

Lecture 6 - Optimization for control - Dynamic Programming and LQR

Dinesh Krishnamoorthy

Department of Engineering Cybernetics, NTNU

Spring 2026

Optimization in Engineering

Design Optimization

Example: Choose which car to buy

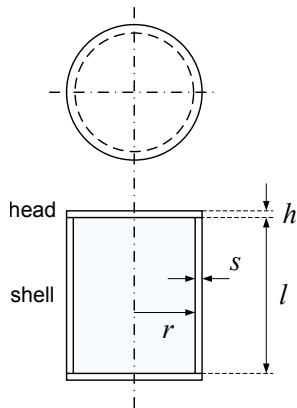


Operational Optimization

Example: Optimize driving



Design optimization



Find:

Minimize:

Subject to:

dimensions

metal volume

constraints on:

Total volume ($\geq$)

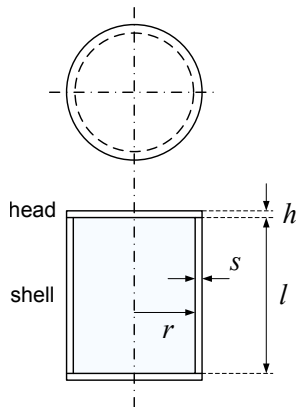
Head thickness ($\geq$)

Shell thickness ($\geq$)

Shell length ($\leq$ & $\geq$)

Outer radius ($\leq$)

Example: airtank design



Find:

$$x = [h, l, r, s]$$

Minimize:

$$2\pi(r + s)^2 h + \pi[(r + s)^2 - r^2]l$$

Subject to:

$$\pi r^2 l \geq 2.12 \cdot 10^7 \text{ cm}^3$$

$$\frac{h}{r} \geq 0.13$$

$$\frac{s}{r} \geq 0.01$$

$$10 \text{ cm} \leq l \leq 750 \text{ cm}$$

$$r + s \leq 150 \text{ cm}$$

Physical relevance: $x \in \mathcal{X} \quad h, l, r, s > 0$

Operational optimization

Static optimization

The model of the process is time independent

- ▶ Common in finance, economic optimization
- ▶ Recall farming example

Dynamic optimization

Model of the process is time dependent

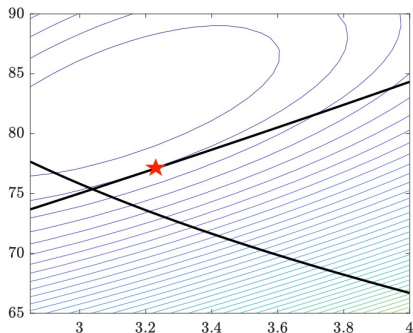
- ▶ The typical case in cybernetics/control engineering
- ▶ The process is a mechanical system, chemical process, electrical/energy system, ... often modelled using differential equations.

Real-time optimization / Quasi-dynamic optimization

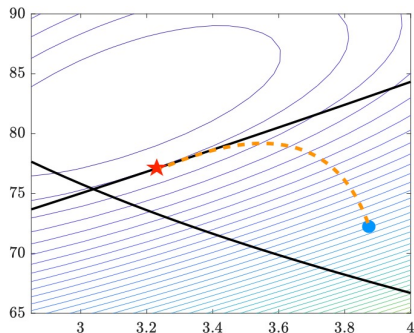
Repeatedly solve static optimization for dynamic systems

Static Vs. Dynamic optimization

Static Optimization



Dynamic Optimization



Quasi-dynamic Optimization / Real-time optimization

- ▶ The process is slowly time-varying, and assumed to be static for the purposes of optimization
- ▶ We take care of the time-variation by re-solving regularly (or when the process has changed sufficiently)
- ▶ Often called Real-time optimization (RTO) in the process industries (typically involving non-linear optimization with economic objectives)

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

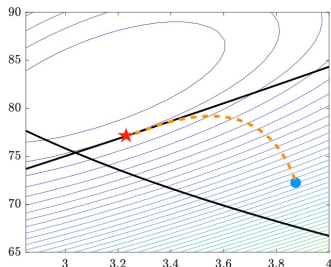
$$\min \sum_{k=0}^{\infty} \ell(x_k, u_k)$$

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?

