Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

AUTOMATIC
CONTROL
LABORATORY

Semester Project

Data-Driven Optimal Control via Linear Programming

Marina Draskovic
mdraskovic@student.ethz.ch

Advisors
Andrea Martinelli
Matilde Gargiani
Prof. Dr. John Lygeros

June 2021

Abstract

In the context of the linear programming (LP) approach to approximate dynamic programming, a recent paper introduced a relaxed version of the Bellman operator that can be employed to solve more scalable and efficient LPs. In this work, we aim at extending their fixed point analysis from linear quadratic settings to affine systems with generalized quadratic cost and non-zero mean noise. To this regard, we first compute analytic solutions to the fixed point of the Bellman operator for value and q-functions. Then, we show that the relaxed version of the operator preserves the optimal policy even though the associated optimal q-function is shifted. Theoretical results are backed up in simulation where we solve sampled versions of the LPs by letting the system interact with the environment and collect data in a reinforcement learning fashion. We perform an empirical study to compare iterative and non-iterative, on and off-policy methods based on the introduced LP framework.

Contents

Chapter 1

Introduction

Optimal control describes the problem of finding an input to a dynamical system for which some measure of a system's performance over a period of time is minimized. In 1952, Bellman suggested a way of solving this class of problems by using *dynamic programming* (DP), which relies on the *principle of optimality*, the fact that all subsolutions of an optimal control problem are themselves optimal [1]. A central place in this method is occupied by the *Bellman equation*. Bellman equation is a functional equation and its solution is a *value function*, a function of a state that describes the best possible value of the objective. Three standard dynamic programming algorithms for computing the value function are *policy iteration* (PI), *value iteration* (VI) and *linear programming* (LP).

One of the main drawbacks of dynamic programming methods is that they suffer from so-called *curse of dimensionality*, meaning that they are often computationally intractable. For this reason, a several *approximate dynamic programming* (ADP) methods work around this intractability.

Some of the ADP methods are model-free and do not use the system's dynamical equation. In these methods, the value function is learnt by letting the system interact with the environment and collecting the cost. This approach is called *reinforcement learning*. In order to use the reinforcement learning in DP methods, the Bellman equation is reformulated and, instead of the value function, the *q-function* is used. The main property of q-function is that the optimal control can be found without knowing the model of the system.

Linear systems with quadratic stage cost constitute a special class of optimal control problems for which analytic solutions can often be found ([2], [3]). That is why a lot of the aforementioned methods are extensively studied on linear time-invariant systems. As nonlinear dynamical equations that describe most real-world systems are challenging to solve, nonlinear systems can be locally approximated by the linear ones.

Another relevant class of dynamical systems is described by affine equations, which differ from their linear counterparts only by a constant shift. Because of this similarity to the linear systems, extending well-known control methods to *affine systems* is the first step towards working with nonlinear dynamics. Affine systems come in handy when working with various applications where control input appears in the dynamics linearly, for example modelling flexible joints or room cooling systems. In practice, when one encounters a system with affine dynamics, it is usually dealt with by shifting the coordinates by a constant term that appears in the dynamics. In the dynamic programming setup, however, such shift would affect the stage cost, and thus well-known derivations that hold for a linear system would not hold for an affine one.

This thesis aims is to extend some of the recent model-free optimal control techniques to the class of affine systems. Furthermore, numerous papers suggest different versions of data-driven LP and PI techniques for learning the optimal control policy ([4], [5], [6], [7], [8]). Thus, a part of this thesis is also dedicated to the empirical analysis of performance and precision of different data-driven techniques applied to affine systems.

1.1 Related work

An extensive literature on dynamic programming proposes PI, VI and LP algorithms for solving the Bellman equation, for example [3], [9]. Sources of intractability that often cause such methods to suffer from the curse of dimensionality are detailed in [10]. Some of the methods developed to alleviate such intractabilities, including the model-free methods that exploit the q-function formulation, are described in [9], [11]. The authors in [5] and [6] describe an LP approach to ADP and provide performance guarantees for both value and q-functions. The authors in [4] propose a relaxed Bellman operator and use it to build a new LP, called a relaxed linear program, which is more computationally efficient yet preserves essential mathematical properties. Moreover, in [7] and [8] the authors propose two data-driven versions of the policy iteration algorithm, which exploit an LP formulation in the policy evaluation step while ensuring the crucial monotonic behaviour property of PI algorithm. Much of the literature focuses on providing optimal control for linear systems where the cost function is pure quadratic. Finally, the authors in [12] suggest that dynamic programming can be tractably carried out for stochastic systems with affine dynamics and extended quadratic cost functions.

1.2 Contribution

Contributions of this thesis are as follows:

- Relaxing the linear quadratic assumptions and zero-mean noise assumptions and generalizing the discounted infinite-horizon LQR problem for stochastic time-invariant systems.
- Fixed point analysis using both regular and relaxed Bellman operator for stochastic time-invariant systems with affine dynamics, extended quadratic cost, and non-zero mean noise. We show that both versions of the operator yield the same optimal control policy.
- Building a linear program based on the relaxed Bellman operator for affine systems. Intractability sources in such program are discussed, and ways to alleviate the issues are proposed.
- The proposition of two q-function based policy iteration algorithms for stochastic affine systems, namely on-policy and off-policy. Both algorithms use linear programming in the policy evaluation step.
- Theoretical derivations are confirmed with a simulation, where the performance of LP and two versions of PI algorithms for different system settings are evaluated and compared through an empirical study.

1.3 Thesis outline

In the next chapter, the background information about the Bellman equation and dynamic programming is given. q-function is introduced, and a linear program and policy iteration algorithms that search for optimal q-function are formulated. Moreover, the standard Bellman operator is relaxed and utilized in order to build a relaxed linear program. The discussion continues on intractability sources of DP methods and proposing approximation methods that alleviate the curse of dimensionality. Next, two settings of a data-driven policy iteration algorithm that exploit the linear program in a policy evaluation step are introduced.

Chapter 3 introduces the stochastic affine time-invariant system with non-zero mean noise and extended quadratic cost. The rest of the chapter is dedicated to the derivation of an LQR and fixed point analysis for both Bellman operator and relaxed Bellman operator for the aforementioned system. In chapter 4, the simulation study that demonstrates the performance of q-function based DP methods on affine systems is described. Finally, the last chapter gives a conclusion and a short summary of the gained insights.

Chapter 2

Dynamic Programming Background

2.1 Problem Setup and Bellman Equation

We consider stochastic discrete-time systems of the form

$$x_{k+1} = f(x_k, u_k, \xi_k), \tag{2.1}$$

$f : \mathcal{X} \times \mathcal{U} \times \mathcal{E} \rightarrow \mathcal{X}$, where variable $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ denotes the state of the system at the discrete time k, and $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ denotes the control or decision variable applied to the system at time k. The system is stochastic because it depends on a random variable $\xi_k \in \mathcal{E} \subseteq \mathbb{R}^{n_\xi}$, called noise or disturbance.

The stage cost function $l(x_k, u_k),\ l : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^+$ measures the cost of playing input u_k at state x_k. The objective of DP algorithms is to minimize the total cost - a sum of stage costs over time:

$$\sum_{k=0}^{\infty} \gamma^k l(x_k, u_k). \tag{2.2}$$

$\gamma \in (0, 1)$ is a positive scalar called the discount factor. The presence of this factor implies that the costs incurred in the current time matter more than the ones from the future. Because of the disturbance ξ_k that appears in the dynamics, the above objective is reformulated as minimizing the expected value of the total cost:

$$\mathop{\mathbb{E}}_{\xi} \left\{ \sum_{k=0}^{\infty} \gamma^k l(x_k, u_k) \right\}. \tag{2.3}$$

If we denote with π the policy $\pi = \{\mu_0, \mu_1, ...\mu_{N-1}\}$, where each function μ_k maps states into actions as $u_k = \mu_k(x_k)$, then the system equation can be reformulated as

$$x_{k+1} = f(x_k, \mu_k(x_k), \xi_k). \tag{2.4}$$

and the expected value of the cost function, also known as the value function, associated with the policy π starting at initial state x_0 as

$$V_\pi(x_0) = \mathop{\mathbb{E}}_{\xi} \left\{ \sum_{k=0}^{\infty} \gamma^k l(x_k, \mu_k(x_k)) \right\}. \tag{2.5}$$

It is often the case that a policy π has the form $\pi = \{\mu, \mu, ...\}$, in which case it is called a stationary policy μ. The optimal policy π^* minimizes the expected total cost as

$$\pi^* = \mathop{\arg\min}_{\pi \in \mathcal{U}} V_\pi(x). \tag{2.6}$$

The value function associated with the optimal policy is thus also optimal:

$$V^*(x_0) = V_{\pi^*}(x_0) = \min_{\pi} V_\pi(x_0) = \min_{\pi} \mathop{\mathbb{E}}_{\xi} \left\{ \sum_{k=0}^{\infty} \gamma^k l(x_k, \mu_k(x_k)) \right\}. \tag{2.7}$$

The optimal value function at time k can be reformulated as

$$V_k^*(x_k) = \min_{\mu_k} \left\{ l(x_k, \mu_k(x_k)) + \gamma \mathop{\mathbb{E}}_{\xi} \left\{ V_{k+1}^*(f(x_k, \mu_k(x_k), \xi_k)) \right\} \right\}. \tag{2.8}$$

As time horizon goes to infinity, sequence of value functions V_k^* converges to the optimal value, and (2.8) becomes

$$\begin{aligned} V^*(x) &= l(x, \mu^*(x)) + \gamma \mathop{\mathbb{E}}_{\xi} \left\{ V^*(f(x, \mu^*(x), \xi)) \right\} \\ &= l(x, \mu^*(x)) + \gamma \mathop{\mathbb{E}}_{\xi} \left\{ V^*(x_{\mu^*}^+) \right\} \\ &= \min_{u \in \mathcal{U}} \left\{ l(x, u) + \gamma \mathop{\mathbb{E}}_{\xi} \left\{ V^*(x_u^+) \right\} \right\} \\ &= \mathcal{T} V^*(x), \quad \forall x \in \mathcal{X}. \end{aligned} \tag{2.9}$$

Equation (2.9) is known as the Bellman equation associated with the optimal policy, and the operator $\mathcal{T}$ as the Bellman operator. Note that, for the sake of compactness, we denote $f(x, u, \xi) = x_u^+$. Solution of this Bellman equation is the optimal value function. Bellman equation can also be defined for a stationary policy μ as

$$\begin{aligned} V_\mu(x) &= l(x, \mu(x)) + \gamma \mathop{\mathbb{E}}_{\xi} \left\{ V_\mu(x_\mu^+) \right\} \\ &= \mathcal{T}_\mu V_\mu(x). \end{aligned} \tag{2.10}$$

The optimal policy is stationary and thus time-invariant, and solving the minimization problem (2.9) for every $x \in \mathcal{X}$ results in that policy. The issue with the Bellman equation is that it needs to be solved for all $x \in \mathcal{X}$ simultaneously, which is possible to do analytically only for simple problems, for example LQR.

2.2 q-Function

Learning by trial and error comes as a natural method of problem-solving. Reinforcement learning uses this idea to learn how to optimally regulate the system without knowing the system's dynamics and form of the cost. Namely, the system can observe the environment, select and perform actions, and the ensuing outcomes and costs are collected over time. This comes in handy when one wants to obtain optimal control policy without the knowledge of dynamics f and cost l. In order to do so, equations from the previous subsection need to be reformulated in terms of a q-function [13].

q-function associated with policy μ is

$$q_\mu(x, u) = l(x, u) + \gamma \mathop{\mathbb{E}}_{\xi} \{ q_\mu(x_u^+, \pi(x_u^+)) \}, \quad \forall (x, u) \in \mathcal{X} \times \mathcal{U}, \tag{2.11}$$

and optimal q-function is

$$q^*(x,u) = l(x,u) + \gamma \mathop{\mathbb{E}}_{\xi}\{\min_{\omega\in\mathcal{U}} q_{\mu^*}(x^+_{\mu^*},\omega)\} = \mathcal{F}q^*(x,u), \quad \forall (x,u) \in \mathcal{X}\times\mathcal{U}. \tag{2.12}$$

This can be interpreted as a cost of applying applying control input u at state x and following a policy π afterwards. $\mathcal{F}$ is a Bellman operator for a q-function. Value and q-functions are related as

$$V^*(x) = \min_{u\in\mathcal{U}} q^*(x,u). \tag{2.13}$$

As was already mentioned, q-function formulation allows calculating the optimal policy without knowing the system. Optimal policy is calculated as

$$\pi^*(x) = \arg\min_{u\in\mathcal{U}} q^*(x,u). \tag{2.14}$$

2.3 Linear Programming

Bellman equation for a value function (2.9) can be relaxed to an inequality, namely

$$V(x) \le \mathcal{T}V(x) \quad ,\forall x \in \mathcal{X}. \tag{2.15}$$

It can be proven that Bellman operator $\mathcal{T}$ satisfies properties of monotonicity and value iteration convergence ([9]), and thus any V satisfying (2.15) will be a pointwise lower bound to optimal V^*:

$$V(x) \le \mathcal{T}V(x) \implies V(x) \le V^*(x) = \mathcal{T}V^*(x). \tag{2.16}$$

Based on this, a search for the V^* can be viewed as a search for the largest V that satisfies the constraint

$$V(x) \le \mathcal{T}V(x) \iff V(x) \le \min_{u\in\mathcal{U}} \left\{ l(x,u) + \gamma \mathop{\mathbb{E}}_{\xi}\left\{V^*(x^+_u)\right\} \right\}. \tag{2.17}$$

