Optimal Control and Reinforcement Learning

Heng Yang

2025-09-02

Contents

Preface

This is the textbook for Harvard ES/AM 158: Introduction to Optimal Control and Reinforcement Learning.

Feedback

I would like to invite you to provide feedback to the textbook via inline comments with Hypothesis:

- Go to Hypothesis and create an account
- Install the Chrome extension of Hypothesis
- Provide public comments to textbook contents and I will try to address them

Offerings

2025 Fall

Time: Mon/Wed 2:15 - 3:30pm

Location: SEC 1.413

Instructor: Heng Yang

Teaching Fellow: Haoyu Han, Han Qi

[Syllabus], [Problem Sets], [Canvas]

2023 Fall

The course was previously offered as Introduction to Optimal Control and Estimation.

Starting Fall 2025, contents about reinforcement learning have been added to the course.

Chapter 1

Markov Decision Process

Optimal control (OC) and reinforcement learning (RL) address the problem of making **optimal decisions** in the presence of a **dynamic environment**.

- In **optimal control**, this dynamic environment is often referred to as a *plant* or a *dynamical system*.
- In **reinforcement learning**, it is modeled as a *Markov decision process* (MDP).

The goal in both fields is to evaluate and design decision-making strategies that optimize long-term performance:

- **RL** typically frames this as maximizing a long-term *reward*.
- **OC** often formulates it as minimizing a long-term *cost*.

The emphasis on **long-term** evaluation is crucial. Because the environment evolves over time, decisions that appear beneficial in the short term may lead to poor long-term outcomes and thus be suboptimal.

With this motivation, we now formalize the framework of Markov Decision Processes (MDPs), which are discrete-time stochastic dynamical systems.

1.1 Finite-Horizon MDP

We begin with finite-horizon MDPs and introduce infinite-horizon MDPs in the following section. An abstract definition of the finite-horizon case will be presented first, followed by illustrative examples.

A finite-horizon MDP is given by the following tuple:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, T),$$

where

- $\mathcal{S}$: state space (set of all possible states)
- $\mathcal{A}$: action space (set of all possible actions)
- $P(s' \mid s, a)$: probability of transitioning to state s' from state s under action a (i.e., dynamics)
- $R(s, a)$: reward of taking action a in state s
- T: horizon, a positive integer

For now, let us assume both the state space and the action space are discrete and have a finite number of elements. In particular, denote the number of elements in $\mathcal{S}$ as $|\mathcal{S}|$, and the number of elements in $\mathcal{A}$ as $|\mathcal{A}|$. This is also referred to as a *tabular MDP*.

Policy. Decision-making in MDPs is represented by policies. A policy is a function that, given any state, outputs a distribution of actions: $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$. That is, $\pi(a \mid s)$ returns the probability of taking action a in state s. In finite-horizon MDPs, we consider a tuple of policies:

$$\pi = (\pi_0, \dots, \pi_t, \dots, \pi_{T-1}), \tag{1.1}$$

where each π_t denotes the policy at step $t \in [0, T-1]$.

Trajectory and Return. Given an initial state $s_0 \in \mathcal{S}$ and a policy π, the MDP will evolve as

1. Start at state s_0
2. Take action $a_0 \sim \pi_0(a \mid s_0)$ following policy π_0
3. Collect reward $r_0 = R(s_0, a_0)$ (assume R is deterministic)
4. Transition to state $s_1 \sim P(s' \mid s_0, a_0)$ following the dynamics
5. Go to step 2 and continue until reaching state s_T

This evolution generates a trajectory of states, actions, and rewards:

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T).$$

The cumulative reward of this trajectory is $g_0 = \sum_{t=0}^{T-1} r_t$, which is called the *return* of the trajectory. Clearly, g_0 is a random variable due to the stochasticity of both the policy and the dynamics. Similarly, if the state at time t is s_t, we denote:

$$g_t = r_t + \cdots + r_{T-1}$$

as the return of the policy starting at s_t.

1.1.1 Value Functions

State-Value Function. Given a policy π as in (1.1), which states are preferable at time t? The (time-indexed) state-value function assigns to each $s \in \mathcal{S}$ the expected return from t onward when starting in s and following π thereafter. Formally, define

$$V_t^\pi(s) := \mathbb{E}\left[g_t \mid s_t = s\right] = \mathbb{E}\left[\sum_{i=t}^{T-1} R(s_i, a_i) \middle| s_t = s, a_i \sim \pi_i(\cdot \mid s_i), s_{i+1} \sim P(\cdot \mid s_i, a_i)\right]. \tag{1.2}$$

The expectation is over the randomness induced by both the policy and the dynamics. Thus, if $V_t^\pi(s_1) > V_t^\pi(s_2)$, then at time t under policy π it is better in expectation to be in s_1 than in s_2 because the former yields a larger expected return.

$V_t^\pi(s)$: given policy π, how good is it to start in state s at time t?

Action-Value Function. Similarly, the action-value function assigns to each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ the expected return obtained by starting in state s, taking action a first, and then following policy π thereafter:

$$\begin{aligned} Q_t^\pi(s, a) :=& \mathbb{E}\left[R(s, a) + g_{t+1} \mid s_{t+1} \sim P(\cdot \mid s, a)\right] \\ =& \mathbb{E}\left[R(s, a) + \sum_{i=t+1}^{T-1} R(s_i, a_i) \middle| s_{t+1} \sim P(\cdot \mid s, a)\right]. \end{aligned} \tag{1.3}$$

The key distinction is that the action-value function evaluates the return when the first action may deviate from policy π, whereas the state-value function assumes strict adherence to π. This flexibility makes the action-value function central to improving π, since it reveals whether alternative actions can yield higher returns.

$Q_t^\pi(s, a)$: At time t, how good is it to take action a in state s, then follow the policy π?

It is easy to verify that the state-value function and the action-value function satisfy:

$$V_t^\pi(s) = \sum_{a \in \mathcal{A}} \pi_t(a \mid s) Q_t^\pi(s, a), \tag{1.4}$$

$$Q_t^\pi(s, a) = R(s, a) + \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V_{t+1}^\pi(s'). \tag{1.5}$$

From these two equations, we can derive the Bellman Consistency equations.

Proposition 1.1 (Bellman Consistency). *The state-value function $V_t^\pi(\cdot)$ in* (1.2) *satisfies the following recursion:*

$$\begin{aligned} V_t^\pi(s) &= \sum_{a\in\mathcal{A}} \pi_t(a \mid s)\left(R(s,a) + \sum_{s'\in\mathcal{S}} P(s' \mid s,a) V_{t+1}^\pi(s')\right) \\ &=: \mathbb{E}_{a\sim\pi_t(\cdot\mid s)}\left[R(s,a) + \mathbb{E}_{s'\sim P(\cdot\mid s,a)}[V_{t+1}^\pi(s')]\right]. \end{aligned} \tag{1.6}$$

Similarly, the action-value function $Q_t^\pi(s,a)$ in (1.3) *satisfies the following recursion:*

$$\begin{aligned} Q_t^\pi(s,a) &= R(s,a) + \sum_{s'\in\mathcal{S}} P(s' \mid s,a)\left(\sum_{a'\in\mathcal{A}} \pi_{t+1}(a' \mid s') Q_{t+1}^\pi(s',a')\right) \\ &=: R(s,a) + \mathbb{E}_{s'\sim P(\cdot\mid s,a)}\left[\mathbb{E}_{a'\sim\pi_{t+1}(\cdot\mid s')}[Q_{t+1}^\pi(s',a')]\right]. \end{aligned} \tag{1.7}$$

1.1.2 Policy Evaluation

The Bellman consistency result in Proposition 1.1 is fundamental because it directly yields an algorithm for evaluating a given policy π—that is, for computing its state-value and action-value functions—provided the transition dynamics of the MDP are known.

Policy evaluation for the state-value function proceeds as follows:

- **Initialization:** set $V_T^\pi(s) = 0$ for all $s \in \mathcal{S}$.
- **Backward recursion:** for $t = T-1, T-2, \dots, 0$, update each $s \in \mathcal{S}$ by

$$V_t^\pi(s) = \mathbb{E}_{a\sim\pi_t(\cdot\mid s)}\left[R(s,a) + \mathbb{E}_{s'\sim P(\cdot\mid s,a)}[V_{t+1}^\pi(s')]\right].$$

Similarly, policy evaluation for the action-value function is given by:

- **Initialization:** set $Q_T^\pi(s,a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}$.
- **Backward recursion:** for $t = T-1, T-2, \dots, 0$, update each $(s,a) \in \mathcal{S}\times\mathcal{A}$ by

$$Q_t^\pi(s,a) = R(s,a) + \mathbb{E}_{s'\sim P(\cdot\mid s,a)}\left[\mathbb{E}_{a'\sim\pi_{t+1}(\cdot\mid s')}[Q_{t+1}^\pi(s',a')]\right].$$

The essential feature of this algorithm is its backward-in-time recursion: the value functions are first set at the terminal horizon T, and then propagated backward step by step through the Bellman consistency equations.

Example 1.1 (MDP, Transition Graph, and Policy Evaluation)**.** It is often useful to visualize small MDPs as transition graphs, where states are represented by nodes and actions are represented by directed edges connecting those nodes.

As a simple illustrative example, consider a robot navigating on a two-state grid. At each step, the robot can either Stay in its current state or Move to the other state. This finite-horizon MDP is fully specified by the tuple of states, actions, transition dynamics, rewards, and horizon:

- States: $\mathcal{S} = \{\alpha, \beta\}$
- Actions: $\mathcal{A} = \{\text{Move}, \text{Stay}\}$
- Transition dynamics: we can specify the transition dynamics in the following table

State s	Action a	Next State s'	Probability $P(s' \mid s, a)$
α	Stay	α	1
α	Move	β	1
β	Stay	β	1
β	Move	α	1

- Reward: $R(s, a) = 1$ if $a = \text{Move}$ and $R(s, a) = 0$ if $a = \text{Stay}$
- Horizon: $T = 2$.

This MDP can be represented by the transition graph in Fig. 1.1. Note that for this MDP, the transition dynamics is deterministic. We will see a stochastic MDP soon.

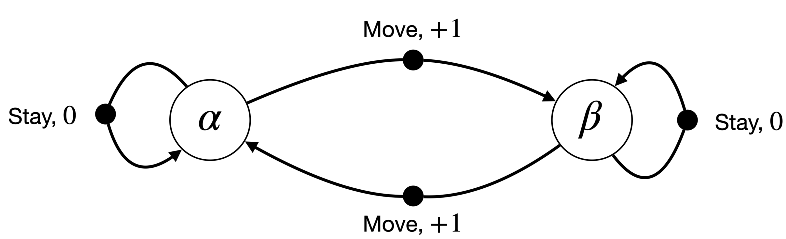

Figure 1.1: A Simple Transition Graph.

At time $t = 0$, if the robot starts at $s_0 = \alpha$, first chooses action $a_0 = \text{Move}$, and then chooses action $a_1 = \text{Stay}$, the resulting trajectory is

$$\tau = (\alpha, \text{Move}, +1, \beta, \text{Stay}, 0, \beta).$$

The return of this trajectory is:

$$g_0 = +1 + 0 = +1.$$

Policy Evaluation. Given a policy

$$\pi = (\pi_0, \pi_1), \quad \pi_0(a \mid s) = \begin{cases} 0.5 & a = \text{Move} \\ 0.5 & a = \text{Stay} \end{cases}, \quad \pi_1(a \mid s) = \begin{cases} 0.8 & a = \text{Move} \\ 0.2 & a = \text{Stay} \end{cases}. \tag{1.8}$$

We can use the Bellman consistency equations to compute the state-value function. We first initialize:

$$V_2^\pi = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

where the first row contains the value at $s = \alpha$ and the second row contains the value at $s = \beta$. We then perform the backward recursion for $t = 1$. For $s = \alpha$, we have

$$V_1^\pi(\alpha) = \begin{bmatrix} \pi_1(\text{Move} \mid \alpha) \\ \pi_1(\text{Stay} \mid \alpha) \end{bmatrix}^\top \begin{bmatrix} R(\alpha, \text{Move}) + V_2^\pi(\beta) \\ R(\alpha, \text{Stay}) + V_2^\pi(\alpha) \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}^\top \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0.8 \tag{1.9}$$

For $s = \beta$, we have

$$V_1^\pi(\beta) = \begin{bmatrix} \pi_1(\text{Move} \mid \beta) \\ \pi_1(\text{Stay} \mid \beta) \end{bmatrix}^\top \begin{bmatrix} R(\beta, \text{Move}) + V_2^\pi(\alpha) \\ R(\beta, \text{Stay}) + V_2^\pi(\beta) \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}^\top \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0.8. \tag{1.10}$$

Therefore, we have

$$V_1^\pi = \begin{bmatrix} 0.8 \\ 0.8 \end{bmatrix}.$$

We then proceed to the backward recursion for $t = 0$:

$$V_0^\pi(\alpha) = \begin{bmatrix} \pi_0(\text{Move} \mid \alpha) \\ \pi_0(\text{Stay} \mid \alpha) \end{bmatrix}^\top \begin{bmatrix} R(\alpha, \text{Move}) + V_1^\pi(\beta) \\ R(\alpha, \text{Stay}) + V_1^\pi(\alpha) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}^\top \begin{bmatrix} 1.8 \\ 0.8 \end{bmatrix} = 1.3. \tag{1.11}$$

$$V_0^\pi(\beta) = \begin{bmatrix} \pi_0(\text{Move} \mid \beta) \\ \pi_0(\text{Stay} \mid \beta) \end{bmatrix}^\top \begin{bmatrix} R(\beta, \text{Move}) + V_0^\pi(\alpha) \\ R(\beta, \text{Stay}) + V_0^\pi(\beta) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}^\top \begin{bmatrix} 1.8 \\ 0.8 \end{bmatrix} = 1.3. \tag{1.12}$$

Therefore, the state-value function at $t = 0$ is

$$V_0^\pi = \begin{bmatrix} 1.3 \\ 1.3 \end{bmatrix}.$$

You are encouraged to carry out the similar calculations for the action-value function.