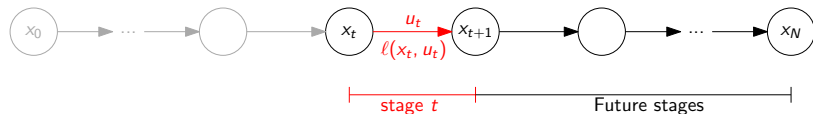
- ▶ A control policy $u = \pi(x)$ is a formal mathematical object that expresses this decision-making logic.

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

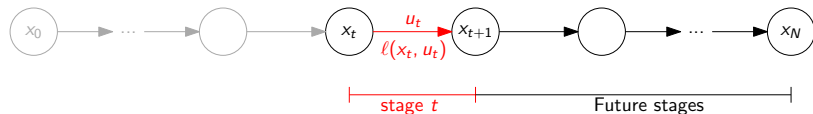
Finding the optimal policy



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

cost now + future costs

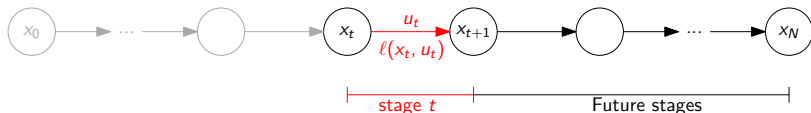
Finding the optimal policy



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?

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

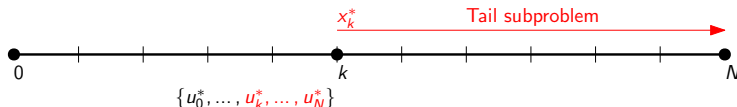
Optimal cost function

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

Optimal policy

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

Principle of optimality



- ▶ Let $\{u_0^*, \dots, u_{N-1}^*\}$ denote the optimal control sequence, with the corresponding optimal state sequence, $\{x_1^*, \dots, x_N^*\}$.
- ▶ Consider the **tail subproblem** at time k , starting at x_k^* , and minimizes over $\{u_k, \dots, u_{N-1}\}$, the “cost-to-go” from k to N

$$\ell_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} \ell_m(x_m, u_m) + \ell_N(x_N)$$

- ▶ Then the tail optimal control sequence $\{u_k^*, \dots, u_{N-1}^*\}$ is optimal for the tail subproblem.

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

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

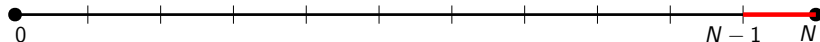
and so on,

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

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

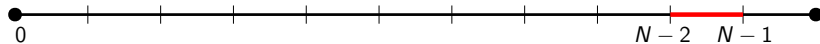
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



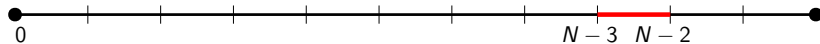
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



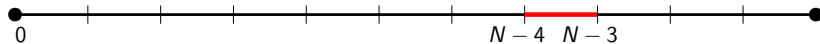
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



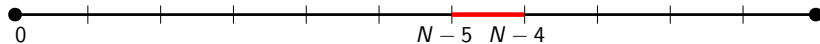
Exact Dynamic Programming (DP)

Idea of Exact DP - Make optimal decision in stages



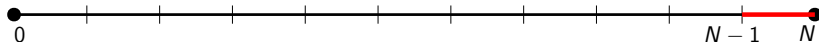
Exact Dynamic Programming (DP)

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

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

Quadratic objective

Often quadratic functions are used to define the objective:

1. Because it is convenient, mathematically
 - ▶ Smooth is good, both for analysis and numerical optimization
 - ▶ Give linear gradients
2. Because it is natural; the effect is often desirable
 - ▶ Tends to ignore small deviations
 - ▶ Tends to punish large deviations

For a linear system: $x_{k+1} = Ax_k + Bu_k$, with quadratic cost, we have

$$\begin{aligned} \min_w \quad & \sum_{k=0}^{\infty} x_k^\top Q x_k + u_k^\top R u_k \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, \dots \end{aligned}$$

Sum of quadratic functions

Given two quadratic functions

$$V_1(u) = u^\top R u, \quad V_2(u) = (Bu - \alpha)^\top P (Bu - \alpha)$$

the sum $V(u) = V_1(u) + V_2(u)$ is also quadratic with

$$V(u) = (u - v)^\top H (u - v) + d$$

$$H := R + B^\top P B$$

$$v := H^{-1} B^\top P \alpha$$

$$d := \alpha^\top (P - P^\top B H^{-1} B^\top P) \alpha$$

Proof

Expand $V(u)$ and equate to $V_1(u) + V_2(u)$

Optimizing Multistage functions

Because of the special structure, we can solve by optimizing a sequence of three 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).

