From Supervised to Reinforcement Learning: an Inverse Optimization Approach

Ioannis Dimanidis

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

From Supervised to Reinforcement Learning: an Inverse Optimization Approach

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Ioannis Dimanidis

December 10, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

European Research Council

Established by the European Commission

Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

From Supervised to Reinforcement Learning: an Inverse Optimization Approach

by

Ioannis Dimanidis

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: December 10, 2021

Supervisor(s):	dr. P. Mohajerin Esfahani
Reader(s):	dr.ir. M. Mazo Jr.
	dr. B. Atasoy

Abstract

We propose a novel method combining elements of supervised- and Q-learning for the control of dynamical systems subject to unknown disturbances. By using the Inverse Optimization framework and in-hindsight information we can derive a causal parametric optimization policy that approximates a non-causal MPC expert. Furthermore, we propose a new min-max MPC scheme that robustifies against a ball around a disturbance trajectory. This scheme yields an exact convex reformulation using the S-Lemma, and is also approximated using Inverse Optimization. Finally, simulation studies clarify and verify our approach.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my supervisor dr. Peyman Mohajerin Esfahani for his guidance and support during the writing of this thesis. His insights and comments were invaluable.

Furthermore, I would like to thank the Bodossaki Foundation which, through its scholarship program, supported me during my studies here in Delft.

Finally, I would like to thank my friends and family, who stood by my side and were always there for me.

Delft, University of Technology
December 10, 2021

Ioannis Dimanidis

Chapter 1

Introduction

Reinforcement Learning has been growing in popularity immensely in recent years due to promising advances in the field, such as mastering the game of Go [28], or playing Atari games at or even beyond human level [23]. While works like these are advancing the frontiers of what is possible for RL to achieve, they require a lot of computational resources. At the same time, there is great demand for more practical, low-complexity solutions, that are specialized for simpler tasks, such as the control of dynamical systems.

Such problems have been studied for decades under the scope of Control Theory, with he first examples of data-driven control appearing as far back as the 70s [5]. Because of the simple structure of such problems, we can leverage analytical tools to obtain optimal solutions (LQR) or solve efficient optimization programs (MPC).

In this work we will attempt to derive a low-complexity solution that is building upon the Inverse Optimization framework laid out by [3], with the goal to match and even surpass the performance of simple MPC schemes, while cutting down on computational costs significantly.

1-1 Problem Statement

We consider linear discrete-time dynamical systems of the form

$$x_{t+1} = Ax_t + Bu_t + Ew_{t+1}, \tag{1-1a}$$
$$y_t = Cx_t, \tag{1-1b}$$

subject to unknown, potentially stochastic, disturbances w_t, which are not necessarily i.i.d. nor zero-mean.

The objective is to derive an MPC-like policy $\pi(\cdot)$ that will minimize the following cost

$$\text{Cost}_N^{\pi}(x_t, \mathbf{w}) = \sum_{k=0}^{N-1} c(x_{t+k}, u_{t+k}) + c_f(x_{t+N})$$

subject to dynamics (1-1), constraints $u_t \in \mathcal{U}(s_t)$, and $u_t = \pi(s_t)$, where s_t are some potentially non-Markovian features of past states and inputs, i.e. $s_t = \phi(\mathbf{x}_{1:t}, \mathbf{u}_{1:t})$, for some feature map $\phi(\cdot,\cdot)$. We want to use past information to mitigate the effect of disturbances in a data-driven fashion.

Assumptions The system matrices A, B, E, C are assumed to be known, but not the disturbance dynamics. Furthermore, E is assumed to be full column rank such that its left pseudoinverse exists, i.e, $E^\dagger E = I$. Additionally, the state x_t is available for measurement.

1-2 Our contributions

To achieve our goal we will need to find a parameterized state-action value function $\mathrm{Q}_\theta(s,u)$ such that the greedy policy

$$\pi_\theta(s) = \arg\min_{u \in \mathcal{U}(s)} \mathrm{Q}_\theta(s,u)$$

accomplishes the stated objective. In order to do that we will use the Inverse Optimization framework [3] to approximate a non-causal MPC policy that at time t has knowledge of the future disturbances that are going to occur. We will use past disturbance realizations up to time t as features, such that the resulting controller is causal and implementable.

We also propose a new min-max MPC scheme that will robustify against a ball around a future disturbance trajectory. This yields an exact convex reformulation due to the S-Lemma and, as far as we know, is a novel result. This will also act as a robust non-causal expert and we will try to approximate it with Inverse Optimization as well.

1-3 Structure of the thesis

In Section 2, we review some of the related literature in Reinforcement Learning and we also present a comparison of some recent results. In Section 3, we formulate the Inverse Optimization problem, and describe how it can approximate both a causal MPC expert and a non-causal one. In Section 4 we formulate the Robust MPC problem and provide its convex reformulation, as well as describe how it can be approximated by Inverse Optimization. In Section 5 we numerically showcase our proposal on a benchmark system. Finally, in Section 6 we conclude our work and offer some possible future research directions.

1-4 Notation

The dimension of a variable x is denoted by n_x. We denote with N the MPC horizon, with H the number of steps we look back, and with T the size of a dataset. We denote with bold a sequence of stacked variables, i.e, $\mathbf{x}_{1:N} = (x_1, x_2, \ldots, x_N)$, unless noted otherwise. When no exact range is given in the subscript, the default length of a bold variable is N, i.e. $\mathbf{x} = \mathbf{x}_{0:N-1}$ and $\mathbf{x}_t = \mathbf{x}_{t:t+N-1}$. With $\langle\cdot,\cdot\rangle$ we denote an inner product and with $\|x\|_A^2 = \langle x, Ax\rangle$. When a norm has no subscript, the default Euclidean norm is implied, i.e, $\|x\| = \langle x, x\rangle$. With $\otimes$ we denote the Kronecker product. Since the letter Q will be used to indicate both matrices and action-value functions, we will denote with Q the former and with $\mathrm{Q}(s,u)$ the latter, although it should usually be clear from the context. Finally, we will denote $\mathbb{N}_{[1,T]}$ the set of natural numbers between 1 and T, i.e, $\{1,\ldots,T\}$.

Chapter 2

Related Works

Here we will attempt to categorize some recent advances in Reinforcement Learning. Most works fall under one of three categories: supervised learning, model-free, and model-based. A summary of some advances in Reinforcement Learning for the control of dynamical systems (a.k.a data-driven control) can be found in Table 2-1. The entries of this table will be discussed in more detail in the subsections that follow.

2-1 Supervised Learning

Supervised Learning is when one has access to labeled data and tries to infer the relationship between input and labels. The two main components that define a Supervised Learning method are the hypothesis class, which defines how the approximation is parameterized, and the loss function which dictates how the parameters will be updated. In the context of Reinforcement Learning, such methods are named Imitation Learning, as they usually involve an expert, whose demonstrations take the place of labels. There are two general approaches in Imitation Learning: Behavioral Cloning, where the learner attempts to learn the policy directly; and Inverse Reinforcement Learning, where the learner tries to reconstruct the expert's reward function. Here we will focus more on the latter approach, as it is more relevant to our work.

2-1-1 Inverse Reinforcement Learning

The main motivation behind searching for the reward function instead of a policy stems from the assumption that the expert has an unknown reward function that shapes their decision. Therefore, if the reward function is discovered, then the expert's behavior can be imitated accurately. Furthermore, the reward is usually more dense in information than the policy itself, and learning it can allow for better generalization than just approximating the policy [26].

Table 2-1: Classification of some Reinforcement Learning methods for the control of dynamical systems.

Paper	Policy class [a]	Training		Dynamic [b]
		Model-based	Learning policy parameters	
Bradtke et al., 1994 [8]	Linear	✗	Policy Iteration	✗
Kiumarsi et al., 2014 [18]	Linear	✗	Policy Iteration	✗
Dean et al., 2018 [14]	Linear	✓	System Level Synthesis (SDP)	✗
Cohen et al., 2019 [9]	Linear	✓	Robust LQR (SDP)	✗
Lale et al., 2020 [19]	Linear	✓	Riccati	✗
Simchowitz & Foster, 2020 [29]	Linear	✓	Riccati	✗
Agarwal et al., 2019 [2]	Linear	✓	Online convex optimization	✓
Hazan et al., 2020 [17]	Linear	✓	Online convex optimization	✓
Foster & Simchowitz, 2020 [15]	Linear	✓	Online convex optimization	✓
De Persis & Tesi, 2020a;b [12, 13]	Linear	✓[2]	LMI-based design (SDP)	✗
van Waarde et al., 2020 [31]	Linear	✓[2]	S-Lemma-based design (SDP)	✗
Coulson et al., 2019; 2020 [10, 11]	MPC-N[1]	✓[2]	—	✗
Our work	MPC-1[1]	✓	Inverse Optimization	✓[3]

[a] Refers to the mapping between state/feature and input/action.
[b] Refers to whether the policy has states and dynamics of its own.
[1] The policy's action stems from the minimization of a (quadratic) objective subject to linear constraints. Such a policy is optimized over a N-step horizon.
[2] Instead of casting the model identification and policy learning as two independent optimization problems, these methods combine them into a single program, thereby finding the most optimal model for the control objective.
[3] While not necessary, these policies can be designed to be dependent on previous instances of the states and/or actions.

One of the first works in this area was [1], where reward functions were parameterized as linear combinations of feature mappings and policies were sampled from a set. Then feature expectations were computed for each policy, and the policies were linearly mixed until the feature expectations matched the expert's. The resulting algorithm was very similar to an SVM. Afterwards, a multitude of works started appearing that combined Feature Matching with Game Theory [30], or Entropy Maximization [35]. The interested reader can look for further information in the recent survey by [4]. One common issue with the aforementioned approaches, is that they usually deal with discrete state (and/or action) spaces.

Another line of work that is similar to the above is that of Inverse Optimization [3], where instead of searching for the reward function, the learner is looking for the action-value function that the expert is minimizing when making a decision, from which the expert's policy follows naturally as an optimization problem. Depending on the hypothesis class this approach can deal with continuous spaces. A more detailed look into Inverse can be found in Section 3.

2-2 Model-free methods

Model-free methods, as the name suggests, attempt to find the optimal policy without requiring any knowledge of the system's dynamics. There are two main subcategories: policy gradient methods, where the policy is directly parameterized and is updated by minimizing a performance measure; and Q-learning methods, where the objective is to find the optimal

Q-function, from which the optimal policy also follows. Here we are going to focus on the latter, as it is more relevant to our work.

There are also algorithms that combine elements from both categories, such as the Actor-Critic architecture, but these are outside of the scope of this discussion. As the most relevant method to our work is Q-Learning, this is what we are going to focus on.

