lecture notes:

Online and Nonstochastic Control Theory

version 0.51

Elad Hazan

Contents

Notation

We use the following mathematical notation in this writeup:

Control

- The dynamics function is denoted by f
- State at time t: $\mathbf{x}_t \in \mathbb{R}^{d_x}$.
- Control at time t: $\mathbf{u}_t \in \mathbb{R}^{d_u}$.
- perturbation at time t: $\mathbf{w}_t \in \mathbb{R}^{d_w}$
- Observation at time t: $\mathbf{y}_t \in \mathbb{R}^{d_y}$
- Dimensions of states, controls, perturbations and observations: d_x, d_u, d_w, d_y

Reinforcement Learning

- An MDP is parametrized by states, actions, discount factor, reward function and transitions denoted S, A, γ, R, P. Starting state is s_0.
- Reward at time t, random variable, denoted R_t
- $\mathbf{P}^a_{ss'}$ is the probability of transitioning from s to s' given action a.
- The value and Q functions are denoted $\mathbf{v}(s)$, $Q(s, a)$.
- A policy is denoted $\pi : S \mapsto A$.

Chapter 1

Introduction to Control

1.1 What is this course about?

Deep learning pioneer and Turing award winner professor Yann Lecun has a tallent of nailing down the crux of the matter very succinctly. The following quote he said in Barbados workshop, circa Feb 2020,

> "Control = reinforcement learning with gradients".

The key difference between Reinforement Learning and Control Theory is the spaces on which they operate. While the field of Reinforcement Learning typically lies within discrete state spaces (e.g. the games of Go, Chess), Control Theory typically tackles problems involving physics and continuous spaces (e.g. robotics). The knowledge of physics and structured environments allows us to make use of differential information.

The latter allows us to use the powerful technology of mathematical optimization and convex relaxations to design efficient algorithms. This is the topic of adaptive non-stochastic control theory, and this course.

1.2 The history of control

The industrial revolution was in many ways shaped by the steam engine that came to prominence in the 18th century. One of its most critical components was introduced by the Scottish inventor James Watt: one of the ealiest examples of a control mechanism, the Centrifugal Governor. One of the major losses in efficiency of earlier designs was due to the difficulty in regulating the engine. Too much or too little steam could easily result in engine failure. The Centrifugal Governor was created to solve this problem.

Figure 1.1: A Centrifugal Governor, source: Wikipedia

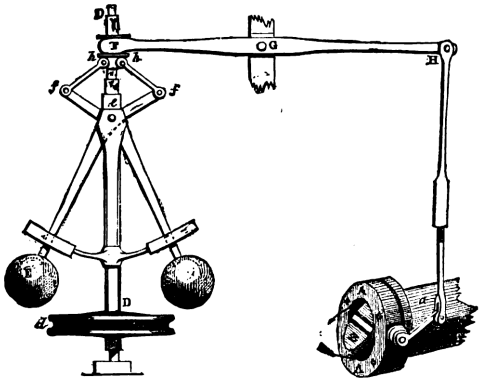


FIG. 4.—*Governor and Throttle-Valve.*

The Centrifugal Governor consists of a shaft connected to two metal balls that are allowed to rotate. Increased shaft speed causes the balls to spin faster, and the resulting centrifugal force causes the balls to lift. The balls are in turn attached to a pin that controls the amount of steam entering the engine, so as the balls spin faster, the pin lifts and limits the amount of fuel that can enter the system. On the other hand, if the balls spin slower, the pin lowers, allowing more steam to enter the system. This provides a natural, direct and proportional, control to the system.

The impact of the centrifugal governor for the steam engine, and in turn the industrial revolution, illustrates the importance of the field of control in engineering and the sciences. Since the 18th century our world has changed quite a bit, and the importance of control has only grown.

The challenges of controlling systems has moved from the steam engine to design of aircraft, robotics and autonomous vehicles, but the importance of designing efficient and robust methods remains.

1.3 Formalizing the optimal control problem

With the physical example of the centrifugal governor in mind, we now formally define the Control Problem. First, we define a dynamical system.

Definition 1.1 (Dynamical system)**.** *A dynamical system is given by the following equations:*

$$x_{t+1} = f_t(x_t, u_t, w_t), \quad y_t = g(x_t).$$

Here $x_t \in \mathbb{R}^{d_x}$ is the state of the system, $y_t \in \mathbb{R}^{d_y}$ is the observation, $u_t \in \mathbb{R}^{d_u}$ is the control input, and $w_t \in \mathbb{R}^{d_w}$ is the perturbation.

The function g_t is a mapping from the state to the observed state, and the function f_t is the transition function, given current state x_t and control input u_t. The subscript t indicates the relevant quantities at time step t.

Given a dynamical system, the control problem is to minimize long-term horizon cost given by a sequence of cost functions:

Definition 1.2 (The optimal control problem)**.** *Given a dynamical system, the control problem is to iteratively construct controls u_t such to mininimize a sequence of loss functions $c_t(y_t, u_t)$, $c_t : \mathbb{R}^{d_y \times d_u} \to \mathbb{R}$. This is given by the following mathematical program*

$$\begin{aligned} &\min_{u_{1:T}} \sum_t c_t(x_t, u_t) \\ s.t. \quad &x_{t+1} = f_t(x_t, u_t, w_t), \quad y_t = g(x_t) \end{aligned}$$

1.3.1 Example: Medical Ventilators

A recent relevant example of control is in the case of medical ventilators. The ventilator connects trachea in the lungs to a central pressure chamber, which helps maintain positive-end expiratory pressure (PEEP) to prevent the lungs from collapsing and simulate healthy breathing in patients with respiratory problems.

The ventilator needs to take into account the structure of the lung to determine the optimal pressure to induce. Such structural factors include *compliance*, or the change in lung volume per unit pressure, and *resistance*, or the change in pressure per unit flow.

A simplistic formalization of the dynamical system of the ventilator and lung can be derived from the physics of a connected two-balloon system. The dynamics equations can be written as

$$v_{t+1} = v_t + u_t * \Delta_t$$

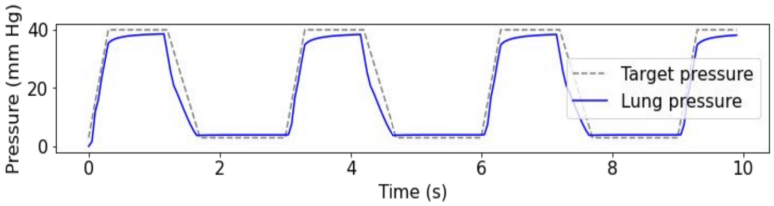
$$p_t = p_0 + \left(1 - \left(\frac{r_t}{r_0}\right)^6\right) \cdot \frac{1}{r_t r_0^2}, \quad r_t = \left(\frac{3v_t}{4\pi}\right)^{1/3},$$

where p_t is the measured pressure, v_t is volume, r_t is radius of the lung, u_t is the input flow (control) and Δ_t is the unit of time measured by the device precision. p_0 and r_0 are constants that depend on the particular system/lung-ventilator pair.

The goal of ventilator control is to regulate pressure according to some predetermined pattern. As a dynamical system, we can write $p_{t+1} = f(p_t, u_t, w_t)$, and we can define the cost function to be $c_t(p_t, u_t) = |p_t - p_t^*|^2$, where p_t^* is the desired pressure at time t. The goal, then, is to choose a sequence of control signals u_t to minimize the total cost over T iterations.

In the figure below we see an illustration of a simple controller, called the PID controller, as it performs on a a ventilator and lung simulator.

Figure 1.2: Medical Ventilator Performance of the PID controller



1.4 Simple Control Algorithms

We start by examining the most natural control algorithms, those based on linear functions.

For example, conider a control method that chooses the control to be a linear function of the observed states, i.e.

$$u_t = \sum_{i=0}^{k} A_i x_{t-i},$$

for some coefficients A_i, where above we consider only the $k+1$ most recent states. One of the most useful special cases of this linear family is the PID controller.

1.4.1 The PID Controller

A special sub-family of linear controllers is that of linear controllers that optimize over a pre-defined basis of historical observations. More precisely,

1. Proportional Control (e.g. Centrifugal Governor): $u_t = \alpha_0 x_t$.
2. Integrable Control: $u_t = \beta \sum_{i<t} x_{t-i}$.

3. Differential Control: $u_t = \gamma(x_t - x_{t-1})$.

The PID controller specifies three coefficients α, β, γ to generate a control signal. In that sense it is extremely sparse, which form a learning theoretic perspective is a good indication of generalization.

Indeed, the PID controller is the method of choice for numerous applications in engineering, including control of medical ventilator.

1.5 Computational hardness of the Control Problem

To motivate the future lessons in this course we start by showing that even basic problems in control are computationally hard. We start by the problem of finding the optimal linear controller.

Theorem 1.3. *It is NP-Hard to compute the optimal linear controller for a given control problem.*

The control problem with a linear controller, i.e. $u_t = \sum_{i=0}^{k} A_i x_{t-i}$, can be written as

$$\min \sum_t c_t \left(x_t, \sum_{i=0}^{k} A_i x_{t-i} \right), \quad x_{t+1} = f \left(x_t, \sum_{i=0}^{k} A_i x_{t-i} \right).$$

There are two sources of computational hardness for the above mathematical program, the general non-convex cost functions, and the dynamics function.

1.5.1 Computational hardness from optimizing costs

Even without dynamics, minimizing the costs for a general non-convex cost function is NP-hard. The following lemma and proof are taken from [Haz19b].

Lemma 1.4. *Mathematical optimization is NP-hard, even for a convex continuous constraint set $\mathcal{K}$ and quadratic objective functions.*

Informal sketch. Consider the MAX-CUT problem: given a graph $G = (V, E)$, find a subset of the vertices that maximizes the number of edges cut. Let A be the negative adjacency matrix of the graph, i.e.

$$A_{ij} = \begin{cases} -1, & (i,j) \in E \\ \\ 0, & o/w \end{cases}$$

Also suppose that $A_{ii} = 0$.

Next, consider the mathematical program:

$$\min \left\{ f_A(\mathbf{x}) = \frac{1}{4}(\mathbf{x}^\top A\mathbf{x} - 2|E|) \right\} \qquad (1.1)$$
$$\|\mathbf{x}\|_\infty = 1 \ .$$

Consider the cut defined by the solution of this program, namely

$$S_{\mathbf{x}} = \{i \in V | \mathbf{x}_i = 1\},$$

for $\mathbf{x} = \mathbf{x}^\star$. Let $C(S)$ denote the size of the cut specified by the subset of edges $S \subseteq E$. Observe that the expression $\frac{1}{2}\mathbf{x}^\top A\mathbf{x}$, is exactly equal to the number of edges that are cut by $S_{\mathbf{x}}$ minus the number of edges that are uncut. Thus, we have

$$\frac{1}{2}\mathbf{x}A\mathbf{x} = C(S_{\mathbf{x}}) - (E - C(S_{\mathbf{x}})) = 2C(S_{\mathbf{x}}) - E,$$

and hence $f(\mathbf{x}) = C(S_{\mathbf{x}})$. Therefore, maximizing $f(\mathbf{x})$ is equivalent to the MAX-CUT problem, and is thus NP-hard. We proceed to make the constraint set convex and continuous. Consider the mathematical program

$$\min \{f_A(\mathbf{x})\} \qquad (1.2)$$
$$\|\mathbf{x}\|_\infty \leq 1 \ .$$

This is very similar to the previous program, but we relaxed the equality to be an inequality, consequently the constraint set is now the hypercube. We now claim that the solution is w.l.o.g. a vertex. To see that, consider $\mathbf{y}(\mathbf{x}) \in \{\pm 1\}^d$ a rounding of $\mathbf{x}$ to the corners defined by:

$$\mathbf{y}_i = \mathbf{y}(\mathbf{x})_i = \begin{cases} 1, & w.p. \frac{1+\mathbf{x}_i}{2} \\ -1, & w.p. \frac{1-\mathbf{x}_i}{2} \end{cases}$$

Notice that

$$\mathbf{E}[\mathbf{y}] = \mathbf{x} \ , \ \forall i \neq j \ . \ \mathbf{E}[\mathbf{y}_i\mathbf{y}_j] = \mathbf{x}_i\mathbf{x}_j,$$

and therefore $\mathbf{E}[\mathbf{y}(\mathbf{x})^\top A\mathbf{y}(\mathbf{x})] = \mathbf{x}^\top A\mathbf{x}$. We conclude that the optimum of mathematical program 1.2 is the same as that for 1.1, and both are NP-hard. □

1.5.2 Computational hardness from the dynamics

Consider the following problem: given a dynamical system, is it stable, or is there a sequence of controls that will result in the system state becoming arbitrarily large?

We now show that this problem can be reduced to the MAX-2-SAT problem, which is known to be NP-hard. This implies that determining the stability of a dynamical system is NP-hard. The following proof is essentially from [BT00].

Consider the Multilinear Polynomial Family (MPF) of the following form

$$\{c_0(u_{1:m}) + c_1(u_{1:m})x + ... + c_n(u_{1:m})x^n, u_i \in [u_i^-, u_i^+]\}.$$

This is a family of polynomials of degree at most n in the variable x. The coefficients are polynomial functions of variables $u_1, ..., u_m$, each being a real number in a given range.

A polynomial is called stable if and only if all its roots have negative real parts. We consider the following problem: given an MPF, are all polynomials in it stable?

We now show that this problem is NP-hard by reduction to Max-2-SAT, a classical NP-hard problem. Recall that the Max-2-SAT problem is, given a Boolean formula over n variables and with m clauses, for example

$$F(x_{1:n}) = (x_i \vee x_j) \wedge ... \wedge (\neg x_i \vee x_j),$$

decide if there exists a Boolean assignment that satisfies at least k clauses.

Given a Max-2-SAT instance, we construct a MPF instance as follows

$$\begin{aligned} p(F) &= \sum_{ij} p(x_i, x_j) \quad \text{for} \\ p(x_i \vee x_j) &\to u_i + u_j - u_i u_j \\ p(x_i \vee \neg x_j) &\to 1 + u_j - u_i u_j \\ p(\neg x_i \vee \neg x_j) &\to 1 - u_i u_j. \end{aligned}$$

Thus, we rewrite every clause from the given formula in the form of a function of u. Now consider the MPF:

$$\{x + k - p(F), u_i \in [0, 1]\}. \tag{1.3}$$

For every polynomial in this family to be stable, the root of every polynomial needs to have a negative real part, which means at least k of the

clauses in p(F) have to be true. But checking if there exists an assignment of u_i so that at least k of the clauses are true requires solving Max 2 SAT. Hence in general, checking for the stability of all polynomials in a MPF is NP-hard.

Now, consider the dynamical equation given by

$$x_{t+1} = x_t \times e^{k - p(u_{1:m})}$$

with p being the polynomial family we just constructed. For this dynamics to not explode, we would need to verify if there is an assignment of $u_{1:m}$ such that $p(u_{1:m}) \geq k$, which we just proved is NP-hard. Hence we conclude that determining if there exists a control that makes this system stable is NP-hard.

1.6 Summary

In this lecture, we formalized the control problem and saw examples of how to use physics to model dynamical systems, as in the case of the medical ventilator. We then explored linear controllers and the PID controller, and saw some hardness results related to general cost functions and of verifying stability of dynamical systems.

1.7 Bibliographic Remarks

Control theory is one of the most established mathematical disciplines in engineering and has been studied for centuries. For an enlightening historical survey and context see [FCZI03]. We will address specific subfields and methods in future chapters, where appropriate references are given. A very thorough introduction to control is the book of [Ber07].

For an excellent survey on the computational complexity of control is studied in the survey of [BT00]. The computional complexity of mathematical programming, both constrained and unconstrained, is commented upon in e.g. [Haz19b], which also contains numerous other references.

For a historical account on the development of the PID controller see [Ben93]. The theoretical and practical properties of this controller are spelled out in detail in [ÅH95].

Chapter 2

Introduction to Reinforcement Learning

The previous lecture described and motivated the problem of optimal control. In conclusion, we have seen that in full generality, the control problem is computationally hard. This motivates the topic of this chapter: reinforcement learning with full (inefficient) state space representation.

Reinforcement learning (RL) is a sub-field of machine learning that formally models the setting of learning through interaction in a reactive environment. This is contrast to learning from examples, as in supervised learning, or inference according to a probabilistic model, as in Bayesian modeling. In RL, we have an agent and an environment. The agent observes its position (or state) in the environment and takes actions that transition it to a new state. The environment looks at an agent's state and hands out rewards based on a hidden set of criteria.

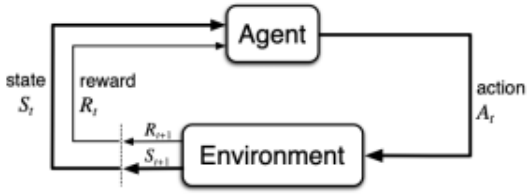


Figure 2.1: Source: Reinforcement Learning: An Introduction. Sutton & Barto, 2017.

Typically, the goal of RL is for the agent to learn behavior that maximizes the total reward it receives from the environment. This methodology has led

to some notable successes: machines have learned how to play Atari games, how to beat human masters of Go, and how to write long-form responses to an essay prompt.

Examples of problems tackled with RL. The reinforcement learning model has numerous success stories, many owing much of the recent advancement to the advent of deep learning as a function approximator, for example:

- Flying a helicopter / drone
- Play Go/backgammon/chess/Atari games
- Control a power station / thermometer / inverted pendulum
- Make a humanoid walk

The Markovian assumption: crucial to RL is the assumption that the state, observation and reward are independent of past given the current state.

2.1 An example of an RL instance

The following is a classical example from [SB18] that can be tackled with RL. In this example, a robot starts in the bottom leftmost square, and its goal is to reach the +1 reward.

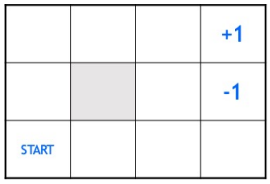


Figure 2.2: Source: Reinforcement Learning: An Introduction. Sutton & Barto, 2017.

In this example, the robot has controls that allow it to move left / right / forward / backward. The controls, however, are not reliable, and with 0.8 probably the robot moves to wherever it intends. With the remaining 0.2 probability the robot moves sideways, 0.1 probability to each side.

→	→	→	+1
↑	Blocked	↑	-1
↑	←	←	←

Figure 2.3: The optimal policy (mapping from state to action) with reward for each step taken being -0.04s

→	→	→	+1
↑	Blocked	→	-1
→	→	→	↑

Figure 2.4: Here, we change the reward of each step to be -2. The optimal policy changes: now the robot is more concerned with ending the game as quickly as possible rather than trying to get to the +1 reward.