Optimizing Multistage LQ functions

Given x_0 , find the control sequence u_k

$$\sum_{k=0}^{\infty} \underbrace{x_k^\top Q x_k}_{:= \|x_k\|_Q^2} + \underbrace{u_k^\top R u_k}_{:= \|u_k\|_R^2} \quad \text{s.t. } x_{k+1} = Ax_k + Bu_k$$

Finite horizon Multistage LQ problem

$$\min \sum_{k=0}^{N-1} (\|x_k\|_Q^2 + \|u_k\|_R^2) + \|x_N\|_P^2 \quad \text{s.t. } x_{k+1} = Ax_k + Bu_k$$

Dynamic programming solution

The problem to be solved at the last stage $N - 1$

$$\min_{u_{N-1}, x_N} \|x_{N-1}\|_Q^2 + \|u_{N-1}\|_R^2 + \|x_N\|_{P_f}^2 \quad \text{s.t. } x_N = Ax_{N-1} + Bu_{N-1}$$

$$\begin{aligned} \min_{u_{N-1}, x_N} & \|x_{N-1}\|_Q^2 + \underbrace{\|u_{N-1}\|_R^2 + \|Ax_{N-1} + Bu_{N-1}\|_{P_f}^2}_{= \|x_{N-1}\|_Q^2 + \|u_{N-1} - v\|_H^2 + d} \\ & = \|x_{N-1}\|_Q^2 + \|u_{N-1} - v\|_H^2 + d \end{aligned}$$

$$H := R + B^\top P_f B$$

$$v := -H^{-1} B^\top P_f A x_{N-1}$$

$$d := x_{N-1}^\top \left(A^\top P_f A - A^\top P_f^\top B H^{-1} B^\top P_f A \right) x_{N-1}$$

Proof: Set $V_1(u) = \|u_{N-1}\|_R^2$ and $V_2(u) = \|Ax_{N-1} + Bu_{N-1}\|_{P_f}^2$ with $\alpha = -Ax_{N-1}$

From this it is clear that, optimal input for $u_{N-1} = v = -\underbrace{H^{-1} B^\top P_f A}_{:= K_{N-1}} x_{N-1}$

with optimal cost $= \|x_{N-1}\|_Q^2 + d$

Dynamic Programming solution

From this it is clear that, optimal input for $u_{N-1} = v = -\underbrace{H^{-1}B^\top P_f A}_{:=K_{N-1}} x_{N-1}$

with optimal cost $= \|x_{N-1}\|_Q^2 + d$

Summarizing

For the last stage, we have the following feedback law, and the corresponding optimal cost

$$u_{N-1}(x_{N-1}) = -K_{N-1}x_{N-1}$$

$$V_{N-1}(x_{N-1}) = \|x_{N-1}\|_Q^2 + d = \|x_{N-1}\|_{P_{N-1}}^2$$

where $P_{N-1} := Q + A^\top P_f A - A^\top P_f B H^{-1} B^\top P_f A$

Cost-to-go

$V_{N-1}(x_{N-1})$ denotes the cost-to go from state x_{N-1} to x_N .

Dynamic programming solution

The problem to be solved at the second last stage $N - 2$

$$\min_{u_{N-2}, x_{N-1}} \|x_{N-2}\|_Q^2 + \|u_{N-2}\|_R^2 + \|x_{N-1}\|_{P_{N-1}}^2 \quad \text{s.t. } x_{N-1} = Ax_{N-2} + Bu_{N-2}$$

$$\min_{u_{N-2}, x_{N-1}} \|x_{N-2}\|_Q^2 + \underbrace{\|u_{N-2}\|_R^2 + \|Ax_{N-2} + Bu_{N-2}\|_{P_{N-1}}^2}_{=\|u_{N-2} - v\|_H^2 + d}$$

$$H := R + B^\top P_{N-1} B, \quad v := -H^{-1} B^\top P_{N-1} A x_{N-2}$$

$$d := x_{N-2}^\top \left(A^\top P_{N-1} A - A^\top P_{N-1} B H^{-1} B^\top P_{N-1} A \right) x_{N-2}$$

