Lecture 5: Optimal Control Introduction and Unconstrained Linear Quadratic Control

I. Introduction

- **Initial Remarks: Concept of Optimal Control and General Formulation**

Optimal Control Introduction

Discrete-time **optimal control** is concerned with **choosing an optimal input sequence** (as measured by some objective function) $U_{0\to N-1}=[u_0^{\mathrm{T}},\ u_1^{\mathrm{T}},\ \ldots]^{\mathrm{T}}$, **over a finite or infinite time horizon**, in order to apply it to a system with a given initial state $x(0)$.

The objective, or cost, function is often defined as a sum of **stage costs** $q(x_k,u_k)$ and, when the horizon has a finite length N, a **terminal cost** $p(x_n)$:

$$J_{0\to N}(x_0,U_{0\to N\to 1})=p(x_N)+\sum_{k=0}^{N-1}q(x_k,u_k)$$

The states $\{x_N\}_{k=0}^{N}$ must satisfy the system dynamics:

$$x_{k+1}=g(x_k,u_k),\ k=0,\ \ldots,N-1$$
$$x_0=x(0)$$

and there may be state and/or input constraints

$$h(x_k,u_k)\le 0,\ k=0,\ \ldots,N-1$$

In the finite horizon case, there may also be a **constraint that the final state** x_N lies in a set $\mathcal{X}_f$:

$$x_N\in\mathcal{X}_f$$

Formal General Problem Formulation

Consider the nonlinear time-invariant system:

$$x(t+1)=g(x(t),u(t))$$

subject to the constraints:

$$h(x(t),u(t))\le 0,\forall x\ge 0$$

With $x(t)\in\mathbb{R}^n$ and $u(t)\in\mathbb{R}^m$ are state and input vectors. Assume the system's initial condition is $g(0,0)=0$ and $h(0,0)\le 0$.

Consider the following objective or cost function

$$J_{0\to N}(x_0,U_{0\to N\to 1})=p(x_N)+\sum_{k=0}^{N-1}q(x_k,u_k)$$

where:

N is the time horizon, $x_{k+1}=g(x_k,u_k),k=0,\ \ldots,N-1$ and $x_0=x(0)$

$U_{0\to N-1}=[u_0^{\mathrm{T}},\ u_1^{\mathrm{T}},\ \ldots]^{\mathrm{T}}\in\mathbb{R}^s,s=mN$

$q(x_k,u_k)$ and $p(x_N)$ are stage cost and terminal cost, respectively

Consider the **C**onstrained **F**inite **T**ime **O**ptimal **C**ontrol (CFTOC) problem:

$$J_0^*(x(0)) = \min_{U_{0\to N-1}} J_{0\to N}(x(0), U_{0\to N-1})$$

$$\begin{aligned} \text{subj. to.} \quad & x_{k+1} = g(x_k, u_k) \quad k = 0, \ldots, N-1 \\ & h(x_k, u_k) \le 0 \qquad k = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned}$$

where

$\mathcal{X}_f \subset \mathbb{R}^n$ is a terminal region, $\mathcal{X}_{0\to N} \subset \mathbb{R}^n$ is the set of feasible initial conditions $x(0)$

The optimal cost $J_0^*(x(0))$ is called the **value function** and denotes $U_{0\to N}^*$ one of the minima

In addition, we assume that there exists a minimum.

Optimal Control Objectives (Topics)

1. Finite Time Solution
   a general nonlinear programming problem (batch approach)
   recursively by invoking Bellman's Principle of Optimality (recursive approach)
   discuss in detail the linear system case
2. Infinite Time Solution
   if a solution exists as $N \to \infty$
   the properties of this solution
   approximate of the solution by using a receding horizon technique
3. Uncertainty
   how to extend the problem description and consider uncertainty

- **Solution Approaches: Batch Approach**

Solution via Batch Approach

Rollout the equality constraints from system constraints (i.e. system dynamics), then the optimal control problem can be written as:

$$J_0^*(x(0)) = \min_{\substack{U_{0\to N-1} \\ x_1 \ldots, x_N}} J_{0\to N}(x(0), U_{0\to N-1})$$

$$\begin{aligned} \text{subj. to.} \quad & x_1 = g(x_0, u_0) \\ & x_2 = g(x_1, u_1) \\ & \vdots \\ & x_N = g(x_{N-1}, u_{N-1}) \\ & h(x_k, u_k) \le 0 \qquad k = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned}$$

which is a general Non-linear Programming (NLP) with variables $u_0,\dots,u_{N-1}$ and $x_1,\dots,x_N$.

Note:

1. Here x_0 represents the computed value that has to be constrained, while $x(0)$ is the measured value that is given. Though they are the same value, they still have different meanings.
2. In this course, we will "trust" a general-purpose nonlinear solver
3. More efficient solvers for tailored problem
4. Sometimes still some work to do in writing (rollout) the inequality constraints $h(x_k,u_k)\le 0$ (e.g. the cases of obstacle avoidance)

- **Solution Approaches: Recursive Approach**

Theorem (Principle of Optimality):

For a trajectory $x_0,x_1^*,\dots,x_N^*$ to be optimal, the trajectory starting from any intermediate point x_j^*, i.e. $x_j^*,x_{j+1}^*,\dots,x_N^*, 0\le j\le N-1$, must be optimal.

