Chance Constraints for Stochastic Optimal Control and Stochastic Optimization

Probabilistische Nebenbedingungen in stochastich optimaler Regelung und stochastischer Optimierung

Master-Thesis von Onur Celik

Dezember 2018

TECHNISCHE UNIVERSITÄT DARMSTADT

IAS

Chance Constraints for Stochastic Optimal Control and Stochastic Optimization
Probabilistische Nebenbedingungen in stochastich optimaler Regelung und stochastischer Optimierung

Vorgelegte Master-Thesis von Onur Celik

1. Gutachten: M. Sc. Hany Abdulsamad
2. Gutachten: Prof. Jan Peters
3. Gutachten: Prof. Ulrich Konigorski

Tag der Einreichung:

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

Hiermit versichere ich, Onur Celik, die vorliegende Master-Thesis 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 überein.

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

Datum / Date: ____________________ Unterschrift / Signature: ____________________

Abstract

Constraints in optimization problems for static systems is well studied nowadays. Effective numerical methods for convex and non-convex problems exist. However, for optimal control problems state and action constraints can be challenging, especially in case it is dealt with unknown dynamics in the Markov Decision Process. Dynamic Programming approaches, as Differential Dynamic Programming or iterative Linear Quadratic Gaussian, are powerful methods for Trajectory Optimization. In unknown dynamics tasks these methods generally gain linearized dynamics around the trajectory distribution and optimize subject to the linearized dynamics, which lead to local Linear Quadratic Gaussian controllers in closed-form. If constraints are not considered, the algorithms exploit the linearized dynamics and violate the constraints, such that validity of linearizations are not ensured anymore. However, action and state constraints in robotics are common and often known from construction. Thus, incorporating these constraints into the optimization procedure should yield advantages in terms of convergence and better overall performance. As the transition dynamics in Markov Decision Processes are stochastic, considering hard constraints is not possible.

In this work, we introduce Chance Constraints within the iterative Linear Quadratic Gaussian framework, in order to increase convergence performance. For this purpose, we reformulate the Chance Constraints as a deterministic constraint and use a direct shooting method to formulate an optimization problem after the backward pass of iterative Linear Quadratic Gaussian, which is solved by using numerical solver CasADi.

Known model-free policy search algorithms, like Relative Entropy Policy Search use information-theoretic bounds to prevent loss of information. These methods bound the policy update by using the Kullback-Leibler Divergence. For Relative Entropy Policy Search this results in an entropic risk measure in the corresponding dual. By updating the policy in direction of high reward regions, the algorithm is explicitly risk-seeking in terms of expected reward difference. This risk behavior can be tuned by the upper bound ϵ for Kullback-Leibler Divergence.

In the second part of this work we propose a new point of view for Regularized Stochastic Optimization inspired by the risk behavior of Relative Entropy Policy search. We bound the reward change directly by using Chance Constraints instead of bounding the policy change. We show the proposed method's behavior on finding the minimum value of two one-dimensional functions.

Zusammenfassung

Nebenbedingungen in Optimierungsproblemen für statische Systeme sind heute gut untersucht. Es gibt effektive numerische Methoden für konvexe und nicht konvexe Probleme. Für optimale Regelung können jedoch Zustands- und Stellgrößenbeschränkungen eine Herausforderung darstellen, insbesondere wenn es sich um eine unbekannte Dynamik im Markov-Entscheidungsprozess handelt. Dynamische Programmieransätze wie Differential Dynamic Programming oder iterative Linear Quadratic Gaussian sind leistungsfähige Methoden zur Trajektorienoptimierung. In unbekannten dynamischen Problemen linearisieren diese Verfahren die Dynamik in der Regel um die Trajektorienverteilung und optimieren in Abhängigkeit von der linearisierten Dynamik, was zu lokalen linearen quadratischen Reglern in geschlossener Form führt. Wenn Nebenbedingungen nicht berücksichtigt werden, nutzen die Algorithmen zu sehr die linearisierte Dynamik und verletzen die Einschränkungen, so dass die Gültigkeit von Linearisierungen nicht mehr gewährleistet ist. In der Robotik sind Stellgrößen- und Zustandsbeschränkungen allerdings üblich und oft aus der Konstruktion bekannt. Die Einbeziehung dieser Einschränkungen in das Optimierungsverfahren sollte daher Vorteile in Bezug auf Konvergenz und bessere Gesamtleistung bringen. Da die Übergangsdynamik in Markov Entscheidungsprozessen stochastisch ist, ist es nicht möglich, harte Einschränkungen zu berücksichtigen. In dieser Arbeit führen wir probabilistische Nebenbedingungen innerhalb des iterative Linear Quadratic Gaussian Algorithmus' ein, um die Konvergenzleistung zu erhöhen. Zu diesem Zweck formulieren wir die probabilistischen Nebenbedingungen als deterministische Nebenbedingung und formulieren mit einer direkten Methode ein Optimierungsproblem nach dem backward pass des iterative Linear Quadratic Gaussian Algorithmus', das mit dem numerischen Solver CasADi gelöst wird.

Bekannte modellfreie Algorithmen zur policy Suche, wie der Relative Entropy Policy Search Algorithmus, verwenden informationstheoretische Nebenbedingungen, um Informationsverluste zu vermeiden. Diese Methoden stellen eine Oberschranke für die Aktualisierung der policy mit Hilfe der Kullback-Leibler Divergenz. Für den Relative Entropy Policy Search Algorithmus ergibt sich daraus ein entropisches Risikomaß in der entsprechenden Dualform. Durch die Aktualisierung der policy in Richtung der Regionen mit hoher Belohnung sucht der Algorithmus explizit nach Risiken in Bezug auf die erwartete Belohnungsdifferenz. Dieses Risikoverhalten kann durch die obere Grenze von ϵ für die Kullback-Leibler Divergenz eingestellt werden. Im zweiten Teil dieser Arbeit schlagen wir einen neuen Standpunkt für die regularisierte stochastische Optimierung vor, was durch das Risikoverhalten des Relative Entropy Policy Search Algorithmus' inspiriert ist. Wir beschränken die Belohnungsänderung direkt mithilfe von probabilistischen Nebenbedingungen, anstatt die policy Änderung zu beschränken. Wir untersuchen das Verhalten der vorgeschlagenen Methode anhand eines eindimensionalen Minimierungsproblems für eine konvexe und nicht konvexe Funktion.

Acknowledgments

First, I would like to express my sincere appreciation to Hany Abdulsamad for providing an interesting thesis topic, always taking the time for interesting, informative and motivating discussions and his open-door policy. He always gave me the needed information, whenever it was necessary to keep the right direction. I have learned many new things from him.

I further want to thank the members of the Intelligent Autonomous Systems (IAS) group for providing the opportunity to write this thesis.
I want to thank Prof. Ulrich Konigorski from the Institut für Regelungstechnik und Mechatronik (RTM) at the Electrical Engineering Department, who agreed to co-supervise me and showed interest in my work.

I also want to thank the DLab Crew for the amusing working atmosphere and helpful advices and discussions.

Finally, I want to thank my family, who always supported and motivated me during my whole studies. Without them this accomplishment would not have been possible.

Contents

Figures and Tables

List of Figures

List of Tables

Abbreviations

List of Abbreviations

Notation	Description
cdf	cummulative density function
DDP	Differential Dynamic Programming
DP	Dynamic Programming
EKF	Extended Kalman Filter
iLQG	iterative Linear Quadratic Gaussian
KKT	Karush-Kuhn-Tucker
KL	Kullback-Leibler Divergence
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
MDP	Markov Decision Process
MPC	Model Predictive Control
OC	Optimal Control
pdf	probability density function
QP	Quadratic Program
REPS	Relative Entropy Policy Search
SMPC	Stochastic Model Predictive Control
SQP	Sequentiell Quadratic Programming
SSOP	Static Stochastic Optimization Problem

TO	Trajectory Optimization
UKF	Unscented Kalman Filter
UT	Unscented Transformation

1 Introduction

1.1 Local Validity of Linearized Dynamics with State and Action Constraints

Dealing with non-linear unknown dynamics is commonly done by linearizing around trajectories in the Dynamic Programming (DP) framework. As we will see throughout this work, Differential Dynamic Programming (DDP) and iterative Linear Quadratic Gaussian (iLQG) perform these linearizations in order to solve the optimal control problem effectively [1].
However, care has to be taken to stay in the validity regions of these linearizations while finding the optimal solution. For unconstrained problems, this issue is addressed by regularizations to keep the resulting trajectories near to the old. The principle of these iterative algorithms are described in section 3.1.1.
In many physical systems as in robotics, constraints on the actions and the states exist and often are known from construction. Neglecting these constraints during the optimization procedure can lead to convergence issues, as the algorithms plan the system to drive in regions it is not possible to reach due to constraints.
The results are invalid linear representations of the system and therefore totally inappropriate policies. In the worst case, this can lead to unstable behavior. However, neglecting constraints during the optimization has big impacts on the convergence of the algorithms. Satisfying the constraints is a hard task for non-linear systems. But for linear dynamics, this issue is easy to consider during the optimization procedure for finite horizon problems, when direct methods are used. Optimizing with respect to constraints for linear dynamics is a well-studied issue and thus effective methods exist. In case of considering the constraints during optimization, the validity of the linearized dynamics is kept in addition to the standard regularizations and possibly more robust controller updates can be achieved.

1.2 Considering Constraints in Real System vs. Optimizing Under Linear Dynamics Subject to Constraints

Throughout this work we will consider unknown dynamics in the Markov Decision Process (MDP). We will use iLQG, to obtain linearized dynamics by sampling from the system. Thus, when we talk about satisfying constraints, we will generally mean satisfying constraints for the linearized dynamics. Of course, it is desired that the non-linear system also satisfies the constraints. However, as we do not have an exact model, this can not be guaranteed. In contrast to that, we can guarantee to regard constraints with a predefined certainty due to MDP. Thus, if the linearized models are representative for the real model, we will be more likely to satisfy given constraints for the real systems. Nevertheless, although we are not able to give guarantees on the constraints for the non-linear dynamic system, considering constraints during the optimization procedure should yield better results and better convergence behavior. This can be exploited by further algorithms like Guided Policy Search [2] for example, which uses Stochastic Optimal Control algorithms to learn global policies.

1.3 Stochasticity and Constraints

In MDP we deal with stochastic transition dynamics, which yield to uncertainties in the dynamics. Consequently, if we want to consider constraints within an uncertain environment, we can not assume deterministic constraints anymore. As a result, we have to consider stochastic constraints called Probabilistic Constraints or Chance Constraints. Solving this kind of constraints turn out to be a difficult issue. However, due to linear dynamics which we will consider in our approach, we are able to relax these constraints and reformulate them as a linear deterministic constraint.

1.4 Preliminaries

In this section the main methods and assumptions, which are important for this work are presented.

1.4.1 Markov Decision Process

An MDP is defined by the stochastic transition dynamics $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, which only depends on the current state $\mathbf{s}_t$ and action $\mathbf{a}_t$. The dependency of the stochastic transition on only the current state and action is referred to as Markov

Property. Furthermore the state space $\mathcal{S}$ contains all possible state realizations $\mathbf{s}$ and the action space $\mathcal{A}$ contains all possible action realizations $\mathbf{a}$. A reward for state action pairs is defined as $r_t(\mathbf{s}, \mathbf{a})$. Furthermore the states are initialized with an initial state probability $\mu_0(\mathbf{s})$ [3]. Throughout this work we are going to assume MDPs.

1.4.2 Information Theoretic Measures

In the second part of this work, we are going to deal with information theoretic measures. For this purpose, we briefly want to introduce the Differential Entropy and the Relative Entropy of distributions.

Differential Entropy

In general, the entropy of a distribution $p(\mathbf{s})$ with random variable $\mathbf{s}$ is defined as

$$H = -\int_{\mathbf{s}} p(\mathbf{s}) \log p(\mathbf{s}) d\mathbf{s}.$$

The negative sign is in order to obtain positive entropy values. For a normal distribution $p(\mathbf{s})$ the entropy gives information about the variance of the distribution and therefore measures the amount of information within the distribution [4]. In chapter 4 we will use the entropy of the current policy to be maximized.

1.4.3 Kullback-Leibler Divergence

The Kullback-Leibler Divergence (KL) measures the information loss between two probability density functions $p(\mathbf{s})$ and $q(\mathbf{s})$ [4] with

$$D_{KL}(p(\mathbf{s}||q(\mathbf{s}))) = \int_{\mathbf{s}} p(\mathbf{s}) \log\left(\frac{p(\mathbf{s})}{q(\mathbf{s})}\right).$$

In Information Theoretic Policy Search algorithms like Relative Entropy Policy Search (REPS) KL, plays a central role. In chapter 4 we will use REPS as a motivation to formulate a new point of view for Stochastic Optimization.

2 Foundations

Before starting to describe our methods, we want to introduce fundamentals.
Optimization is the scientific field of finding the best possible solution for a mathematical problem. The solution is characterized by a cost function and possible equality or inequality constraints [5].
On a high level, optimization can be divided into optimizing static and dynamic systems, where each of them can be divided into convex and non-convex optimization problems [5].
In this chapter, we will briefly introduce the fundamentals of Optimization, then go further with a small insight in Model Predictive Control and introduce Chance Constraints.

2.1 Static Optimization

In static optimization, the optimized system is static and thus, does not have any dynamics. How we will see later in this work, it is possible to reformulate a dynamic optimization problem into a static problem and solve it with known techniques of static optimization.

2.1.1 Problem Formulation

In [6], a general mathematical optimization problem and its properties are given in the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \varphi(\mathbf{x}), \\ \text{s.t.} \quad & f_i(\mathbf{x}) \leq b_i, \quad i = 1, \dots, m, \\ & g_j(\mathbf{x}) = c_j, \quad j = 1, \dots, n, \end{aligned}$$

where $\mathbf{x}$ forms the optimization or the decision variable, function $\varphi : \mathbb{R}^n \to \mathbb{R}$ describes the objective function, functions $f_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \dots, m$ are constraint functions with bounds $b_1, \dots, b_m$ and $g_j : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \dots, m$ are equality constraints. Since a minimization problem is considered here, the optimal solution $\mathbf{x}^*$ minimizes the objective, while considering the constraints. Even if a solution $\mathbf{z}$, which fulfills the constraints is given, it will always hold that $\varphi(\mathbf{z}) \geq \varphi(\mathbf{x}^*)$ [6].
Finding $\mathbf{x}^*$ turns out to be difficult for arbitrary functions $f_i(\mathbf{x}), g_i(\mathbf{x}), \varphi(\mathbf{x})$, except for convex objectives, convex inequality functions and linear equality functions. If so, the optimization problem turns to a convex optimization problem. These problems have a global solution, which can be found very efficiently, even if a closed form solution does not exist [6].
A function f_i is **convex**, if it fulfills the inequality

$$f_i(\alpha\mathbf{x} + \beta\mathbf{y}) \leq \alpha f_i(\mathbf{x}) + \beta f_i(\mathbf{y}), \quad \forall \mathbf{x,y} \in \mathbb{R}^n, \quad \alpha, \beta, \in \mathbb{R}_0^+, \quad \alpha + \beta = 1.$$

A convex optimization problem is given when the objective function, all inequality constraint functions are convex and the equality constraints are linear.

2.1.2 Static Stochastic Optimization

In many cases, variables of a static optimization problem are not deterministic. If it is not known from the beginning that there exists stochasticity in the optimization problem, first the deterministic case is solved to detect stochasticity in the optimization procedure as the outcome changes in each repetition. Then the random variables are figured out and the optimization procedure is formulated respecting the probability density function of the random variables. If we do not wait for the occurrence of the next realization of randomness after solving the Optimization Problem, a Static Stochastic Optimization Problem (SSOP) is considered [7].
For the case that random variables are present in the objective, a common way how to deal with this stochasticity is to minimize the expected value of the objective [7]. For some special cases, the expectation can be calculated in closed form and if stochastic variables occur only in the objective, the optimization problem turns to a deterministic problem. If a closed form solution is not tractable, calculating the well-known estimator for the mean is a general choice. Considering the expectation is a risk-neutral approach [8].

Several models of Stochastic Optimization problems exist. The most important cases are covered here. For detailed review, we especially want to refer to [7] and [9].
For the case of **Probability Maximization** [7], the formulation

$$\begin{aligned} \max_{\mathbf{x}} \quad & Pr(T\mathbf{x} \geq \xi), \\ \text{s.t.} \quad & \mathbf{x} \in D, \end{aligned}$$

is considered. The probability, of **x** being in a certain space within the feasible set D of the decision variable **x**. Note that in this problem, the random variable is the bound ξ.
Random variables can occur not only in the objective but also in the constraints. A way to handle these constraints is by considering the mean of it. But the expected value does not implicit guarantees for satisfying the constraints. In general, for stochastic constraints, there are no guarantees to fulfill these constraints. If one wants to ensure that the constraints are satisfied for a certain level of safety, we can assign a probability level to them and therefore guarantee to be fulfilled with a certain probability.
This type of model or problem is referred to as **Programming under Probabilistic Constraints** or **Programming under Chance Constraints**. For this case [7], the optimization problem is established as

$$\begin{aligned} \min_{\mathbf{x}} \quad & h(\mathbf{x}), \\ \text{s.t.} \quad & Pr(g_1(\mathbf{x}, \xi) \geq 0, g_2(\mathbf{x}, \xi) \geq 0, ..., g_\tau(\mathbf{x}, \xi) \geq 0) \geq p, \\ & \mathbf{x} \in D. \end{aligned}$$

Note that the probabilistic constraint can be rewritten in a short manner as

$$Pr(T\mathbf{x} \geq \xi) \geq p, \tag{2.1}$$

if g_i is linear [9]. The probability level p gives information on the reliability of the system. Thus, the probability constraint ensures the decision variables being in a safe region with given probability. For purposes of visualization, we have modeled a stochastic problem in figure 2.1. For this purpose, we assumed the x variable to be stochastic and claimed to consider a constraint for the x value. Because of the stochasticity, the optimal value is not exactly at the bound as we want the solution to be in the feasible region for a specific probability quantile, it takes an appropriate safety distance to the constraint.
However, in engineering problems the probability level p is chosen to be very high as we want to have safe systems. The concept is also used in several other branches as the finance branch, where one wants to ensure low monetary loss with a certain probability.
Generally, there exist an effort to replace constraint (2.1) with individual constraints of the form

$$Pr(g_i(\mathbf{x}, \xi) \geq 0) \geq p_i, \quad i = 1, 2, ..., r.$$

Handling individual constraints turn out to be easier than handling one constraint of form 2.1 [7]. However, as long as the constraints are not independent of each other, pulling them out from the probability integral distorts the problem. It is possible to choose the probability levels p_i such that the original constraint in (2.1) is reconstructed. But finding p_i is very hard and can lead to high values in the objective function [7].
By introducing utility functions, it is possible to represent risk-averse and risk-seeking behavior. For this purpose, several different risk measures exist. However, calculating these types of risk measures turn out to be very difficult [10]. We will discuss risk measures in chapter 4.2.2.

2.1.3 Numerical Methods to Solve Static Optimization Problems

In many cases of optimization problems, including convex problems, a closed-form solution cannot be derived. Therefore, numerical optimization methods are indispensable.
There exist gradient free and gradient-based optimization methods. In this section, we will focus on the most important gradient-based optimization methods for static, non-linear programs. The advantage of gradient-based optimization methods are fast convergence and high efficiency even for high dimensions [5]. All given information throughout this subsection can be found in detail in [5].
The general structure of a gradient-based optimization problem is given through the iteration

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \cdot \mathbf{d}^k, \quad k = 0, 1, 2, ...,$$

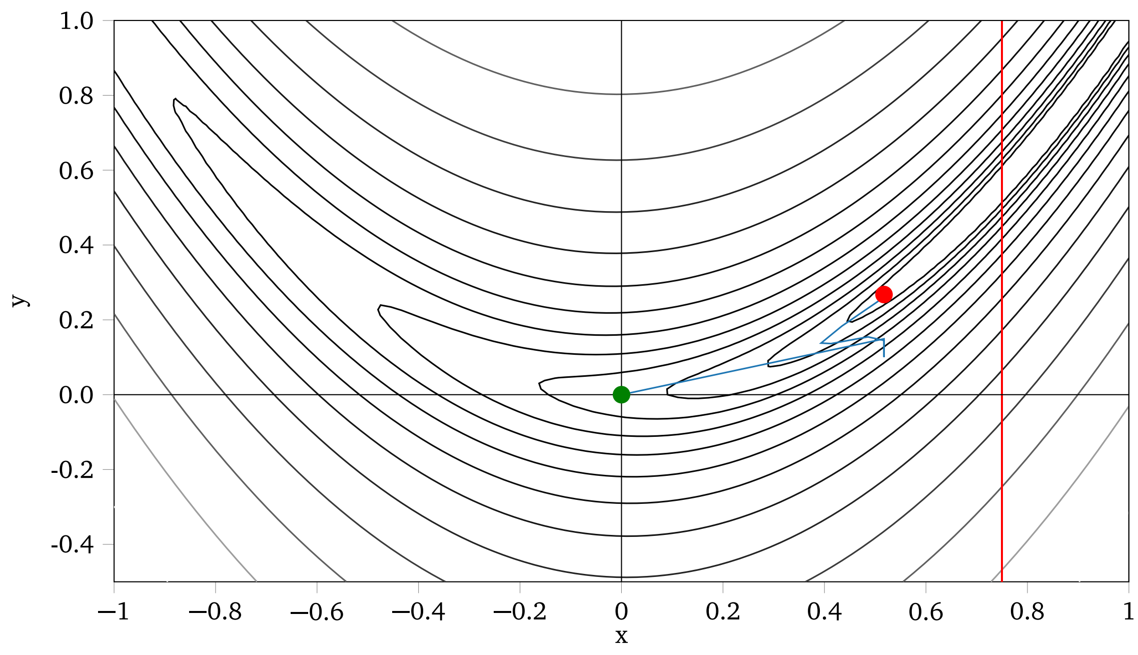

Figure 2.1: Finding the optimal value of the static Rosenbrock Function. We included a constraint at 0.75 on the x-value. Additionally, we modeled the x-values to be Gaussian distributed and thus, we have a Probabilistic constraint on the x-values. Note that we just modeled the decision variable to be random to demonstrate the distance of the endpoint to the bound. The probabilistic constraint is read as "Probability of not exceeding 0.75 shall be higher than $1-\alpha$". As we have chosen a very small value for α (0.01), the optimal value (red point) takes a high distance to the constraint, to ensure the probabilistic constraint.

where $\mathbf{d}$ is called the **search direction**, α the **stepwidth** and $\mathbf{x}$ the **decision variable**.
We first focus on numerical methods for **unconstrained** optimization problems.
The **gradient descent method** is a first order optimization method, which uses the gradient of the objective to determine the search direction [5] in iteration k

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \nabla \varphi(\mathbf{x}^k).$$

Setting α to one equals to the **steepest descent** method. Choosing the hyperparameter α correctly in each iteration is crucial for fast convergence. Thus, performing line search in each iteration to obtain the optimal value for α will improve the convergence behavior. Although first order methods only need gradient information, Newton based methods provide much faster convergence behavior, by incorporating information about the Hessian $\mathbf{H}_\varphi$.
The **Newton method** uses the Taylor series to approximate the gradient of the objective φ at the optimal value $\mathbf{x}^*$ as

$$\nabla \varphi(\mathbf{x}^*) = \nabla \varphi(\mathbf{x}^k) + \mathbf{H}_\varphi(\mathbf{x}^k)(\mathbf{x}^* - \mathbf{x}^k). \tag{2.2}$$

Using the first optimality condition in unconstrained optimization theory, equation (2.2) is set to zero and the search direction $\mathbf{d}^\mathbf{k}$ is defined as $\mathbf{x}^* - \mathbf{x}^k$, such that $\mathbf{d}^\mathbf{k}$ is the solution of the linear equation

$$\mathbf{H}_\varphi(\mathbf{x}^k)\mathbf{d}^k = -\nabla \varphi(\mathbf{x}^k).$$

Again with performing line search, the optimal value for α in the optimization iteration can be calculated.
The second optimality condition claims that the Hessian $\mathbf{H}_\varphi$ has to be positive definite, which is not always given in non-convex optimization problems. Introducing a regularization term $\mu > 0$ leads to the modified equation

$$(\mathbf{H}_\varphi(\mathbf{x}^k) + \mu \mathbf{I}) \cdot \mathbf{x}^k = -\nabla \varphi(\mathbf{x}^k).$$

Adding $\mu \cdot \mathbf{I}$ to the negative definite Hessian will lead to a modified, positive definite Hessian, which guarantees a negative search direction. However, there also exist other approximation methods for the Hessian. Care has to be taken that the Quasi-Newtion condition for approximating the Hessian

$$\tilde{H}^{k+1} \cdot \mathbf{d}^k = \nabla \varphi(\mathbf{d}^{k+1}) - \nabla \varphi(\mathbf{x}^k),$$

is fulfilled. Although the Newton method converges quadratically in the presence of local minima, it is often hard to compute the Hessian. Therefore, the **Quasi-Newton method** replaces the Hessian directly with a positive definite approximation and thus, no direct and sophisticated calculations of the Hessian are necessary anymore. A well-known approximation update in each iteration is given by the **BFGS** method.
We mentioned that performing line search is crucial for faster convergence. An alternative to optimization methods using line search is the **Trust region method**. While gradient-based methods calculate a search direction $\mathbf{d}^k$ and perform line search afterwards, optimization methods based on Trust region adapt the length and the direction of $\mathbf{d}^k$ through the optimization

$$\begin{aligned} \min_{\mathbf{d}} \quad & \nabla\varphi(\mathbf{x}^k)^T \cdot \mathbf{d} + \frac{1}{2}\mathbf{d}^T \mathbf{H}_\varphi(\mathbf{x}^k) \cdot \mathbf{d}, \\ \text{s.t.} \quad & \| \mathbf{d} \|_2 \leq \delta. \end{aligned}$$

The bounding parameter δ describes the region where the quadratized objective provides a reliable approximation of the global objective. Choosing the value of δ is decisive for the convergence of the optimization procedure.

By moving into the field of constrained optimization problems, also the optimality conditions change. Detailed description of the Karush-Kuhn-Tucker (KKT)-conditions can be found in [6]. In fact, the Lagrangian is formulated by adding the constraints, multiplied with a Lagrangian multiplier, to the objective.
However, we want to focus on some known methods on solving constrained optimization problems.
Penalty function methods transform the constrained optimization problem into an unconstrained optimization problem by augmenting the objective with penalty terms. Several forms of penalty functions exist. A well-known and widely used penalty term is given by the interior point method, which augments the objective by adding a natural logarithm of the current solution $\mathbf{x}$

$$\Phi(\mathbf{x}, \mathbf{r}) = \varphi(\mathbf{x}) - \mathbf{r} \cdot \ln(\mathbf{f}(\mathbf{x})).$$

The parameter $\mathbf{r}$ increases in each iteration to guarantee that the constraints $\mathbf{f}(\mathbf{x})$ are satisfied with a desired accuracy.

Sequential Quadratic Programing is an optimization method which was strongly investigated in the beginning of 1980. It is based on calculating the zero point of the Lagrangian-derivative using Newton's method. In general, this results in a linear equation. However, the solution of the resulting linear equation is equivalent to the solution of the Quadratic Program (QP)

$$\begin{aligned} \min_{\mathbf{d}} \quad & \varphi(\mathbf{x}^k) + \nabla\varphi(\mathbf{x}^k)^T \cdot \mathbf{d} + \frac{1}{2}\mathbf{d}^T, \mathbf{H}_\varphi(\mathbf{x}^k, \mu^k, \sigma^k) \cdot \mathbf{d}, \\ \text{s.t.} \quad & \mathbf{a}(\mathbf{x}^k) + \mathbf{J}_a^T(\mathbf{x}^k) \cdot \mathbf{d} = 0, \\ & \mathbf{f}(\mathbf{x}^k) + \mathbf{J}_b^T(\mathbf{x}^k) \cdot \mathbf{d} \geq 0, \end{aligned}$$

where the parameters μ, σ are defined as the Lagrangian multipliers, $\mathbf{a}$ is the equality constraint function and $\mathbf{J}$ is the Jacobian respectively. Approximating the Hessian in the QP should be done by the reduced Hessian, which is sparse at the dimensions, in which the constraints are active. Therefore high dimensioned optimization problems can be calculated efficiently. This is also a reason why the QP is solved instead of the linear equation. In contrast to the Newton method, SQP solves a QP in every sequence, thats where its name come from.
At this point we want to mention the special case of non-linear Least Squares optimization, since it will play a role later on.
In non-linear Least Squares, the quadratic objective

$$\varphi(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{r}_i(\mathbf{x}))^2,$$

is considered. As already mentioned, (Quasi-) Newton methods solve the optimization problem by approximating the zero point of the objective gradient with a Taylor series. To guarantee positive definiteness, the Hessian can further be modified. However, for this special case, the **Gauß-Newton** method approximates the Hessian by the equation

$$\mathbf{H}_\varphi(\mathbf{x}) \approx \mathbf{J}_\mathbf{r}(\mathbf{x}^k) \cdot \mathbf{J}_\mathbf{r}{}^T(x)^k.$$

The solution for the search direction can therefore be obtained by the linear equation

$$\left(\mathbf{J}_\mathbf{r}(\mathbf{x}^k) \cdot \mathbf{J}_\mathbf{r}{}^T(x)^k\right) \cdot \mathbf{d}^k = -\mathbf{J}_r(\mathbf{x}^k) \cdot \mathbf{r}(\mathbf{x}^k).$$

The assumption for the approximation by the Gauß-Newton method lies in small values for $\mathbf{r}(\mathbf{x})$.
If this assumption is violated, a regularization can be added to the Hessian approximation as

$$\left(\mathbf{J}_\mathbf{r}(\mathbf{x}^k) \cdot \mathbf{J}_\mathbf{r}{}^T(\mathbf{x}^k) + \gamma\mathbf{I}\right) \cdot \mathbf{d}^k = -\mathbf{J}_r(\mathbf{x}^k) \cdot \mathbf{r}(\mathbf{x}^k).$$

This is referred to as **Levenberg-Marquardt**-method. In [5] it is mentioned that adding a regularization term has high connections to the bounding parameter δ of Trust region methods.

2.2 Optimal Control

Optimal Control considers optimizing an objective function subject to the dynamics of a system. This field differs from static optimization problems, as the dynamics are formulated as a constraint to the optimization problem.

