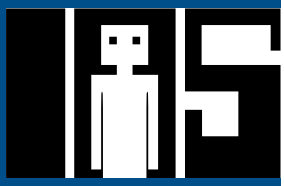
Distributionally Robust Optimization for Optimal Control

Verteilungsrobuste Optimierung für Optimalregelung
Master thesis by Tim Dorau
Date of submission: October 20, 2020

1. Review: M.Sc. Hany Abdulsamad
2. Review: M.Sc. Boris Belousov
3. Review: Prof. Jan Peters
4. Review: Prof. Debora Clever

TECHNISCHE
UNIVERSITÄT
DARMSTADT



Master Thesis

TECHNISCHE UNIVERSITÄT DARMSTADT

Tim Dorau | 1918997

Topic:
Distributionally Robust Optimization for Optimal Control

Verteilungsrobuste Optimierung für Optimalregelung

Trajectory optimization approaches are powerful techniques for optimal control of dynamical systems. These approaches rely on the existence of differentiable dynamics models, that are often hard to obtain analytically. Recent research in the area of data-driven trust-region trajectory optimization has proven fruitful for iteratively learning the dynamics and solving the local optimal control problem in an online fashion. The resulting statistical models of the learned dynamics are however often regarded as "true" during the optimization phase, which may lead to overconfident control updates.

Distributionally robust optimization (DRO) is an approach which explicitly takes into account ambiguity in the probability distribution underlying a stochastic optimization problem. Instead of assuming a given distribution, a worst-case problem is solved over an ambiguity set of distributions typically defined as containing all distributions close to a nominal one in terms of some statistical distance measure. Work in recent years has shown that for appropriately chosen ambiguity sets, the problem remains computationally tractable while yielding powerful performance guarantees.

In this thesis, we want to take advantage of Bayesian methods for model learning and incorporate them in an optimal control framework that considers a trust-region around the learned dynamics and robustly optimizes for the worst-case instance given the statistical model. In this context, we want to investigate the applicability of the DRO approach.

We think that such an approach would result in a conservative but robust iterative approach for control of critical systems that can avoid catastrophic control signal updates.

Start date: 27. April 2020
Workload: 900 h
Supervisors: Hany Abdulsamad, M. Sc., Boris Belousov, M. Sc.

[redacted] Dr. rer. nat. Debora Clever
Department of Mechanical Engineering

Prof. Jan Peters, Ph.D.
Department of Computer Science

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Tim Dorau, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 20. Oktober 2020

T. Dorau

Abstract

Optimal control techniques often rely on the validity of a dynamical system model. Since good models of complex systems are difficult to obtain analytically, data-driven methods which learn a stochastic model based on trajectories of the real system play an important role. A model learned from limited data has some inherent ambiguity which should be considered during the optimization. Distributionally robust optimization provides a principled approach to account for ambiguity in the probability distributions underlying a stochastic optimization problem by optimizing expected performance under a worst-case distribution. In this work, distributional robustness is applied to simple optimal control problems.

A first approach aims to find a controller for a linear system with quadratic cost which is robust to arbitrarily distributed additive noise. Based on the assumption that some noise samples are available, we use recent techniques from distributionally robust optimization based on Wasserstein distances to reformulate the robust problem as a convex optimization problem. Comparison against the standard LQR controller shows possible improved performance when evaluated under the worst-case noise but also dependence on hyperparameters.

Our second approach is based on data-driven trajectory optimization with linearized dynamics. We solve the trajectory optimization problem with a relative entropy bound on the policy, while also considering ambiguity in the distribution of model parameters. As expected, the resulting policy improves performance under the worst-case parameter distribution while being slightly conservative and lowering performance for the nominal distribution. We also discuss some problems related to the solvability and existence of this worst-case distribution.

Zusammenfassung

Techniken der Optimalregelung beruhen oft auf der Gültigkeit eines dynamischen Systemmodells. Da gute Modelle komplexer Systeme analytisch schwer zu erhalten sind, spielen datenbasierte Methoden, die ein stochastisches Modell auf der Grundlage von Trajektorien des realen Systems erlernen, eine wichtige Rolle. Ein Modell, welches basierend auf begrenzten Daten gelernt wird, hat eine gewisse Unsicherheit, die bei der Optimierung berücksichtigt werden sollte. Die verteilungsrobuste Optimierung bietet einen prinzipiellen Ansatz zur Berücksichtigung der Unsicherheit in den Wahrscheinlichkeitsverteilungen, die einem stochastischen Optimierungsproblem zugrunde liegen, indem der Erwartungswert eines Gütemaßes unter einer Worst-Case-Verteilung optimiert wird. In dieser Arbeit wird die verteilungsrobuste Optimierung auf einfache Optimalregelungsprobleme angewendet.

Ein erster Ansatz zielt darauf ab, einen Regler für ein lineares System mit quadratischen Kosten zu finden, der robust gegenüber beliebig verteilter additiver Störungen ist. Basierend auf der Annahme, dass einige Messwerte der Störung verfügbar sind, verwenden wir moderne Techniken der verteilungsrobusten Optimierung auf der Basis von Wasserstein-Distanzen, um das robuste Problem als konvexes Optimierungsproblem zu formulieren. Der Vergleich mit dem klassischen LQR-Regler zeigt mögliche Verbesserungen unter der Worst-Case-Störung, aber auch eine starke Abhängigkeit von Hyperparametern.

Unser zweiter Ansatz basiert auf einer datenbasierten Trajektorienoptimierung mit linearisierter Systemdynamik. Wir lösen das Trajektorienoptimierungsproblem mit einer Nebenbedingung für die relative Entropie des stochastischen Regelgesetzes, wobei wir auch die Unsicherheit in der Verteilung der Modellparameter berücksichtigen. Wie erwartet verbessert der resultierende Regler die Leistung unter der Worst-Case-Parameterverteilung, während er leicht konservativ ist und die Leistung für die nominale Verteilung verschlechtert. Wir diskutieren auch einige Probleme im Zusammenhang mit der Lösbarkeit und Existenz der Worst-Case-Verteilung.

Acknowledgments

First of all, I would like to thank Hany Abdulsamad and Boris Belousov for supervising this thesis, guiding me with their many ideas and encouraging me to pursue my own. I learned a lot from their detailed explanations and during the discussions for which they always took time.

To Prof. Jan Peters, I am thankful for the great research environment at IAS, offering such a variety of interesting topics and organizing countless fascinating talks in the Oberseminar.

Lastly, I also want to thank Prof. Debora Clever for agreeing to co-supervise my thesis and offering her support.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

BFGS Broyden–Fletcher–Goldfarb–Shanno algorithm

DDP Differential Dynamic Programming

DRO Distributionally Robust Optimization

GPS Guided Policy Search

ILQG Iterative Linear Quadratic Gaussian

KL Kullback-Leibler divergence

LQR Linear Quadratic Regulator

MDP Markov Decision Process

QCQP Quadratically Constrained Quadratic Program

RL Reinforcement Learning

SDP Semidefinite Program

SGD Stochastic Gradient Descent

1. Introduction

Control of complex systems in uncertain environments is a difficult challenge with many applications such as robotics and autonomous driving. A popular paradigm for addressing these problems is reinforcement learning, an umbrella term for approaches in which an agent learns to achieve a goal by interacting with its environment and getting feedback about the choices made [1]. While applications like playing games at a super-human level [2] have received significant attention, there are additional challenges in solving real world physical control tasks. Random exploration in search of a suitable control strategy can lead to unsafe and unpredictable behavior. In many cases, as for example if the system is a self-driving car or a robot working close to humans, this behavior clearly needs to be avoided. Another important factor is that interaction with a real system takes significantly more time and resources than with a simulated one.

One method to approach these difficulties is by building a computational model of the real system and training the agent in simulation. This model can be derived completely from scratch, or by collecting a limited amount of samples from the real system and fitting a suitable model to them. Given the complexity of the real world, including changing environmental conditions, wear and tear of the system and hard-to-model physical effects like friction [3], the model will never be perfect. An imperfect model in turn can strongly influence performance of the learned control strategy on the real system [4]. For the model-based approach to work well, it is therefore essential to account for the possible model misspecification.

Motivated by the above, in this thesis we look at an approach from the optimization community for including uncertainty in a decision problem and try to apply it to simple control problems. We start by introducing some terminology and key ideas, and in the following two chapters lay out two somewhat distinct approaches. For each, we refer to some more closely related work and discuss results and issues regarding implementation.

1.1. Optimal Control

Broadly speaking, the field of optimal control is concerned with designing controllers for a system evolving over time which are optimal with respect to some performance measure. Ideas from optimal control are at the basis of much of modern control theory such as reinforcement learning [1], model predictive control [5] and trajectory optimization [6].

In general, the system dynamics can be stochastic, which means at time t, they are described by $\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a})$, a probability distribution over the next state $\mathbf{s}'$ given the current state $\mathbf{s}$ and the control input or action $\mathbf{a}$. This formulation already includes the typical assumption that the system possesses the Markov property and the next state depends only on the current state and action, not their past values. The aim of stochastic optimal control is then finding a policy $\pi_t(\mathbf{a}|\mathbf{s})$ which is a distribution over actions given the current state, such that it maximizes a measure of reward $R_t(\mathbf{s},\mathbf{a})$ summed up over the possibly infinite time horizon considered. The optimization can equivalently be formulated as minimizing a measure of cost instead. The type of stochastic control process described here is generally called a Markov decision process (MDP).

One of the main tools of optimal control is dynamic programming. This solution method is based on the insight that the problem of optimizing over the whole time horizon can be split into solving an optimization at each timestep going backwards in time. The value of a state is characterized by the so-called state value function $V_t(\mathbf{s})$ as a measure of expected reward under a given policy starting from state $\mathbf{s}$, while the value of taking action $\mathbf{a}$ and then following the given policy is captured by the state-action value function $Q_t(\mathbf{s}, \mathbf{a})$. Dynamic programming uses a connection between the two functions known as Bellman's equations [7], where

$$V_t(\mathbf{s}) = \mathbb{E}_\pi[Q_t(\mathbf{s}, \mathbf{a})]$$
$$Q_t(\mathbf{s}, \mathbf{a}) = R_t(\mathbf{s}, \mathbf{a}) + \mathbb{E}_{\mathcal{P}_t}[V_{t+1}(\mathbf{s}')].$$

These two value functions can be computed recursively and the optimization is reduced to finding the policy which maximizes $Q_t(\mathbf{s}, \mathbf{a})$.

1.1.1. Linear Quadratic Regulator

One of the fundamental problems in optimal control is that of controlling a linear system with dynamics given as

$$\mathcal{P}_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = \mathcal{N}(\mathbf{s}'|\mathbf{A}_t\mathbf{s} + \mathbf{B}_t\mathbf{a}, \mathbf{\Sigma}_{s',t})$$

in which $\mathbf{A}_t$ and $\mathbf{B}_t$ are the system and input matrices and $\mathbf{\Sigma}_{s',t}$ is the covariance matrix of the Gaussian noise. The problem of minimizing an expected quadratic cost

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \mathbf{s}_t^\mathsf{T}\mathbf{Q}\mathbf{s}_t + \mathbf{a}_t^\mathsf{T}\mathbf{R}\mathbf{a}_t\right]$$

under the above dynamics can be solved by dynamic programming, yielding a solution for the optimal control law which is linear in the state [8]. The controller parameter is found by solving a so-called Riccati equation. This controller is known as the linear quadratic regulator (LQR). The above problem and many variations of it have been studied extensively in the control literature [9]. Its usefulness lies in being easy to solve while approximating many real world problems in their operating range quite well [10].

Furthermore, the above solution can be used as a building block in iterative methods to find locally optimal controllers for nonlinear systems such as the iterative linear quadratic regulator (ILQR) [11]. For these iterative methods, linear dynamics are used to approximate the nonlinear ones along a given trajectory which allows finding a controller by solving the LQR problem. A new trajectory is then collected by applying the controller to the actual nonlinear system and the whole process is repeated until convergence. Since the linearization is only valid around the current trajectory, the policy update should not be too drastic, as otherwise the updated policy might actually perform worse on the nonlinear system than the previous one.

1.1.2. Robustness

A generally important aspect in control is robustness. While this idea has many interpretations and nuances, it can roughly be understood as coping with misspecification in the system model used to obtain a controller. Under some specified level of uncertainty, the resulting controller should then guarantee a certain level of performance [12, 13]. For the linear dynamics case, misspecification might for example include uncertainty about the matrices $\mathbf{A}_t$, $\mathbf{B}_t$ (which we treat in chapter 3), the noise distribution (which we consider in chapter 2) or even unmodeled effects such as nonlinearities.

1.2. Distributionally Robust Optimization

Distributionally robust optimization (DRO) is a paradigm for optimization under uncertainty which has received significant interest in recent years. To introduce its main idea and the connection to other concepts, we will first consider a simple, deterministic optimization problem

$$\min_{\mathbf{x}\in\mathcal{X}} L(\mathbf{x}, \mathbf{w}),$$

in which the decision maker has to choose decision variables $\mathbf{x}$ from a given set $\mathcal{X}$, such that the loss function $L(\mathbf{x}, \mathbf{w})$, depending on $\mathbf{x}$ and some known parameters $\mathbf{w}$, is minimized. In real world examples, the parameters will typically not be known exactly and the solution to the above problem might be meaningless. Several approaches exist to include this uncertainty in the problem formulation, where an important aspect is the description of uncertainty.

In the paradigm of robust optimization, the parameters are assumed to lie in some set $\mathcal{W}$, typically chosen around a nominal estimate of the parameters. A possible formulation of the optimization problem for taking this into account is then

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{w}\in\mathcal{W}} L(\mathbf{x}, \mathbf{w}).$$

An optimal solution to this minimizes the worst-case loss and thereby gives an upper bound on the incurred loss for all parameters in the given set [14]. Furthermore, the optimal solution must be feasible for all possible parameter values. Robust optimal solutions typically introduce conservatism, meaning they worsen performance under the nominal parameter values [15].

In a sense, robust optimization considers all parameter values from the uncertainty set as equally likely. Stochastic optimization, in contrast, considers parameters which are random variables with a distribution $p(\mathbf{w})$ which is known exactly. A typical formulation might then minimize the expected loss under that distribution

$$\min_{\mathbf{x}\in\mathcal{X}} \mathbb{E}_{\mathbf{w}\sim p}[L(\mathbf{x}, \mathbf{w})].$$

Distributionally robust optimization is a superset of the two approaches to consider uncertainty. It considers parameters which are stochastic but with a probability distribution which is itself uncertain. We can write the problem as

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{p\in\mathcal{P}} \mathbb{E}_{\mathbf{w}\sim p}[L(\mathbf{x}, \mathbf{w})], \tag{1.1}$$

where we again minimize an expected loss, but do so under the worst-case distribution from some set $\mathcal{P}$ of distributions, the so-called ambiguity set. A connection to the other mentioned approaches becomes clear when considering two extreme cases. If the set $\mathcal{P}$ contains merely a single distribution, we recover stochastic optimization, if it contains all distributions supported on $\mathcal{W}$, we get robust optimization [16]. The idea of introducing an ambiguity set of distributions is that for stochastic optimization, the parameter distribution is generally estimated from a limited amount of data. Relying too much on the estimated distribution can then again lead to unexpectedly bad performance under the real underlying distribution. With this idea of partial knowledge about the distribution, the distributionally robust approach falls in between the robust approach assuming zero distributional knowledge and the stochastic optimization approach assuming perfect knowledge.

Both robust and distributionally robust optimization approaches can also be interpreted from a game theoretic perspective. The decision maker has to choose $\mathbf{x}$ in order to minimize the loss, without knowing the choice of

'nature', which picks the parameters of the loss or their distribution. Assuming 'nature' is a malicious adversary aiming to maximize the same loss, this amounts to a zero-sum game. A solution of the robust problem then gives a choice $\mathbf{x}$ for which the adversary can not increase the loss over some limit.

1.2.1. Ambiguity Sets

A key component of the distributionally robust approach is the choice of ambiguity set. Intuitively, it should be chosen such that it contains all distributions which can reasonably explain the existing data we have. There is again a trade-off between increasing robustness, for which we would want to include all possible distributions, and limiting conservatism, for which we would tend towards including only the empirical sample distribution. Another important consideration not yet mentioned, is the tractability of problem (1.1), which often means reformulating the DRO problem as a convex optimization problem.

There is a multitude of possible choices for the type of ambiguity set used as detailed in [16]. Here, we focus on those characterized by some discrepancy measure. This particular type of ambiguity set means considering all distributions in

$$\mathcal{P} = \left\{p : D\big(p(\mathbf{w}), \hat{p}(\mathbf{w})\big) \leq \varepsilon\right\},$$

where D is a discrepancy measure between two distributions, ε is the radius of the set considered and $\hat{p}(\mathbf{w})$ is a reference distribution. The reference distribution can be made up of the empirical data samples or may encode a belief about the distribution of $\mathbf{w}$. As discrepancy measures, in this thesis, we use Kullback-Leibler divergence and Wasserstein distance.

Kullback-Leibler divergence or relative entropy between two distributions on the same support is defined as

$$\mathrm{KL}\big(p(\mathbf{w}), q(\mathbf{w})\big) = \int_{\mathcal{W}} p(\mathbf{w}) \log \frac{p(\mathbf{w})}{q(\mathbf{w})} \, \mathrm{d}\mathbf{w}$$

and quantifies how much information is lost when approximating distribution p by another distribution q [17, 18]. It can be understood as a measure of closeness between the two distributions and has successfully been used in reformulating distributionally robust optimization [19] in a tractable way.

Wasserstein distances are a concept from optimal transport theory [20, 21]. The Wasserstein distance of order k is defined by the solution the optimization problem

$$W_k(p, q)^k = \inf_{\pi \in \Pi} \int_{\mathcal{W}_p \times \mathcal{W}_q} \|\mathbf{w}_p - \mathbf{w}_q\|^k \, \mathrm{d}\pi(\mathbf{w}_p, \mathbf{w}_q),$$

where the optimization is over all joint probability distributions on $\mathcal{W}_p \times \mathcal{W}_q$ with marginals p and q. An intuitive explanation, stemming from the origins in optimal transport, is that the Wasserstein distance measures the minimum cost for transforming a pile of material, represented by the distribution p, into another one, represented by q. In this analogy, the cost for transporting a unit of material from point $\mathbf{w}_p$ to point $\mathbf{w}_q$ given by some norm (e.g. Euclidean) measuring the distance between the points. The optimization variable π can then be understood as a transport plan specifying how many units to transport from point $\mathbf{w}_p$ to point $\mathbf{w}_q$. Contrary to the Kullback-Leibler divergence, Wasserstein distances can be used to define distance of distributions on different supports, for example between one with continuous and a one with discrete support. In distributionally robust optimization, Wasserstein distance based ambiguity sets have gained significant traction in the recent years [22, 23, 24].

1.3. Distributionally Robust Optimization for Optimal Control

Distributionally robust optimization gives an approach to include uncertainty due to limited data in an optimization problem, while limiting the introduced conservatism. Optimal control is an interesting application, especially when system dynamics are learned from limited data and therefore have inherent uncertainty about them. The goal of this thesis is investigating some approaches which combine these two fields on linear control problems with continuous state and action spaces and different types of uncertainty. During the last years, applications of DRO in control have seen a significant amount of attention.

In finite state and action space Markov decision processes, the distributionally robust approach is used to account for uncertainty in the transition probabilities as well as the reward. As for the single stage DRO problem, also in the sequential decision making case the main difference between the formulations are choice of ambiguity set and tractability of the reformulation. Early work on this topic presented in [25] uses so-called nested ambiguity sets, which give multiple sets of different sizes, each associated with a probability that the real parameter lies inside them. These sets and probabilities are assumed to be known a priori and the distributionally robust MDP is reformulated as a standard robust MDP with a single uncertainty set making its solution tractable. Another approach developed in [26] uses an ambiguity set based on the Wasserstein distance introduced above. This set can be constructed based on a limited number of samples for the transition probability and reward vectors, a case for which a convex optimization formulation is obtained. In [27], a more general approach is taken which encompasses ambiguity sets based on a discrepancy measure as well as ones defined by constraints on the moments of the distribution. The proposed solution procedure requires solving a sequence of convex optimization problems.

In a continuous control setting, distributional robustness has for example been used for so-called chance constraints. These require a certain constraint for a stochastic system to hold at least with a given probability. The extension to distributional robustness is then done by requiring the chance constraint to hold for all distributions from an ambiguity set. In [28], the problem of controlling a linear system under such distributionally robust chance constraints is tackled for an ambiguity set containing all distributions with given first and second moment. Another application presented in [29], aims to solve a predictive control problem. Here, the objective is a worst-case expectation considering noise distributions from a Wasserstein distance based ambiguity set.

The examples given here show that there are many ways of incorporating distributional ambiguity in optimal control problems. Both the fields of optimal control and distributionally robust optimization provide a variety of problem settings and solution approaches. When defining our two problem settings in the next chapters, we also refer to some closely related approaches for each.

2. Wasserstein Distributionally Robust LQR

Our first approach is based on extending the LQR problem by considering additive noise coming from an unknown distribution. Assuming some samples of the noise are available, we derive a controller robust to any noise distribution close to this sample distribution in terms of Wasserstein distance.