Accordingly, a constrained optimization problem is formulated as

$$\begin{gathered} \max_{V\in\mathcal{S}(\mathcal{X})} \int_{\mathcal{X}} V(x)c(x)dx \\ \text{s.t. } V(x) \le \mathcal{T}V(x), \quad \forall x \in \mathcal{X} \\ \text{where } \mathcal{T}V(x) = \min_{u\in\mathcal{U}} \left\{ l(x,u) + \gamma \mathop{\mathbb{E}}_{\xi}\left\{V(x^+_u)\right\} \right\}, \end{gathered} \tag{2.18}$$

and a solution to this problem is a solution to a Bellman equation (2.9). Decision variable V is optimized over a function space $\mathcal{S}(\mathcal{X})$ of real-valued functions. $c(x)$ is a finite measure on $\mathcal{X}$ that assigns positive mass to all open subsets of $\mathcal{X}$, often referred to as relevance weighting. Notice that above problem is not a linear program, as $\mathcal{T}$ is a nonlinear operator. This issue is resolved by dropping the minimum in the constraints which yields an infinite dimensional linear program of form

$$\begin{gathered} \max_{V\in\mathcal{S}(\mathcal{X})} \int_{\mathcal{X}} V(x)c(x)dx \\ \text{s.t. } V(x) \le \mathcal{T}_L V(x), \quad \forall x \in \mathcal{X} \\ \text{where } \mathcal{T}_L V(x) = l(x,u) + \gamma \mathop{\mathbb{E}}_{\xi}\left\{V(x^+_u)\right\}. \end{gathered} \tag{2.19}$$

Similar procedure can be repeated for a q-function. Bellman equation (2.12) is relaxed to an inequality

$$q(x,u) \le \mathcal{F}q(x,u) = l(x,u) + \gamma \mathop{\mathbb{E}}_{\xi}\{\min_{\omega\in\mathcal{U}} q(x^+_u,\omega)\}, \quad \forall (x,u) \in \mathcal{X}\times\mathcal{U}, \tag{2.20}$$

and used to form constraints for a constrained optimization problem

$$\begin{aligned}&\max_{q\in\mathcal{S}(\mathcal{X}\times\mathcal{U})}\int_{\mathcal{X}\times\mathcal{U}} q(x,u)c(x,u)dxdu\\ &\text{s.t. } q(x,u)\leq \mathcal{F}q(x,u), \quad \forall (x,u)\in\mathcal{X}\times\mathcal{U}.\end{aligned} \tag{2.21}$$

A solution to this problem is a solution to a Bellman equation (2.12). Similarly to value functions, decision variable q is optimized over a function space $\mathcal{S}(\mathcal{X}\times\mathcal{U})$. $c(x,u)$ is a finite measure on $\mathcal{X}\times\mathcal{U}$ that assigns positive mass to all open subsets of $\mathcal{X}\times\mathcal{U}$. Here, however, nonlinear constraints cannot be simply reformulated as linear ones by omitting the minimum. This is due to the order of expectation and minimum operators. One way to deal with the nonlinearity in constraints is to reformulate (2.21) by introducing additional decision variables and constraints [5]. An alternative method to tackle this is introducing a relaxed Bellman operator and using it to build relaxed linear programs.

2.3.1 The Relaxed Linear Program

As previously mentioned, the Bellman operator for q-function $\mathcal{F}$ is nonlinear; hence, the previously defined constrained optimization problem has nonlinear constraints. In attempt to resolve this issue, the authors in [4] introduced *the relaxed Bellman operator* as a map $\hat{\mathcal{F}}:(\mathcal{X}\times\mathcal{U})\rightarrow(\mathcal{X}\times\mathcal{U})$

$$\hat{\mathcal{F}}q(x,u)=l(x,u)+\gamma\min_{\omega\in\mathcal{U}}\big\{\mathop{\mathbb{E}}_{\xi}\{q(x_u^+,\omega)\}\big\}. \tag{2.22}$$

In this operator, the positions of expectation and minimum are exchanged, yet the fundamental properties of the Bellman operator still hold ([4], proposition 1). A solution to the relaxed Bellman equation (2.19) $\hat{q}$ is a pointwise upper bound to the solution of the regular Bellman equation (2.19) q^* ([4], proposition 2). Namely,

$$q^*(x,u)\leq\hat{q}(x,u), \quad \forall(x,u)\in\mathcal{X}\times\mathcal{U}. \tag{2.23}$$

When it comes to the optimal policy, q^* in (2.14) is replaced by $\hat{q}$, and an obtained

$$\hat{\pi}(x)=\arg\min_{u\in\mathcal{U}}\hat{q}(x,u). \tag{2.24}$$

is referred to as a *greedy policy.* Using this new, relaxed operator, a new version of an optimization problem can be defined by replacing the $\mathcal{F}$ in (2.18) with $\hat{\mathcal{F}}$. Nonlinear constraints are linearized by dropping the minimum in the relaxed Bellman inequality. Obtained relaxed linear program (RLP) is

$$\begin{aligned}&\max_{q\in\mathcal{S}(\mathcal{X}\times\mathcal{U})}\int_{\mathcal{X}\times\mathcal{U}} q(x,u)c(x,u)dxdu\\ &\text{s.t. } q(x,u)\leq\hat{\mathcal{F}}_Lq(x,u,\omega), \quad \forall(x,u,\omega)\in\mathcal{X}\times\mathcal{U}\times\mathcal{U}\\ &\text{where } \hat{\mathcal{F}}_Lq(x,u,\omega)=l(x,u)+\gamma\mathop{\mathbb{E}}_{\xi}\{q(x_u^+,\omega)\},\end{aligned} \tag{2.25}$$

and the optimal solution it gives is an upper bound to the optimal q-function q^* obtained from (2.21). A sequence of greedy policies calculated in each iteration converges to the optimal policy. Clearly, this method is used for stochastic dynamical systems since deterministic systems do not have an expectation operator in the Bellman equation and, therefore, the linearization of constraints is obtained by just dropping the minimum.

2.4 Policy Iteration

In this section, the policy iteration algorithm which uses an iterative procedure to search for the optimal stationary policy is described. It generates a sequence of stationary policies where each has improved cost over the preceding one. PI for a q-function can be summarized as follows:

- *Initialization*: Initialize the policy to any proper policy $\mu^0 \in \mathcal{U}$. A stationary policy μ is said to be proper if, when using this policy, there is a positive probability that the optimal solution will be reached regardless of the initial state.

- *Policy evaluation*: Given a policy μ^k in the current step k, compute the corresponding q-function by solving the system of equations

$$q_{\mu^k}(x,u) = l(x,u) + \gamma \underset{\xi}{\mathbb{E}}\{q_{\mu^k}(x_u^+, \mu^k(x_u^+))\}, \quad \forall(x,u) \in \mathcal{X} \times \mathcal{U}. \tag{2.26}$$

- *Policy improvement*: Obtain a new policy μ^{k+1} as

$$\mu^{k+1}(x) = \arg\min_{u\in\mathcal{U}} q_{\mu^k}(x,u), \quad \forall x \in \mathcal{X}. \tag{2.27}$$

  Iterate between policy evaluation and improvement steps until $q_{\mu^{k+1}}(x,u) = q_{\mu^k}(x,u)$, $\forall(x,u) \in \mathcal{X} \times \mathcal{U}$.

The fundamental property of the PI algorithm is that the sequence of generated policies is non-increasing and converges to the optimal one ([7], [9]). A sequence of policies $\{\mu^k\}$ is improving in the sense that $q_{\mu^{k+1}} \leq q_{\mu^k}, \forall k$.

Looking at the policy evaluation step and a linear program defined in the last section, one can notice that q_{μ^k} can be obtained as a solution to the following relaxed linear program:

$$\begin{aligned} &\max_{q\in\mathcal{S}(\mathcal{X}\times\mathcal{U})} \int_{\mathcal{X}\times\mathcal{U}} q(x,u)c(x,u)dxdu \\ &\text{s.t. } q(x,u) \leq l(x,u) + \gamma \underset{\xi}{\mathbb{E}}\, q(x_u^+, \mu^k(x_u^+)), \quad \forall(x,u) \in \mathcal{X} \times \mathcal{U}. \end{aligned} \tag{2.28}$$

The Bellman equation used above is the one associated with a greedy policy. The policy evaluation step can be reformulated and written based on a Bellman equation for a stationary policy. In that case, the equation does not contain a minimum operator. Therefore, the relaxation of the Bellman equation is not needed, and a standard linear program could be used in a policy evaluation step.

2.5 Curse of Dimensionality

Linear programs defined in section 2.3 and later used in the previous section are infinite dimensional problems. A few difficulties make computing an optimal q-function via an LP and retrieving an optimal policy via (2.14) computationally intractable. These difficulties relate to the curse of dimensionality and are summarized as:

(1) Optimization is performed over infinite dimensional space $\mathcal{S}(\mathcal{X} \times \mathcal{U})$.

(2) (Relaxed) linear program has an infinite number of constraints.

(3) Objective in a (relaxed) linear program involves a multidimensional integral.

(4) Policy (2.14) may be intractable since q^* can be any element of an infinite dimensional space $\mathcal{S}(\mathcal{X} \times \mathcal{U})$.

These difficulties also hold for an LP with a value function. That is why the direct (relaxed) LP (equations 2.19 and 2.24), as well as the policy evaluation step of PI, may be impossible to implement in the exact formulation. This may happen even when the state space is finite, but the number of states is significant. As a consequence, the original optimization problem needs to be approximated by a tractable counterpart. One is thus led to consider ADP methods.

2.6 Approximate Dynamic Programming

In this chapter, the methods for solving challenging and computationally intensive problems are considered. The discussion is based on working with q-functions and the RLP, but similar approaches hold for value functions and other methods.

The search for q^* over the infinite dimensional vector space of all real-valued measurable functions is narrowed down to tackle the first difficulty. The q-function is restricted in the span of a finite family of basis functions $q_i \in \bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U})$ and the search is restricted to the function space $\bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U}) := \{ \sum_{i=1}^{N} \alpha_i q_i \mid \alpha_i \in \mathbb{R} \}$, $i = 0, ..., N$ ([5], [4]). An approximate solution of the relaxed linear program is obtained by

$$\begin{aligned} &\max_{q \in \bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U})} \int_{\mathcal{X} \times \mathcal{U}} q(x,u) c(x,u) dx du \\ &\text{s.t. } \; q(x,u) \leq \hat{\mathcal{F}}_L q(x,u,\omega), \quad \forall (x,u,\omega) \in \mathcal{X} \times \mathcal{U} \times \mathcal{U}, \end{aligned} \tag{2.29}$$

where the only difference is that $\bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U})$ replaces $\mathcal{S}(\mathcal{X} \times \mathcal{U})$. Optimization is now performed over the basis functions weights α_i. The solution of (2.29) is $\bar{q}$, and generally, it will not solve the Bellman equation. The quality of a *greedy* policy $\bar{\pi}$ with respect to $\bar{q}$ depends on the distance between optimal q^* and $\bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U})$. Ideally of course, $q^* \in \bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U})$. In that case, $\bar{q} = q^*$, $\bar{\pi} = \pi^*$ and the choice of relevance weights $c(x,u)$ would not affect the calculation of the q^*. However, when $q^* \notin \bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U})$, one can influence the quality of an approximation by choosing appropriate $c(x,u)$.

The next issue that is addressed is the infinite number of constraints that occur in an RLP. The constraints can be relaxed via sampling. Namely, state and input variables x_i and u_i can be measured when the system interacts with the environment, and the next state x_i^+ and the cost l_i that arise can be observed. The procedure is repeated M times. These together create a sequence of M data tuples $\{(x_i, u_i, x_i^+, l_i)\}_{i=1}^{M}$. The infinite set of constraints from (2.28) is replaced by a set of sampled constraints

$$q(x_i, u_i) \leq l_i(x_i, u_i) + \gamma \mathop{\mathbb{E}}_{\xi} \{q(x_i^+, \omega_i)\}, \; i = \{1, 2, ..., M\}. \tag{2.30}$$

The third source of intractability can be overcome if $c(x,u)$ is defined as a probability density function with mean μ_c and variance Σ_c [5]. Using the definition of moments for the probability density function, the multidimensional integral can be rewritten as

$$\int_{\mathcal{X} \times \mathcal{U}} q(x,u) c(x,u) dx du = \mathop{\mathbb{E}}_{c} q. \tag{2.31}$$

The conditions for which (2.14) has a finite solution and the greedy policy is measurable are found in the [14]. One can conclude that the careful choice of $\bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U})$ helps with overcoming intractability sources of an RLP. Possible choices depend on the class of stage cost and system dynamics.

2.7 Data-Driven Policy Iteration

This section combines the policy iteration algorithm from section 2.4 with an approximate relaxed LP portrayed in the previous section. Two versions of a data-driven PI algorithm are introduced. The policy is evaluated by solving a sampled version of an RLP.

2.7.1 On-policy PI

A fundamental property which makes the PI algorithm effective is that the sequence of q-functions generated in each iteration converges to the optimal one. In the data-driven approach, this property is lost since the function space where we search for an optimal q-function is restricted, as well as because the constraints are sampled.

To overcome this issue, the authors in [7] suggest keeping all the constraints from previous iterations when solving a sequence of LPs. Using this idea, the number of constraints would grow significantly in each iteration. Alternatively, only the binding constraints from the previous iteration can be saved and added to the list of constraints in the current iteration. To distinct which constraints are binding, a dual linear program is solved, and every constraint with a non-zero Lagrange multiplier is considered to be binding.

