Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Learning and Intelligent Systems
Prof. Dr. rer. nat. Marc Toussaint

Master Thesis

Constrained Sampling

A Study on Methods for Sampling from Constraint Manifolds

by
Philip Oedi
Matriculation Number: 340896

Examiner: Prof. Dr. rer. nat. Marc Toussaint
Prof. Dr.-Ing. Jörg Krüger

Supervisor: Ph.D. Jung-Su Ha
M.Sc. Joaquim Ortiz-Haro

February 22, 2022

Acknowledgements

First of all, I would like to thank Prof. Dr. Marc Toussaint at the LIS (Learning and Intelligent Systems) for allowing me to carry out state-of-the-art research in this field and being my primary examiner. I also would like to thank Prof. Dr. Jörg Krüger from IAT (Industrielle Automatisierungstechnik) for acting as my second examiner.

Special thanks to my supervisors Jung-Su Ha and Quim Ortiz-Haro for their guidance. It was a great pleasure to discuss the approaches and results of this project. I am very happy that you pointed me in the right direction and gave critical feedback. This project was an incredible learning journey and taught me a new approach to problem-solving.

Furthermore, I would like to thank Ilaria Cicchetti-Nilsson for her support on any organizational and administrative matters, which made me feel very welcome at LIS.

Statutory Declaration

Hereby, I declare that I have developed and written this research completely by myself and that I have not used sources or means without declaration in the text. Any external thought, content, media, or literal quotation is explicitly marked and attributed to its respective owner or author. As of the date of submission, this piece of document and its content have not been submitted anywhere else but to my supervisors.

Berlin February 22, 2022

.....
(Signature [Philip Oedi])

Abstract

Sampling from constraint manifolds is a challenging problem. In practice, this problem appears in motion planning where next to generating the set of samples in a reasonable time, it is further required that its distribution shows high uniformity and covers the manifold well. A standard approach like rejection-sampling is generally not applicable, and the sampling involves solving an optimization problem.

In this thesis, various methods for sampling from constraint manifolds are studied. These methods are i.i.d.-Biased-Sampler, RRT-on-Tangent, and Grid-Walk-on-Tangent. i.i.d.-Biased-Sampler is a method that combines a uniform sampler and biased optimization to project onto the manifold. It generally covers the feasible manifold well. On the other hand, RRT-on-Tangent and Grid-Walk-on-Tangent are Markov-Chain-like samplers that can be considered walking on the tangent space of the manifold. They generally have good uniformity and need less computational resources. Furthermore, we propose the Composite-Sampler that combines the properties of i.i.d.-like and Markov-Chain-like samplers by using i.i.d.-Biased-Sampler to generate the seeds for multiple Markov-Chains. Extensions like filter operation, rejection-sampling, and bandit-sampler are presented to improve uniformity and coverage further.

The algorithms are first applied to a 1-d manifold embedded in 2-d space, a 2-d manifold in 3-d space (sphere and disconnected sphere). Robotics experiments like sampling goal states of a robot arm are then conducted. Uniformity, coverage, and computational efficiency are evaluated. The 1-d and 2-d experiment results show that the shape of the manifold and where it is situated in the configuration space affects the sampling distribution. i.i.d.-Biased-Sampler generates a very non-uniform sampling distribution when the manifold lies in the corner of the configuration space. Markov-Chain-like samplers struggle to cover the manifold well when it consist of disconnected components. Furthermore, it is demonstrated that Markov-Chain-like samplers are considerably more computationally efficient than i.i.d.-Biased-Sampler. From the sphere experiments, it is concluded that the composite sampler can overcome the above challenges. The robotics experiments bring forth that choosing the proper parameters for the composite sampler poses a complex problem and brings a significant advantage of i.i.d.-Biased-Sampler, its ease of use and robustness.

Zusammenfassung

Das Sampling aus Constraint-Mannigfaltigkeiten ist ein anspruchsvolles Problem. In der Praxis tritt dieses Problem bei der Bewegungsplanung auf, wo neben der Generierung der Stichprobenmenge in angemessener Zeit auch eine hohe Uniformität der Verteilung und eine gute Abdeckung der Mannigfaltigkeit erforderlich ist. Ein Standardansatz wie das Rejection-Sampling ist im Allgemeinen nicht anwendbar. Das Sampling wird weiter schwert da es die Lösung eines Optimierungsproblems erfordert.

In dieser Arbeit werden verschiedene Methoden für das Sampling von Constraint Mannigfaltigkeiten untersucht. Diese Methoden sind i.i.d.-Biased-Sampler, RRT-on-Tangent, und Grid-Walk-on-Tangent. Der i.i.d.-Biased-Sampler ist eine Methode, die einen uniformen Sampler und eine Biased-Optimierung zur Projektion auf die Mannigfaltigkeit kombiniert. Sie deckt die Mannigfaltigkeit im Allgemeinen gut ab. Andererseits sind RRT-on-Tangent und Grid-Walk-on-Tangent Markov-Chain-ähnliche Sampler, die als Wandern auf dem Tangentenraum der Mannigfaltigkeit betrachtet werden können. Sie weisen im Allgemeinen eine gute Uniformität auf und benötigen weniger Rechenressourcen. Darüber hinaus schlagen wir den Composite-Sampler vor, der die Eigenschaften von i.i.d.-ähnlichen und Markov-Ketten-ähnlichen Samplern kombiniert, indem er i.i.d.-Biased-Sampler verwendet, um Startpunkte für mehrere Markov-Ketten-ähnliche Sampler zu erzeugen. Erweiterungen wie Filteroperationen, Rejection-Sampling und Bandit-Sampler werden vorgestellt, um die Uniformität und Abdeckung weiter zu verbessern.