2-2-1 Q-Learning

One of the seminal algorithms for RL is Watkin's Q-Learning [33], which considers discrete state and action spaces. The core idea behind it is that the learner can discover the optimal (greedy) policy $\pi^*(s) = \arg\min_u \mathrm{Q}^*(s,u)$ by performing Dynamic Programming iterations over the Q-function estimate $\mathrm{Q}_\theta(s,u)$ with updates in the direction that minimizes the Bellman error

$$\ell_\theta^{\mathrm{BE}}(s,u,s') = \mathrm{Q}_\theta(s,u) - \left(c(s,u) + \gamma \min_{u'} \mathrm{Q}_\theta(s',u')\right) \tag{2-1}$$

since, if θ is such that $\mathrm{Q}_\theta = \mathrm{Q}^*$, then $\ell_\theta^{\mathrm{BE}}(s,u,s') = 0$. The advantage of Q-Learning is that it is independent of the policy that is actually being executed to collect data.

However, a lot of problems necessitate the use of continuous spaces, which cannot be discretized due to the curse of dimensionality. This difficulty has been overcome in the context of LQR problems [8, 18], where a modified version of Q-Learning using Policy Iteration has been proposed. However, due to the scope of these works they are still considering the unconstrained setting, and the resulting policy is linear.

Most modern Reinforcement Learning research uses function approximators in a deep configuration. Specifically, Deep Reinforcement Learning methods use Neural Networks as approximators for the Q-function [23], and as such lift the restriction of working with only finite-dimensional state spaces. However, this approach still relies on low-dimensional discrete action spaces; otherwise, the non-convexity of the Q-function would require solving inefficient optimization programs at each iteration. This was overcome in [20], by using the aforementioned Deep Q-Network and an Actor-Critic architecture. This meant that the policy had its own network which approximated the greedy policy. Even in these Deep Reinforcement Learning approaches however, the updates for the Q-Network is still driven by gradients of the Bellman loss (2-1).

2-3 Model-based methods

The traditional approach has been that of system identification and control design through a variety of methods such as solving the Riccati equation, robust controller design, etc. This is reflected in the following lines of work, where in regret minimization approaches, it is shown that these traditional methods can also work in iterative/online schemes, whereas research based on Willems' lemma combines system identification and control design in one step.

2-3-1 Regret Minimization

There has also been a lot of recent research in the model-based setting in the context of online optimization. Specifically, these works target systems of the form (1-1), where the model parameters are unknown, and are searching for causal policies of the form:

$$\pi_K(x_t) = -K_0 x_t - \sum_{\tau=1}^{H} K_\tau w_{t-\tau} \tag{2-2}$$

The aim is to find the appropriate gains K such that the regret in cost between the played policy and the best policy in hindsight is minimized. Note that even though that's the aim, regret is not minimized explicitly; rather it's proven to scale logarithmically or as a square root of time.

When w_t is entirely stochastic and i.i.d, then it is known from Optimal Control Theory that certainty equivalence is optimal and the disturbance feedback gains can be ignored. Therefore, it is known that the optimal policy is when K_0 is the solution the to Riccati equation. Therefore, works focusing on this setting usually require an initial stabilizing controller, and by using ϵ-greedy exploration they are able to find the A, B matrices through (online) least-squares methods. The estimates for A, B along with their confidence intervals are then either used to find robust-in-the-uncertainty LQR controllers by solving semidefinite optimization programs [14, 9], or they are projected to some safe sets determined by the model's confidence and then the LQR gain is found by solving the Riccati equation [19, 29].

On the other hand, when the disturbances have their own (unknown) dynamics the rightmost terms of (2-2) can no longer be neglected. Furthermore, a good model is necessary in order to properly observe the disturbances, and thus there needs to be a distinct identification phase, instead of having continual exploration as before. Works like [2, 17, 15] exploit the linearity of the dynamics by minimizing the cost contribution incurred by the disturbances alone. This happens in hindsight, as the disturbances are not immediately observable at time t.

2-3-2 Willems' Lemma

This lemma states that trajectories of linear systems can be described as a linear combination of previous trajectories that are sufficiently long and rich in information [34]. This means that one can replace a model of the form (1-1) with a Hankel matrix containing past trajectories, and thereby express any new trajectory as a linear combination of the columns of this matrix. While it's not a very recent result, and it has been used for some time in Subspace Identification methods [32], its newfound applications for direct data-driven control have recently gathered a lot of research attention from the control community.

The idea of combining identification and control design in a single step can be both detrimental and advantageous. It is common for more than one system to be explained by the data – especially when stochastics are involved – and by combining the two phases we get the system that minimizes the control cost. Furthermore, even though these works are often described as model-free, in reality they are not: the data itself serves as an input-output model.

In [12, 13], the authors utilize Willems' Lemma to formulate controller synthesis for a system with process noise as a semidefinite optimization program. By utilizing LMIs they are able to exploit their inherent robustness properties, and thus the resulting controller is robust to additive noise. No statistics on the noise are assumed, only that it is sufficiently small compared to the underlying states of the system. Another approach to this can be found in [31], where it is shown that the conditions on the data can be relaxed such that only the feasibility of the optimization program is required, not the stricter persistency of excitation condition. However, for LQR synthesis the stricter PE condition is still necessary. Further, the authors propose a different representation of the dynamics along with LQR synthesis program based on the S-Lemma, which makes the decision variables independent of the size of the dataset, allowing this method to scale to larger dimensions.

While a lot of research in this topic only considers the unconstrained scenario, there are also works such as DeePC [10] where the constrained setting is examined. Specifically, as in other works based on Willems' Lemma, the linear dynamics constraints in an MPC are substituted with a Hankel matrix containing past trajectories. One major caveat is that the decision variable has a significantly higher dimension than a typical MPC scheme, making the problem computationally harder for large system dimensions. Under the presence of stochastics, this approach warrants regularization on the decision variable or else trajectories not belonging to the system's original subspace may be predicted, deteriorating the controller's performance [11].

Chapter 3

Inverse Optimization for RL

3-1 Inverse Optimization as supervised learning

The goal of Inverse Optimization is to learn the behavior of an expert whose actions depend on an external signal. We assume that, for a given state $s \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$, the expert's decisions $u^{\text{ex}} \in \mathcal{U}(s) \subseteq \mathbb{R}^{n_u}$ stem from the minimization of a value function $\mathrm{Q}^{\text{ex}}(s, u)$ which assesses the cost of each decision. In other words, the expert follows the greedy policy

$$\pi^{\text{et}}(s) = \arg \min_{u \in \mathcal{U}(s)} \mathrm{Q}^{\text{et}}(s, u). \tag{3-1}$$

Since $\mathrm{Q}^{\text{ex}}(s, u)$ is unknown to us, our goal is to approximate it with a parametric model $\mathrm{Q}_\theta(s, u)$. To that end, we will use the suboptimality loss function, which was first introduced in [24]:

$$\ell_\theta^{\text{sub}}(s, u^{\text{et}}) = \mathrm{Q}_\theta(s, u^{\text{et}}) - \min_{u \in \mathcal{U}(s)} \mathrm{Q}_\theta(s, u) \tag{3-2}$$

The main motivation behind this loss is that since u^{ex} is the minimizer of Q^{ex}, we want to find the parameters in the hypothesis space that best explain u^{ex}. The loss goes to zero only if the parameterization of Q^{ex} is exact. Furthermore, as we will see later on, this type of loss function has beneficial computational properties when the hypothesis class for Q_θ is quadratic.

Remark 1 (Relationship to Q-Learning). There are parallels that can be drawn between Q-Learning and Inverse Optimization: the policy class is the same and the goal of both methods is to recover the optimal Q-function. However, the notion of optimality is different between that two methods, since the losses involved are different. As mentioned in Section 2-2-1, Q-Learning is an unsupervised method where the aim is to minimize the Bellman error loss (2-1). On the other hand, in Inverse Optimization we want to minimize the degree of suboptimality for a certain hypothesis between what the expert demonstrated and what is actually its current minimum.

Remark 2 (Relationship to Regret). At first one might think that the suboptimality loss (3-2) is similar with the notion of Regret that was discussed in Section 2-3-1. While that is somewhat true in a conceptual level, as both measure the degree of suboptimality there are key differences between them. Specifically, Regret is usually not directly optimized over; rather, it is used a performance metric for how well an online optimization algorithm approximates the full offline solution of the same problem.

3-1-1 Hypothesis class

Since we will be dealing with linear systems such as (1-1) with quadratic costs, it makes sense to choose the following quadratic hypothesis class:

$$\mathcal{Q} = \{\mathrm{Q}_\theta(s,u) = \langle u, \theta_{uu} u\rangle + 2\langle s, \theta_{su} u\rangle : \theta \in \Theta\} \tag{3-3}$$

where the parameter space $\Theta \subseteq \mathbb{R}^{(n_s+n_u)\times(n_s+n_u)}$ represents a subset of all square matrices. Specifically, we will work with strongly convex quadratic matrices, i.e:

$$\Theta = \left\{ \theta = \begin{bmatrix} 0 & \theta_{su} \\ \theta_{su}^\intercal & \theta_{uu} \end{bmatrix} : \theta_{uu} \succcurlyeq I_{n_u} \right\} \tag{3-4}$$

Notice that the mapping $\theta \mapsto \mathrm{Q}_\theta(s,u)$ is linear, and therefore the suboptimality loss (3-2) is convex in θ. Additionally, since we are only interested in the minimizers of Q_θ, we do not care about the cost incurred by the current state, thus the relevant term is omitted from (3-3) and (3-4).

3-1-2 Learning the optimal parameters

Given a dataset $\mathcal{D} = \{(\hat{s}_t, \hat{u}_t^{\mathrm{ex}})\}_{t=1}^T$ of states $\hat{s}_t$ and expert actions $\hat{u}_t^{\mathrm{ex}} = \pi^{\mathrm{ex}}(\hat{s}_t)$ we can formulate the learning problem of Q_θ as the following optimization program:

$$\min_{\theta\in\Theta} \sum_{t=1}^{T} \ell_\theta^{\mathrm{sub}}(\hat{s}_t, \hat{u}_t^{\mathrm{ex}}) \tag{3-5}$$

We will consider the case when the input constraint set is $\mathcal{U}(s)$ is linear, and it can be written as:

$$\mathcal{U}(s) = \{u : G(s)u \le h(s)\}.$$

For the sake of notation we will denote $\hat{G}_t = G(s_t)$ and $\hat{h}_t = h(s_t)$. Then, (3-5) admits the following convex reformulation (Corollary 3, [3]):

$$\begin{aligned}
\min_{\theta,\gamma_{1:T},\boldsymbol{\lambda}_{1:T}} \quad & \sum_{t=1}^{\intercal} \mathrm{Q}_\theta(\hat{s}_t, \hat{u}_t^{\mathrm{ex}}) + \frac{1}{4}\gamma_t + \hat{h}_t^\intercal \lambda_t \\
\text{s.t.} \quad & \theta \in \Theta, \\
& \lambda_t \ge 0, \; t \in \mathbb{N}_{[1,T]} \\
& \begin{bmatrix} \theta_{uu} & \hat{G}_t^\intercal \lambda_t + 2\theta_{su}^\intercal \hat{s}_t \\ \star & \gamma_t \end{bmatrix} \succcurlyeq 0, \; t \in \mathbb{N}_{[1,T]}
\end{aligned} \tag{3-6}$$

Let us note here that even the above has been formulated for a single dataset, we can easily extend this to multiple ones by just concatenating everything into a single one. This is possible because the Inverse Optimization policy depends only on the current feature s_t.