Furthermore, authors in [7] propose that in each iteration the policy is evaluated using an approximate LP. For a stochastic system, this corresponds to (2.29). It should be noted that, unlike in the exact RLP, any $\omega \in \mathcal{U}$ can be picked when forming the constraints. This allows exploring different control strategies, as well as incorporating prospective knowledge about the system. It is also possible to create multiple constraints using a fixed data tuple and just changing ω values. This, together with the fact that only the binding constraints from previous iterations are kept, is exactly what makes this algorithm memory-efficient.

On-policy PI can be summarized as follows:

- *Initialization*:
  - Initialize the policy to any proper policy $\mu^0 \in \mathcal{U}$.
  - Initialize the number of data tuples M and the number of constraints P formed for each tuple.
  - Initialize the set of binding constraints $\mathcal{B}$ to an empty set.
- *Policy evaluation*:
  - Create data tuples $\{(x_i, u_i, x_i^+, l_i)\}_{i=1}^M$ by observing the system.
  - Create constraints by picking or sampling (possibly multiple) values of ω for each data tuple.

– Solve an approximate relaxed linear program

$$\begin{aligned}\max_{q\in\bar{\mathcal{S}}(\mathcal{X}\times\mathcal{U})} &\int_{\mathcal{X}\times\mathcal{U}} q(x,u)c(x,u)dxdu \\ \text{s.t. } q(x_i,u_i) &\leq l_i(x_i,u_i) + \gamma \mathop{\mathbb{E}}_{\xi}\{q(x_i^+,\omega_i)\},\ \forall i\in\{1,...,M\cdot P\} \\ q(x_j,u_j) &\leq l_j(x_j,u_j) + \gamma \mathop{\mathbb{E}}_{\xi}\{q(x_j^+,\omega_j)\},\ \forall j\in\mathcal{B}.\end{aligned} \tag{2.32}$$

♦ *Policy improvement*:

– Obtain a new policy μ^{k+1} as

$$\mu^{k+1}(x) = \arg\min_{u\in\mathcal{U}} q_k(x,u), \quad \forall x\in\mathcal{X}. \tag{2.33}$$

– Update the set of binding constraints $\mathcal{B}$.

Iterate between policy evaluation and improvement steps until some threshold between $q_{k+1}(x,u)$ and $q_k(x,u)$, $\forall(x,u)\in\mathcal{X}\times\mathcal{U}$ is reached.

2.7.2 Off-policy PI

This variant of a PI algorithm was proposed in [8]. Once again, the PI method is reformulated as a data-driven, finite-dimensional relaxed linear program. Before the algorithm starts iterating between policy evaluation and improvement steps, a buffer that contains data tuples is generated by letting the system interact with the environment and collecting the measured data. Unlike the on-policy version, which generates new data in every iteration, this version uses the same set of data tuples in all iterations. The off-policy setting inherits the convergence guarantees of the standard PI.

Off-policy PI can be summarized as follows:

♦ *Initialization*:

– Initialize the policy to any proper policy $\mu^0\in\mathcal{U}$.

– Construct a buffer which contains M data tuples, namely $\{(x_i,u_i,x_i^+,l_i)\}_{i=1}^M$.

♦ *Policy evaluation*: Solve an approximate relaxed linear program

$$\begin{aligned}\max_{q\in\bar{\mathcal{S}}(\mathcal{X}\times\mathcal{U})} &\int_{\mathcal{X}\times\mathcal{U}} q(x,u)c(x,u)dxdu \\ \text{s.t. } q(x_i,u_i) &\leq l_i(x_i,u_i) + \gamma \mathop{\mathbb{E}}_{\xi}\{q(x_i^+,\mu^k(x_i^+))\},\ \forall i\in\{1,...,M\}.\end{aligned} \tag{2.34}$$

♦ *Policy improvement*: Obtain a new policy μ^{k+1} as

$$\mu^{k+1}(x) = \arg\min_{u\in\mathcal{U}} q_{\mu^k}(x,u), \quad \forall x\in\mathcal{X}. \tag{2.35}$$

Iterate between policy evaluation and improvement steps until some threshold between $q_{\mu^{k+1}}(x,u)$ and $q_{\mu^k}(x,u)$, $\forall(x,u)\in\mathcal{X}\times\mathcal{U}$ is reached.

Chapter 3

Fixed Point Analysis for Affine Systems

In this chapter, we study stochastic discrete-time time-invariant systems. In particular, we focus on a system whose dynamics is defined by an affine function, namely

$$f(x_k, u_k, \xi_k) = Ax_k + Bu_k + b + \xi_k, \tag{3.1}$$

where $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, $\xi_k \in \mathcal{E} \subseteq \mathbb{R}^{n_x}$, $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_u \times n_x}$ and $b \in \mathbb{R}^{n_x}$. The associated cost has an extended quadratic form:

$$\begin{aligned} l(x,u) &= \begin{bmatrix} x \\ u \end{bmatrix}^\top L \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^\top L_L + l_{11} \\ &= \begin{bmatrix} x \\ u \end{bmatrix}^\top \begin{bmatrix} L_{xx} & L_{xu} \\ L_{xu}^\top & L_{uu} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + 2 \begin{bmatrix} x \\ u \end{bmatrix}^\top \begin{bmatrix} l_x \\ l_u \end{bmatrix} + l_{11}. \end{aligned} \tag{3.2}$$

L_{xx} is a positive semidefinite symmetric $n_x \times n_x$ matrix, and L_{uu} is a positive definite symmetric $n_u \times n_u$ matrix. Dimensions of L_{xu}, l_x, l_u and l_{11} are $n_x \times n_u$, $n_x \times 1$, $n_u \times 1$ and 1×1, respectively. The system differs from the one used in a standard LQR setup by a constant term b, constant term in the cost l_{11}, additional elements in the cost that depend on $x \cdot u$, as well as ξ_k an independent and identically distributed (i.i.d.) random variable with a non-zero mean μ_ξ.

In the following subsections, we present an extended version of an LQR, that results in the optimal policy for an above mentioned affine system, general quadratic cost function, and a non-zero mean noise. Then we present two variations of a q-function optimization problem, the first variation which uses the standard Bellman operator, and the second which uses the relaxed one. We show that all three setups yield the same optimal policy.

3.1 Generalized LQR

Solving optimal control problems is generally complex, yet there are a few special cases where analytic solutions can be found. One example is the widely studied linear quadratic regulator (LQR), which consists of linear dynamics and quadratic cost. It is well-known that the resulting optimal policy in this case stabilizes the system to the origin. We show that the method can be extended to a wider class of problems.

Definition 1. ***A generalized LQR*** *is a controller that provides an optimal control policy π^* that minimizes*

$$V(\pi) = \mathop{\mathbb{E}}_{\xi}\left\{\sum_{k=0}^{\infty}\gamma^k l(x_k, \pi(x_k))\right\},$$

for the affine system evolving as

$$x_{k+1} = Ax_k + Bu_k + b + \xi_k,$$

with a non-zero mean noise ξ_k, and an extended quadratic stage cost

$$l(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}^\top L \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^\top L_L + l_{11}.$$

Proposition 1. *Analytic solution of a generalized LQR is given by*

$$\mathbf{V^*(x) = x^\top P x + 2x^\top P_L + e},$$

where

$$P = Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{xu}^\top$$
$$P_L = \left(I - \gamma A^\top + \gamma Q_{xu}Q_{uu}^{-1}B^\top\right)^{-1}\left(l_x + \gamma A^\top P\hat{b} - Q_{xu}Q_{uu}^{-1}(\gamma B^\top P\hat{b} + l_u)\right)$$
$$e = \frac{l_{11} + \gamma\hat{b}^\top P\hat{b} + \gamma 2P_L^\top\hat{b} - (\gamma B^\top P\hat{b} + \gamma B^\top P_L + l_u)^\top Q_{uu}^{-1}(\gamma B^\top P\hat{b} + \gamma B^\top P_L + l_u + \gamma tr(P\Sigma))}{1-\gamma}$$
$$Q_{xx} = L_{xx} + \gamma A^\top PA, \quad Q_{xu} = L_{xu} + \gamma A^\top PB, \quad Q_{uu} = L_{uu} + \gamma B^\top PB$$
$$\hat{b} = b + \mu_\xi$$
$$q_u = l_u + \gamma B^\top P\hat{b} + \gamma B^\top P_L.$$

Optimal control policy is

$$\boldsymbol{\pi}^*\mathbf{(x) = -Q_{uu}^{-1}\,Q_{xu}^\top\,x - Q_{uu}^{-1}\,q_u}.$$

Proof. In order to be able to use already established derivations for a discounted infinite-horizon LQR, one should aim to reformulate the affine system (3.1) as a linear one. This can be done through augmentation of the state variable x_k as

$$\tilde{\mathrm{x}}_k = \begin{bmatrix} x_k \\ 1 \end{bmatrix}. \tag{3.3}$$

Using this, the system dynamics can be rewritten as

$$\begin{bmatrix} x_{k+1} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} A & b+\mu_\xi \\ 0 & 1 \end{bmatrix}}_{\tilde{\mathrm{A}}}\begin{bmatrix} x_k \\ 1 \end{bmatrix} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\tilde{\mathrm{B}}}u + \underbrace{\begin{bmatrix} \xi - \mu_\xi \\ 0 \end{bmatrix}}_{\tilde{\xi}}$$
$$\Updownarrow \tag{3.4}$$
$$\tilde{\mathrm{x}}_{k+1} = \tilde{\mathrm{A}}\,\tilde{\mathrm{x}}_k + \tilde{\mathrm{B}}\,u + \tilde{\xi}_k\,.$$

Note that the noise $\tilde{\xi}_k$ has mean $\mu_{\tilde{\xi}} = 0$ and variance $\tilde{\Sigma} = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}$. The cost for the augmented state becomes

$$
\begin{aligned}
l(\widetilde{\mathrm{x}}, u) &= \begin{bmatrix} x \\ 1 \\ u \end{bmatrix}^\top \begin{bmatrix} L_{xx} & l_x & L_{xu} \\ l_x^\top & l_{11} & l_u^\top \\ L_{xu}^\top & l_u & L_{uu} \end{bmatrix} \begin{bmatrix} x \\ 1 \\ u \end{bmatrix} \\
&= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}, \\
\widetilde{L}_{xx} &= \begin{bmatrix} L_{xx} & l_x \\ l_x^\top & l_{11} \end{bmatrix}, \quad \widetilde{L}_{xu} = \begin{bmatrix} L_{xu} \\ l_u^\top \end{bmatrix}, \quad \widetilde{L}_{uu} = L_{uu}.
\end{aligned}
\tag{3.5}
$$

Now well-known derivations of an LQR can be used on the augmented system since it is linear and the cost is pure quadratic. When searching for an optimal value function for an affine system, the function space is restricted to a subspace of extended quadratic functions, i.e., we search for an optimal value function of form

$$
V^*(x) = x^\top P x + 2x^\top P_L + e. \tag{3.6}
$$

If the system was linear, the assumed form of an optimal value function would be a sum of quadratic term and constant. We start from stipulating that $V^*(\widetilde{\mathrm{x}}) = \widetilde{\mathrm{x}}\,\widetilde{\mathrm{P}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{e}}$, $\widetilde{\mathrm{P}} = \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix}$, and evaluating the Bellman equation (2.9):

$$
\begin{aligned}
V(\widetilde{\mathrm{x}}) &= \min_{u \in \mathcal{U}} \left\{ l(\widetilde{\mathrm{x}}, u) + \gamma \mathop{\mathbb{E}}_{\widetilde{\xi}} \left\{ V(\widetilde{\mathrm{x}}_u^+) \right\} \right\} \\
&= \min_{u \in \mathcal{U}} \left\{ l(\widetilde{\mathrm{x}}, u) + \gamma \mathop{\mathbb{E}}_{\widetilde{\xi}} \left\{ V(\widetilde{\mathrm{A}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\,u + \widetilde{\xi}) \right\} \right\} \\
&= \min_{u \in \mathcal{U}} \left\{ \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma \mathop{\mathbb{E}}_{\widetilde{\xi}} \left\{ (\widetilde{\mathrm{A}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\,u + \widetilde{\xi})^\top \widetilde{\mathrm{P}} (\widetilde{\mathrm{A}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\,u + \widetilde{\xi}) + \widetilde{\mathrm{e}} \right\} \right\}.
\end{aligned}
\tag{3.7}
$$

Using that

$$
\begin{aligned}
\mathop{\mathbb{E}}_{\widetilde{\xi}}(\widetilde{\xi}^\top \widetilde{\mathrm{P}}\,\widetilde{\xi}) &= \mathop{\mathbb{E}}_{\widetilde{\xi}}(tr(\widetilde{\xi}^\top \widetilde{\mathrm{P}}\,\widetilde{\xi})) = \mathop{\mathbb{E}}_{\widetilde{\xi}}(tr(\widetilde{\mathrm{P}}\,\widetilde{\xi}\,\widetilde{\xi}^\top)) = tr(\mathop{\mathbb{E}}_{\widetilde{\xi}}(\widetilde{\mathrm{P}}\,\widetilde{\xi}\,\widetilde{\xi}^\top)) = tr(\widetilde{\mathrm{P}} \cdot \mathop{\mathbb{E}}_{\widetilde{\xi}}(\widetilde{\xi}\,\widetilde{\xi}^\top)) \\
&= tr(\widetilde{\mathrm{P}}\,\widetilde{\Sigma}) \\
&= tr\left(\begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \right) \\
&= tr(P\Sigma),
\end{aligned}
\tag{3.8}
$$

and the fact that $\mu_{\widetilde{\xi}} = 0$, we calculate the expectation

$$
V(\widetilde{\mathrm{x}}) = \min_{u \in \mathcal{U}} \Big\{ \underbrace{\widetilde{\mathrm{x}}^\top \widetilde{L}_{xx}\,\widetilde{\mathrm{x}} + u^\top \widetilde{L}_{uu} u + 2\,\widetilde{\mathrm{x}}^\top \widetilde{L}_{xu} u + \gamma(\widetilde{\mathrm{A}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\,u)^\top \widetilde{\mathrm{P}} (\widetilde{\mathrm{A}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\,u) + \gamma tr(P\Sigma) + \gamma\,\widetilde{\mathrm{e}}}_{\star} \Big\}.
\tag{3.9}
$$