A simple application intuition: Suppose that the fastest route from Los Angeles to Boston passes through Chicago. The principle of optimality formalizes the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston.

Define the cost from j to N (also called the j step cost-to-go):

$$J_{j\to N}(x_j,u_j,u_{j+1},\dots,u_{N-1}) = p(x_N)+\sum_{k=j}^{N-1} q(x_k,u_k)$$

Then the optimal j step cost to go $J_{j\to N}^*$ is:

$$J_{j\to N}^*(x_j) = \min_{u_j,u_{j+1},\dots,u_{N-1}} J_{j\to N}(x_j,u_j,u_{j+1},\dots,u_{N-1})$$

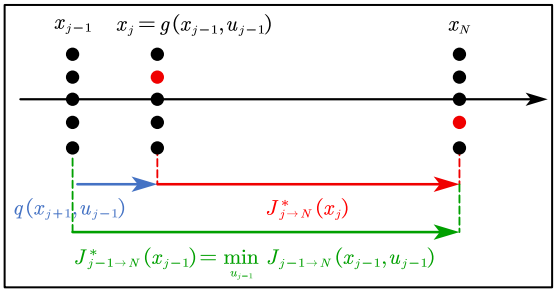
$$\begin{aligned}\text{subj. to.}\quad & x_{k+1}=g(x_k,u_k) \quad k=j,\dots,N-1\\ & h(x_k,u_k)\le 0 \qquad k=j,\dots,N-1\\ & x_N\in\mathcal{X}_f\end{aligned}$$

Note: It can be observed that $J_{j\to N}^*(x_j)$ depends only on the initial state x_j

Then, by the **principle of optimality** the cost $J_{j-1\to N}^*$ can be found by solving:

$$J_{j-1\to N}^*(x_{j-1}) = \min_{u_{j-1}} \overbrace{\underbrace{q(x_{j-1},u_{j-1})}_{\text{stage cost}} + \underbrace{J_{j\to N}^*(x_j)}_{j \text{ step optimal cost to go}}}^{j-1 \text{ step cost to go}}$$

$$\begin{aligned}\text{subj. to.}\quad & x_j = g(x_{j-1},u_{j-1})\\ & h(x_{j-1},u_{j-1})\le 0\\ & x_j\in\mathcal{X}_{j\to N}\end{aligned}$$



Note:

1. The **only decision** variable is u_{j-1}, because the inputs $u_j^*,u_{j+1}^*,\dots,u_N^*$ **have already been selected optimally** to yield the optimal cost-to-go $J_{j\to N}^*(x_j)$.

2. in $J^*_{j\to N}(x_j)$, the states x_j can be replaced by $g(x_{j-1},u_{j-1})$, and the set $\mathcal{X}_{j\to N}$ is the set of states x_j for which the above problem is feasible. The discussion of this set will also be covered in the future.

Dynamic Programming for Solving Optimal Control

With the above formulation, the following (recursive) dynamic programming algorithm can be used to compute the optimal control law.

Starting from the final state:

$$J^*_{N\to N} = p(x_N)$$
$$x_{N\to N} = \mathcal{X}_f$$

Then recursively propagate backward:

$$J^*_{N-1\to N}(x_{N-1}) = \min_{u_{N-1}} \; q(x_{N-1},u_{N-1}) + J^*_{N\to N}(g(x_{N-1},u_{N-1}))$$
$$\text{subj. to} \;\; h(x_{N-1},u_{N-1}) \le 0$$
$$g(x_{N-1},u_{N-1}) \in \mathcal{X}_{N\to N}$$
$$\vdots$$
$$J^*_{0\to N}(x_0) = \min_{u_0} \; q(x_0,u_0) + J^*_{1\to N}(g(x_0,u_0))$$
$$\text{subj. to} \;\; h(x_0,u_0) \le 0$$
$$g(x_0,u_0) \in \mathcal{X}_{N\to N}$$
$$x_0 = x(0)$$

Note:

1. DP algorithm is appealing because at each step j only u_j is computed.
2. However, it means we need to construct the optimal cost-to-go $J^*_{N-j}(x_j)$, i.e. a function defined over $\mathcal{X}_{j\to N}$. In a few special cases (e.g. linear system, quadratic cost), we know the type of function and we can find it efficiently.
3. For general cases, use "brute force" approach. Construct $J_{j-1\to N}$ by griding the set $\mathcal{X}_{j-1\to N}$ and computing the optimal cost-to-go function on each grid point.
4. A nonlinear feedback (time-varying) control law is implicitly defined:

$$u^*_j(x_j) = \operatorname*{argmin}_{u_j} \; q(x_j,u_j) + J^*_{j+1\to N}(g(x_j,u_j))$$
$$\text{subj. to } h(x_j,u_j) \le 0$$
$$g(x_j,u_j) \in \mathcal{X}_{j-1\to N}$$

Bellman Equation

For the sake of simplicity, we will use the following shorter notation

$$J^*_j(x_j) = J^*_{j\to N}(x_j), \;\; j=0,\ldots,N \qquad \mathcal{X}_j = \mathcal{X}_{j\to N}, \;\; j=0,\ldots,N$$
$$J^*_\infty(x_j) = J^*_{0\to\infty}(x_j) \qquad \mathcal{X}_\infty = \mathcal{X}_{0\to\infty} \qquad U_0 = U_{0\to N} - 1$$