3-2 MPC expert for Inverse Optimization

Consider a dynamics model like (1-1), but without any external disturbances. Let us introduce the quadratic stage costs

$$c(x,u) = \|x\|_{Q_x}^2 + \|u\|_{Q_u}^2 \text{ and } c_f(x,u) = \|x\|_{Q_f}^2$$

where $Q_x, Q_f \succcurlyeq 0$ and $Q_u \succ 0$. Then, we can define the following MPC problem with horizon N:

$$\begin{aligned} V_N^{\text{mpc}}(x_t) := \min_{\mathbf{u}_t} & \sum_{k=0}^{N-1} c(x_{t+k}, u_{t+k}) + c_f(x_{t+N}) \\ \text{s.t. } & x_{t+k+1} = Ax_{t+k} + Bu_{t+k}, \ k \in \mathbb{N}_{[0,N-1]}, \\ & G_x x_{t+k+1} \le h_x, \ k \in \mathbb{N}_{[0,N-1]}, \\ & G_u u_{t+k} \le h_u, \ k \in \mathbb{N}_{[0,N-1]}, \end{aligned} \tag{3-7}$$

Due to the principle of optimality we can break up the above problem as

$$\begin{aligned} V_N^{\text{mpc}}(x_t) = \min_{u_t} \ & c(x_t, u_t) + V_{N-1}^{\text{mpc}}(Ax_t + Bu_t) \\ \text{s.t. } & G_x(Ax_t + Bu_t) \le h_x, \\ & G_u u_t \le h_u \end{aligned}$$

where V_{N-1}^{mpc} is defined accordingly. Therefore, the Q-function of the MPC agent is $\mathrm{Q}^{\text{mpc}}(x,u) = c(x,u) + V_{N-1}^{\text{mpc}}(Ax + Bu)$ which is defined over the constraint set

$$\mathcal{U}^{\text{mpc}}(x) := \left\{ u : \begin{bmatrix} G_x B \\ G_u \end{bmatrix} u \le \begin{bmatrix} h_x - G_x A x \\ h_u \end{bmatrix} \right\}$$

To approximate $\mathrm{Q}^{\text{mpc}}(x,u)$ with Inverse Optimization, we must solve (3-6) with $s_t = x_t$ and $u_t^{\text{ex}} = \pi^{\text{mpc}}(x_t)$, where

$$\begin{aligned} \pi^{\text{mpc}}(x) = \arg\min_u \ & \mathrm{Q}^{\text{mpc}}(x,u) \\ \text{s.t. } & u \in \mathcal{U}^{\text{mpc}}(x) \end{aligned} \tag{3-8}$$

Since the constraints are linear, it is known that Q^{mpc} will be piecewise quadratic; therefore, we cannot hope for an exact approximation by Q_θ. Despite that fact, this has been shown to work quite successfully in [3]. Approximating (3-8) does not require knowledge of the stage costs as it can be seen in (3-6), but if there are state constraints, then a dynamics model could be required; otherwise, solving (3-6) does not require any knowledge of the dynamics.

Remark 3 (Trajectory tracking). The above problem has been formulated with stabilization in mind, but it is equally as easy to track a trajectory r_t: the stage costs will have to be time varying, i.e, $c_{t+k}(x_{t+k}, u_{t+k}) = \|y_{t+k} - r_{t+k}\|_{Q_y}^2 + \|u_{t+k}\|_{Q_u}^2$, and the features s_t will have to contain the trajectory up until time $t+N$, i.e, $s_t = (x_t, \mathbf{r}_{t+1})$. Notice that r_t does not need to be included in the features even though it could exist in the MPC objective function, since it does not contribute to the cost incurred by u_t.

Remark 4 (Unconstrained case). If there are no constraints, then we have a finite-horizon LQR problem, whose Q-function is known to be quadratic and positive definite. Furthermore, if the horizon N is large, then the Q-function will approach the stationary solution of the discrete Riccati equation. Also, notice that when there are no constraints, the approximate policy becomes $\pi_\theta(s) = -\theta_{uu}^{-1}\theta_{su}^\intercal s$, which implies that instead of searching for Q_θ, we can search directly for the optimal feedback gain.

3-3 Exploiting in-hindsight information

When the disturbances have dynamics of their own there is a relationship between past and future realizations, i.e, we can use past information to make an estimate about the future. Our goal is to infer this relationship directly from the data itself and at the same time find the best way to use that information in a feedback setting.

When we have access to the model parameters of (1-1), A, B, E, at time t we can measure w_t since

$$w_t = E^\dagger(x_t - Ax_{t-1} - Bu_{t-1})$$

If the next N disturbance realizations were available at time t, we would be able to compute the optimal disturbance-aware input sequence. Hence, we can define the non-causal MPC problem:

$$\begin{aligned} V_N^{\text{nc-mpc}}(x_t, \mathbf{w}_{t+1}) \coloneqq \min_{\mathbf{u}_t} & \sum_{k=0}^{N-1} c(x_{t+k}, u_{t+k}) + c_f(x_{t+N}) \\ \text{s.t. } & x_{t+k+1} = Ax_{t+k} + Bu_{t+k} + Ew_{t+k+1},\ k \in \mathbb{N}_{[0,N-1]}, \\ & G_x x_{t+k+1} \le h_x,\ k \in \mathbb{N}_{[0,N-1]}, \\ & G_u u_{t+k} \le h_u,\ k \in \mathbb{N}_{[0,N-1]}, \end{aligned} \tag{3-9}$$

Similarly to Section 3-2, we can define $\mathrm{Q}^{\text{nc-mpc}}(x, \mathbf{w}, u)$ and $\mathcal{U}^{\text{nc-mpc}}(x, \mathbf{w})$ accordingly and therefore we obtain the non-causal MPC expert policy:

$$\begin{aligned} \pi^{\text{nc-mpc}}(x, \mathbf{w}) = \arg\min_{u}\ & \mathrm{Q}^{\text{nc-mpc}}(x, \mathbf{w}, u) \\ \text{s.t. }\ & u \in \mathcal{U}^{\text{nc-mpc}}(x, \mathbf{w}) \end{aligned} \tag{3-10}$$

As this policy is non-causal it cannot be implemented in real time, but only in hindsight, as $\mathbf{w}_{t+1}$ is not available at time t. However, when the disturbances have their own dynamics, an estimate for $\mathbf{w}_{t+1}$ can be inferred from the past H realizations. Hence, we can use $\mathbf{w}_{t-H+1:t}$ as features when approximating $\pi^{\text{nc-mpc}}(x, \mathbf{w})$ with Inverse Optimization, and by doing so we implicitly infer the prediction relationship between $\mathbf{w}_{t-H+1:t}$ and $\mathbf{w}_{t+1}$ and also the optimal way to counteract $\mathbf{w}_{t+1}$.

The procedure used to approximate the non-causal MPC expert with Inverse Optimization is outlined in Algorithm (1). Let us stress again the fact that while during training a non-causal controller is used, the resulting controller is completely causal as it only depends on disturbance realizations that are observable at time t.

Remark 5 (Handling of non-causal constraints)*.* When solving problem (3-6) and there are state constraints involved, the future disturbance trajectory $\mathbf{w}_{t+1}$ will not be known to the Inverse Optimization policy. Therefore, we will have to use a causal version of the constraints, such as $\mathcal{U}^{\text{mpc}}$, potentially with some constraint softening to avoid infeasibility issues. However, as the expert takes into account $\mathbf{w}_{t+1}$, the learned Q-function Q_θ will have an embedded cost on $\mathbf{w}_{t-H+1:t}$ that will aid in constraint satisfaction. Learning or approximating non-causal constraints with causal ones is a potential research avenue for the future.

Remark 6 (Comparisons with recent literature)*.* This idea is similar to the Disturbance Feedback Control scheme used in some recent papers related to online control for adversarial

Algorithm 1 Using in-hindsight information for Inverse Optimization

```
Input: Trajectory $\{(\hat{x}_t, \hat{u}_t)\}_{t=1}^{T}$, horizon $N$, non-causal expert $\pi^{\text{nc}}(x, \mathbf{w})$
for $t = 1$ to $t = T$ do
    Observe $\hat{w}_{t+1} = E^{\dagger}(\hat{x}_{t+1} - A\hat{x}_t - B\hat{u}_t)$
    Let $\tau = t - N + 1$
    if $\tau \geq H + 1$ then
        Let $\hat{s}_\tau = (\hat{x}_\tau, \hat{\mathbf{w}}_{\tau-H+1:\tau})$
        Let $\hat{u}_\tau^{\text{et}} = \pi^{\text{nc}}(\hat{x}_\tau, \hat{\mathbf{w}}_{\tau+1})$
    end if
end for
Solve (3-6) with $\{\hat{s}_\tau, \hat{u}_\tau^{\text{et}})\}_{\tau=H+1}^{T-N+1}$ to obtain $\theta^*$.
Return: $\theta^*$
```

disturbances [17, 2, 15]. Additionally, a similar problem was also considered in [16], where the authors wanted to approximate a non-causal controller with a causal one, but in an offline fashion. Contrary to these works, which consider a linear policy class akin to (2-2) and do not support constraints on neither state or input, our controller can handle constraints and is nonlinear in nature. As mentioned previously, in the unconstrained scenario the Inverse Optimization controller would be equivalent to policy (2-2), and it would be more efficient to search directly for the optimal linear feedback gains instead of Q_θ.

Remark 7 (Model mismatch)*.* In the case of model mismatch, the non-causal MPC expert can be used as an approximation for an MPC expert formulated with the correct model. However, the quality of this approximation largely depends on how wrong our model is, and whether the mismatch is due to non-linearities. In the case of the latter, the closed-loop system might end up in a different equilibrium. If the non-causal MPC expert yields good results, then we can expect its approximation by Inverse Optimization to be similarly good as well.