The toy example was small enough to carry out policy evaluation by hand; in realistic MDPs, we will need the help from computers.

1.2 Principle of Optimality

1.3 Dynamic Programming

Appendix A

Convex Analysis and Optimization

A.1 Theory

A.1.1 Sets

Convex set is one of the most important concepts in convex optimization. Checking convexity of sets is crucial to determining whether a problem is a convex problem. Here we will present some definitions of some set notations in convex optimization.

Definition A.1 (Affine set). A set $C \subset \mathbb{R}^n$ is affine if the line through any two distinct points in C lies in C, i.e., if for any $x_1, x_2 \in C$ and any $\theta \in \mathbb{R}$, we have $\theta x_1 + (1-\theta)x_2 \in C$.

Definition A.2 (Convex set). A set $C \subset \mathbb{R}^n$ is convex if the line segment between any two distinct points in C lies in C, i.e., if for any $x_1, x_2 \in C$ and any $\theta \in [0,1]$, we have $\theta x_1 + (1-\theta)x_2 \in C$.

Definition A.3 (Cone). A set $C \subset \mathbb{R}^n$ is a cone if for any $x \in C$ and any $\theta \geq 0$, we have $\theta x \in C$.

Definition A.4 (Convex Cone). A set $C \subset \mathbb{R}^n$ is a convex cone if C is convex and a cone.

Below are some important examples of convex sets:

Definition A.5 (Hyperplane). A hyperplane is a set of the form

$$\{x | a^T x = b\}$$

Definition A.6 (Halfspaces)**.** A (closed) halfspace is a set of the form

$$\{x|a^T x \leq b\}$$

Definition A.7 (Balls)**.** A ball is a set of the form

$$B(x, r) = \{y|\|y - x\|_2 \leq r\} = \{x + ru|\|u\|_2 \leq 1\}$$

where $r > 0$.

Definition A.8 (Ellipsoids)**.** A ellipsoid is a set of the form

$$\mathcal{E} = \{y|(y - x)^T P^{-1}(y - x) \leq 1\}$$

where P is symmetric and positive definite.

Definition A.9 (Polyhedra)**.** A polyhedra is defined as the solution set of a finite number of linear equalities and inequalities:

$$\mathcal{P} = \{x|a_j^T x \leq b_j, j = 1, ..., m, c_k^T x = d_k, k = 1, ..., p\}$$

Definition A.10 (Norm ball)**.** A norm ball B of radius r and a center x_c associated with the norm $\|\cdot\|$ is defined as:

$$B = \{x|\|x - x_c\| \leq r\}$$

Definition A.11 (Norm cone)**.** A norm cone C associated with the norm $\|\cdot\|$ is defined as:

$$C = \{(x, t)|\|x\| \leq t\} \subset \mathbb{R}^{n+1}$$

Simplexes are important family of polyhedra. Suppose the $k + 1$ points $v_0, ..., v_k \in \mathbb{R}^n$ are affinely independent, which means $v_1 - v_0, ..., v_k - v_0$ are linearly independent.

Definition A.12 (Simplex)**.** A simplex C defined by points $v_0, ..., v_k$ is:

$$C = \textbf{conv}\{v_0, ..., v_k\} = \{\theta_0 v_0 + ...\theta_k v_k|\theta \succeq 0, \mathbf{1}^T\theta = 1\}$$

Extremely important examples of convex sets are positive semidefinite cones:

Definition A.13 (Symmetric,positive semidefinite,positive definite matrices)**.**

1. Symmetric matrices: $\mathbf{S}^n = \{X \in \mathbb{R}^{n\times n}|X = X^T\}$
2. Symmetric Positive Semidefinite matrices: $\mathbf{S}^n_+ = \{X \in \mathbf{S}^n|X \succeq 0\}$
3. Symmetric Positive definite matrices: $\mathbf{S}^n_{++} = \{X \in \mathbf{S}^n|X \succ 0\}$

In most scenarios, the set we encounter is more complicated. In general it is extermely hard to determine whether a set in convex or not. But if the set is 'generated' by some convex sets, we can easily determine its convexity. So let's focus on operations that preserve convexity:

Proposition A.1. *Assume S is convex, $S_\alpha, \alpha \in \mathcal{A}$ is a family of convex sets. Following operations on convex sets will preserve convexity:*

1. *Intersection: $\bigcap_{\alpha \in \mathcal{A}} S_\alpha$ is convex.*
2. *Image under affine function: A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is affine if it has the form $f(x) = Ax + b$. The image of S under affine function f is convex. I.e. $f(S) = \{f(x)|x \in S\}$ is convex*
3. *Image under perspective function: We define the perspective function P : $\mathbb{R}^{n+1}$, with domain $\textbf{dom}P = \mathbb{R}^n \times \mathbb{R}_{++}$(where $\mathbb{R}_{++} = \{x \in \mathbb{R}|x > 0\}$) as $P(z,t) = z/t$. The image of S under perspective function is convex.*
4. *Image under linear-fractional function: We define linear fractional function $f : \mathbb{R}^n \to \mathbb{R}^m$ as:$f(x) = (Ax+b)/(c^T x+d)$ with $\textbf{dom}f = \{x|c^T x+d > 0\|$. The image of S under linear fractional functions is convex.*

In some cases, the restrictions of **interior** is too strict. For example, imagine a plane in $\mathbb{R}^3$. The interior of the plane is $\emptyset$. But intuitively many property should be extended to this kind of situation. Because the points in the plane also lies 'inside' the convex set. Thus, we will define **relative interior**. First we will define **affine hull**.

Definition A.14 (Affine hull)**.** The affine hull of a set S is the smallest affine set that contains S, which can be written as:

$$\text{aff}(S) = \{\sum_{i=1}^{k} \alpha_i x_i | k > 0, x_i \in S, \alpha_i \in \mathbb{R}, \sum_{i=1}^{k} \alpha_i = 1\}$$

Definition A.15 (Relative Interior)**.** The relative interior of a set S (denoted relint(S)) is defined as its interior within the affine hull of S. I.e.

$$\text{relint}(S) := \{x \in S : \text{there exists } \epsilon > 0 \text{ such that } N_\epsilon \cap \text{aff}(S) \subset S\}$$

where $N_\epsilon(x)$ is a ball of radius ϵ centered on x.

A.1.2 Convex function

In this section, let's define convex functions:

Definition A.16 (Convex function)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if **dom** f is convex and $\forall x, y \in$ **dom** f and with $\theta \in [0, 1]$, we have:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$

The function is **strictly convex** if the inequality holds whenever $x \neq y$ and $\theta \in (0, 1)$.

If a function is differentiable, it will be easier for us to check its convexity:

Proposition A.2 (Conditions for Convex function)**.** *1.(First order condition) Suppose f is differentiable, then f is convex if and only if* ***dom****f is convex and $\forall x, y \in$* ***dom****f,*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

2.(Second order conditions) Suppose f is twice differentiable, then f is convex if and only if ***dom****f is convex and $\forall x \in$* ***dom****f,*

$$\nabla^2 f(x) \succeq \boldsymbol{0}$$

For the same purpose, some operations that preserve the convexity of the convex functions are presented here:

Proposition A.3 (Operations that preserve convexity)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and $g_1, ..., g_n$ be convex functions. The following operations will preserve convexity of the function:*

1.(Nonnegative weighted sum): A nonnegative weighted sum of convex functions:

$$f = \omega_1 f_1 + ... + \omega_m f_m$$

2.(Composition with an affine mapping) Suppose $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$, then $g(x) = f(Ax + b)$ is convex.

3.(Pointwise maximum and supremum) $g(x) = \max\{g_1(x), ..., g_n(x)\}$ is convex. If $h(x, y)$ is convex in x for each $y \in \mathcal{A}$, then $\sup_{y \in \mathcal{A}} h(x, y)$ is also convex in x.

4.(Minimization) If $h(x, y)$ is convex in (x, y), and C is a convex nonempty set, then $\inf_{x \in C} h(x, y)$ is convex in x.

5.(Perspective of a function) The perspective of f is the function $h : \mathbb{R}^{n+1} \to \mathbb{R}$ defined by: $h(x, t) = tf(x/t)$ with domain ***dom*** *$h = \{(x, t) | x/t \in$* ***dom****$f, t > 0\}$. And h is convex.*

A.1.3 Lagrange dual

We consider an optimization problem in the standard form (without assuming convexity of anything):

$$\begin{aligned} p^* = \quad & \min_x \quad f_0(x) \\ & \text{s.t.} \quad f_i(x) \leq 0 \quad i = 1..., m \\ & \qquad\;\; h_i(x) = 0 \quad i = 1, ..., p \end{aligned} \tag{A.1}$$

Definition A.17 (Lagrange dual function)**.** The Lagrangian related to the problem above is defined as:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

The Lagrange dual function is defined as:

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu)$$

When the Lagrangian is unbounded below in x, the dual function takes on the value $-\infty$. Note that since the Lagrange dual function is a pointwise infimum of a family of affine functions of (λ, ν), so it's concave. The Lagrange dual function will give us lower bounds of the optimal value of the original problem:

$$g(\lambda, \nu) \le p^*$$

. We can see that, the dual function can give a nontrivial lower bound only when $\lambda \succeq 0$. Thus we can solve the following dual problem to get the best lower bound.

Definition A.18 (Lagrange dual problem)**.** The lagrangian dual problem is defined as follows:

$$\begin{aligned} d^* = \quad & \max_{\lambda,\nu} \quad g(\lambda, \nu) \\ & \text{s.t.} \quad \lambda \succeq 0 \end{aligned} \tag{A.2}$$

This is a convex optimization problem.

We can easily see that

$$d^* \le p^*$$

always hold. This property is called **weak duality**. If

$$d^* = p^*$$

, it's called **strong duality**. Strong duality does not hold in general, but it usually holfs for convex problems. We can find conditions that guarantee strong duality in convex problems, which are called constrained qualifications. Slater's constraint qualification is a useful one.

Theorem A.1 (Slater's constraint qualification)**.** *Strong duality holds for a convex problem*

$$\begin{aligned} p^* = \quad & \min_{x} \quad f_0(x) \\ & s.t. \quad f_i(x) \le 0 \quad i = 1..., m \\ & \qquad\; Ax = b \end{aligned} \tag{A.3}$$

if it is strictly feasible, i.e.

$$\exists x \in \textbf{\textit{relint}}\mathcal{D}: \quad f_i(x) < 0, \quad i = 1...m, \quad Ax = b$$

And the linear inequalities do not need to hold with strict inequality.

A.1.4 KKT condition

Note that if strong duality holds, denote x^* to be primal optimal, and (λ^*, ν^*) to be dual optimal. Then:

$$
\begin{aligned}
f_0(x^*) = g(\lambda^*, \nu^*) = & \inf_x (f_0(x) + \sum_{i=1}^{m} \lambda_i^* f_i(x) + \sum_{i=1}^{p} \nu_i^* h_i(x)) \\
\le & f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* f_i(x) + \sum_{i=1}^{p} \nu_i^* h_i(x) \\
\le & f_0(x^*)
\end{aligned}
\tag{A.4}
$$

from this, combining $\lambda^* \geq 0$ and $f_i(x^*) \leq 0$, we can know that: $\lambda_i^* f_i(x^*) = 0 \quad i = 1 \cdots m$. This means for λ_i^* and $f_i(x^*)$, one of them must be zero, which is known as complementary slackness).

Thus we arrived at the following four conditions, which are called KKT conditions.

Theorem A.2 (Karush-Kuhn-Tucker(KKT) Conditions)**.** *The following four conditions are called KKT conditions (for a problem with differentiable f_i, h_i)*

1. *Primal feasible:* $f_i(x) \leq 0, i, \cdots, m,\ \ h_i(x) = 0, i = 1, \cdots, p$
2. *Dual feasible:* $\lambda \succeq 0$
3. *Complementary slackness:* $\lambda_i f_i(x) = 0, i = 1, \cdots, m$
4. *Gradient of Lagrangian with respect to x vanishes:*$\nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{p} \nu_i \nabla h_i(x) = 0$

From the discussion above, we know that if strong duality holds and x, λ, ν are optimal, then they must satisfy the KKT conditions.

Also if x, λ, ν satisfy KKT for a convex problem, then they are optimal. However, the converse is not generally true, since KKT condition implies strong duality. If Slater's condition is satisfied, then x is optimal if and only if there exist λ, ν that satisfy KKT conditions. Sometimes, by solving the KKT system, we can derive the closed-form solution of a optimization directly. Also, sometimes we will use the residual of the KKT system as the termination condition.

In general, f_i, h_i may not be differentiable. There are also KKT conditions for them, which will include knowledge of subdifferential and will not be included here.