Assume that a solution exists for $N \to \infty$, then we obtain the **Bellman Equation**:

$$J_\infty^*(x) = \min_u \; q(x,u) + J_\infty^*(g(x,u))$$

$$\text{subj. to. } h(x,u) \leq 0$$

$$g(x,u) \in \mathcal{X}_\infty$$

II. Finite Horizon

- **Initial Remarks: Unconstrained Finite Horizon Linear Quadratic Optimal Control**

In this section, only linear discrete-time time-invariant systems:

$$x(k+1) = Ax(k) + Bu(k)$$

And quadratic cost functions:

$$J_0(x_0, U) = x_N^{\mathrm{T}} P x_N + \sum_{k=0}^{N-1} (x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k)$$

are considered, and we consider only the problem of **regulating the state to the origin, without state or input constraints over a finite horizon.** To be more specific, our goal is to find a sequence of input $U_{0 \to N-1} = [u_0^{\mathrm{T}}, \; \ldots, \; u_{N-1}^{\mathrm{T}}]^{\mathrm{T}}$ that minimizes the objective function:

$$J_0^*(x(0)) = \min_{U_0 \to N-1} \; x_N^{\mathrm{T}} P x_N + \sum_{k=0}^{N-1} (x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k)$$

$$\text{subj. to. } x_{k+1} = Ax_k + Bu_k$$

$$x_0 = x(0)$$

where:

$P \succeq 0$, with $P = P^{\mathrm{T}}$, is the terminal weight

$Q \succeq 0$, with $Q = Q^{\mathrm{T}}$, is the state weight

$R \succ 0$, with $R = R^{\mathrm{T}}$, is the input weight

N is the horizon length and $x(0)$ is the current state, whereas $x_0, \ldots, x_N$ and $u_0, \ldots, u_{N-1}$ are **optimization variables** that are constrained to obey the system dynamics and the initial condition.

The two most common solution approaches will be described here

1. Batch Approach, which yields a series of numerical values for the input
2. Recursive Approach, which uses Dynamic Programming to compute control policies or laws, i.e. functions that describe how the control decisions depend on the system states.

- **Batch Approach for Unconstrained LQC**

Derivation Details

The batch solution explicitly represents all future states x_k in terms of the initial condition x_0 and the inputs $u_0, \ldots, u_{N-1}$ (i.e. doing the rollout of system dynamics).

Starting with $x_0 = x(0)$, we have $x_1 = Ax(0) + Bu_0$, $x_2 = Ax_1 + Bu_1 = A^2 x(0) + ABu_0 + Bu_1$ and so on. Continuing up to x_N we obtain:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

The equation above can also be represented as:

$$\mathcal{X} = \mathcal{S}^x x(0) + \mathcal{S}^u U$$

Defining $\bar{Q} = \text{blockdiag}(Q, \ldots, Q, P)$ and $\bar{R} = \text{blockdiag}(R, \ldots, R)$, then the finite horizon cost function can be written as:

$$J_0(x(0), U) = \mathcal{X}^{\mathrm{T}} \bar{Q} \mathcal{X} + U^{\mathrm{T}} \bar{R} U$$

Eliminating $\mathcal{X}$ by substituting the rollout dynamics we can express the cost function as:

$$\begin{aligned} J_0(x(0), U) &= (\mathcal{S}^x x(0) + \mathcal{S}^u U)^{\mathrm{T}} \bar{Q} (\mathcal{S}^x x(0) + \mathcal{S}^u U) + U^{\mathrm{T}} \bar{R} U \\ &= U^{\mathrm{T}} H U + 2x(0)^{\mathrm{T}} F U + x(0)^{\mathrm{T}} (\mathcal{S}^x)^{\mathrm{T}} \bar{Q} \mathcal{S}^x x(0) \end{aligned}$$

where $H = (\mathcal{S}^u)^{\mathrm{T}} \bar{Q} \mathcal{S}^u + \bar{R}$ and $F = (\mathcal{S}^x)^{\mathrm{T}} \bar{Q} \mathcal{S}^u$, and note that $H \succ 0$ since $R \succ 0$ and $(\mathcal{S}^u)^{\mathrm{T}} \bar{Q} \mathcal{S}^u \succeq 0$

Since the problem is unconstrained and $J_0(x(0), U)$ is a positive definite quadratic function of U, we can solve for the optimal input U^* by setting the gradient with respect to U to zero, i.e.:

$$\nabla_U J_0(x(0), U) = 2HU + 2F^{\mathrm{T}} x(0) = 0$$

$$\Rightarrow U^*(x(0)) = -H^{-1} F^{\mathrm{T}} x(0) = -\left((\mathcal{S}^u)^{\mathrm{T}} \bar{Q} \mathcal{S}^u + \bar{R}\right)^{-1} (\mathcal{S}^u)^{\mathrm{T}} \bar{Q} \mathcal{S}^x x(0)$$

which is a linear function of the initial state $x(0)$. Note that H^{-1} always exists, since $H \succ 0$ and therefore has full rank. Also note that since the cost $J_0(x(0), U)$ is a strictly convex quadratic function of U, the solution above is unique.