Die Algorithmen werden zunächst auf eine 1-d-Mannigfaltigkeit, die in einen 2-d-Raum eingebettet ist, und eine 2-d-Mannigfaltigkeit im 3-d-Raum (Kugel und unzusammenhängende Kugel) angewendet. Anschließend werden Robotikexperimente wie das Sampling von Zielzuständen eines Roboterarms durchgeführt. Uniformität, Abdeckung und Berechnungseffizienz werden bewertet. Die Ergebnisse der 1-d und 2-d Experimente zeigen, dass die Form der Mannigfaltigkeit und deren Position im Konfigurationsraum die Stichprobenverteilung beeinflusst. Der i.i.d.-Biased-Sampler erzeugt eine sehr ungleichmäßige Sampling-Verteilung, wenn die Mannigfaltigkeit in der Ecke des Konfigurationsraums liegt. Markov-Ketten-ähnliche Sampler haben Schwierigkeiten, die Mannigfaltigkeit gut abzudecken, wenn sie aus unverbundenen Komponenten besteht. Darüber hinaus wird gezeigt, dass Markov-Ketten-ähnliche Sampler wesentlich rechnerisch effizienter sind als i.i.d.-Biased-Sampler. Aus den 2-d Kugelexperimenten wird geschlossen, dass der Composite Sampler die oben genannten Herausforderungen überwinden kann. Die Robotik-Experimente zeigen, dass die Wahl der richtigen Parameter für den Composite Sampler ein komplexes Problem darstellt und einen wesentlichen Vorteil des i.i.d.-Biased-Samplers, nämlich, dass er wesentlich einfacher zu nutzen und robuster ist.

Contents

List of Tables

List of Figures

List of Algorithms

Acronyms

CBiRRT2	Constrained BiDirectional RRT
CDF	Cumulative Distribution Function
DOF	Degree of Freedom
MCMC	Markov Chain Monte Carlo
PDF	Probability Density Function
PMF	Probability Mass Function
PRM	Probabilistic Roadmaps
RRT	Rapidly-Exploring Random Tree
MH	Metropolis-Hastings-Algorithm
ECDF	Estimated Cumulative Density Function
std	Standard Deviation
GW	Grid-Walk
i.i.d	Independent and identically distributed

Symbols

$\mathcal{X}$	Feasible Space
K	Configuration Space
$g(x)$	Inequality Constraint
$h(x)$	Equality Constraint
$d(a, b)$	Distance metric for a and b
b_{low}	Lower bounds of the configuration space
b_{up}	Upper bounds of the configuration space
$P(x)$	Probability Mass Function
$F(x)$	Cumulative Distribution Function
$\text{Var}(X)$	Variance
$H(X)$	Entropy
$\mathbb{E}(X)$	Expectation or Expected Value
$Ber(p)$	Bernoulli Distribution
$f(x)$	Probability Density Function
$U(a, b)$	Uniform Distribution
$\mathbf{Cov}(X)$	Covariance Matrix
$\boldsymbol{P}$	Transition Matrix of a Markov Chain
$\boldsymbol{\pi}$	Stationary Distribution of a Markov Chain
$\hat{f}(x)$	Kernel Density Estimate
h	Bandwidth
σ	Standard Deviation
$\hat{H}$	Estimate of the Entropy
$\text{ADTS}(R, S)$	Average Distance To Sample
R	Uniform Reference Dataset
$q(x)$	Proposal Distribution
$p(x)$	Target Distribution
w_{GW}	Neighborhood width in Grid-Walk
w_{RRT}	Neighborhood/Local bounds in RRT
α	Stepsize in RRT
x_{biased}	Bias in Biased Optimization
β	Distance threshold in Filter
N_{iter}	Set of iterations
AIPS	Average Iterations per Sample
n_{samples}	Number of samples per experiment run
n_{runs}	Number of samples runs per experiment
n_{local}	Number of samples generated in the local sampling stage of a composite sampler
n_{global}	Number of samples generated in the global sampling stage of a composite sampler
std	Standard Deviation

1 Introduction

Although having a long history in the natural sciences and engineering, it was just over the last decades that artificial intelligence has gotten attention as never before. This is driven by increased availability and performance of computing power, which Moore studied and manifested in Moore's law [Moo65]. Furthermore, vast amounts of globally connected devices respond to each other and create large amounts of data pushing science and industry to develop more methods to learn and interpret these. Industry 4.0 and Digitalization, Artificial Intelligence, and Data Science are terms that coin the agenda of current decision-makers. Especially manufacturing companies are trying to find an edge by investing in autonomous systems and integrating mathematical models to improve their workflows and cut costs. Robots are a significant aspect of this and are used to accomplish various tasks. The field of robotics itself is very complex as it is required to integrate different methods from computer vision, optimization, control theory, and hardware components. From a scientific standpoint, robotics is furthermore interesting as the development and study of algorithms meant to govern the behavior of a robot give insights into how intelligent behavior and decisions emerge.