To obtain u which minimizes (3.9), compute the derivative with respect to u and set it to zero:

$$
\begin{aligned}
&\frac{d(\star)}{du} = 2\widetilde{L}_{uu} u + 2\widetilde{L}_{xu}^\top \widetilde{\mathrm{x}} + 2\gamma\,\widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\,\widetilde{\mathrm{A}}\,\widetilde{\mathrm{x}} + 2\gamma\,\widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\,\widetilde{\mathrm{B}}\,u := 0 \\
&\qquad\qquad\Downarrow \\
&u^* = -(\widetilde{L}_{uu} + \gamma\,\widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\,\widetilde{\mathrm{B}})^{-1} \, (\gamma\,\widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\,\widetilde{\mathrm{A}} + \widetilde{L}_{xu}^\top) \; \widetilde{\mathrm{x}}\,.
\end{aligned}
\tag{3.10}
$$

Plugging u^* back into (3.9) and rewriting the equation in a shorter way by introducing additional terms, yields

$$\begin{aligned} V(\widetilde{\mathrm{x}}) &= \widetilde{\mathrm{x}}^\top(\widetilde{\mathrm{Q}}_{xx} - \widetilde{\mathrm{Q}}_{xu}\,\widetilde{\mathrm{Q}}_{uu}^{-1}\,\widetilde{\mathrm{Q}}_{xu}^\top)\,\widetilde{\mathrm{x}} + \gamma tr(P\Sigma) + \gamma\,\widetilde{\mathrm{e}} \\ \widetilde{\mathrm{Q}}_{xx} &= \widetilde{L}_{xx} + \gamma\,\widetilde{\mathrm{A}}^\top\,\widetilde{\mathrm{P}}\,\widetilde{\mathrm{A}} \\ \widetilde{\mathrm{Q}}_{xu} &= \widetilde{L}_{xu} + \gamma\,\widetilde{\mathrm{A}}^\top\,\widetilde{\mathrm{P}}\,\widetilde{\mathrm{B}} \\ \widetilde{\mathrm{Q}}_{uu} &= \widetilde{L}_{uu} + \gamma\,\widetilde{\mathrm{B}}^\top\,\widetilde{\mathrm{P}}\,\widetilde{\mathrm{B}}\,. \end{aligned} \tag{3.11}$$

Looking back at the assumed form of the optimal value function, we can write

$$\widetilde{\mathrm{x}}^\top\,\widetilde{\mathrm{P}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{e}} = \widetilde{\mathrm{x}}^\top(\widetilde{\mathrm{Q}}_{xx} - \widetilde{\mathrm{Q}}_{xu}\,\widetilde{\mathrm{Q}}_{uu}^{-1}\,\widetilde{\mathrm{Q}}_{xu}^\top)\,\widetilde{\mathrm{x}} + \gamma tr(P\Sigma) + \gamma\,\widetilde{\mathrm{e}}, \tag{3.12}$$

and from here infer that $\widetilde{\mathrm{P}}$ and $\widetilde{\mathrm{e}}$ are solutions to

$$\begin{aligned} \widetilde{\mathrm{P}} &= \widetilde{\mathrm{Q}}_{xx} - \widetilde{\mathrm{Q}}_{xu}\,\widetilde{\mathrm{Q}}_{uu}^{-1}\,\widetilde{\mathrm{Q}}_{xu}^\top \\ \widetilde{\mathrm{e}} &= \frac{\gamma tr(P\Sigma)}{1-\gamma}. \end{aligned} \tag{3.13}$$

Replacing augmented variables with the original ones yields

$$\begin{aligned} \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} &= \begin{bmatrix} L_{xx} & l_x \\ l_x^\top & l_{11} \end{bmatrix} + \gamma \begin{bmatrix} A & b+\mu_\xi \\ 0 & 1 \end{bmatrix}^\top \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} A & b+\mu_\xi \\ 0 & 1 \end{bmatrix} - \\ &- \left(\gamma \begin{bmatrix} A & b+\mu_\xi \\ 0 & 1 \end{bmatrix}^\top \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} + \begin{bmatrix} L_{xu} \\ l_u^\top \end{bmatrix}\right) \left(L_{uu} + \gamma \begin{bmatrix} B^\top & 0 \end{bmatrix} \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix}\right)^{-1} \\ &\cdot \left(\gamma \begin{bmatrix} B^\top & 0 \end{bmatrix} \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} A & b+\mu_\xi \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} L_{xu}^\top & l_u \end{bmatrix}\right). \end{aligned} \tag{3.14}$$

Multiplying and summing all matrices in the above equation gives solutions to P, P_L and P_C. P is a solution to an algebraic Riccati equation (ARE). It is known that if $(\sqrt{\gamma}A, \sqrt{\gamma}B)$ is a stabilizable pair, then the solution to discrete ARE is a unique and positive definite matrix of dimensions $n_x \times n_x$. P_L and P_C have closed-form solutions. Additional terms are introduced in order to shorten the equations:

$$\begin{aligned} P &= Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{xu}^\top \\ P_L &= \left(I - \gamma A^\top + \gamma Q_{xu}Q_{uu}^{-1}B^\top\right)^{-1}\left(l_x + \gamma A^\top P\hat{b} - Q_{xu}Q_{uu}^{-1}(\gamma B^\top P\hat{b} + l_u)\right) \\ P_C &= \frac{l_{11} + \gamma\hat{b}^\top P\hat{b} + \gamma 2P_L^\top\hat{b} - (\gamma B^\top P\hat{b} + \gamma B^\top P_L + l_u)^\top Q_{uu}^{-1}(\gamma B^\top P\hat{b} + \gamma B^\top P_L + l_u)}{1-\gamma} \\ Q_{xx} &= L_{xx} + \gamma A^\top PA \\ Q_{xu} &= L_{xu} + \gamma A^\top PB \\ Q_{uu} &= L_{uu} + \gamma B^\top PB \\ \hat{b} &= b + \mu_\xi. \end{aligned} \tag{3.15}$$

To obtain the complete optimal value function, we go back to the term from which we started the derivation, and replace tilde with the original variables:

$$\begin{aligned} V(\widetilde{\mathrm{x}}) &= \widetilde{\mathrm{x}}^\top\,\widetilde{\mathrm{P}}\,\widetilde{\mathrm{x}} + \widetilde{\mathrm{e}} \\ V(x) &= \begin{bmatrix} x \\ 1 \end{bmatrix}^\top \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} + \frac{\gamma tr(P\Sigma)}{1-\gamma} \\ V(x) &= x^\top Px + 2x^\top P_L + P_C + \frac{\gamma tr(P\Sigma)}{1-\gamma}. \end{aligned} \tag{3.16}$$

In conclusion, analytic solution of a generalized LQR is given by

$$V^*(x) = x^\top P x + 2x^\top P_L + e, \tag{3.17}$$

where $e = P_C + \frac{\gamma tr(P\Sigma)}{1-\gamma}$ and P, P_L and P_C are obtained from (3.15). Optimal control policy is obtained by replacing augmented variables in equation (3.10):

$$\begin{aligned} \pi^* &= -(L_{uu} + \gamma B^\top P B)^{-1} \left(\gamma B^\top P A x + L_{xu}^\top x + \gamma B^\top P \hat{b} + \gamma B^\top P_L + l_u\right) \\ &= -Q_{uu}^{-1}\, Q_{xu}^\top\, x - Q_{uu}^{-1}\, q_u. \end{aligned} \tag{3.18}$$

We emphasize here that the optimal control policy also contains an affine term. ■

Note that the same solution to this LQR problem is obtained through a 'pure derivation' method, where the solution is not calculated using state augmentation. Instead, we work from the beginning with an affine system and assume an optimal value function (3.6). This method is straightforward but quite tedious. 'Pure derivation' was used here as a verification of the presented calculation.

3.2 q-function Optimization Problem

Definition 2. *A solution to the* ***q-function optimization problem based on the Bellman operator*** *is an optimal control policy π^* associated to the optimal q-function*

$$q^*(x,u) = l(x,u) + \gamma \mathop{\mathbb{E}}_{\xi} \left\{ \min_{\omega \in \mathcal{U}} q(x^+_{\pi^*}, \omega) \right\},$$

defined for the affine system evolving as

$$x_{k+1} = Ax_k + Bu_k + b + \xi_k,$$

with a non-zero mean noise ξ_k, and an extended quadratic stage cost

$$l(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}^\top L \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^\top L_L + l_{11}.$$

If we search for the optimal policy $\hat{\pi}$ associated to the

$$\hat{q}(x,u) = l(x,u) + \gamma \min_{\omega \in \mathcal{U}} \left\{ \mathop{\mathbb{E}}_{\xi} q(x^+_{\hat{\pi}}, \omega) \right\},$$

where essentially minimum and expectation operators exchange the place, we talk about the ***q-function optimization problem based on the relaxed Bellman operator****.*

The derivation of an optimal q-function is similar to the one of a value function, but it allows the policy extraction without knowing the explicit model of the system and the cost structure. q-function was introduced in section 2.2, particularly in the equation (2.12). Here we note that x_u^+ and $l(x,u)$ can be generated from the pair (x,u) by simulation. Therefore, the q-function optimization problem can be viewed as a combination of an LQR and simulation ([9]).

Standard optimization problem exploits standard Bellman operator for obtaining optimal q-function and policy. However, building a linear program whose solutions correspond to the analytic ones is not so straightforward, as the Bellman operator for q-functions is not linear.

That is why, apart from utilizing the standard Bellman operator, we here provide a solution to the optimization problem based on the relaxed Bellman operator.

3.2.1 Optimization Problem Based on the Bellman Operator

Proposition 2. *Analytic solution of a q-function optimization problem based on the Bellman operator is given by*

$$\mathbf{q}^*(\mathbf{x},\mathbf{u}) = \begin{bmatrix}\boldsymbol{x}\\\boldsymbol{u}\end{bmatrix}^\top \mathbf{Q} \begin{bmatrix}\boldsymbol{x}\\\boldsymbol{u}\end{bmatrix} + \begin{bmatrix}\boldsymbol{x}\\\boldsymbol{u}\end{bmatrix}^\top \mathbf{Q_L} + \mathbf{e},$$

where

$$Q = \begin{bmatrix}Q_{xx} & Q_{xu}\\ Q_{xu}^\top & Q_{uu}\end{bmatrix}, \quad Q_L = 2\begin{bmatrix}q_x\\ q_u\end{bmatrix}$$
$$Q_{xx} = L_{xx} + \gamma A^\top P A, \quad Q_{xu} = L_{xu} + \gamma A^\top P B, \quad Q_{uu} = L_{uu} + \gamma B^\top P B$$
$$q_x = l_x + \gamma A^\top P\hat{b} + \gamma A^\top P_L$$
$$q_u = l_u + \gamma B^\top P\hat{b} + \gamma B^\top P_L$$
$$e = \gamma \hat{b}^\top P\hat{b} + 2\gamma P_L^\top \hat{b} + \gamma P_C$$
$$P = Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{xu}^\top$$
$$P_L = \left(I - \gamma A^\top + \gamma Q_{xu}Q_{uu}^{-1}B^\top\right)^{-1}\left(l_x + \gamma A^\top P\hat{b} - Q_{xu}Q_{uu}^{-1}(\gamma B^\top P\hat{b} + l_u)\right)$$
$$P_C = \frac{l_{11} + \gamma\hat{b}^\top P\hat{b} + \gamma 2P_L^\top\hat{b} - (\gamma B^\top P\hat{b} + \gamma B^\top P_L + l_u)^\top Q_{uu}^{-1}(\gamma B^\top P\hat{b} + \gamma B^\top P_L + l_u)}{1-\gamma}$$
$$\hat{b} = b + \mu_\xi.$$

Optimal control policy is the same as the one for the generalized LQR:

$$\boldsymbol{\pi}^*(\mathbf{x}) = -\mathbf{Q_{uu}^{-1}}\,\mathbf{Q_{xu}^\top}\,\mathbf{x} - \mathbf{Q_{uu}^{-1}}\,\mathbf{q_u}.$$

Proof. Firstly, the function space is restricted to a subspace of extended quadratic functions, i.e. we search for the q-function of the form

$$\begin{aligned} q^*(x,u) &= \begin{bmatrix}x\\u\end{bmatrix}^\top Q \begin{bmatrix}x\\u\end{bmatrix} + \begin{bmatrix}x\\u\end{bmatrix}^\top Q_L + e \\ &= \begin{bmatrix}x\\u\end{bmatrix}^\top \begin{bmatrix}Q_{xx} & Q_{xu}\\ Q_{xu}^\top & Q_{uu}\end{bmatrix}\begin{bmatrix}x\\u\end{bmatrix} + 2\begin{bmatrix}x\\u\end{bmatrix}^\top\begin{bmatrix}q_x\\q_u\end{bmatrix} + e. \end{aligned} \tag{3.19}$$