Chapter 4

Robust disturbance-aware MPC

The non-causal MPC expert (3-10) optimizes directly against the noisy disturbance trajectory directly. However, due to stochastics and/or potential distribution shifts in data, performance might be degraded and we could even have instability arise. Therefore, an expert that is robust to such issues might be warranted. Before we proceed, let us consider the non-causal MPC problem (3-9) written in vectorized form:

$$\begin{aligned} \min_{\mathbf{u}_t} \ & \|\mathbf{A}x_t + \mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{w}_{t+1}\|^2_{\mathbf{Q_x}} + \|\mathbf{u}_t\|^2_{\mathbf{Q_u}} \\ \text{s.t. } & \mathbf{F}x_t + \mathbf{G}\mathbf{u}_t \leq \mathbf{h}(\mathbf{w}_{t+1}) \end{aligned} \tag{4-1}$$

where

$$\mathbf{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \mathbf{B} = \begin{bmatrix} B & 0 & 0 & 0 \\ AB & B & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix},$$

$$\mathbf{E} = \begin{bmatrix} E & 0 & 0 & 0 \\ AE & E & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}E & A^{N-2}E & \cdots & E \end{bmatrix},$$

$$\mathbf{Q_x} = \begin{bmatrix} I_{N-1} \otimes Q_x & 0 \\ 0 & Q_f \end{bmatrix}, \mathbf{Q_u} = I_N \otimes Q_u$$

and

$$\mathbf{F} = \begin{bmatrix} \mathbf{G_x}\mathbf{A} \\ 0 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} \mathbf{G_x}\mathbf{B} \\ \mathbf{G_u} \end{bmatrix}, \mathbf{h}(\mathbf{w}) = \begin{bmatrix} \mathbf{h_x} - \mathbf{G_x}\mathbf{E}\mathbf{w} \\ \mathbf{h_u} \end{bmatrix}$$

with $\mathbf{G_x} = I_N \otimes G_x$, $\mathbf{G_u} = I_N \otimes G_u$, $\mathbf{h_x} = \mathbf{1}_N \otimes h_x$, $\mathbf{h_u} = \mathbf{1}_N \otimes h_u$. In what is to follow, we are going to omit the time indexes in variables for the sake of notation.

4-1 Robustification around disturbance trajectory

Suppose that we access to a disturbance trajectory $\mathbf{w}$ of length N and we want to robustify against possible variations around it. Therefore let us define a ball around the measured trajectory

$$\mathfrak{B}_{\varrho,P}(\mathbf{w}) = \left\{\bar{\mathbf{w}} : \|\bar{\mathbf{w}} - \mathbf{w}\|_P^2 \leq \varrho^2\right\} \tag{4-2}$$

Let us denote the constraint set of (4-1) as

$$\mathcal{U}_N^{\text{nc-mpc}}(x, \mathbf{w}) = \{\mathbf{u} : \mathbf{F}x + \mathbf{G}\mathbf{u} \leq \mathbf{h}(\mathbf{w})\}$$

so that we can define the following robust constraint set over the uncertainty set (4-2)

$$\mathcal{U}_N^{\text{nc-rmpc}}(x, \mathbf{w}) = \{\mathbf{u} : \mathbf{u} \in \mathcal{U}_N^{\text{nc-mpc}}(x, \bar{\mathbf{w}}),\ \forall \bar{\mathbf{w}} \in \mathfrak{B}_{\varrho,P}(\mathbf{w})\} \tag{4-3}$$

We are then interested in solving the following min-max optimization problem

$$\min_{\mathbf{u}\in\mathcal{U}_N^{\text{nc-rmpc}}(x,\mathbf{w})} \max_{\bar{\mathbf{w}}\in\mathfrak{B}_{\varrho,P}(\mathbf{w})} \|\mathbf{A}x + \mathbf{B}\mathbf{u} + \mathbf{E}\bar{\mathbf{w}}\|_{\mathbf{Q}_\mathbf{x}}^2 + \|\mathbf{u}\|_{\mathbf{Q}_\mathbf{u}}^2 \tag{4-4}$$

While min-max MPC problems such as the above have been considered before in the literature [21], they relied on conservative relaxations. Contrary to such works however, problem (4-4) has an exact convex reformulation, but before we proceed with it we need to introduce the the following Lemma.

Lemma 1 (Polytopic representation of robust constraint set)**.** *Given that $P \succ 0$, the constraint set* (4-3) *has the following polytopic representation*

$$\mathbf{F}x + \mathbf{G}\mathbf{u} \leq \underline{\mathbf{h}}(\mathbf{w})$$

where $\underline{\mathbf{h}}(\mathbf{w})^\intercal = \begin{bmatrix}(\mathbf{h}_\mathbf{x} - \overline{g}(\mathbf{w}))^\intercal & \mathbf{h}_\mathbf{u}{}^\intercal\end{bmatrix}$, $\overline{g}(\mathbf{w})^\intercal = \begin{bmatrix}\overline{g}_1(\mathbf{w}) & \overline{g}_2(\mathbf{w}) & \dots\end{bmatrix}$, *and* $\overline{g}_i(\mathbf{w}) = \varrho \left\|P^{-1/2}g_i\right\| + g_i^\intercal\mathbf{w}$, $\forall i$. *The vectors g_i are such that* $[\mathbf{G}_\mathbf{x}\mathbf{E}\bar{\mathbf{w}}]_i = g_i^\intercal\bar{\mathbf{w}}$.

Proof. The original constraint expresses an row-wise inequality. With the parameterization $[\mathbf{G}_\mathbf{x}\mathbf{E}\bar{\mathbf{w}}]_i = g_i^\intercal\bar{\mathbf{w}}$, the inequality

$$\mathbf{F}x + \mathbf{G}\mathbf{u} \leq \mathbf{h}(\bar{\mathbf{w}}),\ \forall \bar{\mathbf{w}} \in \mathfrak{B}_{\varrho,P}(\mathbf{w})$$

is equivalent to solving the following optimization program for every i:

$$\overline{g}_i(\mathbf{w}) = \max_{\bar{\mathbf{w}}}\left\{g_i^\intercal\bar{\mathbf{w}},\ \text{s.t.}\ \|\bar{\mathbf{w}} - \mathbf{w}\|_P^2 \leq \varrho^2\right\}$$

To that end, let $\tilde{\mathbf{w}} = \varrho^{-1}P^{1/2}(\bar{\mathbf{w}} - \mathbf{w})$. Then the above becomes

$$\overline{g}_i(\mathbf{w}) = \max_{\tilde{\mathbf{w}}}\left\{g_i^\intercal(\varrho P^{-1/2}\tilde{\mathbf{w}} + \mathbf{w}),\ \text{s.t.}\ \|\tilde{\mathbf{w}}\| \leq 1\right\}$$

The maximization of a linear function on the unit disk has an analytical solution and that is

$$\overline{g}_i(\mathbf{w}) = \varrho\left\|P^{-1/2}g_i\right\| + g_i^\intercal\mathbf{w}$$

By putting everything together we conclude the proof. □

We are now in position to introduce the following theorem:

Theorem 1 (Semidefinite reformulation)**.** *The min-max MPC problem* (4-4) *admits the following convex reformulation*

$$\begin{aligned}\min_{\mathbf{u},\lambda,\gamma_1,\gamma_2} \quad & \gamma_1+\gamma_2 \\ \text{s.t.} \quad & \lambda \geq 0, \\ & \mathbf{F}x+\mathbf{G}\mathbf{u} \leq \underline{\mathbf{h}}(\mathbf{w}), \\ & \begin{bmatrix} \mathbf{E}^\intercal \mathbf{Q_x}\mathbf{E}-\lambda P & \mathbf{E}^\intercal\mathbf{Q_x}\left(\mathbf{A}x+\mathbf{B}\mathbf{u}\right)+\lambda P\mathbf{w} \\ \star & -\gamma_1-\lambda\left(\|\mathbf{w}\|_P^2-\varrho^2\right)\end{bmatrix} \preccurlyeq 0 \\ & \begin{bmatrix} -I_N & \mathbf{Q}^{1/2}\mathbf{u} \\ \star & 2\left\langle \mathbf{B}^\intercal\mathbf{Q_x}\mathbf{A}x,\mathbf{u}\right\rangle+\|\mathbf{A}x\|_{\mathbf{Q_x}}^2-\gamma_2\end{bmatrix} \preccurlyeq 0 \end{aligned} \tag{4-5}$$

Proof. From Lemma 1, we have that the problem (4-4) is equivalent to

$$\begin{aligned}\min_{\mathbf{u}} \max_{\bar{\mathbf{w}}\in\mathfrak{B}_{\varrho,P}(\mathbf{w})} \; & \|\mathbf{A}x+\mathbf{B}\mathbf{u}+\mathbf{E}\bar{\mathbf{w}}\|_{\mathbf{Q_x}}^2+\|\mathbf{u}\|_{\mathbf{Q_u}}^2 \\ \text{s.t.} \; & \mathbf{F}x+\mathbf{G}\mathbf{u}\leq\underline{\mathbf{h}}(\mathbf{w})\end{aligned}$$

Let us denote the inner maximization as

$$J(\mathbf{u})=\max_{\bar{\mathbf{w}}\in\mathfrak{B}_{\varrho,P}(\mathbf{w})}\|\mathbf{A}x+\mathbf{B}\mathbf{u}+\mathbf{E}\bar{\mathbf{w}}\|_{\mathbf{Q_x}}^2+\|\mathbf{u}\|_{\mathbf{Q_u}}^2$$

and its corresponding Lagrangian as

$$\begin{aligned}\mathcal{L}_J(\lambda,\mathbf{u},\bar{\mathbf{w}})=&\|\mathbf{A}x+\mathbf{B}\mathbf{u}+\mathbf{E}\bar{\mathbf{w}}\|_{\mathbf{Q_x}}^2 \\ &+\|\mathbf{u}\|_{\mathbf{Q_u}}^2-\lambda\left(\|\bar{\mathbf{w}}-\mathbf{w}\|_P^2-\varrho^2\right)\end{aligned}$$

After some manipulations and rearrangements we have

$$\begin{aligned}\mathcal{L}_J(\lambda,\mathbf{u},\bar{\mathbf{w}})=&\left\langle\bar{\mathbf{w}},\left(\mathbf{E}^\intercal\mathbf{Q_x}\mathbf{E}-\lambda P\right)\bar{\mathbf{w}}\right\rangle \\ &+2\left\langle\mathbf{E}^\intercal\mathbf{Q_x}\left(\mathbf{A}x+\mathbf{B}\mathbf{u}\right)+\lambda P\mathbf{w},\bar{\mathbf{w}}\right\rangle \\ &+\|\mathbf{A}x+\mathbf{B}\mathbf{u}\|_{\mathbf{Q_x}}^2+\|\mathbf{u}\|_{\mathbf{Q_u}}^2 \\ &-\lambda\left(\|\mathbf{w}\|_P^2-\varrho^2\right)\end{aligned}$$