Tasks like navigating a robot over a factory floor, finding the optimal trajectory of a manipulator to move an object from one place to another, or legged locomotion are known as path or motion planning. These groups of problems have in common that their solution is to find a sequence of valid configurations. In the case of the factory floor navigation problem, this would correspond to a sequence of positions on the factory floor that are not occupied by any objects. Although these problems might seem different at first sight, they have the same mathematical representation. The underlying problem is that the set of valid configurations, also called the feasible set, is defined by a set of inequality and equality constraints and is therefore known as a manifold. The solution is then to generate a diverse set of samples from this feasible manifold.

Of particular interest in this thesis is the sampling of goal states in motion planning. The goal states lie on such a constraint manifold. Generating a diverse set of goal states in a short amount of time is therefore an essential aspect of motion planning.

1.1 Related Work

This section will briefly introduce already developed methods for solving the problem of generating a set of samples $S \subset \mathcal{X}$ with

$$\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0,\ h(x) = 0\} \tag{1.1}$$

such that S is diverse and covers $\mathcal{X}$ well. The feasible set $\mathcal{X}$ is defined by the equality constraints $h : \mathbb{R}^n \to \mathbb{R}^l$ and inequality constraints $g : \mathbb{R}^n \to \mathbb{R}^m$.

Generating such S is not straightforward because $\mathcal{X}$ is embedded on a manifold of lower dimension, whose shape is unknown and potentially consists of multiple disconnected components. Since the manifold does not have any volume, standard approaches like rejection sampling cannot be applied to this problem. Some methods that have been developed to solve these are the following.

Kingston et al., 2018, describe a variety of methods to sample form constraint manifolds such as relaxation, projection, the use of tangent spaces, an atlas, and reparameterization [KMK18]. AtlasRRT, for example, is an algorithm that combines Rapidly-Exploring Random Tree (RRT), tangent spaces, and projection in order to simultaneously explore and build up an atlas of the feasible configurations [JP11]. Paul Breiding and Orlando Marigliano further proposed a method to sample from manifolds defined by polynomial constraints [BM20]. A linear subspace of the configuration space is sampled, and feasible samples are generated by finding the subspace and the manifolds intersection. A probabilistically complete algorithm designed explicitly for pose constrained manipulation tasks is the Constrained BiDirectional RRT (CBiRRT2). As the name suggests, CBiRRT2 applies bidirectional RRT's (two RRT's are grown simultaneously with their roots at the starting and goal position to merge the trees and find valid configurations) and ensures that pose constraints are being fulfilled by projecting on the manifold [BSK11]. Not specifically focusing on path planning but a more general study and development of algorithms that sample distributions defined over manifolds has been done by Brubaker et al., 2012. By using projection methods Hamiltonian Monte Carlo, a Markov Chain Monte Carlo (MCMC), has been extended, and its convergence to target distribution proved [BSU12]. Another approach, Deep Generative Constraint Sampling (DGCS), is propsed by Ortiz-Haro et. al. DGCS uses a deep generative model to generate points near the manifold [OHHDT21].

1.2 Research Purpose

Various algorithms can be used for sampling from constraint manifolds. Trade-offs are made in terms of uniformity, coverage and efficiency. Therefore, it is interesting to evaluate the algorithms in a standardized setting and find the causes for the potential lack of coverage or uniformity. This thesis investigates the sampling behavior of i.i.d.-Biased-Sampler, RRT-on-Tangent and Grid-Walk-on-Tangent on a qualitative and quantitative level. It proposes extensions to these algorithms that other researchers can use to design and refine algorithms for their specific problems and application domains. The primary research purposes of this thesis are formulated as the following:

1. Find a set of measures to compare sampling algorithms for uniformity and coverage.
2. Define a set of experiments to study certain aspects of the algorithms sampling behavior that can be evaluated visually and quantitatively.
3. Study phenomena such as local non-uniformity or not-covered subspaces depending on the algorithm.
4. Find causes for the above-discovered phenomena.
5. Propose new algorithms that extend or combine existing approaches.

1.3 Structure

The previous section 1.1 of this document has given a broad overview of the type of problems and algorithms studied, their scientific and industrial relevance, the state-of-the-art approaches, drawbacks, and still missing aspects to be further studied. A description of the purpose, possible outcomes and questions answered with this thesis were given then (see 1.2). The overall steps taken to reach these outcomes are very much reflected in the structure of this document. A solid mathematical understanding of the applied methods is key to the design and evaluation of algorithms. The following section 3 is therefore devoted to introducing constrained spaces, probability distributions, sampling, and optimization methods. In section 4, a framework for the evaluation of the algorithms for coverage, uniformity, and computational performance is then introduced, along with the design of experiments/simulations to be conducted. The outcome of the experiments is then described and visualized in section 5. The outcomes are then thoroughly discussed, interpreted, and relevant conclusions made. Eventually, a critical perspective of this thesis will be taken, shortcomings assessed, and a roadmap of subsequent research outlined.

2 Background

Various methods are used in this thesis, an understanding of which is necessary to draw meaningful conclusions and understand the narrative of this document. Therefore, the following pages are devoted to first establishing a common understanding of the studied problem and a notation that this document will follow. Such as what a constrained space (2.1), a manifold (2.1.1), and a tangent space are (2.1.2), how these are mathematically realized, and how they relate to robotic tasks such as motion planning.

