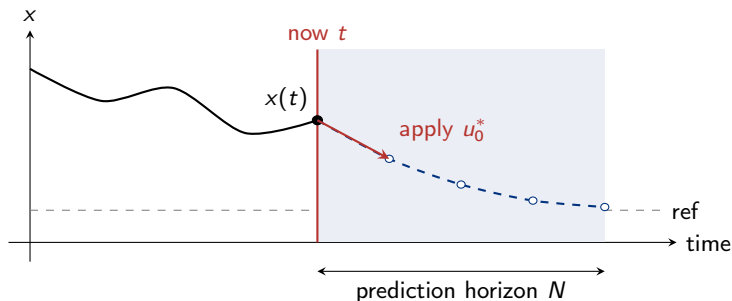
Constraints change everything

With constraints $x_k \in \mathcal{X}$, $u_k \in \mathcal{U}$:

- ▶ the DP value function is **no longer quadratic** – there is no closed-form gain K ;
- ▶ naively saturating the LQR input, $u = \text{sat}(-Kx)$, is **suboptimal** and can **destabilize** the system;
- ▶ we must solve a **constrained optimization** that looks ahead over a horizon.

Idea: solve the constrained finite-horizon problem *online*, using the current measured state.

Receding horizon control: the MPC idea



At each time t : solve a finite-horizon constrained problem from $x(t)$, apply only the first input u_0^* , then **shift the horizon** and re-solve at $t + 1$ with the new measurement.

MPC = online, constrained dynamic programming

	handles constraints?	computation
Exact DP	yes (in principle)	offline, intractable (curse of dim.)
LQR	no	offline, closed-form K
MPC	yes	online, solve an OCP each step

- ▶ MPC evaluates the (implicit) optimal feedback **only at the current state** – avoiding DP over the whole state space.
- ▶ It is an **implementable approximation of the DP feedback law** that respects constraints and extends to nonlinear systems.

Summary



Open loop policy

$$\{0, 1, \dots\} \rightarrow \mathcal{U}$$

$$\begin{aligned} \min_{\mathbf{w}} \quad & \sum_{k=0}^{\infty} \ell(\mathbf{x}_k, \mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\ & \mathbf{x}_0 = \mathbf{x}(t) \end{aligned}$$

Control Policy $\pi(x)$

$$\pi : \mathcal{X} \rightarrow \mathcal{U}$$

$$k = N-1, N-2, \dots, 0$$

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \underbrace{\ell_k(x_k, u_k)}_{\text{cost now}} + \underbrace{J_{k+1}^*(f(x_k, u_k))}_{\text{cost-to-go}}$$