Let us introduce the following notation

$$\begin{aligned}\Lambda(\lambda)=&\mathbf{E}^\intercal\mathbf{Q_x}\mathbf{E}-\lambda P \\ M(\lambda,\mathbf{u})=&\mathbf{E}^\intercal\mathbf{Q_x}\left(\mathbf{A}x+\mathbf{B}\mathbf{u}\right)+\lambda P\mathbf{w} \\ \nu_1(\lambda)=&-\lambda\left(\|\mathbf{w}\|_P^2-\varrho^2\right) \\ \nu_2(\mathbf{u})=&\|\mathbf{A}x+\mathbf{B}\mathbf{u}\|_{\mathbf{Q_x}}^2+\|\mathbf{u}\|_{\mathbf{Q_u}}^2 \\ \nu(\lambda,\mathbf{u})=&\nu_1(\lambda)+\nu_2(\mathbf{u})\end{aligned}$$

The dual of this problem is then

$$d_J(\lambda, \mathbf{u}) = \max_{\bar{\mathbf{w}}} \mathcal{L}_J(\lambda, \mathbf{u}, \bar{\mathbf{w}}) = \begin{cases} -M(\lambda, \mathbf{u})^\intercal \Lambda(\lambda)^\dagger M(\lambda, \mathbf{u}) + \nu(\lambda, \mathbf{u}), \\ \text{ if } \Lambda(\lambda) \preccurlyeq 0 \text{ and} \\ \quad \left(I - \Lambda(\lambda)\Lambda(\lambda)^\dagger\right) M(\lambda, \mathbf{u}) = 0 \\ +\infty, \text{ otherwise} \end{cases}$$

Strong duality holds due to the S-Lemma [7]. Therefore, $J(\mathbf{u}) = \min_{\lambda \geq 0} d_J(\lambda, \mathbf{u})$. Now consider the following epigraph reformulation

$$\begin{aligned} J(\mathbf{u}) = \min_{\lambda} & \gamma_1 + \nu_2(\mathbf{u}) \\ \text{s.t. } & \lambda \geq 0, \\ & \Lambda(\lambda) \preccurlyeq 0, \\ & \left(I - \Lambda(\lambda)\Lambda(\lambda)^\dagger\right) M(\lambda, \mathbf{u}) = 0, \\ & - M(\lambda, \mathbf{u})^\intercal \Lambda(\lambda)^\dagger M(\lambda, \mathbf{u}) + \nu_1(\lambda) \leq \gamma_1 \end{aligned}$$

The last three constraints can be cast as an LMI using the non-strict Schur complement [6] and we have

$$\begin{aligned} J(\mathbf{u}) = \min_{\lambda, \gamma_1} & \gamma_1 + \nu_2(\mathbf{u}) \\ \text{s.t. } & \lambda \geq 0, \\ & \begin{bmatrix} \Lambda(\lambda) & M(\lambda, \mathbf{u}) \\ \star & \nu_1(\lambda) - \gamma_1 \end{bmatrix} \preccurlyeq 0 \end{aligned}$$

Therefore the overall robust MPC problem can now be written as $\min_{\mathbf{u}} \{J(\mathbf{u}), \text{ s.t. } \mathbf{F}x + \mathbf{G}\mathbf{u} \leq \underline{\mathbf{h}}\}$. In order to write this in the standard SDP form, we will have to use another epigraph reformulation, that of $\nu_2(\mathbf{u})$:

$$\begin{aligned} \min_{\mathbf{u}, \lambda, \gamma_1, \gamma_2} \quad & \gamma_1 + \gamma_2 \\ \text{s.t. } & \lambda \geq 0, \\ & \begin{bmatrix} \Lambda(\lambda) & M(\lambda, \mathbf{u}) \\ \star & \nu_1(\lambda) - \gamma_1 \end{bmatrix} \preccurlyeq 0, \\ & \mathbf{F}x + \mathbf{G}\mathbf{u} \leq \underline{\mathbf{h}}(\mathbf{w}), \\ & \|\mathbf{A}x + \mathbf{B}\mathbf{u}\|^2_{\mathbf{Q_x}} + \|\mathbf{u}\|^2_{\mathbf{Q_u}} \leq \gamma_2 \end{aligned}$$

The last constraint can now be written as

$$\gamma_2 \geq \langle \mathbf{u}, (\mathbf{B}^\intercal \mathbf{Q_x} \mathbf{B} \mathbf{Q_u})\, \mathbf{u} \rangle + 2 \langle \mathbf{B}^\intercal \mathbf{Q_x} \mathbf{A} x, \mathbf{u} \rangle + \|\mathbf{A}x\|^2_{\mathbf{Q_x}}$$

Denote

$$\begin{aligned} \mathbf{Q} =& \mathbf{B}^\intercal \mathbf{Q_x} \mathbf{B} + \mathbf{Q_u} \\ \delta(\mathbf{u}) =& 2 \langle \mathbf{B}^\intercal \mathbf{Q_x} \mathbf{A} x, \mathbf{u} \rangle + \|\mathbf{A}x\|^2_{\mathbf{Q_x}} \end{aligned}$$

and then we get that this constraint is equivalent to the following LMI

$$\begin{bmatrix} -I_N & \mathbf{Q}^{1/2}\mathbf{u} \\ \star & \delta(\mathbf{u}) - \gamma_2 \end{bmatrix} \preccurlyeq 0$$

Let us denote the LMIs in the constraints as

$$L_1(\lambda, \mathbf{u}) = \begin{bmatrix} \Lambda(\lambda) & M(\lambda, \mathbf{u}) \\ \star & \nu_1(\lambda) - \gamma_1 \end{bmatrix}$$

and

$$L_2(\mathbf{u}) = \begin{bmatrix} -I_N & \mathbf{Q}^{1/2}\mathbf{u} \\ \star & \delta(\mathbf{u}) - \gamma_2 \end{bmatrix}$$

Hence, by putting everything together we arrive at

$$\begin{aligned} \min_{\mathbf{u},\lambda,\gamma_1,\gamma_2} \quad & \gamma_1 + \gamma_2 \\ \text{s.t.} \quad & \lambda \geq 0, \\ & \mathbf{F}x + \mathbf{G}\mathbf{u} \leq \underline{\mathbf{h}}(\mathbf{w}), \\ & L_1(\lambda, \mathbf{u}) \preccurlyeq 0, \\ & L_2(\mathbf{u}) \preccurlyeq 0 \end{aligned}$$

which corresponds to the formulation in the theorem statement, and with this we conclude the proof. □

Remark 8 (Uncertainty set). The uncertainty set (4-2) is not necessarily uniform in time as it is a ball on Nn_w-dimensional space, i.e, not all $\bar{w}_k$ components of $\bar{\mathbf{w}}$ need to be distanced equally from w_k. For instance, consider the case when $P = I_{Nn_w}$; then we have that

$$\mathfrak{B}_{\varrho, I_{Nn_w}}(\mathbf{w}) = \left\{ \bar{\mathbf{w}} : \sum_{k=1}^{N} \|w_k - \bar{w}_k\|^2 \leq \varrho^2 \right\}$$

Therefore, our proposed uncertainty set includes disturbances with similar measures of energy to $\mathbf{w}$. Therefore, the program (4-4) might try to robustify against variations of $\mathbf{w}$ that have all their energy concentrated on a single time instance, and this in turn might make our proposed solution too conservative. Other similar approaches [21] try to mitigate this by considering robustifying against sets such as

$$\varrho^2 \geq \max_k \|w_k - \bar{w}_k\|^2$$

where each realization is bounded uniformly in time. However, a set like the above still introduces conservativeness, since it introduces multiple quadratic inequalities as constraints, for which the inexact S-Lemma is used [6]. On the other hand, our approach uses the exact version of the S-Lemma, since only one quadratic inequality is involved in the constraints, which allows for an exact reformulation.

Remark 9 (Choice of metric). The disturbances considered in our uncertainty set (4-2) are greatly influenced by the choice of the underlying distance metric P. An obvious choice would be to choose it as the identity matrix which, as mentioned before, would consider disturbances that are close in energy to the measured trajectory $\mathbf{w}$. One more interesting choice might be to pick $P = \mathbf{E}^\intercal \mathbf{Q_x} \mathbf{E}$, which can be linked to the cost contribution of $\mathbf{w}$. Furthermore, if information regarding the disturbance dynamics is available, it could potentially be used in P; for instance, P could be linked to the variance/confidence propagation of $\mathbf{w}$ in time. In conclusion, the choice of P is not trivial at all as it can greatly affect the end result, and provides an interesting research direction for future work.

4-2 Approximating with Inverse Optimization

The non-causal policy (4-4) can be expressed in the standard form

$$\begin{aligned} \pi^{\text{nc-rmpc}}(x, \mathbf{w}) = \arg\min_{u} \ & \mathrm{Q}^{\text{nc-rmpc}}(x, \mathbf{w}, u) \\ \text{s.t. } \ & u \in \mathcal{U}^{\text{nc-rmpc}}(x, \mathbf{w}) \end{aligned} \tag{4-6}$$

with $\mathcal{U}^{\text{nc-rmpc}}(x, \mathbf{w})$ and $\mathrm{Q}^{\text{nc-rmpc}}(x, \mathbf{w}, u)$ are defined accordingly. The procedure to approximate (4-6) with Inverse Optimization is exactly the same as with the non-robust disturbance-aware MPC of Section 3-3 and it outlined by Algorithm 1. The only difference is in the expert policy used, where now policy (4-6) should be used instead of (3-10).

One key difference with (3-10), is that (4-6) requires solving a semidefinite program at each iteration –instead of a quadratic one– so we can expect greater computational improvement. This of course comes with a cost, as now the quality of the approximation will be potentially worse. However, as it will be seen in Section 5, the performance of the resulting approximation can be promising.

Chapter 5

Numerical experiments

To showcase our proposal on a practical level, we will use two linear systems as benchmarks. To implement our work we used MATLAB, the YALMIP toolbox [22], and the MOSEK optimizer [25]. Each benchmark will be used in two scenarios: in the first one, the disturbances used for training the Inverse Optimization policy will have the same dynamics/distribution during evaluation, whereas in the second scenario, the disturbances will undergo a distribution shift between training and evaluation. The first scenario will be useful for demonstrating the Inverse Optimization approximation of the non-robust MPC, while the second will showcase the validity of the robust MPC approximation.

Since our disturbances will have a stochastic component and different initial conditions will be used over many trials, a good way to evaluate our results is to make use of histograms of the running costs $c(x, u)$, such the one in Figure 5-1. For the sake of simplicity and in order for this to be an equivalent criterion to the control objective mentioned in the Introduction, we will choose $Q_f = Q_x$.

5-1 Three-dimensional system

Here, we will use an unstable linear system found in [21] as a benchmark, that has the following dynamics:

$$\begin{gathered} A = \begin{bmatrix} 2.938 & -0.7345 & 0.25 \\ 4 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0.25 \\ 0 \\ 0 \end{bmatrix} \\ E = \begin{bmatrix} 0.0625 \\ 0 \\ 0 \end{bmatrix}, \; C^{\mathsf{T}} = \begin{bmatrix} -0.2072 \\ 0.04141 \\ 0.07256 \end{bmatrix} \end{gathered} \tag{5-1}$$

The initial conditions of this system are sampled from a normal distribution as $x_0 \sim \mathcal{N}(0, 0.2I_3)$.