The optimal cost can be shown (by back-substitution) to be:

$$\begin{aligned} J_0^*(x(0)) &= U^{*\mathrm{T}} H U^* + 2x(0)^{\mathrm{T}} F U^* + x(0)^{\mathrm{T}} (\mathcal{S}^x)^{\mathrm{T}} \bar{Q} \mathcal{S}^x x(0) \\ &= -x(0)^{\mathrm{T}} F H F x(0) + x(0)^{\mathrm{T}} (\mathcal{S}^x)^{\mathrm{T}} \bar{Q} \mathcal{S}^x x(0) \\ &= x(0)^{\mathrm{T}} \left[(\mathcal{S}^x)^{\mathrm{T}} \bar{Q} \mathcal{S}^x x - (\mathcal{S}^x)^{\mathrm{T}} \bar{Q} \mathcal{S}^u \left((\mathcal{S}^u)^{\mathrm{T}} \bar{Q} \mathcal{S}^u + \bar{R} \right)^{-1} (\mathcal{S}^u)^{\mathrm{T}} \bar{Q} \mathcal{S}^x \right] x(0) \end{aligned}$$

which is a quadratic function of the initial state $x(0)$

Note:

Is it a good idea to roll out dynamics and compute lots of the matrix power? In fact, it is not always good in practice since generally the system dynamics matrix A would have eigenvalues very close to the unit circle. Therefore, in the end, the numerical error would accumulate in the process of calculating the matrix power and hinder the solving process.

Summary: Important Results

1. The problem is unconstrained. **A unique** solution exists and is **always feasible**

2. The solution is solved by setting the gradient to zero and we get the solution a linear function of the initial state $x(0)$, i.e. a **linear, open-loop controller** function of the initial state:

$$U_0^*(x(0)) = Kx(0) \quad \text{which implies} \quad \begin{aligned} u^*(0)\,(x(0)) &= K_0 x(0), \\ &\vdots \\ u^*(N-1)\,(x(0)) &= K_{N-1} x(0) \end{aligned}$$

3. The optimal cost is a **positive definite quadratic function** of the initial state $x(0)$
4. If there are state or input constraints, solving this problem by matrix inversion is not guaranteed to result in a feasible input sequence

- **Recursive Approach for Unconstrained LQC**

Derivation Details

We can also use dynamic programming to solve the same problem. Same with the general case, we define "j-step optimal cost to go" as the optimal cost attainable for the step j problem:

$$J_j^*(x(j)) = \min_{U_{j\to N-1}} x_N^{\mathrm{T}} P x_N + \sum_{k=j}^{N-1} (x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k)$$
$$\text{subj. to.} \quad x_{k+1} = Ax_k + Bu_k$$
$$x_j = x(j)$$

In other words, this is the minimum cost attainable for the remainder of the horizon after step j.

1. Starting from optimal 1-step solution

From the last time step, consider the 1-step problem (solved at time $N-1$)

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} x_N^{\mathrm{T}} P x_N + x_{N-1}^{\mathrm{T}} Q x_{N-1} + u_{N-1}^{\mathrm{T}} R u_{N-1}$$
$$\text{subj. to.} \quad x_N = Ax_{N-1} + Bu_{N-1}$$
$$P_N = P$$

where we introduced the notation P_j to express the **optimal cost-to-go** $x_j^{\mathrm{T}} P x_j$. In particular, here we have final state cost $P_N = P$. Substituting the system dynamics and final cost into the objective to transform into unconstrained optimization we get:

$$\begin{aligned} J_{N-1}^*(x_{N-1}) &= \min_{u_{N-1}} (Ax_{N-1} + Bu_{N-1})^{\mathrm{T}} P_N (Ax_{N-1} + Bu_{N-1}) + x_{N-1}^{\mathrm{T}} Q x_{N-1} + u_{N-1}^{\mathrm{T}} R u_{N-1} \\ &= \min_{u_{N-1}} x_{N-1}^{\mathrm{T}} (A^{\mathrm{T}} P_N A + Q) x_{N-1} + u_{N-1}^{\mathrm{T}} (B^{\mathrm{T}} P_N B + R) u_{N-1} + 2x_{N-1}^{\mathrm{T}} A^{\mathrm{T}} P_N B u_{N-1} \end{aligned}$$

This objective is quadratic about u_{N-1}, unconstrained, and $(B^{\mathrm{T}} P_N B + R) \succ 0$ since $B^{\mathrm{T}} P_N B \succeq 0$ and $R \succ 0$, hence we can solve by setting the gradient to zero:

$$2(B^{\mathrm{T}} P_N B + R) u_{N-1} + 2B^{\mathrm{T}} P_N A x_{N-1} = 0$$

Hence we get our optimal 1-step solution:

$$\begin{aligned} u_{N-1}^* &= -(B^{\mathrm{T}} P_N B + R)^{-1} B^{\mathrm{T}} P_N A x_{N-1} \\ &= F_{N-1} x_{N-1} \end{aligned}$$

And the corresponding 1-step optimal cost to go:

$$J_{N-1}^*(x_{N-1}) = x_{N-1}^{\mathrm{T}} P_{N-1} x_{N-1}$$

where

$$P_{N-1} = A^{\mathrm{T}} P_N A + Q - A^{\mathrm{T}} P_N B (B^{\mathrm{T}} P_N B + R)^{-1} B^{\mathrm{T}} P_N A$$

2. Consider the 2-step problem

Recall the **principle of optimality:** For any solution for steps 0 to N to be optimal, any solution for steps j to N with $j \geq 0$, taken from the 0 to N solution, must itself be optimal for the j to N problem.