2.2 Markov Processes

Before describing the full RL setting, we start with a reminder of Markov Chains, a.k.a. Markov Processes. A Markov Process is a model of a memoryless stochastic process as a weighted directed graph. It consists of a set of states, the nodes, and a transition matrix describing the probability of moving from one state to another (these are the weights on the edges). Figure 2.2 desribes a Markov Chain corresponding to a weather process.

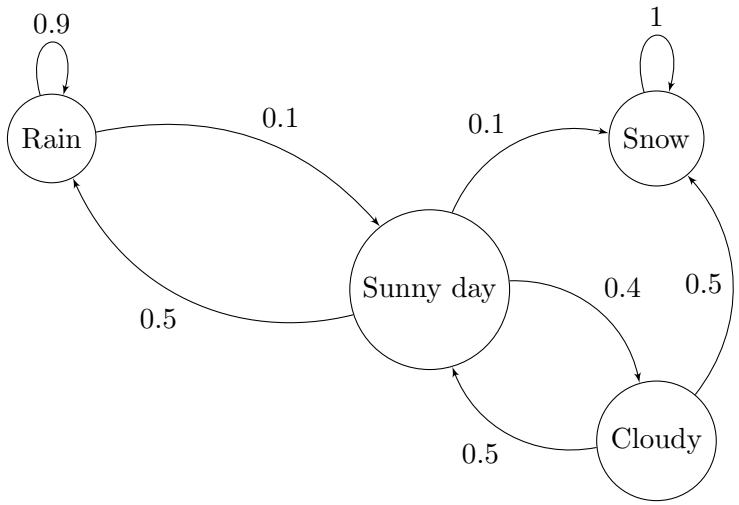


Figure 2.2 depicts a Markov Process with transition matrix

$$\mathbf{P} = \begin{pmatrix} 0.9 & 0.1 & 0 & 0 \\ 0.5 & 0 & 0.4 & 0.1 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Denote the distribution over states at time t by $\mathbf{p}_t$. Then it can be seen that

$$\mathbf{p}_{t+1} = \mathbf{p}_t \mathbf{P} = \mathbf{p}_0 \mathbf{P}^t,$$

for $\mathbf{p}_0$ being the starting state in vector representation.

Definition 2.1. *A stationary distribution of a Markov Process is a limiting probability distribution*

$$\boldsymbol{\pi} = \lim_{t \to \infty} \mathbf{p}_0 \mathbf{P}^t$$

A stationary distribution satisfies

$$\boldsymbol{\pi} = \boldsymbol{\pi} \mathbf{P}$$

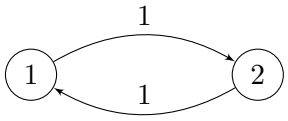


Figure 2.5: In this very simple Markov Chain, the state will alternate every time step, so there is no stationary distribution. This MC has periodicity of two.

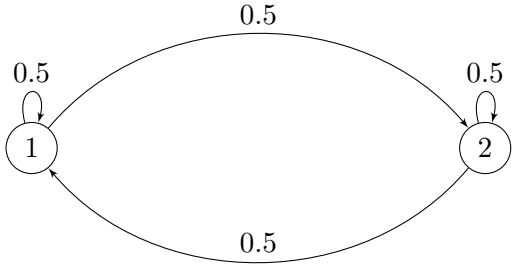


Figure 2.6: Adding a self-loop to every node makes the periodicity to be one, and this chain is also irriducible. The stationary distribution exists and is unique, it is equal to $\boldsymbol{\pi} = (\frac{1}{2}, \frac{1}{2})$.

The stationary does not always exist nor is it always unique. Existence, uniqueness and convergence to stationarity are the subject of Ergodic theory, and full discussion is beyond our scope.

Below we survey **sufficient** conditions for stationary distribution to exist. They are:

1. **Irreducibility.** A Markov Chain is irreducible if and only if $\forall i, j \; \exists t \; s.t. \; \mathbf{P}^t_{i,j} > 0$. In other words, every state is reachable from every state.

2. **A-periodicity.** A Markov Chain is a-periodic if and only if $d(s_i) = 1$ for all states $s_i \in S$, where

$$d(s_i) = gcd\left\{t \in \mathbb{N}_+ \;\middle|\; \mathbf{P}^t_{ii} > 0\right\},$$

   is the periodicity of state s.

The **Ergodic theorem** asserts that every irreducible and a-periodic Markov chain has a unique stationary distribution.

2.3 Markov Reward Processes

Markov Reward Processes (MRPs) add in the concept of reward to Markov Processes. At every state, the agent now receives a reward, which we assume w.l.o.g is in the range $\in [0, 1]$. Rewards are discounted over time, so rewards are worth more today than rewards in the future.

Definition 2.2. *Formally, a Markov Reward Process is a tuple* $(S, \mathbf{P}, \mathbf{R}, \gamma)$.

- S *is the set of states*
- $\mathbf{P} \in \mathbb{R}^{S \times S}$ *is the transition matrix.*
- $\mathbf{R} \in [0, 1]^{S \times S}$ *is the reward function, where* $\mathbf{R}(s, s')$ *is the reward for moving from state* s *to* s'.
- $\gamma \in [0, 1]$ *is the discount factor*

We henceforth also use the following notation:

1. R_{t+1} = random variable indicating the reward received at time $t + 1$, for taking action a_t from state S_t.
2. We denote by $S_t \in S$ the random variable which is the state at time t. v
3. $\hat{R} \in \mathbb{R}^S$ = vector that for each state, denote the expected reward in the next iteration as a function of the current state,
$$\hat{R}(s) = \mathbf{E}[R_{t+1} | S_t = s].$$
4. G_t random variable denoting the sum of the discounted rewards starting from time t onwards.
$$G_t = \sum_{i=1}^{\infty} R_{t+i} \gamma^{i-1}.$$

Exponentially diminishing rewards are standard in economics and biology. They are also mathematically convenient, since if $\gamma < 1$, G_t is always well defined regardless of stationarity.

In the case of infinite horizon RL over an Ergodic MRP and $\gamma = 1$, it is important to notice that

$$G := \lim_{t \to \infty} \frac{1}{t} \sum_{i=1}^{t} R_i = \sum_s \boldsymbol{\pi}(s) \hat{R}(s).$$

One of the most important definitions in RL is the value function:

Definition 2.3. *The value function maps states to their expected infinite-horizon reward,*

$$\mathbf{v}(s) = \mathbb{E}[G_t | S_t = s]$$

The value function tells how valuable a state is, inclusive of future, discounted rewards. One way to compute the value function is the Bellman equation for MRPs, which is the following recursive equation:

$$\mathbf{v}(s) = \hat{R}(s) + \gamma \sum_{s'} \mathbf{P}_{s,s'} \mathbf{v}(s')$$

Proof.

$$\begin{aligned}
\mathbf{v}(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[\sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i} | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+1+i} | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma v(s_{t+1}) | S_t = s] \\
&= \hat{R}(s) + \gamma \sum_{s'} \mathbf{P}_{s,s'} \mathbf{v}(s')
\end{aligned}$$

□

In matrix form, $\mathbf{v} = \hat{R} + \gamma \mathbf{P} v$, so to solve $\mathbf{v} = (\mathbf{I} - \gamma \mathbf{P})^{-1} \hat{R}$. This is a linear system and can be solved with Gaussian elimination in cubic time.

2.4 Markov Decision Processes

Definition 2.4. *A Markov Decision Process is a tuple* $(S, \mathbf{P}, R, A, \gamma)$ *such that:*

1. S *is a set of states*
2. A *is a set of possible actions*
3. $\mathbf{P}$ *is a transition matrix:*

$$\mathbf{P}^a_{ss'} = \Pr[S_{t+1} = s | S_t = s, A_t = a]$$

4. *R = reward function that gives each transition and and action a reward $R_{s,s',a}$ = reward for moving from s to s' using action a. We henceforth assume w.l.o.g. that $R \in [0,1]^{S \times A}$, where we define*

$$R_{sa} = \sum_{s' \in S} \mathbf{P}_{s,s',a} R_{s,s',a}.$$

5. *discount factor $\gamma > 0$.*

A policy is a mapping from state to distribution over actions, i.e.

$$\pi : S \mapsto A \ , \ \pi(a|s) = \Pr[A_t = a | S_t = s]$$

The goal of Reinforcement Learning is to determine a sequence of actions to maximize the long term reward. The Markovian structure of the problem suggests that this optimal value is realized by a stationary policy, i.e. there exists a fixed map from states to actions that maximize the expected rewards.

RL problems come in a variety of forms that have to do with the structure of the transitions, states, actions and most importantly which information is available to the learner. While it is natural that the set of actions is known to the decision maker, the states, transitions and rewards may be known and observable, or unknown and/or partially observable.

Thus, the goal is RL is to find a policy which maximizes the expected reward starting from a certain starting state, i.e.

$$\max_{\pi \in S \mapsto \Delta(A)} \mathbf{E} \left\{ \sum_{t=1}^{\infty} \gamma^{t-1} R_t | a_t \sim \pi(s_t) \ , \ S_1 = s_1 \right\},$$

where R_t is a random variable that denotes the reward at time t, that depends on the randomness in the policy as well as the transitions.

Important definitions:

1. The value function for a state s given a policy π is defined to be the expected reward from this state onwards where the actions are chosen according to π at each state, i.e.

$$\mathbf{v}_\pi(s) = \mathop{\mathbf{E}}_{\pi,P} \left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | S_1 = s \right]$$

2. Similarly, the Q function is defined over state-action pairs given a policy π, to be the expected reward from this state and choosing action

a onwards where the actions are chosen according to π at each state, i.e.

$$Q_\pi(s,a) = \mathop{\mathbf{E}}_{\pi,P}\left[\sum_{t=1}^{\infty}\gamma^{t-1}R_t|S_1=s,A_1=a\right]$$

A fundamental theorem of reinforcement learning, which we prove henceforth, is the following,

Theorem 2.5. *There exists a deterministic optimal policy π^*. All optimal policies achieve the same optimal value $\mathbf{v}^*(s)$, as well as $Q^*(s,a)$, and this holds for every state and action.*

2.4.1 The Bellman Equation

A fundamental property of MDP is that they satisfy the Bellman equation for dynamic programming.

$$\mathbf{v}_\pi(s) = \sum_{a\in A}\pi(a|s)\left[R_{sa} + \gamma\sum_{s'\in S}\mathbf{P}^a_{ss'}\mathbf{v}_\pi(s')\right]$$

In matrix form, we can write:

$$\mathbf{v}_\pi = R^\pi + \gamma\mathbf{P}^\pi\mathbf{v}_\pi,$$

where

1. R^π is the induced expected reward vector when playing policy π, i.e. $R^\pi(s) = \mathbf{E}_{a\sim\pi(s)}[R_{sa}]$.
2. $\mathbf{P}^\pi$ is the induced transition matrix when playing policy π, i.e.

$$\mathbf{P}^\pi_{ss'} = \sum_a \mathbf{P}^a_{ss'}\cdot\pi(a|s)$$

Bellman optimality equation: for every state and action, we have

$$\mathbf{v}^*(s) = \max_a\left\{Q^*(s,a)\right\}.$$

This implies the Bellman optimality equations:

$$\mathbf{v}^*(s) = \max_a\left\{R_{sa} + \gamma\sum_{s'}\mathbf{P}^a_{ss'}\mathbf{v}^*(s')\right\}$$

$$Q^*(s,a) = R_{sa} + \gamma\sum_{s'}\mathbf{P}^a_{ss'}\max_{a'}\left\{\mathbf{v}^*(s',a')\right\}$$

2.5 RL is a special case of linear programming

Consider the problem of computing the optimal value function and/or policy exactly. According to the Bellman optimality equation, this amounts to computing:

$$\mathbf{v}^*(s) = \max_a \left\{ R_{sa} + \gamma \sum_{s'} \mathbf{P}^a_{ss'} \mathbf{v}^*(s') \right\},$$

or in vector form,

$$\mathbf{v}^* = \max_{\mathbf{a} \in A^S} \left\{ R^{\mathbf{a}} + \gamma \mathbf{P}^{\mathbf{a}} \mathbf{v}^* \right\}.$$

Equivalently, we can write the following mathematical program:

$$\forall a \in A \;.\; s \in S \;.\; \mathbf{v}^*(s) \geq R_{sa} + \gamma \sum_{s'} \mathbf{P}^a_{ss'} \mathbf{v}^*(s').$$

This is a linear program! Then number of variables is S, and the number of constraints is $A+S$. It follows that this constrained mathematical program can be solved in time $\text{poly}(S, A)$. In particular, it follows that a solution exists, and is not necessarily unique, but the set of solutions is convex.

It is not immediate that the set of solutions is bounded, but that follows from the value iteration algorithm we study next.

2.6 The value iteration algorithm

Two of the most fundamental algorithms for solving MDP are policy and value iteration. These iterative methods start from an arbitrary solution and improve it till a certain measure of optimality.

The value iteration algorithm is given in figure 1, and its performance guarantee is particularly elegant. We thus present it with full analysis as per the theorem below, analysis idea from [Aro20],

Theorem 2.6. *Define the residual vector* $\mathbf{r}_t(s) = \mathbf{v}^*(s) - \mathbf{v}_t(s)$. *Then the optimal value vector* $\mathbf{v}^*$ *is unique and value iteration converges to the optimal value function and policy at a geometric rate of*

$$\|\mathbf{r}_{t+1}\| \leq \gamma^t \|\mathbf{r}_1\|_\infty$$

Proof. For every state we have by algorithm definition that,

$$\mathbf{v}_{t+1}(s) \geq \max_a \left\{ R_{sa} + \gamma \sum_{s'} \mathbf{P}^a_{ss'} \mathbf{v}_t(s') \right\} \geq R_{sa^*} + \gamma \sum_{s'} \mathbf{P}^{a^*}_{ss'} \mathbf{v}_t(s')$$

Algorithm 1 Value Iteration

Input: MDP
Set $\mathbf{v}_0(s) = 1$ to all states
for $t = 1$ to T **do**
Update for all states,

$$\mathbf{v}_{t+1}(s) = \max_a \left\{ R_{sa} + \gamma \sum_{s'} \mathbf{P}^a_{ss'} \mathbf{v}_k(s') \right\}$$

end for
return π_T which is derived from the last value function as

$$\pi_T(s) = \arg\max_a \left\{ R_{sa} + \gamma \sum_{s'} \mathbf{P}^a_{ss'} \mathbf{v}_T(s') \right\}$$

On the other hand, from Bellman optimality equation, we have that

$$\mathbf{v}^*(s) = R_{sa^*} + \gamma \sum_{s'} \mathbf{P}^{a^*}_{ss'} \mathbf{v}^*(s')$$

Subtracting the two equations we get

$$\begin{aligned} \mathbf{r}_{t+1}(s) &= \mathbf{v}^*(s) - \mathbf{v}_{t+1}(s) \\ &\leq \gamma \textstyle\sum_{s'} \mathbf{P}^{a^*}_{ss'} (\mathbf{v}^*(s') - \mathbf{v}_t(s')) \\ &= \gamma \textstyle\sum_{s'} \mathbf{P}^{a^*}_{ss'} \mathbf{r}_t(s') \end{aligned}$$

Or in vector form, $\mathbf{r}_{t+1} \leq \gamma \mathbf{P}^{\mathbf{a}^*} \mathbf{r}_t$. Recursive application, and applying the holder inequality, and using the fact that the rows of $\mathbf{P}$ are distributions, we get that

$$\|\mathbf{r}_{t+1}\| \leq \gamma^t \|(\mathbf{P}^{\mathbf{a}^*})^t \mathbf{r}_1\| \leq \gamma^t \|\mathbf{P}^{\mathbf{a}^*}\|_1^t \|\mathbf{r}_1\|_\infty \leq \gamma^t \|\mathbf{r}_1\|_\infty.$$

□

2.7 Bibliographic Remarks

The classical text of reinforcement learning that contains the examples presented in this lecture is [SB18]. Many other excellent texts exist, for a recent proof-based one see [AJK19]. Another great reference are the lecture notes of [Sil15].

The proof of convergence and rate of convergence for the value iteration algorithm with explicit rate are from discussion with [Aro20].

Chapter 3

Solution concepts of dynamical systems

In this chapter we return to the general problem of optimal control from definition 1.2, i.e. choosing a sequence of control $\{u_t\}$ such to minimize the long term cost,

$$\min_{u_{1:T}} \sum_t c_t(x_t, u_t)$$
$$s.t. \;\; x_{t+1} = f_t(x_t, u_t, w_t), \quad y_t = g(x_t).$$

Before moving to algorithms for optimal control, it is important to reason about what kind of solutions are even possible in general.

In the first lesson, we have argued that there are two sources of difficulty in optimal control: the dynamics and the costs. We first focus on a fully observable dynamical system of the form

$$x_{t+1} = f(x_t),$$

and ask ourselves what kind of solutions even exist?

3.1 Solution concepts for dynamical systems

During the first lecture, we showed that verifying if a system is stable is NP hard. In this section we define stability more precisely and introduce more evidence for the verifying it. To see that, let us first define the concept of equilibria and stability in a dynamical system formally.

Definition 3.1. *A point x is an equilibrium point of a dynamical system f if and only if $x = f(x)$. An equilibrium point is stable and attractive if and only if $x = \lim_{t\to\infty} f(x_t)$ for some neighborhood of x. If the neighborhood includes the entire set of states, we say that the equilibrium is globally attractive.*

An example of an equilibrium is a ball standing at rest at a bottom of a pit. If the ball were to be slightly moved, it would return to the bottom of the pit. Thus, this equilibrium is stable and attractive.

If the ball lies at rest on the top of a mountain, pushing the ball in any direction would cause it to fall. This is an equilibrium which is unstable.

Another example is given by a linear transformation in Euclidean space. Consider the system

$$x_{t+1} = Ax_t.$$

Clearly 0 is an equilibrium point as $A0 = 0$ for all A. If all eigenvalues are strictly between 1 and -1, then note that after repeated operations of A all vectors will tend to 0. Thus 0 is a **globally attractive, stable equilibrium**.