2.2.1 Problem Formulation

In [5] the OC problem is given in the form

$$\begin{aligned} \min_{\mathbf{a}} \ & J(\mathbf{a}) = \phi_T(\mathbf{s}_0, \mathbf{s}_T) + \int_0^T L(s_t, a_t) dt, \\ \text{s.t.} \ & \dot{s}_t = \mathcal{P}(\mathbf{s}_t, \mathbf{a}_t), \\ & \mathbf{s}_0 = \mathbf{b}, \\ & \mathbf{r}_T(\mathbf{s}_T) = 0, \\ & \mathbf{g}(\mathbf{s}_t, \mathbf{a}_t) \geq \mathbf{0}, \\ & \mathbf{g}(\mathbf{s}_t) \geq \mathbf{0}, \end{aligned}$$

with the transition dynamics $\mathcal{P}(\mathbf{s}_t, \mathbf{a}_t)$, initial condition $\mathbf{b}$, the terminal condition $\mathbf{c}$, the action constraints $\mathbf{g}(\mathbf{s}_t, \mathbf{a}_t)$ and the state constraints $\mathbf{g}(\mathbf{s}_t)$.
For the case of quadratic cost functions and linear dynamics and no further constraints, the Optimal Control (OC) problem can be solved in closed-form and is known as Linear Quadratic Regulator (LQR) [5].

2.2.2 Trajectory Optimization Methods

The optimization problem introduced in section 2.2 is the general optimal control formulation. The solution to this optimization problem is an interesting and well-studied issue.
Several optimization methods exist. A big distinction is made between **direct** and **indirect** methods [11]. Before discussing the methods, we briefly want to show the optimality conditions for Optimal Control problems. The optimality conditions are gained by the calculus of variations. For this purpose, the Hamiltonian is defined as

$$\mathbf{H}(\mathbf{s}, \mathbf{a}, \lambda) = L(\mathbf{s}, \mathbf{a}) + \lambda^T \mathcal{P}(\mathbf{s}, \mathbf{a}), \tag{2.3}$$

which arises by coupling the dynamics constraint with a Lagrangian multiplier [5].
The extended Mayer-term is given by

$$\Phi_t(\mathbf{s}, \nu) = \phi_t(\mathbf{s}) + \nu \mathbf{r}_t(\mathbf{s}).$$

After calculus of variations, the first order optimality conditions [5]

$$\begin{aligned} \dot{s}_i &= +\frac{\partial H}{\partial \lambda_i} = \mathcal{P}_i(\mathbf{s}, \mathbf{a}), \quad i = 1, ..., n, \\ \dot{\lambda}_i &= -\frac{\partial H}{\partial x_i} = -\frac{\partial L(\mathbf{s}, \mathbf{a})}{\partial s_i} - \sum_{k=1}^{n} \lambda_k \frac{\partial \mathcal{P}_k(\mathbf{s}, \mathbf{a})}{s_i}, \\ \frac{\partial H}{\partial a_j} &= 0, \quad j = 1, ..., n_a, \end{aligned}$$

and their corresponding boundary values are obtained [5].
Furthermore the second order optimality conditions [5] are given as

$$\frac{\partial^2 H(\mathbf{s}_t^*, \mathbf{a}_t^*, \lambda_t^*)}{\partial a_i \partial a_k} = \begin{bmatrix} \frac{\partial^2 H}{\partial a_1 \partial a_1} & \cdots & \frac{\partial^2 H}{\partial a_1 \partial a_{n_u}} \\ \vdots & \cdots & \vdots \\ \frac{\partial^2 H}{\partial a_{n_u} \partial a_1} & \cdots & \frac{\partial^2 H}{\partial a_{n_u} \partial a_{n_u}} \end{bmatrix} \geq 0.$$

For a detailed derivation of the optimality conditions we want to refer to [12].

Indirect methods make use of the shown optimality conditions, gained through the calculus of variations, and the maximum principle shown in the equations before. These methods solve the boundary value problem obtained by the optimality conditions for Optimal Control [11]. Methods counted to the group of indirect methods are for example shooting methods. The time-varying optimal actions can also be gained analytically by setting the derivative of the Hamiltonian to zero and using Newton methods to solve the equation [11]. If a Trajectory Optimization method is solved by an indirect method, it is most likely the optimal solution and therefore accurate. However, indirect methods not only require expert knowledge to derive the boundary value problem but deriving them can also be very costly [5].
Direct methods follow another concept. In these methods, the Optimal Control problem is reformulated as a non-linear program and is solved with static optimization methods. For that, the actions, and in some methods additionally the states, are discretized with piecewise polynomial functions [5]. The parameters of the discretized functions are the decision variables of the static optimization method. Shooting methods or direct collocation methods are examples for solving the Trajectory Optimization problem in the manner of direct method [5] [11]. Direct collocation method discretizes the actions as well as the states for the planning horizon. The parameters of the discretizations are the decision variables for the non-linear optimization problem. Depending on the constraints, the gradients and Jacobi matrices for direct collocation methods have a sparse structure which can be explicitly used and lead to faster convergence [5]. Direct Shooting methods for direct methods discretize the actions with piecewise polynomial functions. For many cases, linear functions are assumed. With numerical integration, the corresponding trajectory is gained. Thus, the decision variables for the non-linear program are the values for the corresponding parameterization parameters [5].
Thus, shooting methods are used in both areas of Trajectory Optimization methods. The distinction is made in the equations which are solved. Shooting methods for the indirect case solve the boundary value problem, whereas the shooting methods for the direct case solve the non-linear program gained through the discretization [13]. This also explains the accuracy of indirect methods compared to direct methods.
Dynamic Programming is another method of reformulating Trajectory Optimization problems. An interesting connection of the maximum Principle and Dynamic Programming (DP) exist. While the maximum Principle infers that minimizing the Hamiltonian returns the optimal action [5] [12]

$$H(\mathbf{s}^*, \mathbf{a}^*, \boldsymbol{\lambda}^*) = \min_{\mathbf{a}} \quad H(\mathbf{s}^*, \mathbf{a}, \boldsymbol{\lambda}^*), \tag{2.4}$$

Dynamic Programming makes usage of the Principle of Optimality [12] and introduces the Hamilton-Jacobi-Bellman (HJB) [5] equation

$$-\frac{\partial V_t(\mathbf{s})}{\partial t} = \min_{\mathbf{a}} \quad \left[L_t(\mathbf{s}, \mathbf{a}) + \left(\frac{\partial V_t(\mathbf{s})}{\partial \mathbf{s}} \right)^T \cdot \mathcal{P}_t(\mathbf{s}, \mathbf{a}) \right]. \tag{2.5}$$

The optimal solution $\mathbf{a}^*$ is given by minimizing the HJB equation. Considering the definition of the Hamiltonian in (2.3) and comparing the minimizations in (2.4, 2.5), the HJB-equation turns out to be very similar to the maximum Principle in (2.4). In fact, both minimize the general Optimal Control problem and the relation

$$\boldsymbol{\lambda}^* = \frac{\partial V_t(\mathbf{s})}{\partial \mathbf{s}}$$

exists. Thus, if the Value Function is known, calculating the optimal feedback control for every initial condition is possible [5].
For the case of linear dynamics and quadratic cost/reward functions, the value function is also quadratic and a closed form solution to this optimization problem is provided and is referred to as LQG. In section 3.1.1, when we discuss related work, this derivation is shown in terms of the Dynamic Programming principle.

2.3 Model Predictive Control

MPC is strongly related to Optimal Control. Especially in the chemical industry MPC has gained much attention due to slow dynamics. Given the current state, MPC plans the horizon by solving an optimal control problem repeatedly. It applies the first control signal of the planned horizon, measures the current state and solves again an Optimal Control problem of finite horizon [1]. By warm-starting the optimization problem with the last solution of the previous time step, the method often converges after one optimization step. Using a short horizon generally leads to faster convergence, but

it can also lead to nearsighted solutions and therefore to unwanted behavior. As long as the dynamics are slow enough, MPC can be used for online optimization. In robotics for example this is problematic [1].
Generally, MPC is formulated as a static optimization problem using direct methods which brings the advantage of handling constraints effectively (see 2.2.2) [14].
However, different types of MPC exist. The general linear MPC-framework is the case for quadratic cost and linear dynamics. If the problem is unconstrained, an LQR optimization problem is given. If further constraints on the inputs and the states exist, the optimization problem generally has to be solved by a numerical solver. Nevertheless, the optimization problem is convex [14].
For non-linear MPC an arbitrary cost function as well as a non-linear dynamic system is given [15]. As non-linear Optimal Control problems are mostly formulated as a non-linear program, non-linear MPC problems generally are solved using direct methods [15]. In some literature, a distinction between sequential and simultaneous approaches is made. In sequential approaches, the system simulation and optimization is performed after each other, whereas in the simultaneous approach both steps are performed simultaneously. Direct Collocation or direct multiple shooting methods are examples for simultaneous approach [15].
However, iLQG can be used in the MPC framework and recent research has shown remarkable results in robotics [1].

2.4 Chance Constraints

In robotics, the constraints result from mechanical design. Incorporating these constraints into the Model-Based Reinforcement Learning framework is, therefore, an obvious concept. However, since we have a MDP, it is never guaranteed that the constraints are satisfied in every optimization iteration. Besides the fact that incorporating constraints into the DP framework is very difficult, handling them in stochastic environments could lead to infeasible solutions, since there is no guaranteed knowledge of the system's state and therefore all possible solutions cannot be considered. Using **Probabilistic Constraints**, or **Chance Constraints** is a way of dealing with constraints in a stochastic environment [9].
Not only in robotics, where we want to consider action or state constraints, but also in financial markets, probabilistic constraints are common. To give an illustrative example, we consider the financial market, where one seeks to minimize risky investments. Beyond using risk-averse objectives, one could also consider the risk-neutral objective, which is the expected loss [8]. However, the expected loss does not bound the loss into a certain space, such that high losses can appear, although the expected loss is minimized. Bounding it is, therefore, an advantage. However, as already mentioned, we are acting in a stochastic environment and therefore we have to consider stochastic constraints.

2.4.1 General Form

The general Chance Constrained Stochastic Optimal Control problem is given as

$$\max_{\pi(\mathbf{s})} \quad \mathbb{E}\left[J(\mathbf{s},\mathbf{A})\right], \tag{2.6}$$

$$\text{s.t.} \quad Pr(\mathbf{s}_{0:T} \in \mathcal{S}) \geq 1-\alpha, \tag{2.7}$$

$$Pr(\mathbf{a}_{0:T-1} \in \mathcal{A}) \geq 1-\beta. \tag{2.8}$$

In this optimization problem we want to maximize the expected reward $J(\mathbf{s},\mathbf{A})$, subject to the system transition dynamics $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$, the initial state distribution $\mu(\mathbf{s}_0)$ and the state and action constraints formulated as Chance Constraints. In these, we claim that the states $\mathbf{s}_{0:T}$ are in the feasible state space $\mathcal{S}$ with probability greater or equal to 1-α and the actions $\mathbf{a}_{\mathbf{0:T-1}}$ are in the feasible action space $\mathcal{A}$ with probability $1-\beta$. Note that if the actions are independent of the stochastic states and not underlying any stochasticity, which is the case for deterministic open-loop control, constraining the actions in form of Chance Constraints does not make sense. However, if the actions are generated from a distribution, or depending on a random variable, Chance Constraints are necessary.
Evaluating Chance Constraints is a problematic issue since the probability density functions of the random variables are often not known in real applications. Additionally, calculating a multivariate integral is necessary. To still estimate the violation of the constraints, typically Monte-Carlo simulation at a given point is used. But this procedure becomes computationally costly, if we claim α to be very small, which is desired in many applications. A possibility to overcome these so-called rare-events is to use Cross-Entropy method for rare event probabilities. In the Cross-Entropy method importance sampling is used and the KL-Divergence of current probability density function and the unknown true density function is minimized [16]. However, even if the probability density function is known, it is multi-dimensional in addition and the constraint is still non-convex [17].
Nevertheless, complex mathematical approximations of Chance-Constraints are given in [17], [18], [19]. For the purpose of this thesis, we will not focus on these approximations due to assumptions on linear dynamics with quadratic loss/reward functions, which will be clarified in the next subsection.

2.4.2 Constraint Relaxations

For the case of Gaussian probability density function with linear dynamics, Chance Constraints can be approximated with two mainly known methods. The first one is by using Boole's Inequality [6] [17] [20], the second one is by using ellipsoidal approximations for the feasible sets of the Gaussian distributed random variable [21] [22].
Although we will mention the ellipsoidal relaxation method, we will mainly focus on approximating the Chance Constraint using Boole's inequality, since ellipsoidal relaxation is more conservative compared to relaxing the constraints using Boole's Inequality [22].
With Boole's Inequality, the Chance Constraint can be relaxed as [20]

$$\begin{aligned} Pr(\bigcap_{i=0}^{N} x_i \leq b_i) = 1 - Pr(\bigcup_{i=0}^{N} x_i > b_i) \geq 1 - \sum_{i=0}^{N} 1 - Pr(x_i \leq b_i) &\geq 1-\alpha, \\ \sum_{i=0}^{N} 1 - Pr(x_i \leq b_i) &\leq \alpha, \\ Pr(x_i \leq b_i) \geq 1-\alpha_i, \ \sum_{i=0}^{N} \alpha_i &\leq \alpha. \end{aligned} \tag{2.9}$$

Thus, the probability on the left hand side can be conservatively approximated by a sum of univariate single probabilities [20] [22].
Knowing, that x_i is a normal distributed random variable with mean μ_{x_i} and variance Σ_{x_i}, the cummulative density function (cdf) Φ of normal distributed random variables can be used, to calculate the probability $Pr(x_i \leq b_i)$ [22] as

$$Pr(x_i \leq b_i) = \Phi\left(\frac{b_i - \mu_{x_i}}{\sqrt{\Sigma_{x_i}}}\right).$$

A Chance Constraint

$$Pr(x_i \leq b_i) \geq 1 - \alpha_i,$$

can therefore be written as

$$\Phi\left(\frac{b_i - \mu_{x_i}}{\sqrt{\Sigma_{x_i}}}\right) \geq 1 - \alpha_i.$$

The cdf of the normal distribution is given by the errorfunction (erf) [20]

$$\Phi\left(\frac{b_i - \mu_{x_i}}{\sqrt{\Sigma_{x_i}}}\right) = \frac{1}{2}\left[1 + erf\left(\frac{b_i - \mu_{x_i}}{\sqrt{2\Sigma_{x_i}}}\right)\right].$$

In general, the errorfunction can not be calculated analytically, but it can be approximated by the Maclaurin series [20].
The Chance Constraint can therefore be reformulated as

$$\begin{aligned} \frac{1}{2}\left[1 + erf\left(\frac{b_i - \mu_{x_i}}{\sqrt{2\Sigma_{x_i}}}\right)\right] &\geq 1 - \alpha_i, \\ erf\left(\frac{b_i - \mu_{x_i}}{\sqrt{2\Sigma_{x_i}}}\right) &\geq 1 - 2\alpha_i, \\ b_i - \mu_{x_i} &\geq \sqrt{2\Sigma_{x_i}} erf^{-1}(1 - 2\alpha_i), \\ b_i - \mu_{x_i} - \sqrt{2\Sigma_{x_i}} erf^{-1}(1 - 2\alpha_i) &\geq 0. \end{aligned} \tag{2.10}$$

If it is known that the decision variable of the optimization problem is linear in μ_{x_i}, which it is for our case, how we will see later, then the last inequality in (2.10) is linear and therefore convex [6] [20] [22]. Note that the terms b_i, $\sqrt{2\Sigma_{x_i}} erf^{-1}(1 - 2\alpha_i)$ are constants, then.
In general, the constraint

$$Pr(x_{0:N} \leq b) \geq 1 - \alpha \tag{2.11}$$

is reformulated as $N + 1$ inequalities

$$b_i - \mu_{x_i} - \sqrt{2\Sigma_{x_i}} erf^{-1}(1 - 2\alpha_i) \geq 0.$$

Care has to be taken when choosing the probability value α. If one wants to relax the Chance Constraints using Boole's Inequality, it has to be ensured that the argument of the inverse errorfunction is smaller than 1, since it is not defined for values ≥ 1 and ≤ -1. As we know for the derived example, the argument in the linear inequality is $1-2\alpha_i$. Thus, we have to ensure $\alpha_i < 0.5$. Since probabilities can not be negative, the range of α_i is $0 \leq \alpha_i < 0.5$. In general, this issue is addressed by claiming the total probability α to be < 0.5 and defining the single probabilities α_i as [20]

$$\sum_{i=1}^{N} \alpha_i \leq \alpha.$$

This condition makes this type of relaxation conservative. However, in nearly all robotic applications we want to have a high confidence level $1-\alpha$, because we want to cover as much realizations as possible. Thus, we set α to a very low probability level from the beginning.

A similar inequality is obtained by using the **ellipsoidal relaxation** method [22]

$$b_i - \mu_{x_i} - r\sqrt{\Sigma_{x_i}} \geq 0,$$

where r denotes the radius of the ellipsoid $\varepsilon_r = \{\xi : \xi^T \Sigma_{x_i}^{-1} \xi \leq r^2\}$ and it holds that $\mu_{x_i} + \varepsilon_r$ is in the feasible set of x. However, we will, for already mentioned reasons, not go further into detail for this method.

For the case of unknown probability density functions, or for the case, that one does not want to restrict to the probability level to $\alpha < 0.5$, the **scenario approach** is a possible method to use.
In [23] the authors introduce the Scenario approximation of Chance Constraints by using stochastic sampling. They introduce the unknown probability density function $f(\mathbf{x})$ as

$$Pr(\mathbf{x} \in P) = \int_{\mathbf{x} \in P} f(\mathbf{x}) d\mathbf{x} \geq 1 - \alpha,$$

where P denotes the feasible set of $\mathbf{x}$. The probability density $f(\mathbf{x})$ is approximated by

$$f(\mathbf{x}) \approx \frac{1}{s} \sum_{j=1}^{s} \delta(\mathbf{x} - \mathbf{d}_j),$$

where s is the number of samples, $\mathbf{d}_j$ denote the samples from $\mathbf{x}$ and δ is the dirac function. The Probability of the left hand side of the Chance Constraint is approximated by

$$Pr(\mathbf{x} \in P) \approx \frac{1}{s} \sum_{j=1}^{s} \mathbf{1}_P(\mathbf{d}_j).$$

The indicator function is defined as

$$\mathbf{1}_P(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in P, \\ 0, & \mathbf{x} \notin P. \end{cases}$$

The samples $\mathbf{d}_j$ can be obtained for example by sampling from a multivariate Gaussian, where the mean is given by the current $\mathbf{x}$ and the Covariance is predefined as sample variance $\Sigma_{\mathbf{d}}$. All in all, the Chance Constraint approximation for the scenario method is given by [23]

$$Pr(\mathbf{x} \in P) \approx \frac{1}{s} \sum_{j=1}^{s} \mathbf{1}_P(\mathbf{d}_j) \geq 1 - \alpha. \tag{2.12}$$

The advantage of this method is to be applicable for unknown density functions and to have no restrictions on choosing the probability level α. One disadvantage is that while the accuracy increases with the number of samples, the computational effort also increases. For some cases, it can also happen that the approximation (2.12) presents a feasible set with probability 1, whereas it can contain solutions, which are infeasible for the original problem (2.11) (left side of inequality) [19].

2.4.3 Relation to Robust Optimization

We have discussed approximations and relaxations of Chance Constraints in the last chapter since they are not tractable or computable for the most cases. Except for the Gaussian case, approximating the probabilities in a convex manner turns out to be very hard.
However, convex approximations exist. For detailed review we refer to [17] [18] [19] [9] [7].
Approximating Chance Constraints turns out to have a relation to robust Optimization. In [19] the author addresses this relation as described below. Assume the linear constraint in (2.11) without formulating it as a Chance Constraint

$$\mathbf{x} \leq \mathbf{b}. \tag{2.13}$$

Further assume, that $\mathbf{x}$ is a random variable with unknown probability density function. For the case of Robust Optimization, it is desired that the constraint is in a convex uncertain set Z, where the constraint is formulated as

$$\mathbf{x} \leq \mathbf{b}, \quad \forall \mathbf{x} \in Z.$$

Robust feasible solutions are realizations of $\mathbf{x}$ which are within the feasible set.
The Stochastic Optimization point of view treats the inequality (2.13) stochastic with a certain probability density function P and therefore connects the Chance Constraint (2.11) to it.
However, it can be shown that for a certain group of Chance Constraint approximations, which are called *normal approximations* [19] the set of possible solutions to the Chance Constraint (2.11) are the same set Z from the Robust Optimization point of view. For further information and details, we want to refer to [19].
The same procedure can be transferred to our case in this thesis, where we treat linear dynamics. In general, we want to find a solution to our optimization problem with constraints, which can be stochastic state and/or input constraint, such that the solution holds for a possible set Z for the Robust Optimization point of view and for the possible realizations in a $1-\alpha$ quantile, for the Stochastic Optimization point of view. A strong relation to robust control is therefore given.

2.5 Unscented Transformation

As we will use the Unscented Transformation in our approach for Regularized Stochastic Optimization, we briefly want to show the concept of Unscented Transformation (UT) in this section.
In [24] the weaknesses of Extended Kalman Filter (EKF) are addressed by introducing the Unscented Kalman Filter (UKF). UKF uses the UT to provide the statistics of a random variable $\mathbf{x}$ after a non-linear transformation $\mathbf{y} = \mathrm{g}(\mathbf{x})$. In this section we briefly want to present the way UT works, according to [24] and [25].
For a random variable $\mathbf{x}$ with mean $\boldsymbol{\mu}_x$ and covariance $\boldsymbol{\Sigma}_x$, UT calculates the corresponding moments after a non-linear transformation $\mathbf{y} = \mathrm{g}(\mathbf{x})$. For this purpose a matrix $\boldsymbol{\chi}$ with $2 \cdot n + 1$ Sigma vectors $\boldsymbol{\chi}_i$ is defined. Note that n denotes the system's dimension. The Sigma vectors are chosen according to

$$\boldsymbol{\chi}_0 = \boldsymbol{\mu}_x, \tag{2.14}$$
$$\boldsymbol{\chi}_i = \boldsymbol{\mu}_x + \left(\sqrt{(n+\lambda)\boldsymbol{\Sigma}_x}\right)_i \quad i = 1, \ldots, n, \tag{2.15}$$
$$\boldsymbol{\chi}_i = \boldsymbol{\mu}_x - \left(\sqrt{(n+\lambda)\boldsymbol{\Sigma}_x}\right)_i \quad i = n+1, \ldots, 2n. \tag{2.16}$$

Furthermore the weights

$$w_0^m = \frac{\lambda}{n+\lambda}, \tag{2.17}$$
$$w_0^c = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta), \tag{2.18}$$
$$w_i^m = w_i^c = \frac{1}{2(n+\lambda)} \quad i = 1, \ldots, 2n, \tag{2.19}$$

are introduced. The scaling parameter λ is given through the equation $\lambda = \alpha^2(n+\kappa) - n$, where α determines the distance of the sigma points from the mean $\boldsymbol{\mu}_x$ and is generally set to a small value. The secondary scaling parameter κ is generally set to zero and β is used to incorporate prior knowledge and is set to $\beta = 2$ for Gaussian distributions. Note that the square root in the equations donate the ith row of the matrix square root.
After obtaining the sigma points, they are evaluated with non-linear transformation $\psi_i = g(\chi_i)$ for $i = 1, \ldots, 2n$. Using

the transformed Sigma points, the mean $\boldsymbol{\mu}_y$ and the covariance $\boldsymbol{\Sigma}_y$ of the non-linear density transformation are estimated with

$$\boldsymbol{\mu}_y \approx \sum_{i=0}^{2n} w_i^m \boldsymbol{\psi}_i, \tag{2.20}$$

$$\boldsymbol{\Sigma}_y \approx \sum_{i=0}^{2n} w_i^c (\boldsymbol{\psi} - \boldsymbol{\mu}_y)(\boldsymbol{\psi} - \boldsymbol{\mu}_y)^T. \tag{2.21}$$

3 Chance Constrained Iterative Linear Quadratic Gaussian

In this section we are going to describe our approach to consider constraints in the DDP framework to improve performance. Considering additional state constraints to the dynamics is challenging. In DP, constraints are generally incorporated by augmenting the cost function with penalty terms [26]. We want to show, a new approach which additionally solves an optimization problem after the backward pass of iLQG, to consider constraints for the linearized dynamics, in order to ensure additional validity of the linearized dynamics.

3.1 Related Work

Before presenting our approach, we want to present the works which inspired the development of this thesis. We will have a detailed insight into DDP and iLQG. In addition to the mathematical description of the algorithms, we will classify them into the appropriate Trajectory Optimization methods. We further will show other related work.

3.1.1 Differential Dynamic Programming (DDP) and iterative Linear Quadratic Gaussian (iLQG)

First introduced in [27], Differential Dynamic Programming is an iterative trajectory optimization method, which can deal non-linear dynamics and non quadratic reward functions. It uses the Dynamic Programming principle [28] to break down the optimization problem into a single control problem. By linearizing the dynamics around trajectories and approximating the state action-value function, which is also referred to as Q-function, or the Hamiltonian (see 2.2.2), locally with second order Taylor approximation, the algorithm fulfills the LQG assumptions. Thus, a linear local optimal policy can be calculated in closed-form. The iLQG algorithm is strongly related to the DDP algorithm with the main difference in neglecting the second-order derivatives in the backward pass and adding some improved regularizations as shown later. Thus, the algorithmic flow of both algorithms is very similar. First, an initial action sequence is applied to the non-linear dynamics to obtain the initial trajectory. Then, in each time step, a local model is gained by linearizing around the trajectory. The state-action value function is approximated as a quadratic function. By this, an LQG problem is formulated and a local optimal control policy is gained. With the new policy, the non-linear system is executed and the progress is repeated.

Mathematical Formulation of DDP

Having an initial trajectory $\tau = \{\mathbf{s_1}, \mathbf{a_1}, ..., \mathbf{s_T}, \mathbf{a_T}\}$ generated by the dynamics $\mathcal{P}(s_{t+1}|s_t, a_t)$ and the action sequence $\mathbf{A}$, the algorithm tries to maximize the reward

$$J(\mathbf{s}, \mathbf{A}) = \sum_{t=0}^{T-1} R_t(\mathbf{s}_t, \mathbf{a}_t) + R_T(\mathbf{s_T}),$$

by finding the optimal action sequence $\mathbf{A}^*$.
The algorithm uses the Dynamic Programming Principle to reduce the optimization problem from finding a sequence of actions to a single action, solved backwards in time. Thus, the reward to go is defined as

$$J_t(\mathbf{s}, \mathbf{A_t}) = \sum_{j=t}^{T-1} R_j(\mathbf{s}_j, \mathbf{a}_j) + R_T(\mathbf{s}_T).$$

Maximizing the reward to go leads to the state Value function

$$V_t(\mathbf{s}) = \max_{\mathbf{A}_t} \; J_t(\mathbf{s}, \mathbf{A}_t).$$

Thus, the Value function can be calculated backwards in time by the recursion

$$V_t(\mathbf{s}) = \max_{\mathbf{a}} \; [R(\mathbf{s}, \mathbf{a}) + \mathbb{E}[V_{t+1}(\mathbf{s}_{t+1})]],$$

which can be written as

$$V_t(\mathbf{s}) = \max_{\mathbf{a}} \left[R(\mathbf{s},\mathbf{a}), + \sum_{\mathbf{s}_{t+1}} V_{t+1}(\mathbf{s}_{t+1}) \mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s},\mathbf{a}) \right],$$
$$= \max_{\mathbf{a}} \left[R(\mathbf{s},\mathbf{a}) + V_{t+1}\left(\boldsymbol{\mathcal{P}}(\mathbf{s},\mathbf{a})\right) \right].$$

Note that linear dynamic systems $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s_t},\mathbf{a_t})$ with zero mean Gaussian noise are considered. Introducing the perturbed argument of the optimization problem leads to the perturbed state-action Value function

$$Q_t(\delta\mathbf{s},\delta\mathbf{a}) = R_t(\mathbf{s}+\delta\mathbf{s}, a+\delta\mathbf{a}) - R_t(\mathbf{s},\mathbf{a}) + V_{t+1}\left(\mathcal{P}(\mathbf{s}+\delta\mathbf{s},\mathbf{a}+\delta\mathbf{a})\right) - V_{t+1}\left(\mathcal{P}(\mathbf{s},\mathbf{a})\right),$$

which is approximated with a second order Taylor series to

$$Q_t(\delta\mathbf{s},\delta\mathbf{a}) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{s} \\ \delta\mathbf{a} \end{bmatrix}^T \begin{bmatrix} 0 & Q_{s,t}^T & Q_{a,t}^T \\ Q_{s,t} & Q_{ss,t} & Q_{sa,t} \\ Q_{a,t} & Q_{as,t} & Q_{aa,t} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{s} \\ \delta\mathbf{a} \end{bmatrix},$$

where the elements of the matrix are calculated as

$$\begin{aligned} Q_{s,t} &= R_{s,t} + \mathcal{P}_{s,t}^T V_{s,t+1}, \\ Q_{a,t} &= R_{a,t} + \mathcal{P}_{a,t}^T V_{s,t+1}, \\ Q_{ss,t} &= R_{ss,t} + \mathcal{P}_{s,t}^T V_{ss,t+1} \mathcal{P}_{s,t} + V_{s,t+1} \mathcal{P}_{ss,t}, \\ Q_{aa,t} &= R_{aa,t} + \mathcal{P}_{a,t}^T V_{ss,t+1} \mathcal{P}_{a,t} + V_{s,t+1} \mathcal{P}_{aa,t}, \\ Q_{as,t} &= R_{as,t} + \mathcal{P}_{a,t}^T V_{ss,t+1} \mathcal{P}_{s,t} + V_{s,t+1} \mathcal{P}_{as,t}. \end{aligned}$$

Solving the optimization problem

$$\operatorname*{argmax}_{\delta\mathbf{a}} \; Q(\delta\mathbf{s},\delta\mathbf{a})$$

leads to the local optimal linear policy

$$\delta\mathbf{a}^* = -Q_{aa,t}^{-1}(Q_a + Q_{as,t}\delta\mathbf{s}) = \mathbf{K}_t \delta\mathbf{s} + \mathbf{k}_t.$$

To update the Value function, the optimal policy $\delta\mathbf{a}^*$ is plugged into the perturbed state action value function $Q(\delta\mathbf{s},\delta\mathbf{a})$ and the quadratic value function update is obtained

$$\begin{aligned} \Delta V_t &= -\frac{1}{2} Q_{a,t} Q_{aa,t}^{-1} Q_{a,t}, \\ V_{s,t} &= Q_{s,t} - Q_{a,t} Q_{aa,t}^{-1} Q_{as,t}, \\ V_{ss,t} &= Q_{ss,t} - Q_{sa,t} Q_{aa,t}^{-1} Q_{as,t}. \end{aligned}$$

After calculating the optimal policy for every time step t in the planning horizon, a new trajectory is calculated in the forward pass

$$\begin{aligned} \hat{\mathbf{s}}_0 &= \mathbf{s}_0, \\ \hat{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{k}_t + \mathbf{K}_t(\hat{\mathbf{s}}_t - \mathbf{s}_t), \\ \hat{\mathbf{s}}_{t+1} &= \mathcal{P}(\hat{\mathbf{s}}_{t+1}|\hat{\mathbf{s}}_t, \hat{\mathbf{a}}_t). \end{aligned}$$

For the purpose of this thesis, we have to mention at this point, that we have state distributions, due to MDP.
However, care has to be taken during the policy draft, since the method exploits greedily the local linear dynamics [4], such that in each iteration the obtained policies can be highly different. This can lead to oscillations and unstable behavior by violating the validity region of the linearized dynamics. Applying a regularization to the Hessian $Q_{aa,t}$ does not only guarantee always positive definiteness of the Hessian, it also addresses the problem of highly exploiting the linearized dynamics by punishing high changes in the policy

$$\tilde{Q}_{aa,t} = Q_{aa,t} + \mu I.$$

We already have discussed such a regularization in chapter 2.1.3, when Levenberg-Marquardt method was introduced, where we have mentioned that this method has high relation to the bounding parameter of the Trust region methods. In the context of DDP, the Levenberg-Marquardt parameter is adding a quadratic cost around the current policy sequence, such that the policy updates are more conservative and not highly different in every time step and thus, the validity of the linearized dynamics are tried to keep. The parameter μ is updated in each iteration of DDP depending on the reward change compared to the iteration before. If the algorithm could increase the cumulative expected reward, the algorithm decreases μ by a predefined factor. If the reward decreases, μ is increased.