Now we have the last step optimal, we then go backward recursively. Therefore, we solve the 2-step problem, i.e. problem posed at time $N-2$ and simplified the expression **given the 1-step solution.**

$$J_{N-2}^*(x_{N-2}) = \min_{u_{N-1}, u_{N-2}} x_N^{\mathrm{T}} P x_N + \sum_{k=N-2}^{N-1} (x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k)$$

$$\text{subj. to. } x_{k+1} = A x_k + B u_k \quad k = N-2, N-1$$

$$\xRightarrow[\text{Optimality}]{\text{Principle of}} J_{N-2}^*(x_{N-2}) = \min_{u_{N-2}} x_{N-2}^{\mathrm{T}} Q x_{N-2} + u_{N-2}^{\mathrm{T}} R u_{N-2} + J_{N-1}^*(x_{N-1})$$

$$= \min_{u_{N-2}} x_{N-2}^{\mathrm{T}} Q x_{N-2} + u_{N-2}^{\mathrm{T}} R u_{N-2} + x_{N-1}^{\mathrm{T}} P_{N-1} x_{N-1}$$

$$\text{subj. to. } x_{k+1} = A x_k + B u_k \quad k = N-2$$

Again, the objective has exactly the same structure as 1-step solution, and we can solve by setting the gradient to zero to get the optimal 2-step solution:

$$u_{N-2}^* = -(B^{\mathrm{T}} P_{N-1} B + R)^{-1} B^{\mathrm{T}} P_{N-1} A x_{N-2}$$

$$= F_{N-2} x_{N-2}$$

And the corresponding 1-step optimal cost to go:

$$J_{N-1}^*(x_{N-1}) = x_{N-1}^{\mathrm{T}} P_{N-1} x_{N-1}$$

where

$$P_{N-2} = A^{\mathrm{T}} P_{N-1} A + Q - A^{\mathrm{T}} P_{N-1} B (B^{\mathrm{T}} P_{N-1} B + R)^{-1} B^{\mathrm{T}} P_{N-1} A$$

3. Recursion to k-step solution

With 1-step and 2-step solutions, now we recognize the recursion for P_j and u_j^*, $j = N-1, \ldots, 0$ and for any given time step $k = 1, \ldots, N$ in the horizon, we have:

$$u_k^* = -(B^{\mathrm{T}} P_{k+1} B + R)^{-1} B^{\mathrm{T}} P_{k+1} A x_k$$

$$= F_k x_k$$

where any P_k can be found by recursive evaluation from $P_N = P$, using:

$$P_k = A^{\mathrm{T}} P_{k+1} A + Q - A^{\mathrm{T}} P_{k+1} B (B^{\mathrm{T}} P_{k+1} B + R)^{-1} B^{\mathrm{T}} P_{k+1} A$$

This is called the **Discrete Time Riccati equation** or **Riccati Difference equation (RDE)**. Evaluating down to P_0, we obtain the N-step optimal cost to go:

$$J_0^*(x(0)) = x(0)^{\mathrm{T}} P_0 x(0)$$

Summary: Important Results

1. From the Principle of Optimality, the optimal control policy for any step k **a linear, time-varying state-feedback controller.** Each F_k is computed by using P_{k+1}

$$u_k^* = -(B^{\mathrm{T}} P_{k+1} B + R)^{-1} B^{\mathrm{T}} P_{k+1} A x_k = F_k x_k$$

2. Each P_k is related to P_{k+1} by the Riccati Difference equation

$$P_k = A^{\mathrm{T}} P_{k+1} A + Q - A^{\mathrm{T}} P_{k+1} B (B^{\mathrm{T}} P_{k+1} B + R)^{-1} B^{\mathrm{T}} P_{k+1} A$$

which can be initialized with the given terminal weight $P_N = P$

3. The optimal cost from time step k is quadratic about the state at time step k, i.e.:

$$J_k^*(x_k) = x_k^{\mathrm{T}} P_k x_k$$

4. The problem is unconstrained and therefore is always feasible. When inequality constraints are introduced, then the problem might not be feasible.

- **Comparison and Discussion of Batch and Recursive Approaches**

Fundamental difference:

Batch optimization returns a sequence $U^*(x(0))$ of **open-loop numeric values** depending only on the initial state $x(0)$, while dynamic programming yields **closed-loop feedback policies** $u_k^* = F_k x_k$, $k = 0, \ldots, N-1$ depending on each x_k. In other words, batch approach generates a **linear open loop controller** while dynamic programming generates a **close loop state feedback controller**. In other words, batch approach relies a lot on prediction and does not care about what would happen afterward, while dynamic programming tries to control (fix) based on the result in time.

Notes (Discussions):

1. If the state evolves exactly as modeled (no model error and disturbances), then the sequences of control actions obtained from the two approaches are identical. Since the problem we solve essentially is a quadratic program, which is convex, and no matter in what way we solve, the solution hence is unique. So under no error and disturbance assumption, if we roll out x_k in $u_k^* = F_k x_k$, then the result (value) would be the same as $U^*(x(0))$. **But always keep in mind the philosophy of these two solutions are different!**
2. The recursive solution should be more robust to disturbances and model errors, because if the future states later deviate from their predicted values, the exact optimal input can still be computed. Essentially, it is because the recursive solution gives the state feedback solution (policy).
3. If we want a similar closed-loop state-feedback effect by batch approach, we can consider recalculating the problem at timestep $k = 1, \ldots, N-1$ **with the shrinking horizon** $k \to N$. This approach (redo optimization), in some sense, also motivates the idea of receding horizon control (MPC), also see the discussions about constraint satisfaction below.
4. The recursive approach is computationally more attractive because it breaks the problem down into

single-step problems. For large horizon length, the Hessian in the batch approach, which must be inverted, becomes very large. However, this is not a big deal nowadays.