A.2 Practice

A.2.1 CVX Introduction

In the last section, we have learned basic concepts and theorems in convex optimization. In this section, on the other hand, we will introduce you how to model basic convex optimization problems with CVX, an easy-to-use MATLAB package. To install CVX, please refer to this page. Note that every time you what to use the CVX package, you should add it to your MATLAB path. For example, if I install CVX package in the parent directory of my current directory with default directory name `cvx`, the following line should be added before your CVX codes:

```
addpath(genpath("../cvx/"));
```

With CVX, it is incredibly easy for us to define and solve a convex optimization problem. You just need to:

1. define the variables.
2. define the objective function you want to minimize or maximize.
3. define the constraints.

After running your codes, the optimal objective value is stored in the variable `cvx_optval`, and the problem status is stored in the variable `cvx_status` (when your problem is well-defined, this variable's value will be `Solved`). The optimal solutions will be stored in the variables you define.

Throughout this section, we will study five types of convex optimization problems: linear programming (LP), quadratic programming (QP), (convex) quadratically constrained quadratic programming (QCQP), second-order cone programming (SOCP), and semidefinite programming (SDP). Given two types of optimization problems A and B, we say $A < B$ if A can always be converted to B while the inverse is not true. Under this notation, we have

$$\text{LP} < \text{QP} < \text{QCQP} < \text{SOCP} < \text{SDP}$$

A.2.2 Linear Programming (LP)

Definition. An LP has the following form:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & c^T x \\ \text{subject to } & Ax \le b \end{aligned} \tag{A.5}$$

where x is the variable, $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$ are the parameters. Note that the constraint $Ax \leq b$ already incorporates linear equality constraints. To see this, consider the constraint $A'x = b'$, we can reformulate it as $Ax \leq b$ by

$$\begin{bmatrix} A' \\ -A' \end{bmatrix} x \leq \begin{bmatrix} b' \\ -b' \end{bmatrix}$$

Example. Consider the problem of minimizing a linear function $c_1 x_1 + c_2 x_2$ over a rectangle $[-l_1, l_1] \times [-l_2, l_2]$. We can convert it to the standard LP form in (A.5) by simply setting c as $[c_1,\ c_2]^T$ and the linear inequality constraint as

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} l_1 \\ l_1 \\ l_2 \\ l_2 \end{bmatrix}$$

Corresponding CVX codes are shown below:

```
%% Define the LP example setting
c1 = 2;
c2 = -5;
l1 = 3;
l2 = 7;
% parameters: c, A, b
c = [c1; c2];
A = [1, 0; -1, 0; 0, 1; 0, -1];
b = [l1; l1; l2; l2];

%% solve LP
cvx_begin
    variable x(2); % define variables [x1, x2]
    minimize(c' * x); % define the objective
    subject to
        A * x <= b; % define the linear constraint
cvx_end
```

A.2.3 Quadratic Programming (QP)

Definition. A QP has the following form:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T P x + q^T x \tag{A.6}$$

$$\text{subject to } Gx \leq h \tag{A.7}$$

$$Ax = b \tag{A.8}$$

where $P \in \mathcal{S}_+^n, q \in \mathbb{R}^n, G \in \mathbb{R}^{m\times n}, h \in \mathbb{R}^m, A \in \mathbb{R}^{p\times n}, b \in \mathbb{R}^p$. Here $\mathcal{S}_+^n$ denotes the set of positive semidefinite matrices of size $n \times n$. Obviously, if we set P as zero, QP will degenerate to LP.

Example. Consider the problem of minimizing a quadratic function

$$f(x_1, x_2) = p_1 x_1^2 + 2p_2 x_1 x_2 + p_3 x_2^2 + q_1 x_1 + q_2 x_2$$

over a rectangle $[-l_1, l_1] \times [-l_2, l_2]$. Since $P = 2\begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix} \succeq 0$, the following two conditions must hold:

$$\begin{cases} p_1 \geq 0 \\ p_1 p_3 - 4p_2^2 \geq 0 \end{cases}$$

Same as in the LP example, G and h can be expressed as:

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} l_1 \\ l_1 \\ l_2 \\ l_2 \end{bmatrix}$$

Corresponding CVX codes are shown below:

```
%% Define the QP example setting
p1 = 2;
p2 = 0.5;
p3 = 4;
q1 = -3;
q2 = -6.5;
l1 = 2;
l2 = 2.5;
% check if the generated P is positive semidefinite
tmp1 = (p1 >= 0);
tmp2 = (p1*p3 - 4*p2^2 >= 0);
if ~(tmp1 && tmp2)
    error("P is not positve semidefinite!");
end
% parameters: P, q, G, h
P = 2 * [p1, p2; p2, p3];
q = [q1; q2];
G = [1, 0; -1, 0; 0, 1; 0, -1];
h = [l1; l1; l2; l2];

%% Solve the QP problem
cvx_begin
    variable x(2); % define variables [x1; x2]
```

```
    % define the objective, where quad_form(x, P) = x'*P*x
    obj = 0.5 * quad_form(x, P) + q' * x;
    minimize(obj);
    subject to
        G * x <= h; % define the linear constraint
cvx_end
```

A.2.4 Quadratically Constrained Quadratic Programming (QCQP)

Definition. An (convex) QCQP has the following form:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^T P_0 x + q_0^T x \tag{A.9}$$

$$\text{subject to } \frac{1}{2}x^T P_i x + q_i^T x + r_i \leq 0,\ i = 1\dots m \tag{A.10}$$

$$Ax = b \tag{A.11}$$

where $P_i \in \mathcal{S}_+^n, i = 0\dots m$, $q_i \in \mathbb{R}^n, i = 0\dots m$, $A \in \mathbb{R}^{p\times n}$, and $b \in \mathbb{R}^p$. Note that in other literature, you may find a more general form of QCQP: they don't require P_i's to be positive semidefinite. Yet in this case, the problem is non-convex and beyond our scope.

Example. We study the problem of getting the minimum distance between two ellipses. By convention, when the ellipses overlap, we set the minimum distance as 0. This problem can be exactly solved by (convex) QCQP. Consider two ellipses of the following form:

$$\begin{cases} \frac{1}{2}\begin{bmatrix} y_1 \\ z_1 \end{bmatrix}^T K_1 \begin{bmatrix} y_1 \\ z_1 \end{bmatrix} + k_1^T \begin{bmatrix} y_1 \\ z_1 \end{bmatrix} + c_1 \leq 0 \\ \frac{1}{2}\begin{bmatrix} y_2 \\ z_2 \end{bmatrix}^T K_2 \begin{bmatrix} y_2 \\ z_2 \end{bmatrix} + k_2^T \begin{bmatrix} y_2 \\ z_2 \end{bmatrix} + c_2 \leq 0 \end{cases}$$

where $[y_1, z_1]^T$ and $[y_2, z_2]^T$ are arbitrary points inside the two ellipses respectively. Also, two ensure the ellipses are well defined, we should enforce the following properties in $(K_i, k_i, c_i), i = 1, 2$: (1) $K_i \succ 0$; (2) Let $K_i = L_i L_i^T$ be the Cholesky decomposition of K_i. Then, ellipse i can be rewritten as:

$$\frac{1}{2} \parallel L_i^T \begin{bmatrix} y_i \\ z_i \end{bmatrix} - L_i^{-1}k_i \parallel^2 \leq \frac{1}{2} \parallel L_i^{-1}k_i \parallel^2 - c_i$$

Thus,

$$\frac{1}{2} \parallel L_i^{-1}k_i \parallel^2 - c_i > 0$$

With these two assumptions, we want to minimize:

$$\frac{1}{2}(y_1 - y_2)^2 + (z_1 - z_2)^2$$

Now, we construct P, q, r's in QCQP with the above parameters. Define the variable x as $[y_1, z_1, y_2, z_2]$.

(1) P_0 can be obtained from:

$$\frac{1}{2}(y_1 - y_2)^2 + (z_1 - z_2)^2 = \frac{1}{2}\begin{bmatrix} y_1 \\ z_1 \\ y_2 \\ z_2 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ z_1 \\ y_2 \\ z_2 \end{bmatrix}$$

(2) P_1, q_1, r_1 can be obtained from:

$$\frac{1}{2}\begin{bmatrix} y_1 \\ z_1 \end{bmatrix}^T K_1 \begin{bmatrix} y_1 \\ z_1 \end{bmatrix} + k_1^T \begin{bmatrix} y_1 \\ z_1 \end{bmatrix} + c_1 = \frac{1}{2}x^T \begin{bmatrix} K_1 & O \\ O & O \end{bmatrix} + \begin{bmatrix} k_1 \\ O \end{bmatrix}^T x + c_1 \le 0$$

(3) P_2, q_2, r_2 can be obtained from:

$$\frac{1}{2}\begin{bmatrix} y_2 \\ z_2 \end{bmatrix}^T K_2 \begin{bmatrix} y_2 \\ z_2 \end{bmatrix} + k_2^T \begin{bmatrix} y_2 \\ z_2 \end{bmatrix} + c_2 = \frac{1}{2}x^T \begin{bmatrix} O & O \\ O & K_2 \end{bmatrix} + \begin{bmatrix} O \\ k_2 \end{bmatrix}^T x + c_2 \le 0$$

The corresponding codes are shown below. In this example, we test the minimum distance between a circle $y_1^2 + z_1^2 \le 1$ and another circle $(y_2 - 2)^2 + (z_2 - 2)^2 \le 1$. You can check whether the result from QCQP aligns with your manual calculation.

```
%% Define the QCQP example setting
K1 = eye(2);
k1 = zeros(2, 1);
c1 = -0.5;
K2 = eye(2);
k2 = [2; 2];
c2 = 3.5;
if ~(if_ellipse(K1, k1, c1) && if_ellipse(K2, k2, c2))
    error("The example setting is not correct");
end
% define parameters P0, P1, P2, q1, q2, r1, r2
P0 = [1,0,-1,0; 0,1,0,-1; -1,0,1,0; 0,-1,0,1];
P1 = zeros(4, 4);
P1(1:2, 1:2) = K1;
```

```
P2 = zeros(4, 4);
P2(3:4, 3:4) = K2;
q1 = [k1; zeros(2, 1)];
q2 = [zeros(2, 1); k2];
r1 = c1;
r2 = c2;

%% Solve the QCQP problem
cvx_begin
    variable x(4); % define variables [y1; z1; y2; z2]
    % define the objective, where quad_form(x, P) = x'*P*x
    obj = 0.5 * quad_form(x, P0);
    minimize(obj);
    subject to
        0.5 * quad_form(x, P1) + q1' * x + r1 <= 0;
        0.5 * quad_form(x, P2) + q2' * x + r2 <= 0;
cvx_end

%% detect whether (K, k, c) generates a ellipse
function flag = if_ellipse(K, k, c)
    L = chol(K);
    radius_square = 0.5 * norm(L \ k)^2 - c; % L \ k = inv(L) * k
    flag = (radius_square > 0);
end
```

A.2.5 Second-Order Cone Programming (SOCP)

Definition. An SOCP has the following form:

$$\min_{x\in\mathbb{R}^n} f^T x \tag{A.12}$$

$$\text{subject to } ||A_i x + b_i||_2 \leq c_i^T x + d_i,\ i = 1 \dots m \tag{A.13}$$

$$Fx = g \tag{A.14}$$

where $f \in \mathbb{R}^n, A_i \in \mathbb{R}^{n_i\times n}, b_i \in \mathbb{R}^{n_i}, c_i \in \mathbb{R}^n, d_i \in \mathbb{R}, F \in \mathbb{R}^{p\times n}$, and $g \in \mathbb{R}^p$.

Example. We consider the problem of stochastic linear programming:

$$\min_{x} c^T x \tag{A.15}$$

$$\text{subject to } \mathbb{P}(a_i^T x \leq b_i) \geq p,\ i = 1 \dots m \tag{A.16}$$

$$a_i \sim \mathcal{N}(\bar{a}_i, \Sigma_i),\ i = 1 \dots m \tag{A.17}$$

Here p should be more than 0.5. We show that this problem can be converted to a SOCP:

Since $a_i \sim \mathcal{N}(\bar{a}_i, \Sigma_i)$, then $(a_i^T x - b_i) \sim \mathcal{N}(\bar{a}_i^T x - b_i, x^T \Sigma_i x)$. Standardize it:

$$t := ||\Sigma_i^{\frac{1}{2}} x||_2^{-1} \left\{ (a_i^T x - b_i) - (\bar{a}_i^T x - b_i) \right\} \sim \mathcal{N}(0,1)$$

Then,

$$\begin{aligned}
\mathbb{P}(a_i^T x \le b_i) &= \mathbb{P}(a_i^T x - b_i \le 0) && \text{(A.18)} \\
&= \mathbb{P}(t \le -||\Sigma_i^{\frac{1}{2}} x||_2^{-1} (\bar{a}_i^T x - b_i)) && \text{(A.19)} \\
&= \Phi(-||\Sigma_i^{\frac{1}{2}} x||_2^{-1} (\bar{a}_i^T x - b_i)) && \text{(A.20)}
\end{aligned}$$

Here $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution:

$$\Phi(\xi) = \int_{-\infty}^{\xi} e^{-\frac{1}{2}t^2} \; dt$$

Thus,

$$\begin{aligned}
&\mathbb{P}(a_i^T x \le b_i) \ge p && \text{(A.21)} \\
\iff &\Phi(-||\Sigma_i^{\frac{1}{2}} x||_2^{-1} (\bar{a}_i^T x - b_i)) \ge p && \text{(A.22)} \\
\iff &-||\Sigma_i^{\frac{1}{2}} x||_2^{-1} (\bar{a}_i^T x - b_i) \ge \Phi^{-1}(p) && \text{(A.23)} \\
\iff &\Phi^{-1}(p) ||\Sigma_i^{\frac{1}{2}} x||_2 \le b_i - \bar{a}_i^T x && \text{(A.24)}
\end{aligned}$$

which is exactly the same as inequality constraints in SOCP formulation. (You can see why we enforce $p > 0.5$ here: otherwise $\Phi^{-1}(p)$ will be negative and the constraint with not be an second-order cone.)