2.1. Related Work

In [30], dynamic programming equations for distributionally robust stochastic control with a Wasserstein ambiguity set are given in a general form and an exact reformulation as a semi-infinite program is obtained. As the nominal distribution, the empirical sample distribution is used. The derived value and policy iteration schemes can be solved by discretization, as used in [31] on a specific example system.

For the linear quadratic regulator, in [30], a Lagrangian relaxation of the Wasserstein distributionally robust problem is formulated and a solution in terms of a linear controller found by solving an algebraic Riccati equation given. Here, the ambiguity is assumed to only be with respect to the additive process noise distribution. This work is extended in [32] to the finite-horizon problem.

A different approach is taken in [33], where the case of the system and input matrices depending on a random variable with unknown distribution is treated. From samples of this variable assumed given, an ambiguity set based on moment constraints using the results of [34] is constructed. The controller synthesis is then done by solving a semidefinite program (SDP) and can guarantee closed-loop stability with high probability.

2.2. Problem Formulation

For the approach described here, we formulate the distributionally robust Bellman equation akin to the approach taken in [30], but make an assumption on the value function. This assumption allows for an exact reformulation, not requiring the solution of a semi-infinite program. The random disturbance which we have only limited distributional knowledge about is assumed to be additive noise only.

In particular, we consider the problem of finding a policy $\pi(\mathbf{s})$ minimizing the expected infinite horizon quadratic cost

$$\mathbb{E}\left[\sum_{t=0}^{\infty} c(\mathbf{s}_t, \mathbf{a}_t)\right], \quad c(\mathbf{s}, \mathbf{a}) = \mathbf{s}_t^\mathsf{T}\mathbf{Q}\mathbf{s}_t + \mathbf{a}_t^\mathsf{T}\mathbf{R}\mathbf{a}_t$$

for a linear system

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{w}_t) = \mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim p,$$

where the noise probability distribution p is unknown, but a limited number of samples $\{\hat{\mathbf{w}}_i, \dots \hat{\mathbf{w}}_N\}$ are available to us. To find a policy which is robust with respect to the ambiguity stemming from our partial knowledge of the noise distribution, we follow the approach by Yang [30] and write a distributionally robust Bellman equation

$$\begin{aligned}\pi(\mathbf{s}) &= \operatorname*{argmin}_{\mathbf{a}} \max_{p \in \mathcal{P}_{\varepsilon,k}} \mathbb{E}_{\mathbf{w}\sim p}\Big[c(\mathbf{s},\mathbf{a}) + V(f(\mathbf{s},\mathbf{a},\mathbf{w}))\Big] \\ &= \operatorname*{argmin}_{\mathbf{a}} \max_{p \in \mathcal{P}_{\varepsilon,k}} \mathbb{E}_{\mathbf{w}\sim p}\Big[\mathbf{s}^\mathsf{T}\mathbf{Q}\mathbf{s} + \mathbf{a}^\mathsf{T}\mathbf{R}\mathbf{a} + V(\mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{a} + \mathbf{w})\Big].\end{aligned}$$

Where the set of probability distributions, over which the inner maximization problem needs to be solved, is

$$\mathcal{P}_{\varepsilon,k} = \big\{p : W_k(p, \hat{p}_N) \le \varepsilon\big\},$$

constituting a Wasserstein ball of order k and radius ε around the empirical sample distribution

$$\hat{p}_N = \frac{1}{N}\sum_{i=1}^{N} \delta(\hat{\mathbf{w}}_i),$$

where $\delta(\hat{\mathbf{w}}_i)$ is the Dirac delta distribution supported at $\hat{\mathbf{w}}_i$. For a proper choice of the radius, which depends on the number of samples N, the real noise distribution will lie inside this set with high probability [22]. Solving the above optimization problem will then yield a policy with a probabilistic performance guarantee.

In [31], a similar approach is taken and made tractable by discretizing the state space and approximating the value function by a piecewise linear function. Contrary to this, we assume a value function $V(\mathbf{s}) = \mathbf{s}^\mathsf{T}\mathbf{P}\mathbf{s} + \mathbf{p}^\mathsf{T}\mathbf{s}$ quadratic in the state with a positive definite and symmetric matrix $\mathbf{P}$. This assumption is motivated by the fact that for normal LQR, the value function can be shown to be quadratic. For the distributionally robust case, this is merely an approximation which does not require us to discretize the state space but still leads to a tractable reformulation.

With this assumption, we have

$$\pi(\mathbf{s}) = \operatorname*{argmin}_{\mathbf{a}}\left[\mathbf{a}^\mathsf{T}\mathbf{R}\mathbf{a} + (\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a})^\mathsf{T}\mathbf{P}(\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a}) + \mathbf{p}^\mathsf{T}(\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a}) + \max_{p\in\mathcal{P}_{\varepsilon,k}} \mathbb{E}_{\mathbf{w}\sim p}\Big[\mathbf{w}^\mathsf{T}\mathbf{P}\mathbf{w} + \Big(2(\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a})^\mathsf{T}\mathbf{P} + \mathbf{p}^\mathsf{T}\Big)\mathbf{w}\Big]\right].$$

The inner maximization problem over the Wasserstein ball is untractable in this form. Using a Wasserstein ball of order $k = 2$, we can however reformulate it as

$$\begin{aligned}\min_{\gamma,\mathbf{z}} \quad & \gamma\varepsilon^2 + \frac{1}{N}\sum_{i=1}^{N} z_i \\ \text{s.t.} \quad & \gamma \in \mathbb{R}_+,\ \mathbf{z} \in \mathbb{R}_+^N \\ & \begin{bmatrix} \gamma\mathbf{I} - \mathbf{P} & \mathbf{P}(\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a}) + \frac{1}{2}\mathbf{p} + \gamma\hat{\mathbf{w}}_i \\ (\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a})^\mathsf{T}\mathbf{P} + \frac{1}{2}\mathbf{p}^\mathsf{T} + \gamma\hat{\mathbf{w}}_i^\mathsf{T} & z_i + \gamma\|\hat{\mathbf{w}}_i\|_2^2 \end{bmatrix} \succeq 0 \quad \forall i \in [N],\end{aligned}$$

which is a tractable semidefinite program given by [22, Theorem 11]. Thus, we can now rewrite the policy as

$$\begin{aligned}\pi(\mathbf{s}) = \operatorname*{argmin}_{\mathbf{a},\gamma,\mathbf{z}} \quad & \mathbf{a}^\mathsf{T}\mathbf{R}\mathbf{a} + (\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a})^\mathsf{T}\mathbf{P}(\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a}) + \mathbf{p}^\mathsf{T}(\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a}) + \gamma\varepsilon^2 + \frac{1}{N}\sum_{i=1}^{N} z_i \\ \text{s.t.} \quad & \gamma \in \mathbb{R}_+,\ \mathbf{z} \in \mathbb{R}_+^N \\ & \begin{bmatrix} \gamma\mathbf{I} - \mathbf{P} & \mathbf{P}(\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a}) + \frac{1}{2}\mathbf{p} + \gamma\hat{\mathbf{w}}_i \\ (\mathbf{A}\mathbf{s}+\mathbf{B}\mathbf{a})^\mathsf{T}\mathbf{P} + \frac{1}{2}\mathbf{p}^\mathsf{T} + \gamma\hat{\mathbf{w}}_i^\mathsf{T} & z_i + \gamma\|\hat{\mathbf{w}}_i\|_2^2 \end{bmatrix} \succeq 0 \quad \forall i \in [N].\end{aligned} \tag{2.1}$$

For a given $\mathbf{s}$, we can now solve the above convex optimization problem to find the optimal control $\mathbf{a}$. While we cannot find a closed form solution for this problem as we have for standard LQR, SDPs allow for numerical solution with essentially the same tractability as linear programs [35]. However, the optimization problem depends on the parameters of the quadratic value function approximation $\mathbf{P}$ and $\mathbf{p}$, so we can not solve it without choosing parameter values which yield a good approximation.

2.2.1. Reformulation as Disciplined Parametrized Program

Control policies given by such a parametrized convex optimization problem have recently been studied by Agrawal et al. [36]. Their approach is based on the differentiability of so-called disciplined parametrized programs with respect to the parameters [37]. Using simulated trajectories of the closed loop system to compute a Monte Carlo approximation of the expected trajectory cost, the controller parameters which minimize this cost can be found by stochastic gradient descent.

To apply this approach for the policy given in equation (2.1), we have to rewrite the optimization problem slightly to make it compliant with the disciplined parametrized programming paradigm. We have

$$\begin{aligned}
\pi(\mathbf{s}) = \underset{\mathbf{a},\gamma,\mathbf{z}}{\operatorname{argmin}} \quad & \mathbf{a}^\mathsf{T}\mathbf{R}\mathbf{a} + \|\boldsymbol{\theta}\mathbf{s}'\|_2^2 + \mathbf{p}^\mathsf{T}\mathbf{s}' + \gamma\varepsilon^2 + \frac{1}{N}\sum_{i=1}^{N} z_i \\
\text{s.t.} \quad & \gamma \in \mathbb{R}_+,\ \mathbf{z} \in \mathbb{R}_+^N \\
& \mathbf{s}' = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{a} \\
& \mathbf{s}'' = \boldsymbol{\theta}\mathbf{s}' \\
& \boldsymbol{\theta}' = \boldsymbol{\theta} \\
& \begin{bmatrix} \gamma\mathbf{I} - \boldsymbol{\theta}'^\mathsf{T}\boldsymbol{\theta} & \boldsymbol{\theta}^\mathsf{T}\mathbf{s}'' + \frac{1}{2}\mathbf{p} + \gamma\hat{\mathbf{w}}_i \\ \mathbf{s}''^\mathsf{T}\boldsymbol{\theta} + \frac{1}{2}\mathbf{p}^\mathsf{T} + \gamma\hat{\mathbf{w}}_i^\mathsf{T} & z_i + \gamma\|\hat{\mathbf{w}}_i\|_2^2 \end{bmatrix} \succeq 0 \quad \forall i \in [N],
\end{aligned}$$

in which we change the parameter from $\mathbf{P}$ to $\boldsymbol{\theta}$, which is defined by $\mathbf{P} = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta}$, allowing us to rewrite the quadratic form in the objective as $\mathbf{s}'^\mathsf{T}\mathbf{P}\mathbf{s}' = \|\boldsymbol{\theta}\mathbf{s}'\|_2^2$, since $\mathbf{P}$ is positive definite. Additionally, we introduce the auxiliary variables $\mathbf{s}', \mathbf{s}''$ and $\boldsymbol{\theta}'$. These additional steps finally leave us with a policy given as a convex optimization problem which we can differentiate with respect to the parameters of the value function.

2.2.2. Worst-Case Distribution

For evaluation, it is useful to obtain the worst-case distribution explicitly which, following [22], can be done by solving the quadratically constrained quadratic problem (QCQP)

$$\begin{aligned}
\max_{\alpha,\boldsymbol{\rho}_i} \quad & \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{w}}_i + \boldsymbol{\rho}_i)^\mathsf{T}\mathbf{P}(\hat{\mathbf{w}}_i + \boldsymbol{\rho}_i) + \left((\mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{a})^\mathsf{T}\mathbf{P}^\mathsf{T} + \frac{1}{2}\mathbf{p}^\mathsf{T}\right)(\hat{\mathbf{w}}_i + \boldsymbol{\rho}_i) + \alpha\lambda_{\max}(\mathbf{P}) \\
\text{s.t.} \quad & \alpha \in \mathbb{R}_+,\ \boldsymbol{\rho}_i \in \mathbb{R}^m \\
& \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{\rho}_i\|_2^2 + \alpha \leq \varepsilon^2,
\end{aligned} \tag{2.2}$$

in which $\lambda_{\max}(\mathbf{P})$ is the maximum eigenvalue of $\mathbf{P}$. The worst-case noise distribution is then given by

$$P_{\text{worst}} = \frac{1}{N}\sum_{i\neq i_0}^{N}\delta(\hat{\mathbf{w}}_i + \boldsymbol{\rho}_i^*) + \frac{n-1}{nN}\delta(\hat{\mathbf{w}}_{i_0} + \boldsymbol{\rho}_{i_0}^*) + \frac{1}{nN}\delta\big(\hat{\mathbf{w}}_{i_0} + \sqrt{nN\alpha^*}\mathbf{v}_{\max}(\mathbf{P})\big) \tag{2.3}$$

for $n \to \infty$ and with $\mathbf{v}_{\max}(\mathbf{P})$ as an eigenvector corresponding to $\lambda_{\max}(\mathbf{P})$. The optimal solutions of problem (2.2) are denoted by $\alpha^*, \boldsymbol{\rho}_i^*$. This distribution assigns probability $1/N$ to the training samples perturbed by $\boldsymbol{\rho}_i^*$, while one sample indicated by i_0 is perturbed infinitely far in the direction of $\mathbf{v}_{\max}(\mathbf{P})$ and assigned vanishing probability.

2.3. Implementation

Here we introduce our implementation of the controller described above, for which we use *cvxpy*[38] for the formulation of the optimization problems and the *PyTorch*[39] implementation of *cvxpylayers*[37] to differentiate through them.

We first need to find the optimal parameter $\boldsymbol{\theta}$ of the controller. For this, we assume that some samples from the real noise distribution are available as well as the system and input matrices $\mathbf{A}$ and $\mathbf{B}$. We can initialize the parameter by using the solution of the normal LQR problem or just choosing an arbitrary initial value. We then improve it by stochastic gradient, with the objective of minimizing the rollout trajectory loss. For the rollouts, we sample the noise from an arbitrary distribution P_{train}, for which we used a multivariate Gaussian. While this seems counterintuitive at first, the empirical samples around which we construct the Wasserstein ambiguity set determine the constraints of the optimization based control policy. Hence, the controller will always consider the distributional robustness with respect to that sample distribution, while the value function parameters have to be adapted to the system dynamics at hand.

One interpretation of this approach would be to understand the limited samples as coming from a real system and costly to obtain. In the present case, the assumption would be that we have a good model $\mathbf{A}, \mathbf{B}$ of the system dynamics, but our knowledge of the noise distribution is limited to the few samples obtained from the real system. We could then solve for the distributionally robust controller in simulation by the approach described here and apply it on the real system. With some probability depending on the number of samples and the robustness level ε chosen, we could then guarantee that the controller performs at least as good under the real noise distribution as under the worst-case one in simulation.

For finding the worst case distribution which is interesting to evaluate and compare the robust to the nominal controller, we need to solve the QCQP given in equation (2.2). The matrix $\mathbf{P}$ being positive definite makes this problem non-convex, since it aims to maximize a convex function. To solve it directly, we utilized the *dccp* [40] extension to *cvxpy*, which uses a heuristic approach for approximately solving so-called convex-concave programs. Another option is reformulation to a convex SDP, which is possible following the approach described in [41, Appendix B.1]. The reformulation for our specific problem is given in appendix A. We also implemented this approach, which significantly speeds up the optimization and thereby the evaluation of the worst-case distribution. Since the problem depends on the state $\mathbf{s}$ both directly and through the control $\mathbf{a}$, which we obtain from $\pi(\mathbf{s})$, we need to solve it at each timestep during the rollout to sample from the worst-case distribution.

input : T // rollout time horizon
$\mathbf{A}, \mathbf{B}$ // system and input matrices
$\hat{\mathbf{w}}_i, i = 1 \dots N$ // samples from real noise distribution
ε // radius of Wasserstein ball
n_{batch} // rollout batch size
P_{train} // rollout noise distribution

output : $\boldsymbol{\theta}^*, \mathbf{p}^*$ // optimal value function parameters

initialize : $\boldsymbol{\theta}, \mathbf{p}$ // initial guess value function parameters

while *not converged* **do**
// get Monte Carlo estimate of loss from batch of trajectories under current policy
$\hat{J} \leftarrow \text{rollout\_loss}\big(\mathbf{A}, \mathbf{B}, P_{\text{train}}, T, n_{\text{batch}}, \pi(\varepsilon, \hat{\mathbf{w}}_i, \boldsymbol{\theta}, \mathbf{p})\big)$
// get stochastic gradient estimate by backpropagation
$\nabla_{\boldsymbol{\theta},\mathbf{p}} \hat{J} \leftarrow \text{backprop}(\hat{J}, \boldsymbol{\theta}, \mathbf{p})$
// update parameters by stochastic gradient descent
$\boldsymbol{\theta}, \mathbf{p} \leftarrow \text{SGD\_step}(\boldsymbol{\theta}, \mathbf{p}, \nabla_{\boldsymbol{\theta},\mathbf{p}} \hat{J})$
$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
$\mathbf{p}^* \leftarrow \mathbf{p}$

Algorithm 1: Pseudo code for tuning parameters of Wasserstein distributionally robust LQR policy

2.4. Evaluation

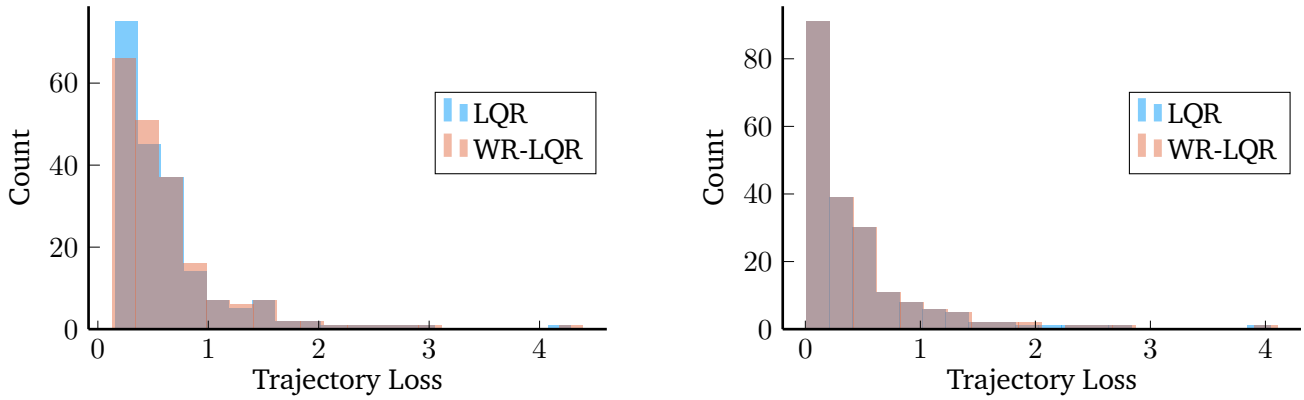
With both controller and worst-case distribution formulated in a tractable form, we can compare the distributionally robust controller to the normal LQR solution. Also, we investigate the influence of the ambiguity set size, as well as getting some intuition on the form of worst-case distribution.

2.4.1. Example System

Here, we first look at a one dimensional unstable example system. For this, we used a rollout horizon of $T = 50$ timesteps with the initial state sampled from a Normal distribution with zero mean and variance $\sigma_0^2 = 2$. For the additive noise distribution P_{train} during the parameter fitting process, we picked a Gaussian with zero mean and $\sigma^2 = 10^{-3}$. We also took the $N = 10$ empirical noise samples, which make up the distribution at the center of our Wasserstein ambiguity set, from this training distribution. We then solved for the normal infinite horizon LQR controller by the discrete time algebraic Riccati equation, and for the distributionally robust one by running algorithm 1 for a maximum of 200 steps or until convergence. *Adam*[42] was used as the stochastic gradient descent algorithm.

Performance evaluation was done by obtaining 200 trajectories for both controllers, once under training noise and once under worst-case noise, as given by equation (2.3). For a Wasserstein ball radius of $\varepsilon = 0.1$, figure 2.1 shows a comparison of the resulting trajectory losses. The almost complete overlap of the two histograms shows that for this size of the ambiguity set, the difference between the two controllers is marginal both when sampling Gaussian noise and when sampling from the worst-case noise. Increasing the radius to $\varepsilon = 1.0$

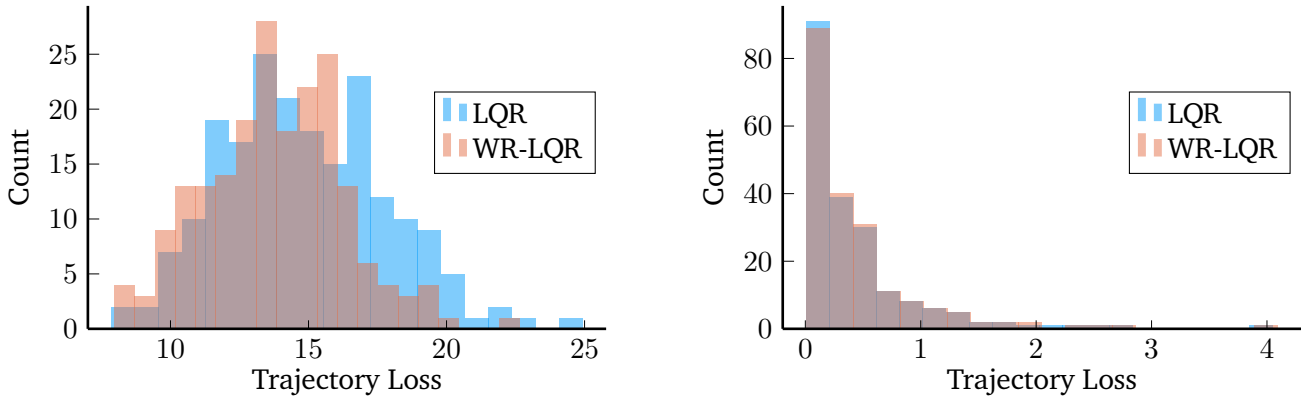
however results in a slightly different picture. As shown figure 2.1, the distributionally robust controller outperforms the Riccati based LQR controller on average if noise is sampled from the worst-case distribution, while not worsening performance under Gaussian noise significantly. Still, the improvement is relatively small and, as we will see in the following, the perturbations to the original samples forming the worst case distribution are already quite large for this case.