About constraints:

Without any modification, both solution methods will break down when inequality constraints on x_k or u_k are added.

Notes:

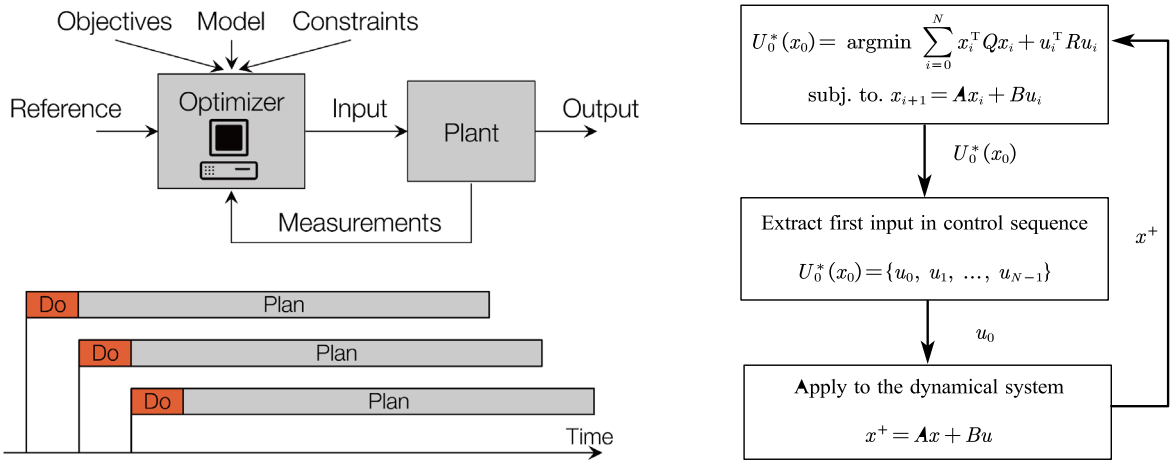
The batch approach is far easier to adapt when constraints are present: just perform a constrained minimization for the current state. The recursive approach is almost impossible to deal with constraints in practice due to problem complexity: it has to compute (find) the cost-to-go under constraints, which is no longer quadratic about the state (we will have deeper discussions in the next lecture). In order to integrate both approaches (find a middle ground), one can choose to do constrained minimization at every time step within the time available, and then use only the first input from the resulting sequence, which amounts to **receding horizon control (model predictive control)**.

III. Receding Horizon Control

- **Concepts of Receding Horizon Control**

In the above discussion, we found that the core idea of the recursive approach is that we have the **feedback policy**, making it more robust, while the core idea of the batch approach is that it is **easier to execute and better at dealing with constraints**.

Receding horizon control is using the way that batch approach does (forward rollout and plan), but only executing the first step of the control sequence, and then redoing the optimization, as the figure shows. It aims to mimic the feedback policy inherently generated from the recursive approach and insert it into the batch approach. Thus it combines the strengths of the two approaches.



For unconstrained systems, it is a constant linear controller and has no difference with what we have discussed before. However, the concept can be extended to much more complex and constrained systems. The important point is that the receding horizon strategy **introduces feedback.**

- **Example: Receding Horizon Control for Simple System**

Consider the **lightly damped, stable** system:

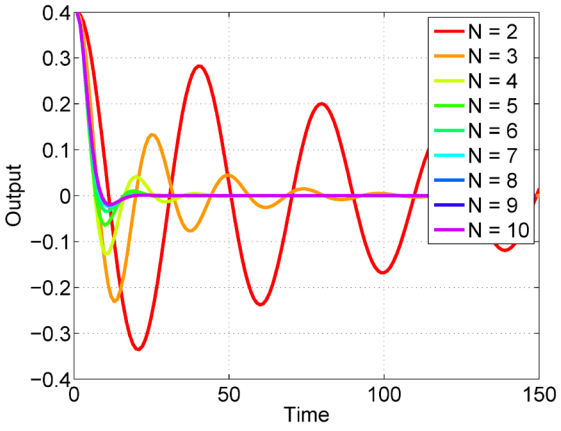
$$G(s)=\frac{\omega^2}{s^2+2\zeta\omega s+\omega^2}$$

where $\omega=1,\zeta=0.01$. We sample at 10 Hz and set $P=Q=I,R=1$. Hence the discrete-time state-space model is:

$$x^+=\begin{bmatrix}1.988 & -0.998\\ 1 & 0\end{bmatrix}x+\begin{bmatrix}0.125\\ 0\end{bmatrix}u$$