In the following code example, we set up four inequality constraints and let $\bar{a}_i^T x \le b_i, \; i = 1 \dots 4$ form an square located at the origin of size 2. Then, for convenience, we set $\Sigma_i \equiv \sigma^2 I$.

```
%% Define the SOCP example setting
bar_a1 = [1; 0];
b1 = 1;
bar_a2 = [0; 1];
b2 = 1;
bar_a3 = [-1; 0];
b3 = 1;
bar_a4 = [0; -1];
b4 = 1;
sigma = 0.1;
c = [2; 3];
p = 0.9; % p should be more than 0.5
Phi_inv = norminv(p); % get Phi^{-1}(p)
```

```
%% Solve the SOCP problem
cvx_begin
    variable x(2); % define variables [x1; x2]
    minimize(c' * x);
    subject to
        sigma*Phi_inv * norm(x) <= b1 - bar_a1' * x;
        sigma*Phi_inv * norm(x) <= b2 - bar_a2' * x;
        sigma*Phi_inv * norm(x) <= b3 - bar_a3' * x;
        sigma*Phi_inv * norm(x) <= b4 - bar_a4' * x;
cvx_end
```

A.2.6 Semidefinite Programming (SDP)

Definition. An SDP has the following form:

$$\min_{X_i, x_i} \sum_{i=1}^{n_s} C_i \cdot X_i + \sum_{i=1}^{n_u} c_i \cdot x_i \tag{A.25}$$

$$\text{subject to } \sum_{i=1}^{n_s} A_{i,j} \cdot X_i + \sum_{i=1}^{n_u} a_{i,j} \cdot x_i = b_j, \quad j = 1 \ldots m \tag{A.26}$$

$$X_i \in \mathcal{S}_+^{D_i}, \quad i = 1 \ldots n_s \tag{A.27}$$

$$x_i \in \mathbb{R}^{d_i}, \quad i = 1 \ldots n_u \tag{A.28}$$

where $C_i, A_{i,j} \in \mathbb{R}^{D_i \times D_i}$, $c_i, a_{i,j} \in \mathbb{R}^{d_i}$, and $\cdot$ means element-wise product. For two square matrices A, B, the dot product $A \cdot B$ is equal to $\operatorname{tr}(AB)$; for two vectors a, b, the dot product $a \cdot b$ is the same as inner product $a^T b$.

Note that actually there are many "standard" forms of SDP. For example, in the convex optimization theory part, you may find an SDP that looks like:

$$\min_X C \cdot X \tag{A.29}$$

$$\text{subject to } A \cdot X = b \tag{A.30}$$

$$X \succeq 0 \tag{A.31}$$

It is convenient for us to analyze the theoretical properties of SDP with this form. Also, in SDP solvers' User Guide, you may see more complex SDP forms which involve more general convex cones. For example, see MOSEK's MATLAB API docs. Here we turn to use the form of (A.25) for two reasons: (1) it is general enough: our SDP example below can be converted to this form (also, SDPs from sum-of-squares programming in this book are exactly of the form (A.25)); (2) it is more readable than more complex forms.

Example. We consider the problem of finding the minimum eigenvalue for a positive semidefinite matrix S. We will show that this problem can be converted to (A.25). Since S is positive semidefinite, the finding procedure can be cast as

$$\max_{\lambda} \lambda \tag{A.32}$$

$$\text{subject to } S - \lambda I \succeq 0 \tag{A.33}$$

Now define an auxiliary matrix $X := S - \lambda I$. We have

$$\min_{\lambda, X} \; -\lambda \tag{A.34}$$

$$\text{subject to } X + \lambda I = S \tag{A.35}$$

$$X \succeq 0 \tag{A.36}$$

It is obvious that the linear matrix equality constraint $X+\lambda I = S$ can be divided into several linear scalar equality constraints in (A.25). For example, we consider $S \in \mathbb{S}_+^3$. Thereby $X + \lambda I = S$ will lead to 6 linear equality constraints (We don't consider X is a symmetric matrix here, since most solvers will implicitly consider this. Thus, only the upper-triangular part of X and S are actually used in the equality construction.):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot X + \lambda = S[0,0], \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot X = S[0,1], \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot X = S[0,2] \tag{A.37}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot X + \lambda = S[1,1], \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot X = S[1,2], \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot X + \lambda = S[2,2] \tag{A.38}$$

Seems tedious? Fortunately, CVX provides a high-level API to handle these linear equality constraints: you just need to write down

```
X + lam * eye(3) == S; % linear equality constraints: X + lam *I = S
```

CVX will autometically convert this high-level constraint to (A.25) and pass them to the underlying solver.

To generate a ramdom $S \in \mathcal{S}_+^3$, you just need to assign three nonnegative eigenvalues to the program. After that, an random S will be generated by $S = Q \operatorname{diag}(\lambda_1, \lambda_2, \lambda_3)\, Q^T$, where Q is random orthonormal matrix.

```
%% Define the SDP example setting
lam_list = [0.7; 2.4; 3.7];
S = generate_random_PD_matrix(lam_list); % get a PD matrix S
```

```
%% Solve the SDP problem
cvx_begin
    variable X(3, 3) symmetric;
    variable lam;
    maximize(lam);
    subject to
        % here "==" should be read as "is in"
        X == semidefinite(3);
        X + lam * eye(3) == S;
cvx_end

% this function help to generate PD matrix of size 3*3
% if you provide the eigenvalues [lam_1, lam_2, lam_3]
function S = generate_random_PD_matrix(lam_list)
    if ~all(lam_list >= 0) % all eigenvalues >= 0
        error("All eigenvalues must be nonnegative.");
    end
    D = diag(lam_list);
    % use QR factorization to generate a random orthonormal matrix Q
    [Q, ~] = qr(rand(3, 3));
    S =  Q * D * Q';
end
```

A.2.7 CVXPY Introduction and Examples

Apart from CVX MATLAB, we also have a Python package called CVXPY, which functions almost the same as CVX MATLAB. To define and solve a convex optimization problem CVXPY, basically, there are three steps (apart from importing necessary packages):

- Step 1: Define parameters and variables in a certain type of convex problem. Here variables are what you are trying to optimize or “learn”. Parameters are the “coefficients” of variables in the objective and constraints.
- Step 2: Define the objective function and constraints.
- Step 3: Solve the problem and get the results.

Here we provide the CVXPY codes for the above five convex optimization examples.

A.2.7.1 LP

```
import cvxpy as cp
import numpy as np

## Define the LP example setting
c1 = 2
c2 = -5
l1 = 3
l2 = 7

## Step 1: define variables and parameters
x = cp.Variable(2) # variable: x = [x1, x2]^T
# parameters: c, A, b
c = np.array([c1, c2])
A = np.array([[1, 0], [-1, 0], [0, 1], [0, -1]])
b = np.array([l1, l1, l2, l2])

## Step 2: define objective and constraints
obj = cp.Minimize(c.T @ x)
constraints = [A @ x <= b]
prob = cp.Problem(obj, constraints) # form the problem

## Step 3: solve problem and get results
prob.solve()
print("status: ", prob.status) # check whether the status is "optimal"
print("optimal value: ", prob.value) # optimal objective
print("optimal solution: ", x.value) # optimal x
```

A.2.7.2 QP

```
import cvxpy as cp
import numpy as np

## Define the LP example setting
p1 = 2
p2 = 0.5
p3 = 4
q1 = -3
q2 = -6.5
l1 = 2
```

```
l2 = 2.5
# check if the generated P is positive semidefinite
tmp1 = (p1 >= 0)
tmp2 = (p1*p3 - 4*p2**2 >= 0)
assert(tmp1 and tmp2, "P is not positve semidefinite!")

## Step 1: define variables and parameters
x = cp.Variable(2) # variable: x = [x1, x2]^T
# parameters: P, q, G, h
P = 2*np.array([[p1, p2], [p2, p3]])
q = np.array([q1, q2])
G = np.array([[1, 0], [-1, 0], [0, 1], [0, -1]])
h = np.array([l1, l1, l2, l2])

## Step 2: define the objective and constraints
fx = 0.5 * cp.quad_form(x, P) + q.T @ x
obj = cp.Minimize(fx)
constraints = [G @ x <= h]
prob = cp.Problem(obj, constraints) # form the problem

## Step 3: solve the problem and get results
prob.solve()
print("status: ", prob.status) # check whether the status is "optimal"
print("optimal value: ", prob.value) # optimal objective
print("optimal solution: ", x.value) # optimal x
```

A.2.7.3 QCQP

```
import cvxpy as cp
import numpy as np
from numpy.linalg import cholesky, inv, norm

## Define the QCQP example setting
def if_ellipse(K, k, c):
    # examine whether 0.5*x^T K x + k^T x + c <= 0 is a ellipse
    # if K is not positive semidefinite, Cholesky will raise an error
    L = cholesky(K)
    radius_square = 0.5 * norm(inv(L) @ k)**2 - c
    return radius_square > 0
K1 = np.eye(2)
k1 = np.zeros(2)
c1 = -0.5
```

```
K2 = np.array([[1, 0], [0, 1]])
k2 = np.array([2, 2])
c2 = 3.5
if not (if_ellipse(K1, k1, c1) and if_ellipse(K2, k2, c2)):
    raise ValueError("The example setting is not correct")

## Step 1: define variables and parameters
P0 = np.array([[1,0,-1,0], [0,1,0,-1], [-1,0,1,0], [0,-1,0,1]])
P1 = np.zeros((4,4))
P1[:2, :2] = K1
P2 = np.zeros((4,4))
P2[2:, 2:] = K2
q1 = np.concatenate([k1, np.zeros(2)])
q2 = np.concatenate([np.zeros(2), k2])
r1 = c1
r2 = c2

## Step 2: define objective and constraints
x = cp.Variable(4) # variable: x = [y1, z1, y2, z2]^T
fx = 0.5 * cp.quad_form(x, P0)
obj = cp.Minimize(fx)
con1 = (0.5 * cp.quad_form(x, P1) + q1.T @ x + r1 <= 0) # ellipse 1
con2 = (0.5 * cp.quad_form(x, P2) + q2.T @ x + r2 <= 0) # ellipse 2
constraints = [con1, con2]
prob = cp.Problem(obj, constraints) # form the problem

## Step 3: solve problem and get results
prob.solve()
print("status: ", prob.status) # check whether the status is "optimal"
print("optimal value: ", prob.value) # optimal objective
print("optimal solution: ", x.value) # optimal x
```

A.2.7.4 SOCP

```
import cvxpy as cp
import numpy as np
from scipy.stats import norm

## Define the SOCP example setting
# define bar_ai, bi (i = 1, 2, 3, 4)
bar_a1 = np.array([1, 0])
b1 = 1
```

```
bar_a2 = np.array([0, 1])
b2 = 1
bar_a3 = np.array([-1, 0])
b3 = 1
bar_a4 = np.array([0, -1])
b4 = 1
sigma = 0.1
c = np.array([2, 3])
p = 0.9 # p should be more than 0.5

## Step 1: define variables and parameters
Phi_inv = norm.ppf(p) # get Phi^{-1}(p)

## Step 2: define objective and constraints
x = cp.Variable(2) # variable: x = [x1, x2]^T
obj = cp.Minimize(c.T @ x)
# use cp.SOC(t, x) to create the SOC constraint ||x||_2 <= t
constraints = [
    cp.SOC(b1 - bar_a1.T @ x, sigma*Phi_inv*x),
    cp.SOC(b2 - bar_a2.T @ x, sigma*Phi_inv*x),
    cp.SOC(b3 - bar_a3.T @ x, sigma*Phi_inv*x),
    cp.SOC(b4 - bar_a4.T @ x, sigma*Phi_inv*x),
]
prob = cp.Problem(obj, constraints) # form the problem

## Step 3: solve problem and get results
prob.solve()
print("status: ", prob.status) # check whether the status is "optimal"
print("optimal value: ", prob.value) # optimal objective
print("optimal solution: ", x.value) # optimal x
```

A.2.7.5 SDP

```
import cvxpy as cp
import numpy as np
from scipy.stats import ortho_group

## Define the SDP example setting
# this function help to generate PD matrix of size 3*3
# if you provide the eigenvalues [lam_1, lam_2, lam_3]
def generate_random_PD_matrix(lam_list):
    assert np.all(lam_list >= 0) # all eigenvalues >= 0
```

```
    # S = Q @ D @ Q.T
    D = np.diag(lam_list)
    Q = ortho_group.rvs(3)
    return Q @ D @ Q.T
lam_list = np.array([0.5, 2.4, 3.7])
S = generate_random_PD_matrix(lam_list) # get a PD matrix S

## Step 1: define variables and parameters
# get coefficients for equality constraints
A_00 = np.array([[1, 0, 0], [0, 0, 0], [0, 0, 0]]) # tr(A_00 @ X) + lam = S_00
A_01 = np.array([[0, 1, 0], [0, 0, 0], [0, 0, 0]]) # tr(A_01 @ X) = S_01
A_02 = np.array([[0, 0, 1], [0, 0, 0], [0, 0, 0]]) # tr(A_02 @ X) = S_02
A_11 = np.array([[0, 0, 0], [0, 1, 0], [0, 0, 0]]) # tr(A_11 @ X) + lam = S_11
A_12 = np.array([[0, 0, 0], [0, 0, 1], [0, 0, 0]]) # tr(A_12 @ X) = S_12
A_22 = np.array([[0, 0, 0], [0, 0, 0], [0, 0, 1]]) # tr(A_22 @ X) + lam = S_22

## Step 2: define objective and constraints
# define a PD matrix variable X of size 3*3
X = cp.Variable((3, 3), symmetric=True)
constraints = [X >> 0] # the operator >> denotes matrix inequality
lam = cp.Variable(1)
constraints += [
    cp.trace(A_00 @ X) + lam == S[0,0],
    cp.trace(A_01 @ X) == S[0,1],
    cp.trace(A_02 @ X) == S[0,2],
    cp.trace(A_11 @ X) + lam == S[1,1],
    cp.trace(A_12 @ X) == S[1,2],
    cp.trace(A_22 @ X) + lam == S[2,2],
]
obj = cp.Minimize(-lam)
prob = cp.Problem(obj, constraints) # form the problem

## Step 3: solve problem and get results
prob.solve()
print("status: ", prob.status) # check whether the status is "optimal"
print("optimal value: ", prob.value) # optimal objective
print("optimal solution: ", lam.value) # optimal lam
```

Appendix B

Linear System Theory

Thanks to Shucheng Kang for writing this Appendix.