(a) Worst-case noise

(b) Gaussian noise

Figure 2.1.: Trajectory loss for Riccati based LQR and Wasserstein distributionally robust LQR with $\varepsilon = 0.1$. Loss distribution is obtained by collecting 200 rollouts with each controller and sampling the additive noise from the respective noise distribution.



(a) Worst-case noise

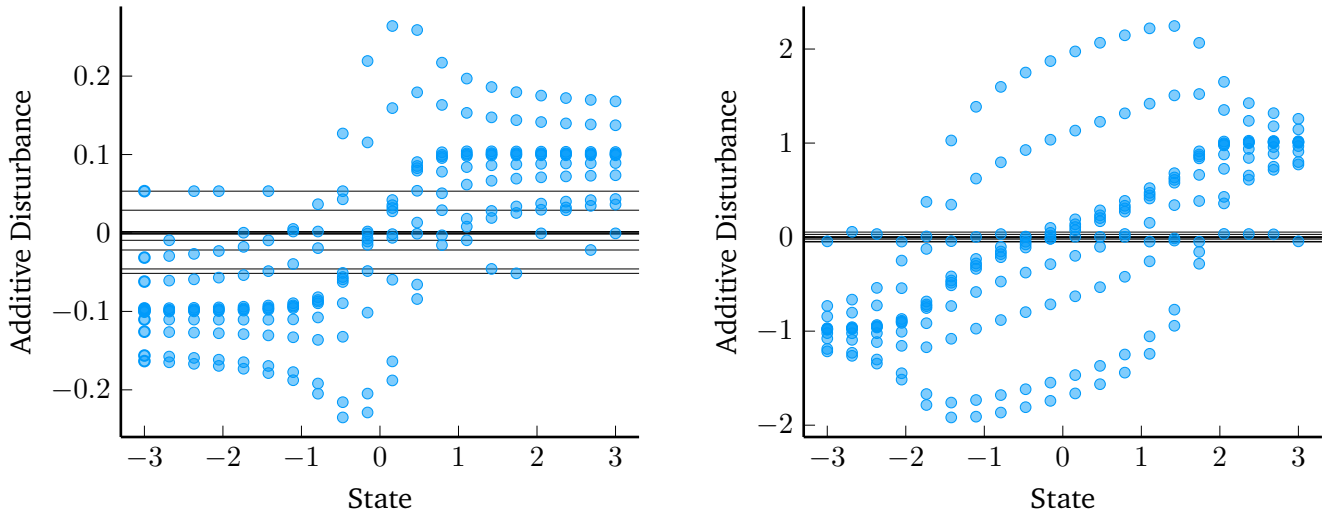
(b) Gaussian noise

Figure 2.2.: Trajectory loss for Riccati based LQR and Wasserstein distributionally robust LQR with $\varepsilon = 1.0$. Loss distribution is obtained by collecting 200 rollouts with each controller and sampling the additive noise from the respective noise distribution.

For an understanding of what the worst-case noise looks like, we show it for both values of the parameter ε in figure 2.3. Equation (2.3) shows, that the worst-case distribution consists of $N + 1$ probability atoms and is state dependent. Hence, we show the support of this distribution for multiple values of the state. At each value, the vertically aligned dots show the values at which the worst case distribution is supported. The original samples, around which the ambiguity set is constructed are indicated by horizontal black lines. For negative value of the state, the whole noise distribution is shifted to more negative values. For positive state, the converse is true. Intuitively this makes sense, as it will lead the system state to stray even further away

from zero, accumulating large losses. For a state close to zero, the noise variance is increased significantly with some samples being perturbed in a negative, some in a positive direction.

To confirm that the obtained distributions fall within the prescribed Wasserstein ambiguity set, we use the *Python Optimal Transport* library [43] for numerically calculating the Wasserstein distance between empirical and worst-case sample distribution. For all cases checked, the result matched the previously set value of ε well, both when solving problem (2.2) directly using *dccp* and by solving the reformulated SDP.



(a) $\varepsilon = 0.1$ (b) $\varepsilon = 1.0$

Figure 2.3.: Visualization of worst-case distribution for two different ambiguity set sizes. Horizontal black lines indicate the empirical noise samples, while at each state value, vertically aligned dots indicate the perturbed samples forming the worst-case distribution. The empirical samples are the same for both cases.

For the example system treated here, we can see the distributionally robust controller improving performance under the worst-case distribution, while performing similarly to the normal LQR controller under Gaussian noise. However, the radius of ambiguity set at which the performance difference is seen leads to quite significantly perturbing the empirical distribution as seen in figure 2.3.

2.4.2. Randomly Generated Systems

For the example system above, our controller approach showed some improved robustness to a perturbed noise distribution when a large enough ambiguity set is considered. Here, we show that this is not generally the case by considering some randomly generated systems for which we run the same process as above and give some possible explanations. For 10 different systems, table 2.1 shows the average reward over 30 rollouts under the worst-case distribution. All parameters were chosen as for the above example system. While for some systems, highlighted by green coloring in the table, performance was slightly improved compared to the LQR solution, for the rest this was not the case. Some tests showed pretty much equal performance (yellow) and some even much worse performance (red) of the controller which should be robust to the worst-case distribution used. While this is not an extensive evaluation, it provides evidence that the approach does not work equally well for different systems.

As we find the robust controller by stochastic gradient descent, one explanation for this is choice of hyperparameters such as initialization of parameters, batch size, learning rate and convergence criterion leading to a policy which has not properly converged. In the last column of the table, the results for the same systems with a different learning rate are given. While the problem of underperforming with respect to LQR persists for the same systems, changing the learning rate lowers the gap in this case. The best choice of hyperparameters will depend on the particular problem and system dynamics to be solved.

Another explanation for why our method fails to improve performance might be the value function approximation chosen. While it allows the reformulation as a tractable SDP, the choice of a value function quadratic in the state might not be an equally good approximation for all system dynamics. More flexible approximations such as the piecewise linear approach used in [31] might be more generally usable.

Table 2.1.: Loss averaged over 30 trajectories for 10 randomly generated unstable systems evaluated under the worst-case noise distribution. The standard LQR controller is compared to the Wasserstein distributionally robust controller with two different learning rates during the gradient descent parameter optimization. Rows colored in green indicate improvement over normal LQR, yellow similar performance and red worse.

	Average Trajectory Loss		
System	LQR	WR-LQR, $\eta = 0.1$	WR-LQR, $\eta = 0.2$
1	$7.89 \cdot 10^1$	$7.50 \cdot 10^2$	$1.70 \cdot 10^2$
2	$1.07 \cdot 10^1$	$1.06 \cdot 10^1$	$1.08 \cdot 10^1$
3	$1.31 \cdot 10^1$	$1.31 \cdot 10^1$	$1.29 \cdot 10^1$
4	$1.52 \cdot 10^1$	$1.33 \cdot 10^1$	$1.34 \cdot 10^1$
5	$1.49 \cdot 10^1$	$1.39 \cdot 10^1$	$1.25 \cdot 10^1$
6	$6.05 \cdot 10^1$	$6.45 \cdot 10^2$	$5.51 \cdot 10^2$
7	$1.21 \cdot 10^2$	$1.13 \cdot 10^3$	$2.34 \cdot 10^2$
8	$2.79 \cdot 10^1$	$2.47 \cdot 10^1$	$2.04 \cdot 10^1$
9	$5.18 \cdot 10^1$	$1.55 \cdot 10^3$	$2.90 \cdot 10^2$
10	$4.57 \cdot 10^3$	$6.25 \cdot 10^6$	$6.10 \cdot 10^6$

Overall, the evaluation of our approach suggests that it can lead to some increase in robustness against non-Gaussian additive noise. However, the improvement might be quite small and comes with an increase in computational cost when comparing to the closed-form LQR solution. Furthermore, the introduction of hyperparameters needed for the stochastic gradient descent fitting of controller parameters adds additional complexity. Optimal parameter choice seems to also depend on the specific problem at hand, making the approach less generally applicable.

2.5. Future Work

From the above evaluation, some possible paths for improvement arise, which we will detail here.

Influence of Value Function Approximation

An important point would be the investigation of different ways of approximating the value function, which still result in a tractable reformulation. Part of this investigation would also be studying the influence particular system dynamics have on the quality of the quadratic approximation chosen here. Another option would be directly solving the semi-infinite problem derived in [30] to compute the value function using for example the *SIPAMPL* software package [44].

Choice of Ambiguity Set

To really make use of the probabilistic performance guarantees from distributionally robust optimization, a principled approach of choosing an appropriate size for the Wasserstein distance based ambiguity set would need to be included in our approach.

Another interesting comparison would be with regard to completely different ambiguity sets. Some options include Wasserstein ambiguity sets based on different norms in the Wasserstein distance used or centered at a Normal distribution instead of the empirical sample distribution used here.

Ambiguity in Dynamics Parameters

Ambiguity of the additive noise distribution might also not be the most important to robustify against. Here, it was chosen mainly as an approachable first step to applying the distributionally robust methods to an optimal control problem. Including the case of ambiguity in the dynamics parameters, stemming for example from learning them from data, would thus be another important extension of the approach taken here.

Comparison to Related Approaches

In developing our approach, we focused on deriving a tractable reformulation and evaluating if the controller improves performance under the worst-case noise. Detailed comparison against other approaches is another important aspect for future work. This is somewhat difficult due to different approaches aiming for robustness within a different ambiguity set, such as the one in [32]. These differences make performance under the worst-case distribution hard to compare. A good way to evaluate the approaches would thus be comparing performance on a set of realistic benchmark problems.

3. Distributionally Robust Trajectory Optimization with Kullback-Leibler Divergence Constraint

Ultimately, we would like to consider the ambiguity introduced by learning system dynamics from trajectories directly in designing the controller. The approach discussed in chapter 2 only considered ambiguity in the additive noise distribution with no straightforward extension to include parameter ambiguity. Additionally, the controller execution still needed the numerical solution of a convex optimization problem, which also grows in size with more empirical data points considered.

In this chapter, we therefore extend the framework of trajectory optimization with linearized dynamics to include distributional robustness with respect to the distribution of dynamics parameters. Furthermore, the ambiguity set chosen here is based on the KL-divergence, allowing a closed form solution of the worst-case distribution assuming a Gaussian nominal distribution.

3.1. Related Work

3.1.1. Trajectory Optimization with Linearized Dynamics

Stochastic optimal control with linearized dynamics is a powerful technique for controlling nonlinear systems. Based on approximating the nonlinear dynamics by a linear one around a given trajectory, a locally optimal controller can be found by dynamic programming techniques. Using the updated controller to collect new trajectories, the process is then iterated until convergence. While original methods such as *Differential Dynamic Programming* (DDP)[45] or *Iterative Linear Quadratic Gaussian* (ILQG) [46] are based on a given nonlinear model, we here assume the linear model is obtained by fitting directly to trajectory data collected from an unknown nonlinear system.

As the linearized dynamics are only a good approximation close to the linearization point, it is important to limit the optimism in the controller update of each iteration. One approach which has been successfully used is enforcing a relative entropy or KL bound between the trajectory distributions of successive iterations. This approach is used for trajectory optimization in the context of *Guided Policy Search* (GPS) [47, 48]. In this policy search method, a reward maximizing trajectory is optimized using the local linear model and an arbitrary parametrized policy is then optimized to match the obtained trajectory. Here we limit ourselves to the trajectory optimization step and use the formulation derived in [49] which shows the constraint on trajectory distributions to be equivalent to one on expected relative entropy between the policies of successive iterations.

3.1.2. Distributionally Robust Optimization with Kullback-Leibler Divergence Constraint

Distributionally robust optimization with a KL-divergence constraint on the ambiguity set has been formulated in [19] where it is solved via strong duality reformulation as a convex optimization problem. More work on ϕ-divergence based ambiguity sets, which KL-divergence is a special case of, appears for example in [50, 51], with a more extensive overview given in [16].

In [52], KL based DRO is used in a reinforcement learning setting to account for the finite number of samples utilized for estimating the value of a policy in modified policy iteration [53]. While this approach does not directly relate to the one taken here, it illustrates another method of using KL based DRO for control and leads to a worst-case adversarial policy with a similar exponential reweighting term as we find for the worst-case parameter distribution.

3.2. General Problem Formulation

In the stochastic optimal control formulation chosen here, the system dynamics $\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta})$ are a distribution over the next state $\mathbf{s}'$ given the current state $\mathbf{s}$, action $\mathbf{a}$ and dynamics parameters $\boldsymbol{\theta}$. Similarly, the policy will generally be a distribution over actions given a state written as $\pi_t(\mathbf{a}|\mathbf{s})$ and at each timestep, we will obtain a state distribution $\mu_t(\mathbf{s})$. Contrary to the previous chapter, we here formulate the problem as reward maximization instead of minimization of a loss. Based on the idea of robustifying the trajectory optimization approach against uncertainty in the system dynamics, we formulate the distributionally robust optimization problem as

$$
\begin{aligned}
\max_{\pi_t(\mathbf{a}|\mathbf{s})} \min_{p_t(\boldsymbol{\theta})} \quad & \sum_{t=1}^{T-1} \int_{\mathbf{s}} \int_{\mathbf{a}} R_t(\mathbf{s},\mathbf{a}) \mu_t(\mathbf{s}) \pi_t(\mathbf{a}|\mathbf{s}) \,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s} + \int_{\mathbf{s}} R_T(\mathbf{s}) \mu_T(\mathbf{s}) \,\mathrm{d}\mathbf{s} \\
\text{subject to} \quad & \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \,\mathrm{d}\mathbf{a} = 1, \quad \forall \mathbf{s}, \forall t < T \\
& \int_{\boldsymbol{\theta}} \int_{\mathbf{s}} \int_{\mathbf{a}} \mu_{t-1}(\mathbf{s}) \pi_{t-1}(\mathbf{a}|\mathbf{s}) \mathcal{P}_{t-1}(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}) p_{t-1}(\boldsymbol{\theta}) \,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s}\,\mathrm{d}\boldsymbol{\theta} = \mu_t(\mathbf{s}'), \quad \forall \mathbf{s}', \forall t > 1 \\
& \int_{\mathbf{s}} \mu_t(\mathbf{s}) \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \log \frac{\pi_t(\mathbf{a}|\mathbf{s})}{q_t(\mathbf{a}|\mathbf{s})} \,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s} \leq \varepsilon_\pi, \quad \forall t < T \qquad (3.1) \\
& \mu_t(\mathbf{s}) = \mu_1(\mathbf{s}), \quad \forall \mathbf{s}, t = 1 \\
& \int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta} = 1, \quad \forall t < T \\
& \int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta}) \log \frac{p_t(\boldsymbol{\theta})}{\hat{p}_t(\boldsymbol{\theta})} \,\mathrm{d}\boldsymbol{\theta} \leq \varepsilon_p, \quad \forall t < T. \qquad (3.2)
\end{aligned}
$$

The goal is to find a time-dependent policy $\pi(\mathbf{a}|\mathbf{s})$ maximizing the expected sum of rewards over the trajectory. However, since we want to make this robust to variations in the dynamics parameters, the inner optimization aims to minimize the same objective by changing the distribution of parameters $p_t(\boldsymbol{\theta})$. As in GPS, the policy is constrained by equation (3.1) to not fall to far from a previous policy $q_t(\mathbf{a}|\mathbf{s})$ in terms of expected KL-divergence. This constraint is necessary, since the dynamics $\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta})$ and reward $R_t(\mathbf{s},\mathbf{a})$ are assumed to be valid only locally and thus optimizing the policy globally might lead to unstable behavior. For the parameter distribution, we introduce a similar constraint as given in equation (3.2). Here, the distribution $\hat{p}_t(\boldsymbol{\theta})$ expresses the decision

maker's prior belief about the parameter. In practice, this should be obtained together with the parametrized dynamics when fitting them to given trajectories at each timestep. The constraint can be understood as limiting the possible changes the imaginary adversary can make to the parameter distribution. Another interpretation is limiting the distributions of the parameter, which we want robustness against, to ones which are close to the empirical distribution and thus good candidates for the actual parameter distribution.

Our approach aims to directly account for the fact that not only is the dynamics approximation local, but also learned from limited data. The unknown real stochastic and nonlinear dynamics will lead to varying parameter estimates for the locally valid linear dynamics we use in the optimization. The aim of introducing the kind of robustness used here is to yield a tractable algorithm which accounts for this apparent parameter stochasticity during the optimization process and thus yields a policy update which is less likely to underperform on the real nonlinear system.

We will try to solve the distributionally robust problem in an alternating fashion, first finding the worst-case parameter distribution for a given policy $\pi_t(\mathbf{a}|\mathbf{s})$ by solving

$$\begin{aligned}
\min_{p_t(\boldsymbol{\theta})} \quad & \sum_{t=1}^{T-1} \int_{\mathbf{s}} \int_{\mathbf{a}} R_t(\mathbf{s},\mathbf{a}) \mu_t(\mathbf{s}) \pi_t(\mathbf{a}|\mathbf{s}) \, \mathrm{d}\mathbf{a} \, \mathrm{d}\mathbf{s} + \int_{\mathbf{s}} R_T(\mathbf{s}) \mu_T(\mathbf{s}) \, \mathrm{d}\mathbf{s} \\
\text{subject to} \quad & \int_{\boldsymbol{\theta}} \int_{\mathbf{s}} \int_{\mathbf{a}} \mu_{t-1}(\mathbf{s}) \pi_{t-1}(\mathbf{a}|\mathbf{s}) \mathcal{P}_{t-1}(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}) p_{t-1}(\boldsymbol{\theta}) \, \mathrm{d}\mathbf{a} \, \mathrm{d}\mathbf{s} \, \mathrm{d}\boldsymbol{\theta} = \mu_t(\mathbf{s}'), \quad \forall \mathbf{s}', \forall t > 1 \\
& \int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta} = 1, \quad \forall t < T \\
& \int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta}) \log \frac{p_t(\boldsymbol{\theta})}{\hat{p}_t(\boldsymbol{\theta})} \, \mathrm{d}\boldsymbol{\theta} \leq \varepsilon_p, \quad \forall t < T \\
& \mu_t(\mathbf{s}) = \mu_1(\mathbf{s}), \quad \forall \mathbf{s}, t = 1
\end{aligned} \tag{3.3}$$

and then fixing $p_t(\boldsymbol{\theta})$ and solving

$$\begin{aligned}
\max_{\pi_t(\mathbf{a}|\mathbf{s})} \quad & \sum_{t=1}^{T-1} \int_{\mathbf{s}} \int_{\mathbf{a}} R_t(\mathbf{s},\mathbf{a}) \mu_t(\mathbf{s}) \pi_t(\mathbf{a}|\mathbf{s}) \, \mathrm{d}\mathbf{a} \, \mathrm{d}\mathbf{s} + \int_{\mathbf{s}} R_T(\mathbf{s}) \mu_T(\mathbf{s}) \, \mathrm{d}\mathbf{s} \\
\text{subject to} \quad & \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \, \mathrm{d}\mathbf{a} = 1, \quad \forall \mathbf{s}, \forall t < T \\
& \int_{\boldsymbol{\theta}} \int_{\mathbf{s}} \int_{\mathbf{a}} \mu_{t-1}(\mathbf{s}) \pi_{t-1}(\mathbf{a}|\mathbf{s}) \mathcal{P}_{t-1}(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}) p_{t-1}(\boldsymbol{\theta}) \, \mathrm{d}\mathbf{a} \, \mathrm{d}\mathbf{s} \, \mathrm{d}\boldsymbol{\theta} = \mu_t(\mathbf{s}'), \quad \forall \mathbf{s}', \forall t > 1 \\
& \int_{\mathbf{s}} \mu_t(\mathbf{s}) \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \log \frac{\pi_t(\mathbf{a}|\mathbf{s})}{q_t(\mathbf{a}|\mathbf{s})} \, \mathrm{d}\mathbf{a} \, \mathrm{d}\mathbf{s} \leq \varepsilon_\pi, \quad \forall t < T \\
& \mu_t(\mathbf{s}) = \mu_1(\mathbf{s}), \quad \forall \mathbf{s}, t = 1
\end{aligned}$$

to obtain the new policy. These two steps are then repeated until convergence, where for each policy optimization step, we fully optimize the worst-case parameter distribution using the current policy. In practice, it turns out that we first need to find policy which is not too bad in some sense such that the inner minimization problem becomes solvable.