We subjected system (1-1, 5-1) to a disturbance model with the following dynamics:

$$\xi_{t+1} = \Phi \xi_t + \zeta_{t+1} \tag{5-2a}$$

$$w_t = \Psi \xi_t \tag{5-2b}$$

with $\zeta_t \sim \mathcal{N}(0, \Sigma_\zeta)$, and

$$\Phi = \begin{bmatrix} 0 & 0.99 & 0 \\ -0.99 & 1.5 & 0 \\ -0.5 & 0.5 & 0.99 \end{bmatrix}, \quad \Psi^\mathsf{T} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix},$$

$$\Sigma_\zeta = \begin{bmatrix} 0.05 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

The disturbances have their initial conditions sample a normal distribution, specifically $\xi_0 \sim \mathcal{N}(0, 0.1 I_3)$.

The control objective is for the output of our system y_t to track a constant reference $r = 1$ with the costs $Q_y = Q_f = 100$ and $Q_u = 1$. We use an MPC horizon of $N = 10$. Furthermore, we will only impose constraints on the control input such that $\mathcal{U} = \{u : |u| \leq 1\}$.

5-1-1 Approximating the non-causal MPC with Inverse Optimization

In order to use the Inverse Optimization framework we first need to define a feature map. Since we want to track a constant reference, we can add a constant bias term to our features. Additionally, we use the past 3 disturbances as features, since that's the rank of the disturbance dynamics. Therefore, the features we will use will be $s_t := (x_t, 1, \mathbf{w}_{t-2:t})$.

The expert policy we will use is that of the non-causal MPC, i.e, (3-10). To collect data we will run an MPC policy that is oblivious to the disturbances for 10 different initial conditions, to create 10 trajectories of length $T = 50$ each. Then, after computing features for each dataset separately, we concatenate everything into a single dataset and we feed it to Algorithm 1.

We will compare the following three policies:

MPC (obl) An MPC that is oblivious to the disturbance dynamics, akin to that described in (3-7). In other words, this policy can only measure x_t and has knowledge of A, B, E, C.

MPC (dst) An MPC with access to the disturbance dynamics model but not their stochastic inputs. This policy can measure both x_t and ξ_t, and has knowledge of A, B, E, C, Φ, Ψ. This is the best that can be achieved without resorting to non-causal approaches that also know the stochastics that are going to occur.

IO-MPC This is the result of applying Algorithm 1 to the dataset resulting from MPC (obl). During training there is only access to x_t and A, B, E, C are used.

The cost histograms obtained by running each policy 1000 times are shown in Figure 5-1 for the whole trajectory and for the steady state. As we can see that, while IO-MPC does not

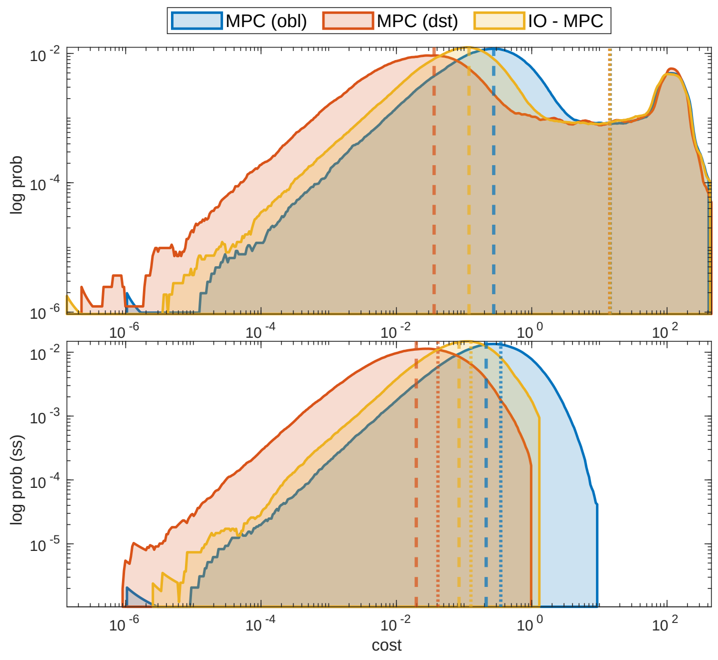

Figure 5-1: Cost histogram in log-log scale of the stage cost for the experiments described in Section 5-1-1. The top plot showcases the histogram for the whole experiment duration, whereas the bottom plot only after the steady state. The dashed line represents the mean value and the dotted line represents the median.

fully recover the performance of the disturbance-aware policy MPC (dst), it offers a significant improvement in the mean over the oblivious policy MPC (obl). The reason that MPC (dst) still outperforms the IO-MPC is because it has access to the full state ξ_t, which offers more accurate predictions than just the disturbance's output w_t.

5-1-2 Approximating the non-causal RMPC with Inverse Optimization

Our experimental setup remains the same, with the same disturbance dynamics as before being used while collecting data and training. However, during evaluation there is a distribution shift in the disturbance by adding a constant bias to w_t, specifically

$$w_t = \Psi \xi_t + 1.$$

In this section we will also evaluate the Inverse Optimization approximation of robust MPC policies like (4-4). Furthermore, we want to investigate the effect of different choices of the metric P used to define the uncertainty set (4-2). We will focus on two possible metrics: the identity matrix, and $\mathbf{E}^\intercal \mathbf{Q_x} \mathbf{E}$. Therefore, in this section we will evaluate the following policies:

MPC (obl) Same as before.

MPC (p-dst) An MPC with partial access to the disturbance dynamics model but not their stochastic inputs. This policy can measure both x_t and ξ_t, and has knowledge of A, B, E, C, Φ, Ψ, and is exactly the same as MPC (dst) of the previous section.

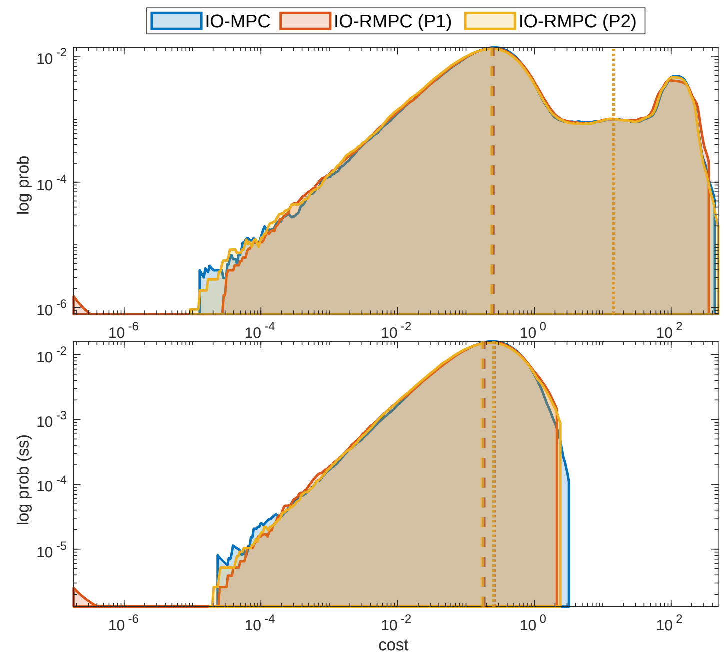

Figure 5-2: Cost histogram in log-log scale of the stage cost for each Inverse Optimization approximation as described in Section 5-1-2.

MPC (f-dst) Similar to MPC (p-dst), with the exception that it is also aware of the additional bias added to w_t.

IO-MPC Same as before.

IO-RMPC (P1) A robust MPC of the form (4-4), trained the same way as IO-MPC, with access to the same knowledge. The metric is chosen to be $P = \mathbf{E}^\intercal \mathbf{Q_x} \mathbf{E}$ and the uncertainty ball radius is $\varrho^2 = 0.0005$.

IO-RMPC (P2) Same as IO-RMPC (P1), but equipped with the identity metric and $\varrho^2 = 1$.

In Figure 5-2 we compare the three Inverse Optimization approximations. We can see that their cost distributions are largely the same, with the robust versions offering slightly tighter variance. This is partly due to the ball radius being small in both cases, which means that only disturbance trajectories very close to the original are considered in the robust versions. During experiments, using significantly larger radii led to worse performance than IO-MPC. However, upon inspecting the cost distribution of MPC (p-dst) in Figure 5-3, we can see that even the non-robust policy IO-MPC outperforms MPC (p-dst) when faced with a distribution shift. This is very interesting, as it seems – at least in this simulation example – that even the non-robust policy has some inherent robustness. Overfitting to the noise is unlikely as these results were obtained over 1000 trials which were not used during training. The policy MPC (f-dst), having full knowledge of the new disturbance dynamics, outperforms all three data-driven approaches by a good margin, but that is to be expected as the bias term in w_t was not present during training.

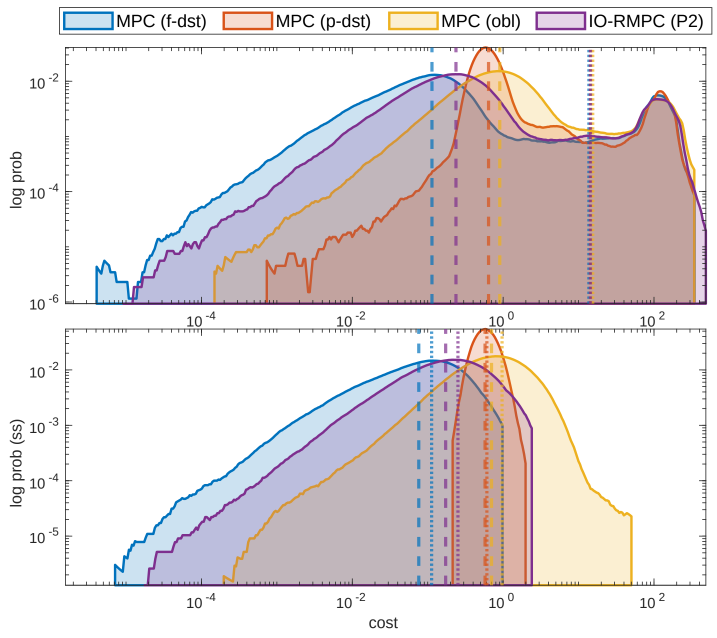

Figure 5-3: Cost histogram in log-log scale of the stage cost for the best Inverse Optimization approximation along with the MPC policies as described in Section 5-1-2.