B.1 Stability

B.1.1 Continuous-Time Stability

Consider the continuous-time linear time-invariant (LTI) system

$$\dot{x} = Ax. \tag{B.1}$$

the system is said to be "diagonalizable" if A is diagonalizable.

Definition B.1 (Asymptotic and Marginal Stability)**.** The diagonalizable, LTI system (B.1) is

1. "asymptotically stable" if $x(t) \to 0$ as $t \to \infty$ for every initial condition x_0
2. "marginally stable" if $x(t) \nrightarrow 0$ but remains bounded as $t \to \infty$ for every initial condition x_0
3. "stable" if it is either asymptotically or marginally stable
4. "unstable" if it is not stable

One can show that A's eigenvalues determine the LTI system's stability, as the following Theorem states:

Theorem B.1 (Stability of Continuous-Time LTI System)**.** *The diagonalizable*[1]*, LTI system* (B.1) *is*

1. *asymptotically stable if* $Re(\lambda_i) < 0$ *for all* i
2. *marginally stable if* $Re(\lambda_i) \leq 0$ *for all* i *and there exists at least one* i *for which* $Re(\lambda_i) = 0$
3. *stable if* $Re(\lambda_i) \leq 0$ *for all* i
4. *unstable if* $Re(\lambda_i) > 0$ *for at least one* i

Proof. Here we only represent the proof of (1). Similar procedure can be adopted for the proof of (2) - (4).

Since A is diagonalizable, there exists an similarity transformation matrix T, s.t. $A = T\Lambda T^{-1}$, where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$. Then, under the coordinate transformation $z = T^{-1}x$, $\dot{x} = Ax$ can be restated as $\dot{z} = \Lambda z$. Consider the i's component of z:

$$\dot{z}_i = \lambda_i z_i \Longrightarrow z_i(t) = e^{\lambda_i t} z_i(0)$$

Since $\text{Re}(\lambda_i) < 0$, $z_i(t)$ will go to 0 as $t \to 0$ regardless how we choose $z_i(0)$.

□

B.1.2 Discrete-Time Stability

Now consider the diagonalizable, discrete-time linear time-invariant (LTI) system

$$x_{t+1} = Ax_t. \tag{B.2}$$

Theorem B.2 (Stability of Discrete-Time LTI System)**.** *The diagonalizable, discrete-time LTI system* (B.2) *is*

1. *asymptotically stable if* $|\lambda_i| < 1$ *for all* i
2. *marginally stable if* $|\lambda_i| \leq 1$ *for all* i *and there exists at least one* i *for which* $|\lambda_i| = 1$
3. *stable if* $|\lambda_i| \leq 1$ *for all* i
4. *unstable if* $|\lambda_i| > 1$ *for at least one* i*.*

Note that $|\lambda_i| < 1$ *means the eigenvalue lies strictly inside the unit circle in the complex plane.*

---

[1] when A is not diagonalizable, similar results can be derived via Jordan decomposition.

Proof. Here we only represent the proof of (1). Similar procedure can be adopted for the proof of (2) - (4).

Since A is diagonalizable, there exists an similarity transformation matrix T, s.t. $A = T\Lambda T^{-1}$, where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$. Then, under the coordinate transformation $z = T^{-1}x$, $x_{t+1} = Ax$ can be restated as $z_{t+1} = \Lambda z_t$. Expanding the recursion, we have

$$z_t = \Lambda^{t-1} z_0 \Longrightarrow z_{t,i} = \lambda_i^{t-1} z_{0,i}$$

Since $|\lambda_i| < 1$, $z_{t,i}$ will go to 0 as $t \to 0$ regardless how we choose $z_{0,i}$. □

B.1.3 Lyapunov Analysis

Theorem B.3 (Lyapunov Equation)**.** *The following is equivalent for a linear time-invariant system $\dot{x} = Ax$*

1. *The system is globally asymptotically stable, i.e., A is Hurwitz and $\lim_{t\to\infty} x(t) = 0$ regardless of the initial condition;*
2. *For any positive definite matrix Q, the unique solution P to the Lyapunov equation*
$$A^T P + PA = -Q \tag{B.3}$$
is positive definite.

Proof. (a): $2 \Rightarrow 1$. Suppose we are given two positive definite matrices $P, Q \succ 0$ that satisfies the Lyapunov equation (B.3). Define a scalar function

$$V(x) = x^T P x.$$

It is clear that $V > 0$ for any $x \neq 0$ and $V(x) = 0$ (i.e., $V(x)$ is positive definite). We also see $V(x)$ is radially unbounded because:

$$V(x) \geq \lambda_{\min}(P)\|x\|^2 \Rightarrow \lim_{x\to\infty} V(x) \to \infty.$$

The time derivative of V reads

$$\dot{V} = 2x^T P\dot{x} = x^T(A^T P + PA)x = -x^T Q x.$$

Clearly, $\dot{V} < 0$ for any $x \neq 0$ and $\dot{V}(0) = 0$. According to Lyapunov's global stability theorem **??**, we conclude the linear system $\dot{x} = Ax$ is globally asymptotically stable at $x = 0$.

(b): $1 \Rightarrow 2$. Suppose A is Hurwitz, we want to show that, for any $Q \succ 0$, there exists a unique $P \succ 0$ satisfying the Lyapunov equation (B.3). In fact, consider the matrix

$$P = \int_{t=0}^{\infty} e^{A^T t} Q e^{At} dt.$$

Because A is Hurwitz, the integral exists, and clearly $P \succ 0$ due to $Q \succ 0$. To show this choice of P satisfies the Lyapunov equation, we write

$$A^T P + PA = \int_{t=0}^{\infty} \left(A^T e^{A^T t} Q e^{At} + e^{A^T t} Q e^{At} A \right) dt \tag{B.4}$$

$$= \int_{t=0}^{\infty} d\left(e^{A^T t} Q e^{At} \right) \tag{B.5}$$

$$= e^{A^T t} Q e^{At}|_{t=\infty} - e^{A^T t} Q e^{At}|_{t=0} = -Q, \tag{B.6}$$

where the last equality holds because $e^{A\infty} = 0$ (recall A is Hurwitz).

To show the uniqueness of P, we assume that there exists another matrix P' that also satisfies the Lyapunov equation. Therefore,

$$P' = e^{A^T t} P' e^{At}|_{t=0} - e^{A^T t} P' e^{At}|_{t=\infty} \tag{B.7}$$

$$= - \int_{t=0}^{\infty} d\left(e^{A^T t} P' e^{At} \right) \tag{B.8}$$

$$= - \int_{t=0}^{\infty} e^{A^T t} \left(A^T P' + P' A \right) e^{At} dt \tag{B.9}$$

$$= \int_{t=0}^{\infty} e^{A^T t} Q e^{At} dt = P, \tag{B.10}$$

leading to $P' = P$. Hence, the solution is unique. $\square$

Convergence rate estimation. We now show that Theorem B.3 can allow us to quantify the convergence rate of a (stable) linear system towards zero.

For a Hurwitz linear system $\dot{x} = Ax$, let us pick a positive definite matrix Q. Theorem B.3 tells us we can find a unique $P \succ 0$ satisfying the Lyapunov equation (B.3). In this case, we can upper bound the scalar function $V = x^T P x$ as

$$V \leq \lambda_{\max}(P) \|x\|^2.$$

The time derivative of V is $\dot{V} = -x^T Q x$, which can be upper bounded by

$$\dot{V} \leq -\lambda_{\min}(Q) \|x\|^2 \tag{B.11}$$

$$= -\frac{\lambda_{\min}(Q)}{\lambda_{\max}(P)} \underbrace{\left(\lambda_{\max}(P) \|x\|^2 \right)}_{\geq V} \tag{B.12}$$

$$\leq -\frac{\lambda_{\min}(Q)}{\lambda_{\max}(P)} V. \tag{B.13}$$

Denoting $\gamma(Q) = \frac{\lambda_{\min}(Q)}{\lambda_{\max}(P)}$, the above inequality implies

$$V(0) e^{-\gamma(Q) t} \geq V(t) = x^T P x \geq \lambda_{\min}(P) \|x\|^2.$$

As a result, $\|x\|^2$ converges to zero exponentially with a rate at least $\gamma(Q)$, and $\|x\|$ converges to zero exponentially with a rate at least $\gamma(Q)/2$.

Best convergence rate estimation. I have used $\gamma(Q)$ to make it explict that the rate γ depends on the choice of Q, because P is computed from the Lyapunov equation as an implicit function of Q. Naturally, choosing different Q will lead to different $\gamma(Q)$. So what is the choice of Q that maximizes the convergence rate estimation?

Corollary B.1 (Maximum Convergence Rate Estimation). *$Q = I$ maximizes the convergence rate estimation.*

Proof. let us denote P_0 as the solution to the Lyapunov equation with $Q = I$

$$A^T P_0 + P_0 A = -I.$$

Let P be the solution corresponding to a different choice of Q

$$A^T P + PA = -Q.$$

Without loss of generality, we can assume $\lambda_{\min}(Q) = 1$, because rescaling Q will recale P by the same factor, which does not affect $\gamma(Q)$. Subtracting the two Lyapunov equations above we get

$$A^T(P - P_0) + (P - P_0)A = -(Q - I).$$

Since $Q - I \succeq 0$ (due to $\lambda_{\min}(Q) = 1$), we know $P - P_0 \succeq 0$ and $\lambda_{\max}(P) \geq \lambda_{\max}(P_0)$. As a result,

$$\gamma(Q) = \frac{\lambda_{\min}(Q)}{\lambda_{\max}(P)} = \frac{\lambda_{\min}(I)}{\lambda_{\max}(P)} \leq \frac{\lambda_{\min}(I)}{\lambda_{\max}(P_0)} = \gamma(I),$$

and $Q = I$ maximizes the convergence rate estimation. □

B.2 Controllability and Observability

Consider the following linear time-invariant (LTI) system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{B.14}$$

where $x \in \mathbb{R}^n$ the state, $u \in \mathbb{R}^m$ the control input, $y \in \mathbb{R}^p$ the output, and A, B, C, D are constant matrices with proper sizes. If we know the initial state

$x(0)$ and the control inputs $u(t)$ over a period of time $t \in [0, t_1]$, the system trajectory $(x(t), y(t))$ can be determined as

$$\begin{aligned} x(t) &= e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{B.15}$$

To study the internal structure of linear systems, two important properties should be considered: controllability and observability. In the following analysis, we will see that they are actually dual concepts. Their definitions (Chen, 1984) are given below.

Definition B.2 (Controllability)**.** The LTI system (B.14), or the pair (A, B), is controllable, if for any initial state $x(0) = x_0$ and final state x_f, there exists a sequence of control inputs that transfer the system from x_0 to x_f in finite time.

Definition B.3 (Observability)**.** The LTI system (B.14), or the pair (C, A), is observable, if for any unknown initial state $x(0)$, there exists a finite time $t_1 > 0$, such that knowing y and u over $[0, t_1]$ suffices to determine $x(0)$.

Sometimes it will become more convenient for us to analyze the system (B.14) under another coordinate basis, i.e., $z = Tx$, where the coordinate transformation T is nonsingular (i.e., full-rank). Define $A' = TAT^{-1}, B' = PB, C' = CT^{-1}, D' = D$, we get

$$\begin{aligned} \dot{z} &= A'z + B'u \\ y &= C'z + D'u \end{aligned}$$

Since the coordinate transformation only changes the system's coordinate basis, physical properties like controllability and observability will not change.

B.2.1 Cayley-Hamilton Theorem

In the analysis of controllability and observability, Cayley Hamilton Theorem lays the foundation. The statement of the theory and its (elegant) proof are given blow. Some useful corollaries are also presented.