Here, we will focus on solving the constrained reward minimization problem for the parameter distribution, since the policy optimization step is equivalent to the one of GPS as derived in [49] just with a modified

dynamics constraint. For the linear Gaussian case we will provide an approach to incorporate this modification. To find conditions for the optimality of the problem given in (3.3), we write its Lagrangian

$$\begin{aligned}L_p(p_t,\mu_t,V_t,\alpha_t,\beta_t) = &\sum_{t=1}^{T-1}\int_{\mathbf{s}}\int_{\mathbf{a}} R_t(\mathbf{s},\mathbf{a})\mu_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s})\,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s} + \int_{\mathbf{s}} R_T(\mathbf{s})\mu_T(\mathbf{s})\,\mathrm{d}\mathbf{s}\\ &+ \int_{\mathbf{s}} V_1(\mathbf{s})\mu_1(\mathbf{s})\,\mathrm{d}\mathbf{s} - \int_{\mathbf{s}'} V_T(\mathbf{s}')\mu_T(\mathbf{s}')\,\mathrm{d}\mathbf{s}' - \sum_{t=1}^{T-1}\int_{\mathbf{s}'} V_t(\mathbf{s}')\mu_t(\mathbf{s}')\,\mathrm{d}\mathbf{s}'\\ &+ \sum_{t=1}^{T-1}\int_{\mathbf{s}'} V_{t+1}(\mathbf{s}')\int_{\boldsymbol{\theta}}\int_{\mathbf{s}}\int_{\mathbf{a}} \mu_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s})\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta})p_t(\boldsymbol{\theta})\,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s}\,\mathrm{d}\boldsymbol{\theta}\,\mathrm{d}\mathbf{s}'\\ &+ \sum_{t=1}^{T-1}\alpha_t\left(\int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta})\log\frac{p_t(\boldsymbol{\theta})}{\hat{p}_t(\boldsymbol{\theta})}\,\mathrm{d}\boldsymbol{\theta} - \varepsilon_p\right) + \sum_{t=1}^{T-1}\beta_t\left(\int_{\theta} p_t(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} - 1\right),\end{aligned}$$

where $\alpha_t > 0, \beta_t > 0$. Solving

$$\frac{\partial L_p}{\partial p_t} = 0, \quad \frac{\partial L_p}{\partial \beta_t} = 0$$

gives an expression for the worst-case parameter distribution

$$p_t(\boldsymbol{\theta}) \propto \hat{p}_t(\boldsymbol{\theta})\exp\left(-\frac{1}{\alpha_t}\int_{\mathbf{s}} \mu_t(\mathbf{s})Q_t(\mathbf{s},\boldsymbol{\theta})\,\mathrm{d}\mathbf{s}\right), \tag{3.4}$$

where

$$Q_t(\mathbf{s},\boldsymbol{\theta}) = \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}')\int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s})\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta})\,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s}'$$

can be understood as the state-action value function of the adversary whose actions are the dynamics parameters. Solving for β_t results in the necessary normalization constant.

Furthermore, by solving

$$\frac{\partial L_p}{\partial \mu_T} = 0, \quad \frac{\partial L_p}{\partial \mu_t} = 0,$$

we get the backward pass

$$V_t(\mathbf{s}) = \begin{cases} R_T(\mathbf{s}), & t = T \\ \displaystyle\int_{\mathbf{a}} R_t(\mathbf{s},\mathbf{a})\pi_t(\mathbf{a}|\mathbf{s})\,\mathrm{d}\mathbf{a} + \int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta})Q_t(\mathbf{s},\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}, & 1 \le t < T \end{cases}, \tag{3.5}$$

which gives us an iterative formula for computing the Lagrange multipliers $V_t(\mathbf{s})$ at each timestep backwards in time. To find the forward pass, we solve

$$\frac{\partial L_p}{\partial V_1} = 0, \quad \frac{\partial L_p}{\partial V_t} = 0, \tag{3.6}$$

yielding an iterative formula

$$\mu_t(\mathbf{s}) = \begin{cases} \mu_1(\mathbf{s}) & t = 1 \\ \displaystyle\int_{\boldsymbol{\theta}} p_{t-1}(\boldsymbol{\theta})\int_{\mathbf{s}}\int_{\mathbf{a}} \mu_{t-1}(\mathbf{s})\pi_{t-1}(\mathbf{a}|\mathbf{s})\mathcal{P}_{t-1}(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta})\,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s}\,\mathrm{d}\boldsymbol{\theta} & 2 \le t \le T \end{cases} \tag{3.7}$$

for the state distribution. Using the above results, we can simplify

$$L_p(\mu_t, V_t, \alpha_t, p_t) = \int_{\mathbf{s}} V_1(\mathbf{s})\mu_1(\mathbf{s})\,\mathrm{d}\mathbf{s} + \sum_{t=1}^{T-1} \alpha_t\Big(\mathrm{KL}(p_t(\boldsymbol{\theta})\|\hat{p}_t(\boldsymbol{\theta})) - \varepsilon_p\Big)$$

and finally get the last optimality condition

$$\frac{\partial L_p}{\partial \alpha_t} = \mathrm{KL}(p_t(\boldsymbol{\theta})\|\hat{p}_t(\boldsymbol{\theta})) - \varepsilon_p$$

for the Lagrange multipliers of the KL constraint. In the following, the linear Gaussian case is considered.

3.3. Linear Quadratic Gaussian Case

Here, we will look at a problem where the dynamics at each timestep are linear in the state and have Gaussian noise, while the reward function is quadratic in both state and action. Additionally, we assume ambiguity just in the parameter matrix multiplying the state, which results in linear dynamics with random parameter matrix $\boldsymbol{\Theta} \in \mathbb{R}^{n\times n}$ given by

$$\mathcal{P}_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\Theta}) = \mathcal{N}(\mathbf{s}'|\boldsymbol{\Theta}\mathbf{s} + \mathbf{b}_t\mathbf{a} + \mathbf{c}_t, \boldsymbol{\Sigma}_{s',t}).$$

Defining $\boldsymbol{\theta} = \mathrm{vec}(\boldsymbol{\Theta})$, we can write this equivalently as

$$\mathcal{P}_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{s}'|(\mathbf{s}^\mathsf{T} \otimes \mathbf{I}_n)\boldsymbol{\theta} + \mathbf{b}_t\mathbf{a} + \mathbf{c}_t, \boldsymbol{\Sigma}_{s',t}),$$

where $\otimes$ denotes the Kronecker product. The quadratic reward function is

$$\begin{aligned} R_t(\mathbf{s}, \mathbf{a}) &= (\mathbf{s}-\mathbf{z})^\mathsf{T}\mathbf{M}_t(\mathbf{s}-\mathbf{z}) + \mathbf{a}^\mathsf{T}\mathbf{H}_t\mathbf{a}, \\ R_T(\mathbf{s}) &= (\mathbf{s}-\mathbf{z})^\mathsf{T}\mathbf{M}_T(\mathbf{s}-\mathbf{z}), \end{aligned}$$

with appropriately sized matrices $\mathbf{M}_t, \mathbf{H}_t, \mathbf{M}_T$ and goal state vector $\mathbf{z}$ assumed given. From the policy optimization step, we obtain a linear Gaussian policy

$$\pi_t(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mathbf{a}|\mathbf{K}_t^\pi\mathbf{s} + \mathbf{k}_t^\pi, \boldsymbol{\Sigma}_{a,t}^\pi).$$

We also assume that the value function will remain quadratic through the backward pass and that the state distribution will remain Gaussian through the forward pass. These assumptions allow us to write them as

$$\begin{aligned} V_{t+1}(\mathbf{s}) &= \mathbf{s}^\mathsf{T}\mathbf{V}_{t+1}\mathbf{s} + \mathbf{s}^\mathsf{T}\mathbf{v}_{t+1} + v_{t+1}, \\ \mu_t(\mathbf{s}) &= \mathcal{N}(\mathbf{s}|\boldsymbol{\tau}_{s,t}^\mu, \boldsymbol{\Sigma}_{s,t}^\mu) \end{aligned}$$

for all timesteps. As we will show below, the assumption on the value function is warranted. The parameter distribution however leads to nonlinear dynamics, which in turn make it necessary to propagate the state distribution approximately for it to remain Gaussian. Lastly, we will use a Gaussian

$$\hat{p}_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_t, \hat{\boldsymbol{\Sigma}}_{\theta,t})$$

to express the prior believe the parameter distribution.

3.3.1. Backward Pass

We will now derive the backward pass for the linear quadratic Gaussian case in closed form. Details are given in the appendix B. To evaluate the integral given in the backward pass (3.5) in closed form, we first need to evaluate the worst-case parameter distribution $p_t(\boldsymbol{\theta})$. For the integral in equation (3.4), we find

$$\int_{\mathbf{s}} \mu_t(\mathbf{s}) Q_t(\mathbf{s}, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{s} = (\boldsymbol{\theta} - \mathbf{w}_t)^\mathsf{T} \mathbf{W}_t (\boldsymbol{\theta} - \mathbf{w}_t) + c_{w,t},$$

in which

$$\begin{aligned} \mathbf{W}_t &= \left(\boldsymbol{\tau}_{s,t}^{\mu} (\boldsymbol{\tau}_{s,t}^{\mu})^\mathsf{T} + \boldsymbol{\Sigma}_{s,t}^{\mu} \right) \otimes \mathbf{V}_{t+1}, \\ \mathbf{w}_t &= -\mathrm{vec}(\mathbf{b}_t \mathbf{K}_t^{\pi}) - \mathbf{W}_t^{-1} (\boldsymbol{\tau}_{s,t}^{\mu} \otimes I_n)^\mathsf{T} \left(\mathbf{V}_{t+1} (\mathbf{b}_t \mathbf{k}_t^{\pi} + \mathbf{c}_t) + \frac{1}{2} \mathbf{v}_{t+1} \right) \end{aligned} \tag{3.8}$$

and $c_{w,t}$ is a constant term. For the new parameter distribution, we then get

$$p_t(\boldsymbol{\theta}) \propto \hat{p}_t(\boldsymbol{\theta}) \exp\left(-\frac{1}{\alpha_t} (\boldsymbol{\theta} - \mathbf{w}_t)^\mathsf{T} \mathbf{W}_t (\boldsymbol{\theta} - \mathbf{w}_t) \right).$$

Plugging in the Gaussian prior distribution, we find

$$p_t(\boldsymbol{\theta}) \propto \exp\left(-\frac{1}{2} \left((\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t)^\mathsf{T} \hat{\boldsymbol{\Sigma}}_{\theta,t}^{-1} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t) + (\boldsymbol{\theta} - \mathbf{w}_t)^\mathsf{T} \left(\frac{\alpha_t}{2} \mathbf{W}_t^{-1} \right)^{-1} (\boldsymbol{\theta} - \mathbf{w}_t) \right) \right),$$

which will be Gaussian again, if the resulting quadratic form in the exponent is positive definite. We can express this condition as

$$\hat{\boldsymbol{\Sigma}}_{\theta,t}^{-1} + \frac{2}{\alpha_t} \mathbf{W}_t \succ 0. \tag{3.9}$$

If the above is fulfilled, we can get the updated parameter distribution

$$p_t(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_{\theta,t}, \boldsymbol{\Sigma}_{\theta,t} \right), \tag{3.10}$$

where

$$\begin{aligned} \boldsymbol{\Sigma}_{\theta,t} &= \left(\hat{\boldsymbol{\Sigma}}_{\theta,t}^{-1} + \frac{2}{\alpha_t} \mathbf{W}_t \right)^{-1} \\ \boldsymbol{\mu}_{\theta,t} &= \boldsymbol{\Sigma}_{\theta,t} \left(\hat{\boldsymbol{\Sigma}}_{\theta,t}^{-1} \hat{\boldsymbol{\theta}}_t + \frac{2}{\alpha_t} \mathbf{W}_t \mathbf{w}_t \right). \end{aligned}$$

Finally, we get the backward pass in closed form as

$$V_t(\mathbf{s}) = \mathbf{s}^\mathsf{T} \mathbf{V}_t \mathbf{s} + \mathbf{s}^\mathsf{T} \mathbf{v}_t + v_t$$

which is again quadratic in the state with the iteration for the quadratic term given by

$$\mathbf{V}_t = \mathbf{M}_t + (\mathbf{K}_t^{\pi})^\mathsf{T} \mathbf{H}_t \mathbf{K}_t^{\pi} + (\bar{\boldsymbol{\Theta}} + \mathbf{b}_t \mathbf{K}_t^{\pi})^\mathsf{T} \mathbf{V}_{t+1} (\bar{\boldsymbol{\Theta}} + \mathbf{b}_t \mathbf{K}_t^{\pi}) + \mathbf{P}_t. \tag{3.11}$$

Here, $\bar{\boldsymbol{\Theta}}_t$ is defined by $\boldsymbol{\mu}_{\theta,t} = \mathrm{vec}(\bar{\boldsymbol{\Theta}}_t)$.

To understand if the condition given in (3.9) can be violated, we will consider the definiteness of the involved matrices. Since the formulation here is with respect to a reward, the matrices $\mathbf{M}_t$ and $\mathbf{R}_t$ will be negative definite, thereby yielding a reward function concave in state and action. For a negative semi-definite matrix $\mathbf{A}$, $\mathbf{B}^\mathsf{T}\mathbf{A}\mathbf{B}$ is negative semi-definite for any matrix $\mathbf{B}$, which is easily seen from

$$\mathbf{x}^\mathsf{T}\mathbf{B}^\mathsf{T}\mathbf{A}\mathbf{B}\mathbf{x} = (\mathbf{B}\mathbf{x})^\mathsf{T}\mathbf{A}(\mathbf{B}\mathbf{x}).$$

Hence, at least the first three terms in calculating $\mathbf{V}_t$ in (3.11) will be negative semi-definite. While this does not allow us to judge the definiteness of $\mathbf{W}_t$ at each timestep, it at least gives some idea that $\mathbf{V}_t$ may remain negative definite throughout the backward pass. From equation (3.8) we can then see that for a negative definite $\mathbf{V}_{t+1}$, also $\mathbf{W}_t$ can be negative definite. An example easily illustrates this connection: if $\boldsymbol{\tau}^{\mu}_{s,t}$ is a vector of zeros, the eigenvalues of $\mathbf{W}_t$ are all products of eigenvalues of $\mathbf{V}_{t+1}$ (negative) and eigenvalues of $\boldsymbol{\Sigma}^{\mu}_{s,t}$ (positive) and thus all negative [54].

If $\mathbf{W}_t$ is negative definite, there can be some $\alpha_t > 0$ for which the condition given in equation (3.9) is no longer met. This fact can again be demonstrated by example: for $\boldsymbol{\Sigma}_{\theta,t} = \mathbf{I}_n$ and $\mathbf{W}_t = -\mathbf{I}_n$, any $\alpha_t \geq 1$ leads to a violation of the condition. Overall, we can therefore not assume that the condition is always met and indeed observe its violation in the practical implementation. In the following, we try to give a somewhat better understanding of what this means in general and why a solution to the inner minimization problem might not exist.

3.3.2. Existence of the Worst-Case Distribution

As observed for the linear Gaussian case, calculation of the worst-case distribution does not always yield a reasonable result. Going back to the general form of the worst-case distribution

$$p_t(\boldsymbol{\theta}) \propto \hat{p}_t(\boldsymbol{\theta}) \exp\left(-\frac{1}{\alpha_t}\int_{\mathbf{s}} \mu_t(\mathbf{s}) Q_t(\mathbf{s}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{s}\right),$$

we can give some intuition on what makes the solution diverge in some cases and what factors influence it. We obtain the new parameter distribution by reweighting the previously assumed distribution. Since the exponent has negative sign and $\alpha_t > 0$, parameter values which lead to a large value of the adversary state-action value function $Q(\mathbf{s}, \boldsymbol{\theta})$ will have lowered probability in the new distribution and vice versa. The problem described above for the Gaussian case emerges if the distribution $\hat{p}_t(\boldsymbol{\theta})$ has infinite support and decreases slower when approaching infinity than the reweighting term increases. In that case, the whole expression goes to infinity for large $\boldsymbol{\theta}$, which is clearly nonsensical.

Since the integral in the reweighting exponential is an expectation with respect to the current state distribution, the exponent will be larger if the state distribution puts weight on low reward regions of the state space. The state distribution in turn depends on how good the current policy is, as a completely converged policy will focus the state distribution to high reward regions and decrease its variance.

Another factor influencing the magnitude of the exponent is the Lagrange multiplier α_t. Its value at the optimum will depend on the KL bound set for the parameter distribution. Increasing the value of α_t, the exponent becomes smaller and thus $p_t(\boldsymbol{\theta})$ moves closer to $\hat{p}_t(\boldsymbol{\theta})$. Setting a large bound ε_p then corresponds to a lower value of α_t at optimum and larger exponent.

Lastly, the choice of $\hat{p}_t(\boldsymbol{\theta})$ plays an important role. If it decreases very quickly, the parameter values extremely far from the nominal ones will be small even when reweighted with a large exponential. In the limit where

we have perfect parameter knowledge, the distribution will become a Dirac delta function centered at the nominal value and all other values will still have zero probability, no matter the weight they are assigned.

The interplay of the above factors makes it difficult to see directly if the worst-case parameter distribution exists or if the solution diverges. In our setting we assume $\hat{p}_t(\boldsymbol{\theta})$ to be learned from data. If we can not find a solution, we have the options to either decrease the size of the ambiguity set or to first find a better policy. Both options limit how different the worst-case distribution can be from the nominal one and thereby reduce the robustness we can expect from the solution. In the case of the policy, this happens by limiting the region of the state space which the adversary can explore to one of generally high reward.

3.3.3. Forward Pass

For the forward pass as given in equation (3.7), we need to propagate a Gaussian state distribution

$$\mu_t(\mathbf{s}) = \mathcal{N}(\mathbf{s}|\boldsymbol{\tau}^{\mu}_{s,t}, \boldsymbol{\Sigma}^{\mu}_{s,t})$$

through linear dynamics with random parameters obeying the distribution given in equation (3.10). We can find the forced dynamics under the Gaussian parameter distribution from

$$\begin{aligned}\mathcal{P}_t(\mathbf{s}'|\mathbf{s}) &= \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \int_{\boldsymbol{\theta}} \mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}) p_t(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \\ &= \mathcal{N}\Big(\mathbf{s}'|(\bar{\boldsymbol{\Theta}}_t + \mathbf{b}_t\mathbf{K}^{\pi}_t)\mathbf{s} + \mathbf{b}_t\mathbf{k}^{\pi}_t + \mathbf{c}_t, \boldsymbol{\Sigma}_{s',t} + \mathbf{b}_t\boldsymbol{\Sigma}_{a,t}\mathbf{b}_t^{\mathsf{T}} + (\mathbf{s}^{\mathsf{T}}\otimes\mathbf{I}_n)\boldsymbol{\Sigma}_{\theta,t}(\mathbf{s}^{\mathsf{T}}\otimes\mathbf{I}_n)^{\mathsf{T}}\Big).\end{aligned} \tag{3.12}$$

These dynamics seem again linear in the state, but the noise covariance

$$\boldsymbol{\Sigma}_t(\mathbf{s}) = \boldsymbol{\Sigma}_{s',t} + \mathbf{b}_t\boldsymbol{\Sigma}_{a,t}\mathbf{b}_t^{\mathsf{T}} + (\mathbf{s}^{\mathsf{T}}\otimes\mathbf{I}_n)\boldsymbol{\Sigma}_{\theta,t}(\mathbf{s}^{\mathsf{T}}\otimes\mathbf{I}_n)^{\mathsf{T}}$$

is also state-dependent. The resulting distribution

$$\mu_t(\mathbf{s}') = \int_{\mathbf{s}} \mu_t(\mathbf{s})\mathcal{P}_t(\mathbf{s}'|\mathbf{s})\,\mathrm{d}\mathbf{s}$$

will generally not be Gaussian. Here, we will however assume that it can be approximated well enough by a Gaussian which we will obtain in a computationally efficient manner using the unscented transform.