The system dynamics is augmented (3.4) and presented in a linear form. A similar operation is done for the cost (3.5). The optimal q-function for an augmented system is assumed to be

$$\begin{aligned} q^*(\tilde{\mathrm{x}},u) &= \begin{bmatrix}\tilde{\mathrm{x}}\\u\end{bmatrix}^\top \tilde{\mathrm{Q}}\begin{bmatrix}\tilde{\mathrm{x}}\\u\end{bmatrix} + \tilde{\mathrm{e}}, \\ \tilde{\mathrm{Q}} &= \left[\begin{array}{cc:c} Q_{xx} & q_x & Q_{xu}\\ q_x^\top & q_{11} & q_u^\top \\ \hdashline Q_{xu}^\top & q_u & Q_{uu}\end{array}\right] = \begin{bmatrix}\tilde{\mathrm{Q}}_{xx} & \tilde{\mathrm{Q}}_{xu}\\ \tilde{\mathrm{Q}}_{xu}^\top & \tilde{\mathrm{Q}}_{uu}\end{bmatrix}. \end{aligned} \tag{3.20}$$

The Bellman equation (2.12) is evaluated:

$$\begin{aligned} q(\widetilde{\mathrm{x}}, u) &= l(\widetilde{\mathrm{x}}, u) + \gamma \mathop{\mathbb{E}}_{\widetilde{\xi}} \Big\{ \min_{\omega \in \mathcal{U}} \{ q(\widetilde{\mathrm{x}}_u^+, \omega)\} \Big\} \\ &= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma \mathop{\mathbb{E}}_{\widetilde{\xi}} \Big\{ \mathop{min}_{\omega \in \mathcal{U}} \Big\{ \begin{bmatrix} \widetilde{\mathrm{x}}^+ \\ \omega \end{bmatrix}^\top \widetilde{\mathrm{Q}} \begin{bmatrix} \widetilde{\mathrm{x}}^+ \\ \omega \end{bmatrix} + \widetilde{\mathrm{e}} \Big\} \Big\} = \\ &= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma \mathop{\mathbb{E}}_{\widetilde{\xi}} \Big\{ \mathop{min}_{\omega \in \mathcal{U}} \{ (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u + \widetilde{\xi})^\top \widetilde{\mathrm{Q}}_{xx} (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u + \widetilde{\xi}) + \\ &\quad + 2(\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u + \widetilde{\xi})^\top \widetilde{\mathrm{Q}}_{xu}\, \omega + \omega^\top \widetilde{\mathrm{Q}}_{uu}\, \omega + \widetilde{\mathrm{e}} \} \Big\}. \end{aligned} \tag{3.21}$$

To get ω which minimizes above equation, calculate the derivative and set it to zero. Obtained ω is

$$\omega^* = -\widetilde{\mathrm{Q}}_{uu}^{-1} \widetilde{\mathrm{Q}}_{xu}^\top (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u + \widetilde{\xi}). \tag{3.22}$$

Plugging ω^* back into (3.21) gives

$$\begin{aligned} q(\widetilde{\mathrm{x}}, u) &= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma \mathop{\mathbb{E}}_{\widetilde{\xi}} \Big\{ (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u + \widetilde{\xi})^\top \widetilde{\mathrm{Q}}_{xx} (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u + \widetilde{\xi}) + \\ &\quad + 2(\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u + \widetilde{\xi})^\top \widetilde{\mathrm{Q}}_{xu}\, \omega^* + \omega^{*\top} \widetilde{\mathrm{Q}}_{uu}\, \omega^* + \widetilde{\mathrm{e}} \Big\} = \\ &= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \left(\begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} + \gamma \begin{bmatrix} \widetilde{\mathrm{A}}^\top \\ \widetilde{\mathrm{B}}^\top \end{bmatrix} \widetilde{\mathrm{P}} \begin{bmatrix} \widetilde{\mathrm{A}} & \widetilde{\mathrm{B}} \end{bmatrix} \right) \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma tr(\widetilde{\mathrm{P}}\, \widetilde{\Sigma}) + \gamma\, \widetilde{\mathrm{e}} = \\ &= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{\mathrm{L}}_{xx} + \gamma\, \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{xu} + \gamma\, \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{B}} \\ \widetilde{\mathrm{L}}_{xu}^\top + \gamma\, \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{uu} + \gamma\, \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{B}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma tr(\widetilde{\mathrm{P}}\, \widetilde{\Sigma}) + \gamma\, \widetilde{\mathrm{e}}, \end{aligned} \tag{3.23}$$

where

$$\widetilde{\mathrm{P}} = \widetilde{\mathrm{Q}}_{xx} - \widetilde{\mathrm{Q}}_{xu}\, \widetilde{\mathrm{Q}}_{uu}^{-1}\, \widetilde{\mathrm{Q}}_{xu}^\top. \tag{3.24}$$

Comparing the last equality of (3.23) and assumed form of an optimal q-function (3.20), one can conclude that

$$\begin{aligned} \widetilde{\mathrm{Q}} = \begin{bmatrix} \widetilde{\mathrm{Q}}_{xx} & \widetilde{\mathrm{Q}}_{xu} \\ \widetilde{\mathrm{Q}}_{xu}^\top & \widetilde{\mathrm{Q}}_{uu} \end{bmatrix} &= \begin{bmatrix} \widetilde{\mathrm{L}}_{xx} + \gamma\, \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{xu} + \gamma\, \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{B}} \\ \widetilde{\mathrm{L}}_{xu}^\top + \gamma\, \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{uu} + \gamma\, \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}}\, \widetilde{\mathrm{B}} \end{bmatrix} \\ \widetilde{\mathrm{e}} &= \frac{\gamma tr(\widetilde{\mathrm{P}}\, \widetilde{\Sigma})}{1 - \gamma}. \end{aligned} \tag{3.25}$$

We assume $\widetilde{\mathrm{P}}$ has the form $\begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix}$. Tilde values in $\widetilde{\mathrm{e}}$ can be replaced by original ones following (3.8). Looking at form of $\widetilde{\mathrm{Q}}$ from (3.20), comparing it to (3.25) and then plugging in the original variables instead of the augmented ones, we get

$$\widetilde{\mathrm{Q}} = \begin{bmatrix} \widetilde{\mathrm{L}}_{xx} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{xu} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}} \\ \widetilde{\mathrm{L}}_{xu}^\top + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{uu} + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}} \end{bmatrix} =$$

$$= \left[\begin{array}{c|c|c} L_{xx} + \gamma A^\top P A & l_x + \gamma A^\top P b + \gamma A^\top P_L & L_{xu} + \gamma A^\top P B \\ \hline l_x^\top + \gamma b^\top P A + \gamma P_L^\top A & \gamma b^\top P b + 2\gamma P_L^\top b + \gamma P_C & l_u^\top + \gamma b^\top P B + \gamma P_L^\top B \\ \hline L_{xu}^\top + \gamma B^\top P A & l_u + \gamma B^\top P b + \gamma B^\top P_L & L_{uu} + \gamma B^\top P B \end{array}\right] = \tag{3.26}$$

$$= \left[\begin{array}{c|c|c} Q_{xx} & q_x & Q_{xu} \\ \hline q_x^\top & q_{11} & q_u^\top \\ \hline Q_{xu}^\top & q_u & Q_{uu} \end{array}\right].$$

To derive the optimal q-function, we need to firstly derive P, P_L and P_C. Plugging in values from (3.25) into equation (3.24) we obtain

$$\widetilde{\mathrm{P}} = \widetilde{\mathrm{L}}_{xx} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}} - (\widetilde{\mathrm{L}}_{xu} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}}) \, (\widetilde{\mathrm{L}}_{uu} + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}})^{-1} \, (\widetilde{\mathrm{L}}_{xu}^\top + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}}). \tag{3.27}$$

the augmented system is now replaced by the original one, hence (3.27) becomes

$$\begin{aligned} \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} &= \begin{bmatrix} L_{xx} & l_x \\ l_x^\top & l_{11} \end{bmatrix} + \gamma \begin{bmatrix} A & b + \mu_\xi \\ 0 & 1 \end{bmatrix}^\top \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} A & b + \mu_\xi \\ 0 & 1 \end{bmatrix} - \\ &- \left(\begin{bmatrix} L_{xu} \\ l_u^\top \end{bmatrix} + \gamma \begin{bmatrix} A & b + \mu_\xi \\ 0 & 1 \end{bmatrix}^\top \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} \right) \left(L_{uu} + \gamma \begin{bmatrix} B \\ 0 \end{bmatrix}^\top \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} \right)^{-1} \\ &\cdot \left(\begin{bmatrix} L_{xu} \\ l_u^\top \end{bmatrix} + \gamma \begin{bmatrix} A & b + \mu_\xi \\ 0 & 1 \end{bmatrix}^\top \begin{bmatrix} P & P_L \\ P_L^\top & P_C \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} \right)^\top . \end{aligned} \tag{3.28}$$

Multiplying and summing all matrices in the above equation gives solutions to P, P_L and P_C. P is a solution to an algebraic Riccati equation, whereas P_L and P_C have closed-form solutions:

$$\begin{aligned} P &= Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{xu}^\top \\ P_L &= \left(I - \gamma A^\top + \gamma Q_{xu} Q_{uu}^{-1} B^\top\right)^{-1} \left(l_x + \gamma A^\top P \hat{b} - Q_{xu} Q_{uu}^{-1} (\gamma B^\top P \hat{b} + l_u)\right) \\ P_C &= \frac{l_{11} + \gamma \hat{b}^\top P \hat{b} + \gamma 2 P_L^\top \hat{b} - (\gamma B^\top P \hat{b} + \gamma B^\top P_L + l_u)^\top Q_{uu}^{-1} (\gamma B^\top P \hat{b} + \gamma B^\top P_L + l_u)}{1 - \gamma} \\ \hat{b} &= b + \mu_\xi . \end{aligned} \tag{3.29}$$

Note that Q_{xx}, Q_{xu} and Q_{uu} are matrices described in (3.26). Now we can put everything together:

$$\begin{aligned} q(\widetilde{\mathrm{x}}, u) &= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \widetilde{\mathrm{Q}} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \widetilde{\mathrm{e}} \\ q(x, u) &= \begin{bmatrix} \widetilde{\mathrm{x}} \\ 1 \\ u \end{bmatrix}^\top \begin{bmatrix} Q_{xx} & q_x & Q_{xu} \\ q_x^\top & q_{11} & q_u^\top \\ Q_{xu}^\top & q_u & Q_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ 1 \\ u \end{bmatrix} + \frac{\gamma tr(P\Sigma)}{1 - \gamma} \\ q(x, u) &= \begin{bmatrix} x \\ u \end{bmatrix}^\top \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{xu}^\top & Q_{uu} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + 2 \begin{bmatrix} x \\ u \end{bmatrix}^\top \begin{bmatrix} q_x \\ q_u \end{bmatrix} + q_{11} + \frac{\gamma tr(P\Sigma)}{1 - \gamma} . \end{aligned} \tag{3.30}$$

To sum up, analytic solution of a q-function optimization problem is given by

$$
\begin{aligned}
q^*(x,u) &= \begin{bmatrix} x \\ u \end{bmatrix}^\top Q \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^\top Q_L + e \\
Q = \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{xu}^\top & Q_{uu} \end{bmatrix}&, \quad Q_L = 2\begin{bmatrix} q_x \\ q_u \end{bmatrix}, \quad e = q_{11} + \frac{\gamma tr(P\Sigma)}{1-\gamma} \\
Q_{xx} &= L_{xx} + \gamma A^\top P A \\
Q_{xu} &= L_{xu} + \gamma A^\top P B \\
Q_{uu} &= L_{uu} + \gamma B^\top P B \\
q_x &= l_x + \gamma A^\top P\hat{b} + \gamma A^\top P_L \\
q_u &= l_u + \gamma B^\top P\hat{b} + \gamma B^\top P_L \\
q_{11} &= \gamma \hat{b}^\top P\hat{b} + 2\gamma P_L^\top \hat{b} + \gamma P_C,
\end{aligned}
\tag{3.31}
$$

where $\hat{b}$, P, P_L and P_C are obtained from (3.29). Optimal control policy is obtained as

$$
\begin{aligned}
\pi^*(x) &= \underset{u\in U}{argmin}\, q^*(x,u) = \\
&= -Q_{uu}^{-1}Q_{xu}^\top x - Q_{uu}^{-1}q_u.
\end{aligned}
\tag{3.32}
$$

■

3.2.2 Optimization Problem Based on the Relaxed Bellman Operator

Proposition 3. *Analytic solution of q-function optimization problem based on the relaxed Bellman operator ($\hat{\mathcal{F}}$) is is*

$$\hat{\mathbf{q}}(\mathbf{x},\mathbf{u}) = \mathbf{q}^*(\mathbf{x},\mathbf{u}) + \boldsymbol{\Delta}\mathbf{e},$$

where $q^*(x,u)$ is the optimal q-function for the optimization problem based on the Bellman operator, and

$$\Delta e = \frac{\gamma tr(Q_{xu}Q_{uu}^{-1}Q_{xu}^\top \Sigma)}{1-\gamma}.$$

Optimal control policy is the same as optimal policies for the generalized LQR and the q-function optimization problem based on the Bellman operator:

$$\hat{\boldsymbol{\pi}} = -\mathbf{Q}_{\mathbf{uu}}^{-1}\,\mathbf{Q}_{\mathbf{xu}}^\top\,\mathbf{x} - \mathbf{Q}_{\mathbf{uu}}^{-1}\,\mathbf{q}_{\mathbf{u}}.$$