5-2 Fighter jet

Here we will use the (unstable) linearized dynamics of a fighter jet [27] that have been discretized with a sample time of 0.035 s. The system matrices are:

$$A = \begin{bmatrix} 0.9991 & -1.3736 & -0.6730 & -1.1226 & 0.3420 & -0.2069 \\ 0.0000 & 0.9422 & 0.0319 & -0.0000 & -0.0166 & 0.0091 \\ 0.0004 & 0.3795 & 0.9184 & -0.0002 & -0.6518 & 0.4612 \\ 0.0000 & 0.0068 & 0.0335 & 1.0000 & -0.0136 & 0.0096 \\ 0 & 0 & 0 & 0 & 0.3499 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.3499 \end{bmatrix}, E = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{5-3}$$
$$B^{\mathsf{T}} = \begin{bmatrix} 0.1457 & -0.0072 & -0.4085 & -0.0052 & 0.6501 & 0 \\ -0.0819 & 0.0035 & 0.2893 & 0.0037 & 0 & 0.6501 \end{bmatrix}, C = I_6$$

The initial conditions are sampled from a normal distribution $x_0 \sim \mathcal{N}(0, 0.1I_6)$.

We subject our system to the following disturbances

$$w_{t+1} = \begin{bmatrix} 0.5\sin(\omega t + \phi) \\ 0.01 \end{bmatrix} + \begin{bmatrix} 0.01 & 0 \\ 0 & 0.001 \end{bmatrix} v_{t+1}$$

with $\omega = 4.488\,\text{rad/s}$ and $v_t \sim \mathcal{N}(0, I_2)$. The phase ϕ is also sampled randomly from a uniform distribution on $[0, \pi/2]$.

The control objective is stabilization to the origin with costs

$$Q_f = Q_x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; Q_u = I_2$$

and horizon $N = 20$. Furthermore, we have the following state constraints

$$G_x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; h_x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and the following input constraints

$$G_u = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \; h_u = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 3 \end{bmatrix}$$

As our disturbances have a stochastic component, we will soften our state constraints to avoid infeasibility issues by adding some slack, that is penalized quadratically in the objective function with a factor of 10^9.

Approximating the non-causal MPC with Inverse Optimization

Our experimental procedure here is very similar to the one described in Section 5-1-1. Since a sine wave can be described by a second-order system, it makes sense to have $\mathbf{w}_{t-1:t}$ in our features. Furthermore, as there is also bias in our disturbances, we should add a bias term to our features as well. This leads us to the feature map $s_t := (x_t, 1, \mathbf{w}_{t-1:t})$.

As before, the expert policy we will use is that of the non-causal MPC, i.e, (3-10). We will use again 10 trajectories for training, but this time they will contain 61 samples each. Then, after computing features for each dataset separately, we concatenate everything into a single dataset and we feed it to Algorithm 1.

We will compare the same policies as before: **MPC (obl)**, which is oblivious to the disturbances; **MPC (dst)**, which has knowledge of the deterministic part of the disturbances; and **IO-MPC**, which is the result of applying Algorithm 1 to our dataset.

We ran each policy for 100 trials. In Figure 5-4 we can see that the IO-MPC almost completely recovers the performance of MPC (dst), both in terms of mean and of variance. This significantly improved result is likely due to the fact that the stochastics do not affect the disturbance dynamics, but only their output, and as such measurements of w_t yield accurate predictions. Furthermore, in Figure 5-5, we can see the evolution of the state trajectories under each policy. We can see that the constraints were upheld despite the stochastics, but this is to be expected as even MPC (obl) managed to contain the first state in the constraint set. This is due to the fact that the noise does not enter the first state directly and therefore the control can compensate. Furthermore, we can see that the state trajectories induced by IO-MPC are very close to those generated by MPC (dst), but this is to be expected as their cost distributions also line up.

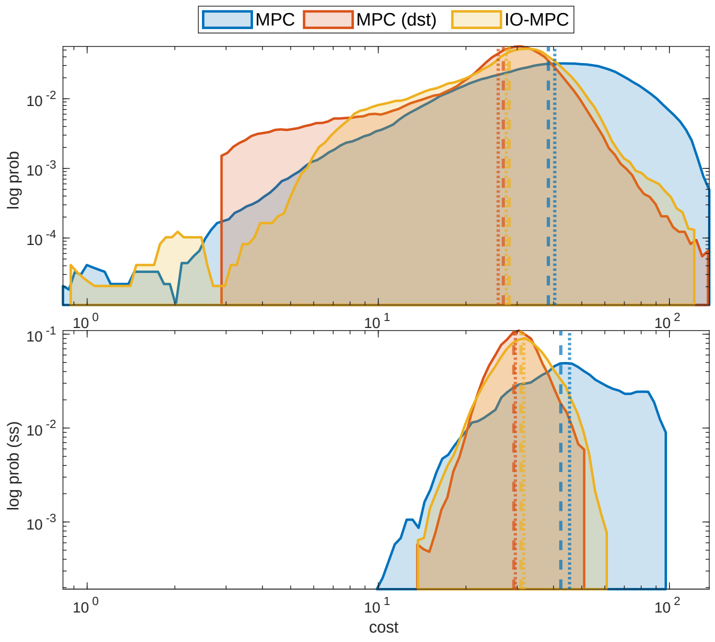

Figure 5-4: Cost histogram in log-log scale of the stage cost for the experiments described in Section 5-2.

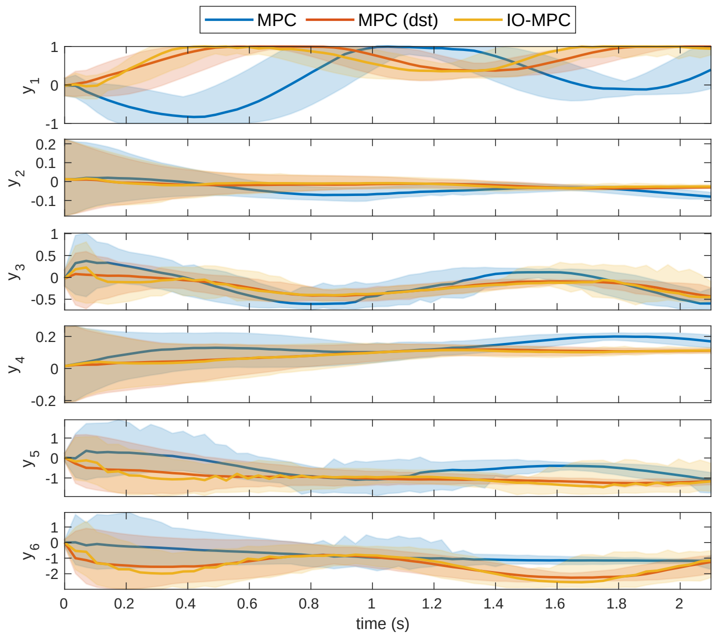

Figure 5-5: State trajectories after for the experiments described in Section 5-2. The solid lines refer to the median values at each timestep and the tubes correspond to the minimum and maximum values attained (pointwise).

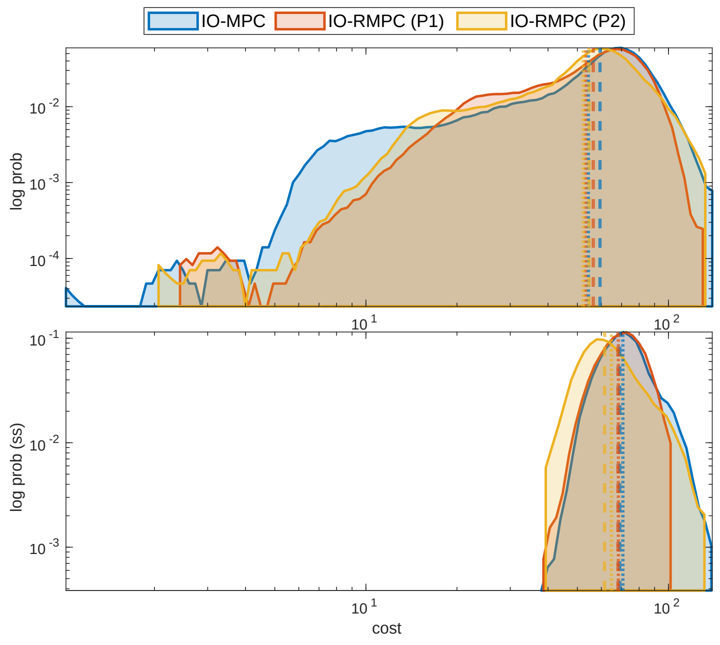

Figure 5-6: Cost histogram in log-log scale of the stage cost for each Inverse Optimization approximation as described in Section 5-2-1.

5-2-1 Approximating the non-causal RMPC with Inverse Optimization

As in Section 5-1-2, here we will also evaluate the Inverse Optimization approximation of the robust MPC (4-4). The data used in training is exactly the same as in Section 5-2. However, during evaluation a bias is added such that the applied disturbances are

$$w_{t+1} = \begin{bmatrix} 0.5\sin(\omega t + \phi) + 0.1 \\ 0.015 \end{bmatrix} + \begin{bmatrix} 0.01 & 0 \\ 0 & 0.001 \end{bmatrix} v_{t+1}$$

In the following we will test the following policies: **MPC (obl)**, which is oblivious to the disturbances; **MPC (p-dst)**, which is the same as MPC (dst) of Section 5-2; **MPC (f-dst)**, which is also aware of the additional bias; **IO-MPC**, which is the same as IO-MPC of Section 5-2; **IO-RMPC (P1)**, which is an approximation of the robust MPC with metric $P = \mathbf{E}^\intercal \mathbf{Q_x} \mathbf{E}$ and an uncertainty radius of $\varrho^2 = 10$; and **IO-RMPC (P2)**, which uses the identity metric and the uncertainty radius $\varrho^2 = 0.0001$.

As in Section 5-2, we ran everything for 100 trials. In Figure 5-6 we can see the cost distributions for each Inverse Optimization based policy. We can see that the identity metric IO-RMPC (P2) outperforms the other two in the mean during the steady-state. However, IO-RMPC (P1) has a tighter variance than the others, with its mean being really close the IO-MPC. When compared to the MPC policies (Figure 5-7), we can see that IO-RMPC (P2) recovers the average performance of MPC (f-dst), which its variance however exceeding that of MPC (p-dst). Overall the performance of all controllers has worsened due to the additional bias.