The unscented transform is typically used to calculate the statistics of a random variable after propagation through nonlinear dynamics [55]. For the case of state dependent noise covariance, we can reformulate our problem in a way to make it applicable. First, we write the dynamics equivalently as

$$\mathbf{s}' = (\bar{\boldsymbol{\Theta}}_t + \mathbf{b}_t\mathbf{K}^{\pi}_t)\mathbf{s} + \mathbf{b}_t\mathbf{k}^{\pi}_t + \mathbf{c}_t + \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_t(\mathbf{s})).$$

We then rewrite the noise term as

$$\boldsymbol{\xi}_t = \sqrt{\boldsymbol{\Sigma}_t(\mathbf{s})}\boldsymbol{\zeta}_t, \quad \boldsymbol{\zeta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

giving us

$$\mathbf{s}' = (\bar{\boldsymbol{\Theta}}_t + \mathbf{b}_t\mathbf{K}^{\pi}_t)\mathbf{s} + \mathbf{b}_t\mathbf{k}^{\pi}_t + \mathbf{c}_t + \sqrt{\boldsymbol{\Sigma}_t(\mathbf{s})}\boldsymbol{\zeta}_t.$$

Here, we take the matrix square root to indicate the lower triangular Cholesky factor. These dynamics can now be interpreted as a nonlinear noise-free dynamics in the augmented state $\mathbf{s}^a = [\mathbf{s}_t^{\mathsf{T}} \quad \boldsymbol{\zeta}_t^{\mathsf{T}}]^{\mathsf{T}}$. Augmenting the state by the process noise is a standard procedure given in [55] and suggested for dealing with state

dependent noise in [56]. Mean and covariance of the augmented state distribution, which we then need to propagate through these dynamics, are given by

$$\bar{\mathbf{s}}_t^a = \begin{bmatrix} \boldsymbol{\tau}_{s,t}^{\mu} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\Sigma}_t^a = \begin{bmatrix} \boldsymbol{\Sigma}_{s,t}^{\mu} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

The propagation can now be straightforwardly performed using unscented or cubature transform. In this work, we use the cubature transform as given in [57], which is a special case of the unscented transform without additional parameters to choose. The whole algorithm for one step of the cubature forward pass is given in algorithm 2. Since the forward pass stems from the optimality condition (3.6), solving it approximately results in fulfilling this condition only approximately and thereby having no guarantee of optimality.

input : $f_t(\mathbf{s})$ // system dynamics
n // state dimension
$\boldsymbol{\Sigma}_t(\mathbf{s})$ // state dependent noise covariance
$\boldsymbol{\tau}_{s,t}^{\mu}, \boldsymbol{\Sigma}_{s,t}^{\mu}$ // state distribution mean and covariance

output : $\boldsymbol{\tau}_{s,t+1}^{\mu}, \boldsymbol{\Sigma}_{s,t+1}^{\mu}$ // next state distribution mean and covariance

begin

// augmented dynamics using lower triangular Cholesky factor for matrix square root
$f_t^a(\mathbf{s}^a) \leftarrow f_t(\mathbf{s}_{1\dots n}^a) + \sqrt{\boldsymbol{\Sigma}_t(\mathbf{s}_{1\dots n}^a)}\mathbf{s}_{n+1\dots 2n}^a$

// form augmented state distribution parameters
$\bar{\mathbf{s}}_t^a \leftarrow \begin{bmatrix} \boldsymbol{\tau}_{s,t}^{\mu} \\ \mathbf{0} \end{bmatrix}$
$\boldsymbol{\Sigma}_t^a \leftarrow \begin{bmatrix} \boldsymbol{\Sigma}_{s,t}^{\mu} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$

// calculate lower triangular Cholesky factor of covariance
$\mathbf{P} \leftarrow \sqrt{\boldsymbol{\Sigma}_t^a}$

// form cubature points, matrix subscript indicates column
$\mathbf{x}_i \leftarrow \sqrt{n}(\bar{\mathbf{s}}_t^a + \mathbf{P}_i) \quad i = 1 \dots n$
$\mathbf{x}_i \leftarrow \sqrt{n}(\bar{\mathbf{s}}_t^a - \mathbf{P}_{n-i}) \quad i = n+1 \dots 2n$

// propagate cubature points through augmented dynamics
$\mathbf{x}_i \leftarrow f_t^a(\mathbf{x}_i)$

// estimate new augmented mean and covariance
$\bar{\mathbf{s}}_t^a \leftarrow \frac{1}{2n}\sum_{i=1}^{2n} \mathbf{x}_i$
$\boldsymbol{\Sigma}_t^a \leftarrow \frac{1}{2n}\sum_{i=1}^{2n} (\mathbf{x}_i - \bar{\mathbf{s}}_t^a)(\mathbf{x}_i - \bar{\mathbf{s}}_t^a)^{\mathsf{T}}$

// extract approximate mean and covariance for next state
$\boldsymbol{\tau}_{s,t+1}^{\mu} \leftarrow (\bar{\mathbf{s}}_t^a)_{1\dots n}$
$\boldsymbol{\Sigma}_{s,t+1}^{\mu} \leftarrow (\Sigma_t^a)_{1\dots n,1\dots n}$

Algorithm 2: Pseudo code for cubature transform method of propagating a Gaussian state distribution through dynamics with state dependent noise covariance

3.3.4. Implementation

We will now give an overview of the robust trajectory optimization algorithm as implemented for this work in the *Julia* programming language [58]. Here, we assume that linearized dynamics $\mathcal{P}_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\theta})$ together with a parameter distribution $\hat{p}_t(\boldsymbol{\theta})$ at each timestep are given. These could be obtained by sampling a number of trajectories under the current policy and fitting linear Gaussian dynamics to subsets of these trajectories. The variation of dynamics parameters between these fits can then be used to estimate the parameter distribution.

As mentioned in section 3.3.1, there is a balance between robustness and solvability of the problem. We will iterate between policy and parameter distribution update steps. For the parameter distribution update to work, we need a somewhat good but not completely converged policy. Thus, we start with a policy update step and then iterate. For each update step of the parameter distribution, a certain number of policy update steps is performed. This number, together with the KL bound between iterations, has to be set in a way to ensure the policy does not converge completely. Even for the Gaussian case, we have no simple expression to check whether the condition (3.9) is met for all timesteps. Instead, the parameter distribution update step is started and, if it fails due to the condition not being met, restarted with the KL bound on the parameter distribution reduced by a predefined factor. As the bound on the parameter distribution is always with respect to the distribution estimated from data, the parameter distribution update for a given policy should converge in a single iteration.

Both policy and parameter distribution update steps consist in performing a backward and a forward pass which are obtained by solving the optimality conditions of the respective optimization problem. For the Lagrange multiplier corresponding to the KL constraint, we use an iterative gradient-based method such as *BFGS* provided by the *Optim.jl* package [59].

The policy update step is based on the algorithm for GPS given in [49]. Since in our case the dynamics have state-dependent noise as seen in equation (3.12), we need to adapt it slightly. Firstly, in the backwards pass for computing the policy value function, we need to use the update rule as given in equation (3.11) and secondly, we have to perform the forward pass approximately using the cubature transform. This step is detailed in algorithm 2, while the whole distributionally robust trajectory optimization algorithm is described in algorithm 3. Backward passes of both policy and parameter optimization step are obtained in closed form and the detailed equations given in appendices B and C.

It also needs to be noted that in our implementation we use time-independent Lagrange multipliers α_π and α_p for the KL constraints of policy and parameter distribution respectively, which amounts to enforcing a constraint on the sum of KL-divergence over all timesteps instead of individual constraints at each timestep.

input : T // time horizon
$\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta})$ // linearized dynamics
$q_t(\mathbf{a}|\mathbf{s})$ // initial policy
$\hat{p}_t(\boldsymbol{\theta})$ // empirical parameter distribution
$\mu_1(\mathbf{s})$ // initial state distribution
$R_t(\mathbf{s},\mathbf{a})$ // quadratic reward function
N_π // number of policy updates
$\varepsilon_\pi, \varepsilon_p$ // KL bounds
λ_p // KL bound reduction factor

output : $\pi_t(\mathbf{a}|\mathbf{s})$ // optimal robust policy
$p_t(\boldsymbol{\theta})$ // worst-case parameter distribution
α_π, α_p // optimal Lagrange multipliers

initialize : α_π, α_p // initial guess for Lagrange multipliers

while *not converged* **do**
  for $i \leftarrow 1$ **to** N_π **do**
    while $L_\pi(\mu_t, V_t, \alpha_\pi)$ *is not at minimum* **do**
      // perform backward pass according to equations in section C.1
      $\left[\pi_t(\mathbf{a}|\mathbf{s}), V_t(\mathbf{s})\right] \leftarrow \text{policy\_backward\_pass}(q_t(\mathbf{a}|\mathbf{s}), p_t(\boldsymbol{\theta}), R_t(\mathbf{s},\mathbf{a}), \mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}), \alpha_\pi)$

      // cubature forward pass as given in algorithm 2
      $\mu_t(\mathbf{s}) \leftarrow \text{cubature\_forward\_pass}(\mu_1(\mathbf{s}), p_t(\boldsymbol{\theta}), \pi_t(\mathbf{a}|\mathbf{s}), \mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}))$

      // compute objective and gradient with equations in section C.2
      $L_\pi(\mu_t, V_t, \alpha_t) \leftarrow \text{policy\_dual\_value}(V_1(\mathbf{s}), \mu_1(\mathbf{s}), \alpha_\pi, \varepsilon_\pi)$
      $\frac{\partial L_\pi}{\partial \alpha_\pi} \leftarrow \text{policy\_dual\_gradient}(\mu_t(\mathbf{s}), \pi_t(\mathbf{a}|\mathbf{s}), q_t(\mathbf{a}|\mathbf{s}), \varepsilon_\pi)$

      // update Lagrange multiplier
      $\alpha_\pi \leftarrow \text{BFGS\_step}(L_\pi(\mu_t, V_t, \alpha_t), \frac{\partial L_\pi}{\partial \alpha_\pi})$
    $q_t(\mathbf{a}|\mathbf{s}) \leftarrow \pi_t(\mathbf{a}|\mathbf{s})$
  while $L_p(\mu_t, V_t, \alpha_p, p_t, \hat{p}_t)$ *is not at maximum* **do**
    // perform backward pass according to equations in section B.1
    $\left[p_t(\boldsymbol{\theta}), V_t(\mathbf{s})\right] \leftarrow \text{parameter\_backward\_pass}(\hat{p}(\boldsymbol{\theta}), \mu(\mathbf{s}), \mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}), \pi(\mathbf{a}|\mathbf{s}), \alpha_p)$

    // cubature forward pass as given in algorithm 2
    $\mu_t(\mathbf{s}) \leftarrow \text{cubature\_forward\_pass}(\mu_1(\mathbf{s}), p_t(\boldsymbol{\theta}), \pi_t(\mathbf{a}|\mathbf{s}), \mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta}))$

    // compute objective and gradient with equations in section B.2
    $L_p(\mu_t, V_t, \alpha_p, p_t, \hat{p}_t) \leftarrow \text{parameter\_dual\_value}(V_1(\mathbf{s}), \mu_1(\mathbf{s}), \alpha_p, \varepsilon_p, p_t, \hat{p}_t)$
    $\frac{\partial L_p}{\partial \alpha_p} \leftarrow \text{parameter\_dual\_gradient}(p_t(\boldsymbol{\theta}), \hat{p}_t(\boldsymbol{\theta}), \varepsilon_p)$

    // update Lagrange multiplier
    $\alpha_p \leftarrow \text{BFGS\_step}(L_p(\mu_t, V_t, \alpha_p, p_t, \hat{p}_t), \frac{\partial L_p}{\partial \alpha_p})$
    if *iteration failed* **then**
      $\varepsilon_p \leftarrow \lambda_p \varepsilon_p$

Algorithm 3: Pseudo code for KL distributionally robust trajectory optimization

3.3.5. Evaluation

In this section, we will evaluate the performance of the algorithm introduced above. We here limit the analysis to linear time invariant systems with a given nominal parameter distribution and focus on showing the difference between robust and nominal controller as well as the influence of hyperparameters on the solvability and robustness of the resulting controller.

Results on Example System

We consider an unstable example system with state dimension $n = 2$, action dimension $m = 1$. The goal is to drive the system state to $\mathbf{z} = [10\,10]^\mathsf{T}$ over a time horizon of $T = 20$. We furthermore assume an estimated parameter distribution with covariance matrix $\hat{\Sigma} = 10^{-4}\mathbf{I}_{n^2}$. Algorithm 3 is used to obtain a robust controller, while the version of GPS adapted to account for the parameter distribution in both forward and backward pass is used as a comparison. For the policy update step, we set $\varepsilon_\pi = 150$ and run $N_\pi = 1$ iteration for each update of the parameter distribution. In this case, four iterations between policy and parameter update steps are enough for convergence. In the nominal case without parameter updates, we also iterate the policy update step four times with $\varepsilon_\pi = 150$. The KL bound for the parameter distribution is initially set to $\varepsilon_p = 50$ but reduced during the optimization to $\varepsilon_p = 4.92$ to make the problem solvable as discussed in 3.3.4.

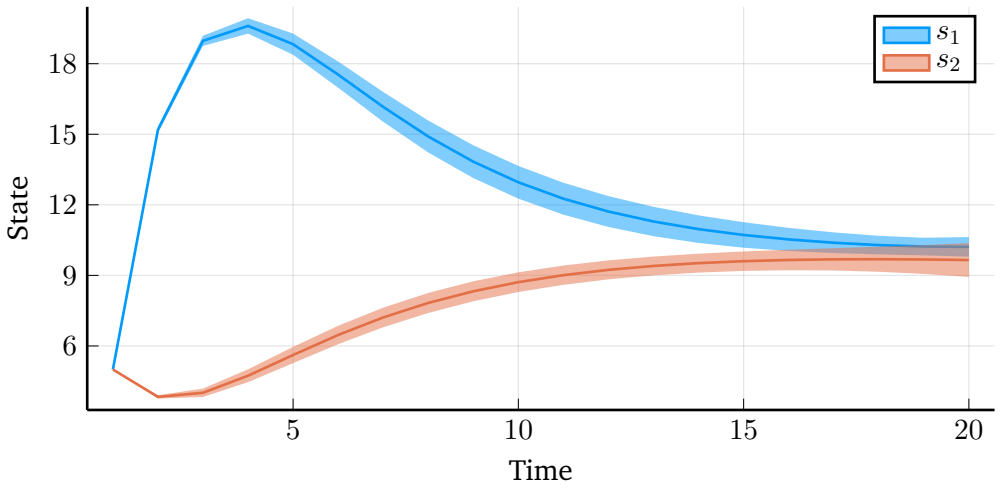


Figure 3.1.: State distribution of example system under nominal parameter distribution over time with controller obtained by solving the modified version of GPS including forward pass by cubature transform and backward pass with parameter distribution dependent terms.

Figure 3.1 shows the state distribution under the nominal parameter distribution and controller over time. The controller is able to steer the system towards the goal state with the remaining variance due to the variance in the nominal parameter distribution.

In figure 3.2, the KL-divergence between the two parameter distributions is shown over time. Since the bound here is on the sum of the KL-divergence over all timesteps, the resulting worst-case distribution can vary with time. Clearly for this particular case, the strongest influence on the trajectory reward is possible in the first few timesteps, where the worst-case distribution differs most from nominal one.

To evaluate the performance of nominal and robust controller under both the prior and worst-case parameter distribution, 10000 trajectory rollouts using both controllers are collected and the resulting trajectory reward

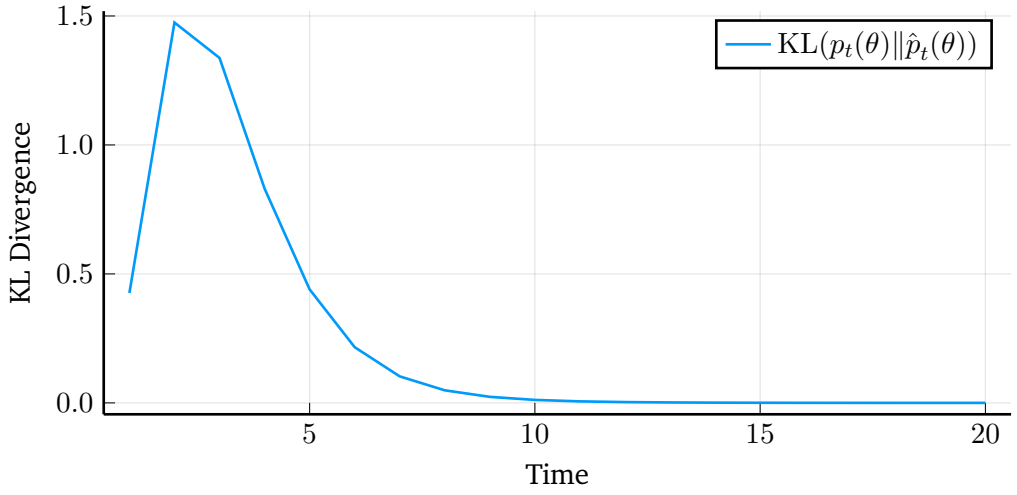


Figure 3.2.: KL-divergence between worst-case and nominal parameter distribution over the trajectory. The constraint on its sum was iteratively adapted to $\varepsilon_p = 4.92$ to make the problem solvable, as indicated in algorithm 3.

distributions compared. For the rollouts, the dynamics matrix multiplying the state is at each timestep sampled from the respective parameter distribution. Figure 3.3 shows estimated densities of the resulting reward distributions. The rightmost curve with the highest average reward corresponds to the nominal controller and dynamics sampled from the nominal parameter distribution. For the same controller and dynamics sampled from the worst-case distribution, the leftmost curve indicates clearly deteriorated performance. On the other hand, the distributionally robust controller obtained by our algorithm shows higher average reward in the worst case it was optimized for, while being worse in the nominal case.

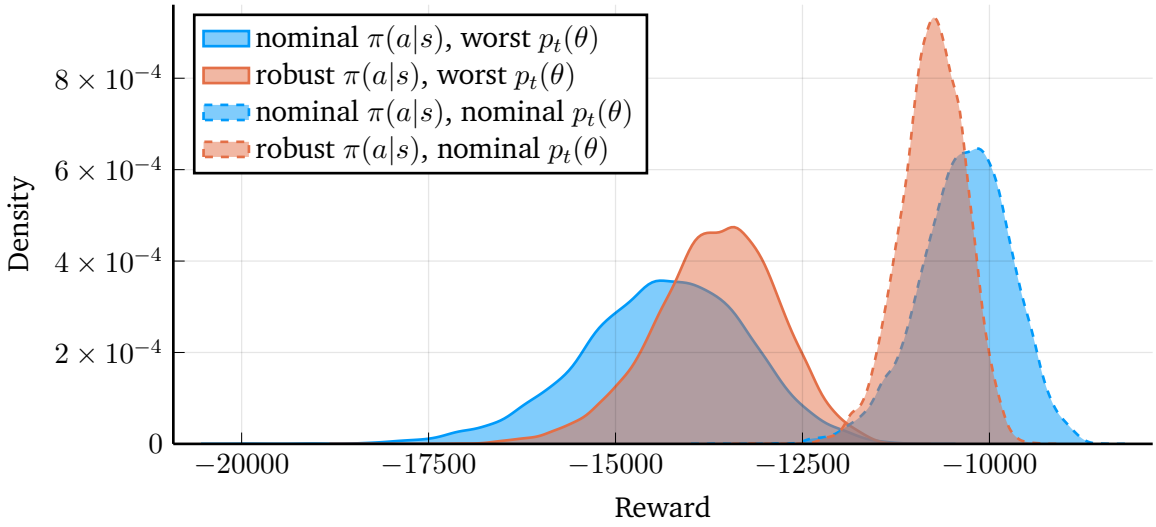


Figure 3.3.: Kernel density estimates of trajectory reward distribution for $N_{\text{traj}} = 10000$ trajectories obtained by sampling dynamics parameters and control from the respective distributions. The robust controller increases worst-case average performance while lowering the average reward in the nominal case.

Results for Randomly Created Systems

Instead of measuring performance on a single example, here we generate 50 random linear unstable systems of state dimension $n = 2$ and action dimension $m = 1$ and compare robust and nominal performance. Figure 3.4 shows the relative improvement in expected trajectory reward of the robust controller for all randomly generated systems. For each system, the average trajectory reward was estimated from 1000 rollouts under the respective policy and parameter distribution. The left histogram shows that under the worst case distribution, the difference is positive for all 50 systems, indicating superior performance of the robust controller. As would be expected, under the nominal distribution, the robust controller underperforms, hence the negative difference in the right histogram. Both plots show rather large peaks close to zero, which means essentially no performance difference between the two controllers. As in this case we again iteratively lowered the KL bound on the worst-case parameter distribution when the optimization failed, one explanation would be that for some systems this needed to happen more times, leading to less robustness and difference in performance.

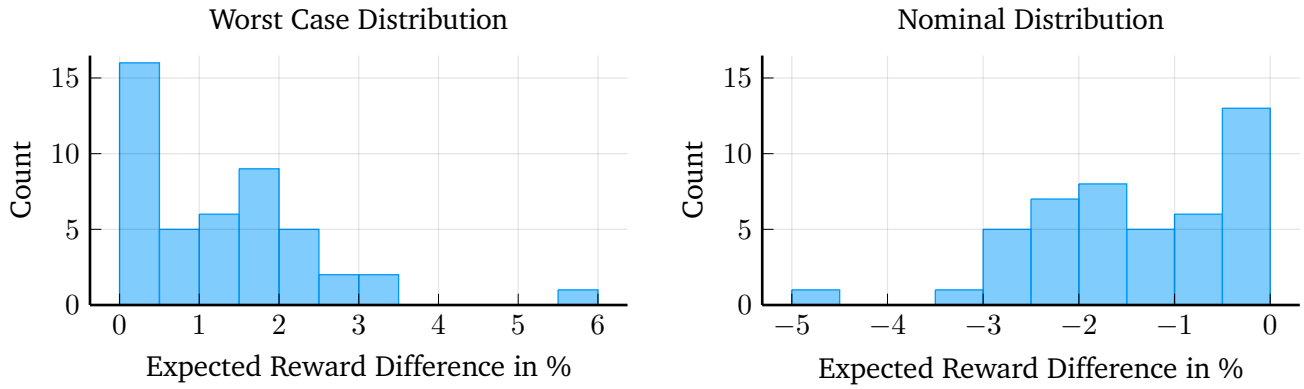


Figure 3.4.: Relative increase in expected trajectory reward by robust over nominal policy. The histogram shows the average value over 1000 trajectories for 50 randomly generated system dynamics. Under the worst-case distribution (left), the robust controller yields better reward while the opposite is true under the nominal one.