For the last stage, we have the following feedback law, and the corresponding optimal cost

$$u_{N-2}(x_{N-2}) = v = -H^{-1} B^\top P_{N-1} A x_{N-2} = -K_{N-2} x_{N-2}$$

$$V_{N-2}(x_{N-2}) = \|x_{N-2}\|_P^2$$

where $P_{N-2} := Q + A^\top P_{N-1} A - A^\top P_{N-1} B H^{-1} B^\top P_{N-1} A$

Dynamic programming solution

Backward Riccati iteration

The recursion from P_{N-1} to P_{N-2} is known as the backward Riccati iteration, and can be generalized as:

$$P_{k-1} := Q + A^\top P_k A - A^\top P_k B H^{-1} B^\top P_k A, \quad k = N, N-1, \dots, 0$$

with the terminal condition that $P_N = P_f$

Backward DP recursion

The optimal control policy at each stage is given by

$$u_k = -K_k x_k$$

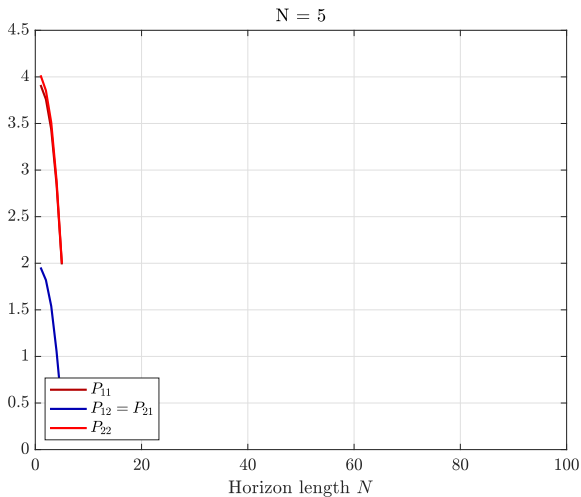
where $K_k = (R + B^\top P_{k+1} B)^{-1} B^\top P_{k+1} A$

Optimal infinite horizon problem

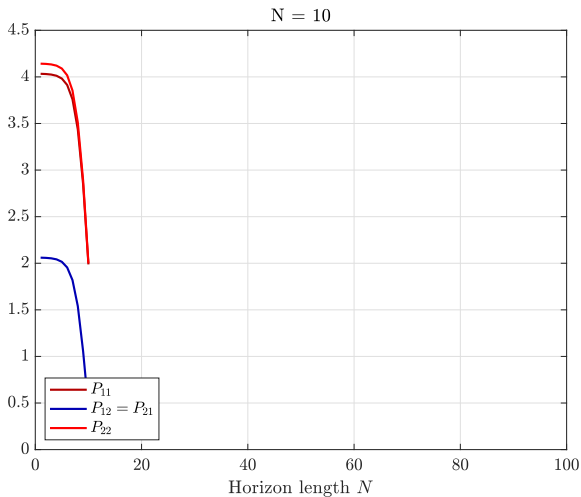
The optimal infinite horizon control policy $u_k = -K x_k$ is given

$$\begin{aligned} K_k &= (R + B^\top P B)^{-1} B^\top P A \\ P &= Q + A^\top P A - A^\top P B H^{-1} B^\top P A \end{aligned}$$