Mathematical Formulation of iLQG

Introduced in [29], Iterative Linear Quadratic Gaussian is very similar to DDP. In fact, the mathematical background of the forward and the backward pass are the same, except for neglecting the second order derivatives of the dynamics $\mathcal{P}_{aa,t}, \mathcal{P}_{ss,t}, \mathcal{P}_{as,t}$ in the equation (3.1.1). Furthermore, in [1] the authors introduce new regularizations, which do not depend on the control perturbation Q_{aa} directly, but rather on the deviations from the states. The regularizations are introduced as

$$\begin{aligned}\tilde{Q}_{aa,t} &= R_{aa,t} + \mathcal{P}_{a,t}^T \left(V_{ss,t+1} + \mu \mathbf{I}\right) \mathcal{P}_{a,t} + V_{s,t+1}\mathcal{P}_{aa,t},\\ \tilde{Q}_{as,t} &= R_{as,t} + \mathcal{P}_{a,t}^T \left(V_{ss,t+1} + \mu \mathbf{I}\right) \mathcal{P}_{s,t} + V_{s,t+1}\mathcal{P}_{as,t}.\end{aligned}$$

The modified Value function updates are adjusted to

$$\begin{aligned}\Delta V_t &= \frac{1}{2}\mathbf{k}_t^T Q_{aa,t}\mathbf{k}_t + \mathbf{k}_t^T Q_{a,t},\\ V_{s,t} &= Q_{s,t} + \mathbf{K}^T Q_{aa,t}\mathbf{k}_t + \mathbf{K}^T Q_{a,t} + Q_{as}^T \mathbf{k}_t,\\ V_{ss,t} &= Q_{ss,t}\mathbf{K}^T Q_{aa,t}\mathbf{K}_t + \mathbf{K}^T Q_{as,t} + Q_{as}^T \mathbf{K}_t.\end{aligned}$$

By introducing this regularization, the authors address the problem that the policy can have high changes depending on the control-transition matrix $\mathcal{P}_{aa,t}$. Thus, adding a regularization on the states should be more effective than adding a regularization on the policy.
In order to improve the convergence of the algorithm, the authors additionally introduce a line search parameter $0 < \alpha \leq 1$, which leads to applying only a certain percentage of the forward term $\mathbf{k}_t$ of the action

$$\hat{\mathbf{a}}_t = \mathbf{a}_t + \alpha\mathbf{k}_t + \mathbf{K}_t(\hat{\mathbf{s}}_t - \mathbf{s}_t).$$

The optimal value for α is calculated with the reward change

$$\Delta J(\alpha) = \alpha \sum_{t=1}^{T-1} \mathbf{k}_t^T Q_{a,t} + \frac{\alpha^2}{2} \sum_{t=1}^{T-1} \mathbf{k}_t^T Q_{aa,t}\mathbf{k}_t.$$

By setting the actual and the expected reward in relation

$$z = \frac{J(\hat{\mathbf{A}}) - J(\mathbf{A})}{\Delta J(\alpha)},$$

the iteration is only accepted, if

$$0 < c_1 < z.$$

Otherwise α is modified appropriately and the forward pass is performed again.
In our implementation of iLQG we have used regularization (3.1.1).

Classification in Trajectory Optimization Methods

Trajectory Optimization methods were already discussed in section 2.2.2. In this subsection, we want to classify DDP and iLQG into the group of Trajectory Optimization (TO) methods.
DDP and iLQG are gaining an optimal policy by maximizing the Hamiltonian, which is used to integrate the dynamics to obtain the corresponding trajectories. Thus, they are performing a shooting method to obtain the state-action trajectory by maximizing state action Value function, or the Hamiltonian gained by Dynamic Programming. As they directly make usage of the Hamiltonian, DDP and iLQG are categorized into the group of indirect methods [30].
Furthermore, DDP is a second-order shooting method, with quadratic convergence for any system with smooth dynamics [30]. During the backward pass the step taken by DDP corresponds to the full Newton step (see section 2.1.3), where the search direction is approximated with second-order Taylor approximation and the Hessian (H_φ in section 2.1.3 and $Q_{aa,t}$ in section 3.1.1) to be inverted is regularized in the same way [1].
In contrast, iLQG drops the second-order derivatives of the dynamics. Thus, it corresponds to the Gauß-Newton approximation for non-linear Least Squares (see 2.1.3), where the Hessian is calculated with the information of the gradients only. The regularization in the iLQG interpretation as Gauß-Newton method is therefore equivalent to the Levenberg-Marquardt regularization as introduced in section 2.1.3.

3.1.2 Constraint-Tightening and Stability in Stochastic Model Predictive Control

In their work [31], the authors address handling constraints in Stochastic Model Predictive Control (SMPC) by finding a nominal bound which guarantees a considering a state constraint with a certain probability. This bound is calculated in an offline optimization problem. In fact, it can be interpreted as finding the optimal distance to the actual state bound in order to comply with the state constraint with certain probability. The approach is restricted to linear-time invariant systems of the form

$$\mathbf{s}_{t+1} = \mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t + \mathbf{B}_w\mathbf{w}_t,$$

where the disturbance w_t is added on the states through the matrix B_w. The disturbance sequence $\mathbb{W}$ is further bounded and is convex with a zero mean independent and identically distribution. Throughout their work, the authors mention that $\mathbb{W}$ is sampled offline. The linear uncertain system is constrained on the state probabilistically and hard on the actions

$$\begin{aligned} Pr([\mathbf{H}]_j\mathbf{s}_{k+l} \geq [\mathbf{h}]_j\mathbf{s}_k) &\geq 1-[\epsilon]_j, \\ G\mathbf{a}_{k+l} &\geq g. \end{aligned}$$

In linear SMPC and robust MPC the system's state $s_{l|k}$ at current horizon k and prediction step l is commonly split into a nominal and error part

$$\mathbf{s}_{l|k} = \mathbf{z}_{l|k} + \mathbf{e}_{l|k},$$

with the expected value $z_{l|k} = \mathbb{E}(s_{l|k})$ and the zero mean stochastic error part $e_{l|k}$. A linear control policy $a_{l|k} = Ke_{l|k} + v_{l|k}$ is used, where the feedback gain is used to stabilize the error system and the forward term $v_{l|k}$ is used to drive the system's state into a desired space. The forward term $v_{l|k}$ is the decision variable in the SMPC problem. The dynamics of the system can therefore be summarized as

$$\begin{aligned} \mathbf{z}_{l+1|k} &= \mathbf{A}\mathbf{z}_{l|k} + \mathbf{B}\mathbf{v}_{l|k}, \\ \mathbf{e}_{l+1|k} &= \mathbf{A}_{cl}\mathbf{e}_{l|k} + \mathbf{B}_w\mathbf{W}_{l+k}. \end{aligned}$$

Generally, in MPC it is important to be fast in calculating the optimal actions, since the optimization problem is solved online. To reduce computation issues, the authors tighten the probabilistic constraints offline, where they define a convex, linear constraint set $\mathbb{Z}_l$ on the predicted nominal state $z_{l|k}$. Defining $\mathbb{Z}_l$ is an optimization problem itself, where the bounds η_l of the predicted states are optimized for the horizon $l \in [1, T]$

$$\mathbb{Z}_l = \{\mathbf{z} \in \mathbb{R}^n \mid \mathbf{H}\mathbf{z} \leq \eta_l\}.$$

The bounds η_l have to be optimized respecting the probabilistic constraints induced by the probabilistic error system $e_{l|k}$

$$\begin{aligned} \eta_l = \max_{\eta} &\quad \eta \\ s.t. &\quad Pr(\eta \leq [h]_j - [\mathbf{H}]_j\mathbf{e}_{l|k} \geq 1-[\epsilon]_j). \end{aligned}$$

Note, that $e_{l|k}$ can be evaluated directly, since the disturbances have been sampled offline. A detailed derivation is given in their work. In general, the idea of this optimization problem is to search the the distance of the nominal state to the constraint, such that with probability $1-\epsilon_j$ the constraint are kept. The distance is determined by the error system, since it implies the stochasticity. After determining the nominal set $\mathbb{Z}$, which gives the state bounds, the Finite Horizon Optimal Control Problem can be formulated as

$$\begin{aligned} \min_{\mathbf{v}_{T|k}} &\quad J(\mathbf{v}_{T|k}), \\ \text{s.t.} &\quad \mathbf{z}_{l+1|k} = \mathbf{A}\mathbf{z}_{l|k} + \mathbf{B}\mathbf{v}_{l|k}, \\ &\quad \mathbf{z}_{l|k} \in \mathbb{Z}_l, \quad l \in [1, T], \\ &\quad \mathbf{v}_{l|k} \in \mathbb{V}_l, \quad l \in [0, T-1], \\ &\quad \mathbf{z}_{T|k} \in \mathbb{Z}_f, \end{aligned}$$

where J denotes a quadratic cost function and $\mathbb{V}$ the feasible set for the forward term. Thus, the SMPC has changed to an MPC problem.

3.1.3 Progressive Closed-Loop Chance-Constrained Control

Several Approaches of dealing with Chance Constraints in Control theory have been considered. In [23] the authors introduce a new approach, where the probabilistic state-constraints are augmented into the objective function progressively.
Similar to [31], the authors start by first defining a time-varying LQG controller

$$\mathbf{a}_t = \mathbf{K}_t \mathbf{s}_t + \mathbf{k}_t$$

from the unconstrained optimization problem for the time-varying linear system

$$\mathbf{s}_{t+1} = \mathbf{A}_t \mathbf{s}_t + \mathbf{B}_t \mathbf{a}_t + \mathbf{w}_t,$$

where w_t is zero mean i.i.d. Gaussian noise. Furthermore, a quadratic cost function for a finite horizon T is considered. Chance Constraints on the state of the form

$$Pr(\mathbf{F} \cdot [\mathbf{s}_0^T, ..., \mathbf{s}_T^T]^T \leq \mathbf{b}) \geq 1 - \alpha,$$

are given. Throughout their work, the authors use an *augmented system* notation, which is common in the MPC framework and transforms the dynamic optimization problem into a static problem. We will also use this notation throughout this thesis and therefore introduce it later.
By formulating the closed loop system

$$\mathbf{s}_{t+1} = \mathbf{A}_{\mathbf{cl},t} \mathbf{s}_t + \mathbf{B}_t \mathbf{k}_t + \mathbf{w}_t,$$

remaining the feedback gain $\mathbf{K}_t$ for every time step, and formulating $\mathbf{k}_t$ as the decision variable, the optimization problem can be formulated as an optimal control problem of the form

$$\begin{aligned} \min_{\mathbf{k}_t} \quad & \mathbb{E}[J(\mathbf{k}_t)], \\ \text{s.t.} \quad & \mathbf{s}_{t+1} = \mathbf{A}_{\mathbf{cl},t} \mathbf{s}_t + \mathbf{B}_t \mathbf{k}_t + \mathbf{w}_t, \\ & Pr\left(\mathbf{F} \cdot [\mathbf{s}_0^T, ..., \mathbf{s}_T^T]^T \leq \mathbf{b}\right) \geq 1 - \alpha, \\ & \mathbb{E}(\mathbf{s}_0) = \hat{\mathbf{s}}_0. \end{aligned}$$

In their approach, the authors reformulate the optimization problem as a direct method by plugging the augmented states into the objective, thus eliminating the dynamics constraint and further evaluating the expected value, which leads to closed form formulation. The closed form is possible since a quadratic cost function with linear dynamics and Gaussian noise is assumed.
Furthermore the scenario approximation introduced in section 2.4.2 is used to evaluate the probabilities in each iteration. By defining $\tilde{J}$ as the augmented cost function, and applying scenario approximation on the state constraints, obtaining the samples $\mathbf{d}_j$, the optimization problem is formulated as

$$\begin{aligned} \min_{\mathbf{k}} \quad & \tilde{J}(\mathbf{k}), \\ \text{s.t.} \quad & \frac{1}{s} \sum_{j=1}^{s} \mathbf{1}_P(\mathbf{d}_j) \geq 1 - \alpha. \end{aligned}$$

This optimization problem is deterministic. However, finding solutions to this optimization problem turns out to be difficult, as an initial feasible solution is needed for the most numerical optimization algorithm, as for the Interior-Point method for example. Therefore, the optimization problem is reformulated as an unconstrained problem by introducing the approximated state constraints in form of a penalty term into the objective.
Several choices of punishment terms exist. In their work, the authors have chosen an exponential barrier function, which punishes positive distances of the sampled states to the bound $\mathbf{b}$

$$p_\gamma(\mathbf{d}_j) = \sum_{i=1}^{g} \exp\left(\gamma \cdot (\mathbf{F}_{i,:} \cdot \mathbf{d}_j - \mathbf{b}_i)\right).$$

Note that the index g denotes the number of constraints, $\mathbf{F}_{i,:}$ is the i-th row of $\mathbf{F}$ and $\gamma > 0$ is a scaling factor. In order to consider probabilities, a set $Z \subset \{1, ..., s\}$ which contains the indices of the $\alpha \cdot s$ samples with the largest punishment values $p_\gamma(\cdot)$.
The new augmented cost function is formulated as

$$\tilde{J}_\gamma = \tilde{J} + \frac{\gamma}{s} \sum_{j=1, j \notin Z}^{s} p_\gamma(\mathbf{d}_j), \tag{3.1}$$

which is an unconstrained optimization problem. In order to achieve a certain accuracy, γ has to be increased in each optimization step. This means, that one starts with a certain value for γ, solves the optimization problem, evaluates the Chance Constraint, resolves the problem with an increased value for γ, if the probability is not achieved.

3.1.4 On Optimal Control Of Stochastic Linear Hybrid Systems

In [20] an Optimal Control problem for Hybrid Systems is considered. Two optimization problems are addressed by the authors. First, the optimal switching times are obtained by an Optimization problem. Second, which is more important for the purpose of this thesis, Chance Constraints with respect on the states are solved. The systems, which are considered, are of linear Gaussian nature with additive noise.
After obtaining the optimal switching times between modes, the authors use Boole's Inequality to obtain a deterministic constraint as described in 2.4.2 for each of these modes. Thus, an optimal Control policy for the whole Hybrid System is obtained. The considered optimization subject to the probabilistic state constraints is an open-loop Optimal Control problem.

3.1.5 Control-Limited Differential Dynamic Programming

In [30] the authors present an approach which incorporates action constraints into the DDP/iLQG framework. They are considering box constraints of the form $\underline{\mathbf{b}} \geq \mathbf{a} \geq \overline{\mathbf{b}}$, where $\underline{\mathbf{b}}$ denotes the lower bound and $\overline{\mathbf{b}}$ denotes the upper bound of the action limits. As naively clamping in the forwardpass does not incorporate informations of constraints into the backward pass, the optimization procedure will return action steps, which will violate the constraints again. Furthermore this clamping does not improve convergence.
Therefore a solution in the backwardpass has to be considered. In their work, the authors consider the optimization problem

$$\begin{aligned} &\min_{\delta \mathbf{a}} \ Q(\delta \mathbf{a}, \delta \mathbf{s}) + \mu \frac{\| \delta \mathbf{a} \|^2}{2}, \\ &s.t. \ \underline{\mathbf{b}} \leq \mathbf{a} + \delta \mathbf{a} \leq \overline{\mathbf{b}}, \end{aligned}$$

in each backwardpass. Note that $Q(\delta \mathbf{s}, \delta \mathbf{a})$ denotes the perturbed Q-function as described in 3.1.1. The authors use a more general description of the optimization process with

$$\begin{aligned} &\min_{\delta \mathbf{x}} \ f(\mathbf{x}) = \mathbf{x}^T \mathbf{q} + \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x}, \\ &s.t. \ \underline{\mathbf{b}} \leq \mathbf{x} \leq \overline{\mathbf{b}}. \end{aligned}$$

The proposed algorithm iteratively identifies the violated constraints and sets the feedback gains in these constraints to zero. This is done by a *Projected-Newton Solution*. First the gradient $\mathbf{g} = \nabla_{\mathbf{x}} f = \mathbf{q} + \mathbf{H}\mathbf{x}$ and the sets of clamped and free indices c and f

$$\begin{aligned} c(\mathbf{x}) &= \begin{cases} \mathbf{x}_j = \underline{\mathbf{b}}_j, & \mathbf{g}_j > 0, \ \ j \in 1, \dots, n, \\ \mathbf{x}_j = \overline{\mathbf{b}}_j, & \mathbf{g}_j < 0 \ \ j \in 1, \dots, n, \end{cases} \\ f(\mathbf{x}) &= \{ j \in 1, \dots, n \ \ j \notin c \}, \end{aligned}$$

are determined. The problem is then repartitioned as

$$\mathbf{x} \leftarrow \begin{bmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{bmatrix}, \ \mathbf{q} \leftarrow \begin{bmatrix} \mathbf{q}_f \\ \mathbf{q}_c \end{bmatrix}, \ \mathbf{H} \leftarrow \begin{bmatrix} \mathbf{H}_{ff} & \mathbf{H}_{fc} \\ \mathbf{H}_{cf} & \mathbf{H}_{cc} \end{bmatrix},$$

and the gradient in the unclamped subspace is

$$g_f = \nabla_{x_f} f = \mathbf{q}_f + \mathbf{H}_{ff} \mathbf{x}_f + \mathbf{H}_{fc} \mathbf{x}_c,$$

which leads to the Newton step

$$\Delta \mathbf{x}_f = -\mathbf{H}_{ff}^{-1} \mathbf{g}_f = -\mathbf{H}_{ff}^{-1} (\mathbf{q}_f + \mathbf{H}_{fc} \mathbf{x}_c) - \mathbf{x}_f.$$

Thus, the full step is given by

$$\Delta \mathbf{x} = \left[\hat{\mathbf{x}}(\alpha) = \| \, \mathbf{x} + \alpha \Delta \mathbf{x} \, \|_{\mathbf{b}} \right].$$

Performing line-search backwards, until the Armijo condition

$$\frac{f(\mathbf{x})-f(\hat{\mathbf{x}}(\alpha))}{\mathbf{g}^T(\mathbf{x}-\hat{\mathbf{x}}(\alpha))} > \gamma \tag{3.2}$$

is satisfied, leads to fast convergence. Note that $0 < \gamma < \frac{1}{2}$.
Summarizing the algorithm, it identifies the indices where the box-constrained will be violated and sets the feedback gain to zero in these dimensions. Thus, these dimensions are independent of the feedback gain, which makes it an open-loop case and therefore handling action constraints is possible.

3.1.6 Constrained Unscented Differential Dynamic Programming

In [26] the authors propose a new method, inspired by the Unscented Kalman Filter. In general, the paper addresses the problem of calculating second order derivatives, which can be difficult in many applications. Furthermore, they use the augmented Lagrangian to incorporate state constraints into the Dynamic Programming Framework. Replacing the calculus of the derivatives makes this extension possible.

3.2 Problem Formulation of Chance Constrained Iterative Linear Quadratic Gaussian

In this section, we first want to show the mathematical formulation of the problem we are considering for our approach. We are considering a Markov Decision Process

$$\mathcal{P}(\mathbf{s}_{\mathrm{t+1}}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{\Sigma}_t),$$

where the disturbances are zero mean i.i.d. Gaussian noise. Note that for the global optimization procedure no mathematical model of the dynamics are known. Furthermore, we are considering a finite horizon optimal control problem

$$J(\mathbf{s},\mathbf{A}) = \sum_{t=0}^{T-1} r_t(\mathbf{s}_t,\mathbf{a}_t) + r_T(\mathbf{s}_\mathbf{T}),$$

with reward function $r_t(\mathbf{s}_\mathrm{t},\mathbf{a}_t)$ and terminal reward $r_t(\mathbf{s}_\mathbf{T})$. Note, that the dynamics are unknown in our task. The Stochastic Optimal Control problem is formulated as

$$\max_{\mathbf{A}} \ \mathbb{E}[J(\mathbf{s},\mathbf{A})], \tag{3.3}$$

$$\text{s.t.} \ Pr(\mathbf{s}_{0:T} \in \mathcal{S}) \geq 1-\beta, \tag{3.4}$$

$$Pr(\mathbf{a}_{0:T-1} \in \mathcal{A}) \geq 1-\vartheta, \tag{3.5}$$

where we have Chance Constraints on states and actions such that they have to be in a feasible state space and action space with probability equal or higher to $1-\beta$ and $1-\vartheta$ respectively. Furthermore, we have probabilistic transition dynamics $\mathcal{P}(\mathbf{s}_{\mathrm{t+1}}|\mathbf{s}_t,\mathbf{a}_t)$ and a normal distributed initial state $\mu(\mathbf{s}_0)$. In our approach, we can not guarantee to satisfy the constraints. However, we will propose a method, which considers the constraints for the linearized dynamics gained by the backward pass and thus, lead to better performance. If the linearized dynamics are representative for the non-linear system in addition, satisfying the constraints from the original non-linear problem becomes more likely.

3.3 Algorithm Structure

Before presenting the mathematical structure in detail, we want to give a brief overview of the approach for better intuition. By this, we hope to clarify the steps we are doing for the derivation of our approach.
The algorithm first solves the unconstrained optimization problem by using iLQG. As we deal with a stochastic system with unknown dynamics, we first sample from the system to obtain state-action pairs $\tau(\mathbf{s},\mathbf{a})$ in the forward pass. By using Linear Regression, the dynamics are linearized around the trajectory at each time point leading to linear dynamics with zero mean Gaussian noise $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t,\mathbf{\Sigma}_t)$. Furthermore the state action pairs $\tau(\mathbf{s},\mathbf{a})$ and time-varying linearized systems are used for calculating the expected value of the perturbed Q-function $Q(\delta\mathbf{s},\delta\mathbf{a})$. Using the gained informations, we obtain a time-varying local feedback policy $\delta\mathbf{a} = \mathbf{K}_t(\delta\mathbf{s}) + \mathbf{k}_t$.

Until now, we performed the standard iLQG for stochastic systems. We now use the linearized dynamics and the local policies, to formulate an optimization problem which considers the constraints from problem (3.3). We further consider quadratic reward functions, which is no loss of generality, as we can approximate the expected reward by using second order Taylor Approximation. Inspired by [31] and [23] and linear Model Predictive Control in general, we stabilize the linearized system with the gained feedback gains from iLQG and use the forward term as degree of freedom to satisfy the constraints. By this, we prevent a high dimensional optimization problem and restrict to optimizing the forward gains only. This is an approximation and the feedback gains are not optimized subject to the Chance Constraints. However, if the feedback gains would be optimized in addition, this would lead to high dimensions in the decision variable [23]. Note that we use a constraint relaxation, which we will explain in detail. We further consider the constraints on the linearized dynamics to improve overall performance. Thus, we modify the forward terms $\mathbf{k}_t$ from the unconstrained case. We use the common augmented and stacked notation in linear MPC to obtain a Quadratic Program [14] [23].
The algorithm can be summarized as:

1. Perform initial forward pass, to obtain state action pairs $\tau(\mathbf{s}, \mathbf{a})$
2. Perform backward pass to obtain linearized dynamics $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{\Sigma}_t)$ and local optimal policies $\delta\mathbf{a}_t = \mathbf{K}_t\delta\mathbf{s}_t + \mathbf{k}_t$
3. Use linearized dynamics $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{\Sigma}_t)$ and optimal policies $\delta\mathbf{a}_t = \mathbf{K}_t\delta\mathbf{s}_t + \mathbf{k}_t$ to formulate a Quadratic Program subject to the Chance Constraints from problem (3.3)
4. Solve the Quadratic Problem with decision variable in $\mathbf{k}_t$
5. New local optimal policy: $\delta\mathbf{a}_t = \mathbf{K}_t\delta\mathbf{s}_t + \alpha_{cc}\mathbf{k}_t^{CC}$
6. Perform forward pass with new policy to obtain new state action pairs $\tau(\mathbf{s}, \mathbf{a})$
7. If not converged to a satisfying solution, start from 2.

By performing the Chance Constrained Optimization after the iLQG's backward pass, we want to improve the overall performance, by considering the action and state constraints, in order to stay in the validity region $\mathcal{S}$ and $\mathcal{A}$.

3.4 Mathematical Formulation

We focus now on the mathematical formulation of the Chance Constrained optimization. We will show, that we result in a Quadratic Program by tightening the Chance Constraints and using the stacked notation. This Quadratic Program is static and thus, can be solved efficiently by state of the art numerical solver. Remember that the information we gain by the backward pass of iLQG are essential for the Chance Constrained optimization problem. Thus, we have linearized dynamics and local optimal controller.
Note that we use throughout this section $\mathbf{M}$ for describing the weights of the states in the objective function. Furthermore we use $\mathbf{D}$ as the weights for the actions of the objective function.

3.4.1 Stacked Notation

We are considering disturbed linearized dynamics

$$\mathbf{s}_{t+1} = \mathbf{A}_t\mathbf{s}_t + \mathbf{B}_t\mathbf{a}_t + \mathbf{w}_t + \mathbf{c}_t,$$

resulting from the linearization with linear regression of the trajectories in the backward pass of iLQG. The global policy from the iLQG-backward pass is given as

$$\mathbf{a}_t = \mathbf{K}_t(\mathbf{s}_t - \mathbf{s}_{c,t}) + \mathbf{k}_t + \mathbf{a}_{c,t},$$

where $\mathbf{s}_{c,t}$ denotes the reference trajectory and $\mathbf{a}_{c,t}$ the action leading to $\mathbf{s}_{c,t}$. The closed-loop system is then given as

$$\begin{aligned}
\mathbf{s}_{t+1} &= \mathbf{A}_t\mathbf{s}_t + \mathbf{B}_t(\mathbf{K}_t(\mathbf{s}_t - \mathbf{s}_{c,t}) + \mathbf{k}_t + \mathbf{a}_{c,t}) + \mathbf{w}_t + \mathbf{c}_t,\\
\mathbf{s}_{t+1} &= \mathbf{A}_t\mathbf{s}_t + \mathbf{B}_t\mathbf{K}_t\mathbf{s}_t - \mathbf{B}_t\mathbf{K}_t\mathbf{s}_{c,t} + \mathbf{B}_t\mathbf{k}_t + \mathbf{B}_t\mathbf{a}_{c,t} + \mathbf{w}_t + \mathbf{c}_t,\\
\mathbf{s}_{t+1} &= (\mathbf{A}_t + \mathbf{B}_t\mathbf{K}_t)\mathbf{s}_t + \mathbf{B}_t\mathbf{k}_t + \mathbf{w}_t + \mathbf{c}_t - \mathbf{B}_t\mathbf{K}_t\mathbf{s}_{c,t} + \mathbf{B}_t\mathbf{a}_{c,t},\\
\mathbf{s}_{t+1} &= \hat{\mathbf{A}}_t\mathbf{s}_t + \mathbf{B}_t\mathbf{k}_t + \mathbf{w}_t + \mathbf{d}_t,
\end{aligned}$$

where we have replaced the constants with $\mathbf{d}_t$ and formulated the closed-loop system matrix as $\hat{\mathbf{A}}_t$. We can now formulate a recursive expression of the closed-loop dynamics by starting at time point zero as

$$\begin{aligned}
\mathbf{s}_1 &= \hat{\mathbf{A}}_0\mathbf{s}_0 + \mathbf{B}_0\mathbf{k}_0 + \mathbf{w}_0 + \mathbf{d}_0,\\
\mathbf{s}_2 &= \hat{\mathbf{A}}_1\mathbf{s}_1 + \mathbf{B}_1\mathbf{k}_1 + \mathbf{w}_1 + \mathbf{d}_1,\\
&= \hat{\mathbf{A}}_1\hat{\mathbf{A}}_0\mathbf{s}_0 + \hat{\mathbf{A}}_1\mathbf{B}_0\mathbf{k}_0 + \hat{\mathbf{A}}_1\mathbf{w}_0 + \hat{\mathbf{A}}_1\mathbf{d}_0 + \mathbf{B}_1\mathbf{k}_1 + \mathbf{w}_1 + \mathbf{d}_1,\\
\mathbf{s}_3 &= \hat{\mathbf{A}}_2\mathbf{s}_2 + \mathbf{B}_2\mathbf{k}_2 + \mathbf{w}_2 + \mathbf{d}_2,\\
&= \hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\hat{\mathbf{A}}_0\mathbf{s}_0 + \hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\mathbf{B}_0\mathbf{k}_0 + \hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\mathbf{w}_0 + \hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\mathbf{d}_0 + \hat{\mathbf{A}}_2\mathbf{B}_1\mathbf{k}_1 + \hat{\mathbf{A}}_2\mathbf{w}_1 + \hat{\mathbf{A}}_2\mathbf{d}_1 + \mathbf{w}_2 + \mathbf{d}_2,\\
\mathbf{s}_4 &= \hat{\mathbf{A}}_3\mathbf{s}_3 + \mathbf{B}_3\mathbf{k}_3 + \mathbf{w}_3 + \mathbf{d}_3,\\
&= \hat{\mathbf{A}}_3\hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\hat{\mathbf{A}}_0\mathbf{s}_0 + \hat{\mathbf{A}}_3\hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\mathbf{B}_0\mathbf{k}_0 + \hat{\mathbf{A}}_3\hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\mathbf{w}_0 + \hat{\mathbf{A}}_3\hat{\mathbf{A}}_2\hat{\mathbf{A}}_1\mathbf{d}_0 + \hat{\mathbf{A}}_3\hat{\mathbf{A}}_2\mathbf{B}_1\mathbf{k}_1 + \hat{\mathbf{A}}_3\hat{\mathbf{A}}_2\mathbf{w}_1 + \hat{\mathbf{A}}_3\hat{\mathbf{A}}_2\mathbf{d}_1 + \hat{\mathbf{A}}_3\mathbf{w}_2 + \hat{\mathbf{A}}_3\mathbf{d}_2 + \mathbf{w}_3 + \mathbf{d}_3,\\
&\vdots
\end{aligned}$$

We further introduce the augmented matrices similar to [23] as

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{I} \\ \hat{\mathbf{A}}_0 \\ \hat{\mathbf{A}}_1 \\ \hat{\mathbf{A}}_2 \\ \vdots \\ \hat{\mathbf{A}}_{T-1}\cdot\ldots\cdot\hat{\mathbf{A}}_0 \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{B}_0 & \mathbf{0} & \ldots & \mathbf{0} \\ \hat{\mathbf{A}}_1\mathbf{B}_0 & \mathbf{B}_1 & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{A}}_{T-1}\cdot\ldots\cdot\hat{\mathbf{A}}_1\mathbf{B}_0 & \hat{\mathbf{A}}_{T-1}\cdot\ldots\cdot\hat{\mathbf{A}}_2\mathbf{B}_1 & \ldots & \mathbf{B}_{T-1} \end{bmatrix}, \quad \tilde{\mathbf{G}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \ldots & \mathbf{0} \\ \hat{\mathbf{A}}_1 & \mathbf{I} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{A}}_{T-1}\cdot\ldots\cdot\hat{\mathbf{A}}_1 & \hat{\mathbf{A}}_{T-1}\cdot\ldots\cdot\hat{\mathbf{A}}_2 & \ldots & \mathbf{I} \end{bmatrix}.$$

Thus, the closed loop system can be rewritten as

$$\tilde{\mathbf{s}} = \tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}},$$

where

$$\tilde{\mathbf{s}} = \begin{bmatrix} \mathbf{s}_0 \\ \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_T \end{bmatrix}, \quad \tilde{\mathbf{k}} = \begin{bmatrix} \mathbf{k}_0 \\ \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_{T-1}, \end{bmatrix} \quad \tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_{T-1} \end{bmatrix}, \quad \tilde{\mathbf{d}} = \begin{bmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_{T-1} \end{bmatrix}.$$

At this point, it makes sense to introduce the weighting matrices for the objective similar to [23], as they will be useful for the derivation of the objective later

$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{M}_0 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_1 & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \ldots & \mathbf{M}_T \end{bmatrix}, \quad \tilde{\mathbf{M}}_{mod} = \begin{bmatrix} \mathbf{M}_0 + \mathbf{K}_0^T\mathbf{D}_0\mathbf{K}_0 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_1 + \mathbf{K}_1^T\mathbf{D}_1\mathbf{K}_1 & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \vdots & \mathbf{M}_{T-1} + \mathbf{K}_{T-1}^T\mathbf{D}_{T-1}\mathbf{K}_{T-1} & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \mathbf{M}_T \end{bmatrix},$$

$$\tilde{\mathbf{D}} = \begin{bmatrix} \mathbf{D}_0 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_1 & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \ldots & \mathbf{D}_{T-1} \end{bmatrix}, \quad \tilde{\mathbf{K}} = \begin{bmatrix} \mathbf{K}_0 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_1 & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{K}_{T-1} \end{bmatrix}.$$

Using these augmented matrices makes the derivation and formulation for a direct method easy.