Intractability of determining stability In addition to the mathematical proof from lesson one, there is a very nice and convincing example is as follows. Consider the following dynamical system over the complex numbers

$$z_{t+1} = f_c(z_t) = z_t^2 + c.$$

The Mandelbrot set is now defined as

$$M_c = \{c \in C \ , \ \lim_t |f_c^t(0)| < \infty\}.$$

That is, the set of all complex numbers c for which the system has a stable attractive equilibrium (that is not ∞). It is known that determining if a given point belongs to the Mandelbrot set is computationally undecidable, i.e. there is no Turing machine that decides it in finite time! (see e.g. [BY08])

As the saying goes, a picture is worth a thousand words... Indeed, the intractablity of the Mandelbrot set is clearly visible in figure 3.1.

The complexity of dynamical equilibria. Another compelling visual evidence of the complexity of equilibrium in dynamical systems is Lorenz attractor. This is a strange equilibrium resulting of a simple dynamics that

can be described with three equations in three variables, called the Lorenz equations [Lor63],

$$\begin{aligned} x_{t+1} &= \sigma(y_t - x_t) \\ y_{t+1} &= x_t(\rho - z_t) - y_t \\ z_{t+1} &= x_t y_t - \beta z_t. \end{aligned}$$

The 3D trajectories of this system are attracted to a 2D manifold with two "wings" that resemble a butterfly as depicted in figure 3.2. For formal properties of this attractor set see [Str14, Tuc99]. This demonstrates that even studying a solution concept by itself is a formidable task.

For a formal proof of attractiveness in the continuous case, see section 9.2 of [Str14] or the original paper of Lorenz [Lor63].

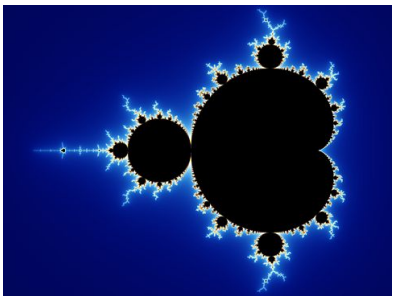


Figure 3.1: Mandelbrot set

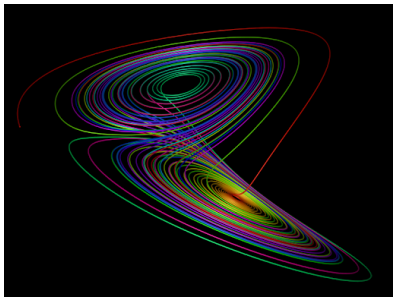


Figure 3.2: Lorenz attractor

3.2 Stabilizability and Controllability

A conclusion from the previous section is that the set of stable solutions of a given dynamical system is complex. Finding a solution, or even determining if a given point is stable, is an intractable problem.

We now returning to the even harder and more general problem of control of dynamical systems. Two fundamental properties of dynamical systems are stabilizability and controllability, as defined next.

Definition 3.2. *A Dynamical System is* ***stabilizable*** *if and only if there exists a sequence of controls that drives the system to the zero state. That is,* $\lim_{t\to\infty} f(x_t, u_t) = 0$.

Definition 3.3. *A Dynamical System is* ***controllable*** *if and only if, for every target state, there exists a sequence of controls that drives the system to that state.*

Note that controllability clearly implies stabilizability as the zero state is one possible target state.

The discussion in this chapter thus far has shown that these notions are very difficult to verify for a general dynamical systems. This motivates our investigation of linear dynmical systems!

3.3 Linear Dynamical Systems

A linear dynamical system (LDS) is a dynamical system that evolves linearly according to the equation

$$\mathbf{x}_{t+1} = A_t \mathbf{x}_t + B_t \mathbf{u}_t + \mathbf{w}_t.$$

If A_t, B_t change over time, this is called a time-varying LDS. If $A_t = A$ and $B_t = B$ are fixed, we say that the LDS is time-invariant.

3.3.1 General dynamics as time-variant LDS

At first, this class of systems seems overly restrictive. However, any dynamics equation f that is smooth, and has additive noise, can be approximated by a time-varying linear system to an arbitrary accuracy determined by the discretization time step.

To see this, in 1D the first-order Taylor expansion of a given dynamics f is

$$x_{t+1} = f(x_t, u_t) + w_t \approx \frac{\partial}{\partial x} f(x,u)\Big|_{x_t,u_t} (x - x_t) + \frac{\partial}{\partial u} f(x,u)\Big|_{x_t,u_t} (u - u_t) + w_t.$$

This simplifies to

$$x_{t+1} = A_t x_t + B_t u_t + w_t.$$

An analogous approximation holds for the higher-dimensional case. The Taylor expansion is

$$x_{t+1} = f(x_t, u_t) + w_t \approx J_x f(x_t, u_t)(x - x_t) + J_u f(x_t, u_t)(u - u_t) + w_t,$$

where $J_x f(x_t, u_t)$ is the Jacobian of f w.r.t. the variables x. Writing the matrices A_t, B_t for the Jacobians, we have

$$x_{t+1} = A_t x_t + B_t u_t + w_t.$$

This derivation explains why in practice, the class of time varying linear dynamical systems is very expressive. This is particularly true with hidden states that can capture complicated dynamics while imposing enough structure for efficient algorithms.

As we see next, stabilizability and controllability can be checked efficiently, in contrast to general dynamics.

3.3.2 Stabilizability of Linear Systems

For this section we consider time-invariant linear dynamical systems, for simplicity, that is, dynamics of the form

$$\mathbf{x}_{t+1} = A_t\mathbf{x}_t + B_t\mathbf{u}_t + \mathbf{w}_t.$$

Determining whether such a system is stabilizable is determined by a particularly simple condition as follows. Henceforth we denote by $\rho(A)$ the spectral radius of a matrix A, that is,

$$\rho(A) = \max\{|\lambda_1|, ..., |\lambda_d|\},$$

the maximum absolute value from all (possibly complex) eigenvalues. The stabilizability of a linear dynamical system can be characterized as follows.

Theorem 3.4. *A time-invariant LDS is stabilizable if and only if there exists a matrix K such that*

$$\rho(A + BK) < 1.$$

We henceforth sufficiency of this condition, and leave necessity as an exercise. Both directions rely on the characterization of the spectral norm as follows:

Fact 3.5. *$\rho(A) < 1$ if and only if $\lim_{t\mapsto\infty} A^t = 0$. Furthermore, if $\rho(A) > 1$, then for any norm, $\lim_{t\mapsto\infty} \|A^t\| = \infty$.*

Proof of this fact is left as an exercise.

Sufficiency. Suppose there exists a K such that $\rho(A + BK) < 1$. Then choose all controls to be $\mathbf{u}_t = K\mathbf{x}_t$. The evolution of the system without perturbation is governed by

$$\mathbf{x}_t = (A + BK)\mathbf{x}_{t-1} = ... = (A + BK)^t\mathbf{x}_0.$$

Thus, using the previous fact concerning the spectral radius, we have

$$\lim_{t\mapsto\infty} \|\mathbf{x}_t\| = \lim_{t\mapsto\infty} \|(A + BK)^t x_0\| = \|0x_0\| = 0$$

□

3.3.3 Controllability of LDS

Linear dynamical systems also exhibit particularly elegant characterization of controllability.

Definition 3.6. *The* Kalman controllability matrix *of a LDS is given by*

$$K_r(A, B) = [B, AB, A^2B, \ldots, A^{r-1}B] \in \mathbb{R}^{d_x \times r d_u}.$$

Theorem 3.7. *A time-invariant LDS is controllable if and only if*

$$\operatorname{rank} K_{d_x}(A, B) = d_x.$$

Proof. Consider the evolution of the LDS without pertrubation starting from $\mathbf{x}_0$, the final state is

$$\begin{aligned}\mathbf{x}_T &= A\mathbf{x}_{T-1} + B\mathbf{u}_{T-1} \\ &= A^2\mathbf{x}_{T-2} + AB\mathbf{u}_{T-2} + B\mathbf{u}_{T-1} \\ &= \ldots = \sum_{t=1}^{T} A^{t-1}B\mathbf{u}_{T-t} + A^T\mathbf{x}_0.\end{aligned}$$

Thus,

$$\mathbf{x}_T - A^T\mathbf{x}_0 = [B, AB, A^2B, \ldots, A^{T-1}B] \begin{bmatrix} \mathbf{u}_{T-1} \\ \vdots \\ \mathbf{u}_1 \end{bmatrix}.$$

By the fundamental theory of linear algebra, this equation has a solution for every $\mathbf{x}_0$, $\mathbf{x}_T$, i.e. a set of vectors $\{\mathbf{u}_t\}$ that satisfy the equation, if and only if $K_T(A, B)$ has full rank. This rank is bounded by the number of rows of K_T, which is d_x.

Henceforth let $r_p = \operatorname{rank}(K_p)$. According to the above, the proof is complete if $r_T = r_{d_x}$) for all $T \geq d_u$. We claim that once $r_p = r_{p+1}$, the rank never increases again. Assuming $r_p = r_{p+1}$, we have

$$\operatorname{span}[B, AB, \ldots, A^pB] = \operatorname{span}[B, AB, \ldots, A^{p-1}B].$$

Since linear transformations retain containment, this implies

$$\operatorname{span}[AB, \ldots, A^{p+1}B] \subseteq \operatorname{span}[AB, \ldots, A^pB].$$

Adding B to both matrices doesn't affect the subset relationship.

$$\operatorname{span}[B, AB, \ldots, A^{p+1}B] \subseteq \operatorname{span}[B, AB, \ldots, A^pB]$$

Therefore, $r_{p+2} \leq r_{p+1}$. Since the rank cannot decrease through the addition of columns, we have $r_{p+2} = r_{p+1}$. Since the rank can grow at most d_x times, the equation $r_{d_x} = r_T$ holds, and this concludes the theorem.

□

3.4 Quantitative definitions for stabilizability

In future chapter, we will require a quantitative definition for stabilizability, one that also holds for time-varying linear dynamical systems. This is required to prove finite-time regret bounds, which depend on the quantitative characterizations of stabilizability below.

Further, the notion of stabilizability we require is more refined than driving the system to zero. We require the ability to drive the system to zero under **adversarial** bounded bounded noise in the dynamics. In classical control theory this is called **BIBO-stability**, for Bounded-Input-Bounded-Output.

This definition can apply to any dynamical system and is an important concept in non-stochastic control.

Definition 3.8. *A policy π for a dynamical system f is γ-**stabilizing** if and only if the following holds. For every starting state $\|\mathbf{x}_0\| \leq 1$ and every sequence of adversarially chosen bounded perturbations $\mathbf{w}_{1:T}, \|\mathbf{w}_t\| \leq 1$, let $\mathbf{x}_t^\pi$ be the state arrived using policy π,*

$$\mathbf{x}_{t+1}^\pi = f(\mathbf{x}_t^\pi, \pi(\mathbf{x}_t)).$$

Then the system state is bounded by an absolute constant, independent of T, at every iteration,

$$\forall t \ , \ \|\mathbf{x}_t^\pi\|^2 \leq 1 + \gamma.$$

A dynamical system is said to be γ-stabilizable if and only if it admits a γ-stabilizing controller.

We proceed to show non-stochastic stabilizing controllers exist for time-invariant and time-varying systems.

3.4.1 Stabilizability of time-invariant linear systems

Consider a linear time-invariant system given by the transformations (A, B). We have already remarked that this system is called stabilizable, according to the classical, noise-free definition, if and only if the condition $\rho(A+BK) < 1$ holds.

A stronger quantitative property can be obtained as follows.

Lemma 3.9. *A linear controller K is γ-stabilizing controller for LTI system (A, B) if the matrix $A + BK$ admits a factorization with the following properties.*

$$A + BK = HLH^{-1} \ , \ \|H\|\|H\|^{-1} \leq \kappa \ , \ \|L\| \leq 1 - \delta \ , \ \frac{\kappa}{\delta} \leq \gamma.$$

Proof. Let $\tilde{K} = A + BK$. By unrolling the dynamics equation we have that

$$\begin{aligned}\mathbf{x}_{t+1} &= A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t = \tilde{K}\mathbf{x}_t + \mathbf{w}_t \\ &= \textstyle\sum_{i=0}^{t-1} \tilde{K}^i \mathbf{w}_{t-i} + \tilde{K}^t \mathbf{x}_0 = \sum_{i=0}^{t} (HLH^{-1})^i \mathbf{w}_{t-i} + (HLH^{-1})^t \mathbf{x}_0 \\ &= \textstyle\sum_{i=0}^{t} HL^i H^{-1} \mathbf{w}_{t-i} + HL^t H^{-1} \mathbf{x}_0\end{aligned}$$

Thus,

$$\begin{aligned}\|\mathbf{x}_{t+1}\| &\leq \textstyle\sum_{i=0}^{t} \|H\|\|L^i\|\|H^{-1}\|\|\mathbf{w}_{t-i}\| + \|H\|\|L^t\|\|H^{-1}\|\|\mathbf{x}_0\| \\ &\leq \kappa \textstyle\sum_{i=0}^{\infty} (1-\delta)^i + (1-\delta)^t \|\mathbf{x}_0\| \leq \frac{\kappa}{\delta} + e^{-\delta t} \leq \gamma + 1\end{aligned}$$

□

A linear time-invariant system is thus γ-stabilizable if we can find a matrix K with the properties above.

3.4.2 Stabilizability of time-variant linear systems

The γ-stabilizability condition, as opposed to classical stabilizability, is particularly important to generalize to time-varying systems, since the spectral radius of the product is **not** bounded by the product of spectral radii.

We now define a sufficient condition for γ stabilizability. Consider a time-varying linear dynamical system given by the transformations $\{A_t, B_t\}$.

Lemma 3.10. *A sequence of linear controllers $K_{1:T}$ is γ-stabilizing controller for the time-varying linear dynamical system $\{A_{1:T}, B_{1:T}\}$ if the matrices $\tilde{K}_t = A_t + B_t K_t$ satisfy*

$$\begin{gathered}\forall t \ . \ \tilde{K}_t = H_t L_t H_t^{-1} \ , \ \|H_t H_{t+1}^{-1}\| \leq 1 + \delta \\ \forall t_1, t_2 \ , \ \|H_{t_1}\|\|H_{t_2}^{-1}\| \leq \kappa \\ \|L_t\| \leq 1 - 3\delta \ , \ \tfrac{\kappa}{\delta} \leq \gamma.\end{gathered}$$

Proof. First, notice that under the condition of the lemma, we have that for any sequence of matrices $\tilde{K}_t$ that

$$\begin{aligned}\left\|\prod_{t=r}^{s} \tilde{K}_t\right\| &= \left\|\textstyle\prod_t (H_t L_t H_t^{-1})\right\| \\ &\leq \|H_r\|\|L_r\|\|H_r^{-1} H_{r+1}\| ... \|L_s\|\|H_s^{-1}\| \\ &\leq \kappa(1-3\delta)^{s-r}(1+\delta)^{r-s} \leq \kappa(1-\delta)^{r-s} \leq \kappa e^{-\delta(r-s)}\end{aligned}$$

Similar to the time-invariant case, by unrolling the dynamics equation we have that

$$\begin{aligned}\mathbf{x}_{t+1} &= A_t\mathbf{x}_t + B_t\mathbf{u}_t + \mathbf{w}_t = \tilde{K}_t\mathbf{x}_t + \mathbf{w}_t \\ &= \textstyle\sum_{i=0}^{t-1}(\prod_{j=1}^{i}\tilde{K}_{t-j})\mathbf{w}_{t-i} + (\prod_{j=1}^{t}\tilde{K}_{t-j})\mathbf{x}_0 \\ &\leq \kappa\textstyle\sum_{i=0}^{\infty}e^{-\delta i} + e^{-\delta t}\|\mathbf{x}_0\| \leq \frac{\kappa}{\delta} + e^{-\delta t} \leq \gamma + 1\end{aligned}$$

□

3.5 Bibliographic Remarks

The Lorenz equations were introduced to model the phenomenon of atmospheric convection [Lor63]. For detailed information about the mathematical properties of the Lorenz attractor see [Str14, Tuc99].

For a rigorous study of Julia sets and the Mandelbrot set see [BY08].

The notion of (κ, δ)-strong-stability, which is used in the derivation for γ-stabilizability, is due to [CHK$^+$18]. An extension of this strong stability notion to time-varying linear dynamics was established in [CKM19].

Chapter 4

Optimal Control of Linear Dynamical Systems

The previous chapter studied more basic notions than control, namely solution concepts of dynamical systems, statibility and controllability. Even for these more basic problems, we have seen that they are computationally intractable in general.

This motivated us to look into simpler dynamics, namely linear dynamical systems, for which we can at least answer basic questions on solutions, stability and controllability... In this chapter we ask whether we can control LDS optimally?

The answer is positive in a very strong sense. A hallmark of optimal control theory is the control of LDS with quadratic cost functions, i.e. the LQR problem we define and study next. It admits an efficient and very practical algorithm.

4.1 The Linear-Quadratic Regulator

The linear quadratic regulator (LQR) problem is that of optimal control of a linear dynamical system with known and fixed quadratic costs. The LQR problem as well as its solution were proposed in the seminal work of Kalman [Kal60]. Formally,

Definition 4.1. *The Linear Quadratic Regulator (LQR) input is a linear dynamical system that evolves according to*

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t,$$

where $\mathbf{w}_t \sim N(0, \sigma^2)$ *is i.i.d. Normal, and convex quadratic cost functions given by matrices* $\mathbf{Q}, \mathbf{R}$.

The goal is to generate a sequence of controls $\mathbf{u}_t$ *such to minimize the expected cost, taken over the randomness of the perturbations, given by quadratic loss functions,*

$$\min_{\mathbf{u}(\mathbf{x})} \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{x}_t^\top \mathbf{Q}\mathbf{x}_t + \mathbf{u}_t^\top \mathbf{R}\mathbf{u}_t\right]$$

The notation we have used is consistent with the rest of our discussion thus far:

1. $\mathbf{x}_t \in \mathbb{R}^{d_x}$ is the state
2. $\mathbf{u}_t \in \mathbb{R}^{d_u}$ is the control input
3. $\mathbf{w}_t \in \mathbb{R}^{d_w}$ is the perturbation sequence, which are from normal distributions, but they can be from any other distribution with zero mean as long as they are independent of states or controls.
4. $\mathbf{A} \in \mathbb{R}^{d_x \times d_x}$, $\mathbf{B} \in \mathbb{R}^{d_x \times d_u}$ are the system matrices;
5. $\mathbf{Q} \in \mathbb{R}^{d_x \times d_x}$, $\mathbf{R} \in \mathbb{R}^{d_u \times d_u}$ are positive semi-definite matrices such that the cost functions are convex.

Such model is very useful since many physical problems, such as the centrifugal governor, the ventilator control problem, and many more, can be approximated in every time step by linear functions.