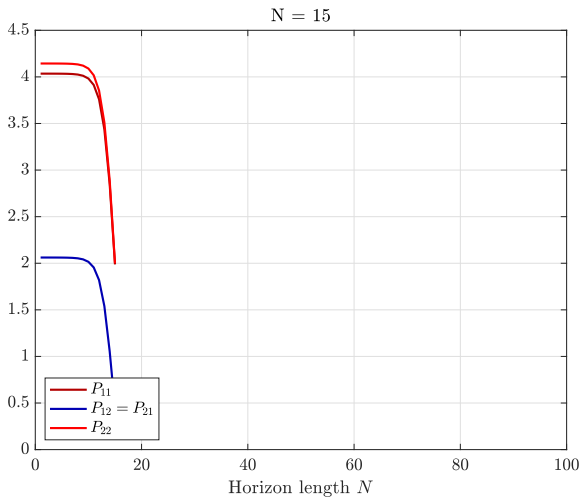
Example: Infinite horizon DP recursion



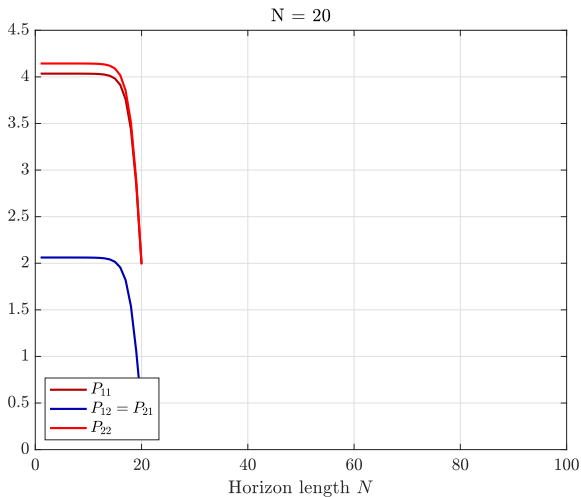
Example: Infinite horizon DP recursion



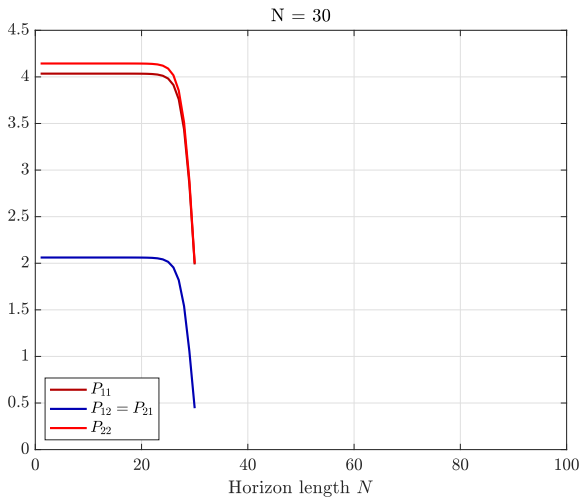
Example: Infinite horizon DP recursion



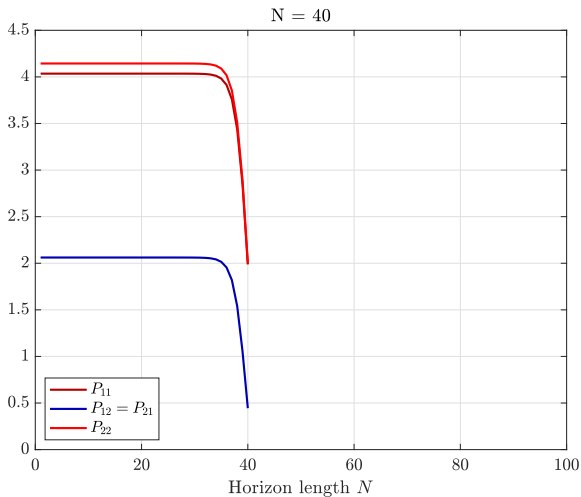
Example: Infinite horizon DP recursion



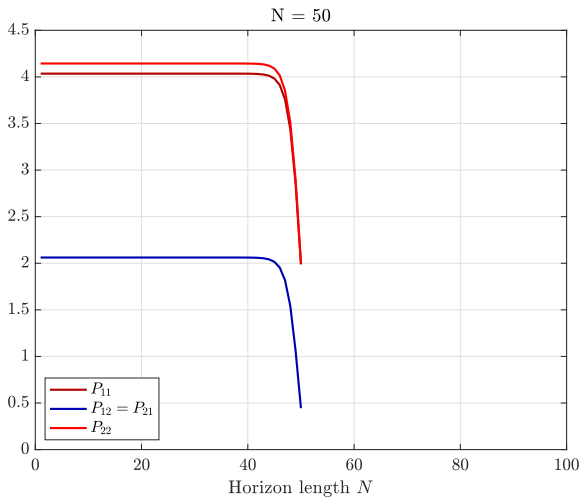
Example: Infinite horizon DP recursion



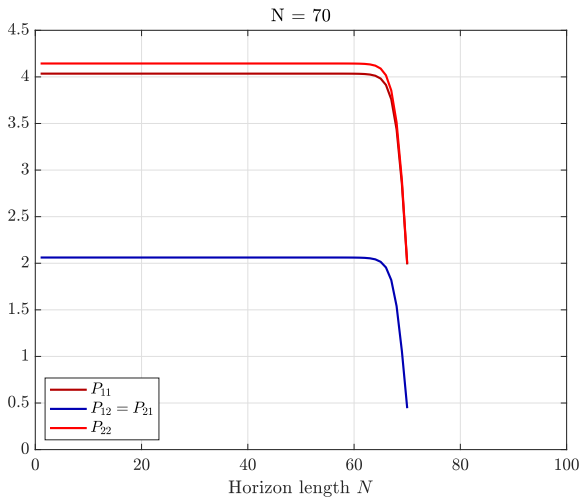
Example: Infinite horizon DP recursion



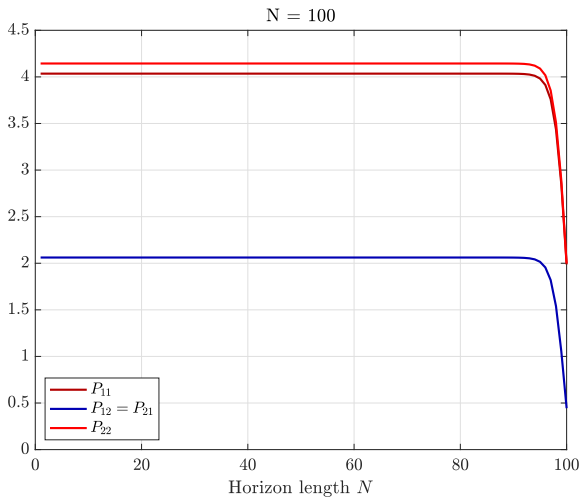
Example: Infinite horizon DP recursion



Example: Infinite horizon DP recursion



Example: Infinite horizon DP recursion



Steady-state ($P_{k+1} = P_k$) DP recursion is infinite horizon solution.

LQR controller

The LQR control law

$$\mathbf{u} = -\mathbf{K}\mathbf{x}$$

and the LQR cost

$$V_{LQR} = \mathbf{x}^\top \mathbf{P} \mathbf{x}$$

for a discrete time system

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$

that optimizes the infinite horizon cost

$$V_{LQR} = \min \sum_{k=0}^{\infty} \mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k$$

is given as the solution to the Discrete Algebraic Riccati Equation (DARE)

$$\mathbf{P} = \mathbf{Q} + \mathbf{A}^\top \mathbf{P} \mathbf{A} - \mathbf{A}^\top \mathbf{P} \mathbf{B} \mathbf{H}^{-1} \mathbf{B}^\top \mathbf{P} \mathbf{A}$$

which gives the LQR gain