3.4.2 System Properties

We will briefly show the system properties as they will be important for the reformulation as Quadratic Program and derivation of the Chance Constraints. We use the stacked notation for our system

$$\tilde{\mathbf{s}} = \tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}}.$$

We first consider the second-order moment $\mathbb{E}\left[\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\right]$ of our system

$$\begin{aligned}
\mathbb{E}\left[\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\right] &= \mathbb{E}\left[(\tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}})(\tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}})^T\right], \\
&= \mathbb{E}[\tilde{\mathbf{A}}\mathbf{s}_0\mathbf{s}_0^T\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{B}}\tilde{\mathbf{k}}\mathbf{s}_0^T\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{w}}\mathbf{s}_0^T\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\mathbf{s}_0^T\tilde{\mathbf{A}}^T + \tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{w}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + \ldots \\
&\qquad \ldots\tilde{\mathbf{G}}\tilde{\mathbf{w}}\tilde{\mathbf{w}}^T\tilde{\mathbf{G}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{w}}^T\tilde{\mathbf{G}}^T + \tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T], \\
&= \tilde{\mathbf{A}}\mathbb{E}\left[\mathbf{s}_0\mathbf{s}_0^T\right]\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{B}}\tilde{\mathbf{k}}\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + \tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + \tilde{\mathbf{G}}\mathbb{E}\left[\tilde{\mathbf{w}}\tilde{\mathbf{w}}^T\right]\tilde{\mathbf{G}}^T + \tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T, \\
&= \tilde{\mathbf{A}}\hat{\mathbf{s}}_0\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + \tilde{\mathbf{A}}\Sigma_{\mathbf{s}_0}\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{B}}\tilde{\mathbf{k}}\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + \tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + \tilde{\mathbf{G}}\Sigma_{\tilde{\mathbf{w}}}\tilde{\mathbf{G}}^T + \tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T.
\end{aligned}$$

We will further need the expression

$$\begin{aligned}
\mathbb{E}[\tilde{\mathbf{s}}]\,\mathbb{E}[\tilde{\mathbf{s}}]^T &= (\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}})(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}})^T, \\
&= \tilde{\mathbf{A}}\hat{\mathbf{s}}_0\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{A}}\hat{\mathbf{s}}_0\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + 2\tilde{\mathbf{A}}\hat{\mathbf{s}}_0\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T + \tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + 2\tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T + \tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T,
\end{aligned} \tag{3.6}$$

to formulate the covariance of our system as

$$\begin{aligned}
\mathbb{E}\left[(\tilde{\mathbf{s}} - \mathbb{E}[\tilde{\mathbf{s}}])(\tilde{\mathbf{s}} - \mathbb{E}[\tilde{\mathbf{s}}])^T\right] &= \mathbb{E}\left[\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T - 2\mathbb{E}[\tilde{\mathbf{s}}]\,\tilde{\mathbf{s}}^T + \mathbb{E}[\tilde{\mathbf{s}}]\,\mathbb{E}[\tilde{\mathbf{s}}]^T\right], \\
&= \mathbb{E}\left[\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\right] - \mathbb{E}[\tilde{\mathbf{s}}]\,\mathbb{E}\left[\tilde{\mathbf{s}}^T\right], \\
\tilde{\Sigma}_{\tilde{\mathbf{s}}} &= \tilde{\mathbf{A}}\Sigma_{\mathbf{s}_0}\tilde{\mathbf{A}}^T + \tilde{\mathbf{G}}\tilde{\Sigma}_{\tilde{\mathbf{w}}}\tilde{\mathbf{G}}^T.
\end{aligned}$$

Thus, the covariance of our system is given as the difference of the second order moment and the expression derived in (3.6). In general, our policy is deterministic. But since we have a feedback gain and therefore information of the state, we also have a covariance of our policy as

$$\begin{aligned}
\mathbb{E}\left[(\tilde{\mathbf{a}} - \mathbb{E}[\tilde{\mathbf{a}}])(\tilde{\mathbf{a}} - \mathbb{E}[\tilde{\mathbf{a}}])^T\right] &= \mathbb{E}\left[(\tilde{\mathbf{K}}\tilde{\mathbf{s}} - \tilde{\mathbf{K}}\tilde{\mathbf{s}}_c + \tilde{\mathbf{a}}_c + \tilde{\mathbf{k}} - \tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}] + \tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - \tilde{\mathbf{a}}_c - \tilde{\mathbf{k}})(\tilde{\mathbf{K}}\tilde{\mathbf{s}} - \tilde{\mathbf{K}}\tilde{\mathbf{s}}_c + \tilde{\mathbf{a}}_c + \tilde{\mathbf{k}} - \tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}] + \tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - \tilde{\mathbf{a}}_c - \tilde{\mathbf{k}})^T\right], \\
&= \mathbb{E}\left[(\tilde{\mathbf{K}}\tilde{\mathbf{s}} - \tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}])(\tilde{\mathbf{K}}\tilde{\mathbf{s}} - \tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}])^T\right], \\
&= \mathbb{E}\left[\tilde{\mathbf{K}}\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\tilde{\mathbf{K}}^T - 2\tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}]\,\tilde{\mathbf{s}}^T\tilde{\mathbf{K}}^T + \tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}]\,\mathbb{E}[\tilde{\mathbf{s}}]^T\tilde{\mathbf{K}}^T\right], \\
&= \mathbb{E}\left[\tilde{\mathbf{K}}\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\tilde{\mathbf{K}}^T\right] - \tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}]\,\mathbb{E}[\tilde{\mathbf{s}}]^T\tilde{\mathbf{K}}^T, \\
&= \tilde{\mathbf{K}}\mathbb{E}\left[\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\right]\tilde{\mathbf{K}}^T - \tilde{\mathbf{K}}\mathbb{E}[\tilde{\mathbf{s}}]\,\mathbb{E}\left[\tilde{\mathbf{s}}^T\right]\tilde{\mathbf{K}}^T, \\
&= \tilde{\mathbf{K}}\left(\mathbb{E}\left[\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\right] - \mathbb{E}[\tilde{\mathbf{s}}]\,\mathbb{E}\left[\tilde{\mathbf{s}}^T\right]\right)\tilde{\mathbf{K}}^T, \\
&= \tilde{\mathbf{K}}\left(\tilde{\mathbf{A}}\Sigma_{\mathbf{s}_0}\tilde{\mathbf{A}}^T + \tilde{\mathbf{G}}\tilde{\Sigma}_{\tilde{\mathbf{w}}}\tilde{\mathbf{G}}^T\right)\tilde{\mathbf{K}}^T, \\
&= \tilde{\mathbf{K}}\tilde{\mathbf{A}}\Sigma_{\mathbf{s}_0}\tilde{\mathbf{A}}^T\tilde{\mathbf{K}}^T + \tilde{\mathbf{K}}\tilde{\mathbf{G}}\tilde{\Sigma}_{\tilde{\mathbf{w}}}\tilde{\mathbf{G}}^T\tilde{\mathbf{K}}^T.
\end{aligned}$$

Thus, the covariance of our policy is a transformation of the covariance of our system by the feedback gain $\tilde{\mathbf{K}}$.

3.4.3 Formulation as Quadratic Program

In this section, we first want to formulate the objective of the Chance Constrained optimization problem and then derive the Quadratic Program representation of it similar to [23].
We consider the quadratic reward objective

$$\begin{aligned}
J(\mathbf{s}, \mathbf{a}) &= -\mathbb{E}\left[\sum_{t=0}^{T-1}(\mathbf{s}_t - \mathbf{s}_{g,t})^T\mathbf{M}_t(\mathbf{s}_t - \mathbf{s}_{g,t}) + \mathbf{a}_t^T\mathbf{D}_t\mathbf{a}_t + (\mathbf{s}_T - \mathbf{s}_{g,T})^T\mathbf{M}_T(\mathbf{s}_T - \mathbf{s}_{g,T})\right], \\
&= -\mathbb{E}\left[\sum_{t=0}^{T-1}\mathbf{s}_t^T\mathbf{M}_t\mathbf{s}_t - 2\mathbf{s}_{g,t}^T\mathbf{M}_t\mathbf{s}_t + \mathbf{s}_{g,t}^T\mathbf{M}_t\mathbf{s}_{g,t} + \mathbf{a}_t^T\mathbf{D}_t\mathbf{a}_t + \mathbf{s}_T^T\mathbf{M}_T\mathbf{s}_T - 2\mathbf{s}_{g,T}^T\mathbf{M}_T\mathbf{s}_T + \mathbf{s}_{g,T}^T\mathbf{M}_T\mathbf{s}_{g,T}\right],
\end{aligned}$$

where $s_{g,t}$ denotes the goal state at time t. As we already know the parametrization of our policy, we can insert it into the objective:

$$
\begin{aligned}
J(\mathbf{s},\mathbf{a}) = -\mathbb{E}[&\mathbf{s}_T^T\mathbf{M}_T\mathbf{s}_T - 2\mathbf{s}_{g,T}^T\mathbf{M}_T\mathbf{s}_T + \mathbf{s}_{g,T}^T\mathbf{M}_T\mathbf{s}_{g,T} + \sum_{t=0}^{T-1}\mathbf{s}_t^T\mathbf{M}_t\mathbf{s}_t - 2\mathbf{s}_{g,t}^T\mathbf{M}_t\mathbf{s}_t + \mathbf{s}_{g,t}^T\mathbf{M}_t\mathbf{s}_{g,t} + \ldots\\
&\ldots(\mathbf{K}_t\mathbf{s}_t - \mathbf{K}_t\mathbf{s}_{c,t} + \mathbf{k}_t + \mathbf{a}_{c,t})^T\mathbf{D}_t(\mathbf{K}_t\mathbf{s}_t - \mathbf{K}_t\mathbf{s}_{c,t} + \mathbf{k}_t + \mathbf{a}_{c,t})],\\
= -\mathbb{E}[&\mathbf{s}_T^T\mathbf{M}_T\mathbf{s}_T - 2\mathbf{s}_{g,T}^T\mathbf{M}_T\mathbf{s}_T + \mathbf{s}_{g,T}^T\mathbf{M}_T\mathbf{s}_{g,T} + \sum_{t=0}^{T-1}\mathbf{s}_t^T\mathbf{M}_t\mathbf{s}_t - 2\mathbf{s}_{g,t}^T\mathbf{M}_t\mathbf{s}_t + \mathbf{s}_{g,t}^T\mathbf{M}_t\mathbf{s}_{g,t} + \ldots\\
&\ldots\mathbf{s}_t^T\mathbf{K}_t^T\mathbf{D}_t\mathbf{K}_t\mathbf{s}_t - 2\mathbf{s}_{c,t}^T\mathbf{K}_t^T\mathbf{D}_t\mathbf{K}_t\mathbf{s}_t + 2\mathbf{a}_{c,t}^T\mathbf{D}_t\mathbf{K}_t\mathbf{s}_t + 2\mathbf{k}_t^T\mathbf{D}_t\mathbf{K}_t\mathbf{s}_t + \mathbf{s}_{c,t}^T\mathbf{K}_t^T\mathbf{D}_t\mathbf{K}_t\mathbf{s}_{c,t} - 2\mathbf{a}_{c,t}^T\mathbf{D}_t\mathbf{K}_t\mathbf{s}_{c,t} - 2\mathbf{k}_t^T\mathbf{D}_t\mathbf{K}_t\mathbf{s}_{c,k} + \ldots\\
&\ldots\mathbf{a}_{c,t}^T\mathbf{D}_t\mathbf{a}_{c,t} + 2\mathbf{k}_t^T\mathbf{D}_t\mathbf{a}_{c,k} + \mathbf{k}_t^T\mathbf{D}_t\mathbf{k}_t].
\end{aligned}
$$

We can now make usage of the stacked notation introduced in section 3.4.1 and rewrite the objective without the sum

$$
\begin{aligned}
J(\mathbf{s},\mathbf{a}) &= -\mathbb{E}[\tilde{\mathbf{s}}^T\tilde{\mathbf{M}}\tilde{\mathbf{s}} - 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}}_g + \tilde{\mathbf{s}}^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}} - 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}} + 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}} + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - \ldots\\
&\quad\ldots 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c + \tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c + \tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{k}}],\\
&= -\mathbb{E}[\tilde{\mathbf{s}}^T\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{s}} - 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}}_g - 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}} + 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}} + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - \ldots\\
&\quad\ldots 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c + \tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c + \tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{k}}]\\
-J(\mathbf{s},\mathbf{a}) &= \mathbb{E}[\tilde{\mathbf{s}}^T\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{s}}] - \mathbb{E}[2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}}] + \mathbb{E}[\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}}_g] - \mathbb{E}[2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}] + \mathbb{E}[2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}] + \mathbb{E}[2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}] + \ldots \qquad (3.7)\\
&\quad\ldots\mathbb{E}[\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c] - \mathbb{E}[2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c] - \mathbb{E}[2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c] + \mathbb{E}[\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c] + \mathbb{E}[2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c] + \mathbb{E}[\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{k}}].
\end{aligned}
$$

We can now evaluate the expected value of our objective. Since we are quadratic in the states and have Gaussian noise, we can formulate the expectation of the objective in closed form. For this purpose we insert the dynamics equation into our objective and consider summands of the stacked objective (3.7). The first summand is

$$
\begin{aligned}
\mathbb{E}[\tilde{\mathbf{s}}^T\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{s}}] = \mathbb{E}[tr(\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T)] &= \mathbb{E}[tr(\tilde{\mathbf{M}}_{mod}(\tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}})(\tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}})^T)],\\
&= \mathbb{E}[tr(\tilde{\mathbf{M}}_{mod}(\tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}})(\mathbf{s}_0^T\tilde{\mathbf{A}}^T + \tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + \tilde{\mathbf{w}}^T\tilde{\mathbf{G}}^T + \tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T))],\\
&= \mathbb{E}[tr(\tilde{\mathbf{M}}_{mod}(\tilde{\mathbf{A}}\mathbf{s}_0\mathbf{s}_0^T\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{B}}\tilde{\mathbf{k}}\mathbf{s}_0\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{w}}\mathbf{s}_0\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\mathbf{s}_0^T\tilde{\mathbf{A}}^T \ldots +\\
&\qquad\qquad \ldots\tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}} + 2\tilde{\mathbf{G}}\tilde{\mathbf{w}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}} + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + \tilde{\mathbf{G}}\tilde{\mathbf{w}}\tilde{\mathbf{w}}^T\tilde{\mathbf{G}}^T + 2\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{w}}^T\tilde{\mathbf{G}}^T + \ldots\\
&\qquad\qquad \ldots\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T))],\\
&= tr(\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{A}}\mathbf{S}_0\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{M}}_{\mathrm{mod}}\tilde{\mathbf{B}}\tilde{\mathbf{k}}\hat{\mathbf{s}}_0\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\tilde{\mathbf{d}}\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + \tilde{\mathbf{M}}_{mod}\tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}} + \ldots\\
&\qquad\qquad \ldots 2\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + \tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\tilde{\boldsymbol{\Sigma}}_{\tilde{\mathbf{w}}}\tilde{\mathbf{G}}^T + \tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T),
\end{aligned}
$$

where $\mathbf{S}_0$ denotes the second order moment of the random variable $\mathbf{s}_0$ and $\tilde{\boldsymbol{\Sigma}}_{\tilde{\mathbf{w}}}$ is the stacked covariance matrix of $\tilde{\mathbf{w}}$. In the last step we calculated the expected value of the equation and made use of the zero mean random variable $\tilde{\mathbf{w}}$. We have dealt with the quadratic part of the objective. The random variable $\tilde{\mathbf{s}}$ occurs only linear in the rest of our objective. Thus, we have to consider

$$
\begin{aligned}
\mathbb{E}[\tilde{\mathbf{s}}] &= \mathbb{E}[\tilde{\mathbf{A}}\mathbf{s}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{w}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}}]\\
&= \tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}}.
\end{aligned}
$$

We can now use this information to evaluate the expectation depending on the random variable $\tilde{\mathbf{s}}$ of the remaining summands

$$
\begin{aligned}
\mathbb{E}[2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}}] &= 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} + 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{G}}\tilde{\mathbf{d}},\\
\mathbb{E}[2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}] &= 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} + 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{G}}\tilde{\mathbf{d}},\\
\mathbb{E}[2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}] &= 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} + 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{G}}\tilde{\mathbf{d}},\\
\mathbb{E}[2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}] &= 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{G}}\tilde{\mathbf{d}}.
\end{aligned}
$$

This gives us the resulting objective

$$
\begin{aligned}
-J(\tilde{\mathbf{k}}) = tr(&\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{A}}\mathbf{S}_0\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{B}}\tilde{\mathbf{k}}\hat{\mathbf{s}}_0\tilde{\mathbf{A}}^T + 2\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\tilde{\mathbf{d}}\hat{\mathbf{s}}_0^T\tilde{\mathbf{A}}^T + \tilde{\mathbf{M}}_{mod}\tilde{\mathbf{B}}\tilde{\mathbf{k}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}} + \dots \\
&\dots 2\tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{k}}^T\tilde{\mathbf{B}}^T + \tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\mathbf{W}\tilde{\mathbf{G}}^T + \tilde{\mathbf{M}}_{mod}\tilde{\mathbf{G}}\tilde{\mathbf{d}}\tilde{\mathbf{d}}^T\tilde{\mathbf{G}}^T) - 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 - 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} - 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{G}}\tilde{\mathbf{d}} + 2\tilde{\mathbf{s}}_g^T\tilde{\mathbf{M}}\tilde{\mathbf{s}}_g - \dots \\
&\dots 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 - 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} - 2\tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{G}}\tilde{\mathbf{d}} + 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} + 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{G}}\tilde{\mathbf{d}} + \dots \\
&\dots 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{B}}\tilde{\mathbf{k}} + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{G}}\tilde{\mathbf{d}} + \tilde{\mathbf{s}}_c^T\tilde{\mathbf{K}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - 2\tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c - 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c + \tilde{\mathbf{a}}_c^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c + 2\tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{a}}_c + \tilde{\mathbf{k}}^T\tilde{\mathbf{D}}\tilde{\mathbf{k}}.
\end{aligned}
$$

To prevent confusions, we define the objective as

$$J_{\text{reward}}(\tilde{\mathbf{k}}) = -J(\tilde{\mathbf{k}}) \tag{3.8}$$

We further want to introduce Chance Constraints on the states and actions with the stacked notation. As we consider linear dynamics and zero mean Gaussian noise, we will use Boole's Inequality to relax our constraints as described in section 2.4.2.
We first consider **state constraints**. We choose the the dimensions of our stacked states vector $\tilde{\mathbf{s}}$ which underly a constraint by an appropriate matrix $\tilde{\mathbf{F}}$. Let $\tilde{\mathbf{s}}$ have the dimension $\mathbb{R}^{T\cdot n}$, n denoting the state's dimension, then $\tilde{\mathbf{F}}$ has the dimension $\mathbb{R}^{mxTn}$, where m denotes the number of constraints on the states. We introduce the upper bound $\tilde{\mathbf{b}}_u$ of dimension $\mathbb{R}^m$ with the entries for the upper bound for each dimension. Thus, the **upper bound Chance Constraint** can be formulated as

$$Pr(\tilde{\mathbf{F}}_u\tilde{\mathbf{s}} \ \leq \ \tilde{\mathbf{b}}_u) \geq \mathbf{1} - \boldsymbol{\beta}_u. \tag{3.9}$$

Note, that this expression equals m inequality constraints. The probability vector $\boldsymbol{\beta}_u$ is of dimension $\mathbb{R}^m$ with the probability entries β_i for constraint at dimension i. Thus, we already have used Boole's inequality, since in inequality (3.9) we are considering m univariate single probabilities. Remember, that the sum $\sum_{i=0}^m \beta_i \leq 0.5$ of each single probability of dimension i has to be fulfilled. We can now use the cdf of a Gaussian to rewrite the constraint (see 2.4.2) as

$$\tilde{\mathbf{b}}_u - \tilde{\mathbf{F}}_u \cdot \mathbb{E}[\tilde{\mathbf{s}}] - \sqrt{2\tilde{\mathbf{F}}_u\tilde{\Sigma}_{\tilde{\mathbf{s}}}\tilde{\mathbf{F}}^T} \odot erf^{-1}(\mathbf{1} - 2\boldsymbol{\beta}_u) \geq \mathbf{0}, \tag{3.10}$$

where $\odot$ is the operator for pointwise multiplication. In section 2.4.2 we already have mentioned, that inequality (3.10) is linear in the decision variable, namely $\tilde{\mathbf{k}}$. By plugging in the expectation of $\tilde{\mathbf{s}}$, this gets clear

$$\tilde{\mathbf{b}}_u - \tilde{\mathbf{F}}_u \cdot (\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}}) - \sqrt{2\tilde{\mathbf{F}}_u\tilde{\Sigma}_{\tilde{\mathbf{s}}}\tilde{\mathbf{F}}_u^T} \odot erf^{-1}(\mathbf{1} - 2\boldsymbol{\beta}_u) \geq \mathbf{0}.$$

The **lower bound probability** is defined with small differences compared to the upper bound probability as

$$Pr(\tilde{\mathbf{F}}_l\tilde{\mathbf{s}} \ \geq \ \tilde{\mathbf{b}}_l) > \mathbf{1} - \boldsymbol{\beta}_l.$$

This form is not applicable to the cdf. Therefore we have to reformulate it as

$$
\begin{aligned}
\mathbf{1} - Pr(\tilde{\mathbf{F}}_l\tilde{\mathbf{s}} \ \leq \ \tilde{\mathbf{b}}_l) &\geq \mathbf{1} - \boldsymbol{\beta}_l \\
Pr(\tilde{\mathbf{F}}_l\tilde{\mathbf{s}} \ \leq \ \tilde{\mathbf{b}}_l) &\leq \boldsymbol{\beta}_l.
\end{aligned}
$$

By using the well-known reformulation, we end up in the linear constraint

$$-\tilde{\mathbf{b}}_l + \tilde{\mathbf{F}}_l(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}}) + \sqrt{2\tilde{\mathbf{F}}_l\tilde{\Sigma}_{\tilde{\mathbf{s}}}\tilde{\mathbf{F}}_l^T} \odot erf^{-1}(2\boldsymbol{\beta}_l - \mathbf{1}) \geq \mathbf{0}.$$

All requirements regarding $\boldsymbol{\vartheta}_l$ mentioned earlier have to be given.
Since our policy is depending on the states vector $\tilde{\mathbf{s}}$, we have to consider probabilities when constraining the inputs. The procedure is the same as for state constraints. For the **upper bound** on our actions we have

$$Pr(\tilde{\mathbf{H}}_u\tilde{\mathbf{a}} \ \leq \ \tilde{\mathbf{z}}_u) \geq \mathbf{1} - \boldsymbol{\vartheta}_u,$$

where $\tilde{\mathbf{H}}_u$ has the same role as $\tilde{\mathbf{F}}_u$ for the action constraints and $\tilde{\mathbf{z}}_u$ is the upper value vector for the maximum action value at each constraint i. The probability vector $\boldsymbol{\beta}_u$ contains the probability value $\beta_{i,u}$ for constraint i and underlies the discussed properties. Applying the procedure, we obtain for the upper bound of the policy

$$\tilde{\mathbf{z}}_u - \tilde{\mathbf{H}}_u(\tilde{\mathbf{K}}(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{k}} + \tilde{\mathbf{G}}\tilde{\mathbf{d}}) - \tilde{\mathbf{K}}\tilde{\mathbf{s}}_c + \tilde{\mathbf{a}}_c + \tilde{\mathbf{k}}) - \sqrt{2\tilde{\mathbf{H}}_u\tilde{\Sigma}_{\tilde{\mathbf{a}}}\tilde{\mathbf{H}}_u^T} \odot erf^{-1}(\mathbf{1} - 2\boldsymbol{\vartheta}_u) \geq \mathbf{0}.$$

For the lower bound of our policy we consider the probability

$$Pr(\tilde{\mathbf{H}}_l\tilde{\mathbf{a}} \;\geq\; \tilde{\mathbf{z}}_l)\geq 1-\boldsymbol{\vartheta}_l,$$

which leads to the inequality constraints

$$-\tilde{\mathbf{z}}_l+\tilde{\mathbf{H}}_l(\tilde{\mathbf{K}}(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0+\tilde{\mathbf{B}}\tilde{\mathbf{k}}+\tilde{\mathbf{G}}\tilde{\mathbf{d}})-\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c+\tilde{\mathbf{a}}_c+\tilde{\mathbf{k}})+\sqrt{2\tilde{\mathbf{H}}_l\tilde{\boldsymbol{\Sigma}}_{\tilde{\mathbf{a}}}\tilde{\mathbf{H}}_l^T}\odot erf^{-1}(2\boldsymbol{\vartheta}_l-1)\geq\mathbf{0}.$$

We can now write the entire Quadratic Program as

$$\max_{\tilde{\mathbf{k}}}\; J_{\text{reward}}(\tilde{\mathbf{k}}) \tag{3.11}$$

$$s.t.\;\; \tilde{\mathbf{b}}_u-\tilde{\mathbf{F}}_u\cdot(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0+\tilde{\mathbf{B}}\tilde{\mathbf{k}}+\tilde{\mathbf{G}}\tilde{\mathbf{d}})-\sqrt{2\tilde{\mathbf{F}}_u\tilde{\boldsymbol{\Sigma}}_{\tilde{\mathbf{s}}}\tilde{\mathbf{F}}_u^T}\odot erf^{-1}(\mathbf{1}-2\boldsymbol{\beta}_u)\geq\mathbf{0} \tag{3.12}$$

$$-\tilde{\mathbf{b}}_l+\tilde{\mathbf{F}}_l(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0+\tilde{\mathbf{B}}\tilde{\mathbf{k}}+\tilde{\mathbf{G}}\tilde{\mathbf{d}})+\sqrt{2\tilde{\mathbf{F}}_l\tilde{\boldsymbol{\Sigma}}_{\tilde{\mathbf{s}}}\tilde{\mathbf{F}}_l^T}\odot erf^{-1}(2\boldsymbol{\beta}_l-\mathbf{1})\geq\mathbf{0} \tag{3.13}$$

$$\tilde{\mathbf{z}}_u-\tilde{\mathbf{H}}_u(\tilde{\mathbf{K}}(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0+\tilde{\mathbf{B}}\tilde{\mathbf{k}}+\tilde{\mathbf{G}}\tilde{\mathbf{d}})-\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c+\tilde{\mathbf{a}}_c+\tilde{\mathbf{k}})-\sqrt{2\tilde{\mathbf{H}}_u\tilde{\boldsymbol{\Sigma}}_{\tilde{\mathbf{a}}}\tilde{\mathbf{H}}_u^T}\odot erf^{-1}(\mathbf{1}-2\boldsymbol{\vartheta}_u)\geq\mathbf{0} \tag{3.14}$$

$$-\tilde{\mathbf{z}}_l+\tilde{\mathbf{H}}_l(\tilde{\mathbf{K}}(\tilde{\mathbf{A}}\hat{\mathbf{s}}_0+\tilde{\mathbf{B}}\tilde{\mathbf{k}}+\tilde{\mathbf{G}}\tilde{\mathbf{d}})-\tilde{\mathbf{K}}\tilde{\mathbf{s}}_c+\tilde{\mathbf{a}}_c+\tilde{\mathbf{k}})+\sqrt{2\tilde{\mathbf{H}}_l\tilde{\boldsymbol{\Sigma}}_{\tilde{\mathbf{a}}}\tilde{\mathbf{H}}_l^T}\odot erf^{-1}(2\boldsymbol{\vartheta}_l-\mathbf{1})\geq\mathbf{0}, \tag{3.15}$$

where $\tilde{\mathbf{b}}_u\in\mathbb{R}^{m_u}$ contains the upper state bound $b_{i,u}$ for dimension i, $\tilde{\mathbf{b}}_l\in\mathbb{R}^{m_l}$ contains the lower state bound $b_{j,l}$ for dimension j, $\tilde{\mathbf{z}}_u\in\mathbb{R}^{k_u}$ contains the action upper bound $z_{r,u}$ for dimension r and $\tilde{\mathbf{z}}_l\in\mathbb{R}^{k_l}$ contains the action lower bound $z_{k,l}$ for dimension k.
As the reward function is quadratic in $\tilde{\mathbf{k}}$ and we have convex, linear inequalities, this optimization problem is a Quadratic Problem and therefore convex. Thus, we have reformulated the dynamic optimization problem as a static optimization problem by parameterizing the policy and plugging in the dynamics into the objective. This is an example of a direct method. We have parameterized our action, eliminated the dynamics and obtained a Quadratic Program [32].
We use CasADi [33] to solve the optimization problem.