The quadratic convex cost functions are also very natural in physical applications.

The LQR is a special case of an MDP from the introductory chapter on reinforcement learning, as the transitions are stochastic. However, naive application of value iteration, policy iteration, or linear programming do not apply, since there are infinitely many states and actions.

Nevertheless, Kalman's solution to this problem exhibits a particularly elegant and succinct structure as we see next.

4.2 Optimal solution of the LQR

We know from Chapter 2 that the LQR is a special case of a finite-time Markov Decision Process, and hence admits an optimal policy. However,

the optimal policy is in general a mapping from states to actions, both are infinite Euclidean spaces in our case.

Nevertheless, it can be shown that the optimal policy for an LQR is a **linear policy**, i.e. a linear mapping from $\mathbb{R}^{d_x}$ to $\mathbb{R}^{d_u}$! This astonishing fact together with a remarkable property of the value function is formally given in the following theorem.

Theorem 4.2. *There exist a positive semi-definite matrix $\mathbf{S}_t$, and a matrix $\mathbf{K}_t$ and scalar c_t, such that the optimal value function and optimal policy satisfy:*

$$\mathbf{v}_t^*(\mathbf{x}) = \mathbf{x}_t^\top \mathbf{S}_t \mathbf{x}_t + c_t, \quad \mathbf{u}_t^*(\mathbf{x}) = \mathbf{K}_t^\top \mathbf{x}$$

Proof. Recall the Bellman Optimality equation for a general MDP is written as

$$\mathbf{v}^*(s) = \max_a \left\{ R_{sa} + \gamma \sum_{s'} \mathbf{P}_{ss'}^a \mathbf{v}^*(s') \right\}.$$

We henceforth write the special case applicable to LQR, and prove the theorem by backward induction on t.
Base case: For $t = T$, we have

$$\mathbf{v}_T^*(x) = \min_{\mathbf{u}} \left\{ \mathbf{x}_T^\intercal Q \mathbf{x}_T + \mathbf{u}^\intercal R \mathbf{u} \right\}$$

Since $R \succcurlyeq 0$, a minimum occurs at $\mathbf{u} = 0$. Hence, $S_T = Q \succcurlyeq 0$, $K_T = 0$ and $c_T = 0$.
Inductive Step: Assume for some $t > 1$ that $\mathbf{u}_t^*(\mathbf{x}) = K_t \mathbf{x}$ and $\mathbf{v}_t^*(\mathbf{x}) = \mathbf{x}^\intercal S_t \mathbf{x} + c_t$. Then we have

$$\begin{aligned}
\mathbf{v}_{t-1}^*(\mathbf{x}) &= \min_{\mathbf{u}} \left\{ \mathbf{x}^\intercal Q \mathbf{x} + \mathbf{u}^\intercal R \mathbf{u} + \mathbf{v}_t^*(A\mathbf{x} + B\mathbf{u} + \mathbf{w}) \right\} \\
&= \min_{\mathbf{u}} \left\{ \mathbf{x}^\intercal Q \mathbf{x} + \mathbf{u}^\intercal R \mathbf{u} + \mathop{\mathbf{E}}_{\mathbf{w}} \left[(A\mathbf{x} + B\mathbf{u} + \mathbf{w})^\intercal S_t (A\mathbf{x} + B\mathbf{u} + \mathbf{w}) \right] \right\} \\
&= \min_{\mathbf{u}} \left\{ \mathbf{x}^\intercal Q \mathbf{x} + \mathbf{u}^\intercal R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^\intercal S_t (A\mathbf{x} + B\mathbf{u}) + \mathop{\mathbf{E}}_{\mathbf{w}} [\|\mathbf{w}\|^2] \right\},
\end{aligned}$$

where in the last equality we used the fact that $\mathbf{E}[\mathbf{w}] = 0$, and the perturbation is independent of the state and control. We henceforth denote the second moment of the noise by $\mathbf{E}[\|\mathbf{w}\|^2] = \sigma^2$.

Since R, S_t are positive semi-definite, the expression above is convex as a function of $\mathbf{u}$. At the global minimum the gradient vanishes, and hence

$$\nabla_{\mathbf{u}} \Big[\mathbf{x}^\intercal Q \mathbf{x} + \mathbf{u}^\intercal R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^\intercal S_t (A\mathbf{x} + B\mathbf{u}) + \sigma^2 \Big] = 0.$$

Since Q, R, S_t are all positive semi-definite and hence symmetric, this gives us

$$2R\mathbf{u} + 2B^\intercal S_t A\mathbf{x} + 2B^\intercal S_t B\mathbf{u} = 0 \implies (R + B^\intercal S_t B)\mathbf{u} = -B^\intercal S_t A\mathbf{x}.$$

Hence, we have that

$$\mathbf{u}_{t-1}(\mathbf{x}) = -K_{t-1}\mathbf{x} \quad K_{t-1} = (R + B^\intercal S_t B)^{-1}(B^\intercal S_t A).$$

If the matrices above are not full rank, we use the notation M^{-1} for the Moore-Penrose pseudo inverse. Plugging this in the expression for the value function, we get

$$\begin{aligned}
&\mathbf{v}^*_{t-1}(\mathbf{x}) \\
&= \mathbf{x}^\intercal Q\mathbf{x} + \mathbf{x}^\intercal K_{t-1}^\intercal R K_{t-1}\mathbf{x} + \mathbf{x}^\intercal (A - BK_{t-1})^\intercal S_t (A - BK_{t-1})\mathbf{x} + \sigma^2 \\
&= \mathbf{x}^\intercal S_{t-1}\mathbf{x} + c_t
\end{aligned}$$

where $S_{t-1} = Q + K_{t-1}^\intercal R K_{t-1} + (A - BK_{t-1})^\intercal S_t (A - BK_{t-1})$ is positive semi-definite $S_{t-1} \succcurlyeq 0$, and $c_t = \sigma^2$.

□

4.3 Infinite Horizon LQR and the Algebraic Riccatti Equation

Although infinite horizon problems can never appear in real world, in practice the horizon length may be unknown. Although there are techniques to iteratively update the horizon length with geometric increasing guesstimates, these are slow and impractical.

An alternative is to solve the LQR for the infinite horizon case, which results in a very elegant solution. Not only that, this turns out to be a practical solution to which the finite-horizon solution converges quickly.

The infinite horizon LQR problem can be written as follows.

$$\begin{aligned}
\min_{\mathbf{u}(\mathbf{x})} \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{x}_t^\top \mathbf{Q}\mathbf{x}_t + \mathbf{u}_t^\top \mathbf{R}\mathbf{u}_t\right] \\
\text{s.t. } \mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t,
\end{aligned}$$

where similar to the finite horizon case we assume that the perturbations $\mathbf{w}_t$ are i.i.d. zero mean and have bounded variance.

We henceforth assume that the perturbations are all zero for simplicity, adding zero-mean bounded variance perturbations is left as an exercise.

Recall the Bellman optimality equation

$$\mathbf{v}^*(\mathbf{x}) = \min_{\mathbf{u}} \left\{ \mathbf{x}^\top Q\mathbf{x} + \mathbf{u}^\top R\mathbf{u} + \mathbf{v}^*(A\mathbf{x} + B\mathbf{u}) \right\}.$$

Assuming that the value function is quadratic, we have

$$\mathbf{x}^\top S\mathbf{x} = \min_{\mathbf{u}} \left\{ \mathbf{x}^\top Q\mathbf{x} + \mathbf{u}^\top R\mathbf{u} + (A\mathbf{x} + B\mathbf{u})^\top S(A\mathbf{x} + B\mathbf{u}) \right\}$$

Similar to the finite time case, we have that

Lemma 4.3. *The optimal policy is given by a transformation* $\mathbf{u}^*(\mathbf{x}) = K\mathbf{x}$.

Proof. At optimality the gradient with respect to $\mathbf{u}$ of the equation above is zero. This implies that

$$2R\mathbf{u} + 2B^\top SB\mathbf{u} + 2B^\top SA\mathbf{x} = 0,$$

which implies

$$K = -(R + S^\top BS)^{-1}(B^\top SA)$$

□

Plugging back into the Bellman equation, we obtain

$$\mathbf{x}^\top S\mathbf{x} = \mathbf{x}^\top Q\mathbf{x} + \mathbf{x}^\top K^\top RK\mathbf{x} + \mathbf{x}^\top (A + BK)^\top S(A + BK)\mathbf{x}$$

removing $\mathbf{x}$, since it's true for all $\mathbf{x}$, we get

$$S = Q + K^\top RK + (A + BK)^\top S(A + BK)$$

Simplifying this expression we obtain

$$\mathbf{S} = \mathbf{Q} + \mathbf{A}^T\mathbf{SA} - \mathbf{A}^T\mathbf{SB}\left(\mathbf{R} + \mathbf{B}^T\mathbf{SB}\right)^{-1}\mathbf{B}^T\mathbf{SA}$$

This is called the **Discrete Time Algebraic Riccati Equation** (DARE), one of the most important equations in control theory.

In 1-D case, the LHS is a convex quadratic function. In higher dimension, it is somewhat more difficult to see, but the equation is still one that involves a convex quadratic function. This implies that iterative convex optimization methods rapidly converge to the globally optimal solution.

4.4 H_∞ Control

The LQR for controlling linear dynamical systems is a hallmark of optimal control theory and one of the most widely used techniques. However, the assumption that the disturbances are stochastic and i.i.d. is a strong one, and numerous attempts have been made over the years to generalize the noise model.

One of the most important and prominent generalizations is that of H_∞-control. The goal of H-infinity is to come up with a more robust solution, by considering the optimal solution vs. any possible perturbation sequence from a certain family. Formally,

$$\begin{gathered}\min_{\mathbf{u}_{1:T}(\mathbf{x})} \max_{\mathbf{w}_{1:T}} \sum_{t=1}^{T} \mathbf{x}_t^\top \mathbf{Q}\mathbf{x}_t + \mathbf{u}_t^\top \mathbf{R}\mathbf{u}_t \\ \text{s.t. } \mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t,\end{gathered}$$

where we use the exact same notation as in previous sections.

Notice that this problem does not fall into the category of a Markov Decision Process, since the objective is not additive. Therefore Bellman Equation does not apply.

Instead, the appropriate techniques to address this problem come from the theory of dynamical games. This is, in face, a two-player zero sum game. The first player is trying to minimize the costs and the other to maximize the costs, both of them subject to the dynamics evolution constraint.

Using techniques from dynamic games, the following can be proven, see e.g [ZDG96, BB08].

Theorem 4.4. *For the H_∞ control problem with quadratic convex loss functions, the optimal solution is a linear policy of the state given by*

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

The proof is somewhat more involved than the LQR derivation and omitted from this course.

4.5 Bibliographic Remarks

Optimal control is considered by many to be the hallmark of control theory, and numerous excellent texts exists that cover the material of this lecture in detail. See e.g. [Ber07, ZDG96, Ted20].

The LQR problem and its solution were introduced in the seminal work of Kalman [Kal60].

For in-depth treatment of H_∞ control see the text [BB08].

Chapter 5

Learning in unknown LDS

In the previous chapter we have considered the most basic setting of control in a *known* linear dynamical system. In this chapter we are going to consider the setting of an *unknown* LDS. However, instead of starting with the more difficult problem of control, we approach unknown dynamics by first considering the task of learning!

5.1 Learning in dynamical systems

There are numerous different interpretations of "learning" in dynamical systems. Indeed, in this course alone we consider several different notions! For this chapter the notion we are concerned with is that of prediction. More precisely, consider a sequence of observable inputs to a dynamical system as in figure 5.1. Our goal is to correctly predict the output sequence according to a certain loss metric.

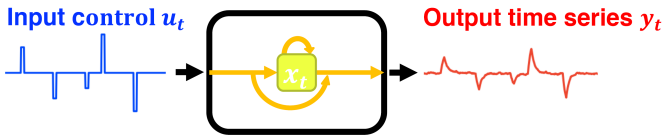


Figure 5.1: In learning we passively observe the input sequence and attempt to predict output.

For the rest of this chapter, we consider a partially observable linear dynamical system that is governed by the following equations.

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \eta_t$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \xi_t.$$

Here, as we have denoted throughout the text, $\mathbf{x}$ is the hidden state, $\mathbf{u}$ is the observed control input, η, ξ are perturbations/noises, and $\mathbf{y}$ is an observed linear projection of the state. We assume that the system matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are unknown to the learner. Furthermore, for the rest of this chapter we consider **open loop prediction**, meaning that the noises are assumed to be all zeros. We now spell out the precise setting of prediction and performance metrics.

5.1.1 Online learning of dynamical systems

In the problem of online learning in LDS, sequentially for $t = 1, ..., T$, the online learning algorithm:

- observes input $\mathbf{u}_t \in \mathbb{R}^{d_u}$
- predicts output $\hat{\mathbf{y}}_t \in \mathbb{R}^{d_y}$
- observes true system output y_t
- suffers loss according to a pre-specified loss function, $\ell(\hat{\mathbf{y}}_t, \mathbf{y}_t)$, e.g.

$$\ell(\hat{\mathbf{y}}_t, \mathbf{y}_t) = \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|^2.$$

Following the online learning literature (see e.g. [H+16]), the goal of the online learner is to minimize total loss over all game iterations. More precisely, we consider a family of prediction rules from available information to prediction, and consider the metric of *regret* vs. the best of these rules in hindsight. An example of a rule could be a fixed linear map, i.e. $\pi(\mathbf{u}_{1:t}, \mathbf{y}_{1:t-1}) = K\mathbf{u}_t$, where K can potentially depend on the system parameters (which are A, B, C for a linear dynamical system).

Formally, let Π be a family of mappings from $\{\mathbf{u}_\tau, \mathbf{y}_\tau\} \mapsto \hat{\mathbf{y}}_t$, i.e.

$$\pi \in \Pi \ , \ \pi(\mathbf{u}_{1:t}, \mathbf{y}_{1:t-1}) = \hat{\mathbf{y}}_t^\pi \in \mathbb{R}^{d_y}.$$

We give examples of such prediction rules henceforth. Let $\mathcal{A}$ be a prediction algorithm that iteratively produces estimates $\hat{\mathbf{y}}_t$ for the setting above. Then its regret w.r.t. Π is defined as

$$\text{regret}_T(\mathcal{A}, \Pi) = \sum_{t=1}^{T} \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|^2 - \min_{\pi \in \Pi} \sum_{t=1}^{T} \|\mathbf{y}_t - \hat{\mathbf{y}}_t^\pi\|^2.$$

The exact meaning of the regret metric depends on Π - the comparator class of predictors. We survey a few reasonable options next.

5.1.2 Classes of prediction rules

We now consider what is a reasonable prediction rule for the next output of a linear dynamical system? If the system transformations are known, and $\mathbf{C} = I$ is the identity matrix, then the optimal prediction can be computed via the system evolution equation $\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$. However, if $\mathbf{C} \neq I$, then we do not know the state from the observation because then the projection is not unique.

We thus focus on this more interesting case. Recall that we focus on open loop prediction, when there is no noise in the system. In such case, we can write

$$\begin{aligned}\mathbf{y}_t^\pi = \pi_{\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{x}_0\}}(t) &= \mathbf{C}\mathbf{x}_t \\ &= \mathbf{C}\mathbf{A}\mathbf{x}_{t-1} + \mathbf{C}\mathbf{B}\mathbf{u}_{t-1} \\ &\vdots \\ &= \sum_{i=1:t-1} \mathbf{C}\mathbf{A}^{i-1}B\mathbf{u}_{t-i} + \mathbf{C}\mathbf{A}^{t-1}\mathbf{x}_0.\end{aligned}$$

Thus, the operator $\pi_{\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{x}_0\}}$ takes a history of states and controls and predicts the next system output. If the spectral radius of $\mathbf{A}$ is larger than one, then the system is unstable and the magnitude of $\mathbf{y}_t$ can grow arbitrarily large, as discussed in section 3.3.2. This is problematic for many reasons, and we return to the issue of how to deal with unstable systems later chapters.

For the rest of this section, assume that $\rho(A) < 1 - \delta < 1$ for some $\delta > 0$. Which prediction rule is reasonable in this situation?

Denote by $\Pi_\star = \Pi_{\star,\delta,\beta,\gamma,U}$ the class of all predictors $\pi_{\{A,B,C,x_0\}}$ for all possible matrices A, B, C such that

$$\rho(A) \leq 1 - \delta, \|B\| \leq \beta, \|C\| \leq \gamma, \|\mathbf{u}_t\| \leq U.$$

This is obviously a reasonable class to consider, as it contains the true prediction assuming that the signal comes from an LDS and has no noise, hence its notation. However, the prediction rule depends on the entire history, whereas the importance of history diminishes exponentially fast due to the spectral norm bound on A. It this makes sense to consider a more restricted

class of predictors that have bounded memory, i.e.

$$\pi_{\{A,B,C,H\}} = \sum_{i=1:H} CA^{i-1}B\mathbf{u}_{t-i},$$

where we define all $\mathbf{u}$ vectors in negative times as zero. Denote by Π_H the class of all such predictors.

Clearly for a large H this class of predictors well approximates the true observations $\mathbf{y}_t$. We formalize what this means next.

Definition 5.1. *We say that a class of predictors Π_1 ε-approximates class Π_2 if the following holds. For every sequence of T inputs and outputs $\{\mathbf{u}_{1:T}, \mathbf{y}_{1:T}\}$, and every $\pi \in \Pi_1$, there exists $\pi_2 \in \Pi_2$ such that,*

$$\sum_{t=1}^{T} \|\mathbf{y}_t^{\pi_1} - \mathbf{y}_t^{\pi_2}\| \leq T\varepsilon,$$

where $\mathbf{y}_t^{\pi_1} = \pi_1(\mathbf{y}_{1:t}, \mathbf{u}_{1:t})$, $\mathbf{y}_t^{\pi_2} = \pi_2(\mathbf{y}_{1:t}, \mathbf{u}_{1:t})$

The significance of this definition is that if a simple predictor class approximates another more complicated class, then it suffices to consider only the first.

Let us demonstrate the usefulness of this definition for the predictor classes defined above. We start by noticing the implication of ε-approximation on the regret.

Lemma 5.2. *Let predictor class Π_1 ε-approximate class Π_2. Then for any prediction algorithm $\mathcal{A}$ we have*

$$\text{regret}_T(\mathcal{A}, \Pi_2) \leq \text{regret}_T(\mathcal{A}, \Pi_1) + \varepsilon T.$$

Proof of this lemma is left as an exercise to the reader. We now observe that the bounded-memory class defined previously approximates the optimal predictor class.