$$\mathbf{K} = (\mathbf{R} + \mathbf{B}^\top \mathbf{P} \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{P} \mathbf{A}$$

Example

$$\begin{aligned} \min \quad & \sum_{k=0}^{\infty} x_k^2 + ru^2 \\ \text{s.t.} \quad & x_{k+1} = 1.2x_k + u_k \end{aligned}$$

Summary: Dynamic Programming and LQR

- **Optimization in control:** design a policy to minimize long-term cost for a dynamical system

$$\min_{\{u_k\}} \sum_{k=0}^{\infty} \ell(x_k, u_k) \quad \text{s.t.} \quad x_{k+1} = f(x_k, u_k), \quad x_0 \text{ given.}$$

- **Policies:** open-loop plans (u_k as a time sequence) vs. feedback policies ($u_k = \pi(x_k)$).
- **Dynamic Programming (exact):** principle of optimality $\Rightarrow$ backward recursion for cost-to-go

$$J_k^*(x) = \min_{u \in U_k(x)} [\ell_k(x, u) + J_{k+1}^*(f(x, u))], \quad J_N^*(x) = \ell_N(x),$$

and a forward pass to generate the optimal trajectory.

- **Limitation:** *curse of dimensionality* — exact DP scales exponentially with state dimension.
- **LQR as a tractable DP case:** linear dynamics + quadratic costs $\Rightarrow$ quadratic value function, optimal linear feedback $u_k = -K_k x_k$, and Riccati recursion / DARE for infinite horizon.

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 $\pi(x)$

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

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

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