In figure 3.5, the wide spread along the x-axis shows that for many of the systems evaluated, the KL bound had to be lowered from the initially set value of $\varepsilon_p = 20$. However, a larger final bound does not seem to correspond directly to bigger performance increase when evaluated under the worst-case distribution. Hence, other factors such as the particular system and input matrices, initial or goal state we generated randomly for all 50 systems seem to influence the relation between KL bound and improvement in worst-case performance rather strongly.

Lastly, we evaluate how the results change when increasing the policy step size per update step of the parameter distribution. For the above results, a value of $\varepsilon_p = 100$ was used, which we will now compare to $\varepsilon_p = 150$. Since this means the policy will converge more before each parameter distribution update, following the discussion in section 3.3.2, less difference between performance of nominal and robust controller should be the result.

Figure 3.6 compares the improvement of robust over nominal policy for the two different parameter choices. Clearly the higher value leads to less difference between the two, confirming our statement.

The above evaluation shows that the robust controller can be successfully found by our algorithm and indicates that it can robustify against ambiguity in the parameter distribution. It is not yet clear from these results

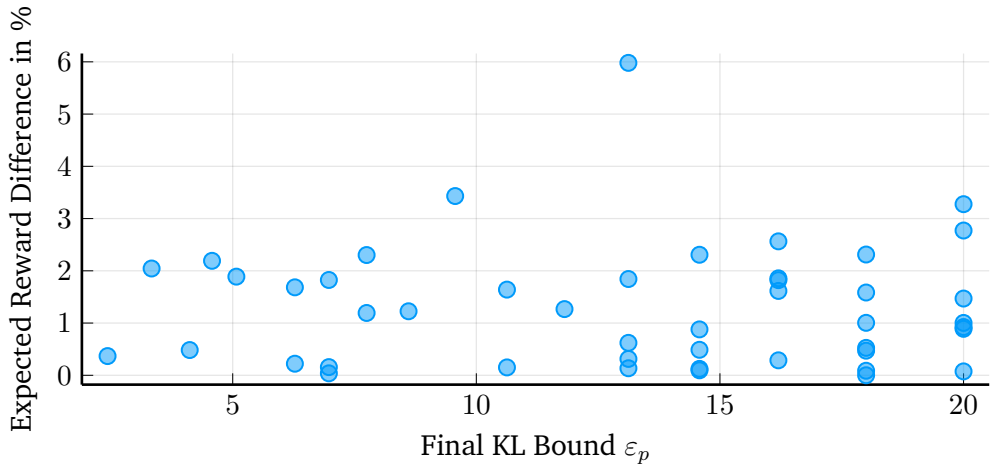


Figure 3.5.: Influence of final KL bound on parameter distribution on the difference in performance of the two controllers under the worst-case distribution.

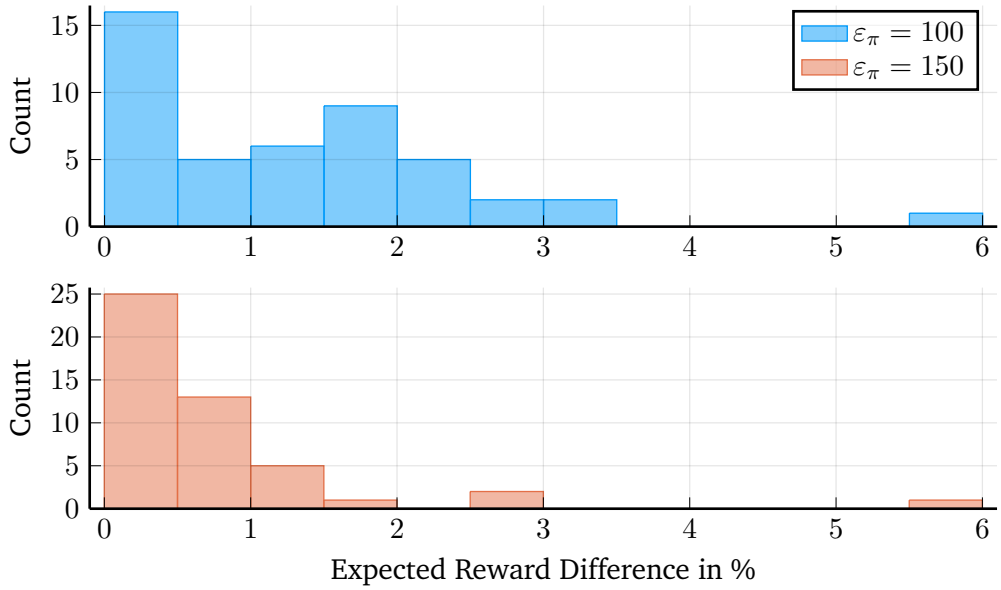


Figure 3.6.: Comparison of worst-case improvement for different values of policy KL bound. Counts indicate the number of random systems for which the expected reward was estimated by averaging over 1000 trajectories. A higher bound (bottom) leads to the robust policy showing less difference to the nominal one.

if the robustness to this ambiguity will provide an advantage for more realistic problems. From evaluating on randomly generated systems, it became already clear however that there is quite a lot of variance in the attainable robustness depending on the system parameters. We therefore suspect the usefulness of our approach will depend on the problem characteristics also in general.

3.4. Future Work

In the following we give some ideas on how the distributionally robust trajectory optimization algorithm presented here can be applied and the extensions we deem necessary.

Evaluation in Model-Based RL Setting

Since the goal of the algorithm developed in this chapter is robustifying the trajectory optimization approach against bad dynamics estimates learned from data, the next important step is to test it on actual learned data from a nonlinear system. In each iteration, we would fit linear time varying dynamics to a set of trajectories and a time varying quadratic reward function to the possibly nonconvex reward. These can then straightforwardly be utilized in our algorithm. The only addition to standard methods would be the estimation of the nominal parameter distribution $\hat{p}_t(\boldsymbol{\theta})$ from a set of collected trajectories, for example by Bayesian linear regression. With these extensions, we could run the algorithm for a completely unknown system in a model-based reinforcement learning setting with the model being learned online. Incorporating this modified trajectory optimization into an approach such as GPS would then allow learning arbitrarily parametrized policies.

Additionally, the algorithm should be extended to include ambiguity not just in the system matrix parameters, but also the input matrix and additive component of the dynamics which typically all would be learned from trajectory data.

Application to Model Predictive Control

Another extension would be using the trajectory optimization approach in a model predictive control fashion. Instead of finding the controllers for the whole trajectory, we would optimize over a certain time horizon and only apply the control of the first timestep to the system which then leads to a new initial state. From there, the process is repeated hence closing the control loop.

Evaluation of Computational Cost

With the application to more complex systems it would also be important to quantify the increased computational cost due to the additional optimization of the parameter distribution for each policy update step.

Different Ambiguity Sets

While the choice of KL-divergence together with the Gaussian assumption on the nominal parameter distribution allowed us closed form solutions of both backward pass and worst-case distribution, other nominal distributions and different types of ambiguity sets can also be considered. For example, other ϕ-divergences such as the reverse KL-divergence may lead to tractable problems with different solution properties.

Resolution of Solvability Problems

Lastly, to make application more straightforward and give some guarantees of what level of robustness in terms of KL-divergence can be achieved, the conditions for which the worst-case distribution exists need further study. While we dealt with this problem heuristically by reducing the KL bound when necessary, this is not satisfactory. A better understanding of influence of problem parameters as well as the choice of

hyperparameters is needed. This understanding might also allow moving to individual timestep bounds on the parameter distribution in the original derivation, a case for which our heuristic approach failed to solve the problem.

4. Conclusion

Distributionally robust optimization provides a principled approach of including ambiguity stemming from a limited amount of data into an optimization problem. In this thesis, we investigated two methods of applying it to the setting of continuous optimal control with the aim to account for model misspecification.

Our first approach, described in chapter 2, is an extension of LQR which is robust to ambiguity in the additive noise distribution. Instead of assuming a Gaussian distribution, we consider the case where some samples from the noise are given and the controller should be able to deal with all distributions which could have generated those samples. For this set of reasonable distributions, we choose a Wasserstein ball which can be constructed directly based on the sample distribution. The resulting controller is given by a tractable parametrized convex optimization problem, for which we find the parametrization by stochastic gradient descent. Evaluation on simple examples using the worst-case noise showed improved robustness in some cases, but also indicated strong dependence on the particular system dynamics and hyperparameter choice.

In chapter 3, we present a trajectory optimization approach based on linearized dynamics. The goal in this approach is to robustify the optimization against uncertainty in the parameters, which we assume come from fitting a linear model to trajectories. We consider all parameter distributions close to the nominal one in terms of Kullback-Leibler divergence and optimize the time-dependent controllers for the worst of these distributions by iterating between policy update steps and parameter distribution update. An evaluation on linear example systems shows improvement in the expected reward under the worst-case distribution. Existence of the worst-case distribution turns out to be a critical issue and is not guaranteed for too large ambiguity sets.

For both approaches presented, we give some ideas for future work and details on issues with the implementation. Further evaluation, especially of the second approach, in a setting where approximate dynamics are learned from data is necessary to judge the robustness and improvement over standard methods.

Bibliography

[1] Richard S. Sutton and Andrew Barto. *Reinforcement Learning : An Introduction*. Second edition. Adaptive Computation and Machine Learning Series. Cambridge, MA, 2018.

[2] David Silver et al. “Mastering the Game of Go without Human Knowledge”. In: *Nature* 550.7676 (7676 Oct. 2017), pp. 354–359.

[3] Henrik Olsson et al. “Friction Models and Friction Compensation”. In: *European Journal of Control* 4.3 (1998), pp. 176–195.

[4] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. “A Survey on Policy Search for Robotics”. In: *Foundations and Trends in Robotics* 2.1-2 (2011), pp. 1–142.

[5] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd edition. Madison, Wisconsin: Nob Hill Publishing, 2017. 623 pp.

[6] John T. Betts. “Survey of Numerical Methods for Trajectory Optimization”. In: *Journal of Guidance, Control, and Dynamics* 21.2 (Mar. 1998), pp. 193–207.

[7] Richard Bellman. *Dynamic Programming*. Princeton, NJ: Princeton Univ. Pr, 1984. 339 pp.

[8] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd. Vol. I. Belmont, MA, USA: Athena Scientific, 2005.

[9] Brian D. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice Hall Information and System Sciences Series. Englewood Cliffs, N.J: Prentice Hall, 1989. 380 pp.

[10] Benjamin Recht. “A Tour of Reinforcement Learning: The View from Continuous Control”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019), pp. 253–279.

[11] Weiwei Li and Emanuel Todorov. “Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems”. In: *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*. First International Conference on Informatics in Control, Automation and Robotics. Setúbal, Portugal: SciTePress - Science and and Technology Publications, 2004, pp. 222–229.

[12] Lars Peter Hansen and Thomas J. Sargent. *Robustness*. Princeton, N.J.: Princeton University Press, 2008. 455 pp.

[13] Ian R. Petersen and Roberto Tempo. “Robust Control of Uncertain Systems: Classical Results and Recent Developments”. In: *Automatica* 50.5 (May 1, 2014), pp. 1315–1335.

[14] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovskiĭ. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton: Princeton University Press, 2009. 542 pp.

[15] Dimitris Bertsimas and Melvyn Sim. “The Price of Robustness”. In: *Operations research* 52.1 (2004), pp. 35–53.

[16] Hamed Rahimian and Sanjay Mehrotra. *Distributionally Robust Optimization: A Review*. Aug. 12, 2019. arXiv: 1908.05659.

[17] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[18] Solomon Kullback and Richard A Leibler. "On Information and Sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[19] Zhaolin Hu and L Jeff Hong. "Kullback-Leibler Divergence Constrained Distributionally Robust Optimization". In: *Optimization Online* (2012), p. 34.

[20] Leonid Kantorovitch. "On the Translocation of Masses". In: *Management Science* 5.1 (1958), pp. 1–4. JSTOR: 2626967.

[21] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport*. Apr. 24, 2019. arXiv: 1803.00567.

[22] Daniel Kuhn et al. "Wasserstein Distributionally Robust Optimization: Theory and Applications in Machine Learning". In: *Operations Research & Management Science in the Age of Analytics*. Ed. by Serguei Netessine, Douglas Shier, and Harvey J. Greenberg. INFORMS, Oct. 2019, pp. 130–166.

[23] Rui Gao and Anton J. Kleywegt. *Distributionally Robust Stochastic Optimization with Wasserstein Distance*. July 16, 2016. arXiv: 1604.02199.

[24] Jose Blanchet, Karthyek Murthy, and Fan Zhang. *Optimal Transport Based Distributionally Robust Optimization: Structural Properties and Iterative Schemes*. June 6, 2020. arXiv: 1810.02403.

[25] Huan Xu and Shie Mannor. "Distributionally Robust Markov Decision Processes". In: *Mathematics of Operations Research* 37.2 (May 2012), pp. 288–300.

[26] Insoon Yang. "A Convex Optimization Approach to Distributionally Robust Markov Decision Processes With Wasserstein Distance". In: *IEEE Control Systems Letters* 1.1 (July 2017), pp. 164–169.

[27] Zhi Chen, Pengqian Yu, and William B. Haskell. *Distributionally Robust Optimization for Sequential Decision Making*. Oct. 9, 2018. arXiv: 1801.04745.

[28] Bart Van Parys et al. "Distributionally Robust Control of Constrained Stochastic Systems". In: *IEEE Transactions on Automatic Control* (2015), pp. 1–1.

[29] Jeremy Coulson, John Lygeros, and Florian Dörfler. *Regularized and Distributionally Robust Data-Enabled Predictive Control*. Nov. 1, 2019. arXiv: 1903.06804.

[30] Insoon Yang. *Wasserstein Distributionally Robust Stochastic Control: A Data-Driven Approach*. Nov. 8, 2019. arXiv: 1812.09808.

[31] Insoon Yang. "Data-Driven Distributionally Robust Stochastic Control of Energy Storage for Wind Power Ramp Management Using the Wasserstein Metric". In: *Energies* 12.23 (Jan. 2019), p. 4577.

[32] Kihyun Kim and Insoon Yang. *Minimax Control of Ambiguous Linear Stochastic SystemsUsing the Wasserstein Metric*. Mar. 30, 2020. arXiv: 2003.13258.

[33] Peter Coppens, Mathijs Schuurmans, and Panagiotis Patrinos. *Data-Driven Distributionally Robust LQR with Multiplicative Noise*. Dec. 20, 2019. arXiv: 1912.09990.

[34] Erick Delage and Yinyu Ye. "Distributionally Robust Optimization Under Moment Uncertainty with Application to Data-Driven Problems". In: *Operations Research* 58.3 (June 2010), pp. 595–612.

[35] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Jan. 1994.

[36] Akshay Agrawal et al. "Learning Convex Optimization Control Policies". In: *Learning for Dynamics and Control*. 2020, pp. 361–373.

[37] Akshay Agrawal et al. "Differentiable Convex Optimization Layers". In: *Advances in Neural Information Processing Systems*. 2019, pp. 9562–9574.

[38] Steven Diamond and Stephen Boyd. “CVXPY: A Python-Embedded Modeling Language for Convex Optimization”. In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[39] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[40] Xinyue Shen et al. “Disciplined Convex-Concave Programming”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2016 IEEE 55th Conference on Decision and Control (CDC). Las Vegas, NV, USA: IEEE, Dec. 2016, pp. 1009–1014.

[41] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge, UK ; New York: Cambridge University Press, 2004. 716 pp.

[42] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. arXiv: 1412.6980.

[43] Rémi Flamary and Nicolas Courty. *POT Python Optimal Transport Library*. 2017.

[44] A. Ismael F. Vaz, Edite M. G. P. Fernandes, and M. Paula S. F. Gomes. “SIPAMPL: Semi-Infinite Programming with AMPL”. In: *ACM Transactions on Mathematical Software (TOMS)* 30.1 (Mar. 1, 2004), pp. 47–61.

[45] David Mayne. “A Second-Order Gradient Method for Determining Optimal Trajectories of Non-Linear Discrete-Time Systems”. In: *International Journal of Control* 3.1 (Jan. 1966), pp. 85–95.

[46] Emanuel Todorov and Weiwei Li. “A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems”. In: *Proceedings of the 2005, American Control Conference, 2005.* Proceedings of the 2005, American Control Conference, 2005. Portland, OR, USA: IEEE, 2005, pp. 300–306.

[47] Sergey Levine and Vladlen Koltun. “Guided Policy Search”. In: *International Conference on Machine Learning*. 2013, pp. 1–9.

[48] Sergey Levine and Pieter Abbeel. “Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 1071–1079.

[49] Hany Abdulsamad. “Stochastic Optimal Control with Linearized Dynamics”. Master’s thesis. Technical University of Darmstadt, 2016.

[50] Aharon Ben-Tal et al. “Robust Solutions of Optimization Problems Affected by Uncertain Probabilities”. In: *Management Science* 59.2 (Feb. 2013), pp. 341–357.

[51] Hongseok Namkoong and John C. Duchi. “Stochastic Gradient Methods for Distributionally Robust Optimization with F-Divergences”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 2208–2216.

[52] Elena Smirnova, Elvis Dohmatob, and Jérémie Mary. *Distributionally Robust Reinforcement Learning*. Feb. 22, 2019. arXiv: 1902.08708.

[53] Martin L. Puterman and Moon Chirl Shin. “Modified Policy Iteration Algorithms for Discounted Markov Decision Problems”. In: *Management Science* 24.11 (July 1978), pp. 1127–1137.

[54] Kathrin Schacke. “On the Kronecker Product”. Master’s thesis. University of Waterloo, Aug. 1, 2013.

[55] Eric A Wan, Rudolph Van Der Merwe, and Simon Haykin. “The Unscented Kalman Filter”. In: *Kalman filtering and neural networks* 5.2007 (2001), pp. 221–280.

[56] Geir Hovland. “Comparison of Different Kalman Filters for Tracking Nonlinear Transmission Torques”. In: *IFAC Proceedings Volumes* 37.14 (Sept. 2004), pp. 459–464.

[57] Arno Solin. “Cubature Integration Methods in Non-Linear Kalman Filtering and Smoothing”. Bachelor’s thesis. Aalto University, 2010.

[58] Jeff Bezanson et al. “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM review* 59.1 (2017), pp. 65–98.

[59] Patrick K. Mogensen and Asbjørn N. Riseth. “Optim: A Mathematical Optimization Package for Julia”. In: *Journal of Open Source Software* 3.24 (Apr. 5, 2018), p. 615.

[60] Ben Grossmann (https://math.stackexchange.com/users/81360/ben-grossmann). *How to Rewrite a Matrix Expression Involving Kronecker Product and Trace as a Quadratic Form?* eprint: `https://math.stackexchange.com/q/3782243`.

A. Reformulation of Non-Convex QCQP

We here show the reformulation of the non-convex QCQP given in equation (2.2). The derivation follows the one given in [41, Appendix B.1].