Lemma 5.3. *Suppose that $\|B\| \leq \beta, \|C\| \leq \gamma$ in Frobenius norm, and $\|\mathbf{u}_t\| \leq U$ for all $t \in [T]$. Then The class $\prod_H$ for $H = \frac{1}{\delta} \log \frac{T\delta\gamma U}{\delta\varepsilon}$, ε-approximates the class $\prod_\star$.*

Proof. Consider any sequence and any predictor from $\prod_\star$, say $\pi_1 = \pi_{A,B,C,\mathbf{x}_0}$. Consider the predictor $\pi_2 = \pi_{A,B,C,H} \in \prod_H$ for the same system matrices.

Then we have

$$\begin{aligned}
&\|\mathbf{y}_t^{\pi_1} - \mathbf{y}_t^{\pi_2}\| \\
&= \|\textstyle\sum_{i=1:t-1} CA^{i-1}B\mathbf{u}_{t-i} + CA^{t-1}\mathbf{x}_0 - \sum_{i=1:H} CA^{i-1}B\mathbf{u}_{t-i}\| \\
&= \|\textstyle\sum_{i=H:T} CA^{i}B\mathbf{u}_{t-i-1}\| \\
&\leq \textstyle\sum_{i=H:T}(1-\delta)^i\|C\|\|B\|\|\mathbf{u}_{t-i-1}\| \\
&\leq \tfrac{\beta\gamma U}{\delta}(1-\delta)^H \leq \tfrac{\beta\gamma U}{\delta}e^{-\delta H}
\end{aligned}$$

Using the value of H, and summing up over all T iterations, we have

$$\begin{aligned}
\sum_t \|\mathbf{y}_t^{\pi_1} - \mathbf{y}_t^{\pi_2}\| &\leq \textstyle\sum_t \frac{\beta\gamma U}{\delta}e^{-\delta H} \\
&\leq \tfrac{T\beta\gamma U}{\delta}e^{-\delta\cdot\frac{1}{\delta}\log\frac{2T\beta\gamma U}{\delta\varepsilon}} \\
&= \varepsilon
\end{aligned}$$

□

5.1.3 Learning by convex relaxation

At this point we have defined a bounded-memory class of predictors that well-approximates the optimal prediction. However, the reader may notice that the parametrization of Π_H is non-convex in the matrices A, B, C. This presents a computational challenge.

Luckily, this challenge is easily circumvented using a technique of **convex relaxation**. Instead of Π_H, we consider yet another simpler class, $\Pi_{M,H}$, given by

$$\Pi_{M,H} = \left\{ \pi_M \;\middle|\; \pi_M(\mathbf{u}_{1:t}, \mathbf{x}_{1:t}) = \sum_{i=1}^{H} M_i \mathbf{u}_{t-i} \right\}.$$

It can be seen that the class $\Pi_{M,H}$ ε-approximates the class Π_H, with $\varepsilon = 0$, which implies that it ε-approximates the class $\Pi_\star$ for an appropriate choice of H.

Most importantly, the natural parametrization of $\Pi_{M,H}$ is convex, and thus it can be learned by online convex optimization (OCO).

In OCO a decision maker iteratively produces a decision in a convex decision set, and the incurs a loss according to an adversarially chosen loss function. The goal of the decision maker is to minimize regret, and the vast scientific literature has produced a plethora of algorithms that are efficient and have provable regret bounds. See [H+16] for an extensive introduction.

The following algorithm learns the class $\Pi_{M,H}$ in the sense that it attains a sublinear regret bound for this class. This, in turn, implies a sublinear regret bound vs. the class $\Pi_\star$ due to the chain of approximations.

Algorithm 2 Learning LDS by online gradient descent

1: Input: $M^1_{1:H}$, convex constraints set $\mathcal{K} \subseteq \mathbb{R}^{H \times d_y \times d_u}$
2: **for** $t = 1$ to T **do**
3: Predict $\hat{\mathbf{y}}_t = \sum_{i=1}^{H} M_i^t \mathbf{u}_{t-i}$ and observe true $\mathbf{y}_t$.
4: Define cost function $f_t(M) = \|\mathbf{y}_t - \sum_{i=1}^{H} M_i \mathbf{u}_{t-i}\|^2$.
5: Update and project:

$$M^{t+1} = M^t - \eta_t \nabla f_t(M^t)$$
$$M^{t+1} = \prod_{\mathcal{K}} (M^{t+1})$$

6: **end for**

This algorithm is an instance of the online gradient descent algorithm, and Theorem 3.1 from [H+16] directly gives the following bound for it. We denote by $\mathcal{K}$ the set of all H-tuple of matrices $M_{1:H}$, such that they are bounded by the appropriate bounds on A, B, C, i.e.

$$\|M_i\| \leq (1-\delta)^{i-1} \beta \gamma,$$

when we assume that $\rho(A) \leq 1 - \delta, \|B\| \leq \beta, \|C\| \leq \gamma$.

Theorem 5.4. *For choice of step sizes $\eta_t = O(\frac{1}{\sqrt{t}})$, we have that*

$$\text{regret}(\mathcal{A}_2, \Pi_{M,H}) = \sum_t \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|^2 - \min_{\pi \in \Pi_{M,H}} \|\mathbf{y}_t - \mathbf{y}_t^\pi\|^2 = O(\sqrt{T}).$$

Chapter 6

Spectral filtering for learning unknown LDS

In the previous section we have showed how to provably learn an unknown linear dynamical system using convex relaxation and online learning. Instead of learning the system matrices directly, we have learned an alternate class of predictors, $\Pi_{M,H}$, that approximates the optimal predictor in a precise sense. We then leveraged the convex structure of the latter class to learn it via online iterative methods.

Despite the clean and elegant solution, there is an unsatisfying issue with the previous chapter. The number of parameters in the new class depends on the parameter $\frac{1}{\delta}$ of the system, which is potentially large and unknown by itself.

In this section we describe a particularly elegant alternative method for learning an LDS that removes this dependence on δ. For the sake of exposition and elegance, we restrict ourselves only to symetric system matrices A. This is a significant restriction, which can be removed, albeit using a significantly more sophisticated approach, and thus omited from these lectures.

6.1 Spectral filtering: the 1-D case

The most natural to explain spectral filtering is to first explain the scalar case. The extension to higher dimensional symmetric systems is not difficult using the eigendecomposition, and applying the scalar intuition for each eigendirection. We will return to it later.

Scalar systems have $d_u = d_y = 1$, even though the hidden dimension can potentially be larger. We assume for simplicity that $\mathbf{x}_0 = 0$. Recall the set

of optimal predictors, the class $\Pi_\star$, which for scalar systems can written and simplified to,

$$\mathbf{y}_t = \mathbf{y}_t^\pi \quad = \sum_{i=1:t} CA^{i-1}B\mathbf{u}_{t-i} = \beta\gamma \sum_{i=1:t} \alpha^{i-1}\mathbf{u}_{t-i}.$$

As standard in these notes, we denote $\mathbf{x}_{1:k} = [\mathbf{x}_1 \ \dots \ \mathbf{x}_k]$. Thus, we can write the difference between two consecutive iterates as

$$\begin{aligned} \mathbf{y}_t^\pi - \mathbf{y}_{t-1} &= \beta\gamma(\textstyle\sum_{i=1:t} \alpha^{i-1}\mathbf{u}_{t-i} - \sum_{i=1:t-1} \alpha^{i-1}\mathbf{u}_{t-1-i}) \\ &= \beta\gamma(\mathbf{u}_{t-1} + \textstyle\sum_{i=1:t-1}(\alpha^i - \alpha^{i-1})\mathbf{u}_{t-1-i}) \\ &= \beta\gamma \cdot \mu_\alpha(\mathbf{u}_{t-1:1}), \end{aligned} \tag{6.1}$$

which is a geometric series involving α multiplying the past controls. We used the operator notation to denote the operator

$$\mu_\alpha(\mathbf{u}_{t-1:1}) = \mathbf{u}_{t-1} + \sum_{i=1:t-1} (\alpha - 1)\alpha^{i-1}\mathbf{u}_{t-1-i} = \mu_\alpha^\top \tilde{\mathbf{u}}_{t-1}.$$

Above we used the vector notation for

$$\mu_\alpha = \begin{bmatrix} 1 & \alpha - 1 & \alpha^2 - \alpha & \dots & \alpha^{T-1} - \alpha^{T-2} \end{bmatrix} \in \mathbb{R}^T,$$

We define $\tilde{\mathbf{u}}_t$ to be

$$\tilde{\mathbf{u}}_t = \begin{bmatrix} \mathbf{u}_t & \dots & \mathbf{u}_1 & 0 \dots & 0 \end{bmatrix} \in \mathbb{R}^T.$$

The idea is going to be to rewrite the vectors μ_α in a more *efficient* basis, as opposed to the standard basis. The notion of efficiency stems from the definition of ε-approximation we have studied in the previous chapter.

To define the basis, we consider the following matrix:

$$Z = \int_0^1 \mu(\alpha) \otimes \mu(\alpha)^\intercal d\alpha.$$

Z is a constant matrix independent from α and it is also symmetric. Let $\phi_1, ..., \phi_k \in \mathbb{R}^T$ to be the top k eigenvectors of Z.

6.1.1 The magic of Hankel matrices

The matrix Z is a special case of a **Hankel Matrix**, which is a square Hermitian matrix with each ascending skew-diagonal from left to right having the same value. An example of a Hankel matrix is the following,

$$Z = \int_0^1 \begin{bmatrix} 1 & \alpha & \alpha^2 & & \\ \alpha & \alpha^2 & \alpha^3 & & \\ \alpha^2 & \alpha^3 & \alpha^4 & & \\ & & & \dots & \\ & & & & \alpha^{2T-2} \end{bmatrix} d\alpha = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & & \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & & \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & & \\ & & & \dots & \\ & & & & \frac{1}{2T-1} \end{bmatrix}$$

The properties of Hankel matrices have been studied extensively, and the following theorem captures one of their crucial properties.

Theorem 6.1 ([BT17, HSZ17])**.** *Let σ_k be the k-th largest singular value of a Hankel matrix Z, then there exists a constant c_1 such that :*

$$\sigma_k(Z) \leq c_1 \cdot e^{-\frac{k}{\log T}}$$

This theorem shows that the eigenvalues of a Hankel matrix decay very rapidly, as shown in Figure 6.1 on a logarithmic scale.

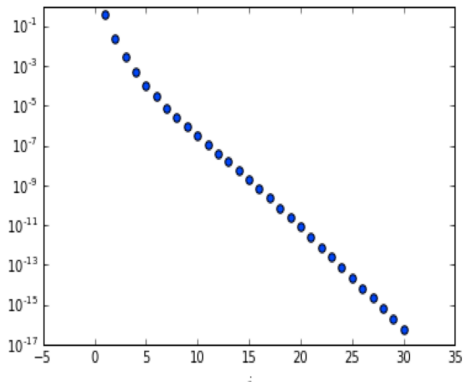


Figure 6.1: Eigenvalues of Hankel matrices decrease geometrically.

It follows that even if we sum up all singular values in this matrix that are bigger than $O(\log \frac{c_1 T}{\varepsilon})$, their magnitude it is going to be smaller than ε. These fact are going to be crucially used when we define the class of spectral predictors next.

6.1.2 The class of spectral predictors

The special basis we have just discussed, gives rise to a very interesting class of predictors. This class of mappings do not directly apply to the history

of inputs $\mathbf{u}_{1:t}$. Instead, we first apply a linear transformation over these vectors, and then apply a linear map.

More formally,

Definition 6.2. *Define the class of spectral linear predictors of order k as*

$$\Pi_k^s = \left\{ \pi_M \;\; | \;\; \pi_M(\mathbf{u}_{1:t}, \mathbf{x}_{1:t}) = \sum_{i=1}^{k} M_i \cdot \phi_i(\tilde{\mathbf{u}}_{t-i}) \right\}.$$

Here the operator $\phi_i(\mathbf{u}_{t-1})$ takes the inner product of the i'th eigenvector of Z with $\tilde{\mathbf{u}}_{t-1}$, where $\tilde{\mathbf{u}}_t = \begin{bmatrix} \mathbf{u}_t & \dots & \mathbf{u}_1 & 0 \dots & 0 \end{bmatrix} \in \mathbb{R}^T$.

One nice property about this class is that it is **linear, just in a different basis**. The reader may wonder at this point: "a linear transformation composed with another linear transform, remains linear. What then is the benefit here?".

Excellent question indeed! While linear transformations are closed under composition, their approximation properties with low rank constraints may differ drastically! This is the case in our setting, and we exploit it to our advantage: the spectral basis arising from the matrix Z gives excellent approximations to the previous class of predictors, as we formally prove next.

The following lemma is analogous to Lemma 5.3, with the significant difference that k does not depend on the parameter δ, which appears in the spectral radius of the system, at all!

Lemma 6.3. *Suppose that $\mathbf{x}_0 = 0$, then the class Π_k^s for $k = 2 \log \frac{4T^2 c_1}{\varepsilon}$, ε-approximates the class $\Pi_\star$.*

The proof of this lemma relies on the following crucial property of the spectral basis, whose proof we defer to later.

Lemma 6.4. *Let $\{\phi_j\}$ be the eigenvalues of Z. Then for all j,a:*

$$|\phi_j^\top \mu_\alpha| \le 4\sigma_j \le 4c_1 e^{-\frac{j}{\log T}}$$

Using this property, we can prove lemma 6.3, by simply shifting to the spectral basis representation as follows.

Proof of Lemma 6.3. Consider any sequence and any predictor from $\prod_\star$, say $\pi_1 = \pi_{A,B,C,0}$. Following equation (6.1), we can write the optimal predictor in the spectral basis:

$$\mathbf{y}_t^{\pi_1} - \mathbf{y}_{t-1} = \beta\gamma \cdot \mu_\alpha(\mathbf{u}_{t-1:1}) = \textstyle\sum_{i=1}^{T} M_i^\star \phi_i(\tilde{\mathbf{u}}_{t-i}),$$

where the optimal coefficients $M_i^\star = \phi_j^\top \mu_\alpha$ satisfy the following by Lemma 6.4

$$\phi_j^\top \mu_\alpha \le 4c_1 e^{-\frac{i}{\log T}}.$$

Consider the predictor $\pi_2 = \pi_M \in \prod_k^s$ for the same system matrices $M^\star$. Then we have

$$\begin{aligned}
&\|\mathbf{y}_t^{\pi_1} - \mathbf{y}_t^{\pi_2}\| \\
&= \|\textstyle\sum_{i=1}^T M_i^\star \phi_i(\tilde{\mathbf{u}}_{t-i}) - \sum_{i=1:k} M_i^\star \phi_i(\tilde{\mathbf{u}}_{t-i})\| \\
&= \|\textstyle\sum_{i=k+1:T} M_i^\star \phi_i(\tilde{\mathbf{u}}_{t-1})\| \\
&\le \textstyle\sum_{i=k+1:T} |M_i^\star| \|\phi_i(\tilde{\mathbf{u}}_{t-1})\| \\
&\le \textstyle\int_{i>k} 4c_1 e^{-\frac{i}{\log T}} \le 4c_1 \log T e^{-\frac{k}{\log T}} < \varepsilon.
\end{aligned}$$

□

It remains to prove Lemma 6.4, which is the main technical lemma of this section. The intuition is by now complete, and the proof is provided only for the mathematically inclined readers.

Proof. First, notice that by definition,

$$\int_\alpha \langle \phi_j, \mu_\alpha \rangle^2 d\alpha = \int_\alpha \phi_j^\intercal \mu_\alpha \mu_\alpha^\intercal \phi_j d\alpha = \phi_j^\intercal Z \phi_j = \sigma_j$$

On the other hand, we claim that the function $g(\alpha) = (\mu_\alpha^\top \phi_j)^2$ is 2-Lipschitz. To see this, first we prove that the function $f(\alpha) = \|\mu_\alpha\|^2$ is 1-Lipschitz. To see this, we take the gradient explicitly, for $\alpha \in [0,1]$.

$$\begin{aligned}
f'(\alpha) &= \tfrac{\partial}{\partial\alpha}|\mu_\alpha|^2 = \tfrac{\partial}{\partial\alpha}\textstyle\sum_{i=0:T-1}(1-\alpha)^2 a^{2i} \le \tfrac{\partial}{\partial\alpha}\tfrac{(1-\alpha)^2}{1-\alpha^2} \\
&= \tfrac{\partial}{\partial\alpha}\tfrac{1-\alpha}{1+\alpha} = \tfrac{2}{(1+\alpha)^2} \le 2
\end{aligned}$$

The function $g(\alpha)$ is a projection of $f(\alpha) = |\mu_\alpha|^2$. onto a line, and therefore it is also 2-Lipschitz. Let $R = \max_\alpha g(\alpha) \le 1$. Any non-negative 2-Lipschitz function that reaches R and has the largest area as an area of $\frac{R^2}{4}$ from 0 to 1. Thus, the maximum value of $g(\alpha)$ is bounded by, using Theorem 6.1,

$$\max_\alpha |\langle \phi_j, \mu_\alpha \rangle| \le 4 \int_\alpha \langle \phi_j, \mu_\alpha \rangle^2 d\alpha \le 4\phi_j \le 4c_1 e^{-\frac{j}{\log T}}.$$

□

6.1.3 Generalization to higher dimensions

For the full details of how to generalize the spectral technique and analysis to higher dimensions, see bibliographic section. Here we just give brief intuition for the symmetric case.

Consider a system with dynamic matrices $A.B.C$, and further assume that A is symmetric, and thus diagonalizable over the real numbers as $A = U^\intercal DU$. For every dimension separately, we take a separate μ_α vector, and using the same notation as equation (6.1):

$$\begin{aligned}
\hat{\mathbf{y}}_t^\pi - \mathbf{y}_{t-1} - CB\mathbf{u}_{t-1} &= \sum_{i=1:t-1} CU^\top (D^{i-1} - D^{i-2}) UB\mathbf{u}_{t-1-i} \\
&= \sum_{i=1:t-1} \tilde{C}^\top (\sum_{j=1}^{d_x} (\alpha^{i-1} - \alpha^{i-2}) e_i e_i^\top) \tilde{B}\mathbf{u}_{t-i-1} \\
&= \sum_{j=1}^{d_x} \tilde{C}_j^\top \tilde{B}_j \sum_{i=1:t-1} (\alpha^{i-1} - \alpha^{i-2}) \mathbf{u}_{t-1-i} \\
&= \sum_{j=1}^{d_x} \tilde{M}_j \mu_{\alpha_j}(\mathbf{u}_{1:t-1})
\end{aligned}$$

Thus, we can apply the same exact approximation techniques and Lemma 6.3 holds for the higher dimensional matrix case as well.