3.5 Linearization

We use Linear Regression to linearize around the trajectories. In this chapter we want to explain our data structure and derive the linearization. The derivation is based on [34].
First we will provide the general structure of our linearized dynamics by applying first order Taylor expansion. Generally, for non-linear dynamics $f(\mathbf{s},\mathbf{a})$, we linearize the system around a state action pair $(\mathbf{s}_t,\mathbf{a}_t)$ at time point t. This results in

$$f(\mathbf{s},\mathbf{a})\approx f(\mathbf{s}_t,\mathbf{a}_t)+\left.\frac{\partial f(\mathbf{s},\mathbf{a})}{\partial\mathbf{s}}\right|_{\mathbf{s}_t}\Delta\mathbf{s}+\left.\frac{\partial f(\mathbf{s},\mathbf{a})}{\partial\mathbf{a}}\right|_{\mathbf{a}_t}\Delta\mathbf{a}.$$

By replacing $\left.\frac{\partial f(\mathbf{s},\mathbf{a})}{\partial\mathbf{s}}\right|_{\mathbf{s}_t}=\mathbf{A}_t$ and $\left.\frac{\partial f(\mathbf{s},\mathbf{a})}{\partial\mathbf{a}}\right|_{\mathbf{a}_t}=\mathbf{B}_t$, the equation turns to

$$f(\mathbf{s},\mathbf{a})\approx f(\mathbf{s}_t,\mathbf{a}_t)+\mathbf{A}_t\Delta\mathbf{s}+\mathbf{B}_t\Delta\mathbf{a}.$$

Plugging in the relation $\Delta\mathbf{s}=\mathbf{s}-\mathbf{s}_t$ and $\Delta\mathbf{a}=\mathbf{a}-\mathbf{a}_t$

$$\begin{aligned}f(\mathbf{s},\mathbf{a})&\approx f(\mathbf{s}_t,\mathbf{a}_t)-\mathbf{A}_t\mathbf{s}_t-\mathbf{B}_t\mathbf{a}_t+\mathbf{A}_t\mathbf{s}+\mathbf{B}_t\mathbf{a}\\&\approx\mathbf{c}_t+\mathbf{A}_t\mathbf{s}+\mathbf{B}_t\mathbf{a},\end{aligned}$$

where we have replaced $\mathbf{c}_t=f(\mathbf{s}_t,\mathbf{a}_t)-\mathbf{A}_t\mathbf{s}_t-\mathbf{B}_t\mathbf{a}_t$.
As the dynamics of our system is unknown and therefore the derivatives $\mathbf{A}_t,\mathbf{B}_t$ and the constant $f(\mathbf{s}_t,\mathbf{a}_t)$ not given, we define the parameters $\mathbf{A}_t,\mathbf{B}_t,\mathbf{c}_t$ by using Linear Regression. Therefore, we first rewrite our linearized dynamic equation in matrix notation as

$$f(\mathbf{s},\mathbf{a})=\begin{bmatrix}\mathbf{A}_t & \mathbf{B}_t & \mathbf{c}_t\end{bmatrix}\begin{bmatrix}\mathbf{s}\\\mathbf{a}\\\mathbf{1}\end{bmatrix}.$$

In the forward pass of iLQG, we shoot k trajectories, leading to k state action pairs for every time step t. Thus we arrange our matrices $\boldsymbol{\theta}$ and $\mathbf{w}$ as

$$\mathbf{w} = \begin{bmatrix} \mathbf{A}_t & \mathbf{B}_t & \mathbf{c}_t \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} s_{1,1} & s_{2,1} & \dots & s_{k,1} \\ s_{1,2} & s_{2,2} & \dots & s_{k,2} \\ \vdots & \vdots & \dots & \vdots \\ s_{1,n} & s_{2,n} & \dots & s_{k,n} \\ a_{1,1} & a_{2,1} & \dots & a_{k,1} \\ \vdots & \vdots & \dots & \vdots \\ a_{1,j} & a_{2,j} & \dots & a_{k,j} \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \dots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}.$$

Note that we have a system of state dimension n and action dimension j and that $\boldsymbol{\theta}$ is representative for time point t of our trajectories. For example the point $s_{1,2}$ for the state means the state's second dimension of the first trajectory at time point t.
We further have to determine the target matrix $\mathbf{y}$. Since we want to predict the next time step, $\mathbf{y}$ is composed of the appropriate state values for the next time step $t+1$ as

$$\mathbf{y} = \begin{bmatrix} s'_{1,1} & s'_{2,1} & \dots & s'_{k,1} \\ s'_{1,2} & s'_{2,2} & \dots & s'_{k,2} \\ \vdots & \vdots & \dots & \vdots \\ s'_{1,n} & s'_{k,2} & \dots & s'_{k,n} \end{bmatrix},$$

where s' denotes the state at next time step $t+1$.
We can now rewrite our Linear Regression problem as

$$\mathbf{y} = \mathbf{w}\boldsymbol{\theta}.$$

The objective in linear regression is to minimize the quadratized error $(\mathbf{y}-\mathbf{w}\boldsymbol{\theta})(\mathbf{y}-\mathbf{w}\boldsymbol{\theta})^T$ [34]. We take the derivative with respect to $\mathbf{w}$, use the common procedure to obtain the weights $\mathbf{w}$

$$\begin{aligned} \frac{\partial(\mathbf{y}-\mathbf{w}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{w}\boldsymbol{\theta})}{\partial\mathbf{w}} &= \frac{\partial(\mathbf{y}^T\mathbf{y}-2\boldsymbol{\theta}^T\mathbf{w}^T\mathbf{y}+\boldsymbol{\theta}\mathbf{w}^T\mathbf{w}\boldsymbol{\theta})}{\partial\mathbf{w}} = \mathbf{0}, \\ \mathbf{0} &= -2\mathbf{y}\boldsymbol{\theta}^T + \mathbf{w}(2\boldsymbol{\theta}\boldsymbol{\theta}^{\mathbf{T}}), \\ \mathbf{w} &= \mathbf{y}\boldsymbol{\theta}^T(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}, \\ \mathbf{w}^T &= (\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\mathbf{y}^T. \end{aligned}$$

For numerical stability we add a small value λ to the argument of the inverse leading to ridge regression as

$$\mathbf{w}^T = (\boldsymbol{\theta}\boldsymbol{\theta}^T + \lambda\mathbf{I})^{-1}\boldsymbol{\theta}\mathbf{y}^T.$$

We further compute the Covariance of each prediction step by

$$\boldsymbol{\Sigma}_t = \frac{1}{k-1}\sum_{j=1}^{k}\left(\mathbf{s}'_j - (\mathbf{A}_t\mathbf{s}_{t,j} + \mathbf{B}_t\mathbf{a}_{t,j} + \mathbf{c}_t)\right)\left(\mathbf{s}'_j - (\mathbf{A}_t\mathbf{s}_{t,j} + \mathbf{B}_t\mathbf{a}_{t,j} + \mathbf{c}_t)\right)^T.$$

As a result we have a covariance matrix for the states at each time step t. This information is used in the relaxed constraints of our main Quadratic Program (3.11).

3.6 Two Possible Policy Representations

In this section, we briefly want to present two possible policy representations, we will investigate in our experiments.

3.6.1 Complete Information Chance Constrained Policy (CICCP)

In the sections before, we have parameterized the policy model as

$$\tilde{\mathbf{a}} = \tilde{\mathbf{K}}(\tilde{\mathbf{s}} - \tilde{\mathbf{s}}_c) + \tilde{\mathbf{a}}_c + \alpha_{cc}\tilde{\mathbf{k}}_{cc}, \tag{3.16}$$

where $\tilde{\mathbf{s}}$ are the current states, $\tilde{\mathbf{s}}_c$ are the reference states, $\tilde{\mathbf{a}}_c$ are the actions leading to the reference trajectory, $\tilde{\mathbf{K}}$ the time-varying feedback gains gained by the backward pass in iLQG and $\tilde{\mathbf{k}}_{cc}$ the forward terms gained through the Chance Constrained Optimization. The limiting parameter, which applies only a certain percentage of the forward term is α_{cc}. We call the policy representation (3.16) as **Complete Information Chance Constrained Policy (CICCP)**. The corresponding QP is exact the one we have derived in the subsection before (3.11).

3.6.2 Partial Information Chance Constrained Policy (PICCP)

We further want to investigate the policy representation

$$\tilde{\mathbf{a}} = \tilde{\mathbf{K}}(\tilde{\mathbf{s}} - \tilde{\mathbf{s}}_c) + \alpha_{cc}\tilde{\mathbf{k}}_{cc}.$$

This policy representation is very similar to (3.16) with the difference of setting $\tilde{\mathbf{a}}_c$ to $\mathbf{0}$ and therefore giving the optimization in (3.11) more freedom, as the systems we will test our algorithm on are underactuated and therefore have action constraints. We want to investigate, if this policy has advantages by dropping the information of the actions $\tilde{\mathbf{a}}_c$ from iteration before leading to trajectory $\tilde{\mathbf{s}}_c$. We will call this representation as **Partial Information Chance Constrained Policy (PICCP)**. The corresponding QP is nearly the same as in (3.11). We simply set $\tilde{\mathbf{a}}_c$ to zero to obtain our new objective.

3.7 Visual Example

We want to visualize, what it means to consider Chance Constraints in the optimiztion procedure. For this purpose we have created figure 3.1. We can see, how the mean is shifted, to increase the probability $P(s_1 \leq b)$ and reach the threshold $\geq 1-\beta$ for time step t. The distance of the state's mean s_1 to the bound b, which is required to fulfill the probability $\geq 1-\beta$ is determined by the covariance $\mathbf{\Sigma}_t$ of the states. Thus, having a high covariance leads to higher distances between mean and bound. For our approach care has to be taken, that too high covariances do not occur, as the feasible area of the mean can be an empty set then. Since we are considering lower and upper bounds for the states, the problem of infeasibility can occur as the lower bound claims the mean to be in an area which is defined as infeasible from the upper bound. Indeed, this issue occurred during our experiments. Thus, having good linearized dynamics with not too high variance is desired.

3.8 Experiments

We have tested our system on the inverted Pendulum of Open AI gym [35] and the cartpole problem in the Pybullet environment [36] which uses the MuJoco engine [37]. In both tasks, the agent has to swing up a link. For the inverted pendulum experiment, we have a state-space dimension of two, whereas we have a state-space dimension of four for the cartpole experiment. The action dimension is one for both experiments. Note that we had to re-define the reward functions appropriately. Both tasks are underactuated and additive disturbance was included to simulate uncertainties in the models. In our experiments we are going to compare the performances of two models of Chance Constrained iLQG (CICCP and PICCP), standard iLQG and box iLQG. The goal is to show that considering constraints during optimization increases performance.
We will first show the constraint violations of the constrained states and the actions during the optimization procedure of the nominal linear systems of each algorithm. Additionally, we will show the linearization errors respectively. For investigating the constraint violations, we consider the nominal, linear systems ($\alpha = 1$) of all algorithms, whereas we shoot trajectories of the linear system with the corresponding line search parameters α and α_{CC} for every algorithm and calculate the mean squared error to the samples from the real system, in order to investigate the linearization error. With the gained information, we will investigate the connection between considering constraints of the nominal linear systems during optimization procedure and the overall performance by showing the reward curve. Note that we plot all trajectories and actions of the optimization procedure independent whether they improved the reward or not for investigating the constraint violations.

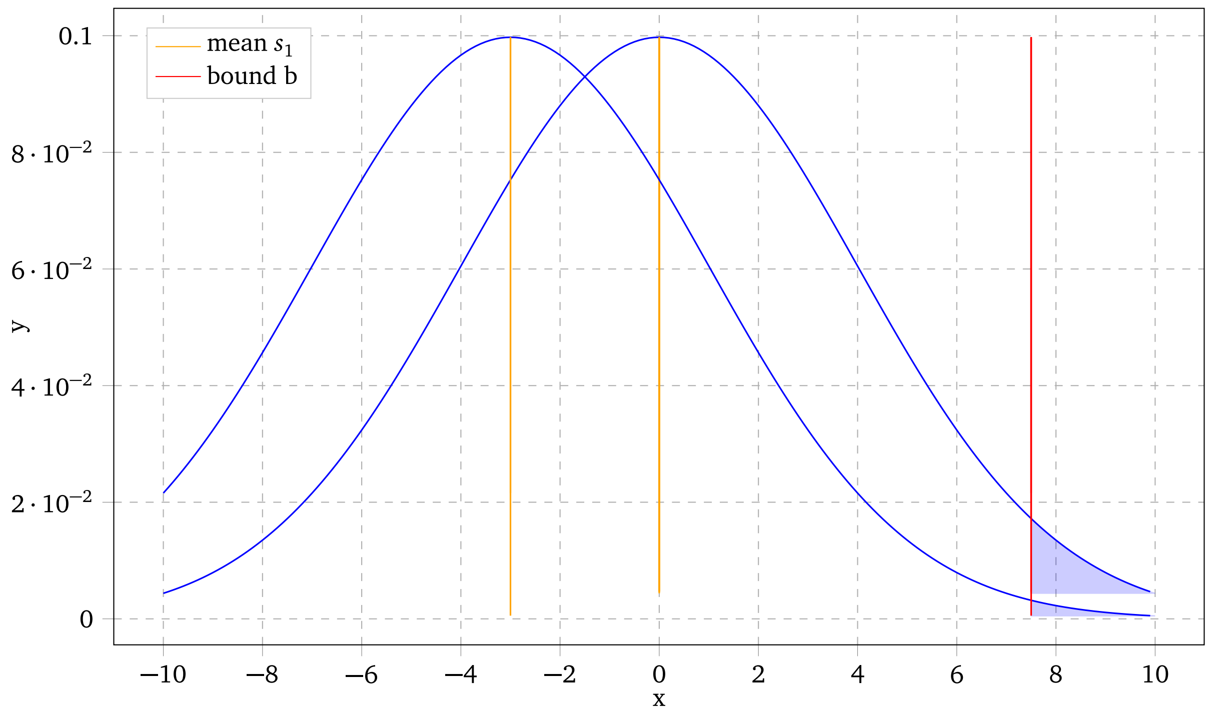

Figure 3.1: Trajectory for a time point t with an upper bound at $x = 7.5$. As the Chance Constrained policy can not influence the variance of the state's distribution, the only degree of freedom is to manipulate the mean. By this, the optimization tries to shift the mean in such a way, that the probability of $s_1 \leq b$ is within the $1-\beta$ quantile, which means it shifts the probability density of the state such that its VaR_β is equal or lower to the bound for this example. Thus we are directly manipulating the density function of the trajectory. For this case, the mean is shifted from 0 to -3 and the tail probability is therefore reduced. Consequently the probability $P(s_1 \leq b)$ is increased, as $P(s_1 \geq b)$ (blue surface) is decreased.

3.8.1 Inverted Pendulum

Before we present our results, it is worth to mention from beginning that Chance Constrained iLQG policies and iLQG policy have managed to swing up the pendulum. The policy obtained by box iLQG did not manage to swing up the pendulum.
The task is to swing up the pendulum from angle $\theta = \pi$ to angle $\theta = 0$, where the initial angle is random initialized with mean π and angles ranging between $\theta = 3.054 \ \ rad$ and $\theta = 3.229 \ \ rad$ within the 95% quantile. The velocity $\dot{\theta}$ is also initialized randomly with mean $\dot{\theta} = 0$ and ranging between -1 and 1 in the 95% quantile. The system is underactuated as the action is constrained. Furthermore the angle velocity $\dot{\theta}$ is constrained.
For the Chance Constrained policy representations a probability level of $\beta = 0.01$ for the action as well as for the state constraints was chosen. Note that this leads to the $1-\beta$ quantile, the state and action probability constraints have to be higher. Furthermore the hyperparameters for all algorithms were optimized before comparison. Table 3.1 summarizes the hyperparameters.
In the following, we will show the trajectories of $\dot{\theta}$ and the actions a_t planned by the algorithms. By this, we investigate the constraint violations during the optimization process. Note that we show the planned trajectories of the linear nominal system without noise and we set the corresponding line search parameters $\alpha = 1$ for all algorithms. Additionally, we will show the linearization error per iteration. For that purpose, we calculated the mean squared error per time step and averaged the linearization error for each iteration then. We consider the difference of the trajectories generated by linear systems with their corresponding line search parameters α and α_{CC} and the corresponding trajectories of the real system. As we run our experiments on 20 seeds, we calculate the variance for each iteration.

	α	β
CICCP	0.7	0.01
PICCP	0.7	0.01
iLQG	0.5	-
Box iLQG	0.6	-

Table 3.1: The Hyperparameters of the algorithms for the experiment were optimized before comparing them to each other. We can see that the line search parameters for the Chance Constrained iLQG policies are higher than compared to standard iLQG and box iLQG. This confirms our assumptions that we can use our policies more greedily by satisfying the constraints in the optimization procedure. Interestingly the policy update for PICCP has the same value as CICCP. We would have assumed to have higher line search values for this policy as we neglect information given from the iteration before and therefore give the method more freedom within the action space.

Constraint Violations of Nominal System and Linearization Error

In figure (3.2) the space where the angle velocity trajectories s_2 of the mean linear systems of each algorithm for $\alpha = \alpha_{CC} = 1$ are located, is shown. For this plot we took the trajectories getting nearest to the constraints and then filled the space between them, as the trajectories generated during the optimization procedure are located within this space. We can clearly observe that the Chance Constraint policies do not violate the state constraints and guarantee to be within the $1 - \beta$ quantile, whereas iLQG and box iLQG plan the unconstrained case for the states and therefore violate the constraints.
Same procedure is done for the planned actions in figure 3.3 during optimization iterations. The actions nearest to the constraints have been plotted and the space between has been filled. This space is visualizing the area, the taken actions during the optimization procedure for the nominal system are located. We can observe that the Chance Constrained iLQG policies and box iLQG are satisfying the constraints.
In figure 3.4, we further show the average linearization error of the algorithms during the iterations. We can clearly see that box iLQG has big linearization errors, which explains the bad reward curve in figure 3.6. Additionally the high linearization error explains that box iLQG has problems dealing underactuated stochastic systems, since the feedback gain is turned off as soon as the action constraint is violated. Turning off the feedback gain leads to an open-loop control. For the pendulum this case occurs often. Additionally, we can observe that iLQG has a low linearization error because the violations of the constraints are not dramatically, as we have seen in the plots before. Nevertheless, the Chance Constrained algorithms show the trend of small linearization errors (see figure 3.5) for the first six iterations. As soon as the Chance Constrained algorithms converge, the feedback gain gets more conservative due to the regularization. Since we do not optimize the feedback gain with respect to the constraints, the forward term and the feedback gain do not correspond to each other as soon as the feedback gain gets too conservative. This issue leads to an increasing linearization error, when convergence enters. Note that the linearization error was calculated by using α and α_{cc} for each algorithm respectively according to table 3.1, when simulating the corresponding linear system of each algorithm. This makes sense as the actions taken for the non-linear system are also considering α and α_{cc} during the forward pass respectively.

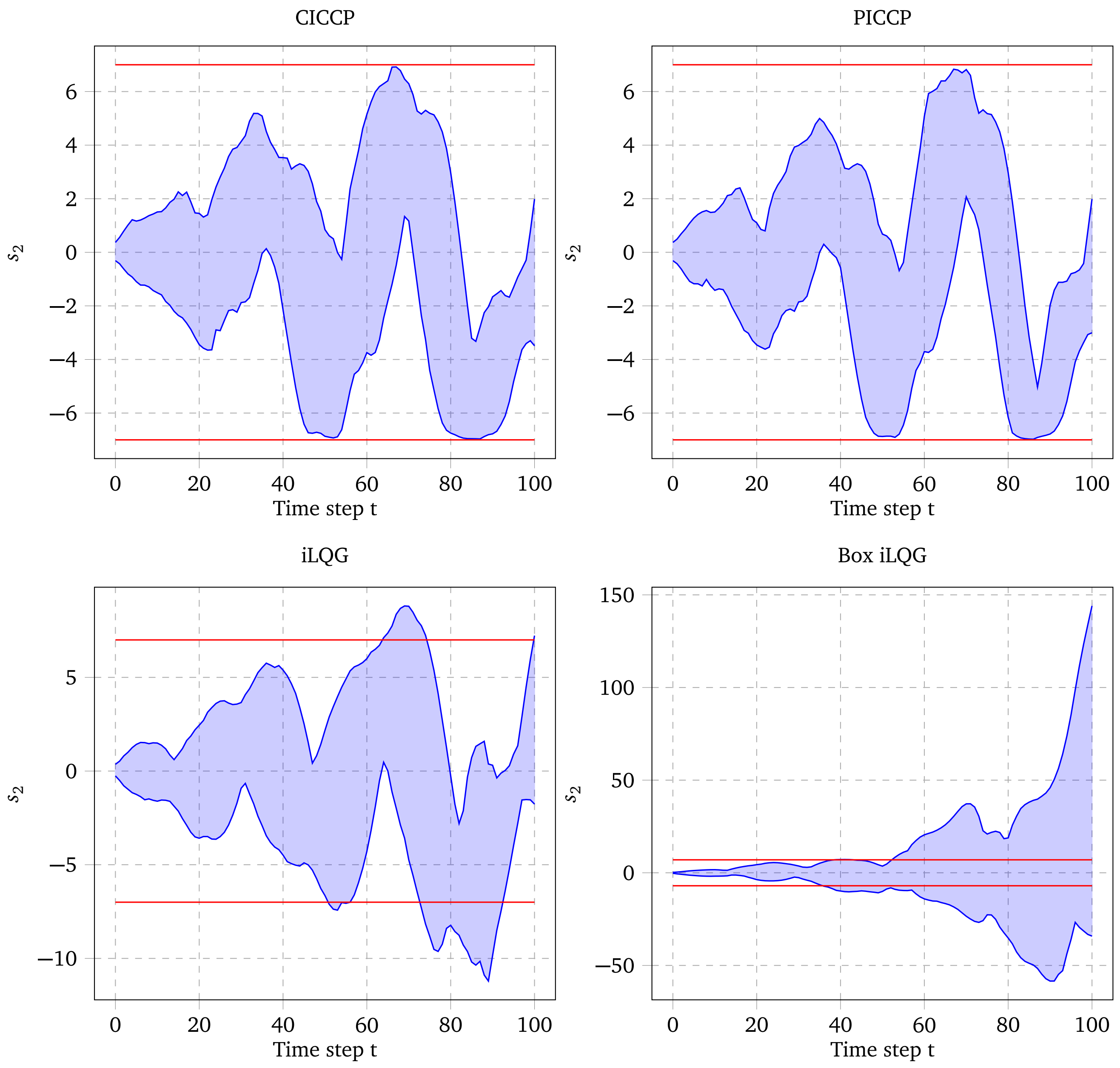

Figure 3.2: Trajectories of $\dot{\theta}$ of the pendulum: First plot: **CICCP** (upper left), second plot: **PICCP** (upper right), third plot: **iLQG** (lower left), fourth plot: **Box iLQG** (lower right).
Note that the location of the trajectories are visualized with the filled space. The trajectories which are nearest to the bounds have been plotted and the space between the two extrema have been filled, visualizing the area the trajectories of the mean linear systems during optimization are located. CICCP and PICCP are satisfying the constraints. For time steps, where the covariance is not high, the trajectories come near to the constraints, but being within the $1-\beta$ quantile is guaranteed due to the Chance Constraints.
Although iLQG does not consider constraints during the optimization, the violations are not big. Nevertheless, we can see violations from time step 60 to the end.
Box iLQG plans the trajectories by only considering action constraints. However, the constraints in the states are not considered which lead to violations. Some planned trajectories have very high violations.

Reward Curve

The overall performance of all algorithms are reflected in the reward curve in figure 3.6. We can see that using the Chance Constrained iLQG policies with higher α_{cc} gains lead to a bit faster convergence (CICCP). PICCP also performs good, but is outperformed by iLQG after iteration 2. We can also see that iLQG performs very good, although it is

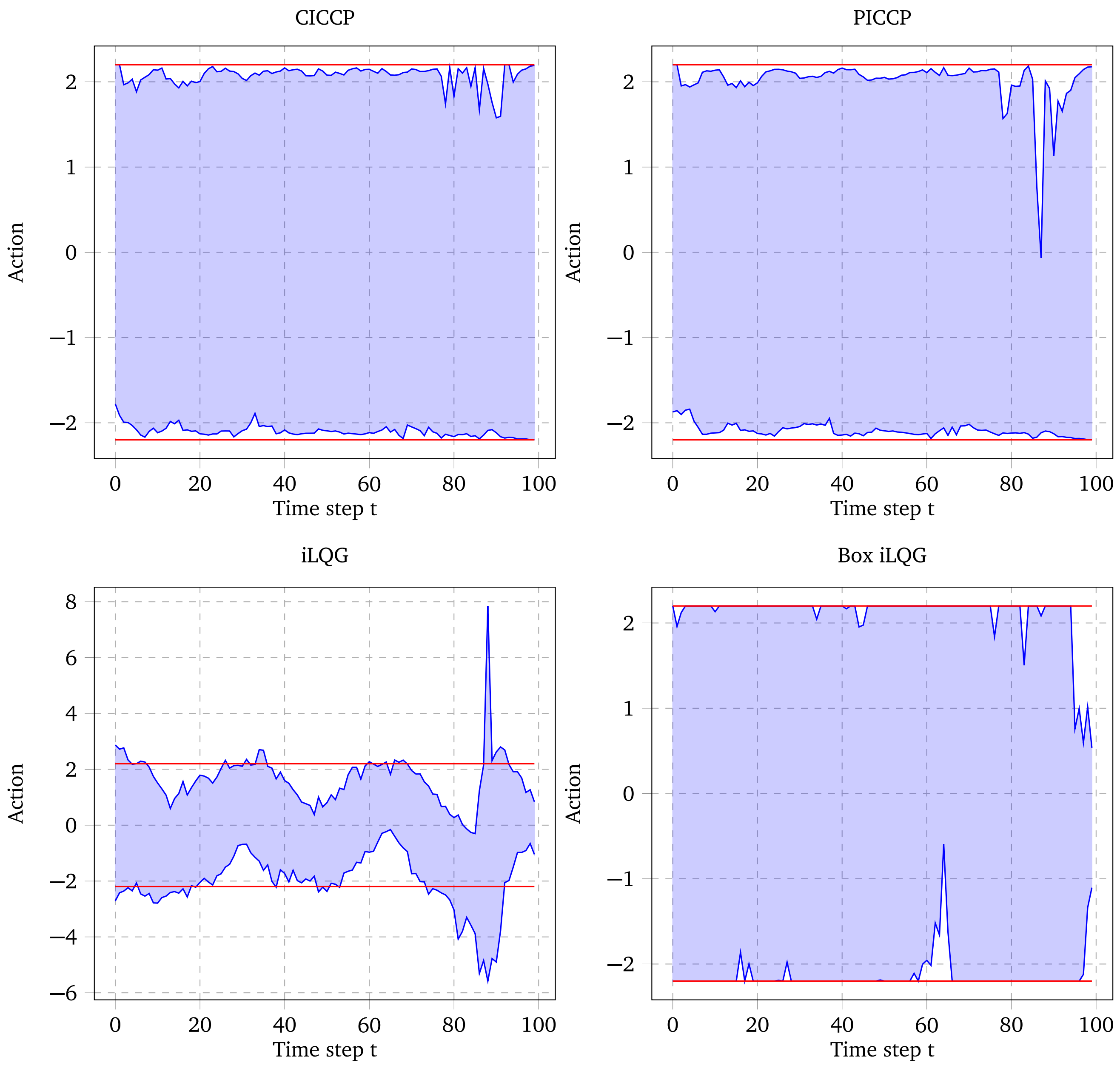

Figure 3.3: Actions taken during planning horizon. First plot (upper left): **CICCP**, second plot (upper right): **PICCP**, third plot (lower left): **iLQG**, fourth plot (lower right): **box iLQG**.
Note that the shadowed area is representative for the space the actions with respect to the mean linear system of the algorithms are located during the optimization. These plots are representative for the actions during all iterations of optimization.
We can clearly see, that CICCP, PICCP and box iLQG satisfy the constraints. CICCP and PICCP guarantee to keep the constraints within the $1-\beta$ quantile, whereas box iLQG always goes to the maximum of the constraints. Although iLQG does not consider constraints during optimization, the constraint violations are not dramatic high.

violating the constraints during the optimization phase for the nominal ($\alpha = 1$) mean linear system. However, as we have seen, iLQG does not violate the constraints much and therefore the learned dynamics stay valid for the planning horizon. Furthermore only a factor of $\alpha = 0.5$ of the forward term is used, which also reduces the violations.
We can also see that box iLQG performs bad as the feedback gains are set to zero for the most time of the planning horizon, which leads to an open-loop control and makes it vulnerable to disturbances. Furthermore no reference trajectory is given in the optimization, when feedback gains are turned off. For a perfect knowledge of the dynamics without disturbances,

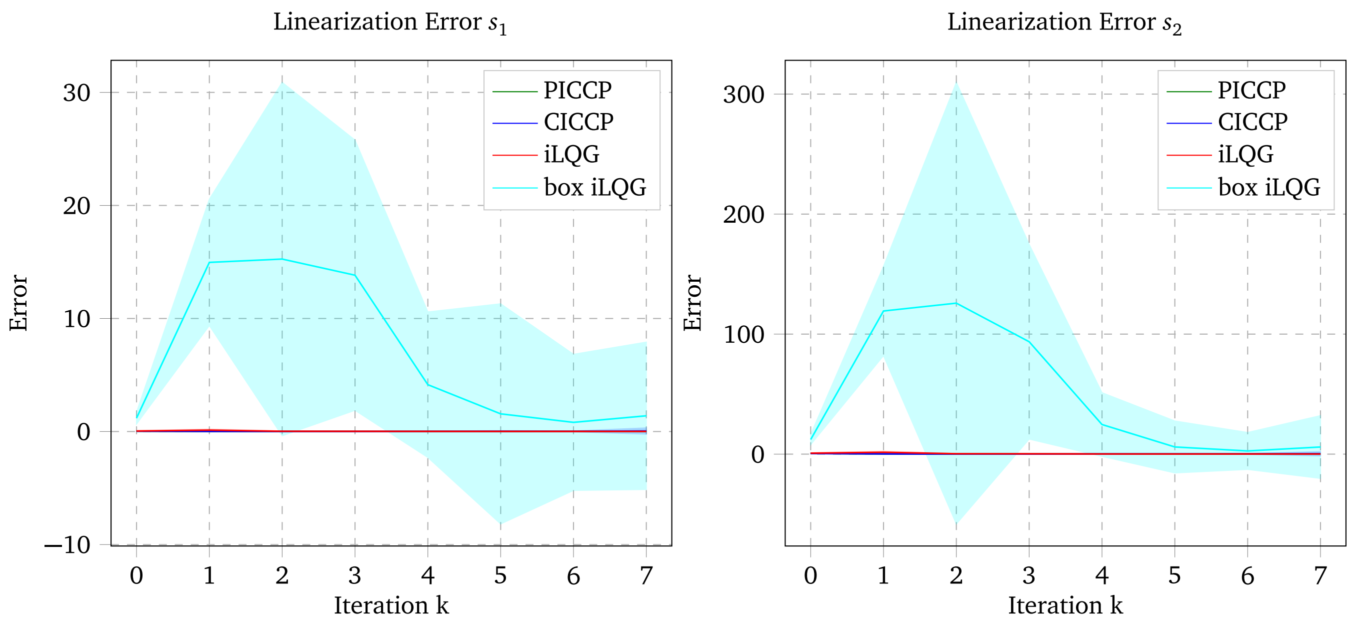

Figure 3.4: The linearization errors of the states. Box iLQG has very high errors, whereas iLQG and the Chance Constrained algorithms have low errors (see figure 3.5). This high error of box iLQG explains the bad performance (see figure 3.6) in addition. The fact that box iLQG "turns off" the feedback gain as soon as the actions constraints will be violated, leads to an open-loop control which is vulnerable to stochasticity.