Theorem B.4 (Cayley-Hamilton)**.** *Let $A \in \mathbb{C}^{n\times n}$ and denote the characteristic polynomial of A as*

$$det(\lambda I - A) = \lambda^n + a_1\lambda^{n-1} + \cdots + a_n \in \mathbb{C}[\lambda],$$

which is a polynomial in a single variable λ with coefficients $a_1, \dots, a_n$. Then

$$A^n + a_1A^{n-1} + \cdots + a_nI = 0$$

Proof. Define the adjugate of $\lambda I - A$ as

$$B = \operatorname{adj}(\lambda I - A)$$

From B's definition, we have

$$(\lambda I - A)B = \det(\lambda I - A)I = (\lambda^n + a_1\lambda^{n-1} + \cdots + a_n)I \tag{B.16}$$

Also, B is a polynomial matrix over λ, whose maximum degree is no more than $n-1$. Therefore, we write B as follows:

$$B = \sum_{i=0}^{n-1} \lambda^i B_i$$

where B_i's are constant matrices. In this way, we unfold $(\lambda I - A)B$:

$$\begin{aligned}(\lambda I - A)B &= (\lambda I - A)\sum_{i=0}^{n-1} \lambda^i B_i \\ &= \lambda^n B_{n-1} + \sum_{i=1}^{n-1} \lambda^i(-AB_i + B_{i-1}) - AB_0\end{aligned} \tag{B.17}$$

Since λ can be arbitrarily set, matching the coefficients of (B.16) and (B.17), we have

$$\begin{aligned} B_{n-1} &= I \\ -AB_i + B_{i-1} &= a_{n-i}I, \quad i = 1 \dots n-1 \\ -AB_0 &= a_n I \end{aligned}$$

Thus, we have

$$\begin{aligned} &B_{n-1} \cdot A^n + \sum_{i=1}^{n-1}(-AB_i + B_{i-1}) \cdot A^i + (-AB_0) \cdot I \\ =&I \cdot A^n + \sum_{i=1}^{n-1}(a_{n-i}I) \cdot A^i + (a_n I) \cdot I \\ =&A^n + a_1 A^{n-1} + a_2 A^{n-2} + \cdots + a_n I \end{aligned}$$

On the other hand, one can easily check that

$$B_{n-1} \cdot A^n + \sum_{i=1}^{n-1}(-AB_i + B_{i-1}) \cdot A^i + (-AB_0) \cdot I = 0$$

since each term offsets completely. Therefore,

$$A^n + a_1 A^{n-1} + a_2 A^{n-2} + \cdots + a_n I = 0,$$

concluding the proof. □

Here are some corollaries of the Cayley-Hamilton Theorem.

Corollary B.2. *For any $A \in \mathbb{C}^{n\times n}, B \in \mathbb{C}^{n\times m}, k \geq n$, A^kB is a linear combination of $B, AB, A^2B, \dots, A^{n-1}B$.*

Proof. Directly from Cayley Hamilton Theorem, A^n can be expressed as a linear combination of $I, A, A^2, \dots, A^{n-1}$. By recursion, it is easy to show that for all $m > n$, A^m is also a linear combination of $I, A, A^2, \dots, A^{n-1}$. Post-multiply both sides with B, we get what we want. □

Corollary B.3. *For any $A \in \mathbb{C}^{n\times n}, B \in \mathbb{C}^{n\times m}, k > n$, the following equality always holds:*

$$rank(\begin{bmatrix}B & AB & \dots & A^{n-1}B\end{bmatrix}) = rank(\begin{bmatrix}B & AB & \dots & A^{k-1}B\end{bmatrix})$$

Proof. First prove LHS $\leq$ RHS. $\forall v \in \mathbb{C}^n$ such that

$$v^* \begin{bmatrix}B & AB & \dots & A^{k-1}B\end{bmatrix} = v^* \begin{bmatrix}B & AB & \dots & A^{n-1}B & \dots A^{k-1}B\end{bmatrix} = 0$$

$v^* \begin{bmatrix}B & AB & \dots & A^{n-1}B\end{bmatrix} = 0$ must hold.

Second prove LHS $\geq$ RHS. For any $v \in \mathbb{C}^n$ such that $v^* \begin{bmatrix}B & AB & \dots & A^{n-1}B\end{bmatrix} = 0$ and any $k > n$, by Corollary B.2, there exists a sequence $c_i, i = 0 \dots n-1$ satisfy the following:

$$v^*A^kB = v^* \sum_{i=0}^{n-1} c_i A^i B = 0$$

Therefore, $v^* \begin{bmatrix}B & AB & \dots & A^{k-1}B\end{bmatrix} = 0$. □

Corollary B.4. *For any $A \in \mathbb{C}^{n\times n}, B \in \mathbb{C}^{n\times m}$, define*

$$\mathcal{C} = \begin{bmatrix}B & AB & \dots & A^{n-1}B\end{bmatrix}$$

If $rank(\mathcal{C}) = k_1 < n$, there exist a similarity transformation T such that

$$TAT^{-1} = \begin{bmatrix}\bar{A}_c & \bar{A}_{12} \\ 0 & \bar{A}_{\bar{c}}\end{bmatrix}, TB = \begin{bmatrix}\bar{B}_c \\ 0\end{bmatrix}$$

where $\bar{A}_c \in \mathbb{C}^{k_1\times k_1}, \bar{B}_c \in \mathbb{C}^{k_1\times m}$. Moreover, the matrix

$$\bar{\mathcal{C}} := \begin{bmatrix}\bar{B}_c & \bar{A}_c\bar{B}_c & \bar{A}_c^2\bar{B}_c & \dots & \bar{A}_c^{k_1-1}\bar{B}_c\end{bmatrix}$$

has full row rank.

Proof. Since $\mathcal{C}$ is not full row rank, we pick k_1 linearly independent columns from $\mathcal{C}$. Denote them as $q_1 \dots q_{k_1}$, $q_i \in \mathbb{C}^n$. Then, we arbitrarily set other $n-k_1$ vectors $q_{k_1+1} \dots q_n$ as long as

$$Q = \begin{bmatrix}q_1 & \dots & q_{k_1} & q_{k_1+1} & \dots & q_n\end{bmatrix}$$

is invertible. Define the similarity transformation matrix by $T = Q^{-1}$. Note that Aq_i can be seen as a column picked from $A^k B, k \in \{1 \dots n\}$, which is guaranteed to be a linear combination of $B, AB, \dots, A^{n-1}B$ from Cayley Hamilton Theorem. Thus, Aq_i is bound to be a linear transformation of columns from $\begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} = \mathcal{C}$. Since $q_1 \dots q_{k_1}$ is the largest linearly independent column vector set from $\mathcal{C}$, this implies Aq_i can be expressed as a linear combination of $q_1 \dots q_{k_1}$:

$$\begin{aligned} AQ = AT^{-1} &= A \begin{bmatrix} q_1 & \dots & q_{k_1} & q_{k_1+1} & \dots & q_n \end{bmatrix} \\ &= \begin{bmatrix} q_1 & \dots & q_{k_1} & q_{k_1+1} & \dots & q_n \end{bmatrix} \begin{bmatrix} \bar{A}_c & \bar{A}_{12} \\ 0 & \bar{A}_{\bar{c}} \end{bmatrix} = T^{-1} \begin{bmatrix} \bar{A}_c & \bar{A}_{12} \\ 0 & \bar{A}_{\bar{c}} \end{bmatrix} \end{aligned}$$

Similarly, B itself is part of $\mathcal{C}$. Therefore, each column of B is naturally a linear combination of $q_1 \dots q_{k_1}$:

$$B = \begin{bmatrix} q_1 & \dots & q_{k_1} & q_{k_1+1} & \dots & q_n \end{bmatrix} \begin{bmatrix} \bar{B}_c \\ 0 \end{bmatrix} = T^{-1} \begin{bmatrix} \bar{B}_c \\ 0 \end{bmatrix}$$

To see $\bar{\mathcal{C}}$ has full row rank, note that $\text{rank}\mathcal{C} = k_1$ and

$$\mathcal{C} = T^{-1} \begin{bmatrix} \bar{B}_c & \bar{A}_c\bar{B}_c & \bar{A}_c^2\bar{B}_c & \dots & \bar{A}_c^{k_1-1}\bar{B}_c & \dots & \bar{A}_c^{n-1}\bar{B}_c \\ 0 & 0 & 0 & \dots & 0 & \dots & 0 \end{bmatrix}$$

Thus,

$$\text{rank} \begin{bmatrix} \bar{B}_c & \bar{A}_c\bar{B}_c & \bar{A}_c^2\bar{B}_c & \dots & \bar{A}_c^{k_1-1}\bar{B}_c & \dots & \bar{A}_c^{n-1}\bar{B}_c \end{bmatrix} = k_1.$$

By Corollary B.3, $\text{rank}\bar{\mathcal{C}} = k_1$. □

The following Corollary is especially useful in the study of pole assignment in the single-input-multiple-output (SIMO) LTI system.

Corollary B.5. *For any $A \in \mathbb{C}^{n\times n}, b \in \mathbb{C}^n$, if*

$$\mathcal{C} = \begin{bmatrix} b & Ab & \dots & A^{n-1}b \end{bmatrix} \in \mathbb{C}^{n\times n}$$

has full rank, then there exists a similarity transformation T such that

$$TAT^{-1} = A_1 := \begin{bmatrix} -a_1 & -a_2 & \dots & -a_{n-1} & -a_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}, \quad Tb = b_1 := \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where $a_1, \dots, a_n$ are the coefficients of A's characteristic polynomial:

$$\det(A - \lambda I) = \lambda^n + a_1\lambda^{n-1} + \dots + a_n\lambda$$

Proof. Since $\mathcal{C}$ is invertible, define its inverse

$$\mathcal{C}^{-1} = \begin{bmatrix} M_1 \\ M_2 \\ \dots \\ M_n \end{bmatrix}$$

where $M_i \in \mathbb{C}^{1\times n}$. Then,

$$I = \mathcal{C}^{-1}\mathcal{C} = \begin{bmatrix} M_1 b & M_1 Ab & \dots & M_1 A^{n-1} b \\ M_2 b & M_2 Ab & \dots & M_2 A^{n-1} b \\ \vdots & \vdots & & \vdots \\ M_n b & M_n Ab & \dots & M_n A^{n-1} b \end{bmatrix} \Longrightarrow \begin{cases} M_n A^{n-1} b = 1 \\ M_n A^i b = 0, \ i = 0, \dots, n-2 \end{cases}$$

Now we claim that the transformation matrix T can be constructed as follows:

$$T = \begin{bmatrix} M_n A^{n-1} \\ M_n A^{n-2} \\ \dots \\ M_n \end{bmatrix}$$

We first show T is invertible by calculating $T\mathcal{C}$:

$$T\mathcal{C} = \begin{bmatrix} M_n A^{n-1} b & \star & \dots & \star \\ M_n A^{n-2} b & M_n A^{n-1} b & \dots & \star \\ \vdots & \vdots & & \vdots \\ M_n b & M_n Ab & \dots & M_n A^{n-1} b \end{bmatrix} = \begin{bmatrix} 1 & \star & \dots & \star \\ 0 & 1 & \dots & \star \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Then we calculate Tb and TA:

$$Tb = \begin{bmatrix} M_n A^{n-1} b \\ M_n A^{n-2} b \\ \vdots \\ M_n b \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$TA = \begin{bmatrix} M_n A^n \\ M_n A^{n-1} \\ \vdots \\ M_n A \end{bmatrix} = \begin{bmatrix} -M_n \cdot \sum_{i=0}^{n-1} a_{n-i} A^i \\ M_n A^{n-1} \\ \vdots \\ M_n A \end{bmatrix}$$

$$= \begin{bmatrix} -a_1 & -a_2 & \dots & -a_{n-1} & -a_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} M_n A^{n-1} \\ M_n A^{n-2} \\ \vdots \\ M_n A \\ M_n \end{bmatrix} = A_1 T$$

where the penultimate equality uses Cayley Hamilton Theorem. □

B.2.2 Equivalent Statements for Controllability

There are a few equivalent statements to express an LTI system's controllability that one should be familiar with:

Theorem B.5 (Equivalent Statements for Controllability)**.** *The following statements are equivalent (Chen, 1984), (Zhou et al., 1996):*

1. (A, B) *is controllable.*
2. *The matrix*
$$W_c(t) := \int_0^t e^{A\tau} BB^* e^{A^*\tau} d\tau$$
is positive definite for any $t > 0$.
3. *The controllability matrix*
$$\mathcal{C} = [B \quad AB \quad A^2B \quad ... \quad A^{n-1}B]$$
has full row rank.
4. *The matrix* $[A - \lambda I, B]$ *has full row rank for all* $\lambda \in \mathbb{C}$.
5. *Let* λ *and* x *be any eigenvalue and any corresponding left eigenvector* A, *i.e.,* $x^* A = x^* \lambda$, *then* $x^* B \neq 0$.
6. *The eigenvalues of* $A+BF$ *can be freely assigned (with the restriction that complex eigenvalues are in conjugate pairs) by a suitable choice of* F.
7. *If, in addition, all eigenvalues of* A *have negative real parts, then the unique solution of*
$$AW_c + W_c A^* = -BB^*$$
is positive definite. The solution is called the controllability Gramian *and can be expressed as*
$$W_c = \int_0^\infty e^{A\tau} BB^* e^{A^*\tau} d\tau$$

Proof. (1. $\Rightarrow$ 2.) Prove by contradiction. Assume that (A, B) is controllable but $W_c(t_1)$ is singular for some $t_1 > 0$. This implies there exists a real vector $v \neq 0 \in \mathbb{R}^n$, s.t.

$$v^* W_c(t_1) v = v^* (\int_0^{t_1} e^{At} BB^* e^{A^*t} dt) v = \int_0^{t_1} v^* (e^{At} BB^* e^{A^*t}) v \; dt = 0$$

Since $e^{At} BB^* e^{A^*t} \succeq 0$ for all t, we must have

$$\begin{aligned} & v^* (e^{At} BB^* e^{A^*t}) v = \| v^* B e^{At} \|^2 = 0, \quad \forall t \in [0, t_1] \\ \Longrightarrow & v^* B e^{At} = 0, \quad \forall t \in [0, t_1] \end{aligned}$$

Setting $x(t_1) = 0$, from (B.15), we have

$$0 = e^{At_1}x(0) + \int_0^{t_1} e^{A(t_1-\tau)}Bu(\tau)d\tau = 0$$

Pre-multiply the above equation by v^*, then

$$0 = v^* e^{At_1}x(0)$$

Since $x(0)$ can be chosen arbitrarily, we set $x(0) = ve^{-At_1}$, which results in $v = 0$. Contradiction!