The class H_k^s can be learned by online gradient descent in the online convex optimization framework, giving rise to the following algorithm:

Algorithm 3 online spectral filtering

1: Input: $M_{1:k}^1$, convex constraints set $\mathcal{K} \subseteq \mathbb{R}^{H \times d_y \times d_u}$, horizon T
2: Compute k top eigenvectors of Z_T, denoted $\phi_1, ..., \phi_k$.
3: **for** $t = 1$ to T **do**
4: Predict $\hat{\mathbf{y}}_t = \sum_{j=1}^k M_j^t \phi_j(\tilde{\mathbf{u}}_{t-i})$, as per definition 6.2, and observe true $\mathbf{y}_t$.
5: Define cost function $f_t(M) = \|\mathbf{y}_t - \tilde{\mathbf{y}}_t\|^2$.
6: Update and project:

$$\begin{aligned}
M^{t+1} &= M^t - \eta_t \nabla f_t(M^t) \\
M^{t+1} &= \prod_{\mathcal{K}} (M^{t+1})
\end{aligned}$$

7: **end for**

The eigenvectors of Z have an interesting structure: their magnitude does not decay with time, which explains their improved approximation as captured formally in Lemma 6.3.

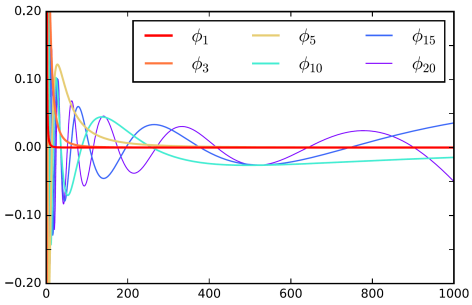


Figure 6.2: The filters obtained by the eigenvectors of Z.

The online gradient descent theorem, as per the previous chapter, directly gives the following corollary,

Theorem 6.5. *For choice of step sizes $\eta_t = O(\frac{1}{\sqrt{t}})$, we have that*

$$\text{regret}(\Pi_k^s) = \sum_t \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|^2 - \min_{\pi \in \Pi_k^s} \|\mathbf{y}_t - \mathbf{y}_t^\pi\|^2 = O(\sqrt{T}).$$

6.2 Bibliographic Remarks

The spectral properties of Hankel matrices used in this chapter are from [BT17]. The improper learning technique using the eigenvectors of the Hankel matrix Z was introduced in [HSZ17]. For an extension to a-symmetric matrices, see [HLS$^+$18].

Chapter 7

Online and Non-Stochatic Control

In this lecture we consider the main new directions that have been taken in the machine learning literature for the control problem.

These new developments are a major departure from classical control, and in recent years produced methods that compete favorably and many times surpass classical methods. A few characteristics of the new approaches:

1. Instead of pre-computing an optimal policy, online control seeks to learn a policy using online learning techniques. This fits the framework called "adaptive control".

2. In contrast to adaptive control, online control optimizes a rigorous performance metric of regret vs. the best policy in hindsight. Online and non-stochastic control algorithms have provable finite-time regret guarantees.

3. In online / non-stochastic control, the loss functions can be general Lipschitz convex losses that are chosen adversarially, as opposed to known quadratic losses in classical control.

4. Online and non-stochastic control deploy improper learning by convex relaxation: this means they compete with a certain policy class using a different policy class. This difference in learned predictors allows us to side-step computational hardness results we have previously explored, and obtain rigorous guarantees for efficient algorithms.

7.1 From Optimal and Robust to Online Control

Recall that the problem of optimal control of a linear time-invariant dynamical system (LDS) is given by the mathematical equations

$$\min_{\mathbf{u}(\mathbf{x})} \sum_{t=1}^{T} c_t(\mathbf{x}_t, \mathbf{u}_t)$$
$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t,$$

where $\mathbf{x}_t$ is the state of the system, $\mathbf{u}_t$ is the control input and $\mathbf{w}_t$ is the perturbation at time t.

Previous lectures showed the optimal control policy is a linear function of the state when we assume $\{\mathbf{w}_t\}_{t\in[T]}$ to be i.i.d. zero mean and bounded variance perturbations and c_t to be quadratic loss functions. Specifically, for positive semi-definite matrices $\mathbf{Q}$ and $\mathbf{R}$, the mathematical control formulation becomes

$$\min_{\mathbf{u}(\mathbf{x})} \sum_{t=1}^{T} \mathbf{x}_t^T \mathbf{Q}\mathbf{x}_t + \mathbf{u}_t^T \mathbf{R}\mathbf{u}_t$$
$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t, \quad \mathbb{E}\mathbf{w}_t = \mathbf{0},\ \|\mathbf{w}_t\| \leq 1 \ w.h.p..$$

We have shown in chapter 4 that the optimal solution to this formulation is given by the policy

$$\mathbf{u}_t^* = \mathbf{K}\mathbf{x}_t,$$

where $\mathbf{K}$ depends on $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$ and $\mathbf{R}$. The proof follows by inductively solving the bellman optimality equation at each step t, starting from time step T and moving backwards to time step 1.
However, this optimal solution requires two conditions:

1. The loss function needs to be a quadratic function.

2. The perturbations are i.i.d. across time (with additional constraints on the mean and the variance).

These conditions are in fact necessary for solving the Bellman optimality equation. Thus, this approach is not as general as we would like. We are missing out on other loss functions commonly used in practice, e.g. cross entropy or loss functions with regularization terms.

Furthermore and more importantly, many times the perturbations incurred in practice are not i.i.d., in which case we loose all provable guarantees!

An alternative to optimal control is robust, or H_∞ control theory. In this formulation, the goal is to optimize the control for an adversarial noise in a minimax sense, as per the mathematical formulation:

$$\min_{\mathbf{u}(\mathbf{x})} \max_{\mathbf{w}_{1:T}:\|\mathbf{w}_t\|\leq 1} \sum_{t=1}^{T} c_t(\mathbf{x}_t, \mathbf{u}_t)$$
$$\mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t,$$

This approach also suffers from several downsides:

1. The above problem is computationally ill-behaved for non quadratic loss functions.

2. It is non-adaptive in nature, i.e. the system is too pessimistic for cases when the perturbations are well-behaved, like i.i.d. Gaussian perturbations.

The natural question at this point is **is there an approach that gives the best of both robust and optimal control, and generalizes to any convex cost function?** That's exactly non-stochastic control theory we define next. But first, a motivating example.

7.1.1 Motivating Example for Online Control

Consider the problem of flying a drone from source to destination subject to unknown weather conditions. The aerodynamics of flight can be modeled sufficiently well by time-varying linear dynamical systems, and existing techniques are perfectly capable of doing a great job for indoors flight. However, the wind conditions, rain and other uncertainties are a different story. Certainly the wind is not an i.i.d. Gaussian random variable! Optimal control theory, which exactly assumes this zero-mean noise, is therefore overly optimistic and inaccurate for our problem.

The designer might resort to robust control theory to account for all possible wind conditions, but this is overly pessimistic. What if we encounter a bright sunny day after all? Planning for the worst case would mean slow and fuel-inefficient flight.

We would like an adaptive control theory which allows us to attain the best of both worlds: an efficient and fast flight when the weather permits,

and a careful drone maneuver when this is required by the conditions. Can we design a control theory that will allow us to take into account the specific instance perturbations and miss-specifications, and give us finite time provable guarantees? This is the subject of online non-stochastic control!

7.2 The Non-stochastic Control Problem

When dealing with non-stochastic, a.k.a. arbitrary or even adversarial perturbations, the optimal policy is not a-priory clear. Rather, an optimal policy for the observed perturbations can be **determined only in hindsight**.

This is the reason that non-stochastic control shifts the performance metric from optimality, to regret vs. the best policy in hindsight from a certain reference class.

At this point the student/reader may wonder why should we compare to a reference class, as opposed to all possible policies? There are two main answers to this question:

First - the best policy in hindsight may be very complicated to describe and reason about. This argument was made by the mathematician Tyrell Rockafellar, see the bibliographic section for more details. Secondly, it can be shown that it is impossible, in general, to obtain sublinear regret vs. the best policy in hindsight. Rather, a different performance metric called competitive ratio can be analyzed. This will be surveyed later in the course.

We now turn to define policy regret with respect to a reference class of policies. In online control, the controller iteratively chooses a control $\mathbf{u}_t$, and then observes the next state of the system $\mathbf{x}_{t+1}$ and suffers a loss of $c_t(\mathbf{x}_t, \mathbf{u}_t)$, according to an adversarially chosen loss function. Let $\Pi = \{\pi : \mathbf{u} \mapsto \mathbf{x}\}$ be a class of policies. The regret of the controller w.r.t. Π is defined as follows.

Definition 7.1. *Let $\{\mathbf{A}_t, \mathbf{B}_t\}$ be a time varying LDS. The regret of an online control algorithm over T iterations w.r.t. class of policies Π is given by:*

$$\text{regret}_T(\mathcal{A}, \Pi) = \max_{\mathbf{w}_{1:T}:\|\mathbf{w}_t\|\leq 1} \left(\sum_{t=1}^{T} c_t(\mathbf{x}_t, \mathbf{u}_t) - \min_{\pi\in\Pi} \sum_{t=1}^{T} c_t(\mathbf{x}_t^\pi, \mathbf{u}_t^\pi) \right),$$

where $\mathbf{u}_t = \mathcal{A}(\mathbf{x}_t)$ are the controls generated by $\mathcal{A}$, and $\mathbf{x}_t^\pi, \mathbf{u}_t^\pi$ are the counterfactual state sequence and controls under the policy π, i.e.

$$\begin{aligned} \mathbf{u}_t^\pi &= \pi(\mathbf{x}_t^\pi) \\ \mathbf{x}_{t+1}^\pi &= \mathbf{A}_t \mathbf{x}_t^\pi + \mathbf{B}_t \mathbf{u}_t^\pi + \mathbf{w}_t \end{aligned}$$

Henceforth, if T and $\Pi, \mathcal{A}$ are clear from the context, we just refer to regret without quantifiers.

The non-stochastic control problem can now be stated, as to find an efficient algorithm that minimizes the worst-case regret vs. meaningful policies classes that we examine henceforth. It is important to mention that **the algorithm does not have to belong to the comparator set of policies Π!** Indeed, in the most powerful results we will see the algorithm will learn a policy class that is strictly larger than the comparator class.

7.3 Classical and new policy classes for linear dynamical systems

In this section we give several prominent classes of policies that have been considered in the literature. We start by giving a formal method of comparing these classes. This treatment is analogous to that of comparing prediction algorithms from Chapter 5.

Definition 7.2. *We say that a class of policies Π_1 ε-approximates class Π_2 if the following holds. For every time-varying dynamics, and every sequence of T disturbances and cost functions, and every $\pi \in \Pi_1$, there exists $\pi_2 \in \Pi_2$ such that,*

$$\sum_{t=1}^{T} |c_t(\mathbf{x}_t^{\pi_1}, \mathbf{u}_t^{\pi_1}) - c_t(\mathbf{x}_t^{\pi_2}, \mathbf{u}_t^{\pi_2})| \leq T\varepsilon.$$

Similar to our treatment of prediction rules, the significance of this definition is that if a simple predictor class approximates another more complicated class, then it suffices to consider only the first. However, notice that seemingly the condition of approximation is much more stringent for control than it is for prediction, since the state evolutions of the two policies π_1, π_2 is potentially very difference.

The usefulness of this definition stems from its implication for regret minimization as follows.

Lemma 7.3. *Let policy class Π_1 ε-approximate policy class Π_2. Then for any control algorithm $\mathcal{A}$ we have*

$$\text{regret}_T(\mathcal{A}, \Pi_2) \leq \text{regret}_T(\mathcal{A}, \Pi_1) + \varepsilon T.$$

Proof of this lemma is left as an exercise to the reader.

We now proceed to defined several important policy classes and the approximation relationships between them.

7.3.1 Linear Policies

The most elementary and useful policy classes for control of linear dynamical systems are linear policies, i.e. those that chose a control signal as a (fixed) linear function of the state. Formally,

Definition 7.4 (Linear Policies)**.** *A linear controller $\mathbf{K} \in \mathbb{R}^{d_u \times d_x}$ is a linear operator mapping state to control. We denote by Π_κ^{LIN} the set of all linear controllers bounded in Fronbenius norm by $\|\mathbf{K}\| \leq \kappa$.*

The most important subcase of linear policies are those that are γ-stabilizing, meaning that the signal they create does not cause an unbounded state. This was formalized in definition 3.8.

We denote by $\Pi_{\kappa,\gamma}^{LIN}$ the set of all linear controllers bounded in Fronbenius norm by $\|\mathbf{K}\| \leq \kappa$, that are γ-stabilizing for a given LDS.

7.3.2 Linear Dynamic Controllers

The next policy class is particularly useful for linear dynamical systems with partial observation. Linear Dynamical Controllers simulate an "internal linear dynamical system" so to recover a hidden state, and play a linear function over this hidden state. The formal definition is given below.

Definition 7.5 (Linear Dynamic Controllers)**.** *A linear dynamic controller π has parameters $(\mathbf{A}_\pi, \mathbf{B}_\pi, \mathbf{C}_\pi, \mathbf{D}_\pi)$, and chooses a control at time t according to*

$$\mathbf{u}_t^\pi = \mathbf{C}_\pi \mathbf{s}_t + \mathbf{D}_\pi \mathbf{x}_t \ , \ \mathbf{s}_{t+1} = \mathbf{A}_\pi \mathbf{s}_t + \mathbf{B}_\pi \mathbf{x}_t.$$

Clearly the class of policies LDC approximates linear policies without any error, since we can take $\mathbf{A}_\pi, \mathbf{B}_\pi, \mathbf{C}_\pi$ to be zero, and get a linear policy.

We denote by $\Pi_{\delta,\kappa,\gamma}^{LDC}$ the class of all LDC that are γ-stabilizing as per definition 3.8 with parameters that satisfy

$$\forall i \ , \ \|\mathbf{A}_\pi^i\| \leq \kappa(1-\delta)^i \ , \ (\|\mathbf{B}_\pi\| + \|\mathbf{C}_\pi\| + \|\mathbf{D}_\pi\|)^2 \leq \kappa.$$

LDCs are considered to be state-of-the-art in control of LDS, even for systems with full observation. We proceed to define even more powerful policy classes.

7.3.3 Generalized Linear Controllers

The following class of controllers is a generalization of linear controllers, that is more powerful than LDC, and thus useful to define and reason about.

Definition 7.6 (Generalized Linear Controllers)**.** *A generalized linear controller π has a parameters $\mathbf{M}_{0:H}, \kappa$, such that $\sum_i \|\mathbf{M}_i\| \leq \kappa$ and chooses a control at time t according to*

$$\mathbf{u}_t^\pi = \sum_{i=0}^{H} \mathbf{M}_i \mathbf{x}_{t-i}.$$

An important concept for generalized linear controllers is that of **lifting**, or creating a larger linear system that subsumes the GLC. Given a certain GLC, consider the system given by

$$\mathbf{z}_{t+1} = \tilde{\mathbf{A}}_t \mathbf{z}_t + \tilde{\mathbf{B}}_t \tilde{\mathbf{M}} \mathbf{z}_t + \mathbf{w}_t,$$

where

$$\tilde{\mathbf{A}}_t = \begin{array}{|c|c|c|c|} \hline \mathbf{A}_t & 0 & \dots & 0 \\ \hline 0 & I & \dots & 0 \\ \hline \vdots & \vdots & \dots & \vdots \\ \hline 0 & 0 & \cdots & I \\ \hline \end{array} \quad , \quad \tilde{\mathbf{B}} = \begin{array}{|c|c|c|c|} \hline \mathbf{B}_t & \mathbf{B}_t & \dots & \mathbf{B}_t \\ \hline 0 & 0 & \dots & 0 \\ \hline \vdots & \vdots & \dots & \vdots \\ \hline 0 & 0 & \dots & 0 \\ \hline \end{array} \quad , \quad \tilde{\mathbf{M}} = \begin{array}{|c|c|c|c|} \hline \mathbf{M}_0 & 0 & \dots & 0 \\ \hline 0 & \mathbf{M}_1 & \dots & 0 \\ \hline \vdots & \vdots & \vdots & \vdots \\ \hline 0 & 0 & \dots & \mathbf{M}_H \\ \hline \end{array}$$

It can be seen that a GLC for the original system is simply a linear controller for the lifted system with $\mathbf{K} = \tilde{\mathbf{M}}$! This observation will be useful to us later on.

We denote by $\Pi^{GLC}_{H,\kappa,\gamma}$ the class of all GLC with $\sum_{i=0}^{H} \|\mathbf{M}_i\| \leq \kappa$, and $\mathbf{M}_{1:H}$ being γ stabilizing as per definition 3.8. The following lemma shows that GLC approximate LDC (and of course linear controllers):

Lemma 7.7. *The class $\Pi^{GLC}_{H,\kappa,\gamma}$ ε-approximates the class $\Pi^{LDC}_{\kappa',\delta,\gamma'}$ for $H = \Omega(\frac{1}{\delta}\log(\frac{\gamma'\kappa'}{\delta\varepsilon}))$.*

Proof. Since we assume γ'-stabilizability, γ' is an upper bound on the magnitude on the states reached by any policy from $\Pi^{LDC}_{\delta,\kappa',\gamma'}$.