Since the goal is to generate a uniform sets of samples from these spaces, basic concepts of probability theory are introduced. The general probability distributions (2.2) and in specific the uniform distribution (2.2.3) and its properties are discussed. Kernel Density Estimation will further be discussed (2.2.7) as a means to approximating arbitrary probability distributions from sampled data (2.2.5).

Next, in section 2.3 sampling methods will be discussed. Rejection Sampling, MCMC, and RRT are part of this section as they make up a core aspect of the studied algorithms. The principles to which they adhere and potential drawbacks and challenges are stated. Then optimization problems are introduced in 2.4, how they are formulated, and how this relates to the problem of sampling over a constrained space.

The algorithms to be studied in this thesis will eventually be elaborated on in section 3.2. The above-mentioned methods are individual parts of these algorithms. This final section in this chapter will discuss how MCMC, RRT, rejection sampling, and optimization are arranged and interplay to solve the sampling problem. A few mechanisms like filtering, the application of rejection sampling, and the bandit-sampler that are assumed to give potential better properties to the algorithms are introduced as well.

2.1 Constrained Spaces

A space will in this thesis be referred to in its simplest form as a set X of objects. A metric space $M = (X, d)$, in particular, extends a general definition of a space by pairing the set X with a function $d(a, b)$ with $d : M \times M \to \mathbb{R}$ and $a, b \in X$ [Kap01]. The function $d(a, b)$ is also called a distance function that satisfies the following:

1. $d(a, a) = 0$.
2. $d(a, b) > 0, \text{for } a \neq b$.
3. $d(a, b) = d(b, a)$.
4. $d(a, c) \leq d(a, b) + d(b, c)$

Relevant for this thesis is the notion of the euclidean space $(X, d_{\text{euclidean}})$ with $X = \{x \in \mathbb{R}^n\}$, so the real points of dimension n and the euclidean metric $d_{\text{euclidean}}$.

$$d_{\text{euclidean}}(x, y) = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2} \tag{2.1}$$

Furthermore, a euclidean space can be bounded, meaning that there is a subset of X such that all points are within a certain interval. In terms of the distance measure $d(x, y)$ this means that for any two points $x, y \in X$ holds that:

$$d(x, y) < r \tag{2.2}$$

Here r specifies the size of the interval. Therefore a space can have an upper bound b_{up} and a lower bound b_{low} such that:

$$b_{\text{low}} < x < b_{\text{up}} \tag{2.3}$$

The points fulfilling these inequalities make up what is an open set. Imagine a rectangle in $\mathbb{R}^2$. All points inside the rectangle, excluding the edges, are part of the bounded space. The points on the rectangle edges are defined as the boundary set. A distinction is when the boundary set is unified with the points within the box, making this a closed set. The closed and open space might seemingly have minor differences, but handling the space as being closed or open can have a meaningful impact on applying sampling algorithms in constrained spaces, as will be seen later. Unquestionably a space can have either an upper or lower bound, but this thesis has studied problems considering the totally bounded case.

Bounds are a special way of constraining a space. A more general form of constraints is if the functions $g(x)$ and $h(x)$ are introduced such that the following holds:

$$\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0,\ h(x) = 0\} \tag{2.4}$$

The set $\mathcal{X}$ that fulfills these constraints will be referred to as the feasible set or space (it is sometimes also referred to as the valid set). The functions $h : \mathbb{R}^n \to \mathbb{R}^l$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ can have arbitrary forms resulting in the feasible space taking arbitrary shapes like various disconnected components of different sizes and shapes. For example, by introducing a linear inequality constraint, one can crop the space into a simplex shape (assuming that the feasible set is not empty). A phenomenon at the core of this thesis is the presence of equality constraints. A consequence of this is that the feasible space is embedded on a manifold (manifolds will be further introduced in section 2.1.1). The dimension k of the manifold is:

$$k = n - l \tag{2.5}$$

Transferring this notion of space and the feasible set to the problem of motion planning, one can define the configuration space $K = \{x \in \mathbb{R}^n : b_{\text{low}} \leq x \leq b_{\text{up}}\}$. Taking the example of a robot navigating a quadratic factory floor the configuration space can be defined as $x \in K$ with $K = \{x \in \mathbb{R}^2 : 0 \leq x \leq 1\}$, whereas x relates to the relative position between the walls bounding the factory floor. Another example is a robot arm with multiple joints, the configuration space are all combinations of angles by which the joints are rotated and realizes a certain pose. It is important to notice that the general definition of the configuration space includes configurations that might not be physically possible, such as the robot

standing inside a machine that resides on the factory floor or the arm reaching through an object. Taking the robot arm example we can further require that the end effector is positioned in a specific point at space, then the feasible set are poses that satisfy this requirement.

Obstacles and physical capabilities of the mechanical parts can therefore be defined as constraints. The target position of the end effector can furthermore be introduced as a constraint.

2.1.1 Manifolds

Manifolds appear in various domains and are at the heart of the studied problems of this thesis since the feasible set, which is of primary interest, lies on a manifold. It is furthermore a submanifold of the configuration space. Therefore, the above terms will be introduced formally and specific properties of a manifold, such as a manifold being compact/closed or with a boundary.