(2. $\Rightarrow$ 1.) For any $x(0) = x_0, t_1 > 0, x(t_1) = x_1$, since $W_c(t_1) \succ 0$, we set the control inputs as

$$u(t) = -B^* e^{A^*(t_1-t)} W_c^{-1}(t_1)[e^{At_1}x_0 - x_1]$$

We claim that the picked $u(t)$ satisfies (B.15) by

$$\begin{aligned}
&e^{At}x_0 + \int_0^{t_1} e^{A(t_1-t)}Bu(t)dt \\
&= e^{At}x_0 - \int_0^{t_1} e^{A(t_1-t)}BB^*e^{A^*(t_1-t)}dt \cdot W_c^{-1}(t_1)[e^{At_1}x_0 - x_1] \\
&\overset{\tau=t_1-t}{=} e^{At}x_0 - \underbrace{\int_0^{t_1} e^{A\tau}BB^*e^{A^*\tau}d\tau}_{W_c(t_1)} \cdot W_c^{-1}(t_1)[e^{At_1}x_0 - x_1] \\
&= e^{At}x_0 - [e^{At_1}x_0 - x_1] = x_1
\end{aligned}$$

(2. $\Rightarrow$ 3.) Prove by contradiction. Suppose $W_c(t) \succ 0, \forall t > 0$ but $\mathcal{C}$ is not of full row rank. Then there exists $v \neq 0 \in \mathbb{C}^n$, s.t.

$$v^* A^k B = 0, \quad k = 0 \dots n-1$$

By Corollary B.2, we have

$$v^* A^k B = 0,\ \forall k \in \mathbb{N} \Longrightarrow v^* e^{At}B = 0,\ \forall t > 0$$

which implies

$$v^* W_c(t) v = v^* (\int_0^t e^{A\tau}BB^*e^{A^*\tau}d\tau) v = 0, \quad \forall t > 0$$

Contradiction!

(3. $\Rightarrow$ 2.) Prove by contradiction. Suppose $\mathcal{C}$ has full row rank but $W_c(t_1)$ is singular at some $t_1 > 0$. Then, similar to the proof in (1. $\Rightarrow$ 2.), there exists $v \neq 0 \in \mathbb{C}^n$, s.t. $F(t) := v^* e^{At}B \equiv 0, \forall t \in [0, t_1]$. Since $F(t)$ is infinitely

differentiable, we get its i's derivative at $t = 0$, where $i = 0, 1, \dots n - 1$. This results in

$$\left.\frac{d^i F}{dt^i}\right|_{t=0} = v^* A^i e^{At} B\big|_{t=0} = v^* A^i B = 0, \quad i = 0 \dots n - 1$$

Thus, $v^* \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} = 0$. Contradiction!

(3. $\Rightarrow$ 4.) Proof by contradiction. Suppose $[A - \lambda I, B]$ does not have full row rank for some $\lambda \in \mathbb{C}$. Then, there exists $v \neq 0 \in \mathbb{C}^n$, s.t. $v^*[A - \lambda I, B] = 0$. This implies $v^* A = v^* \lambda$ and $v^* B = 0$. On the other hand,

$$v^* \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} = v^* \begin{bmatrix} B & \lambda B & \dots & \lambda^{n-1}B \end{bmatrix} = 0$$

Contradiction!

(4. $\Rightarrow$ 5.) Proof by contradiction. If there exists a left eigenvector and eigenvalue pair (x, λ), s.t. $x^* A = \lambda x^*$ while $x^* B = 0$, then $x^*[A - \lambda I, B] = 0$. Contradiction!

(5. $\Rightarrow$ 3.) Proof by contradiction. If the controllability matrix $\mathcal{C}$ does not have full row rank, i.e., $\mathrm{rank}(\mathcal{C}) = k < n$. Then, from Corollary B.4, there exists a similarity transformation T, s.t.

$$TAT^{-1} = \begin{bmatrix} \bar{A}_c & \bar{A}_{12} \\ 0 & \bar{A}_{\bar{c}} \end{bmatrix}, \quad TB = \begin{bmatrix} \bar{B}_c \\ 0 \end{bmatrix}$$

where $\bar{A}_c \in \mathbb{R}^{k \times k}, \bar{A}_{\bar{c}} \in \mathbb{R}^{(n-k)\times(n-k)}$. Now arbitrarily pick one of $\bar{A}_{\bar{c}}$'s left eigenvector $x_{\bar{c}}$ and its corresponding eigenvalue λ_1. Define the vector $x = \begin{bmatrix} 0 \\ x_{\bar{c}} \end{bmatrix}$. Then,

$$x^*(TAT^{-1}) = \begin{bmatrix} 0 & x_{\bar{c}}^* \end{bmatrix} \begin{bmatrix} \bar{A}_c & \bar{A}_{12} \\ 0 & \bar{A}_{\bar{c}} \end{bmatrix} = \begin{bmatrix} 0 & x_{\bar{c}}^* \bar{A}_{\bar{c}} \end{bmatrix} = \begin{bmatrix} 0 & \lambda_1 x_{\bar{c}}^* \end{bmatrix} = \lambda_1 x^*$$

$$x^*(TB) = \begin{bmatrix} 0 & x_{\bar{x}} \end{bmatrix} \begin{bmatrix} B_{\bar{c}} \\ 0 \end{bmatrix} = 0$$

which implies (TAT^{-1}, TB) is not controllable. However, similarity transformation does not change controllability. Contradiction!

(6. $\Rightarrow$ 1.) Prove by contradiction. If (A, B) is not controllable, i.e., $\mathrm{rank}(\mathcal{C}) = k < n$. Then from Corollary B.4, there exists a similarity transformation T s.t.

$$TAT^{-1} = \begin{bmatrix} \bar{A}_c & \bar{A}_{12} \\ 0 & \bar{A}_{\bar{c}} \end{bmatrix}, \quad TB = \begin{bmatrix} \bar{B}_c \\ 0 \end{bmatrix}$$

Now arbitrarily pick $F \in \mathbb{R}^{m \times n}$ and define $FT^{-1} = [F_1, F_2]$, where $F_1 \in$

$\mathbb{R}^{m\times k}, F_2 \in \mathbb{R}^{m\times(n-k)}$. Thus,

$$\begin{aligned}
\det(A+BF-\lambda I) &= \det\left(T^{-1}\begin{bmatrix}\bar{A}_c & \bar{A}_{12}\\ 0 & \bar{A}_{\bar{c}}\end{bmatrix}T + T^{-1}\begin{bmatrix}\bar{B}_c\\ 0\end{bmatrix}F - \lambda\begin{bmatrix}I_1 & 0\\ 0 & I_2\end{bmatrix}\right)\\
&= \det\left(T^{-1}\left\{\begin{bmatrix}\bar{A}_c & \bar{A}_{12}\\ 0 & \bar{A}_{\bar{c}}\end{bmatrix} + \begin{bmatrix}\bar{B}_c\\ 0\end{bmatrix}FT^{-1} - \lambda\begin{bmatrix}I_1 & 0\\ 0 & I_2\end{bmatrix}\right\}T\right)\\
&= \det\left(\begin{bmatrix}\bar{A}_c & \bar{A}_{12}\\ 0 & \bar{A}_{\bar{c}}\end{bmatrix} + \begin{bmatrix}\bar{B}_c\\ 0\end{bmatrix}\begin{bmatrix}F_1 & F_2\end{bmatrix} - \lambda\begin{bmatrix}I_1 & 0\\ 0 & I_2\end{bmatrix}\right)\\
&= \det\begin{bmatrix}\bar{A}_c + \bar{B}_cF_1 - \lambda I_1 & \bar{A}_{12} + \bar{B}_cF_2\\ 0 & \bar{A}_{\bar{c}} - \lambda I_2\end{bmatrix}\\
&= \det(\bar{A}_c + \bar{B}_cF_1 - \lambda I_1)\cdot\det(\bar{A}_{\bar{c}} - \lambda I_2)
\end{aligned}$$

where I_1 is the identity matrix of size k. Similarly, I_2 of size $n-k$. Thus, at least $n-k$ eigenvalues of $A+BF$ cannot be freely assigned by choosing F. Contradiction!

(1. $\Rightarrow$ 6.) Here we only represent the SIMO case. For the MIMO case, the proof is far more complex. Interesting readers can refer to (Davison and Wonham, 1968) (the shortest proof I can find). Since there is only one input, the matrix B degenerate to vector b. From Corollary B.5, there exist a similarity transformation matrix T, s.t.

$$TAT^{-1} = A_1 := \begin{bmatrix}-a_1 & -a_2 & \dots & -a_{n-1} & -a_n\\ 1 & 0 & \dots & 0 & 0\\ 0 & 1 & \dots & 0 & 0\\ \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & \dots & 1 & 0\end{bmatrix}, \quad Tb = b_1 := \begin{bmatrix}1\\0\\0\\ \vdots\\0\end{bmatrix}$$

For any $F \in \mathbb{C}^{1\times n}$, denote FT^{-1} as $[f_1, f_2, \dots, f_n]$. Calculating the characteristic polynomial of $A+bF$:

$$\begin{aligned}
\det(\lambda I - A - bF) &= \det(\lambda I - T^{-1}A_1T - T^{-1}b_1F)\\
&= \det(\lambda I - A_1 - b_1FT^{-1})\\
&= \det\begin{bmatrix}\lambda + a_1 - f_1 & \lambda + a_2 - f_2 & \dots & \lambda + a_{n-1} - f_{n-1} & \lambda + a_n - f_n\\ -1 & \lambda & \dots & 0 & 0\\ 0 & -1 & \dots & 0 & 0\\ \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & \dots & -1 & \lambda\end{bmatrix}\\
&= \lambda^n + (a_1 - f_1)\lambda^{n-1} + \dots + (a_n - f_n)
\end{aligned}$$

By choosing $[f_1, f_2, \dots, f_n]$, $A+bF$'s eigenvalues can be arbitrarily set.

(7. $\Rightarrow$ 1.) Prove by contradiction. Assume that (A, B) is not controllable. Then from 2., there exists $v \neq 0 \in \mathbb{C}^n$ and $t_1 > 0$,

$$F(t) = v^* e^{At} B = 0, \quad \forall t \in [0, t_1]$$

Now consider $F(z) = v^* e^{Az} B, z \in \mathcal{C}$, which is a vector of analytic function in complex analysis. For a arbitrary $t_2 \in (0, t_1)$, we have $F^{(i)}(t_2) = 0, \forall i \in \mathbb{N}$. Then, by invoking the fact from complex analysis: "Let G a connected open set and $f : G \to \mathbb{C}$ be analytic, then $f \equiv 0$ on G, if and only if there is a point $a \in G$ such that $f^{(i)}(a) = 0, \forall n \in \mathbb{N}$", we have $f(z) \equiv 0, \forall z \in \mathbb{C}$.

On the other hand, however, $W_c \succ 0$ implies there exists $t_3 > 0$, such that for the above v, we have $v^* e^{At_3} B \neq 0$. Contradiction!

(1. $\Rightarrow$ 7.) Since (A, B) is controllable, from 2., $W_c(t) \succ 0, \forall t$. Therefore, $W_c \succ 0$. The existence and uniqueness of the solution for $AW_c + W_c A^* = -BB^*$ can be obtained directly from the proof of Theorem B.3, by setting Q there to be positive semidefinite. $\square$

B.2.3 Duality

Although controllability and observability seemingly have no direct connections from their definitions B.2 and B.3, the following theorem (Chen, 1984) states their tight relations.

Theorem B.6 (Theorem of Duality)**.** *The pair (C, A) is observable if and only if (A^*, C^*) is controllable.*

Proof.

(1) We first show that (C, A) is observable if and only if the $n \times n$ matrix $W_o(t) = \int_0^t e^{A^*\tau} C^* C e^{A\tau}$ is positive definite (nonsingular) for any $t > 0$:

"$\Longleftarrow$": From (B.15), given initial state $x(0)$ and the inputs $u(t)$, $y(t)$ can be expressed as

$$y(t) = Ce^{At}x(0) + C \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau + Du(t)$$

Define a known function $\bar{y}(t)$ as $y(t) - C \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau - Du(t)$ and we will get

$$Ce^{At}x(0) = \bar{y}(t)$$

Pre-multiply the above equation by $e^{A^*t} C^*$ and integrate it over $[0, t_1]$ to yield

$$(\int_0^{t_1} e^{A^*t} C^* C e^{At} dt) x(0) = W_o(t_1) x(0) = \int_0^{t_1} e^{A^*t} C^* \bar{y}(t) dt$$

Since $W_o(t_1) \succ 0$,

$$x(0) = W_o(t_1)^{-1} \int_0^{t_1} e^{A^*t} C^* \bar{y}(t) dt$$

can be observed.