Via the definition of the LDC, we have

$$\mathbf{u}_t^\pi = \mathbf{C}_\pi \mathbf{s}_t + \mathbf{D}_\pi \mathbf{x}_t = ... = \sum_{i=1}^{t-1} \mathbf{C}_\pi \mathbf{A}_\pi^{i-1} \mathbf{B}_\pi \mathbf{x}_{t-i} + \mathbf{D}_\pi \mathbf{x}_t.$$

Using the fact that $\|\mathbf{A}_\pi^i\| \leq \kappa'(1-\delta)^i$, we have that all terms farther away than H in history are bounded,

$$\begin{aligned}
\left\| \sum_{i=H+1}^{t} \mathbf{C}_\pi \left(\mathbf{A}_\pi\right)^i \mathbf{B}_\pi \mathbf{x}_{t-i} \right\| &\leq \left\| \sum_{i=H+1}^{\infty} \mathbf{C}_\pi \left(\mathbf{A}_\pi\right)^i \mathbf{B}_\pi \mathbf{x}_{t-i} \right\| \\
&\leq \sum_{i=H+1}^{\infty} \left\| \mathbf{C}_\pi \left(\mathbf{A}_\pi\right)^i \mathbf{B}_\pi \right\| \|\mathbf{x}_{t-i}\| \\
&\leq \gamma' \kappa' \sum_{i=H+1}^{\infty} (1-\delta)^i \\
&\leq \gamma' \kappa' \int_{i=H}^{\infty} e^{-\delta i} \\
&= \gamma' \kappa' \frac{1}{\delta} e^{-\delta H} \leq \varepsilon,
\end{aligned}$$

provided γ' is an upper bound on the states achieved by π, since it is assumed to be γ'-stabilizing, and $H \geq \frac{10}{\delta} \log(\frac{\gamma' \kappa'}{\delta \varepsilon})$. It thus suffices to prove that we can approximate the first H terms, i.e. the term

$$\mathbf{v}_t^\pi \stackrel{\text{def}}{=} \sum_{i=1}^{H} \mathbf{C}_\pi \mathbf{A}_\pi^{i-1} \mathbf{B}_\pi \mathbf{x}_{t-i},$$

as we already shown that $\|\mathbf{v}_t^\pi - \mathbf{u}_t^\pi\| = O(\varepsilon)$. Indeed, $\mathbf{v}_t^\pi$ can be represented exactly in the class GLC by taking $\mathbf{M}_i = \mathbf{C}_\pi \mathbf{A}_\pi^{i-1} \mathbf{B}_\pi$ and $\mathbf{M}_0 = \mathbf{D}_\pi$! □

The class of GLC is thus very general and intuitive. It is, however, difficult to work with for the regret metric, due to its counterfactual structure.

7.3.4 Disturbance Action Controllers

Next, we define a recently-considered set of policies known as the disturbance action controllers, or DAC. The main advantages of DAC as opposed to GLC, LDC and linear controllers are that the cost of control is convex in their parametrization, and thus allows for provably efficient optimization. Further, this set of policies is stronger than GLC in the sense that it ε-approximates them, as we shortly see.

Definition 7.8 (Disturbance Action Controller)**.** *A disturbance action policy $\pi_{\mathbf{K}_{1:T}, \mathbf{M}_{1:H}}$ is parameterized by the matrices $\mathbf{M}_{1:H} = [\mathbf{M}_1, \ldots, \mathbf{M}_H]$ and*

stabilizing controllers $\{\mathbf{K}_t\}$. *It outputs control* $\mathbf{u}_t^\pi$ *at time* t *according to the rule*

$$\mathbf{u}_t^\pi = \mathbf{K}_t \mathbf{x}_t + \sum_{i=1}^{H} \mathbf{M}_i \mathbf{w}_{t-i}.$$

Denote by $\Pi_{H,\gamma,\kappa}^{DAC}$ the set of all disturbance action policies with $\mathbf{K}_{1:T}$ being γ-stabilizing linear controllers for the system and

$$\sum_{i=1}^{H} \|\mathbf{M}_i\| \leq \kappa.$$

The class of DAC **always produces a stabilizing control signal**, if $\mathbf{K}_{1:T}$ are sequentially stabilizing for the entire system $\{\mathbf{A}_t, \mathbf{B}_t\}$. This important fact can be seen in the following lemma.

Lemma 7.9. *Any DAC policy in* $\Pi_{H,\gamma,\kappa}^{DAC}$ *is* $\kappa\gamma$*- stabilizing.*

Proof. Denote by $\mathbf{M}_0 = I$, and the fictitious noise term $\tilde{\mathbf{w}}_t = \sum_{i=0}^{H} \mathbf{M}_i \mathbf{w}_{t-i}$. First, due to the bound on $\mathbf{M}_{1:H}$ and $\mathbf{w}_t$, we have that

$$\|\tilde{\mathbf{w}}_t\| = \|\sum_{i=0}^{H} \mathbf{M}_i \mathbf{w}_{t-i}\| \leq \sum_{i=0}^{H} \|\mathbf{M}_i\|_* \|\mathbf{w}_{t-i}\| \leq \kappa,$$

for $\|\mathbf{M}\|_*$ being the spectral norm of M.

Next, since $\mathbf{K}_t$ are stabilizing, we can bound the historically distant terms as below. Observe that the state at time t, following any DAC policy π, is given by

$$\begin{aligned}
\mathbf{x}_{t+1} &= \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t \\
&= \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t (\mathbf{K}_t \mathbf{x}_t + \textstyle\sum_{i=1}^{H} \mathbf{M}_i \mathbf{w}_{t-i}) + \mathbf{w}_t \\
&= (\mathbf{A}_t + \mathbf{K}_t \mathbf{B}_t) \mathbf{x}_t + \textstyle\sum_{i=0}^{H} \mathbf{M}_i \mathbf{w}_{t-i} \qquad \mathbf{M}_0 = I \\
&= \textstyle\sum_{j=0}^{t-1} \left(\left[\prod_{\tau=t-j}^{t} (\mathbf{A}_\tau + \mathbf{K}_\tau \mathbf{B}_\tau) \right] \sum_{i=0}^{H} \mathbf{M}_i \mathbf{w}_{t-i-j} \right) \\
&= \textstyle\sum_{j=0}^{t-1} \left(\left[\prod_{\tau=t-j}^{t} (\mathbf{A}_\tau + \mathbf{K}_\tau \mathbf{B}_\tau) \right] \tilde{\mathbf{w}}_{t-j} \right)
\end{aligned}$$

These are states generated by a γ-stabilizing policy over noises that are of magnitude κ, and hence bounded by $\kappa\gamma$. □

The above lemma is already a good indication about DAC: they generate controls that at least don't cause a blow up. But how expressive are these policies in terms of reaching the optimal solution?

One of the main appeals of the DAC class is the following approximation lemma, that shows that for time-invariant systems DAC are in fact a stronger class than GLC, and thus LDC. This is a strong guarantee, since it implies that competing with the best policy in DAC implies competing with the best GLC/LDC.

We start by showing that DAC approximate all linear controllers. This is not an empty statement, since the $\mathbf{K}_t$ terms for DAC is now only assumed to be stabilizing, and not necessarily optimal for the actual sequence.

Lemma 7.10. *Consider a time-invariant linear dynamical system* $\{\mathbf{A}, \mathbf{B}\}$. *The class* $\Pi^{DAC}_{H,\gamma}$ ε*-approximates the class* $\Pi^{LIN}_{\kappa,\delta}$ *for* $H = \Omega(\frac{1}{\delta}\log(\frac{\kappa}{\delta\varepsilon}))$.

Proof. Via the definition of a linear controller, we have

$$\mathbf{u}^{\pi}_t = \mathbf{K}\mathbf{x}^{\pi}_t.$$

We proceed to show that every state can be approximated using the previous disturbances as follows:

$$\begin{aligned}
\mathbf{u}^{\pi}_{t+1} = \mathbf{K}\mathbf{x}_{t+1} \qquad &= \mathbf{K}(\mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t) \\
&= \mathbf{K}(\mathbf{A} + \mathbf{B}\mathbf{K})\mathbf{x}_t + \mathbf{K}\mathbf{w}_t \\
&= \textstyle\sum_{i=0}^{t} \mathbf{K}\left(\mathbf{A} + \mathbf{B}\mathbf{K}\right)^i \mathbf{w}_{t-i} \\
&= \textstyle\sum_{i=0}^{H} \mathbf{K}\left(\mathbf{A} + \mathbf{B}\mathbf{K}\right)^i \mathbf{w}_{t-i} + Z_t,
\end{aligned}$$

where $Z_t = \sum_{i=H+1}^{t} \mathbf{K}\left(\mathbf{A} + \mathbf{B}\mathbf{K}\right)^i \mathbf{w}_{t-i}$. It remains to bound the magnitude of this residual term,

$$\begin{aligned}
|Z_t| &\le \left\| \sum_{i=H+1}^{t} \mathbf{K}\left(\mathbf{A} + \mathbf{B}\mathbf{K}\right)^i \mathbf{w}_{t-i} \right\| \\
&\le \sum_{i=H+1}^{t} (1-\delta)^{i+1} \left\|\mathbf{K}\mathbf{w}_{t-i}\right\| \\
&\le \kappa \sum_{i=H+1}^{\infty} (1-\delta)^i \le \kappa \int_{i=H}^{\infty} e^{-\delta i} \\
&= \kappa \frac{1}{\delta} e^{-\delta H} \le \varepsilon,
\end{aligned}$$

□

Recall that GLC can be lifted to a linear dynamical system. Thus a corllary of the above lemma is as follows.

Corollary 7.11. *Consider a time invariant linear dynamical system given by* $\{\mathbf{A}, \mathbf{B}\}$. *The class* $\Pi^{DAC}_{H,\gamma}$ *ε-approximates the class* $\Pi^{GLC}_{H',\gamma',\delta}$, *for* $H \geq \max\{H', \frac{1}{\delta}\log\frac{1}{\delta\varepsilon}\}$.

Proof. Consider a given LDS, GLC policy $\mathbf{u}_t = \sum_{i=0}^{H} \mathbf{M}_i \mathbf{x}_{t-i}$, and derived lifted system:

$$\mathbf{z}_{t+1} = \tilde{\mathbf{A}}\mathbf{z}_t + \tilde{\mathbf{B}}\tilde{\mathbf{M}}\mathbf{z}_t + \tilde{\mathbf{w}}_t,$$

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \cdots & I \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{B} & \dots & \mathbf{B} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}, \quad \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{M}_0 & 0 & \dots & 0 \\ 0 & \mathbf{M}_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \mathbf{M}_H \end{bmatrix}$$

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t-1} \\ \vdots \\ \mathbf{x}_{t-H} \end{bmatrix}, \quad \tilde{\mathbf{w}}_t = \begin{bmatrix} \mathbf{w}_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Following a similar derivation to the previous lemma, we have that

$$\mathbf{u}_t = \sum_{i=0}^{H} \mathbf{M}_i \mathbf{x}_{t-i} = \tilde{\mathbf{M}}\mathbf{z}_t = \sum_{i=0}^{H} \left(\tilde{\mathbf{A}} + \tilde{\mathbf{B}}\tilde{\mathbf{M}}\right)^i \tilde{\mathbf{w}}_{t-i} + Z_t,$$

where as before

$$Z_t = \sum_{i=H+1}^{t} \left(\tilde{\mathbf{A}} + \tilde{\mathbf{B}}\tilde{\mathbf{M}}\right)^i \tilde{\mathbf{w}}_{t-i}.$$

The first part of the expression for $\mathbf{u}_t$ is a DAC controller with H terms. The second part can be bounded assuming sequential stabilizability and $H \geq \frac{1}{\delta}\log\frac{1}{\delta\varepsilon}$,

$$\begin{aligned} |Z_t| &\leq \left\| \sum_{i=H+1}^{t} \left(\tilde{\mathbf{A}} + \tilde{\mathbf{B}}\tilde{\mathbf{M}}\right)^i \mathbf{w}_{t-i} \right\| \\ &\leq \sum_{i=H+1}^{t} (1-\delta)^{i+1} \|\mathbf{w}_{t-i}\| \leq \int_{i=H}^{\infty} e^{-\delta i} \\ &= \frac{1}{\delta} e^{-\delta H} \leq \varepsilon. \end{aligned}$$

The first part of the expression is exactly a DAC. □

7.4 The Gradient Perturbation Controller

After exploring the relationships between different classes of controllers, we are ready to prove the main new algorithm in non-stochastic control, which gives a strong regret guarantee against adversarially chosen perturbations.

The algorithm is described in figure (4), for general changing linear dynamical systems. The main idea is to learn the parametrization of a DAC policy using known online convex optimization algorithms, namely online gradient descent. We henceforth prove that this parametrization is essentially convex, allowing us to prove the main regret bound.

In algorithm (4) we denote $\mathbf{x}_t(\mathbf{M}_{1:H})$ as the state arising by playing DAC policy $\mathbf{M}_{1:H}$ from the beginning of time, and same for $\mathbf{u}_t(\mathbf{M}_{1:H})$.

Algorithm 4 Gradient Perturbation Controller(GPC)

Input: H, η, initialization $\mathbf{M}^1_{1:H}$.
for $t = 1 \dots T$ **do**
  Observe system $\mathbf{A}_t, \mathbf{B}_t$, compute a stabilizing linear controller for the current system $\mathbf{K}_t$ such that it is sequentially stabilizing
  Use Control $\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \sum_{i=1}^{H} \mathbf{M}^t_i \mathbf{w}_{t-i}$
  Observe state $\mathbf{x}_{t+1}$, compute noise $\mathbf{w}_t = \mathbf{x}_{t+1} - \mathbf{A}_t \mathbf{x}_t - \mathbf{B}_t \mathbf{u}_t$
  Construct loss $\ell_t(\mathbf{M}_{1:H}) = c_t(\mathbf{x}_t(\mathbf{M}_{1:H}), \mathbf{u}_t(\mathbf{M}_{1:H}))$
  Update $\mathbf{M}^{t+1}_{1:H} \leftarrow \mathbf{M}^t_{1:H} - \eta \nabla \ell_t(\mathbf{M}^t_{1:H})$
end for

The GPC algorithm comes with a very strong performance guaranteee: namely it guarantees vanishing worst case regret vs. the best DAC policy in hindsight. As per the relationship between DAC and other poicies, this implies vanishing regret vs. LDC, GLC and linear controllers. The formal statement is given as follows.

Theorem 7.12 (Theorem 5.1 in [ABH+19])**.** *Assuming that*

a The costs c_t are convex, bounded and have bounded gradients w.r.t. both arguments $\mathbf{x}_t$ and $\mathbf{u}_t$.

b The matrices $\{\mathbf{A}_t, \mathbf{B}_t\}$ have bounded ℓ_2 norm.

Then the GPC algorithm (4) ensures that

$$\max_{\mathbf{w}_{1:T}:\|\mathbf{w}_t\|\le 1} \left(\sum_{t=1}^{T} c_t(\mathbf{x}_t, \mathbf{u}_t) - \min_{\pi \in \Pi^{DAC}} \sum_{t=1}^{T} c_t(\hat{\mathbf{x}}_t, \pi(\hat{\mathbf{x}}_t)) \right) \le \mathcal{O}(\sqrt{T}).$$

Furthermore, the time complexity of each loop of the algorithm is polynomial in the number of system parameters and logarithmic in T.

Proof sketch only. Online gradient descent (OGD) on convex loss functions ensures that the regret is $\mathcal{O}(\sqrt{T})$ (please refer to Theorem 3.1 in [Haz19a] for more details). However, to directly apply the OGD algorithm to our setting, we need to make sure that the following 3 conditions hold true:

- The loss function should be a convex function in the variables $\mathbf{M}_{1:H}$.
- The loss function is time dependent because the loss depends on the current state, which further depends on the previous states. The state at each time step depends on the current variable estimates. Hence, $\mathbf{x}_t$ can be shown to be a function of $\left\{\mathbf{M}_{1:H}^i\right\}_{i=1}^t$, where $\mathbf{M}_{1:H}^i$ denotes the variables at step i. Thus, we need to make sure that optimizing the loss function is similar to optimizing a function which doesn't have such a dependence on time.

First, we show that the loss function is convex w.r.t. to the variables $\mathbf{M}_{1:H}$. This follows since the states and the controls are linear transformations of the variables.

Lemma 7.13. *The loss functions ℓ_t are convex functions in the variables $\mathbf{M}_{1:H}$.*

Proof. The loss function ℓ_t is given by

$$\ell_t(\mathbf{M}_{1:H}) = c_t(\mathbf{x}_t, \mathbf{u}_t).$$

Since, we assume that the cost function c_t is a convex function w.r.t. its arguments, we simply need to show that $\mathbf{x}_t$ and $\mathbf{u}_t$ depend linearly on $\mathbf{M}_{1:H}$. The state is given by

$$\begin{aligned}\mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t &= \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\left(\mathbf{K}_t\mathbf{x}_t + \sum_{i=1}^{H}\mathbf{M}_i\mathbf{w}_{t-i}\right) + \mathbf{w}_t \\ &= \tilde{\mathbf{A}}_t\mathbf{x}_t + \left(\mathbf{B}\sum_{i=1}^{H}\mathbf{M}_i\mathbf{w}_{t-i} + \mathbf{w}_t\right),\end{aligned}$$

where we denote $\tilde{\mathbf{A}}_t = \mathbf{A}_t + \mathbf{B}_t\mathbf{K}_t$. By induction, we can further simplify

$$\mathbf{x}_{t+1} = \sum_{i=0}^{t}\left(\prod_{j=1}^{i}\tilde{\mathbf{A}}_t\right)\left(\mathbf{B}_t\sum_{j=1}^{H}\mathbf{M}_j\mathbf{w}_{t-i-j} + \mathbf{w}_{t-i}\right),$$

which is a linear function of the perturbations. In the above computation, we have assumed that the initial state $\mathbf{x}_0$ is $\mathbf{0}$, otherwise this introduces just another small constant.

Similarly, the control $\mathbf{u}_t$ is given by

$$\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \sum_{i=1}^{H} \mathbf{M}_i \mathbf{w}_{t-i},$$

and since $\mathbf{x}_t$ was shown to be linear in the perturbations, so is $\mathbf{u}_t$. Thus, we have shown that $\mathbf{x}_t$ and $\mathbf{u}_t$ are linear transformations in $\mathbf{M}_{1:H}$ and hence, the loss function is convex in $\mathbf{M}_{1:H}$. □

Finally, the actual loss at time t is not determined by $\mathbf{x}_t(\mathbf{M}_{1:H})$, but rather different parameters $\mathbf{M}^i_{1:H}$ for various historical times $i < t$. The way to argue about loss functoins that change over time is using the framework of online convex optimization with memory.

In short, as we have argued before, we have that $\ell_t = f_t(\mathbf{M}^{(t-H)}_{1:H}, \cdots, \mathbf{M}^{(t)}_{1:H})$ for some convex function f_t. The crux of the argument is that for any sequence of L-lipschitz functions with memory q, $f_1, \cdots, f_T$, the following holds:

$$\sum_{t=1}^{T} f_t(\mathbf{M}^{t-q}_{1:H}, \cdots, \mathbf{M}^{t}_{1:H}) - \sum_{t=1}^{T} f_t(\mathbf{M}^{t}_{1:H}, \cdots, \mathbf{M}^{t}_{1:H}) \leq \mathcal{O}(\sqrt{q^2 LT}),$$

due to the small learning rate parameter. More detailes are referred to in the bibliographic section.