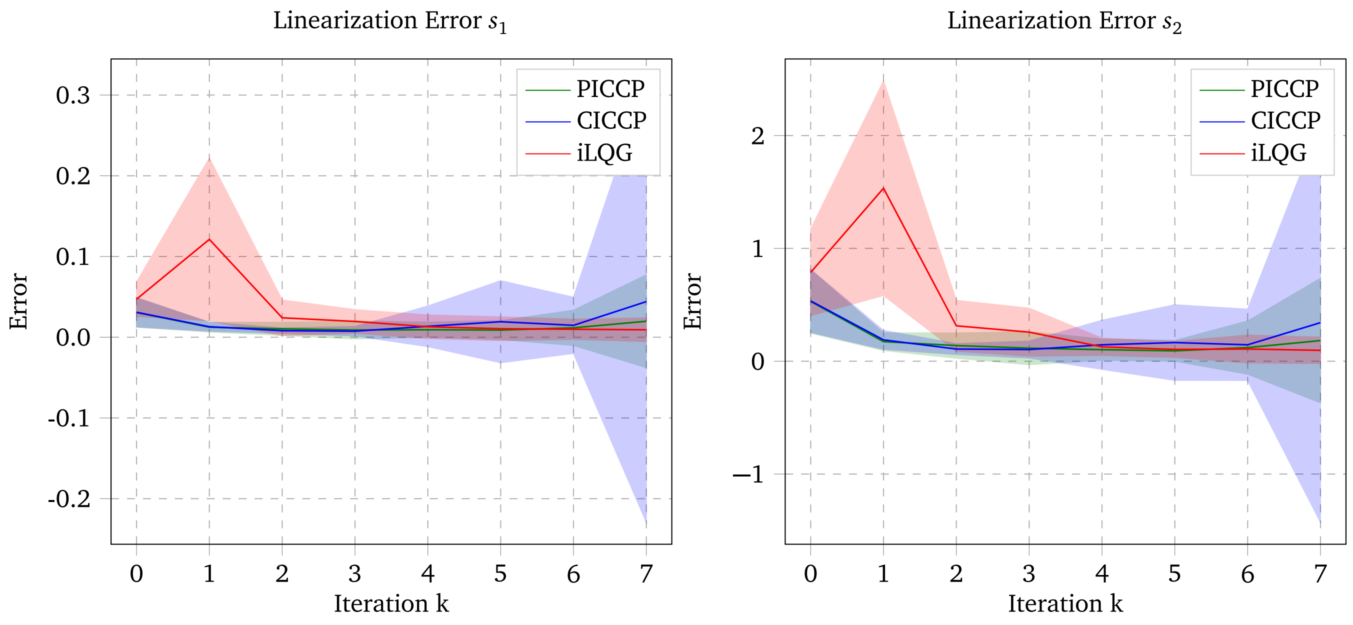

Figure 3.5: The linearization errors of the states without box iLQG error. We can observe that all algorithms have low absolute errors. Especially for the angle s_1, the errors are very small. For the angle velocity s_2, we can see that iLQG first has higher errors, whereas the Chance Constrained algorithms keep the error small until convergence (iteration 3). As soon as convergence begins, the feedback gain gets more conservative, which leads to uncorrelated behavior between the forward term and the feedback gain, since we consider action constraints during optimization and do not optimize with respect to the feedback in terms of satisfying constraints. However, the trend to minimizing the linearization error is given. All in all, no big error for iLQG is observable. This low error stresses that the violations of iLQG are not necessarily leading to invalid linearization dynamics, especially as we use only a percentage α of the forward term. Thus, iLQG performs good.

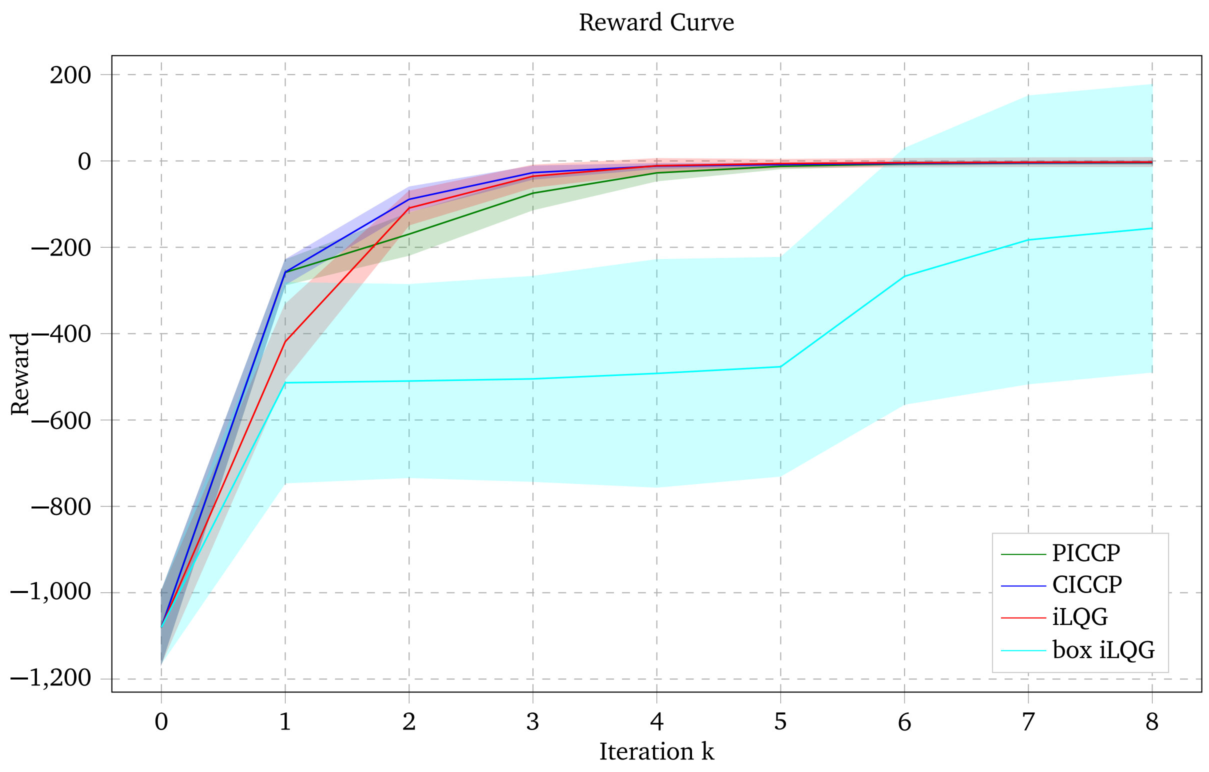

Figure 3.6: The reward curves of all algorithms for the pendulum. While CICCP (blue curve) converges a bit faster than iLQG (red curve), PICCP (green curve) is outperformed by iLQG after second iteration. Box iLQG performs bad since the feedback gains are "turned off" the most time due to the underactuated system, which lead to higher linearization errors and thereofore bad performance.
The small constraint violations of iLQG during the planning do not impair the validity of the linearized dynamics, as we additionally use only 50% of the calculated forward term. However, higher values for the line search parameters can be obtained for the Chance Constrained iLQG policies.

this issue does not provide any problems. But since we do not make assumptions about the non-linear dynamics and learn linear dynamics instead, box iLQG has difficulties to manage this task.

3.8.2 Cartpole

In the cartpole experiment, the agent tries to swing up a pendulum. The pendulum is fixed on a cartpole which is controlled. As the space, the policies can drive the cartpole is restricted, it is a hard task to swing up the pendulum for iLQG. Furthermore, the actions are restricted. The position state and the velocity of the cartpole are initialized at zero (the middle of the space of the cartpole position). The link's angle θ is initialized at $\theta = \pi$ with $\dot{\theta} = 0$. The actions are initialized randomly within the action space $-1 \leq a \leq 1$ underlying a normal distribution with zero mean. The hyperparameters of the algorithms were optimized before comparing the algorithms, which lead to table 3.2.

In this experiment only CICCP managed to swing up the pendulum. Also it sometimes happened that the cartpole has violated the position constraint for the non-linear system. PICCP also managed to swing up the pendulum without colliding with the position constraint, but there always stayed and angle error such that the pendulum rarely reached $\theta = 0$. Note that we bound the position of the cartpole for the optimization procedure at 1.1 for the upper bound and at -1.1 for the lower bound. The real system has its bounds at 1.2 and -1.2. Thus, we make it more conservative for the Chance Constrained iLQG policies and reduce the probability of violating the real constraint in addition to the pre-defined probability.

	α	β
CICCP	0.2	0.3
PICCP	0.2	0.3
iLQG	0.2	-
Box iLQG	0.1	-

Table 3.2: As for the pendulum experiment, the hyperparameters of the algorithms for the cartpole experiment were optimized before comparison. We have chosen the policy updates according to the best reward curve. Furthermore we can see that the policy updates for the Chance Constrained iLQG policies and iLQG are same. The reward function is defined as the link's angle square difference with zero, since the angle for the upper position is zero.

Constraint Violations of Nominal System and Linearization Error

As in the pendulum experiment we plot the space, where the trajectories of the cartpole's position s_1 are located. We therefore plot the trajectories nearest to the constraints obtained by the mean linear system with $\alpha = 1$ and $\alpha_{cc} = 1$ for each algorithm. We consider the trajectories which arise during the iterations of the optimization procedure and thus, are planned by the algorithms. In figure 3.7 we can clearly see that the planned position trajectories are within the constraint space for CICCP and PICCP, whereas box iLQG and iLQG do not consider the constraints and plan by violating them.
The same practice is done to show the space for the actions during the planning horizon for the linear nominal systems ($\alpha = 1$), chosen from the algorithms. The shadowed space in figure 3.8 shows the area, the actions are located. As expected, the Chance Constrained iLQG policies CICCP and PICCP and box iLQG policy are considering the constraints and thus, no violation for the nominal linear systems occur. Since iLQG optimizes the unconstrained case, the planned actions are violating the constraints clearly.
In figure 3.9 the linearization errors are plotted for all four states. We can see high variance for iLQG in every state, where the mean of the error is generally the highest compared to the other algorithms. A clear difference of iLQG's linearization error to the other algorithms' error can be seen for the velocity states.
For the Chance Constrained algorithms, generally the mean and the variance of the linearization error are small. However, as soon as convergence enters, the feedback gain gets more conservative and the linearization error increases therefore. This increasement is clearly observable for the velocity of the link's angle s_4. In iteration 8 the error starts to obtain high values. If we look on the reward curve in figure 3.10, we can see that the Chance Constrained algorithms converge at iteration 8. Thus, no reward improvement enters and the feedback gain is getting more conservative, which lead to problematic behavior, as the feedback gain is not optimized in order to satisfy the Chance Constraints and we are just using a percentage α_{CC} for the forward term of the policies. For the other states, this error increasement is given in form of higher variance for both Chance Constrained iLQG policies.
Nevertheless the higher mean and variance of iLQG proves our expectations. The unconstrained planning leads to violating the constraints and thus, to linearizations which are not valid, causing higher errors. Box iLQG has low linearization errors throughout the cartpole problem. In general, we have lower stochasticity in this task, as we do not initialize stochastically except for the actions, which has positive effects on linearization error of box iLQG. However, Chance Constrained errors and box iLQG's errors are very small for the most iterations for all states.

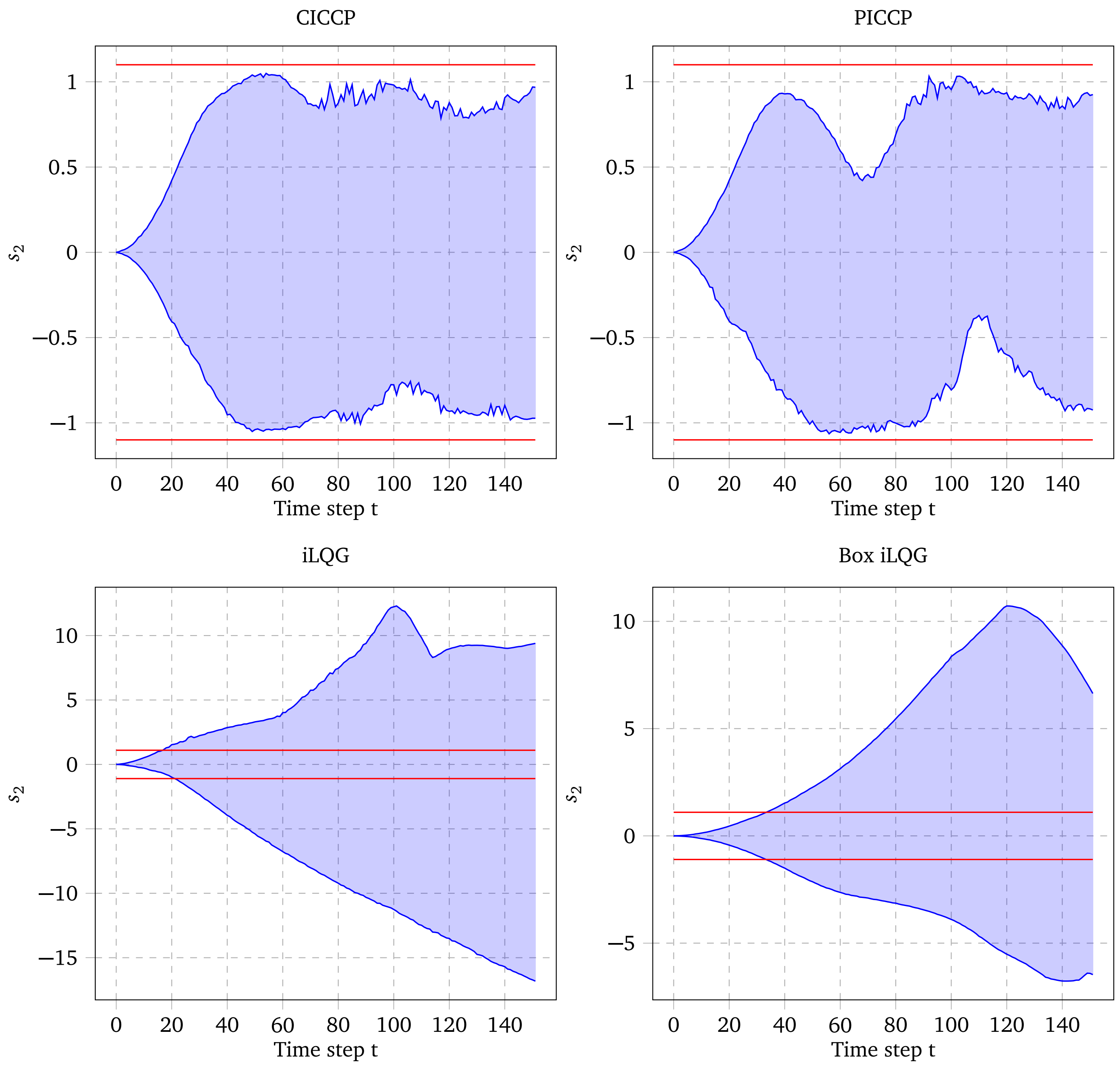

Figure 3.7: Trajectories of the cartpole position s_1. First plot (upper left): **CICCP**, second plot (upper right): **PICCP**, third plot (lower left): **iLQG**, fourth plot (lower right): **box iLQG**. Figure shows the planned trajectories for the nominal linear system for all algorithms with $\alpha = 1$. Note that the real system has a bound at 1.2 and -1.2 as upper and lower bound. For the optimization procedure of CICCP and PICCP we consider 1.1 and -1.1. Thus, we made the task more conservative for the Chance Constrained iLQG policies and decreased the probabilities for violating the the constraints of real system by this in addition. When executing the trajectorieson real system, all algorithms see the real bound 1.2 and -1.2 for upper and lower bound respectively. The filled space is representative for the space the trajectories are located during the optimization procedure. CICCP and PICCP consider the constraints during optimization. As a result, the constraints are not violated during the planning horizon for the linear systems and furthermore, they stay within the $1-\beta$ quantile. The prongs of the Chance Constraint policy's trajectories are due to different covariance values at each time step. The third figure shows the trajectories planned by iLQG, which clearly violates the position constraints. Box iLQG also does not consider state constraints during the optimization, leading to violations.

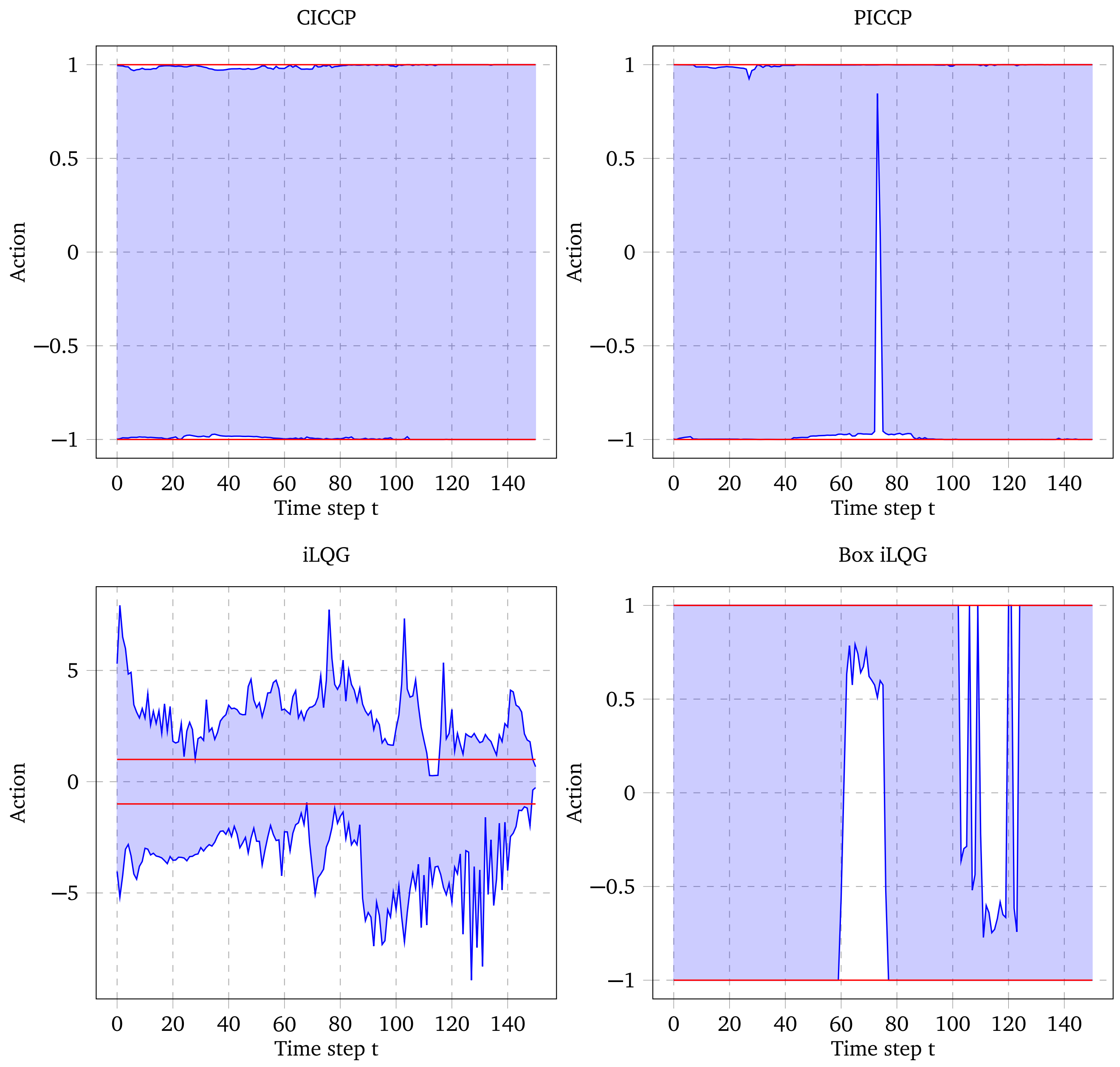

Figure 3.8: Actions during the optimization procedure for linear nominal systems: First plot (upper left): **CICCP**, second plot (upper right): **PICCP**, third plot (lower left): **iLQG**, fourth plot (lower right): **Box iLQG**. CICCP and PICCP are satisfying the action constraints. As we do not have a stochastic initialization on the states, the covariance of the controller is small, leading to be able to exploit the constraints. However, it is guaranteed, that the actions are kept within the $1-\beta$ quantile for the linear nominal systems. As iLQG moves in the unconstrained space, the constraints are not considered and therefore highly violated. Box iLQG considers action constraints in the backward pass, which can be seen in the last figure. However, the feedback gain is set to zero, at time steps, where the constraints would be violated. Note that the blue colored space is representative for the area the actions are located during the optimization procedure of the nominal mean system with $\alpha = 1$ respectively.

Reward Curve

Normally, the reward curve reflects the algorithm's performance. However, iLQG often explicitly used the constraints to swing up the pendulum by driving the cartpole against the constraints and gain rewards through the swing up. This swing up is uncontrolled and can be seen in the higher linearization error of the link's angle velocity s_4 for iLQG, as well as in the high variance of the reward curve of iLQG in figure 3.10. CICCP clearly achieves the highest rewards (see

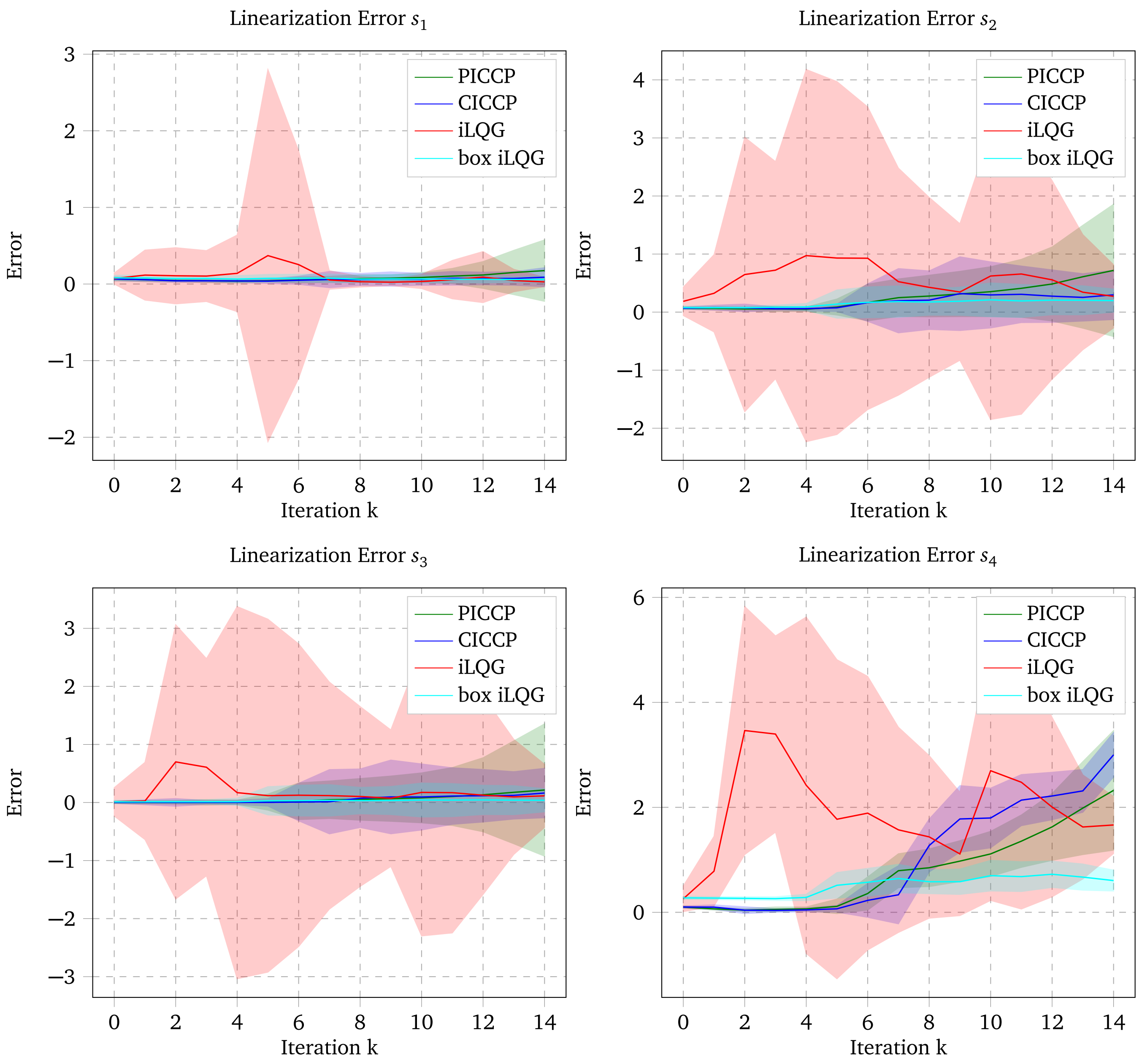

Figure 3.9: The linearization errors for the states. Upper left: s_1. Upper right: s_2. Lower left: s_3. Lower right: s_4. We can generally see higher mean and variance for iLQG compared to other algorithms. Especially for the velocity of the angle s_4 and the velocity of the cartpole s_2 iLQG has a high error. Chance Constrained iLQG policies (CICCP and PICCP) have lower errors until enter of convergence in iteration 8, where the mean increases drastically for the angle's velocitiy s_4 and a bit higher variance can be seen for the other states from this iteration on. This fact is caused by the feedback gain, which gets more conservative as the reward is not improved. Box iLQG has low linearization error, due to no stochastic initialization in this task.

figure 3.10). The linearization errors of the states confirm our assumptions (see figure 3.9). Using constraints within the optimization procedure reduces the linearization errors. A clear relation between lower linearization errors and higher reward performance can be made for the case of Chance Constrained iLQG policies, especially for CICCP. While CICCP achieves the highest reward, PICCP converges to a lower reward value and has higher variance in the reward curve. Box iLQG again has problems with stochasticity, but performs much better than compared to the pendulum task. This performance improvement lies especially in the fact that we do not have as much stochasticity as for the pendulum case since we do not consider stochastic initialization for the states.

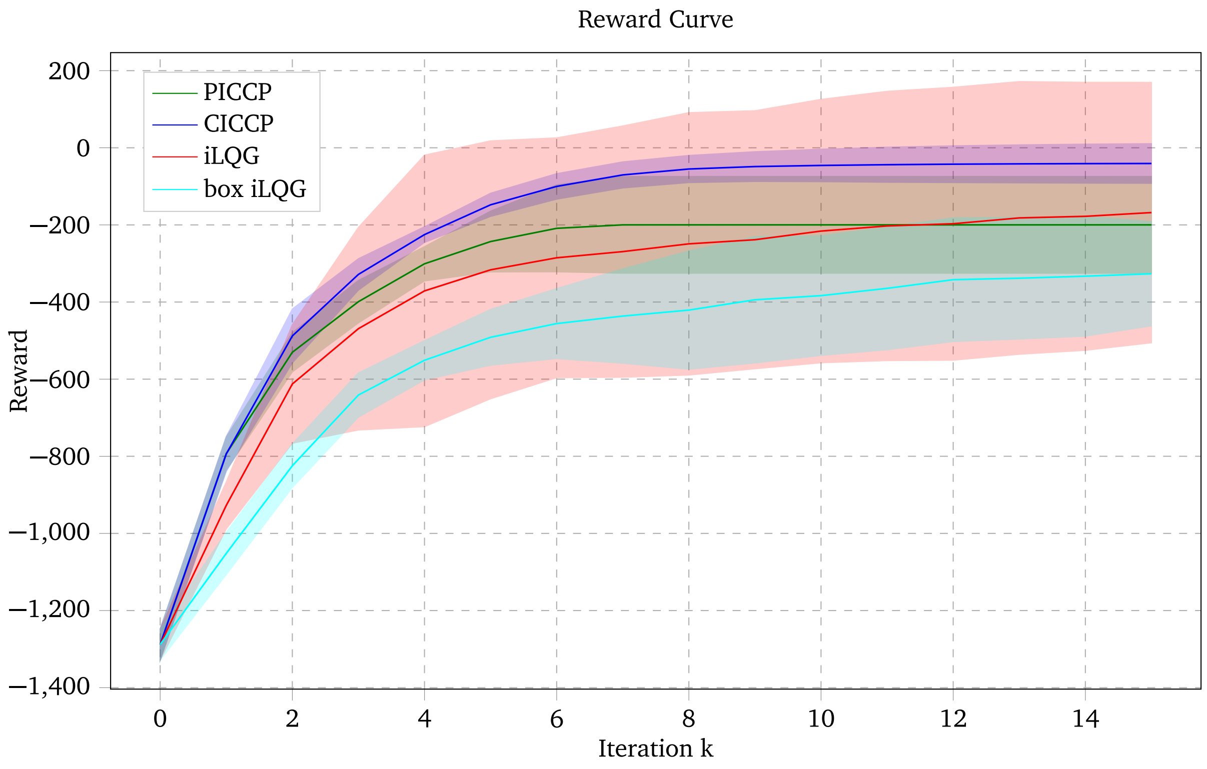

Figure 3.10: The reward curves for the cartpole of all algorithms in comparison. CICCP clearly achieves the highest reward and reduces the reward variance. PICCP converges to a lower bound, which is outperformed by iLQG's mean after 12 iterations. The loss of information lets PICCP stack into a local optimum. iLQG has high variance in the reward curve, since it often drives against the bound. Box iLQG suffers under the lack of the feedback gain for a major of time, which lead to the lowest reward achievement.