"$\Longrightarrow$": Prove by contradiction. Suppose (C, A) is observable but there exists $t_1 > 0$, s.t. $W_o(t_1)$ is singular. This implies there exists $v \neq 0 \in \mathbb{C}^n$, s.t.

$$v^* W_o(t_1) v = 0 \Longrightarrow Ce^{At} v \equiv 0, \ \forall t \in [0, t_1]$$

Similar to the proof of Theorem B.5 (7. $\Rightarrow$ 1.), we can use conclusions from complex analysis to claim that $Ce^{At}v \equiv 0, \forall t > 0$. On the other hand, we set $u(t) \equiv 0$, which results in $y(t) = Ce^{At}x(0)$. In this case $x(0) = 0$ and $x(0) = v \neq 0$ will lead to the same output responses $y(t)$ over $t > 0$, which implies (C, A) is not observable. Contradiction!

(2) Next we show the duality of controllability and observability:

From (1) we know (C, A) is controllable if and only of

$$\int_0^t e^{A^*\tau} C^* C e^{A\tau} d\tau = \int_0^t e^{(A^*)\tau} (C^*)^* (C^*) e^{(A^*)^*\tau} d\tau$$

is nonsingular for all $t > 0$. The latter is exactly the definition of (A^*, C^*)'s controllability Gramian $W_c(t)$.

□

B.2.4 Equivalent Statements for Observability

With the Theorem of Duality B.6, we can directly write down the equivalent statements of observability without any additional proofs:

Theorem B.7 (Equivalent Statements for Observability)**.** *The following statements are equivalent (Chen, 1984), (Zhou et al., 1996):*

1. (C, A) *is observable.*
2. *The matrix*
$$W_o(t) := \int_0^t e^{A^*\tau} C^* C e^{A\tau} d\tau$$
is positive definite for any $t > 0$.
3. *The observability matrix*
$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \dots \\ CA^{n-1} \end{bmatrix}$$
has full column rank.

4. *The matrix* $\begin{bmatrix} A - \lambda I \\ C \end{bmatrix}$ *has full column rank for all* $\lambda \in \mathbb{C}$.

5. *Let* λ *and* y *be any eigenvalue and any corresponding right eigenvector of* A, *i.e.,* $Ay = \lambda y$, *then* $Cy \neq 0$.

6. *The eigenvalues of* $A + LC$ *can be freely assigned (with the restriction that complex eigenvalues are in conjugate pairs) by a suitable choice of* L.

7. (A^*, C^*) *is controllable.*

8. *If, in addition, all eigenvalues of* A *have negative parts, then the unique solution of*
$$A^* W_o + W_o A = -C^* C$$
is positive definite. The solution is called the observability Gramian *and can be expressed as*
$$W_o = \int_0^\infty e^{A^* \tau} C^* C e^{A\tau} d\tau$$

B.3 Stabilizability And Detectability

To define stabilizability and detectability of an LTI system, we first introduce the concept of *system mode*, which can be naturally derived from the fifth definition of controllability B.5 (observability B.7).

Definition B.4 (System Mode)**.** λ is a mode of an LTI system, if it is an eigenvalue of A. The mode λ is said to be:

- stable, if $\text{Re}\lambda < 0$,
- controllable, if $x^* B \neq 0$ for all left eigenvectors of A associated with λ,
- observable, if $Cx \neq 0$ for all right eigenvectors of A associated with λ.

Otherwise, the mode is said to be uncontrollable (unobservable).

With the concept of system mode, the fifth definition of controllability B.5 (observability B.7) can be restated as

> An LTI system is controllable (observable) if and only if all modes are controllable (observable).

Stabilizability (detectability) is defined similarly via loosening part of controllability (observability) conditions.

Definition B.5 (Stabilizability)**.** An LTI system is said to be stabilizable if all of its unstable modes are controllable.

Definition B.6 (Detectability)**.** An LTI system is said to be detectable if all of its unstable modes are observable.

Like in the case of controllability and observability, duality also holds in stabilizability and detectability. Moreover, similarity transformation will not influence an LTI system's stabilizability and detectability.

B.3.1 Equivalent Statements for Stabilizability

Theorem B.8 (Equivalent Statements for Stabilizability)**.** *The following statements are equivalent (Zhou et al., 1996):*

1. *(A, B) is stabilizable.*
2. *For all λ and x such that $x^*A = \lambda x^*$ and $Re\lambda \geq 0$, $x^*B \neq 0$.*
3. *The matrix $[A - \lambda I, B]$ has full rank for all $Re\lambda \geq 0$.*
4. *There exists a matrix F such that $A + BF$ are Hurwitz.*

Proof. (1. $\Leftrightarrow$ 2.) Directly from stabilizability's definition.

(2. $\Leftrightarrow$ 3.) If 2. holds but 3. not hold, then there exists $v \neq 0 \in \mathbb{C}^n$, s.t.

$$v^*[A - \lambda I, B] = 0 \Leftrightarrow v^*A = \lambda v^*, v^*B = 0, \mathrm{Re}\lambda \geq 0$$

Contradiction! Vice versa.

(4. $\Rightarrow$ 2.) Prove by contradiction. Suppose there $x \neq 0 \in \mathbb{C}^n$, s.t.

$$x^*[A - \lambda I, B] = 0 \Leftrightarrow x^*A = \lambda x^*, x^*B = 0, \mathrm{Re}\lambda \geq 0$$

Thus, for any F,

$$x^*(A + BF) = \lambda x^*, \mathrm{Re}\lambda \geq 0$$

On the other hand, suppose $A + BF$ has I Jordon blocks, with each equipped with an eigenvalue $\eta_i, i = 1 \dots I$ (note that η_α may be equal to η_β, i.e., they are equivalent eigenvalues with different Jordon blocks). Since $A + BF$'s eigenvalues all have negative real parts, $\mathrm{Re}(\eta_i) < 0, i = 1 \dots I$. For each $\eta_i, i \in \{1 \dots i\}$, denote its K_i generalized left eigenvectors as $v_{i,1}, v_{i,2}, \dots v_{i,K_i}$. By definition, $\sum_{i=1}^{I} K_i = n$ and

$$\begin{aligned} v_{i,1}^*(A + BF) &= v_{i,1}^* \cdot \eta_i \\ v_{i,2}^*(A + BF) &= v_{i,1}^* + v_{i,2}^* \cdot \eta_i \\ &\vdots \\ v_{i,K_i}^*(A + BF) &= v_{i,K_i-1}^* + v_{i,K_i}^* \cdot \eta_i \end{aligned}$$

for all $i \in \{1 \dots i\}$. Also, $v_{i,k}, i = 1 \dots I, k = 1 \dots K_i$ are linearly independent and spans $\mathbb{C}^n$. Therefore,

$$x^* = \sum_{i=1}^{I} \sum_{k=1}^{K_i} \xi_{i,k} \cdot v_{i,k}^*$$

which leads to

$$\sum_{i=1}^{I} \sum_{k=1}^{K_i} \xi_{i,k} \cdot v_{i,k}^*(A + BF) = \sum_{i=1}^{I} \sum_{k=1}^{K_i} \xi_{i,k} \cdot \lambda \cdot v_{i,k}^*$$

Since $v_{i,k}$'s are $A + BF$'s generalized eigenvectors, we have

$$\begin{aligned}
&\sum_{i=1}^{I} \sum_{k=1}^{K_i} \xi_{i,k} \cdot v_{i,k}^* \cdot (A + BF) \\
=&\sum_{i=1}^{I} \left\{ \xi_{i,1} \cdot \eta_i \cdot v_{i,1}^* + \sum_{k=2}^{K_i} \xi_{i,k}(v_{i,k-1}^* + \eta_i \cdot v_{i,k}^*) \right\} \\
=&\sum_{i=1}^{I} \left\{ \sum_{k=1}^{K_i - 1} (\xi_{i,k} \cdot \eta_i + \xi_{i,k+1}) v_{i,k}^* + \xi_{i,K_i} \cdot \eta_i \cdot v_{i,K_i}^* \right\}
\end{aligned}$$

Combining the above two equations:

$$\sum_{i=1}^{I} \left\{ \sum_{k=1}^{K_i - 1} \left[\xi_{i,k} \cdot (\eta_i - \lambda) + \xi_{i,k+1}\right] v_{i,k}^* + \xi_{i,K_i} \cdot (\eta_i - \lambda) \cdot v_{i,K_i}^* = 0 \right\}$$

Since $v_{i,k}$'s are linearly independent, for any $i \in \{i \dots I\}$:

$$\begin{aligned}
\xi_{i,1} \cdot (\eta_i - \lambda) + \xi_{i,2} = 0 &\Rightarrow \xi_{i,2} = (-1) \cdot \xi_{i,1} \cdot (\eta_i - \lambda) \\
\xi_{i,2} \cdot (\eta_i - \lambda) + \xi_{i,3} = 0 &\Rightarrow \xi_{i,3} = (-1)^2 \cdot \xi_{i,1} \cdot (\eta_i - \lambda)^2 \\
&\vdots \\
\xi_{i,K_i - 1} \cdot (\eta_i - \lambda) + \xi_{i,K_i} = 0 &\Rightarrow \xi_{i,K_i} = (-1)^{K_i - 1} \cdot \xi_{i,1} \cdot (\eta_i - \lambda)^{K_i - 1} \\
\xi_{i,K_i} \cdot (\eta_i - \lambda) = 0 &
\end{aligned}$$

Thus,

$$(-1)^{K_i - 1} \cdot \xi_{i,1} \cdot (\eta_i - \lambda)^{K_i} = 0$$

Denote $\xi_{i,1}$ as $r_1 e^{\theta_1}$, $(\eta_i - \lambda)$ as $r_2 e^{\theta_2}$. Since $\mathrm{Re}\lambda \geq 0, \mathrm{Re}(\eta_i) < 0$, $r_2 > 0$. On the other hand, the following equation suggests

$$r_1 r_2^{K_i - 1} e^{j[\theta_1 + \theta_2 (K_i - 1)]} = 0$$

Thus, r_1 has to be 0, which implies $\xi_{i,1} = 0$. By recursion, $\xi_{i,k} = 0, \forall k = 1 \dots K_i$. Contradiction!

(1. $\Rightarrow$ 4.) If (A, B) is controllable, then from Theorem **??**(thm:lticontrollable)'s sixth definition, we can freely assign the poles of $A+BF$ via choosing F properly.

Otherwise, if (A, B) is uncontrollable, then from Corollary B.4 and proof of Theorem B.5 (6. $\Rightarrow$ 1.), there exists a similarity transformation T, s.t.

$$TAT^{-1} = \begin{bmatrix} \bar{A}_c & \bar{A}_{12} \\ 0 & \bar{A}_{\bar{c}} \end{bmatrix}, \quad TB = \begin{bmatrix} \bar{B}_c \\ 0 \end{bmatrix}$$

and

$$\det(A + BF - \lambda I) = \underbrace{\det(\bar{A}_c + \bar{B}_c F_1 - \lambda I_1)}_{\chi_c(\lambda)} \cdot \underbrace{\det(\bar{A}_{\bar{c}} - \lambda I_2)}_{\chi_{\bar{c}}(\lambda)}$$

where $\bar{A}_c \in \mathbb{C}^{k_1 \times k_1}$, I_1 identity matrix of size k_1, $[F_1, F_2] = FT^{-1}$, and $k_1 = \text{rank}\mathcal{C}$. Additionally, $(\bar{A}_c, \bar{B}_c)$ is controllable. Thus, $\chi_c(\lambda)$'s zeros can be freely assigned by choosing proper F, i.e., system modes with $\chi_c(\lambda)$ is controllable, regardless of its stability. On the other hand, system modes with $\chi_{\bar{c}}(\lambda)$ must be stable. Otherwise, we cannot affect it by assigning F, which is a contradiction to statement (1). Therefore, (TAT^{-1}, TB) is stabilizable. Since similarity transformation does not change stabilizability, (A, B) is stabilizable. $\square$

B.3.2 Equivalent Statements for Detectability

Thanks to duality, we can directly write down the equivalent statements of observability without any additional proofs:

Theorem B.9 (Equivalent Statements for Detectability). *The following statements are equivalent (Zhou et al., 1996):*

1. *(C, A) is detectable.*
2. *For all λ and x such that $Ax = \lambda x$ and $Re\lambda \geq 0$, $Cx \neq 0$.*
3. *The matrix $\begin{bmatrix} A - \lambda I \\ C \end{bmatrix}$ has full rank for all $Re\lambda \geq 0$.*
4. *There exists a matrix L such that $A + LC$ are Hurwitz.*
5. *(A^*, C^*) is stabilizable.*

Bibliography

Chen, C.-T. (1984). *Linear system theory and design*. Saunders college publishing.

Davison, E. and Wonham, W. (1968). On pole assignment in multivariable linear systems. *IEEE Transactions on Automatic Control*, 13(6):747–748.

Zhou, K., Doyle, J., and Glover, K. (1996). Robust and optimal control. *Control Engineering Practice*, 4(8):1189–1190.