Generally, we consider a manifold a topological space that locally looks euclidean. A simple example is a curve that locally looks like $\mathbb{R}$ or a surface that locally looks like $\mathbb{R}^2$. A manifold is then the extension of this concept to arbitrary dimension m. Formally we state that an m-manifold $M \subset \mathbb{R}^n$ for which a local homeomorphism to $\mathbb{R}^m$ exists [Eis20].

A local homeomorphism is the existence of a bijective map $\xi : \ x \in \mathbb{R}^m \rightarrow y \in \mathbb{R}^n$

Some examples of manifolds are the sphere or torus. These are furthermore closed or compact manifolds. Being compact or closed is a property of a manifold. A manifold that is not compact then implies that it has a boundary. The space of all points on the bottom half of a sphere is not a compact manifold and has a boundary at its equator.

2.1.2 Tangent Space

Suppose there is a Manifold M of dimension m in a configuration space $\mathbb{R}^n$ that is implicitly defined by equality constraints $h(x)$. The tangent space $T_p(M)$ at a point $p \in M$ is then a linear subspace of $\mathbb{R}^n$ that approximates $h(p)$.

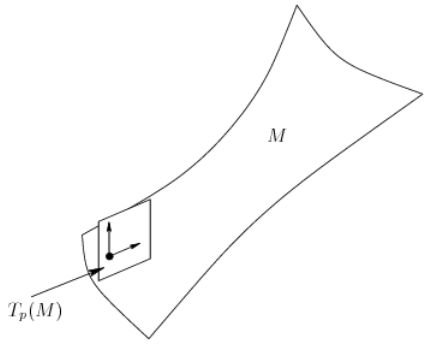

Figure 2.1: Example of a tangent space [Lav06]

When considering direction vectors $v \in \mathbb{R}^n$ the tangent space can further be defined as all v that satisfy the directional derivative:

$$\nabla_v h\Big|_p = \sum_{i=1}^{n} v_i \frac{\partial h}{\partial x_i}\Big|_p \tag{2.6}$$

For above equation further holds that $\nabla_v h\Big|_p = \nabla h$ where ∇h is the gradient of h [Lav06]. Taking into account that the gradient of h at p represents the direction normal to M at p, then v are all direction vectors that are orthogonal to the normal.

As $T_p(M)$ is embedded in $\mathbb{R}^m$ a basis in $\mathbb{R}^m$ can be found and we denote the coordinates in the tangent space as $u \in \mathbb{R}^m$. We want to find $\psi(u) : \mathbb{R}^m \to \mathbb{R}^n$ that maps from the coordinates in the tangent space to the ambient space. We are specifically interested in an orthonormal basis $\boldsymbol{\Theta} \in \mathbb{R}^{m,n}$ of this tangent space that we can use to define the mapping,

$$x = p + \psi(u) \tag{2.7}$$

with

$$\psi(u) = \boldsymbol{\Theta} u \tag{2.8}$$

Following relations between basis and the jacobian of $h(x)$ assure orthonormality of [JP11].

$$\begin{bmatrix} \boldsymbol{J} \\ \boldsymbol{\Theta^T} \end{bmatrix} \boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{I} \end{bmatrix} \tag{2.9}$$

Finding a $\boldsymbol{\Theta}$ that satisfies above can be done using the LQ-factorization of $\boldsymbol{J}$ as has been shown by Rheinboldt [Rhe96].

2.2 Probability Distributions

This thesis aims to generate a set of samples $S \subset \mathcal{X}$. These samples are then analyzed for how they are distributed over the feasible set. In the following section 2.2, a formal definition of a probability distribution is therefore given, and properties like expectation, variance, and entropy are discussed, as these are relevant for the evaluation of the studied algorithms. Special attention will be given to the uniform distribution in section 2.2.3. The uniform distribution reflects the optimal distribution the sample set S can take and should be taken as a reference. A non-parametric approach to approximating arbitrary probability distributions, kernel density estimation (KDE), will be shown and how this, in turn, can be applied to estimating the probability distributions differential entropy.

2.2.1 Discrete Probability Distributions

In the real world, most phenomena are not deterministic but have some underlying notion of being random, and observing these phenomena can be seen as running experiments. It is, for example, not certain which face of dice will show on top. The same applies to the output of a sensor that might have some noise associated with it. The sensor's output is also not

deterministic but follows some random pattern. The game of rolling dice is an experiment, and its sample space contains all unique outcomes (also known as events) of the experiment. Therefore, the sample space would be the faces of the dice shown on top. This can be formalized such that the outcome of the experiment takes on a countable number of distinct values and can be considered a discrete random variable $X = \{x : x \in \{1, 2, 3, 4, 5, 6\}\}$. A probability distribution $P(X)$ is then the function $P : X \to \mathbb{R}$ maps the value of a random variable to the probability of realizing this value when drawing a sample from the distribution.