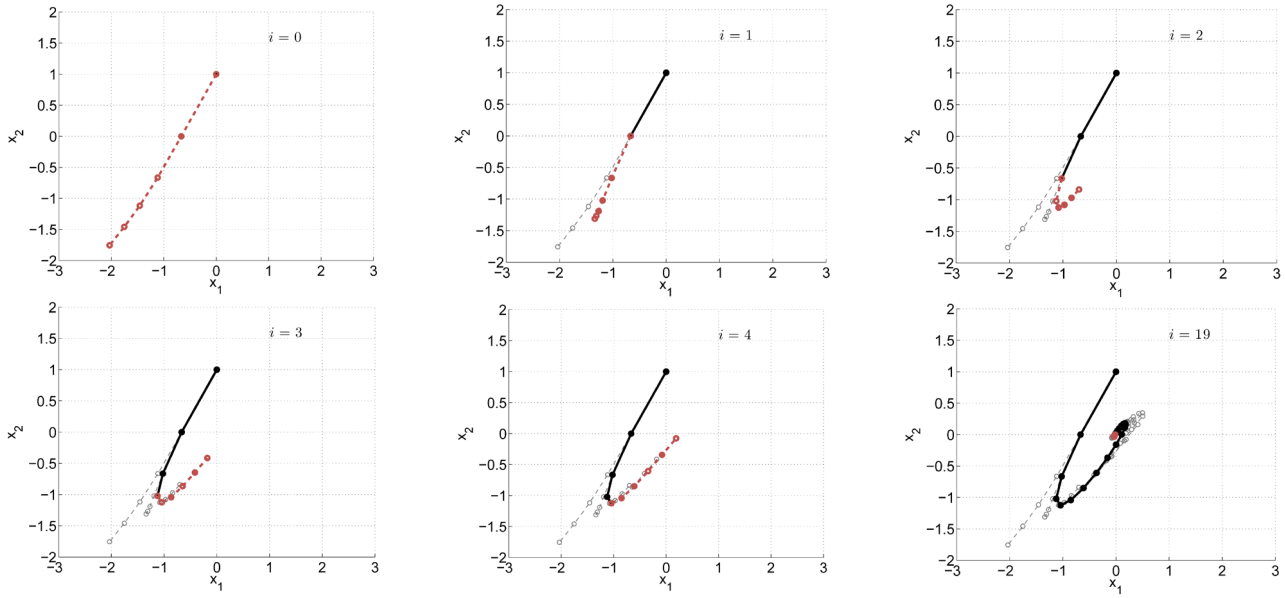
The first decision we must make is to decide the length of horizon we execute.

Rule of thumb: for simple systems, the longer horizon, the better (as the right figure shows). If long-horizon calculation can be done online, then choose the long horizon. It is because for short horizon: Prediction and closed-loop response differ; For long horizon: Prediction and closed-loop match.

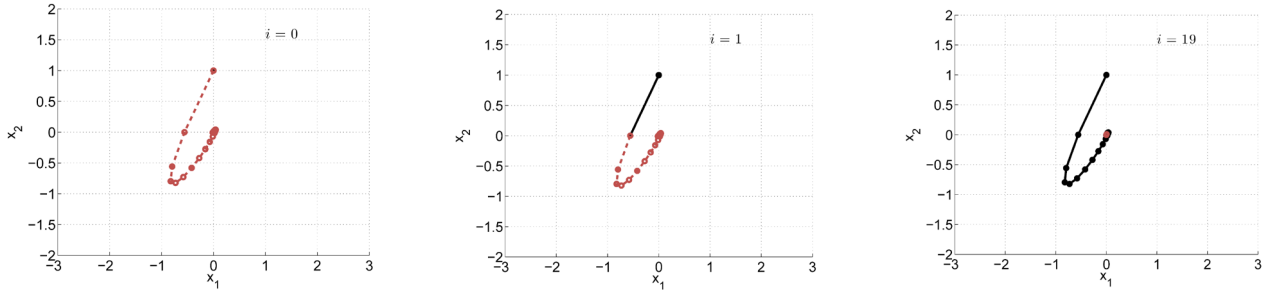


An example of the control effect with different horizons ($N=5$ vs. $N=20$) is shown below:

$N=5$ red: prediction, black: closed-loop response



$N=20$: red: prediction, black: closed-loop response



Note: here, we don't assign the exact terminal cost and terminal set constraint, or the predictions would all be the same for the two cases (also see the section about the choices of terminal weight in finite horizon control below).

- **Example: Receding Horizon Control for Simple System**

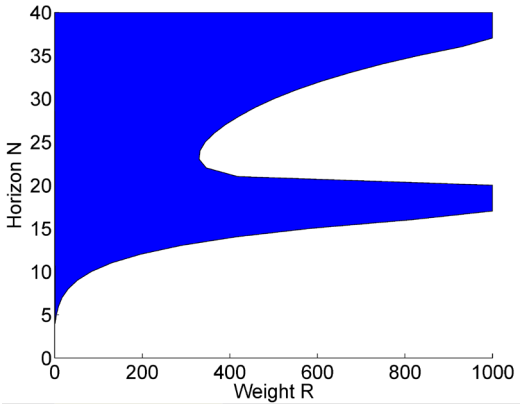
Now consider the system:

$$G(s)=\frac{\omega^2}{s^2+2\zeta\omega s+\omega^2}$$

where $\omega=1,\zeta=-1$ which has been discretized at $1r/\mathrm{s}$ (Note that this system is unstable)

The stability of the system $x^+=(A+BK_{R,N})x$ with different choices of horizon and weight parameters is shown in the figure below, the blue region means the stable region, and the white region means unstable.