Proof. A very similar derivation of optimal q-function and optimal policy follows if, instead of the regular Bellman operator, the relaxed one is utilized. Namely, we start with assuming the form of optimal q-function to be (3.19). Then we perform state augmentation and obtain system and cost in new variables, (3.4) and (3.5), respectively. Here the relaxed Bellman equation is utilized:

$$
\begin{aligned}
q(\widetilde{\mathrm{x}}, u) &= l(\widetilde{\mathrm{x}}, u) + \gamma \min_{\omega \in \mathcal{U}} \left\{ \mathop{\mathbb{E}}_{\xi}\{q(\widetilde{\mathrm{x}}_u^+, \omega)\} \right\} = \\
&= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma \min_{\omega \in \mathcal{U}} \left\{ \mathop{\mathbb{E}}_{\widetilde{\xi}} \left\{ \begin{bmatrix} \widetilde{\mathrm{x}}^+ \\ \omega \end{bmatrix}^\top \widetilde{\mathrm{Q}} \begin{bmatrix} \widetilde{\mathrm{x}}^+ \\ \omega \end{bmatrix} + \widetilde{\mathrm{e}} \right\} \right\} = \\
&= \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{L}_{xx} & \widetilde{L}_{xu} \\ \widetilde{L}_{xu}^\top & \widetilde{L}_{uu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma \min_{\omega \in \mathcal{U}} \Big\{ (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u)^\top \widetilde{\mathrm{Q}}_{xx} (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u) + \\
&\quad + 2(\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u)^\top \widetilde{\mathrm{Q}}_{xu}\, \omega + \omega^\top \widetilde{\mathrm{Q}}_{uu}\, \omega + \gamma tr(Q_{xx}\Sigma) + \widetilde{\mathrm{e}} \Big\}.
\end{aligned}
\tag{3.33}
$$

To get ω which minimizes above equation, calculate the derivative and set it to zero. Obtained ω is

$$
\hat{\omega} = -\widetilde{\mathrm{Q}}_{uu}^{-1} \widetilde{\mathrm{Q}}_{xu}^\top (\widetilde{\mathrm{A}}\, \widetilde{\mathrm{x}} + \widetilde{\mathrm{B}}\, u). \tag{3.34}
$$

Plugging ω^* back into (3.33) yields

$$
q(\widetilde{\mathrm{x}}, u) = \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix}^\top \begin{bmatrix} \widetilde{\mathrm{L}}_{xx} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{xu} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}} \\ \widetilde{\mathrm{L}}_{xu}^\top + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{uu} + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathrm{x}} \\ u \end{bmatrix} + \gamma tr(\widetilde{\mathrm{Q}}_{xx} \widetilde{\Sigma}) + \gamma \widetilde{\mathrm{e}}. \tag{3.35}
$$

Evening out terms (3.19) and (3.35) gives

$$
\begin{aligned}
\widetilde{\mathrm{Q}} &= \begin{bmatrix} \widetilde{\mathrm{L}}_{xx} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{xu} + \gamma \widetilde{\mathrm{A}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}} \\ \widetilde{\mathrm{L}}_{xu}^\top + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{A}} & \widetilde{\mathrm{L}}_{uu} + \gamma \widetilde{\mathrm{B}}^\top \widetilde{\mathrm{P}} \widetilde{\mathrm{B}} \end{bmatrix} \\
\widetilde{\mathrm{e}} &= \frac{\gamma tr(Q_{xx}\Sigma)}{1-\gamma}.
\end{aligned}
\tag{3.36}
$$

The proof that we can replace tilde variables with the original ones in $\widetilde{\mathrm{e}}$ is analogues to (3.8). When it comes to P, the equation (3.24) holds and it can be transformed back to original variables following (3.28) and (3.29). Replacing tilde variables in equation (3.26) also holds.

In conclusion, the analytic solution of a relaxed q-function optimization problem is an optimal relaxed q-function $\hat{q}$:

$$
\begin{aligned}
\hat{q}(x, u) &= \begin{bmatrix} x \\ u \end{bmatrix}^\top Q \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^\top Q_L + e \\
Q = \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{xu}^\top & Q_{uu} \end{bmatrix}&, \quad Q_L = 2 \begin{bmatrix} q_x \\ q_u \end{bmatrix}, \quad e = q_{11} + \frac{\gamma tr(Q_{xx}\Sigma)}{1-\gamma} \\
Q_{xx} &= L_{xx} + \gamma A^\top P A \\
Q_{xu} &= L_{xu} + \gamma A^\top P B \\
Q_{uu} &= L_{uu} + \gamma B^\top P B \\
q_x &= l_x + \gamma A^\top P \hat{b} + \gamma A^\top P_L \\
q_u &= l_u + \gamma B^\top P \hat{b} + \gamma B^\top P_L \\
q_{11} &= \gamma \hat{b}^\top P \hat{b} + 2\gamma P_L^\top \hat{b} + \gamma P_C,
\end{aligned}
\tag{3.37}
$$

where $\hat{b}$, P, P_L and P_C are obtained from (3.29). Optimal control policy for a relaxed q-function is obtained as

$$
\begin{aligned}
\hat{\pi}(x) &= \underset{u \in U}{argmin}\, \hat{q}(x, u) = \\
&= -Q_{uu}^{-1} Q_{xu}^\top x - Q_{uu}^{-1} q_u.
\end{aligned}
\tag{3.38}
$$

One can note that the optimal q-function (3.31) and the optimal relaxed q-function (3.38) are quite similar. In fact, they differ only in the constant term. In the regular q-function, in the trace there is matrix P, whereas for the relaxed q-function there is Q_{xx} instead. The difference between $\hat{q}(x,u)$ and $q^*(x,u)$ can then be calculated as

$$\begin{aligned}
\hat{q}(x,u) - q^*(x,u) &= \frac{\gamma tr(Q_{xx}\Sigma)}{1-\gamma} - \frac{\gamma tr(P\Sigma)}{1-\gamma} = \\
&= \frac{\gamma tr(Q_{xx}\Sigma)}{1-\gamma} - \frac{\gamma tr((Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{xu}^\top)\Sigma)}{1-\gamma} = \\
&= \frac{\gamma tr(Q_{xu}Q_{uu}^{-1}Q_{xu}^\top\Sigma)}{1-\gamma} =: \Delta e.
\end{aligned} \tag{3.39}$$

■

Since $\Delta e \geq 0$, this shows that the optimal relaxed q-function $\hat{q}(x,u)$ is an upper bound for the optimal q-function $q^*(x,u)$, namely $q^* \leq \hat{q}$.

Comparing optimal policies obtained for an LQR (3.18), a q-function optimization problem (3.32), and a relaxed q-function optimization problem (3.38), one can conclude that they are all the same. This proves that it is possible to obtain an optimal control policy for an affine system without knowing the system dynamics and cost. Moreover, the Bellman operator can be relaxed so that it is easier to work with it.

In the next chapter, we provide a simulation study that confirms theoretical results from this chapter. Additionally, we conduct an empirical study that compares the performance of obtaining optimal q-function and control policy via various ADP methods.

Chapter 4

Simulation Study

In this section, we present a numerical study that highlights aspects of the formerly presented theory. First of all, we show that the solution of an approximate relaxed linear program built for an affine system converges to the analytic one. We compare the performance of this relaxed LP with a standard LP. Furthermore, we conduct an empirical study that compares iterative methods for solving this optimization problem. The advantages and disadvantages of various methods are stated. Finally, we discuss the difference between non-iterative and iterative methods of solving an RLP. The non-iterative method means that an RLP is solved in one try, whereas iterative methods denote various versions of a PI algorithm that exploits an RLP in a policy evaluation step.

4.1 Non-iterative Method

The study is performed on a system with affine dynamics and extended quadratic cost. Namely,

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} 1 & 0.1 \\ 0.5 & -0.5 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u_k + \begin{bmatrix} 0.5 \\ -0.3 \end{bmatrix} + \xi_k \\ l(x_k, u_k) &= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} 1 & 0 & 0.1 \\ 0 & 1 & 0.1 \\ 0.1 & 0.1 & 0.01 \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix} + 0.1, \end{aligned} \tag{4.1}$$

$x_k \in \mathcal{X} \subseteq \mathbb{R}^2$, $u_k \in \mathcal{U} \subseteq \mathbb{R}$. State and input variables are sampled from a uniform distribution, $x_k \in [-3, 3]^2$ and $u_k \in [-1, 1]$. Disturbance ξ_k is sampled from a normal distribution with the mean $\mu_\xi = 0.05$ and the variance $\Sigma_\xi = 10^{-5}$. The discount factor γ is 0.95.

In order to be able to create a tractable RLP, function space $\mathcal{S}(\mathcal{X} \times \mathcal{U})$ is restricted to a space of extended quadratic functions

$$\bar{\mathcal{S}}(\mathcal{X} \times \mathcal{U}) = \{q \in \mathcal{S}(\mathcal{X} \times \mathcal{U}) \mid q(x, u) = \begin{bmatrix} x \\ u \end{bmatrix}^\top Q \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} x \\ u \end{bmatrix}^\top Q_L + e\}, \tag{4.2}$$

where Q is a symmetric matrix of dimensions $(n_x + n_u) \times (n_x + n_u)$, and Q_L a matrix of dimensions $(n_x + n_u) \times 1$.

In some real-world applications, the data about the system would be collected as the system would interact with an environment. Here, the oracle is used as a simulator of a physical system where the next state x_u^+ and the cost l would be measured. The oracle takes a sequence of state and input pairs $\{(x_i, u_i)\}_{i=1}^M$, and creates a sequence of data tuples $\{(x_i, u_i, x_i^+, l_i)\}_{i=1}^M$. Because of the stochasticity of the system, for each pair (x_i, u_i) one hundred samples of x_i^+ are used in order to obtain a Monte Carlo approximation of the expectation ([4]). The data tuples are used to form a set of constraints for an RLP.

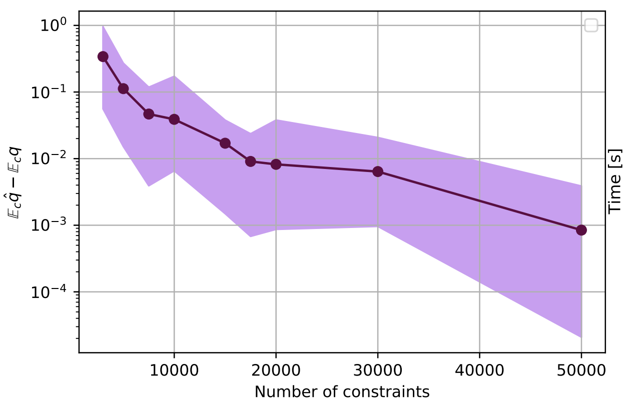
The objective of an RLP is reformulated using the definition of moments for a probability density function $c(x, u)$, as was already mentioned before. Since we know that the optimal relaxed q-function will belong to the restricted functional space, the choice of relevance weights c is not crucial when we consider the objective function. It does play a role, however, since there is another approximation related to the constraints. We used here $\mu_c = 0$, $\Sigma_c = I$. Using that $z = \begin{bmatrix} x \\ u \end{bmatrix}$, the integral in the objective is reformulated as

$$\begin{aligned}
\int\limits_{\mathcal{X}\times\mathcal{U}} q(x,u)\ c(x,u)\ dx\ du &= \int\limits_{\mathcal{X}\times\mathcal{U}} (z^\top Q z + z^\top Q_L + e)\ c(z)\ dz = \\
&= \int\limits_{\mathcal{X}\times\mathcal{U}} z^\top Q z\ c(z)\ dz + Q_L^\top \int\limits_{\mathcal{X}\times\mathcal{U}} z\ c(z)\ dz + e \int\limits_{\mathcal{X}\times\mathcal{U}} c(z)\ dz = \\
&= \int\limits_{\mathcal{X}\times\mathcal{U}} \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} z_j [Q]_{ij} z_i\ c(z)\ dz + Q_L^\top \mu_c + e = \\
&= \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} [Q]_{ij} \Sigma_{ji} + Q_L^\top \mu_c + e = \\
&= \sum_{i=1}^{n+m} [Q\Sigma]_{ii} + Q_L^\top \mu_c + e = \\
&= tr(Q\Sigma_c) + Q_L^\top \mu_c + e.
\end{aligned} \tag{4.3}$$

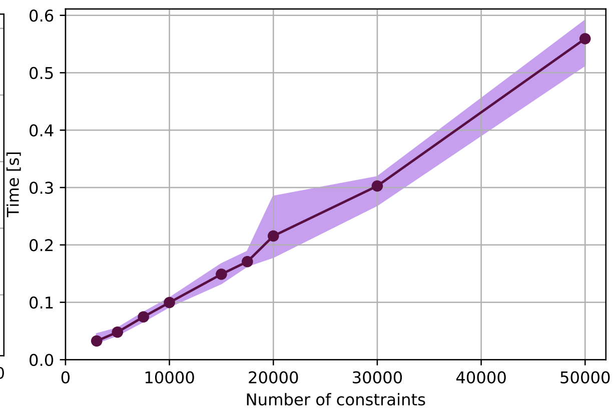
The RLP, whose objective is to maximize (4.3) and whose constraints are sampled as detailed above, is solved using a state-of-art *Gurobi* mathematical optimization solver. [1]

An analytic optimal relaxed q-function for an aforementioned affine system was derived following calculations from section 3.2.2. Figure 4.1a shows the difference between analytic objective function ($\mathbb{E}_c \hat{q}$) and the objective function derived by solving an approximate RLP ($\mathbb{E}_c q$), for a different number of constraints. Figure 4.1b shows solving time for a growing number of constraints. We show a realization of 10 identical runs. The range of results is presented with the shaded area, and the mean of results is presented with a solid line.