For ease of notation, we will not explicitly write the constraints $\alpha \in \mathbb{R}_+$ and $\boldsymbol{\rho}_i \in \mathbb{R}^m$. The problem at hand is then

$$\begin{aligned} \max_{\alpha, \boldsymbol{\rho}_i} \quad & \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{w}}_i + \boldsymbol{\rho}_i)^\mathsf{T}\mathbf{P}(\hat{\mathbf{w}}_i + \boldsymbol{\rho}_i) + \left((\mathbf{As}+\mathbf{Ba})^\mathsf{T}\mathbf{P}^\mathsf{T} + \frac{1}{2}\mathbf{p}^\mathsf{T}\right)(\hat{\mathbf{w}}_i + \boldsymbol{\rho}_i) + \alpha\lambda_{\max}(\mathbf{P}) \\ \text{s.t.} \quad & \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{\rho}_i\|_2^2 + \alpha \leq \varepsilon^2. \end{aligned}$$

Defining

$$\mathbf{q} = \mathbf{P}(\mathbf{As}+\mathbf{Ba}) + \frac{1}{2}\mathbf{p}$$

and rearranging terms in the objective gives

$$\begin{aligned} \max_{\alpha, \boldsymbol{\rho}_i} \quad & \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\rho}_i^\mathsf{T}\mathbf{P}\boldsymbol{\rho}_i + 2\left(\hat{\mathbf{w}}_i + \frac{1}{2}\mathbf{q}\right)^\mathsf{T}\boldsymbol{\rho}_i + \hat{\mathbf{w}}_i^\mathsf{T}\mathbf{P}\hat{\mathbf{w}}_i + \mathbf{q}^\mathsf{T}\hat{\mathbf{w}}_i + \alpha\lambda_{\max}(\mathbf{P}) \\ \text{s.t.} \quad & \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{\rho}_i\|_2^2 + \alpha \leq \varepsilon^2. \end{aligned}$$

The problematic terms are $\boldsymbol{\rho}_i^\mathsf{T}\mathbf{P}\boldsymbol{\rho}_i$, since $\mathbf{P}$ is positive definite. We can rewrite these as

$$\boldsymbol{\rho}_i^\mathsf{T}\mathbf{P}\boldsymbol{\rho}_i = \operatorname{tr}(\boldsymbol{\rho}_i^\mathsf{T}\mathbf{P}\boldsymbol{\rho}_i) = \operatorname{tr}(\mathbf{P}\boldsymbol{\rho}_i\boldsymbol{\rho}_i^\mathsf{T}) = \operatorname{tr}(\mathbf{P}\mathbf{X}_i),$$

where the second equality is due to the cyclic property of the trace, and we defined $\mathbf{X}_i = \boldsymbol{\rho}_i\boldsymbol{\rho}_i^\mathsf{T}$. Using this, the problem becomes

$$\begin{aligned} \max_{\alpha, \boldsymbol{\rho}_i, \mathbf{X}_i} \quad & \frac{1}{N}\sum_{i=1}^{N}\operatorname{tr}(\mathbf{P}\mathbf{X}_i) + 2\left(\hat{\mathbf{w}}_i + \frac{1}{2}\mathbf{q}\right)^\mathsf{T}\boldsymbol{\rho}_i + \hat{\mathbf{w}}_i^\mathsf{T}\mathbf{P}\hat{\mathbf{w}}_i + \mathbf{q}^\mathsf{T}\hat{\mathbf{w}}_i + \alpha\lambda_{\max}(\mathbf{P}) \\ \text{s.t.} \quad & \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{\rho}_i\|_2^2 + \alpha \leq \varepsilon^2 \\ & \mathbf{X}_i = \boldsymbol{\rho}_i\boldsymbol{\rho}_i^\mathsf{T} \quad \forall i \in [N]. \end{aligned}$$

Finally, we relax the equality $\mathbf{X}_i = \boldsymbol{\rho}_i\boldsymbol{\rho}_i^\mathsf{T}$ by replacing it with an inequality $\mathbf{X}_i \succeq \boldsymbol{\rho}_i\boldsymbol{\rho}_i^\mathsf{T}$, which can in turn be written as

$$\mathbf{X}_i \succeq \boldsymbol{\rho}_i\boldsymbol{\rho}_i^\mathsf{T} \quad \Leftrightarrow \quad \begin{bmatrix} \mathbf{X}_i & \boldsymbol{\rho}_i \\ \boldsymbol{\rho}_i^\mathsf{T} & 1 \end{bmatrix} \succeq 0.$$

The resulting problem

$$\begin{aligned}
\max_{\alpha,\boldsymbol{\rho}_i,\mathbf{X}_i} \quad & \frac{1}{N}\sum_{i=1}^{N} \operatorname{tr}(\mathbf{P}\mathbf{X}_i) + 2\left(\hat{\mathbf{w}}_i + \frac{1}{2}\mathbf{q}\right)^{\mathsf{T}} \boldsymbol{\rho}_i + \hat{\mathbf{w}}_i^{\mathsf{T}}\mathbf{P}\hat{\mathbf{w}}_i + \mathbf{q}^{\mathsf{T}}\hat{\mathbf{w}}_i + \alpha\lambda_{\max}(\mathbf{P}) \\
\text{s.t.} \quad & \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{\rho}_i\|_2^2 + \alpha \leq \varepsilon^2 \\
& \begin{bmatrix} \mathbf{X}_i & \boldsymbol{\rho}_i \\ \boldsymbol{\rho}_i^{\mathsf{T}} & 1 \end{bmatrix} \succeq 0 \quad \forall i \in [N]
\end{aligned}$$

is then a convex SDP and strong duality holds with the original problem.

B. Derivation of KL Robust Trajectory Optimization in Linear Quadratic Gaussian Case

B.1. Backward Pass

In this section, we provide some detailed steps for obtaining the closed form backward pass used in the parameter distribution update in section 3.3.1. Starting from the assumptions

$$\begin{aligned}
\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\Theta}) &= \mathcal{N}(\mathbf{s}'|\boldsymbol{\Theta}\mathbf{s}+\mathbf{b}_t\mathbf{a}+\mathbf{c}_t,\boldsymbol{\Sigma}_{s',t})\\
\pi_t(\mathbf{a}|\mathbf{s}) &= \mathcal{N}(\mathbf{a}|\mathbf{K}_t^{\pi}\mathbf{s}+\mathbf{k}_t^{\pi},\boldsymbol{\Sigma}_{a,t}^{\pi})\\
V_{t+1}(\mathbf{s}) &= \mathbf{s}^{\mathsf{T}}\mathbf{V}_{t+1}\mathbf{s}+\mathbf{s}^{\mathsf{T}}\mathbf{v}_{t+1}+v_{t+1}\\
\mu_t(\mathbf{s}) &= \mathcal{N}(\mathbf{s}|\boldsymbol{\tau}_{s,t}^{\mu},\boldsymbol{\Sigma}_{s,t}^{\mu})\\
\hat{p}_t(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_t,\hat{\boldsymbol{\Sigma}}_{\theta,t})\\
R_t(\mathbf{s},\mathbf{a}) &= (\mathbf{s}-\mathbf{z})^{\mathsf{T}}\mathbf{M}_t(\mathbf{s}-\mathbf{z})+\mathbf{a}^{\mathsf{T}}\mathbf{H}_t\mathbf{a}
\end{aligned}$$

we need to first evaluate

$$\begin{aligned}
Q_t(\mathbf{s},\boldsymbol{\theta}) &= \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}')\int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s})\mathcal{P}_t(\mathbf{s}'|\mathbf{s},\mathbf{a},\boldsymbol{\theta})\,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s}'\\
&= \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}')\int_{\mathbf{a}} \mathcal{N}(\mathbf{a}|\mathbf{K}_t^{\pi}\mathbf{s}+\mathbf{k}_t^{\pi},\boldsymbol{\Sigma}_{a,t}^{\pi})\mathcal{N}(\mathbf{s}'|\boldsymbol{\Theta}\mathbf{s}+\mathbf{b}_t\mathbf{a}+\mathbf{c}_t,\boldsymbol{\Sigma}_{s',t})\,\mathrm{d}\mathbf{a}\,\mathrm{d}\mathbf{s}'\\
&= \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}')\mathcal{N}(\mathbf{s}'|\underbrace{(\boldsymbol{\Theta}+\mathbf{b}_t\mathbf{K}_t^{\pi})}_{\boldsymbol{\Theta}^{\mathrm{cl}}}\mathbf{s}+\underbrace{\mathbf{b}_t\mathbf{k}_t^{\pi}+\mathbf{c}_t}_{\mathbf{c}_t^{\mathrm{cl}}},\underbrace{\boldsymbol{\Sigma}_{s',t}+\mathbf{b}_t\boldsymbol{\Sigma}_{a,t}^{\pi}\mathbf{b}_t^{\mathsf{T}}}_{\boldsymbol{\Sigma}_{s',t}^{\mathrm{cl}}})\,\mathrm{d}\mathbf{s}'\\
&= \int_{\mathbf{s}'} \left(\mathbf{s}'^{\mathsf{T}}\mathbf{V}_{t+1}\mathbf{s}'+\mathbf{s}'^{\mathsf{T}}\mathbf{v}_{t+1}+v_{t+1}\right)\mathcal{N}(\mathbf{s}'|\boldsymbol{\Theta}^{\mathrm{cl}}\mathbf{s}+\mathbf{c}_t^{\mathrm{cl}},\boldsymbol{\Sigma}_{s',t}^{\mathrm{cl}})\,\mathrm{d}\mathbf{s}'\\
&= \left(\boldsymbol{\Theta}^{\mathrm{cl}}\mathbf{s}+\mathbf{c}_t^{\mathrm{cl}}\right)^{\mathsf{T}}\mathbf{V}_{t+1}\left(\boldsymbol{\Theta}^{\mathrm{cl}}\mathbf{s}+\mathbf{c}_t^{\mathrm{cl}}\right)+\left(\boldsymbol{\Theta}^{\mathrm{cl}}\mathbf{s}+\mathbf{c}_t^{\mathrm{cl}}\right)^{\mathsf{T}}\mathbf{v}_{t+1}+v_{t+1}+\mathrm{tr}\left(\mathbf{V}_{t+1}\boldsymbol{\Sigma}_{s',t}^{\mathrm{cl}}\right),
\end{aligned}\tag{B.1}$$

where the variables with superscript "cl" are the closed loop dynamics parameters introduced for brevity.

Worst-Case Distribution

For the worst-case distribution, we then compute the integral

$$\int_{\mathbf{s}} \mu_t(\mathbf{s})Q_t(\mathbf{s},\boldsymbol{\theta})\,\mathrm{d}\mathbf{s} = \underbrace{\left(\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\tau}_{s,t}^{\mu}+\mathbf{c}_t^{\mathrm{cl}}\right)^{\mathsf{T}}\mathbf{V}_{t+1}\left(\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\tau}_{s,t}^{\mu}+\mathbf{c}_t^{\mathrm{cl}}\right)}_{Q_1(\boldsymbol{\theta}^{\mathrm{cl}})}+\underbrace{(\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\tau}_{s,t}^{\mu})^{\mathsf{T}}\mathbf{v}_{t+1}+\mathrm{tr}\left((\boldsymbol{\Theta}^{\mathrm{cl}})^{\mathsf{T}}\mathbf{V}_{t+1}\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\Sigma}_{s,t}^{\mu}\right)}_{Q_2(\boldsymbol{\theta}^{\mathrm{cl}})}+c$$

with a constant c which is independent of $\boldsymbol{\theta}$. Using

$$\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\tau}_{s,t}^{\mu} = \underbrace{\left((\boldsymbol{\tau}_{s,t}^{\mu})^{\mathsf{T}} \otimes \mathbf{I}_n\right)}_{\mathbf{T}} \boldsymbol{\theta}^{\mathrm{cl}},$$

the first quadratic form can be rewritten as

$$\begin{aligned} Q_1(\boldsymbol{\theta}^{\mathrm{cl}}) &= \left(\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\tau}_{s,t}^{\mu} + \mathbf{c}_t^{\mathrm{cl}}\right)^{\mathsf{T}} \mathbf{V}_{t+1} \left(\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\tau}_{s,t}^{\mu} + \mathbf{c}_t^{\mathrm{cl}}\right) \\ &= \left(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{T}^{+}\mathbf{c}_t^{\mathrm{cl}}\right)^{\mathsf{T}} \mathbf{T}^{\mathsf{T}} \mathbf{V}_{t+1} \mathbf{T} \left(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{T}^{+}\mathbf{c}_t^{\mathrm{cl}}\right) \\ &= \left(\boldsymbol{\theta}^{\mathrm{cl}} - \mathbf{a}_1\right)^{\mathsf{T}} \mathbf{A}_1 \left(\boldsymbol{\theta}^{\mathrm{cl}} - \mathbf{a}_1\right), \end{aligned}$$

where $\mathbf{T}^{+}$ is the Moore-Penrose inverse of $\mathbf{T}$. Using some basic properties of matrix trace, vectorization and Kronecker product, we also get

$$\begin{aligned} Q_2(\boldsymbol{\theta}^{\mathrm{cl}}) &= (\boldsymbol{\Theta}^{\mathrm{cl}}\boldsymbol{\tau}_{s,t}^{\mu})^{\mathsf{T}} \mathbf{v}_{t+1} + \mathrm{tr}\left((\boldsymbol{\Theta}^{\mathrm{cl}})^{\mathsf{T}} \mathbf{V}_{t+1} \boldsymbol{\Theta}^{\mathrm{cl}} \boldsymbol{\Sigma}_{s,t}^{\mu}\right) \\ &= \mathbf{v}_{t+1}^{\mathsf{T}} \mathbf{T} \boldsymbol{\theta}^{\mathrm{cl}} + \mathrm{tr}\left((\boldsymbol{\Theta}^{\mathrm{cl}})^{\mathsf{T}} \mathbf{V}_{t+1} \boldsymbol{\Theta}^{\mathrm{cl}} \boldsymbol{\Sigma}_{s,t}^{\mu}\right) \\ &= \mathbf{v}_{t+1}^{\mathsf{T}} \mathbf{T} \boldsymbol{\theta}^{\mathrm{cl}} + \mathrm{vec}\left(\mathbf{V}_{t+1} \boldsymbol{\Theta}^{\mathrm{cl}}\right)^{\mathsf{T}} \mathrm{vec}\left(\boldsymbol{\Theta}^{\mathrm{cl}} \boldsymbol{\Sigma}_{s,t}^{\mu}\right) \\ &= \mathbf{v}_{t+1}^{\mathsf{T}} \mathbf{T} \boldsymbol{\theta}^{\mathrm{cl}} + (\boldsymbol{\theta}^{\mathrm{cl}})^{\mathsf{T}} \left(\mathbf{I}_n \otimes \mathbf{V}_{t+1}\right) \left(\boldsymbol{\Sigma}_{s,t}^{\mu} \otimes \mathbf{I}_n\right) \boldsymbol{\theta}^{\mathrm{cl}} \\ &= \mathbf{v}_{t+1}^{\mathsf{T}} \mathbf{T} \boldsymbol{\theta}^{\mathrm{cl}} + (\boldsymbol{\theta}^{\mathrm{cl}})^{\mathsf{T}} \underbrace{\left(\boldsymbol{\Sigma}_{s,t}^{\mu} \otimes \mathbf{V}_{t+1}\right)}_{\mathbf{A}_2} \boldsymbol{\theta}^{\mathrm{cl}} \\ &= \left(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{A}_2^{-1}\left(\frac{1}{2}\mathbf{v}_{t+1}^{\mathsf{T}} \mathbf{T}\right)\right)^{\mathsf{T}} \mathbf{A}_2 \left(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{A}_2^{-1}\left(\frac{1}{2}\mathbf{v}_{t+1}^{\mathsf{T}} \mathbf{T}\right)\right) + c_2 \\ &= \left(\boldsymbol{\theta}^{\mathrm{cl}} - \mathbf{a}_2\right)^{\mathsf{T}} \mathbf{A}_2 \left(\boldsymbol{\theta}^{\mathrm{cl}} - \mathbf{a}_2\right) + c_2. \end{aligned}$$

The integral is then given by

$$\begin{aligned} \int_{\mathbf{s}} \mu_t(\mathbf{s}) Q_t(\mathbf{s}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{s} &= Q_1(\boldsymbol{\theta}^{\mathrm{cl}}) + Q_2(\boldsymbol{\theta}^{\mathrm{cl}}) + c \\ &= (\boldsymbol{\theta} - \mathbf{w}_t)^{\mathsf{T}} \mathbf{W}_t (\boldsymbol{\theta} - \mathbf{w}_t) + c_{w,t}, \end{aligned}$$

with

$$\begin{aligned} \mathbf{W}_t &= \mathbf{A}_1 + \mathbf{A}_2 \\ &= \left(\boldsymbol{\tau}_{s,t}^{\mu} (\boldsymbol{\tau}_{s,t}^{\mu})^{\mathsf{T}} + \boldsymbol{\Sigma}_{s,t}^{\mu}\right) \otimes \mathbf{V}_{t+1} \\ \mathbf{w}_t &= -\mathrm{vec}(\mathbf{b}_t \mathbf{K}_t^{\pi}) + \mathbf{W}_t^{-1}(\mathbf{A}_1 \mathbf{a}_1 + \mathbf{A}_2 \mathbf{a}_2) \\ &= -\mathrm{vec}(\mathbf{b}_t \mathbf{K}_t^{\pi}) - \mathbf{W}_t^{-1} (\boldsymbol{\tau}_{s,t}^{\mu} \otimes I_n)^{\mathsf{T}} \left(\mathbf{V}_{t+1}(\mathbf{b}_t \mathbf{k}_t^{\pi} + \mathbf{c}_t) + \frac{1}{2}\mathbf{v}_{t+1}\right) \end{aligned}$$

found by the addition rules for two quadratic forms and including the term $-\text{vec}(\mathbf{b}_t\mathbf{K}_t^\pi)$ to convert back from $\boldsymbol{\theta}^{\text{cl}}$ to $\boldsymbol{\theta}$. Using the above, the worst-case parameter distribution is

$$\begin{aligned}
p_t(\boldsymbol{\theta}) &\propto \hat{p}_t(\boldsymbol{\theta})\exp\left(-\frac{1}{\alpha_t}\int_{\mathbf{s}}\mu_t(\mathbf{s})Q_t(\mathbf{s},\boldsymbol{\theta})\,\mathrm{d}\mathbf{s}\right)\\
&\propto \hat{p}_t(\boldsymbol{\theta})\exp\left(-\frac{1}{\alpha_t}(\boldsymbol{\theta}-\mathbf{w}_t)^\mathsf{T}\mathbf{W}_t(\boldsymbol{\theta}-\mathbf{w}_t)\right)\\
&\propto \mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_t,\hat{\boldsymbol{\Sigma}}_{\theta,t})\exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\mathbf{w}_t)^\mathsf{T}\left(\frac{\alpha_t}{2}\mathbf{W}_t^{-1}\right)^{-1}(\boldsymbol{\theta}-\mathbf{w}_t)\right)\\
&\propto \exp\left(-\frac{1}{2}\left((\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}_t)^\mathsf{T}\hat{\boldsymbol{\Sigma}}_{\theta,t}^{-1}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}_t)+(\boldsymbol{\theta}-\mathbf{w}_t)^\mathsf{T}\left(\frac{\alpha_t}{2}\mathbf{W}_t^{-1}\right)^{-1}(\boldsymbol{\theta}-\mathbf{w}_t)\right)\right).
\end{aligned}$$

Finally,

$$p_t(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{\mu}_{\theta,t},\boldsymbol{\Sigma}_{\theta,t}\right),$$

where

$$\begin{aligned}
\boldsymbol{\Sigma}_{\theta,t} &= \left(\hat{\boldsymbol{\Sigma}}_{\theta,t}^{-1}+\frac{2}{\alpha_t}\mathbf{W}_t\right)^{-1}\\
\boldsymbol{\mu}_{\theta,t} &= \boldsymbol{\Sigma}_{\theta,t}\left(\hat{\boldsymbol{\Sigma}}_{\theta,t}^{-1}\hat{\boldsymbol{\theta}}_t+\frac{2}{\alpha_t}\mathbf{W}_t\mathbf{w}_t\right).
\end{aligned}$$

Value Function

For $t < T$, the value function is given by

$$V_t(\mathbf{s}) = \int_{\mathbf{a}} R_t(\mathbf{s},\mathbf{a})\pi_t(\mathbf{a}|\mathbf{s})\,\mathrm{d}\mathbf{a} + \int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta})Q_t(\mathbf{s},\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}.$$

The first integral evaluates to

$$\begin{aligned}
I_1(\mathbf{s}) &= \int_{\mathbf{a}} R_t(\mathbf{s},\mathbf{a})\pi_t(\mathbf{a}|\mathbf{s})\,\mathrm{d}\mathbf{a}\\
&= \int_{\mathbf{a}}\left((\mathbf{s}-\mathbf{z})^\mathsf{T}\mathbf{M}_t(\mathbf{s}-\mathbf{z})+\mathbf{a}^\mathsf{T}\mathbf{H}_t\mathbf{a}\right)\mathcal{N}(\mathbf{a}|\mathbf{K}_t^\pi\mathbf{s}+\mathbf{k}_t^\pi,\boldsymbol{\Sigma}_{a,t}^\pi)\,\mathrm{d}\mathbf{a}\\
&= (\mathbf{s}-\mathbf{z})^\mathsf{T}\mathbf{M}_t(\mathbf{s}-\mathbf{z})+(\mathbf{K}_t^\pi\mathbf{s}+\mathbf{k}_t^\pi)^\mathsf{T}\mathbf{H}_t(\mathbf{K}_t^\pi\mathbf{s}+\mathbf{k}_t^\pi)+\text{tr}(\mathbf{H}_t\boldsymbol{\Sigma}_{a,t}^\pi)\\
&= \mathbf{s}^\mathsf{T}\left(\mathbf{M}_t+(\mathbf{K}_t^\pi)^\mathsf{T}\mathbf{H}_t\mathbf{K}_t^\pi\right)\mathbf{s}+2\left((\mathbf{K}_t^\pi)^\mathsf{T}\mathbf{H}_t\mathbf{k}_t^\pi-\mathbf{M}_t\mathbf{z}\right)^\mathsf{T}\mathbf{s}+\mathbf{z}^\mathsf{T}\mathbf{M}_t\mathbf{z}+(\mathbf{k}_t^\pi)^\mathsf{T}\mathbf{M}_t\mathbf{k}_t^\pi+\text{tr}(\mathbf{H}_t\boldsymbol{\Sigma}_{a,t}^\pi).
\end{aligned}$$