3.8.3 Conclusion of the Experiments

We have tested the algorithms on the pendulum and the cartpole. For the inverted pendulum, we have seen that iLQG and the Chance Constrained iLQG policies perform comparable except for box iLQG according to the reward curve. The constraints were not violated much by iLQG, which lead to stable and good linear models and therefore good performance. However, we have seen that we can use the forward terms with higher line search values by considering the constraints in the optimization procedure, which led to slightly faster convergence for CICCP. Box iLQG has shown bad convergence behavior as the feedback gains are turned off because of the underactuated system, which lead to an open-loop control, which makes the algorithm vulnerable for uncertainties and disturbances. We have clearly seen that initial high stochasticity for the states has a big impact on box iLQG, since box iLQG performed much better for the cartpole compared to the pendulum. With this bad performance we can clearly see that the feedback gain is essential for good results.
We further tested the algorithms on the cartpole, which is harder to solve, since the space for the position of the cartpole is restricted. We have seen that both, iLQG and box iLQG violate the position constraints during planning. However, box iLQG did not violate the action constraints in contrast to iLQG. Especially for iLQG these constraint violations lead to higher linearization errors and therefore lower reward curves. In contrast, the Chance Constrained iLQG policies plan the trajectories considering the constraints. We have seen that the linearization error is smaller therefore. But in addition, one could see that the linearization error increases as soon as the Chance Constrained algorithms start to converge. This convergence lead to conservative feedback gains and therefore does not correlate with the forward term, which uses only a percentage α_{CC} for the forward term. However, until convergence, the Chance Constrained algorithms have shown small linearization errors, which lead to better models and therefore to higher rewards and faster convergence. CICCP has the highest reward achievement. The loss of information for PICCP leads to decreased performance compared to CICCP. Again box iLQG has problems because of the missing feedback gain. Although this problem is not given in the intensity as for the inverted pendulum, the consequences are clearly observable.
Regarding the choice between CICCP and PICCP, it is clear to say that the missing information of $\tilde{\mathbf{a}}_c$ lead to poorer performance compared to CICCP. Therefore CICCP policy representation is preferred.

4 Regularized Stochastic Optimization with Chance Constraints

Inspired by Relative Entropy Policy Search, which bounds the policy change per iteration using the KL-Divergence, we will present in this chapter a new approach, which uses Chance Constraints to regularize the reward change. We interpret the introduced constraint as a measure for risk.
For this purpose, we will first introduce a brief overview on Policy Search methods including Relative Entropy Policy Search (REPS) and then give a brief overview on risk measures, in order to show the new approach.

4.1 Policy Gradients and Information-Theoretic Policy Search

Model-Free policy search is a sub area of Reinforcement Learning. For the case of robotics, sampled trajectories are evaluated and the corresponding reward is returned. In general, the average return

$$J_{\boldsymbol{\theta}} = \mathbb{E}[R(\tau)|\tau] = \int R(\tau)p(\tau)d\tau,$$

is tried to maximize, where $p(\tau)$ denotes the trajectory distribution [38]. A major distinction in policy search algorithms is made between Policy Gradient Methods and Information-Theoretic Approaches.

4.1.1 Policy Gradient Methods

In Policy Gradient Methods, one aims to update the policy parameters $\boldsymbol{\theta}$ with the gradient of the expected return $J_{\boldsymbol{\theta}}$ as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}},$$

with α being the learning rate. The gradient of the expected return is defined as

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}} = \int_{\theta} \nabla_{\tau} p_{\boldsymbol{\theta}}(\tau) R(\tau) d\tau.$$

In general, $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$ has to be estimated, which lead to different Policy Gradient Methods [38]. Examples to estimate the gradients are Finite Difference Methods, Likelihood-Ratio Policy Gradients and Natural Gradients. The latter method uses another metric to update the policy parameters. While Finite Difference Methods and Likelihood-Ratio Policy Gradients generally use an Euclidean metric $\delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta}$, with $\delta\boldsymbol{\theta} = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$, Natural Gradients Method use an approximated KL-Divergence [38]. The idea of utilizing the KL-Divergence instead of the euclidean distance arises because, for the euclidean distance, small changes in the policy parameters $\boldsymbol{\theta}$ can infer large changes on the resulting distribution $p_{\boldsymbol{\theta}}$. The KL-Divergence overcomes this problem, as the distance between the probability distributions is measured in terms of information relation [38]. This advantage is used in natural gradient, where the KL between the probability densities of the policy parameters $\mathrm{KL}(p_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}||p_{\boldsymbol{\theta}})$ is approximated using the Fisher information matrix

$$\mathbf{F}_{\boldsymbol{\theta}} = \mathbb{E}_{p(\mathbf{y})}\left[\nabla_{\boldsymbol{\theta}} \log p(\mathbf{y}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{y})^T\right],$$

which results in [38]

$$\mathrm{KL}(p_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}||p_{\boldsymbol{\theta}}) \approx \delta\boldsymbol{\theta}^T \mathbf{F}_{\boldsymbol{\theta}} \delta\boldsymbol{\theta}.$$

4.1.2 Information-Theoretic Policy Search

In Information-Theoretic Policy Search algorithms the motivation is to stay near to the data while updating the policy. This implies, that the trajectory distribution $p(\tau)$ after the policy update should not jump away. As already mentioned, the KL-Divergence as a measure for the distance between probability density functions is an appropriate measure [38].

Natural Policy Gradients already use the KL-Divergence, but the learning rate of the policy update has to be tuned by hand. Expectaion Maximization (EM) based policy search methods can be used to overcome this issue. However, for EM-based algorithms other problems as premature convergence occur [38].
In Relative Entropy Policy Search (REPS), first introduced in [39], the KL-Divergence measure is used too. In addition to that, weighted maximum likelihood estimates of reward samples from the policy from the iteration before are used to update the policy. By this, a learning rate for the policy update drops.
The optimization problem [38] for the episode-based REPS algorithm is formulated as

$$\max_{\pi} \int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{4.1}$$

$$\text{s.t. } \epsilon \geq \int \pi(\boldsymbol{\theta}) \log \frac{\pi(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}, \tag{4.2}$$

$$1 = \int \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{4.3}$$

In this optimization problem the expected reward is maximized subject to the KL-Divergence between the old policy $q(\boldsymbol{\theta})$ and the new policy $\pi(\boldsymbol{\theta})$. The last constraint guarantees the policy to be a probability density function. As mentioned, the KL-Divergence guarantees to maintain information from the old policy. By formulating the Lagrangian and setting it to zero [6], eliminating the Lagrangian variable for last constraint (4.3), the optimal policy [38] is obtained as

$$\pi(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta}\right),$$

with η being the Lagrangian multiplier resulting from the KL-bound constraint. This Lagrangian multiplier is obtained by minimizing the dual function (see [6] for Dual Theory) [38]

$$g(\eta) = \eta\epsilon + \eta \log \int q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta}\right) d\boldsymbol{\theta}. \tag{4.4}$$

The gained dual function is optimized with respect to the Lagrangian multiplier η. The policy can be updated by for example weighted maximum likelihood with reward samples of the policy from the iteration before[38].
The dual form in (4.4) is approximated by samples from the previous policy as

$$g(\eta) = \eta\epsilon + \eta \log \sum_{i} \frac{1}{N} \exp\left(\frac{R\left(\boldsymbol{\theta}^{[i]}\right)}{\eta}\right) d\boldsymbol{\theta}.$$

Thus, the evaluation of the expectation in (4.4) can be done numerically with a sum.
A part of the dual form of REPS corresponds to the Entropic Risk Measure, which will be clarified in section 4.3. Before interpreting this correspondence, we want to give a brief overview on risk measures.

4.2 Risk Measures

In many cases of optimization problems, one is not interested in the mean objective, since it is desirerd to optimize user specific expectation given the objective. Sometimes it also can be that the mean objective is not accurate enough, as we need a certain amount of data to know that the mean is an accurate measure. Additionally, sometimes it is also wished to be risk averse. As the expectation is a risk neutral measure, there is the need of introducing new measures, giving information about the current risk aversion [9]. Of course, this risk aversion can be reversed to risk seeking, which can be exploited for exploration. In Reinforcement Learning problems, encouraging exploration is often wished. For many cases, there is also risk awareness for reward improvements desired. For these cases, risk measures can be used [40].

4.2.1 Utility Functions

A possibility to introduce user specific measures is the use of utility functions. Instead of maximizing, or minimizing the expected value of the random process, the expected value of the utility function is optimized. In financial mathematics this type of risk measures are common [9]. However, also in technical systems it is sometimes preferred to use utility functions. We show a small overview on these approaches in section 4.2.3. However, an example is the Linear Exponential Quadratic Gaussian (LEQG), where the objective C is mapped to a scalar using the utility function [41]

$$u(C) = e^{\gamma C},$$

leading to the new risk averse objective

$$R_\gamma(C) = \frac{1}{\gamma}\log\mathbb{E}\left[e^{\gamma C}\right]. \tag{4.5}$$

The need of log function will be clarified in the next section.
By choosing γ appropriately, i.e. $\gamma < 0$ or $\gamma > 0$, one gets risk-seeking or risk-sensitive respectively [41].
In general, if the case of maximizing the objective is given, the decision leading to the outcome C_2 compared to the decision leading to the outcome C_1 is preferred, if [9]

$$\mathbb{E}[u(C_1)] < \mathbb{E}[u(C_2)].$$

It is also claimed that $u(C)$ is concave. Note that, if the expectation of $u(C)$ shall be minimized, then $u(C)$ is called *disutility function* and is convex [9].
Utility functions have to be defined by the user himself, which turns out to be a difficulty and if found, they are often hard to interpret [9].

4.2.2 Risk measures

To overcome the difficulties given by utility functions, the concept of risk measures exist. Risk measures are functionals which expect an entire collection of realization of random variables [9].
We want to present the most known risk measures in this section.

With **Mean-Risk Models** it is possible to characterize the uncertain outcome Z not only with the mean of Z, but also considering the risk $\mathbb{D}[Z]$, which additionally measures the uncertainty of Z [9]. Thus, the risk measure can be formulated as

$$\rho(Z) = \mathbb{E}[Z] + c\mathbb{D}[Z], \tag{4.6}$$

where c is a scaling factor. A possible and well-known example for risk measure is the variance. By weight c importance on the uncertainty can be placed. Thus, if for example, the risk measure (4.6) shall be minimized, the grade of "risk-sensitivity" can be chosen by weighting c appropriately [9].

Value at Risk (VaR) [9] is a very popular and well-known risk measure. Considering

$$H_Z(z) = Pr(Z \leq z),$$

with $H_Z(z)$ being the cumulative density function of random variable Z, and $Pr(Z \leq z)$ the probability that Z is smaller than z, then the left-side α - quantile is given as

$$H_Z^{-1}(\alpha) = \inf\{t : H_Z(t) \geq \alpha\}.$$

The $1-\alpha$ quantile $H_Z^{-1}(1-\alpha)$ is defined as $\text{VaR}_\alpha(Z)$, if Z represents loss [9]. The mathematical description [9] is given as

$$VaR_\alpha(Z) = H_Z^{-1}(1-\alpha) = \inf\{t : Pr(Z \leq t) \geq 1-\alpha\}. \tag{4.7}$$

Thus, VaR returns t which will be the $1-\alpha$ quantile bound of the underlying probability density function of Z. What VaR gives as information is that values larger than $VaR(Z)_\alpha$ will not occur with probability $1-\alpha$.

Conditional Value at Risk (CVaR) [9] is strongly related to the risk measure VaR. Mathematically it is even defined depending on the VaR [9] as

$$\text{CVaR}_\alpha(Z) = \frac{1}{\alpha}\int_{1-\alpha}^{\alpha} \text{VaR}_{1-\tau}(Z)\ d\tau.$$

An alternative description [9] is given through the conditional expectation

$$\text{CVaR}_\alpha(Z) = \mathbb{E}[Z|Z \geq \text{VaR}_\alpha(Z)].$$

Thus $\text{CVaR}_\alpha(Z)$ is the expected value of the loss Z given that the loss is higher than the $\text{VaR}_\alpha(Z)$. This means that the "worst-case" is considered, where the expected value of the tail is returned, which is not considered in VaR.
For a loss Z, the **Entropic Risk Measure** is defined as [10] [9]

$$\rho^{ent}(Z) = \frac{1}{\theta}\log\mathbb{E}\left[e^{\theta Z}\right].$$

Therefore the utility function $e^{\theta Z}$ was used in addition to considering the logarithm of the expected value. This has to be done, to obtain a coherent risk measure [9]. The Entropic Risk measure first weights the possible loss value Z with parameter θ , takes the exponent of them and returns the log of the expected value with respect to loss Z. As a result, high values for Z are weighted higher. With parameter θ the user can define his own risk awareness.
For a **coherent** risk measure ρ the properties

1. Convexity: $\rho\left(tZ + (1-t)Z^{'}\right) \leq t\rho(Z) + (1-t)\rho(Z^{'})$
2. Monotonicity: If $Z \geq Z^{'}$, then $\rho(Z) \geq \rho(Z^{'})$
3. Translation equivariance: $\rho(Z + a) = \rho(Z) + a$
4. Positive homogeneity: $\rho(tZ) = t\rho(Z)$

are claimed. For a **convex** risk measure, which measures the loss, properties (1.-3.) are claimed [9]. If one wants to measure the reward, property 2. changes the inequality direction and property 3. occurs as $-a$ instead of $+a$. In the shown risk measures, CVaR and the Entropic Risk Measure are convex risk measures [9].
With the Fenchel-Moreau theorem [9], convex risk measures can be represented in their dual form as

$$\rho(Z) = \max_{Q\in M}\left[\mathbb{E}_Q[-Z] - \alpha(Q)\right],$$

where M is a set of probability measures and the functional α is called penalty function. For a given Q in the convex duality theory, one aims to find the minimal penalty function [10] of ρ by

$$\alpha_\rho(Q) = \max_Z\left[E_Q[-Z] - \rho(Z)\right].$$

For the entropic risk measure, the minimal penalty function is the relative entropy $\alpha_\rho(Q) = \frac{1}{\theta}H(Q|P)$, where H denotes the relative entropy and P is the reference probability measure to Q [10].
All in all we can summarize the presented risk measures as follows.
While VaR_α returns the bound for the $1-\alpha$ quantile of the loss probability density function, $CVaR_\alpha$ considers the "worst case" by returning the expected loss value, if the value for VaR_α gets exceeded. Thus both risk measures consider a certain range of losses. In contrast, the entropic risk measure considers all possible risk values, which are weighted by taking the exponent.

4.2.3 Small Overview on Connections to Control

We already have shown as an introductory example for utility functions in section 4.2.1 that LEQG uses a special exponential utility function. At this point the relation of entropic risk measure and LEQG gets clarified. It is obvious, that LEQG uses the entropic utility function presented in section before, to be risk sensitive from the Control Engineering point of view [41].
Indeed, there also exist relations, between LEQG and robust control. In [42] robust properties of risk-sensitive control are shown and a derivation of the stochastic small gain theorem is made. The authors additionally motivate that the stochastic small gain theorem has strong relations to the risk-sensitive criterion and can therefore be expressed with it. In their work, the authors additionally show, how the entropic risk measure can be used to broaden the space of true models by measuring the relative entropy of different stochastic models to each other and with this information bound the standard objective f by explicitly using the dual formulation of the entropic risk measure. For this purpose they make use of the dual formulation

$$log\int_M e^f \ d\theta = \sup_{v\in M}\int_M f \ dv - H(v||\theta).$$

Furthermore θ denotes the design model (for example the dynamic model) and v denotes the true model, whereas f are the costs. To obtain an upper bound of the costs depending on the model v, the inequality

$$\int_M f \quad dv \leq \log \int_M e^f \quad d\theta + H(v||\theta)$$

is used.
In [43] the authors show the relationship of the risk parameter γ (4.5) and the H_∞ controller.

4.3 Dual of Relative Entropy Policy Search and Entropic Risk Measure

The dual formulation of Relative Entropy Policy Search (REPS) from equation (4.4) can be written as

$$g(\eta) = \eta\epsilon + \eta \log \mathbb{E}_{q(\boldsymbol{\theta})}\left[\exp\left(\frac{R(\boldsymbol{\theta})}{\eta}\right)\right].$$

This formulation of the dual is very similar to the entropic risk measure introduced in section 4.2.2 as

$$\rho^{ent}(Z) = \frac{1}{\theta}\log \mathbb{E}\left[e^{\theta Z}\right]. \tag{4.8}$$

Note that Z was declared as loss in the section before, in order to weight high losses and infer these as high risk. For REPS the opposite case is seeked. REPS explicitly searches for high rewards by weighting the reward values by taking the exponent.
In order to reinterpret the log part of the dual of REPS as entropic risk measure we have to interpret the loss as reward $Z = R(\boldsymbol{\theta})$. In addition, the relation

$$\eta = \frac{1}{\theta}$$

is made. Thus, we can rewrite the dual in terms of the entropic risk measure by considering the derived relations as

$$g(\eta) = \eta\epsilon + \rho^{ent}_{\eta}(R(\boldsymbol{\theta})).$$

As the dual is optimized with respect to η in order to obtain the Lagrangian variable, it seems that REPS does not provide the user a possibility to tune his risk-awareness. However, as η is the Lagrangian variable corresponding to the KL-bound, the bridge between constraining the policy update and the risk-awareness with respect to the reward is given. The value for η is depending on the policy update. If we in general allow aggressive policy updates in terms of high ϵ values as KL-bound, η will obtain corresponding values consequently. Thus, by choosing ϵ, we consequently make decision on our behavior regarding risk-awareness. For high ϵ values we tend to risk-seeking behavior, where we want to move in regions where the reward is high by corresponding policy updates, whereas we tend to risk-awareness for small ϵ values and try to make minimal changes in the reward differences by staying close to the old policy. Thus, obviously a relation is existing.

4.4 Maximum Entropy Optimization with Chance Constraints

As discussed in section 4.1, Information-Theoretic Policy Search algorithms try to maximize the expected reward by constraining the policy update using the KL-measure. By this, the loss of information is counteracted and jumps in the policy updates are prevented [38]. In section 4.3, we have suggested that bounding the KL by ϵ corresponds to risk-awareness/risk-seeking behavior, where high ϵ values lead to higher expected reward changes. Inspired by this relation, we want to present a new point of view for Regularized Stochastic Optimization using Chance Constraints. Instead of tuning the risk behavior by constraining the policy update as in REPS, we directly want to bound our reward change per iteration by using Chance Constraints and do not make assumptions on the policy update. In addition, we can directly assign risk awareness by bounding the reward change per iteration and tune the risk behavior on the reward change by assigning a probability value for the declared Chance Constraint which determines the quantile the bound has to be in.
The following optimization problem describes the new approach

$$\max_{\pi(\boldsymbol{\theta})} \; H(\pi(\boldsymbol{\theta})), \tag{4.9}$$

$$\text{s.t.} \; Pr(R(\boldsymbol{\theta}) - R_{\text{old}} \leq b) \geq 1 - \alpha, \tag{4.10}$$

$$Pr(R(\boldsymbol{\theta}) - R_{\text{old}} > 0) \geq 1 - \beta. \tag{4.11}$$

We want to maximize the entropy of the current policy $\pi(\boldsymbol{\theta})$ to ensure exploration. With the entropy as objective we also have a convex objective function.
The first constraint (4.10) determines the risk awareness. As mentioned, two parameters define the risk behavior. First by setting the value b as the upper bound for reward change and second by setting the probability value α for the $1-\alpha$ quantile the reward change has to be in. Note, by fixing the values for b and α, we directly define the corresponding Value at Risk and consequently the Conditional Value at Risk underlying the probability distribution of the reward change, as we fix information to the optimization in form of the bound b of the quantile $1-\alpha$. This fact shows in addition the meaning and the task of constraint (4.10) as a risk measure.
Furthermore, to guarantee reward improvement, we have to ensure that the difference of the reward is positive, which is given in the second constraint (4.11) as the probability of reward difference greater than zero has to be greater or equal to the $1-\beta$ quantile. All in all we are bounding the reward change with an upper and a lower value with Chance Constraints. But in general we do not claim any restrictions on the policy and the policy updates. In fact we set the policy update free in its change such that it can jump within the policy space.
In problem (4.9) the objective is convex, as we consider the entropy to be maximized. Furthermore, we assume a normal distributed policy from the beginning. This makes it possible to formulate the objective in closed-form and no samples are needed to evaluate the entropy.
The reward function is a static function which returns the agent's performance with respect to the task. However, since we consider stochastic policies, we can not determine the reward value deterministically. Thus, the policy implies a probability density for the reward function, which we need to know to determine the probability constraints analytically. But assuming a normal distributed policy does not imply a normal distributed reward function. In fact the only case where this is given is a linear reward function. Furthermore, the reward function is not known in many cases, which additionally makes the task hard.
Crude Monte-Carlo simulation is a possible way to approximate the probabilities in the constraints. If we have rare events, the number of samples explode and the probability approximation gets inefficient. Using Cross-Entropy method to approximate the probability for rare-event probabilities is a possibility [16]. However, using these numerical methods in the optimization procedure does not necessarily lead to solutions in many cases, as an initial feasible solution is needed and finding them turns out to be hard [23]. Furthermore, we will need to sample from the policy as soon as it changes and evaluate the samples on the reward to estimate the probability. This procedure can be very costly, especially for high dimensional systems. In our experiments numerical methods to approximate the constraints only worked well, as long as we had big values for b and low values for $1-\alpha$ and $1-\beta$. This lead to "easy" fulfillable constraints and thus the solver had no problems to find updates for the decision variables. But for the normal case we want to have more strict constraints. We therefore search for another way of relaxing the probabilities.

4.4.1 Constraint Relaxation with Unscented Transformation and Problem Reformulation

Inspired by the UKF [24], we want to use the UT to estimate the probability density function of the reward function. In section 2.5 we gave a brief overview of UT.
In order to estimate the density function of the reward function given the density function of our policy $\pi(\boldsymbol{\theta})$, we use the equations (2.14) to determine the Sigma points $\boldsymbol{\chi}$ by simply plugging in the mean $\boldsymbol{\mu}_\pi$ and the covariance $\boldsymbol{\Sigma}_\pi$ of our current policy into the equations. As the parametrization of our policy is known from beginning, calculating the Sigma points $\boldsymbol{\chi}$ does not provide a problem. In fact, they are predefined by the mean, the covariance of our policy and the hyperparameter λ for UT. Furthermore, the weights are calculated by the equations (2.17) given the hyperparameters. We obtain the mean μ_R^π and the variance $(\sigma_R^\pi)^2$ given the current policy by fitting the Gaussian using the equations (2.20). Note that we have a one dimensional reward function and thus μ_R^π and σ_R^π are one dimensional. Dimension reduction does not provide a problem, since we are performing a non-linear transformation of the sigma points, which automatically adjusts the dimension.
Having fitted a corresponding probability density function of the reward function, reformulating the constraints in the optimization problem (4.9) as a deterministic constraint is a straightforward procedure, which we already have discussed in section 2.4.2. The advantage of the underlying problem is that we do not need to apply Boole's Inequality, since we already have a single probability. Consequently, we can simply use the cdf of a normal distribution to rewrite our first probability constraint (4.9) as

$$\frac{1}{2}[1+\frac{b-(\mu_R^\pi - R_{\text{old}})}{\sqrt{2}\sigma_R^\pi}] \geq 1-\alpha,$$
$$b-(\mu_R^\pi - R_{\text{old}}) - \text{erf}^{-1}(1-2\alpha)\sqrt{2}\sigma_R^\pi \geq 0.$$

The second constraint first has to be rewritten as

$$1 - Pr(R(\boldsymbol{\theta}) - R_{\text{old}} > 0) \geq 1 - \beta,$$
$$Pr(R(\boldsymbol{\theta}) - R_{\text{old}} \leq 0) \leq \beta,$$

in order to apply the cdf, which leads to the constraint

$$(\mu_R^\pi - R_{\text{old}}) + \text{erf}^{-1}(2\beta - 1)\sqrt{2}\sigma_R^\pi \geq 0.$$

Thus, we have reformulated the stochastic optimization problem in (4.9) as

$$\max_{\boldsymbol{\mu}_\pi, \boldsymbol{\Sigma}_\pi} \; H(\pi(\boldsymbol{\mu}_\pi, \boldsymbol{\Sigma}_\pi)), \tag{4.12}$$

$$\text{s.t.} \; b - (\mu_R^\pi - R_{\text{old}}) - \text{erf}^{-1}(1 - 2\alpha)\sqrt{2}\sigma_R^\pi \geq 0, \tag{4.13}$$

$$(\mu_R^\pi - R_{\text{old}}) + \text{erf}^{-1}(2\beta - 1)\sqrt{2}\sigma_R^\pi \geq 0. \tag{4.14}$$

Note, that we also parameterized our policy. Therefore, we can directly optimize with respect to the mean and the covariance $\boldsymbol{\mu}_\pi, \boldsymbol{\Sigma}_\pi$ of our policy. The form of our policy does not change, as we optimize with respect to the mean and the covariance, which implies a Gaussian distribution. The parameterization of the policy additionally brings the advantage that the covariance and the mean of our policy does not have to be estimated by samples. We just need to determine $2n + 1$ Sigma points, to obtain the estimated mean μ_R^π and the estimated variance $(\sigma_R^\pi)^2$ as soon as the constraints are called by the solver. Note that the mean of the policy is already given, which reduces the need of sigma points to $2n$.
Although we have a convex objective, the constraints we obtain through the reformulation are not convex. When estimating the reward's mean and covariance, we are applying a non-linear transformation, with respect to the policy mean $\boldsymbol{\mu}_\pi$ and the policy covariance $\boldsymbol{\Sigma}_\pi$ by evaluating sigma points on the reward function. Nevertheless, solving non-convex optimization problems effectively, is not a challenge nowadays. In our experiments we solved the optimization problem with $SLSQP$ sovler from python's $scipy\ optimize$ package.

4.5 Experiments

We tested the presented approach on finding the minimum value of two static functions. We first show the steps of the algorithm for a convex function and then go further to demonstrate it for a non-convex function. For this purpose we plot the iterations of the algorithm until convergence.

4.5.1 Convex Function

The function we are considering for the convex case is of quadratic form $f(x) = x^2$. We initialize the policy with $\mu_0 = 2$ and $\sigma_0 = 0.9$. Furthermore we introduce the bound for reward change as $b = 0.8$. Additionally we give the algorithm a probability value of $\alpha = \beta = 0.1$. The parameters are summarized in table 4.1.

μ_0	σ_0	b	α	β	f(x)
2	0.9	0.9	0.1	0.1	x^2

Table 4.1: The parameters for demonstrating the proposed method. Initialization of the policy with μ_0 and standard deviation σ_0. The bound for reward change is given as b as well as the probability values α, β. We are considering the convex function $f(x) = x^2$.

In figure 4.1 the iterations of the algorithm for the convex case can be seen. In green, the policy density function is shown, whereas the transformed density function of the reward function is given in orange. For each iteration we marked the mean of the policy as a red dot and visualized the sigma points with red x. Note that we did not plot the transformed sigma points to keep overview. We can clearly see that the algorithm converges to the minimum, where the standard deviation of the reward density function decreases as soon as the minimum is reached. Convergence enters in iteration 10. We also can see that the optimization tries to maximize the entropy of the policy before converging, as in iteration 10 the density function of the policy gets broad.
In figure 4.3 we show the probability values for each iteration. Besides the numerical probability values gained through Monte Carlo sampling, we show the analytical calculated probabilities, which we gained with the Unscented Transformation. As the algorithm uses the approximated density function, it can be clearly seen, that the bounds of the probabilities to be within the $1 - \alpha$ and the $1 - \beta$ quantile respectively are not violated. In addition we can see, that the Unscented Transformation approximates the density function of the reward function precisely, since the differences between the numerically calculated probabilities and the probabilities following from the approximated density function are small.

4.5.2 Non Convex Function

For the non-convex case we consider the function $f(x) = e^{-x+2.7} + e^{x-7} + \sin(x) + \sin\frac{10}{3}x - 0.84x + 3$. The initializations of the problem are the same as in table 4.1, except to the initial mean $\mu_0 = 3$ and the bound parameter $b = 0.8$. In figure 4.2 we show the iterations of the algorithm's procedure. The descriptions are the same. Green function denotes policy's density function, whereas the approximated density function of the reward is given in orange. Note that we did not label the axis due to lack of space.
Until iteration $i = 4$ the algorithm behaves as expected, when it reaches the local optimum of the non-convex function. In iteration $i = 5$, we can see that the policy jumps to $x = 6$, which is allowed, as we do not bound the policy's change. Thus, the method has recognized that the valley it is in, in iteration $i = 4$ is a valley of a local minimum and has detected that other regions of the function have smaller values and therefore return higher rewards. Interestingly, in iteration $i = 6$ the method decides to move into the valley of the local minimum at $x = 7$, where the method indeed converges. Thus, the method has converged to a local minimum. Nevertheless, it has recognized that the first valley is a local optimum.
In figure 4.4 we show the corresponding probability values. We can see that the probability values, which were approximated with the UT are always higher than the bound 0.9. We can also see that there is no high difference between the numerical calculated probability values with Monte Carlo sampling and the approximated probability values. The high peak for the constraint that the reward change should be positive results from the jump, the policy is doing from iteration 4 to iteration 5.