[1] Python code used for the experiments available at `https://gitlab.ethz.ch/mdraskovic/semester-project`



(a) Performance vs constraints



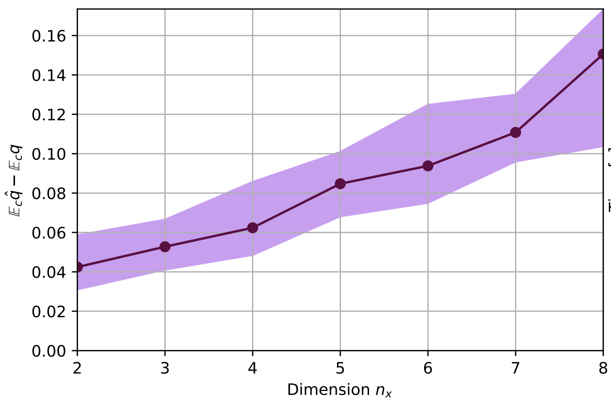
(b) Time vs constraints

Figure 4.1: Non-iterative RLP; Different number of constraints

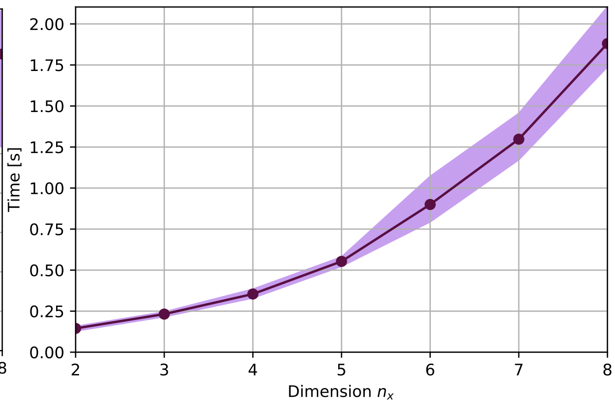
We also note here that the other standard way to deal with the nonlinearity in LP constraints is by introducing additional decision variables. The reader is referred to [5] for more details about this method. However, for the affine problem setup we are unable to provide a simulation result of this LP, as the solver always reported unboundedness.

In order to see how the approximate RLP would perform for the system with a larger number of states, we generate an affine system where $n_x \in \{2, 3, ..., 8\}$ and $n_u = 2$. Coefficients in the dynamics and in the cost are generated in an Erdős-Rényi fashion:

$$
\begin{gathered}
A_{ij}, B_{ij}, bij = \begin{cases} 0.5, & i = j \\ 0, & i \neq j, \text{with probability } 0.1 \\ \mathcal{U}([-0.1, 0.1]), & i \neq j, \text{with probability } 0.9 \end{cases} \\
L_{xx} = I_{n_x \times n_x},\ L_{uu} = I_{n_u \times n_u},\ L_{xu} = 0.001_{n_x \times n_u}, \\
l_x = 0.001_{n_x \times 1},\ l_u = 0.001_{n_u \times 1}, l_{11} = 0.001 \\
x_k \in \mathcal{U}([-0.5, 0.5])_x^n,\ u_k \in \mathcal{U}([-3, 3])^{n_u} \\
\mu_\xi = 0.005,\ \Sigma_\xi = 10^{-6}.
\end{gathered}
\tag{4.4}
$$



(a) Performance vs dimension



(b) Time vs dimension

Figure 4.2: Non-iterative RLP; Different dimension of state variable

For this experiment, we used 10^4 constraints. As one can expect, with a more significant number of state variables, the performance of an approximate RLP decreases, and the time needed to solve the problem increases. Still, even for the system with the dimension $n_x = 8$, the result seems pretty accurate.

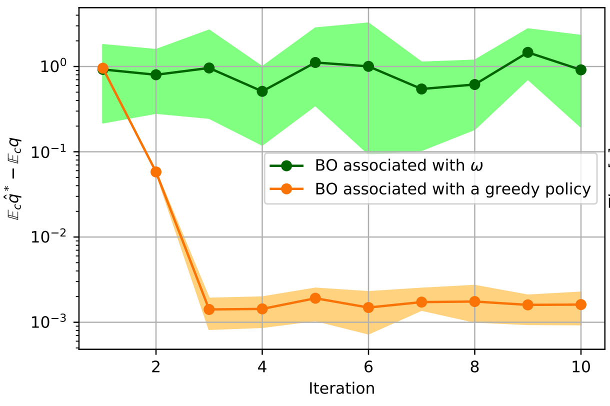
4.2 Iterative Method

In this section, we study performance of different variations of a policy iteration algorithm that exploits an RLP in the policy evaluation step. In the following experiments, we use the system (4.1). Generally, any PI algorithm aims to solve a Bellman equation in a policy evaluation step. For this, one can utilize an RLP, as was explained in (2.28). A greedy policy used in the constraints is calculated in a policy improvement step of the previous iteration.

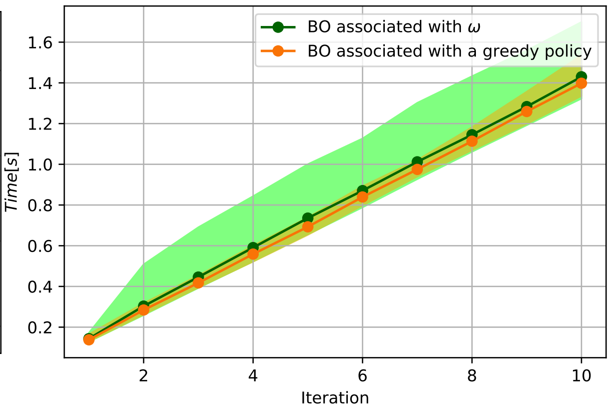
This can, however, be limiting in the model-free approach, as there is no freedom for the system to explore different control strategies. To solve this, one can use an RLP from equation (2.25) in a policy evaluation step. Instead of solving a Bellman equation associated with a greedy policy, the reformulated PI draws ω from distribution in each iteration. In the following text, we will call this reformulation *a PI with the Bellman operator associated with* ω. When this operator is used, however, the fundamental monotonicity property of the PI method is lost. To address this, binding constraints from the previous iteration can be added to the list of constraints in the current one, to ensure a monotonic behaviour ([7], Lemma 1).

4.2.1 Standard Policy Iteration

We start with a standard PI for an affine system. Figure 4.3 depicts the accuracy and time performance of this algorithm. The orange line represents an algorithm where in a policy evaluation step the Bellman operator associated with a greedy policy is exploited. The objective calculated with this algorithm is compared to the analytic optimal relaxed q-function associated with the relaxed Bellman operator. The green line represents an algorithm that works with the Bellman operator associated with ω. When this operator is utilized, there is no need for the relaxation of a Bellman operator in the analytic solution since the Bellman equation does not contain minimum. This means that while for the orange line we compare calculated objective with $\mathbb{E}_c\hat{q}$, for the green line we use $\mathbb{E}_c q^*$, and on the plot we depict these together as $\mathbb{E}_c\hat{q}^*$. $1.5 \cdot 10^3$ constraints are used in each iteration.



(a) Performance vs iterations



(b) Time vs iterations

Figure 4.3: Standard PI; Comparison between Bellman operators

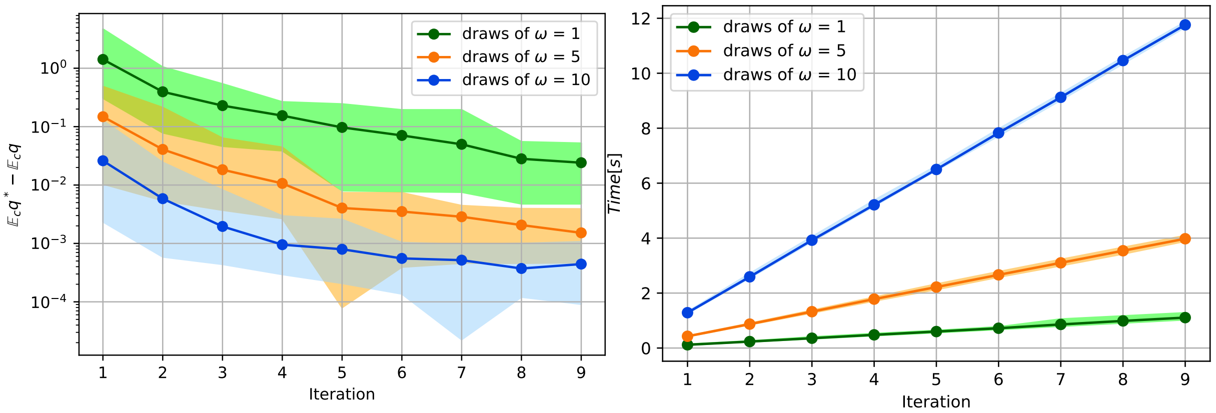
It can be seen that the data-driven standard PI does not hold monotonic behaviour property, as was explained in [7]. Objective calculated with a PI with the Bellman operator associated with a greedy policy converges to some solution and keeps oscillating there. The algorithm that exploits the Bellman operator associated with ω is not particularly useful. It does not preserve memory from the previous iterations, so the solution keeps oscillating around a bad performance. It can be essentially seen as solving an RLP in each iteration from the beginning without using any previous knowledge. Since the number of constraints used is not large enough, there is a significant offset from the analytic objective.

When comparing the time needed to solve an algorithm, it is similar regardless of the used Bellman operator. For the PI with the Bellman operator associated with a greedy policy, three iterations are needed to reach the difference of around 10^{-3} between the calculated and the analytic objective. For solving three iterations, PI needs around 0.3 seconds. Comparing this with the performance from figure 4.1, one can notice that the non-iterative method needs much more constraints to reach the same accuracy. Moreover, the time that the non-iterative method needs is around 0.45 seconds.

4.2.2 On-policy PI

At this point we turn our attention to the on-policy setting of a policy iteration algorithm. The setting was proposed in [7], and it exploits the Bellman operator associated with ω. Such operator allows for the incorporation of expert knowledge and exploration of different control strategies. To bring back the fundamental monotonic behaviour of a PI, authors in [7] propose the inclusion of the binding constraints from the previous iteration in the current one.

Additional possibility when using the Bellman operator associated with ω, is that for one data tuple sampled from the system, multiple ωs can be drawn from distribution. Using this, multiple constraints can be formed using each data tuple. So, it is possible to construct multiple times more constraints than the number of accessible data tuples, which makes this setting of the PI memory-efficient. Figure 4.4 illustrates the performance of the on-policy PI, when for each data tuple 1, 5 or 10 ωs are drawn from distribution.



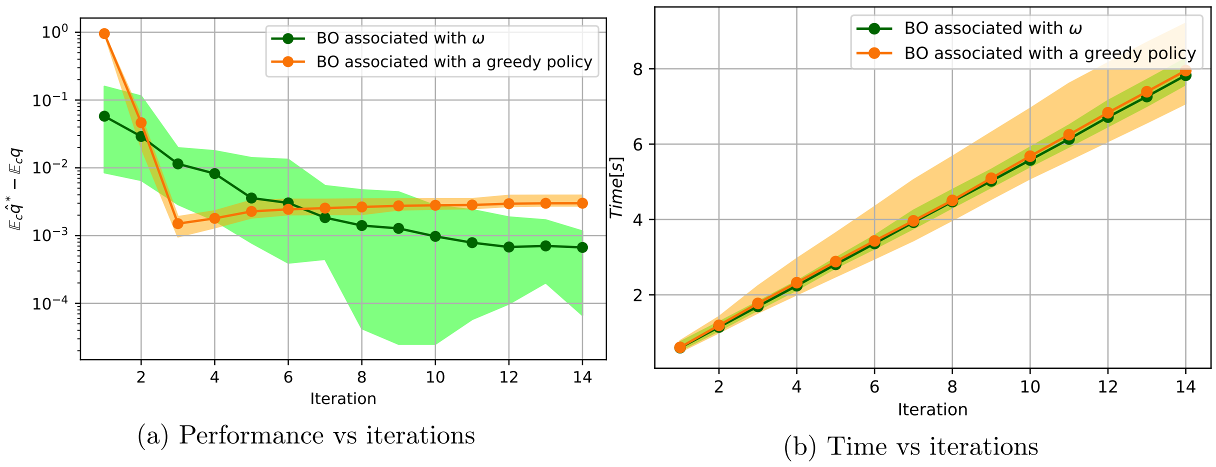
(a) Performance vs iterations

(b) Time vs iterations

Figure 4.4: On-policy PI; Comparison between number of ω draws

Slight increase in the difference between the objectives, that sometimes occurs between two iterations (for example between iterations eight and nine in the blue line), comes from a numerical error in the solver, and is not a conceptual problem. The more draws of ω we use, the more accurate the performance is since the RLP uses more constraints. The number of data tuples used in each iteration is $1.5 \cdot 10^3$. Thus, when the number of ω draws is five or ten, we have $7.5 \cdot 10^3$ and $1.5 \cdot 10^4$ constraints in each iteration, respectively. Of course, more ω samples mean more time needed to come to the solution. In all further experiments, five draws of ω for each data tuple are used.