For the second integral, we use equation (B.1) to write it as

$$\begin{aligned}
I_2(\mathbf{s}) &= \int_{\boldsymbol{\theta}} p_t(\boldsymbol{\theta})Q_t(\mathbf{s},\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}\\
&= \int_{\boldsymbol{\theta}^{\text{cl}}} p_t(\boldsymbol{\theta}^{\text{cl}})\left(\left(\boldsymbol{\Theta}^{\text{cl}}\mathbf{s}+\mathbf{c}_t^{\text{cl}}\right)^\mathsf{T}\mathbf{V}_{t+1}\left(\boldsymbol{\Theta}^{\text{cl}}\mathbf{s}+\mathbf{c}_t^{\text{cl}}\right)+\left(\boldsymbol{\Theta}^{\text{cl}}\mathbf{s}+\mathbf{c}_t^{\text{cl}}\right)^\mathsf{T}\mathbf{v}_{t+1}+\nu_{t+1}+\text{tr}\left(\mathbf{V}_{t+1}\boldsymbol{\Sigma}_{s',t}^{\text{cl}}\right)\right)\mathrm{d}\boldsymbol{\theta}^{\text{cl}}\\
&= \int_{\boldsymbol{\theta}^{\text{cl}}} p_t(\boldsymbol{\theta}^{\text{cl}})\left(\left(\boldsymbol{\Theta}^{\text{cl}}\mathbf{s}+\mathbf{c}_t^{\text{cl}}\right)^\mathsf{T}\mathbf{V}_{t+1}\left(\boldsymbol{\Theta}^{\text{cl}}\mathbf{s}+\mathbf{c}_t^{\text{cl}}\right)+\left(\boldsymbol{\Theta}^{\text{cl}}\mathbf{s}\right)^\mathsf{T}\mathbf{v}_{t+1}\right)\mathrm{d}\boldsymbol{\theta}^{\text{cl}}+\underbrace{\left(\mathbf{c}_t^{\text{cl}}\right)^\mathsf{T}\mathbf{v}_{t+1}+\nu_{t+1}+\text{tr}\left(\mathbf{V}_{t+1}\boldsymbol{\Sigma}_{s',t}^{\text{cl}}\right)}_{c'}
\end{aligned}$$

Similarly to what we did above, we introduce $\mathbf{S} = (\mathbf{s}^\mathsf{T} \otimes \mathbf{I}_n)$ and its Moore-Penrose inverse $\mathbf{S}^+$ to rewrite as

$$
\begin{aligned}
I_2(\mathbf{s}) &= \int_{\boldsymbol{\theta}^{\mathrm{cl}}} p_t(\boldsymbol{\theta}^{\mathrm{cl}}) \Big(\Big(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{S}^+ \mathbf{c}_t^{\mathrm{cl}} \Big)^\mathsf{T} \mathbf{S}^\mathsf{T} \mathbf{V}_{t+1} \mathbf{S} \Big(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{S}^+ \mathbf{c}_t^{\mathrm{cl}} \Big) + \mathbf{v}_{t+1}^\mathsf{T} \mathbf{S} \boldsymbol{\theta}^{\mathrm{cl}} \Big) \mathrm{d}\boldsymbol{\theta}^{\mathrm{cl}} + c' \\
&= \int_{\boldsymbol{\theta}^{\mathrm{cl}}} \mathcal{N}\Big(\boldsymbol{\theta}^{\mathrm{cl}} | \boldsymbol{\mu}_{\theta,t} + \mathrm{vec}(\mathbf{b}_t \mathbf{K}_t^\pi), \boldsymbol{\Sigma}_{\theta,t} \Big) \Big(\Big(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{S}^+ \mathbf{c}_t^{\mathrm{cl}} \Big)^\mathsf{T} \mathbf{S}^\mathsf{T} \mathbf{V}_{t+1} \mathbf{S} \Big(\boldsymbol{\theta}^{\mathrm{cl}} + \mathbf{S}^+ \mathbf{c}_t^{\mathrm{cl}} \Big) + \mathbf{v}_{t+1}^\mathsf{T} \mathbf{S} \boldsymbol{\theta}^{\mathrm{cl}} \Big) \mathrm{d}\boldsymbol{\theta}^{\mathrm{cl}} + c' \\
&= \Big(\Big(\boldsymbol{\mu}_{\theta,t} + \mathrm{vec}(\mathbf{b}_t \mathbf{K}_t^\pi) + \mathbf{S}^+ \mathbf{c}_t^{\mathrm{cl}} \Big)^\mathsf{T} \mathbf{S}^\mathsf{T} \mathbf{V}_{t+1} \mathbf{S} \Big(\boldsymbol{\mu}_{\theta,t} + \mathrm{vec}(\mathbf{b}_t \mathbf{K}_t^\pi) + \mathbf{S}^+ \mathbf{c}_t^{\mathrm{cl}} \Big) + \mathbf{v}_{t+1}^\mathsf{T} \mathbf{S} \Big(\boldsymbol{\mu}_{\theta,t} + \mathrm{vec}(\mathbf{b}_t \mathbf{K}_t^\pi) \Big) \Big) + \dots \\
&\quad + \mathrm{tr}\Big(\mathbf{S}^\mathsf{T} \mathbf{V}_{t+1} \mathbf{S} \boldsymbol{\Sigma}_{\theta,t} \Big) + c' \\
&= \Big(\bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}} \mathbf{s} + \mathbf{c}_t^{\mathrm{cl}} \Big)^\mathsf{T} \mathbf{V}_{t+1} \Big(\bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}} \mathbf{s} + \mathbf{c}_t^{\mathrm{cl}} \Big) + \mathbf{v}_{t+1}^\mathsf{T} \bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}} \mathbf{s} + \mathrm{tr}\Big(\mathbf{S}^\mathsf{T} \mathbf{V}_{t+1} \mathbf{S} \boldsymbol{\Sigma}_{\theta,t} \Big) + c' \\
&= \mathbf{s}^\mathsf{T} \Big((\bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}})^\mathsf{T} \mathbf{V}_{t+1} \bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}} + \mathbf{P} \Big) \mathbf{s} + \Big((\bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}})^\mathsf{T} \mathbf{v}_{t+1} + 2 \mathbf{c}_t^{\mathrm{cl}} \mathbf{V}_{t+1} \bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}} \Big)^\mathsf{T} \mathbf{s} + \mathrm{tr}\Big(\mathbf{S}^\mathsf{T} \mathbf{V}_{t+1} \mathbf{S} \boldsymbol{\Sigma}_{\theta,t} \Big) + c',
\end{aligned}
$$

where $\bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}} = \bar{\boldsymbol{\Theta}}_t + \mathbf{b}_t \mathbf{K}_t^\pi$ and $\mathrm{vec}(\bar{\boldsymbol{\Theta}}_t) = \boldsymbol{\mu}_{\theta,t}$. Following [60], the elements of matrix $\mathbf{P}_t$ in the above are given by $\mathbf{P}_t^{ij} = \mathrm{tr}(\mathbf{V}_{t+1} \boldsymbol{\Sigma}_{\theta,t}^{ij})$ where $\boldsymbol{\Sigma}_{\theta,t}^{ij}$ is obtained by partitioning the matrix $\boldsymbol{\Sigma}_{\theta,t}$ into n^2 block matrices of size $n \times n$ as

$$
\boldsymbol{\Sigma}_{\theta,t} = \begin{bmatrix} \boldsymbol{\Sigma}_{\theta,t}^{11} & \cdots & \boldsymbol{\Sigma}_{\theta,t}^{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\theta,t}^{n1} & \cdots & \boldsymbol{\Sigma}_{\theta,t}^{nn} \end{bmatrix}.
$$

Collecting all the terms above finally yields the closed form update for the value function as

$$
V_t(\mathbf{s}) = \mathbf{s}^\mathsf{T} \mathbf{V}_t \mathbf{s} + \mathbf{s}^\mathsf{T} \mathbf{v}_t + \nu_t,
$$

where

$$
\begin{aligned}
\mathbf{V}_t &= \begin{cases} \mathbf{M}_T & t = T \\ \mathbf{M}_t + (\mathbf{K}_t^\pi)^\mathsf{T} \mathbf{H}_t \mathbf{K}_t^\pi + (\bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}})^\mathsf{T} \mathbf{V}_{t+1} \bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}} + \mathbf{P}_t & t < T \end{cases} \\
\mathbf{v}_t &= \begin{cases} -2\mathbf{M}_T \mathbf{z} & t = T \\ (\bar{\boldsymbol{\Theta}}_t^{\mathrm{cl}})^\mathsf{T} (2\mathbf{V}_{t+1} \mathbf{c}_t^{\mathrm{cl}} + \mathbf{v}_{t+1}) - 2\mathbf{M}_t \mathbf{z} + 2(\mathbf{K}_t^\pi)^\mathsf{T} \mathbf{H}_t \mathbf{k}_t^\pi & t < T \end{cases} \\
\nu_t &= \begin{cases} \mathbf{z}^\mathsf{T} \mathbf{M}_T \mathbf{z} & t = T \\ (\mathbf{c}_t^{\mathrm{cl}})^\mathsf{T} (\mathbf{V}_{t+1} \mathbf{c}_t^{\mathrm{cl}} + \mathbf{v}_{t+1}) + \nu_{t+1} + \mathrm{tr}(\mathbf{H}_t \boldsymbol{\Sigma}_{a,t}^\pi) + \mathrm{tr}\Big(\mathbf{V}_{t+1} (\boldsymbol{\Sigma}_{s',t} + \mathbf{b}_t \boldsymbol{\Sigma}_{a,t}^\pi \mathbf{b}_t^\mathsf{T}) \Big) + \mathbf{z}^\mathsf{T} \mathbf{M}_t \mathbf{z} + (\mathbf{k}_t^\pi)^\mathsf{T} \mathbf{H}_t \mathbf{k}_t^\pi & t < T. \end{cases}
\end{aligned}
$$

B.2. Dual Objective

For evaluation of the objective and its derivative in closed form, we make use of the well known equation for KL divergence between two Gaussians

$$
\begin{aligned}
L_p(\mu_t, V_t, \alpha_p, p_t, \hat{p}_t) &= \int_{\mathbf{s}} V_1(\mathbf{s})\mu_1(\mathbf{s})\,\mathrm{d}\mathbf{s} + \alpha_p \sum_{t=1}^{T-1} \mathrm{KL}(p_t(\boldsymbol{\theta}) \| \hat{p}_t(\boldsymbol{\theta})) - \alpha_p \varepsilon_p \\
&= (\boldsymbol{\tau}_{s,1}^{\mu})^{\mathsf{T}} \mathbf{V}_1 \boldsymbol{\tau}_{s,1}^{\mu} + (\boldsymbol{\tau}_{s,1}^{\mu})^{\mathsf{T}} \mathbf{v}_1 + \nu_1 - \alpha_p \varepsilon_p + \dots \\
&\quad + \frac{\alpha_p}{2} \sum_{t=1}^{T-1} \left(\mathrm{tr}\left(\hat{\boldsymbol{\Sigma}}_t^{-1} \boldsymbol{\Sigma}_{\theta,t} \right) + (\hat{\boldsymbol{\theta}}_t - \boldsymbol{\mu}_{\theta,t})^{\mathsf{T}} \hat{\boldsymbol{\Sigma}}_t^{-1} (\hat{\boldsymbol{\theta}}_t - \boldsymbol{\mu}_{\theta,t}) - n + \log\left(\frac{|\hat{\boldsymbol{\Sigma}}_t|}{|\boldsymbol{\Sigma}_{\theta,t}|} \right) \right) \\
\frac{\partial L_p}{\partial \alpha_p} &= \sum_{t=1}^{T-1} \mathrm{KL}(p_t(\boldsymbol{\theta}) \| \hat{p}_t(\boldsymbol{\theta})) - \varepsilon_p \\
&= \frac{1}{2} \sum_{t=1}^{T-1} \left(\mathrm{tr}\left(\hat{\boldsymbol{\Sigma}}_t^{-1} \boldsymbol{\Sigma}_{\theta,t} \right) + (\hat{\boldsymbol{\theta}}_t - \boldsymbol{\mu}_{\theta,t})^{\mathsf{T}} \hat{\boldsymbol{\Sigma}}_t^{-1} (\hat{\boldsymbol{\theta}}_t - \boldsymbol{\mu}_{\theta,t}) - n + \log\left(\frac{|\hat{\boldsymbol{\Sigma}}_t|}{|\boldsymbol{\Sigma}_{\theta,t}|} \right) \right) - \varepsilon_p .
\end{aligned}
$$

C. Summary of Closed Form Equations for GPS Trajectory Optimization Step

We here give the policy optimization equations for the trajectory optimization of Guided Policy Search as derived in [49] adapted for the case of dynamics which include a distribution over the parameter as given in equation (3.12). Furthermore, we use the case of a single expected KL constraint on the policy.

C.1. Backward Pass

Augmented Reward

$$\begin{aligned} r_t(\mathbf{s},\mathbf{a}) &= R_t(\mathbf{s},\mathbf{a}) + \alpha_\pi \log(q_t(\mathbf{a}|\mathbf{s})) \\ &= \mathbf{s}^\mathsf{T}\mathbf{R}_{ss,t}\mathbf{s} + \mathbf{a}^\mathsf{T}\mathbf{R}_{aa,t}\mathbf{a} + \mathbf{a}^\mathsf{T}\mathbf{R}_{sa,t}^\mathsf{T}\mathbf{s} + \mathbf{s}^\mathsf{T}\mathbf{R}_{sa,t}\mathbf{a} + \mathbf{s}^\mathsf{T}\mathbf{r}_{s,t} + \mathbf{a}^\mathsf{T}\mathbf{r}_{a,t} + r_{0,t}, \end{aligned}$$

where

$$\begin{aligned} \mathbf{R}_{ss,t} &= \mathbf{M}_t - \frac{\alpha_\pi}{2}(\mathbf{K}_t^q)^\mathsf{T}(\mathbf{\Sigma}_{a,t}^q)^{-1}\mathbf{K}_t^q \\ \mathbf{R}_{aa,t} &= \mathbf{H}_t - \frac{\alpha_\pi}{2}\mathbf{\Sigma}_{a,t}^q)^{-1} \\ \mathbf{R}_{sa,t} &= \frac{\alpha_\pi}{2}(\mathbf{K}_t^q)^\mathsf{T}(\mathbf{\Sigma}_{a,t}^q)^{-1} \\ \mathbf{r}_{s,t} &= -\alpha_\pi(\mathbf{K}_t^q)^\mathsf{T}(\mathbf{\Sigma}_{a,t}^q)^{-1}\mathbf{k}_t^q - 2\mathbf{M}_t\mathbf{z} \\ \mathbf{r}_{a,t} &= \alpha_\pi(\mathbf{\Sigma}_{a,t}^q)^{-1}\mathbf{k}_t^q \\ r_{0,t} &= \mathbf{z}^\mathsf{T}\mathbf{M}_t\mathbf{z} - \alpha_\pi \log\sqrt{\left|2\pi\mathbf{\Sigma}_{a,t}^q\right|} - \frac{\alpha_\pi}{2}(\mathbf{k}_t^q)^\mathsf{T}(\mathbf{\Sigma}_{a,t}^q)^{-1}\mathbf{k}_t^q \end{aligned}$$

State-Action Value Function

$$\begin{aligned} Q_t(\mathbf{s},\mathbf{a}) &= \frac{1}{\alpha_\pi}\Big(r_t(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\mathcal{P}}\big[V_{t+1}(\mathbf{s}')\big]\Big) \\ &= \mathbf{s}^\mathsf{T}\mathbf{Q}_{ss,t}\mathbf{s} + \mathbf{a}^\mathsf{T}\mathbf{Q}_{aa,t}\mathbf{a} + \mathbf{a}^\mathsf{T}\mathbf{Q}_{sa,t}^\mathsf{T}\mathbf{s} + \mathbf{s}^\mathsf{T}\mathbf{Q}_{sa,t}\mathbf{a} + \mathbf{s}^\mathsf{T}\mathbf{Q}_{s,t} + \mathbf{a}^\mathsf{T}\mathbf{Q}_{a,t} + Q_{0,t}, \end{aligned}$$

where

$$\begin{aligned}
\mathbf{Q}_{ss,t} &= \frac{1}{\alpha_\pi}\Big(\mathbf{R}_{ss,t} + \bar{\mathbf{\Theta}}_t^\mathsf{T}\mathbf{V}_{t+1}\bar{\mathbf{\Theta}}_t + \mathbf{P}_t\Big) \\
\mathbf{Q}_{aa,t} &= \frac{1}{\alpha_\pi}\Big(\mathbf{R}_{aa,t} + \mathbf{b}_t^\mathsf{T}\mathbf{V}_{t+1}\mathbf{b}_t\Big) \\
\mathbf{Q}_{sa,t} &= \frac{1}{\alpha_\pi}\Big(\mathbf{R}_{sa,t} + \bar{\mathbf{\Theta}}_t^\mathsf{T}\mathbf{V}_{t+1}\mathbf{b}_t\Big) \\
\mathbf{Q}_{s,t} &= \frac{1}{\alpha_\pi}\Big(\mathbf{r}_{s,t} + 2\bar{\mathbf{\Theta}}_t^\mathsf{T}\mathbf{V}_{t+1}\mathbf{c}_t + \bar{\mathbf{\Theta}}_t^\mathsf{T}\mathbf{v}_{t+1}\Big) \\
\mathbf{Q}_{a,t} &= \frac{1}{\alpha_\pi}\Big(\mathbf{r}_{a,t} + 2\mathbf{b}_t^\mathsf{T}\mathbf{V}_{t+1}\mathbf{c}_t + \mathbf{b}_t^\mathsf{T}\mathbf{v}_{t+1}\Big) \\
Q_{0,t} &= \frac{1}{\alpha_\pi}\Big(\mathbf{c}_t^\mathsf{T}\mathbf{V}_{t+1}\mathbf{c}_t + \mathrm{tr}(\mathbf{V}_{t+1}\mathbf{\Sigma}_{s',t}) + \mathbf{v}_{t+1}^\mathsf{T}\mathbf{c}_t + v_{t+1} + r_{0,t}\Big).
\end{aligned}$$

Here, when introducing a distribution over the parameters, the additional matrix $\mathbf{P}_t$ has to be accounted for. Its elements are given by $\mathbf{P}_t^{ij} = \mathrm{tr}(\mathbf{V}_{t+1}\mathbf{\Sigma}_{\theta,t}^{ij})$, where $\mathbf{\Sigma}_{\theta,t}^{ij}$ indicates the (i,j)-th $n \times n$ block of the matrix $\mathbf{\Sigma}_{\theta,t}$. Furthermore, the dynamics matrix is given by the mean of the parameter distribution $\bar{\mathbf{\Theta}}_t$.

Policy

$$\begin{aligned}
\pi(\mathbf{a}|\mathbf{s}) &= \frac{\exp\big(Q_t(\mathbf{s},\mathbf{a})\big)}{\int_\mathbf{a}\exp\big(Q_t(\mathbf{s},\mathbf{a})\big)\,\mathrm{d}\mathbf{a}} \\
&= \mathcal{N}(\mathbf{a}|\mathbf{k}_t^\pi + \mathbf{K}_t^\pi\mathbf{s}, \mathbf{\Sigma}_{a,t}^\pi),
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{k}_t^\pi &= -\frac{1}{2}\mathbf{Q}_{aa,t}^{-1}\mathbf{Q}_{a,t} \\
\mathbf{K}_t^\pi &= -\mathbf{Q}_{aa,t}^{-1}\mathbf{Q}_{sa,t}^\mathsf{T} \\
\mathbf{\Sigma}_{a,t}^\pi &= -\frac{1}{2}\mathbf{Q}_{aa,t}^{-1}.
\end{aligned}$$

State Value Function

$$V_t(s) = \mathbf{s}^\mathsf{T}\mathbf{V}_t\mathbf{s} + \mathbf{s}^\mathsf{T}\mathbf{v}_t + v_t,$$

where

$$\begin{aligned}
\mathbf{V}_t &= \begin{cases} \mathbf{M}_T & t = T \\ \alpha_\pi(\mathbf{Q}_{ss,t} + \mathbf{Q}_{sa,t}\mathbf{K}_t^\pi) & t < T \end{cases} \\
\mathbf{v}_t &= \begin{cases} -2\mathbf{M}_T\mathbf{z} & t = T \\ \alpha_\pi(\mathbf{Q}_{s,t} + 2\mathbf{Q}_{sa,t}\mathbf{k}_t^\pi) & t < T \end{cases} \\
v_t &= \begin{cases} \mathbf{z}^\mathsf{T}\mathbf{M}_T\mathbf{z} & t = T \\ \alpha_\pi\Big(\frac{1}{2}\mathbf{Q}_{a,t}^\mathsf{T}\mathbf{k}_t^\pi + Q_{0,t} + \frac{1}{2}\Big(m\log(2\pi) - \log\big(\big|-2\mathbf{Q}_{aa,t}\big|\big)\Big)\Big) & t < T \end{cases}
\end{aligned}$$

C.2. Dual Objective

$$
\begin{aligned}
L_\pi(\mu_t, V_t, \alpha_\pi) &= \int_{\mathbf{s}} V_1(\mathbf{s})\mu_1(\mathbf{s})\,\mathrm{d}\mathbf{s} + \varepsilon_\pi \alpha_\pi \\
&= (\boldsymbol{\tau}_{s,1}^{\mu})^{\mathsf{T}} \mathbf{V}_1 \boldsymbol{\tau}_{s,1}^{\mu} + (\boldsymbol{\tau}_{s,1}^{\mu})^{\mathsf{T}} \mathbf{v}_1 + v_1 + \varepsilon_\pi \alpha_\pi \\
\frac{\partial L_\pi}{\partial \alpha_\pi} &= \varepsilon_\pi - \sum_{t=1}^{T-1} \mathbb{E}_{\mu_t}\Big[\mathrm{KL}\big(\pi_t(\mathbf{a}|\mathbf{s}) \| q_t(\mathbf{a}|\mathbf{s})\big)\Big]
\end{aligned}
$$