In the case of a discrete and finite sample space, such as rolling a dice, $P(X)$ is called the probability mass function (PMF) that assigns each outcome a probability value. These probabilities adhere to the following conditions, they are non-negative and less than 1, as well as the sum of the probabilities of all outcomes is 1:

1. $0 \leq P(X = x_i) \leq 1$
2. $\sum_{i=1}^{n} P(X = x_i)) = 1$

The cumulative distribution function (CDF) $F(a)$ of a random variable can be further defined as the probability that the random variable takes values less than a.

$$F(a) = \sum_{x_i \leq a} P(x_i) \tag{2.10}$$

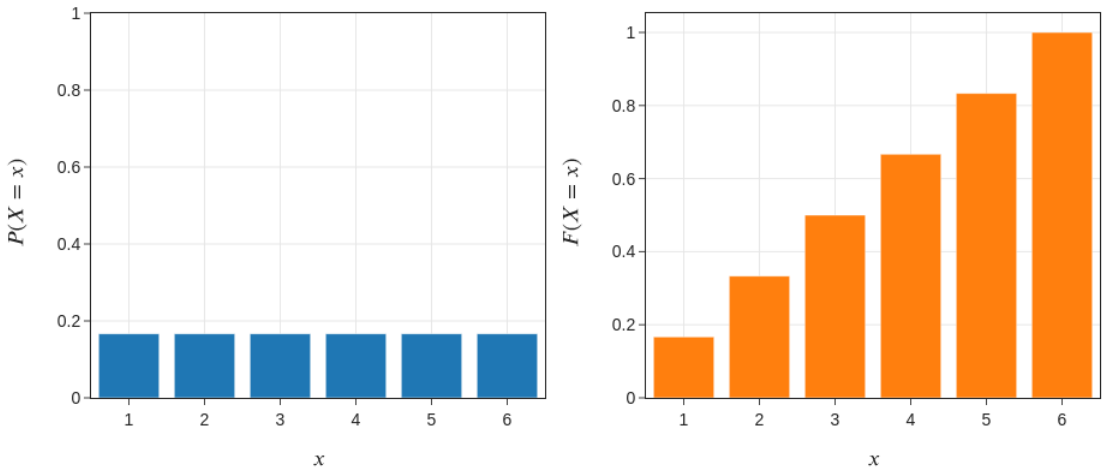

Figure 2.2: PMF and CDF of a dice roll

As long as the probability density and probability mass function satisfies the above-stated conditions, they can realize arbitrary shapes. The visual inspection of the PDF and CDF can give valuable insights into how even or uneven events occur. From the PMF of the dice roll (figure 2.2), one can see that all events are equally likely. Looking at the probability density function (PDF), the continuous analogue to the PMF, of a normal variable (figure 2.4), it

can be seen that the probability varies, and samples are likely to cluster at the middle [Ros97].

Properties

Besides the PMF and the visual interpretation of its shape, two major characteristics carry much information about a distribution. These are expectation and variance Var. The expectation or expected value $\mathbb{E}(X)$ of a discrete random variable X is defined as:

$$\mathbb{E}(X) = \sum_{i=1}^{n} x_i P(x_i) \tag{2.11}$$

The variance $\mathrm{Var}(X)$ of a random variable is closely related to the expectation. It describes the quadratic deviation from the expected value $\mathbb{E}(X)$ and can measure whether a distribution is spiked or widespread. The variance is defined as:

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \tag{2.12}$$

Another quantity that is defined for all probability distributions is the entropy H, also called Shannons Information entropy [CT06]. It is a reflection of the information contained in a distribution. Intuitively one can think of entropy as the amount of uncertainty in a random variable. The entropy of a discrete random variable is defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i)\ \log\big(P(x_i)\big) \tag{2.13}$$

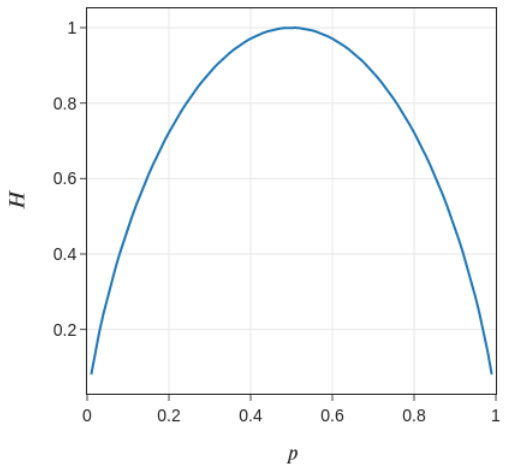

Figure 2.3: Entropy of the Bernoulli distribution for varying p

The entropy's units are bits. Some properties of the entropy are that $0 \leq H \leq 1$. Studying the entropy of a Bernoulli distribution $\mathrm{Ber}(p)$ for example and comparing it for various values

of $0 \leq p \leq 1$ can be seen in figure 2.3. A maximum entropy at $p = 0.5$ can be noticed. This observation goes hand in hand with understanding the entropy as a measure of uncertainty, given a Bernoulli distribution $\text{Ber}(p = 0.5)$ one has the least confidence in predicting the outcome due to the same probabilities for both outcomes. The entropy H can therefore also be used to measure uniformity. When comparing two distributions' entropies, it can be inferred that the distribution of higher entropy is more uniform than the other. The above only applies for discrete random variables, and a detailed discussion of the differential entropy (section 2.2.2) of a continuous random variable in specific the uniform distribution will follow below.

Next to the entropy H, which describes a single distribution, other quantities like mutual information/cross-entropy and relative information are defined. These are other ways to compare how two distributions diverge. These are nonetheless not further elaborated on here but can be read upon in [CT06].