Rule of thumb: for tuning parameters, **do not try to tune the horizon at first**. Fix the horizon and tune the weight parameters. From the figure, we can see that if the weight parameter is fixed at 600 and the horizon is increased from 0 to 40, the system would be unstable → stable → unstable → stable, which is a highly undesirable property. It is because even though a longer horizon usually performs better, the **stability in long-horizon is hard to judge**.



IV. Infinite Horizon

- **Infinite Horizon Linear Quadratic Optimal Control: Solution**

In some cases, we may want to solve the same problem with an infinite horizon:

$$J_\infty^*(x(0))=\min_{u(\cdot)}\sum_{k=0}^{\infty}(x_k^{\mathrm{T}}Qx_k+u_k^{\mathrm{T}}Ru_k)$$

$$\text{subj. to.}\quad x_{k+1}=Ax_k+Bu_k\quad k=0,\ldots,\infty$$

$$x_0=x(0)$$

As with the Dynamic Programming approach, the optimal input is of the form:

$$u_k^*=-(B^{\mathrm{T}}P_\infty B+R)^{-1}B^{\mathrm{T}}P_\infty Ax_k=F_\infty x_k$$

and the infinite-horizon cost-to-go is

$$J_\infty^*(x(k))=x(k)^{\mathrm{T}}P_\infty x(k)$$

The matrix P_∞ comes from an infinite recursion of the RDE, from a notional point infinitely far into the future. Assuming the RDE does converge to some constant matrix P_∞, it must satisfy the following equation (since $P_k=P_{k+1}=P_\infty$):

$$P_\infty=A^{\mathrm{T}}P_\infty A+Q-A^{\mathrm{T}}P_\infty B(B^{\mathrm{T}}P_\infty B+R)^{-1}B^{\mathrm{T}}P_\infty A$$

which is called the **Algebraic Riccati equation (ARE)**. The feedback matrix F_∞ is referred to as the

asymptotic form of the **Linear Quadratic Regulator (LQR).**

Note: In finite horizon cases, we have our feedback gain matrix to be time-variant (F_k is related to time step k since P_k is different). However, in infinite horizon cases, we have a constant feedback gain matrix F_∞ only related to P_∞. This is one of the major differences between the two problems.

- **Stability of Infinite-Horizon**

The solution given above assumes the RDE converges. Therefore, we need to derive conditions for convergence. Further, the stability of the system can also be derived.

Theorem (Necessary Condition of Convergence of RDE)

If (A,B) is stabilizable and $(Q^{1/2},A)$ is detectable, then the RDE (initialized with Q at $k=\infty$ and solved for $k\to 0$ converges to the unique positive definite solution.

Theorem (Stability of Infinite-Horizon LQR)

The closed-loop system with $u_k^* = F_\infty x_k$ is guaranteed to be asymptotically stable, under the stabilizability and detectability assumptions stated above.

Proof Sketch: Substitute control low $u(k)=F_\infty x(k)$ into $x(k+1)=Ax(k)+Bu(k)$, we get the close loop system: $x(k+1)=(A+BF_\infty)x(k)$ and the asymptotic stability can be proven by showing that the infinite horizon cost $J_\infty^*(x(k))=x(k)^{\mathrm{T}}P_\infty x(k)$ is actually a Lyapunov function of the system. i.e.:

$$J_\infty^*(x(k))>0,\ \forall k\neq 0,\ J_\infty^*(0)=0 \text{ and } J_\infty^*(x(k+1))<J_\infty^*(x(k)),\ \forall k$$

Which would then implies that:

$$\lim_{k\to\infty} x(k)=0$$

Note: Intuition for the theorem

(A,B) is stabilizable is obviously the basic requirement for any of the controllers. As for $(Q^{1/2},A)$ detectable, note that the objective term $x_k^{\mathrm{T}}Qx_k$ can be written as $(Q^{1/2}x_k)^{\mathrm{T}}Q^{1/2}x_k$. If $(Q^{1/2},A)$ is not detectable, there might be some unstable state modes that cannot be captured in $Q^{1/2}x_k$, thus not appearing in the objective. Hence the control would be useless.

- **Choices of Terminal Weight in Finite Horizon Control**

The choice of the terminal weight in finite horizon control could sometimes be tricky, and we can get inspiration from the infinite horizon cases. Here, three kinds of choices are listed and analyzed.

1. The terminal cost P of the finite horizon problem can in fact trivially be chosen so that its solution matches the infinite horizon solution:

   To do this, make P equal to the optimal cost from N to ∞ (i.e. the cost with the optimal control choice). This can be computed from the ARE:

   $$P=A^{\mathrm{T}}PA+Q-A^{\mathrm{T}}PB(B^{\mathrm{T}}PB+R)^{-1}B^{\mathrm{T}}PA$$

   This approach rests on the assumption that no constraints will be active after the end of the horizon.

2. Choose P assuming no control action after the end of the horizon, so that:

$$x(k+1) = Ax(k),\ k = N, \ldots, \infty$$

This P can be determined by solving the Lyapunov equation:

$$APA^{\mathrm{T}} + Q = P$$

This approach only makes sense if the system is asymptotically stable (or no positive definite solution P will exist).

3. Assume we want the state and input both to be zero after the end of the finite horizon. In this case, no P but an extra constraint is needed:

$$x_N = 0$$