Furthermore, in Figure 5-8 we can observe the trajectories generated by each Inverse Optimization policy. All three policies managed to maintain the first state in the constraint

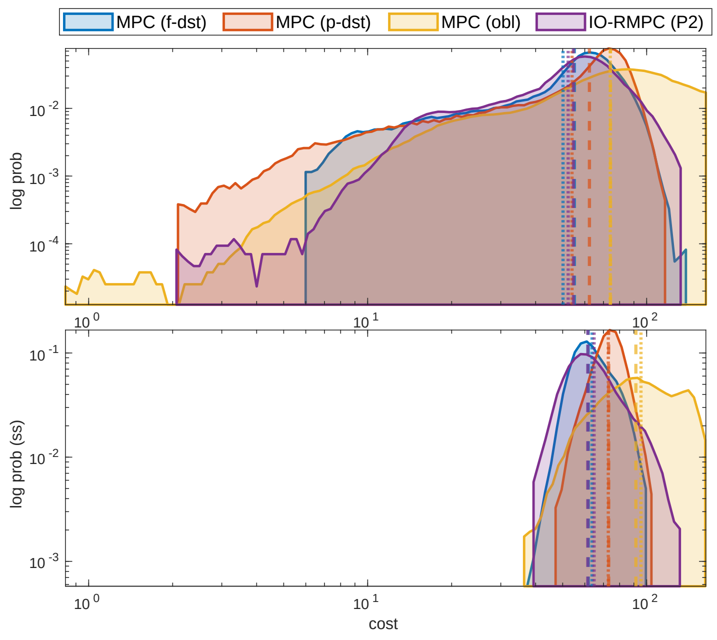

Figure 5-7: Cost histogram in log-log scale of the stage cost for the best Inverse Optimization approximation along with the MPC policies as described in Section 5-2-1.

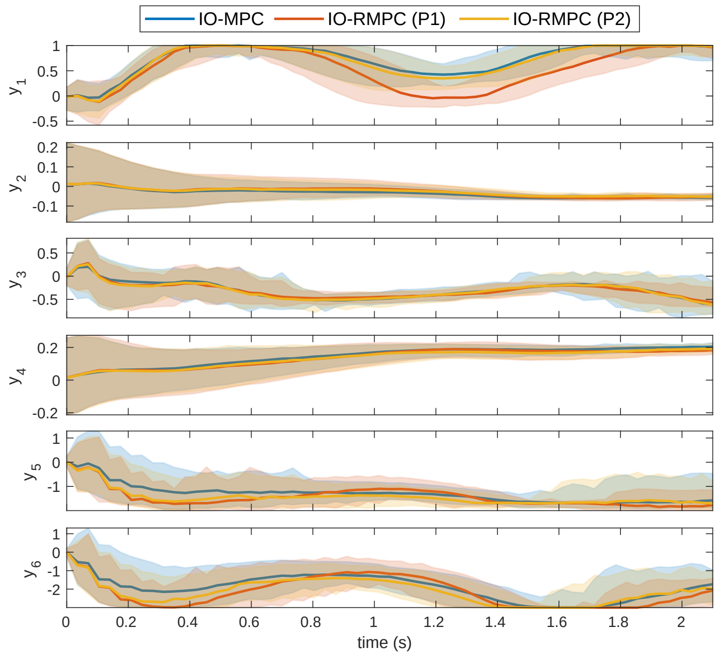

Figure 5-8: State trajectories generated by each Inverse Optimization approximation as described in Section 5-2-1.

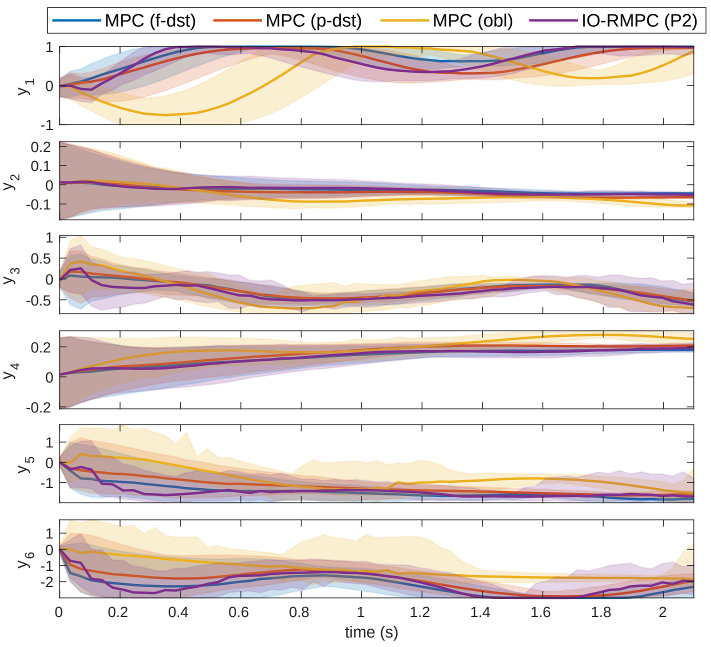

Figure 5-9: State trajectories generated by the best Inverse Optimization approximation along with the MPC policies as described in Section 5-2-1.

set. The policy IO-RMPC (P1) is the most conservative, as around the 1-1.4 second mark we can see that the first state is the furthest from the constraint boundary. This showcases that during run-time there is no knowledge of the disturbances that are going to occur in the future, the learned policy has learned managed to map the cost associated with constraint satisfaction given $\mathbf{w}_{t+1}$ to the features $\mathbf{w}_{t-1:t}$. Finally, when compared to the MPC policies, we can see in Figure 5-9 that IO-RMPC (P2) induces trajectories that are really close to the optimal ones generated by MPC (f-dst).

Chapter 6

Conclusions and future works

To conclude, in this work we presented some potential uses for the Inverse Optimization framework that not only approximate existing policies, but can also aid in the creation of new data-driven ones that mitigate the effect of unknown disturbances. However, as this is a new framework, further research is warranted. Potential future directions can include: (i) approximating controllers in real-time with online optimization; (ii) regularizing θ to avoid accidental over-fitting when stochastics are involved; (iii) investigate on how Inverse Optimization can work with nonlinear systems by including for instance nonlinear features; (iv) explore how non-causal constraints could be approximated with causal ones by mapping future disturbance realizations to features containing past disturbance instances. Furthermore, we developed a novel min-max MPC scheme, which warrants further research on its own, with possible topics including: (i) a game-theoretic analysis to investigate whether there is a Nash equilibrium; (ii) further investigation in the choice and impact of metric; (iii) state-dependent uncertainty sets, with a potential application on nonlinear systems.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 1, 2004.

[2] Naman Agarwal, Brian Bullins, Elad Hazan, Sham M. Kakade, and Karan Singh. Online Control with Adversarial Disturbances. *arXiv:1902.08721 [cs, math, stat]*, February 2019.

[3] S. A. Akhtar, A. S. Kolarijani, and P. M. Esfahani. Learning for Control: An Inverse Optimization Approach. *IEEE Control Systems Letters*, pages 1–1, 2021.

[4] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, August 2021.

[5] Karl J. Åström and Björn Wittenmark. On self tuning regulators. *Automatica*, 9(2):185–199, March 1973.

[6] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* SIAM, 1994.

[7] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, UK ; New York, 2004.

[8] S.J. Bradtke, B.E. Ydstie, and A.G. Barto. Adaptive linear quadratic control using policy iteration. In *Proceedings of 1994 American Control Conference - ACC '94*, volume 3, pages 3475–3479 vol.3, June 1994.

[9] Alon Cohen, Tomer Koren, and Yishay Mansour. Learning Linear-Quadratic Regulators Efficiently with only $\sqrt{T}$ Regret. *arXiv:1902.06223 [cs, stat]*, February 2019.

[10] Jeremy Coulson, John Lygeros, and Florian Dörfler. Data-Enabled Predictive Control: In the Shallows of the DeePC. *arXiv:1811.05890 [math]*, March 2019.

[11] Jeremy Coulson, John Lygeros, and Florian Dörfler. Distributionally Robust Chance Constrained Data-enabled Predictive Control. *arXiv:2006.01702 [math]*, June 2020.

[12] Claudio De Persis and Pietro Tesi. Formulas for Data-Driven Control: Stabilization, Optimality, and Robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, March 2020.

[13] Claudio De Persis and Pietro Tesi. Low-complexity Learning of Linear Quadratic Regulators from Noisy Data. *arXiv:2005.01082 [cs, eess]*, May 2020.

[14] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the Sample Complexity of the Linear Quadratic Regulator. *arXiv:1710.01688 [cs, math, stat]*, December 2018.

[15] Dylan J. Foster and Max Simchowitz. Logarithmic Regret for Adversarial Online Control. *arXiv:2003.00189 [cs, math, stat]*, June 2020.

[16] Gautam Goel and Babak Hassibi. Regret-optimal measurement-feedback control. *arXiv:2011.12785 [cs, eess, math]*, November 2020.

[17] Elad Hazan, Sham Kakade, and Karan Singh. The Nonstochastic Control Problem. In *Algorithmic Learning Theory*, pages 408–421. PMLR, January 2020.

[18] Bahare Kiumarsi, Frank L. Lewis, Hamidreza Modares, Ali Karimpour, and Mohammad-Bagher Naghibi-Sistani. Reinforcement -learning for optimal tracking control of linear discrete-time systems with unknown dynamics. *Automatica*, 50(4):1167–1175, April 2014.

[19] Sahin Lale, Kamyar Azizzadenesheli, Babak Hassibi, and Anima Anandkumar. Explore More and Improve Regret in Linear Quadratic Regulators. *arXiv:2007.12291 [cs, math, stat]*, July 2020.

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[21] Johan Löfberg. *Minimax Approaches to Robust Model Predictive Control.* Number 812 in Linköping Studies in Science and Technology Dissertations. Univ, Linköping, 2003.

[22] Johan Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[24] Peyman Mohajerin Esfahani, Soroosh Shafieezadeh-Abadeh, Grani A. Hanasusanto, and Daniel Kuhn. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167(1):191–234, January 2018.

[25] MOSEK-ApS. MOSEK optimization toolbox for MATLAB. *User's Guide and Reference Manual, Version*, 4, 2019.

[26] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

[27] M. Safonov, A. Laub, and G. Hartmann. Feedback properties of multivariable systems: The role and use of the return difference matrix. *IEEE Transactions on Automatic Control*, 26(1):47–65, February 1981.

[28] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, and Adrian Bolton. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[29] Max Simchowitz and Dylan J. Foster. Naive Exploration is Optimal for Online LQR. *arXiv:2001.09576 [cs, math, stat]*, June 2020.

[30] Umar Syed and Robert E. Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in Neural Information Processing Systems*, pages 1449–1456, 2008.

[31] Henk J. van Waarde, M. Kanat Camlibel, and Mehran Mesbahi. From noisy data to feedback controllers: Non-conservative design via a matrix S-lemma. *arXiv:2006.00870 [math]*, June 2020.

[32] Michel Verhaegen and Vincent Verdult. *Filtering and System Identification: A Least Squares Approach.* Cambridge University Press, Cambridge, 2007.

[33] Christopher John Cornish Hellaby Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[34] Jan C. Willems, Paolo Rapisarda, Ivan Markovsky, and Bart L. M. De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, April 2005.

[35] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.