2.2.2 Continuous Probability Distributions

Contrary to discrete probability distributions, it is more difficult to describe the probability distribution of a continuous variable. In this case, the sample space can take on infinite values, and the actual probability of drawing an exact value goes to 0. This resembles a core challenge in the task of sample from a manifold. The probability of drawing a sample using rejection sampling goes to 0 since the manifold does not have any volume. For a continuous random variable, the probability of realizing any events of the set B is defined as [Ros97]:

$$P(X \in B) = \int_B f(x)\ dx \tag{2.14}$$

The probability density function (PDF) $f(x)$ can be interpreted as the derivative of the CDF F at a point x. Analogously to the probability mass function of a continous random variable.

$$1 = P(-\infty \leq x \leq \infty) = \int_{-\infty}^{\infty} f(x)\ dx \tag{2.15}$$

The probability of sampling from within interval (a, b) and the CDF $F(a)$ are the following.

$$P(a \leq x \leq b) = \int_a^b f(x)\ dx \tag{2.16}$$

$$F(a) = P(-\infty \leq x \leq a) = \int_{-\infty}^{b} f(x)\ dx \tag{2.17}$$

This furthermore underlines previously stated facts about the probability of sampling exactly a. It can easily be noted that the above integral evaluates to 0.

$$0 = P(a \leq x \leq a) = \int_a^a f(x)\ dx \tag{2.18}$$

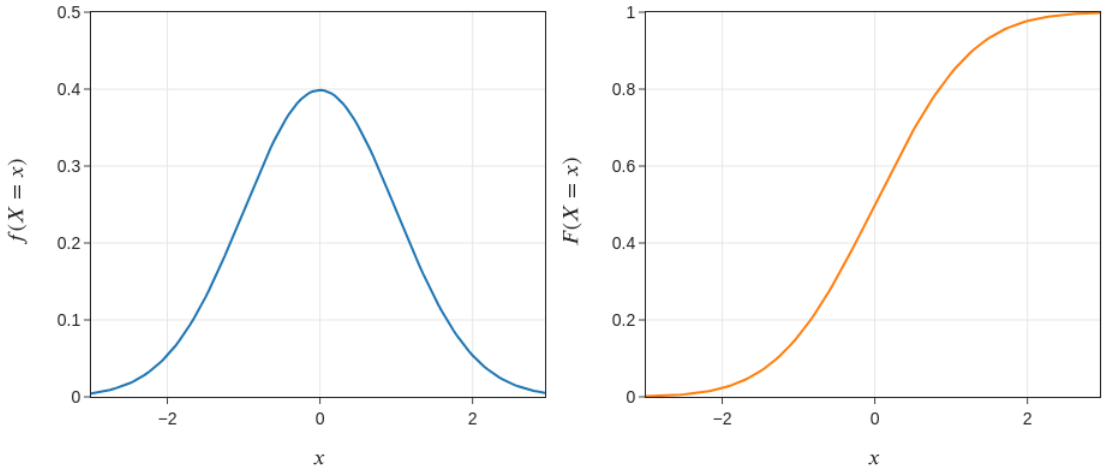

Figure 2.4: PDF and CDF of the normal distribution

Properties

Defining the expectation $\mathbb{E}(X)$ and variance $\mathrm{Var}(X)$ of continuous random variables is analog to discrete random variables. The expectation reflects a weighting of outcomes according to probabilities and is thus defined as the following integral.

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f(x)\ dx \tag{2.19}$$

The entropy of a continuous random variable is referred to as the differential entropy and is:

$$H(X) = -\int_{-\infty}^{\infty} f(x) \log\big(f(x)\big)\ dx \tag{2.20}$$

Essential differences between the discrete and the continuous or differential entropy are the range of values the entropy can take. It has been described before that the entropy of a discrete distribution takes on values between $0 \leq H(X) \leq 1$. On the contrary, the differential entropy may be less than zero, which is the case for a uniform distribution with support over a range smaller than 1 (a detailed discussion follows in 2.2.3).

Mutual information relative information are also defined for continuous random variables but will not be further introduced here but can be found in [CT06].

2.2.3 Uniform Distribution

The probability distribution of particular interest is the uniform distribution $U(a, b)$. Intuitively, it resembles that all values are equally likely to be drawn over an interval between a and b. Its discrete counterpart similarly represents that all elements of the sample set have the same probability. The outcome of rolling unbiased dice follows a discrete uniform distribution.

The uniform distribution is essential to this thesis because it represents the optimal distribution over the feasible set generated by the studied algorithm. The goal of the studied algorithms is a rapid exploration of the feasible set. If the samples generated over the feasible set would follow a distribution other than the uniform distribution would mean that locally some parts of the manifold would be more densely sampled than others.

The uniform distribution is defined by two parameters a and b the lower and upper bound of the support. For a random variable X that is uniformly distributed we can write $X \sim U(a, b)$ [HFC+02]. The PDF of a uniform distribution is constant:

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{if } x \in (a, b) \\ 0, & \text{otherwise} \end{cases}$$

Remembering that the $f(x)$ is the derivative of $F(x)$ we find that $F(x)$ is linear over the support (see figure 2.5).

$$F(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x < b \\ 1, & x \geq b \end{cases}$$