Different Initial Means for Policy

In this section, we want to show the algorithm's behavior for different initial means of the policy. For this purpose, we sampled uniformly 40 different means within the range [0,10]. The bound values and probability values for the non-convex function are the same as described before. Thus, we have a bound value $b = 0.8$ and both probability values $\alpha = \beta = 0.1$. Furthermore we increased the initial standard deviation to $\sigma_0 = 1.2$ for every initialization. For the described non-convex function, it is sufficient, to keep the sampling bounds at [0,10], since it has barriers at its sides. Thus, the global and local minima are located within the range [3,7.5]. Note that we have bounded the mean and the standard deviation to $\mu_{\max} = 100$, $\mu_{\min} = 100$ and $\sigma_{\max} = 100$, $\sigma_{\min} = 10^{-8}$, to avoid numerical problems, which occurred while using $scipy's\ slsqp$ solver.
In figure 4.5 in the upper plot the reward values are shown, whereas the standard deviations among the iterations is shown in the lower plot. Since we initialize uniformly, the variance of the expected reward is very high. Nevertheless, we can see that the variance decreases with the iterations and the mean of the rewards is increasing. This shows that the algorithm tends to find at least a local minimum. Note that we have a maximum possible value for the reward $R_{\max} = 3.01$ at the global minimum and a reward value of $R = 2.18$ for the local minimum at $x = 7$ and a reward value of $R = 0.635$ for the local minimum at $x = 3.5$.
Since we bound the change of reward with probability values, the standard deviation of the policy has to decrease directly in the first iteration. But we also can see the trend of decreasing standard deviation in general. The policy has to reduce its standard deviation, when reaching an optimal value.

4.5.3 Conclusions of the Experiments

We have seen that the proposed algorithm finds the optimal value for the convex function and a local optimal value for the non-convex case. We have seen that the policy can jump between two iterations. We also have shown that the algorithm tends to converge to an optimal value. However, to generate figure 4.5, we had to bound the maximum and minimum mean values to prevent numerical problems, as the numerical solver sometimes jumped to very high mean values, which lead to numerical issues, since we have an exponent in the non-convex function. Generally, this bounding should not impair the algorithm's behavior. Nevertheless, deriving a new way of dealing the underlying optimization problem in (4.12) is a point, which should be considered in future work.
During the experiments we have noticed that the method has problems for reward functions which show periodic behavior. As long as the policy covered these periodic regions, the method had problems to find the optimal value.

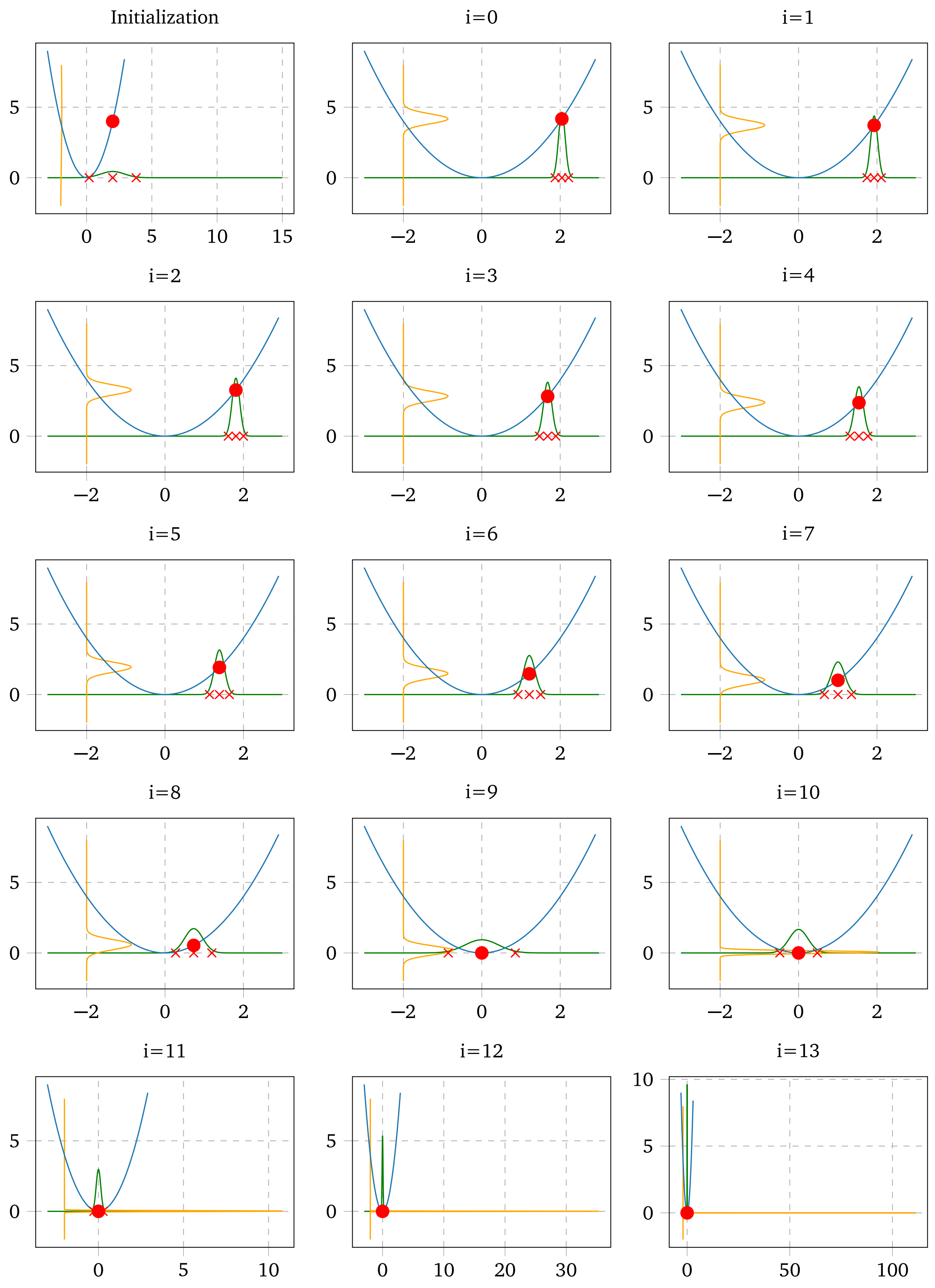

Figure 4.1: The proposed algorithm during optimization of the convex function $f(x) = x^2$. Note that we have dropped the axis labels due to lack of space. Initial Values and constraint values are given in table 4.1. The green function in each iteration is the probability density function (pdf) of the policy, whereas the approximated reward's pdf is given in orange. Red dots show the sigma points of policy's density function. The mean of the policy is marked as a red x. We can observe that the optimal value is found, as the algorithm is converging after 10 iterations.

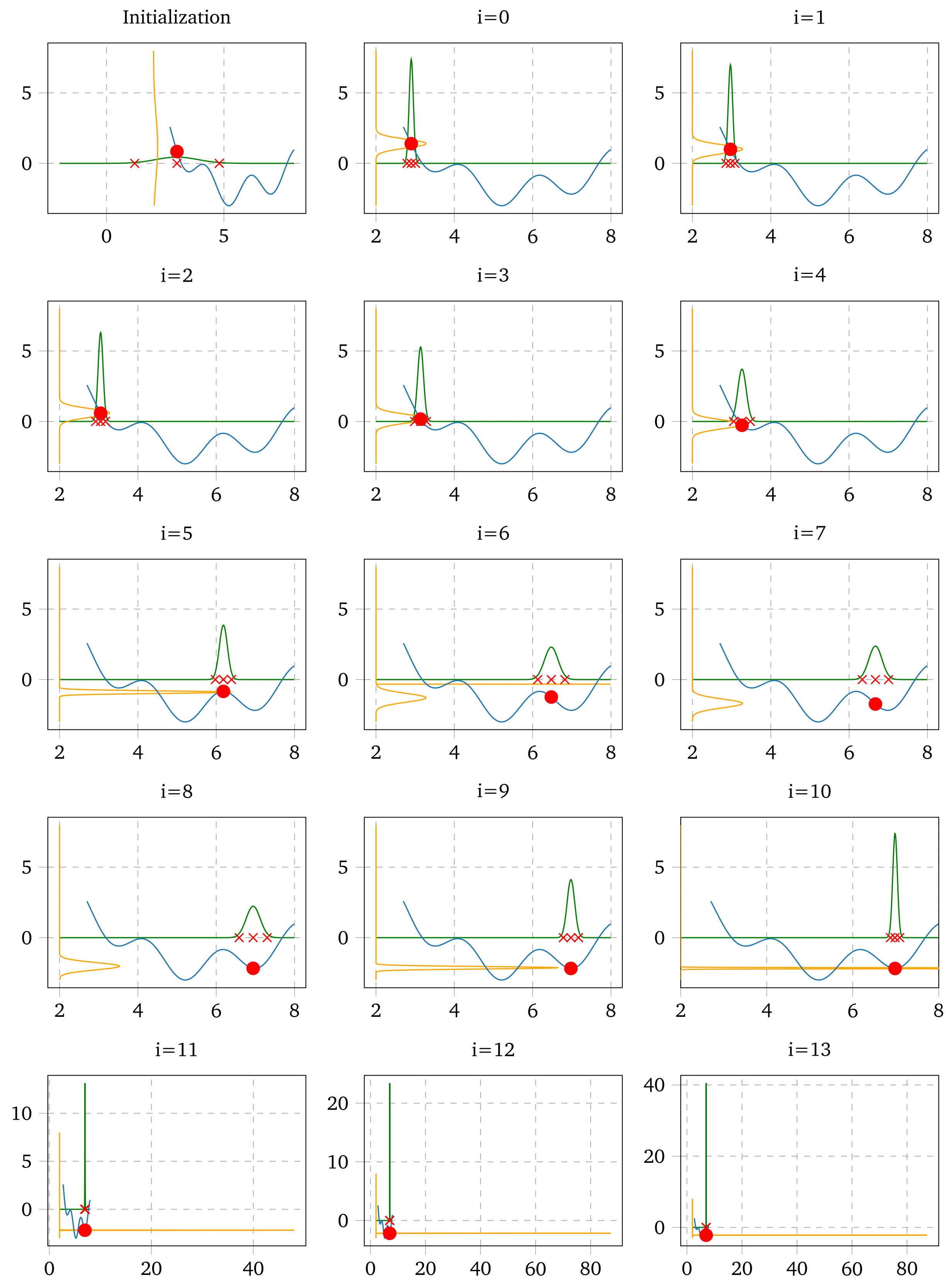

Figure 4.2: Non-convex function optimization. Initialization is given in table 4.1, where $\mu_0 = 3$ and $b = 0.8$ for this case. Note that we have dropped the axis labels due to lack of space. The green function is the pdf of the policy, whereas the approximated reward's pdf is given in orange. Red dots show the sigma points of the policy pdf. The mean of the policy is marked as a red x.The method arrives in the valley of the local optimum in iteration 4, which it leaves in iteration 5, by jumping to a region, where the reward is higher due to lower function values. The constraints are still satisfied (see figure 4.4) during this jump. Interestingly, the method converges to the local optimum x = 7 by moving slightly right in iteration 6.

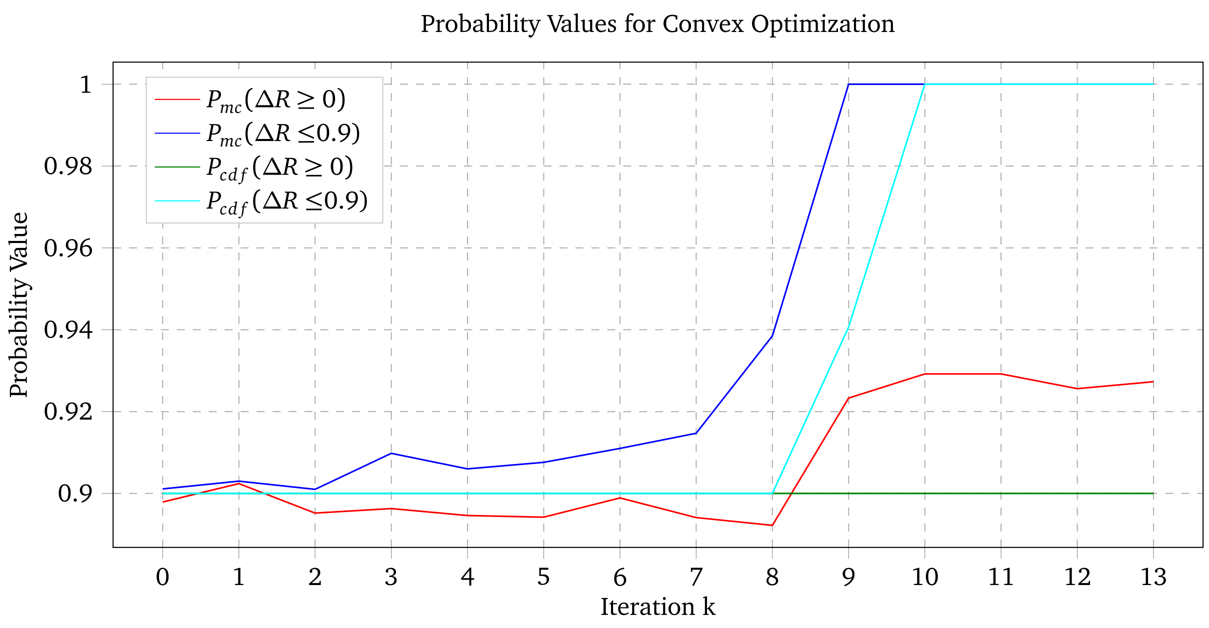

Figure 4.3: The probability values following from the constraints. P_{cdf} (green and cyan) denotes the probability values following from the analytical calculation of cdf of Normal Distribution which is gained by applying Unscented Transformation. We can see that P_{cdf} values satisfy the constraints.
We also calculated the probability values with Monte Carlo Sampling as a reference. The values lie near to each other showing good approximation behavior by the Unscented Transformation. Note that $k = 0$ does not correspond to the initialization in figure 4.1.

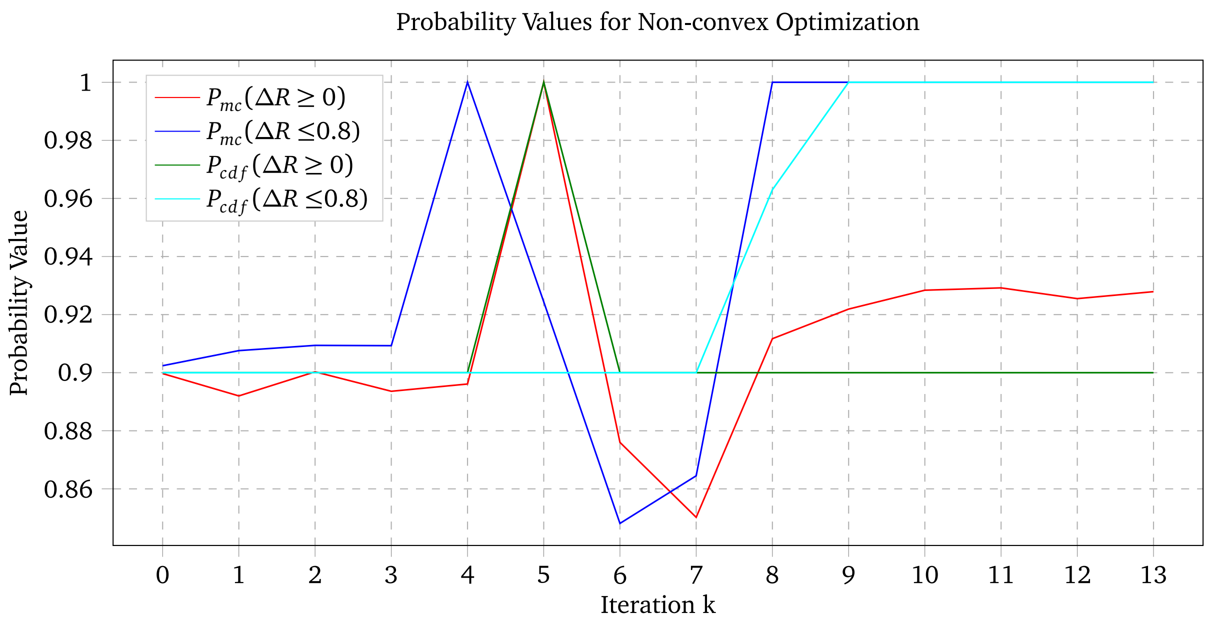

Figure 4.4: We show the probability values calculated from the Unscented Transformation and the numerically calculated probability values. We can see that the probability values gained with UT satisfy the constraints of being greater or equal to 0.9. Additionally, the numerically calculated (Monte Carlo sampling) values are not varying much compared to those of UT.

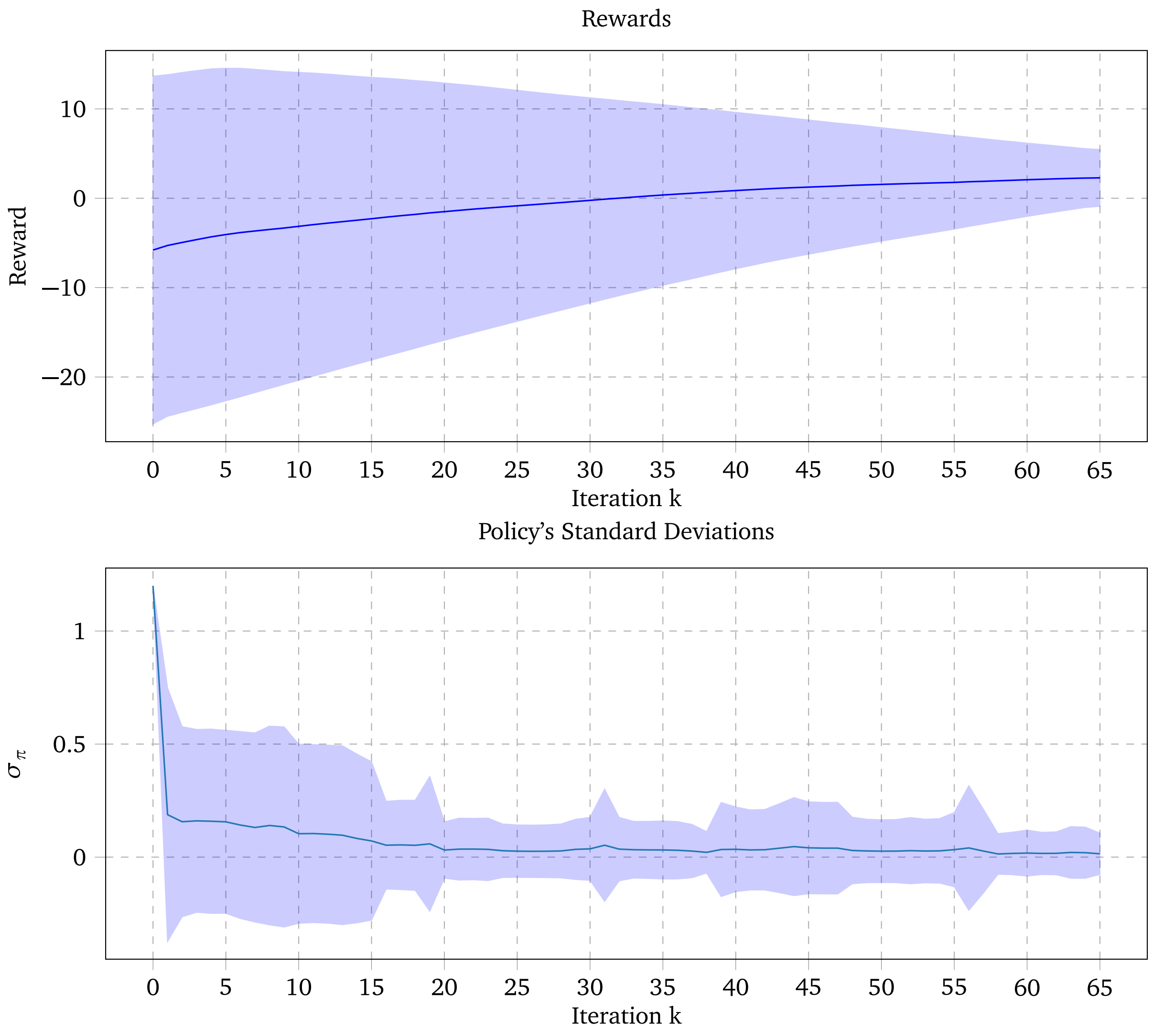

Figure 4.5: Reward curve and policy's standard deviation curve. We uniformly sampled different mean values for the policy in the range [0,10]. We initialized the policy with standard deviation $\sigma_0 = 1.2$ for every mean, to show that the algorithm at least converges to a local minimum. The high variance of the initial reward is due to uniformly sampling. Nevertheless the reward mean is increasing and the variance of the reward curve is decreasing. Note that we have a maximum possible value for the reward $R_{\max} = 3.01$ at the global minimum and a reward value of $R = 2.18$ for the local minimum at $x = 7$ and a reward value of $R = 0.635$ for the local minimum at $x = 3.5$.

We can also see the trend of decreasing σ of the policy, which additionally shows convergence behavior. As we bound the reward change with b, the algorithm needs more iterations to converge. For initial mean values smaller than two and higher than eight the algorithm needs more iterations to converge to an optimal value due to bounding the reward change.

5 Future Work

We have proposed different approaches in this work. Therefore we will mention for each part some suggestions.

5.1 Chance Constrained Iterative Linear Quadratic Gaussian

It would be interesting to test the system on more high dimensional systems in order to see if the Chance Constrained iLQG policies still keep the increased performance. In this work we have demonstrated the algorithm on the pendulum, which is an easy problem to solve. To show the general performance improvement in addition to the cartpole problem, would be interesting.
We have presented the Chance Constrained iLQG method, where we have considered deterministic controllers. One possible new method, which could further be developed is the extension to stochastic controllers. Thus, incorporating Chance Constraints into the Maximum Entropy iLQG framework is a possibility to obtain stochastic policies, which can explore the feasible action space, which is shaped appropriately by the Chance Constraints.
Furthermore, a consideration of optimizing a feedback gain which considers the constraints in addition is an approach, which would uncouple from the iLQG framework, as there would be no need of iLQG's feedback gain. A further improvement could be achieved by this, since the feedback gains get conservative, when the algorithm starts to converge.
Finally, during optimizing the hyperparameters we have recognized that the probability value does not have high impact on the results. When using Boole's inequality to relax the constraints, we obtain very conservative approximations. Thus, considering risk allocation for each time step is a suggestion.

5.2 Risk Measures for Reinforcement Learning

During our literature review on Risk Measures we have recognized similarities to risk sensitive control and risk aware Reinforcement Learning. We tried to give a brief summary on connections to control approaches. Investigating more similarities and parallels would be helpful to fill the gap between existing risk measures and to explore potential new methods.
We also have seen that using risk measures in control have a relation to robust control properties. Investigating this area in detail could also lead to closing the gap and exploring potential new methods.

5.3 Regularized Stochastic Optimization with Chance Constraints

We have presented a new point of view for stochastic optimization, where we regularized the reward change in each iteration using Chance Constraints and interpreted this regularization as risk measure. This method was inspired by the parallels of Relative Entropy Policy Search to the entropic Risk Measure.
However, in this approach we transformed the probability constraints using the Unscented Transformation. The optimization problem was solved by using a numerical solver ($slsqp$ from $scipy$). Due to lack of time, an alternative way of solving the optimization problem could not be considered. Therefore, a deep insight in developing new methods to solve this problem is a possible future work. An example could be to look if common weighted updates for the policy could be applied for this case.
Evaluating the method first on harder tasks with higher dimensions and then possibly extend it for dynamic systems are the next steps to be taken for evaluating the overall performance of the method.

6 Conclusion

This work consists of two propositions for new approaches. First, we introduced an optimization problem after the backward pass of iLQG, in order to consider Chance Constraints for linearized dynamics. We have seen that between iLQG and the new approach nearly no performance difference exists for the pendulum task. For the cartpole task clear differences could be seen. Thus, the method shows the trend of performance improvement by considering constraints during optimization procedure. We have seen that the linearization error decreased by considering Chance Constraints during the optimization procedure, which lead to keep local validity of the learned models.
The second contribution considered Stochastic Optimization with Chance Constraints, where we interpreted the Chance Constraints as risk measures on the reward and thus, bounded the reward change in each iteration to obtain a regularization. This development was motivated by the connection of Relative Entropy Policy Search and the entropic risk measure. We have tested the method on two one-dimensional functions and showed the convergence behavior. However, further work is needed, to investigate this method more in detail.
In the introduction, we first have given an insight into the motivation and the preliminaries for the rest of our work.
In chapter two we have given an overview on foundations of optimization of static and dynamic systems, as well as a detailed introduction to Chance Constraints. We further briefly explained Unscented Transformation.
In section three we have shown our new approach. Therefore we first gave an overview on related work and then went on in order to first give an intuition by providing an algorithmic flow and then introduce the mathematical derivation of reformulating the underlying optimization problem into a quadratic program. We mentioned that we solved the optimization problem by using the numerical solver CasADi [33]. We tested our system on the pendulum and the cartpole task, where we have seen that no clear improvement for the pendulum task was given, whereas we could clearly see an improvement in performance for the cartpole task. However, we could clearly see the trend of smaller linearization errors.
In the second main part of this work we have dealt with Regularized Stochastic Optimization with Chance Constraints. In this part we first gave an overview on policy search algorithms and risk measures. We then suggested a connection of Relative Entropy Policy Search to entropic risk measure, which motivated the new approach. The problem of solving Chance Constraints in the proposed optimization problem was solved by using Unscented Transformation, to fit the reward distribution as Gaussian. In two small experiments we have shown that the proposed method manages to find a minimum value of a one dimensional convex and non-convex function. In the last section we proposed different points for future work.

Bibliography

[1] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and stabilization of complex behaviors through online trajectory optimization,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4906–4913, IEEE, 2012.

[2] S. Levine and V. Koltun, “Guided policy search,” in *International Conference on Machine Learning*, pp. 1–9, 2013.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] H. Abdulsamad, “Masterthesis: Stochastic optimal control with linearized dynamics, ias tu darmstadt,” 2016.

[5] O. von Stryk, “Optimierung statischer und dynamischer systeme,” April 2017.

[6] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[7] A. Prékopa, *Stochastic programming*, vol. 324. Springer Science & Business Media, 2013.

[8] W. H. Fleming *et al.*, “Risk sensitive stochastic control and differential games,” *Communications in Information & systems*, vol. 6, no. 3, pp. 161–177, 2006.

[9] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[10] H. Föllmer and A. Schied, “Convex and coherent risk measures,” *preprint, Humboldt University*, 2008.

[11] O. Von Stryk and R. Bulirsch, “Direct and indirect methods for trajectory optimization,” *Annals of operations research*, vol. 37, no. 1, pp. 357–373, 1992.

[12] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.

[13] A. V. Rao, “A survey of numerical methods for optimal control,” *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[14] J. Löfberg, *Linear model predictive control: Stability and robustness*. PhD thesis, Linköping University Electronic Press, 2001.

[15] M. Diehl, H. J. Ferreau, and N. Haverbeke, “Efficient numerical methods for nonlinear mpc and moving horizon estimation,” in *Nonlinear model predictive control*, pp. 391–417, Springer, 2009.

[16] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, “A tutorial on the cross-entropy method,” *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[17] A. Nemirovski and A. Shapiro, “Convex approximations of chance constrained programs,” *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 969–996, 2006.

[18] A. Nemirovski and A. Shapiro, “Scenario approximations of chance constraints,” in *Probabilistic and randomized methods for design under uncertainty*, pp. 3–47, Springer, 2006.

[19] A. Nemirovski, “On safe tractable approximations of chance constraints,” *European Journal of Operational Research*, vol. 219, no. 3, pp. 707–718, 2012.

[20] S. Jha and V. Raman, “On optimal control of stochastic linear hybrid systems,” in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 69–84, Springer, 2016.

[21] D. Van Hessem and O. Bosgra, “A conic reformulation of model predictive control including bounded and stochastic disturbances under state and input constraints,” in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 4, pp. 4643–4648, IEEE, 2002.

[22] M. P. Vitus and C. J. Tomlin, “On feedback design and risk allocation in chance constrained control,” in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 734–739, IEEE, 2011.

[23] G. Kurz, M. Dolgov, and U. D. Hanebeck, “Progressive closed-loop chance-constrained control,” in *Information Fusion (FUSION), 2016 19th International Conference on*, pp. 67–72, IEEE, 2016.

[24] E. A. Wan and R. Van Der Merwe, “The unscented kalman filter for nonlinear estimation,” in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pp. 153–158, Ieee, 2000.

[25] S. J. Julier, “The scaled unscented transformation,” in *American Control Conference, 2002. Proceedings of the 2002*, vol. 6, pp. 4555–4559, IEEE, 2002.

[26] B. Plancher, Z. Manchester, and S. Kuindersma, “Constrained unscented dynamic programming,” in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 5674–5680, IEEE, 2017.

[27] D. H. Jacobson and D. Q. Mayne, “Differential dynamic programming,” 1970.

[28] R. Bellman, “Dynamic programming,” *Princeton, USA: Princeton University Press*, vol. 1, no. 2, p. 3, 1957.

[29] E. Todorov and W. Li, “A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems,” in *American Control Conference, 2005. Proceedings of the 2005*, pp. 300–306, IEEE, 2005.

[30] Y. Tassa, N. Mansard, and E. Todorov, “Control-limited differential dynamic programming,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 1168–1175, IEEE, 2014.

[31] M. Lorenzen, F. Dabbene, R. Tempo, and F. Allgöwer, “Constraint-tightening and stability in stochastic model predictive control,” *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3165–3177, 2017.

[32] “Optimal control with casadi.” `http://casadi.sourceforge.net/users_guide/html/node8.html`. Accessed: 2018-11-26.

[33] J. Andersson, *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

[34] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[36] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” *GitHub repository*, 2016.

[37] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033, IEEE, 2012.

[38] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, “A survey on policy search for robotics,” *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[39] J. Peters, K. Mülling, and Y. Altun, “Relative entropy policy search.,” in *AAAI*, pp. 1607–1612, Atlanta, 2010.

[40] E. Delage and S. Mannor, “Percentile optimization for markov decision processes with parameter uncertainty,” *Operations research*, vol. 58, no. 1, pp. 203–213, 2010.

[41] D. Jacobson, “Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games,” *IEEE Transactions on Automatic control*, vol. 18, no. 2, pp. 124–131, 1973.

[42] P. Dupuis, M. R. James, and I. Petersen, “Robust properties of risk-sensitive control,” *Mathematics of Control, Signals and Systems*, vol. 13, no. 4, pp. 318–332, 2000.

[43] K. Glover and J. C. Doyle, “State-space formulae for all stabilizing controllers that satisfy an hinf-norm bound and relations to relations to risk sensitivity,” *Systems & control letters*, vol. 11, no. 3, pp. 167–172, 1988.