Next, we compare on-policy PI with the Bellman operator associated with ω (green line), with an on-policy PI with the Bellman operator associated with a greedy policy (orange line). Note that the latter differs from a standard PI only because it incorporates binding constraints.



(a) Performance vs iterations

(b) Time vs iterations

Figure 4.5: On-policy PI; Comparison between Bellman operators

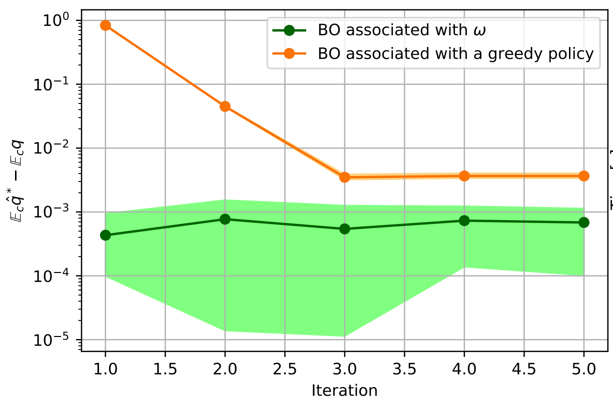
It can be seen that the version with the Bellman operator associated with a greedy policy converges to some optimal solution fast. Unlike in figure 4.3a, there are no oscillations around this solution because of the binding constraints.

On the other hand, we see that the algorithm with the Bellman operator associated with ω has monotonic behaviour, and the difference between the calculated and the analytic objective keeps decreasing with the number of iterations. It outperforms the standard PI in the accuracy, but the time it needs is more significant because it works with five times more constraints. We note here that the non-iterative RLP in figure 4.1a does not even reach the accuracy below 10^{-3} with $5 \cdot 10^4$ constraints, yet this accuracy is reached with only $1.5 \cdot 10^3$ constraints with an on-policy PI. The speed is an apparent downside of this algorithm.

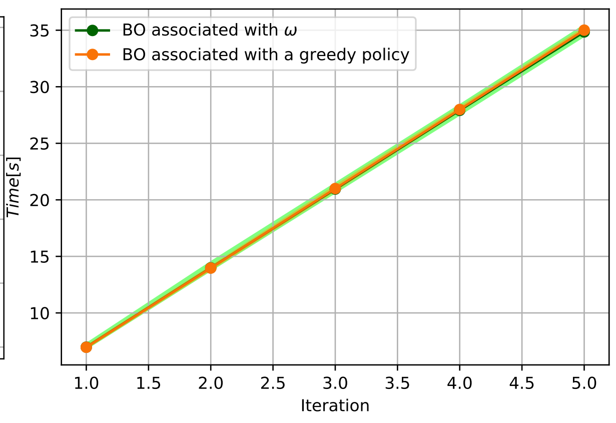
4.2.3 Off-policy PI

Further, we take a look at the off-policy PI algorithm introduced in [8]. The orange line in figure 4.6a depicts the suggested off-policy setting, where the Bellman operator is associated with a greedy policy. The predefined buffer of data tuples used throughout the algorithm should be as big as the physical setting or the computational power allows. We used a buffer of 10^6 data tuples. Therefore an RLP solved in each PI iteration has 10^6 constraints.

The green line depicts an off-policy setting with the Bellman operator associated with ω. This is not especially helpful, as the algorithm does not preserve any information from one iteration to the other. Essentially, an individual RLP is used in each iteration, and the solution oscillates around some optimum. It is very accurate because the number of constraints is huge.



(a) Performance vs iterations

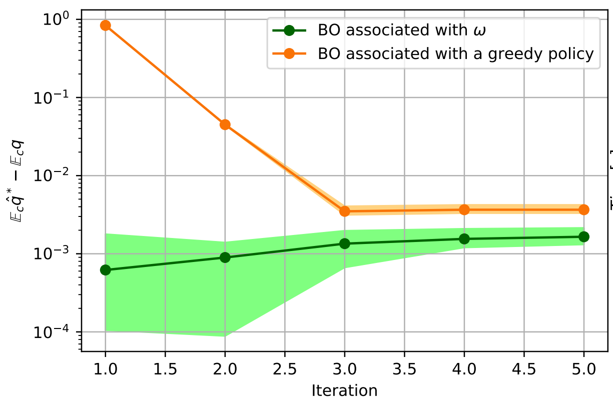


(b) Time vs iterations

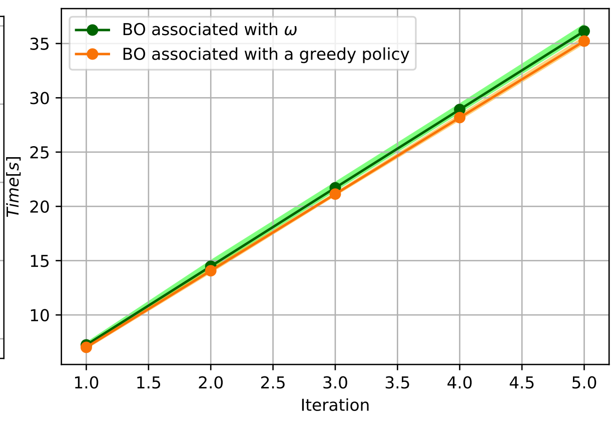
Figure 4.6: Off-policy PI; Comparison between Bellman operators

One can conclude that the off-policy PI with the Bellman operator associated with a greedy policy mimics a standard PI and is limited in the sense that the saturation point depends on the used data. The optimal solution for the predefined buffer is already reached after a few iterations. To obtain an objective that would keep converging to an analytic one, one should use even a larger buffer. This algorithm inherits the convergence guarantees of the standard PI, and can be useful in cases when we have access to the system only at the beginning of the experiment. The time needed is large as an RLP with a large number of constraints is solved in each iteration.

One last thing one can try is adding the binding constraints in an off-policy PI with the Bellman operator associated with ω. With this addition, the green line does not oscillate around an optimal solution, yet the algorithm is still not practically useful.



(a) Performance vs iterations



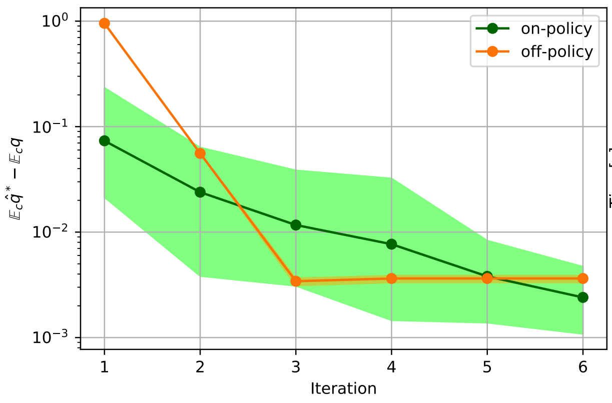
(b) Time vs iterations

Figure 4.7: Off-policy PI; Comparison between Bellman operators; Added binding constraints

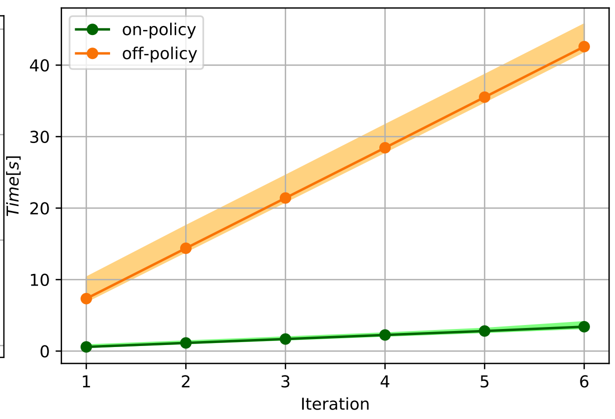
4.2.4 Methods Comparison

Lastly, we provide a comparison between above-mentioned methods. Figure 4.8 depicts the performance comparison of on-policy and off-policy PIs. This comparison is not entirely fair because the algorithms do not use the same amount of data, and therefore solve an RLP with different numbers of constraints. One of the most significant advantages of an on-policy setting is that it does not require much data. On the other hand, for an off-policy setting to make sense, the buffer size needs to be large.

The green line depicts on-policy setting PI with $1.5 \cdot 10^3$ data tuples and five draws of ω for each data tuple. Bellman operator is associated with ω. The orange line presents an off-policy setting algorithm, with a buffer of 10^6 data tuples and Bellman operator associated with a greedy policy.



(a) Performance vs iterations



(b) Time vs iterations

Figure 4.8: Comparison between on-policy and off-policy settings

Table 4.1 depicts comparison between non-iterative and iterative methods for solving an RLP. Values for PI algorithms are the ones obtained after six iterations.

Comparison between non-iterative and iterative methods			
Method	$\mathbb{E}_c\hat{q}^* - \mathbb{E}_c q$	**# data tuples**	**CPU time [s]** [2]
Non-iterative	$1.11 \cdot 10^{-3}$	$5 \cdot 10^4$	0.69
Iterative, standard PI	$1.42 \cdot 10^{-3}$	$1.5 \cdot 10^3$	0.64
Iterative, on-policy PI	$1.99 \cdot 10^{-3}$	$1.5 \cdot 10^3$	3.65
Iterative, off-policy PI	$3.70 \cdot 10^{-3}$	$1 \cdot 10^5$	45.6

Table 4.1

To sum up, a non-iterative method of an RLP converges to the analytic solution as we increase the number of constraints. It needs around $5 \cdot 10^4$ constraints to achieve acceptable performance, and the solving time is not extensive. On the other hand, standard PI loses monotonic properties that are fundamental for a PI. An on-policy setting solves this issue and provides a memory-efficient solution that allows for exploration of different control strategies. An off-policy setting needs a considerable buffer of data tuples but it comes in handy when one can collect data from the system only at the beginning of an experiment.

[2] Simulations are run on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz processor

Chapter 5

Conclusion

In this thesis, a stochastic time-invariant system with affine dynamics, extended quadratic stage cost, and non-zero mean disturbance is considered. For this system, we firstly provide an analytic solution to a generalized LQR problem. Namely, we relax the linear quadratic and zero-mean noise assumptions in the discounted infinite-horizon LQR problem for stochastic time-invariant systems. Then, a solution to a model-free q-function optimization problem is derived for when one intends to control the system without knowing its dynamical equations. The Bellman equation aimed to be solved in this optimization problem is then replaced by the relaxed Bellman equation, where essentially minimum and expectation operators switch place. A new optimization problem based on the relaxed Bellman operator is introduced and solved for an affine system. It is proven that all three mentioned optimization problems yield the same optimal policy.

Relaxing the Bellman operator enables one to build a relaxed linear program whose solution coincides with the solution of the relaxed Bellman equation. The relaxed linear program can be solved non-iteratively, in one shot, or by introducing iterative methods. Iterative methods are essentially policy iteration algorithms that exploit relaxed linear program formulation in the policy evaluation step. Apart from the standard PI, other possible data-driven PI settings are on-policy and off-policy. We carried out an empirical study that compares three PI algorithms between each other, and compares them to a non-iterative method of solving an RLP.

We note several interesting directions for future work. In this thesis, we exploit linear architectures to represent an approximate q-function. It may be interesting to explore algorithms that use nonlinear representations. Furthermore, performance bounds for the RLP can be derived. Finally, it would be interesting to see how proposed methods function on a physical system (or a model of a physical system), as here we only used a theoretical simulation.

Bibliography

[1] R.E. Bellman. *On the Theory of Dynamic Programming.* Proceedings of the National Academy of Sciences, vol. 38, no. 8, pp. 716-719. 1952.

[2] R. Kalman. *Contributions to the Theory of Optimal Control.* Boletin de la Sociedad Matematica Mexicana, vol. 5, pp. 102-119. 1960.

[3] D. Bertsekas. *Dynamic Programming and Optimal Control, volume 1.* Athena Scientific, 3rd ed. 2005.

[4] A. Martinelli, M. Gargiani, and J Lygeros. *Data-Driven Optimal Control with a Relaxed Linear Program.* arXiv:2003.08721. 2020.

[5] P.N. Beuchat, A. Georghiou, and J. Lygeros. *Performance Quarantees for Model-Based Approximate Dynamic Programming in Continuous Spaces.* IEEE Transaction on Automatic Control, vol. 65, no. 1, pp. 143-158. 2019.

[6] D.P. de Farias and B. van Roy. *The Linear Programming Approach to Approximate Dynamic Programming.* Operations Research, vol. 51, no. 6, pp. 850-865. 2003.

[7] G. Banjac and J.Lygeros. *A Data-Driven Policy Iteration Scheme based on Linear Programming.* In 58th IEEE Conference on Decision and Control, pp. 816-821. 2020.

[8] A. Tanzanakis and J.Lygeros. *Data-Driven Control of Unknown Systems: A Linear Programming Approach.* In IFAC-PapersOnLine, arXiv:2003.00779. 2020.

[9] D. Bertsekas. *Dynamic Programming and Optimal Control, volume 2.* Athena Scientific, 3rd ed. 2007.

[10] W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* John Wiley & Sons. 2011.

[11] R.S Sutton and A.G. Barto. *Reinforcement Learning: An Introduction, 2nd ed.* The MIT Press. 2020.

[12] S. Barratt and S. Boyd. *Stochastic Control with Affine Dynamics and Extended Quadratic Costs.* arXiv:1811.00168. 2018.

[13] C.J.C.H. Watkins and P.Dayan. *Q-learning.* Machine Learning, 8(3):279-292. 1992.

[14] O. Hernandez-Lerma and J.B. Lasserre. *Discrete-Time Markov Control Processes: Basic Optimality Criteria.* Springer-Verlag NY. 1996.