□

7.5 Bibliographic Remarks

Online control with stochastic noise. Online control was first consider in the seminal work of [AYS11] in the context of the online LQR problem. Recent work considers efficient algorithms and tighter bounds for the stochastic noise model [AYS11, DMM$^+$18, MTR19, CHK$^+$18, AYLS19]. In these works, a fully-observed linear dynamic system is driven by i.i.d. Gaussian noise and the learner incurs a quadratic state and input cost. Recent algorithms [MTR19, CKM19, CHK$^+$18] attain $\sqrt{T}$ regret for this setting, and are able to cope with changing loss functions. Regret bounds for partially observed systems were studied in [LAHA20b, LAHA20a, LAHA20c], the most general bounds for this setting are in [SSH20].

In a different line of work, direct learning of the optimal linear policy in the Gaussian noise setting via the policy gradient method was studied in [FGKM18].

Non-stochastic control. The bulk of the material of this chapter is from [ABH$^+$19], who consider a significantly more general and challenging setting in which the disturbances and cost functions are adversarially chosen, and the cost functions are arbitrary convex costs. Follow up work by [AHS19] achieves logarithmic pseudo-regret for strongly convex, adversarially selected losses and well-conditioned stochastic noise.

Under the condition of controllability, [HKS19] attain $T^{2/3}$ regret for adversarial noise/losses when the system is *unknown*. The adversarial system identification method was surveyed in the last part of this lecture.

More recently, non-stochastic control was extended to the partial observation setting in [SSH20], attaining tight regret bounds.

Chapter 8

Non-stochastic control with partial observation

In this section we explore the more general setting of control, one that allows for partial observation, in the non-stochastic case. Thus, we return to the full linear control model of

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \eta_t \\ \mathbf{y}_t &= \mathbf{C}\mathbf{x}_t + \xi_t.\end{aligned}$$

The definition of policy regret from the previous chapter needs to be refined a bit to capture this added generality. The setting is similar: the controller iteratively chooses a control $\mathbf{u}_t$, and then observes the next observation of the system $\mathbf{y}_{t+1}$ and suffers a loss of $c_t(\mathbf{y}_t, \mathbf{u}_t)$, according to an adversarially chosen loss function. The states $\mathbf{x}_t$ remain hidden.

Let $\Pi = \{\pi : \mathbf{y}_{1:t} \mapsto \mathbf{u}_t\}$ be a class of policies. The regret of the controller w.r.t. Π is defined as follows.

Definition 8.1. *Let $\{\mathbf{A}_t, \mathbf{B}_t\}$ be a time varying LDS. The regret of an online control algorithm over T iterations w.r.t. class of policies Π is given by:*

$$\text{regret}_T(\mathcal{A}, \Pi) = \max_{\xi_{1:T}, \mathbf{w}_{1:T}: \|\mathbf{w}_t\|, \|\xi_t\| \le 1} \left(\sum_{t=1}^{T} c_t(\mathbf{y}_t, \mathbf{u}_t) - \min_{\pi \in \Pi} \sum_{t=1}^{T} c_t(\mathbf{y}_t^\pi, \mathbf{u}_t^\pi) \right),$$

where $\mathbf{u}_t = \mathcal{A}(\mathbf{x}_t)$ are the controls generated by $\mathcal{A}$, and $\mathbf{y}_t^\pi, \mathbf{u}_t^\pi$ are the counterfactual state sequence and controls under the policy π, i.e.

$$\begin{aligned}\mathbf{u}_t^\pi &= \pi(\mathbf{y}_{1:t}^\pi) \\ \mathbf{x}_{t+1}^\pi &= \mathbf{A}_t \mathbf{x}_t^\pi + \mathbf{B}_t \mathbf{u}_t^\pi + \mathbf{w}_t \\ \mathbf{y}_{t+1}^\pi &= \mathbf{C}_t \mathbf{x}_t^\pi\end{aligned}$$

Henceforth, if T and $\Pi, \mathcal{A}$ are clear from the context, we just refer to regret without quantifiers.

8.0.1 Stabilizability in partial observability and changing systems

Henceforth we assume that we have sequentially stabilizable systems even with partial information. This is analogous to definition 3.10, and means that we have access, or can compute, a sequence of matrices $\{\mathbf{K}_t\}$ such that

$$\forall t \ . \ (\mathbf{A}_t + \mathbf{B}_t \mathbf{K}_t \mathbf{C}_t) = H_t L_t H_t^{-1} \ , \ \|H_t H_{t+1}^{-1}\| \leq 1 + \delta$$
$$\forall t_1, t_2 \ , \ \|H_{t_1}\| \|H_{t_2}^{-1}\| \leq \kappa$$
$$\|L_t\| \leq 1 - 3\delta \ , \ \tfrac{\kappa}{\delta} \leq \gamma.$$

The reason is that we can only apply controls to observation, and require the component $\mathbf{K}_t \mathbf{y}_t = \mathbf{K}_t \mathbf{C}_t \mathbf{y}_t$ to stabilize the system.

8.1 Disturbance Response Controllers

For the partial observability case, we consider another recently-considered set of policies known as the disturbance response controllers, or DRC. Similar to DAC, cost of control is convex in their parametrization, and thus allows for provably efficient optimization.

Definition 8.2 (Disturbance Response Controller)**.** *A disturbance response policy $\pi_{\mathbf{K}_{1:T}, \mathbf{M}_{1:H}}$ is parameterized by the matrices $\mathbf{M}_{1:H} = [\mathbf{M}_1, \ldots, \mathbf{M}_H]$ and stabilizing controllers $\{\mathbf{K}_t\}$. It outputs control $\mathbf{u}_t^\pi$ at time t according to the rule*

$$\mathbf{u}_t^\pi = \mathbf{K}_t \mathbf{y}_t + \sum_{i=1}^{H} \mathbf{M}_i \mathbf{y}_{t-i}^{\mathbf{nat}},$$

where $\mathbf{y}_t^{nat}$ is given by the system response to the zero control at time t, except for the stabilizing control part, i.e.

$$\begin{aligned} \mathbf{x}_{t+1}^{\mathbf{nat}} &= \mathbf{A}_t \mathbf{x}_t^{\mathbf{nat}} + \mathbf{K}_t \mathbf{C}_t \mathbf{x}_t^{\mathbf{nat}} + \mathbf{w}_t \\ &= \textstyle\sum_{i=0}^{t} \left[\prod_{j=0}^{i} (\mathbf{A}_{t-j} + \mathbf{K}_{t-j} \mathbf{C}_{t-j}) \right] \mathbf{w}_{t-i} = \Phi_t \mathbf{w}_{1:t}. \\ \mathbf{y}_{t+1}^{\mathbf{nat}} &= \mathbf{C}_{t+1} \mathbf{x}_{t+1}^{\mathbf{nat}} = \mathbf{C}_{t+1} \Phi_t \mathbf{w}_{1:t}. \end{aligned}$$

Here Φ_t is the linear operator such that $\Phi_t(t-i) = \prod_{j=0}^{i} (\mathbf{A}_{t-j} + \mathbf{K}_{t-j} \mathbf{C}_{t-j})$.

We conclude with the following important lemma on an expression that is important to compute nature's y's, which follows directly from the definition.

Lemma 8.3. *The sequence* $\{\mathbf{y}_t^{\mathbf{nat}}\}$ *can be iteratively computed using only the sequence observations as*

$$\mathbf{y}_t^{\mathbf{nat}} = \mathbf{y}_t - \mathbf{C}_t \sum_{i=0}^{t} \left[\prod_{j=0}^{i} (\mathbf{A}_{t-j} + \mathbf{K}_{t-j}\mathbf{C}_{t-j}) \right] \mathbf{B}_{t-i}\mathbf{u}_{t-i}$$

8.1.1 Relationship to other policy classes

The DRC always produces controls that ensure the system remains state-bounded. This can be shown analogously to lemma 7.9. We proceed to show that this class of policies is contained inside the class of DAC.

Lemma 8.4. *Consider a time-variant linear dynamical system* $\{\mathbf{A}_t, \mathbf{B}_t\}$. *The class* $\Pi_{2H,\gamma}^{DAC}$ ε*-approximates the class* $\Pi_{H',\gamma'}^{DRC}$ *for sufficiently large* H.

Proof. According to the definition of DRC policies, we have that

$$\mathbf{y}_{t+1}^{\mathbf{nat}} = \mathbf{C}_{t+1}\Phi_t\mathbf{w}_{1:t},$$

for a linear operator $\Phi_{1:t}$. For sequentially stabilizing controllers $\mathbf{K}_t$, we have that

$$\|\Phi_t(t-i)\| = \| \prod_{j=0}^{i} (\mathbf{A}_{t-j} + \mathbf{K}_{t-j}\mathbf{C}_{t-j}) \| \leq (1-\delta)^{i+1}.$$

Thus, we can define $\tilde{\Phi}_t$ to zero all entries after H:

$$\tilde{\Phi}_t(i) = \begin{cases} \Phi_t(i), & i \geq t - H \\ \\ 0, & o/w \end{cases},$$

and have, assuming $H > \frac{1}{\delta}\log\frac{1}{\delta\varepsilon}$,

$$\begin{aligned} \|\mathbf{y}_{t+1}^{\mathbf{nat}} - \mathbf{C}_{t+1}\tilde{\Phi}_t\mathbf{w}_{t:t-H}\| &= \|\mathbf{C}_{t+1}\textstyle\sum_{i=H+1}^{t}\Phi_t(i)\mathbf{w}_{t-i}\| \\ &\leq \textstyle\sum_{i=H+1}^{\infty}(1-\delta)^i \leq \frac{1}{\delta}e^{-\delta H} \leq \varepsilon \end{aligned}$$

Hence,

$$\begin{aligned} \mathbf{u}_t^{\pi} &= \mathbf{K}_t\mathbf{y}_t + \textstyle\sum_{i=1}^{H}\mathbf{M}_i\mathbf{y}_{t-i}^{\mathbf{nat}} \\ &= \mathbf{K}_t\mathbf{y}_t + \textstyle\sum_{i=1}^{H}\mathbf{M}_i\mathbf{C}_{t-1}\Phi_{t-i}\mathbf{w}_{1:t-i} \\ &= \mathbf{K}_t\mathbf{y}_t + \textstyle\sum_{i=1}^{H}\mathbf{M}_i\mathbf{C}_{t-1}\tilde{\Phi}_{t-i}\mathbf{w}_{t-H-i:t-i} \pm \varepsilon \\ &= \mathbf{K}_t\mathbf{y}_t + \textstyle\sum_{i=1}^{2H}\mathbf{N}_i\mathbf{w}_{t-i} \pm \varepsilon. \end{aligned}$$

The last expression is, of course, a control played by a certain DAC policy with parameter $2H$. □

While DRC are contained inside the policy class DAC, they are still very expressive. It can be shown that the class DRC with appropriate parameters ε-approximates the class of LDC for linear time-invariant systems.

8.2 The Gradient Response Controller

Partial observation requires a slightly different algorithm than the GPC we have studied in the previous chapter. The algorithm is described in figure (5), for general changing linear dynamical systems with partial observation. Similar to the GPC, the main idea is to learn the parametrization of a DRC policy using known online convex optimization algorithms, namely online gradient descent. We henceforth prove that this parametrization is essentially convex, allowing us to prove the main regret bound.

In algorithm (5) we denote $\mathbf{x}_t(\mathbf{M}_{1:H})$ as the state arising by playing DAC policy $\mathbf{M}_{1:H}$ from the beginning of time, and same for $\mathbf{u}_t(\mathbf{M}_{1:H})$.

Algorithm 5 Gradient Response Controller(GRC)

1: Input: sequence $\{\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t\}$, H, η, initialization $\mathbf{M}^1_{1:H}$.
2: **for** $t = 1 \dots T$ **do**
3: Compute a γ-sequentially stabilizing linear controller for the current system $\mathbf{K}_t$.
4: Use Control $\mathbf{u}_t = \mathbf{K}_t \mathbf{y}_t + \sum_{i=1}^{H} \mathbf{M}_i^t \mathbf{y}_{t-i}^{\mathbf{nat}}$
5: Observe $\mathbf{y}_{t+1}$, compute for $t+1$:

$$\mathbf{y}_t^{\mathbf{nat}} = \mathbf{y}_t - \mathbf{C}_t \sum_{i=0}^{t} \left[\prod_{j=0}^{i} (\mathbf{A}_{t-j} + \mathbf{K}_{t-j}\mathbf{C}_{t-j}) \right] \mathbf{B}_{t-i}\mathbf{u}_{t-i}.$$

6: Construct loss $\ell_t(\mathbf{M}_{1:H}) = c_t(\mathbf{x}_t(\mathbf{M}_{1:H}), \mathbf{u}_t(\mathbf{M}_{1:H}))$
Update $\mathbf{M}^{t+1}_{1:H} \leftarrow \mathbf{M}^t_{1:H} - \eta \nabla \ell_t(\mathbf{M}^t_{1:H})$
7: **end for**

The GRC algorithm comes with a very strong performance guaranteee: namely it guarantees vanishing worst case regret vs. the best DRC policy in hindsight. As per the relationship between DRC and other poicies, this implies vanishing regret vs. LDC. The formal statement is given as follows.

Theorem 8.5 (originally by [SSH20])**.** *Assuming that*

- *a The costs c_t are convex, bounded and have bounded gradients w.r.t. both arguments $\mathbf{y}_t$ and $\mathbf{u}_t$.*
- *b The matrices $\{\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t\}$ have bounded ℓ_2 norm.*

Then the GRC algorithm (5) ensures that

$$\max_{\xi_{1:T}, \mathbf{w}_{1:T}: \|\mathbf{w}_t\|, \|\xi_t\| \leq 1} \left(\sum_{t=1}^{T} c_t(\mathbf{y}_t, \mathbf{u}_t) - \min_{\pi \in \Pi^{DRC}} \sum_{t=1}^{T} c_t(\hat{\mathbf{y}}_t, \pi(\hat{\mathbf{y}}_t)) \right) \leq \mathcal{O}(\sqrt{T}).$$

Furthermore, the time complexity of each loop of the algorithm is polynomial in the number of system parameters and logarithmic in T.

Bibliography

[ABH+19] Naman Agarwal, Brian Bullins, Elad Hazan, Sham M Kakade, and Karan Singh. Online control with adversarial disturbances. *arXiv preprint arXiv:1902.08721*, 2019.

[ÅH95] Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.

[AHS19] Naman Agarwal, Elad Hazan, and Karan Singh. Logarithmic regret for online control. In *Advances in Neural Information Processing Systems*, pages 10175–10184, 2019.

[AJK19] Alekh Agarwal, Nan Jiang, and Sham M Kakade. Lectures on the theory of reinforcement learning. 2019.

[Aro20] Sanjeev Arora. personal communications. 2020.

[AYLS19] Yasin Abbasi-Yadkori, Nevena Lazic, and Csaba Szepesvári. Model-free linear quadratic control via reduction to expert prediction. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3108–3117. PMLR, 2019.

[AYS11] Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 1–26, 2011.

[BB08] Tamer Başar and Pierre Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[Ben93] S. Bennett. Development of the pid controller. *IEEE Control Systems Magazine*, 13(6):58–62, 1993.

[Ber07] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[BT00] Vincent D Blondel and John N Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

[BT17] Bernhard Beckermann and Alex Townsend. On the singular values of matrices with displacement structure. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1227–1248, 2017.

[BY08] Mark Braverman and Michael Yampolsky. *Computability of Julia sets*. Springer, 2008.

[CHK+18] Alon Cohen, Avinatan Hasidim, Tomer Koren, Nevena Lazic, Yishay Mansour, and Kunal Talwar. Online linear quadratic control. In *International Conference on Machine Learning*, pages 1029–1038, 2018.

[CKM19] Alon Cohen, Tomer Koren, and Yishay Mansour. Learning linear-quadratic regulators efficiently with only $\sqrt{T}$ regret. In *International Conference on Machine Learning*, pages 1300–1309, 2019.

[DMM+18] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pages 4188–4197, 2018.

[FCZI03] Enrique Fernández Cara and Enrique Zuazua Iriondo. Control theory: History, mathematical achievements and perspectives. *Boletín de la Sociedad Española de Matemática Aplicada, 26, 79-140.*, 2003.

[FGKM18] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 1467–1476, 2018.

[H+16] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[Haz19a] Elad Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019.

[Haz19b] Elad Hazan. Lecture notes: Optimization for machine learning, 2019.

[HKS19] Elad Hazan, Sham M. Kakade, and Karan Singh. The nonstochastic control problem, 2019.

[HLS+18] Elad Hazan, Holden Lee, Karan Singh, Cyril Zhang, and Yi Zhang. Spectral filtering for general linear dynamical systems. In *Advances in Neural Information Processing Systems*, pages 4634–4643, 2018.

[HSZ17] Elad Hazan, Karan Singh, and Cyril Zhang. Learning linear dynamical systems via spectral filtering. In *Advances in Neural Information Processing Systems*, pages 6702–6712, 2017.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82.1:35–45, 1960.

[LAHA20a] Sahin Lale, Kamyar Azizzadenesheli, Babak Hassibi, and Anima Anandkumar. Logarithmic regret bound in partially observable linear dynamical systems, 2020.

[LAHA20b] Sahin Lale, Kamyar Azizzadenesheli, Babak Hassibi, and Anima Anandkumar. Regret bound of adaptive control in linear quadratic gaussian (lqg) systems, 2020.

[LAHA20c] Sahin Lale, Kamyar Azizzadenesheli, Babak Hassibi, and Anima Anandkumar. Regret minimization in partially observable linear quadratic control, 2020.

[Lor63] Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, mar 1963.

[MTR19] Horia Mania, Stephen Tu, and Benjamin Recht. Certainty equivalence is efficient for linear quadratic control. In *Advances in Neural Information Processing Systems*, pages 10154–10164, 2019.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[Sil15] David Silver. Lecture notes on reinforcement learning. Lecture Notes, 2015.

[SSH20] Max Simchowitz, Karan Singh, and Elad Hazan. Improper learning for non-stochastic control, 2020.

[Str14] Steven H. Strogatz. *Nonlinear Dynamics and Chaos*. Taylor & Francis Inc, 2014.

[Ted20] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)*. 2020.

[Tuc99] Warwick Tucker. The lorenz attractor exists. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 328(12):1197–1202, 1999.

[ZDG96] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice-Hall, Inc., USA, 1996.