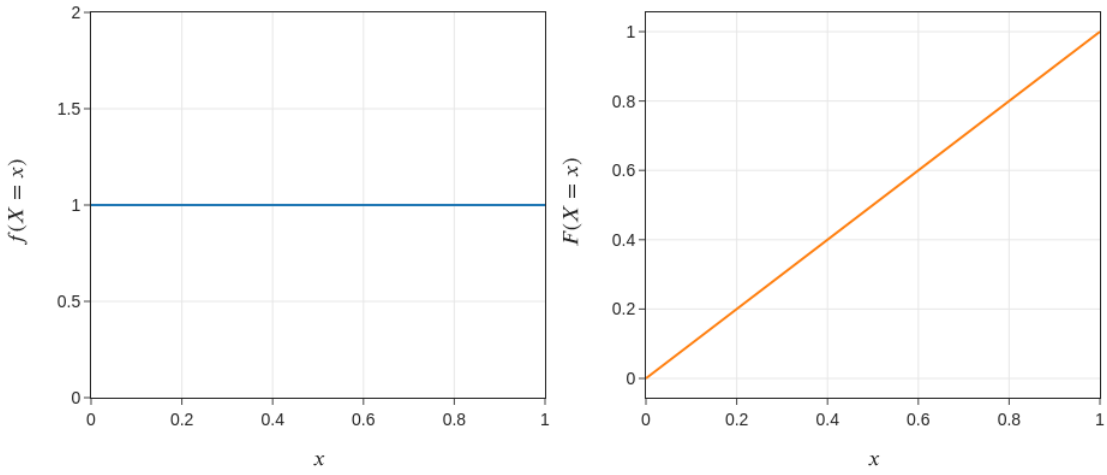

Figure 2.5: PDF and CDF of the uniform distribution

The expectation of a uniform random variable, as well as variance and entropy are a function of the lower and upper bound a and b or the width w of the interval. They can be derived from the standard definition of the expectation as such.

$$\mathbb{E}(x) = \frac{b+a}{2} \tag{2.21}$$

The variance of the uniform distribution is:

$$\mathrm{Var}(x) = \frac{(b-a)^2}{12} = \frac{w^2}{12} \tag{2.22}$$

The entropy of a random variable $X \sim U(a, b)$ is:

$$H(X) = \ln(b-a) = \ln(w) \tag{2.23}$$

Taking a closer look at how the entropy of the uniform distribution is defined, it can be noticed that the entropy can be negative in cases where $w \leq 1$ (see figure 2.6).

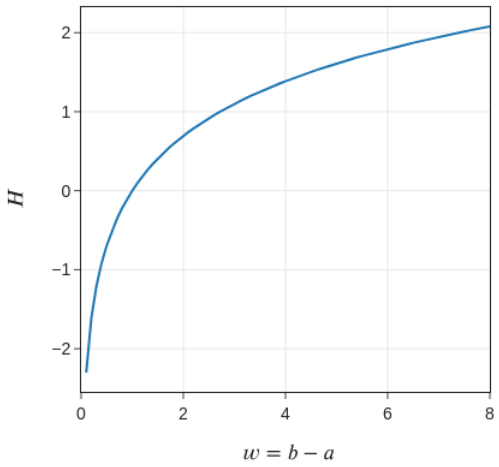

Figure 2.6: Entropy H of the uniform distribution for varying w

2.2.4 Multivariate Probability Distributions

We have only been concerned with univariate probability distributions in the previous sections. This concept can be extended to multiple variables. An event then corresponds to an exact realization of each of the individual random variables. The probability of such an event is defined by the joint probability mass function and density function. The joint probability mass function $P(X)$ for $X \in \mathbb{R}^n$ being a vector of random variables $[X_1, ..., X_n]$ that characterizes the probability of multiple discrete random variables. It can be imagined as querying a table that specifies the probability for all possible combinations of the sample space of the individual random variables.
The continuous counter-part of the joint PMF is the joint PDF (2.25) and the probability as an integral over a region in the sample space.

$$P(X) = P(X_1 = x_1, X_2 = x_2, ..X_n = x_n) \tag{2.24}$$

$$\int f(x_1, x_2, .., x_n) dx_1 \; dx_2 \; ... \; dx_n \tag{2.25}$$

The expectation of a multivariate random variable can be interpreted as its elements' expectations.

$$\mathbb{E}(X) = \begin{bmatrix} \mathbb{E}(X_1) \\ \mathbb{E}(X_2) \\ \vdots \\ \mathbb{E}(X_n) \end{bmatrix} \tag{2.26}$$

There is nonetheless not a single quantity that specifies the variance. The dispersion of multiple random variables is represented as a covariance matrix $\mathbf{Cov}(X)$, which is a square symmetric matrix of dimension $n \times n$.

$$\mathbf{Cov}(X_i, X_j) = \mathbb{E}\big[(X_i - \mathbb{E}(X_i))(X_j - \mathbb{E}(X_j))\big] \tag{2.27}$$

The diagonal elements correspond to the variance of the i-th random variable $\mathbf{Cov}(X_i, X_i) = \mathrm{Var}(X_i)$. The off-diagonal elements then correspond to the covariance between X_i and X_j. The covariance's sign and magnitude can be interpreted as the relation between variable X_i and X_j. If the covariance differs significantly from zero, we can infer that the random variables X_i and X_j are not independent. Figure 2.7 displays the PDF of two multivariate normal distributions in $\mathbb{R}^2$ one with and one without correlation between the two random variables.

$$\mathbf{Cov}(X_i, X_j) = 0 \implies P(X_i, X_j) = P(X_i)P(X_j) \tag{2.28}$$

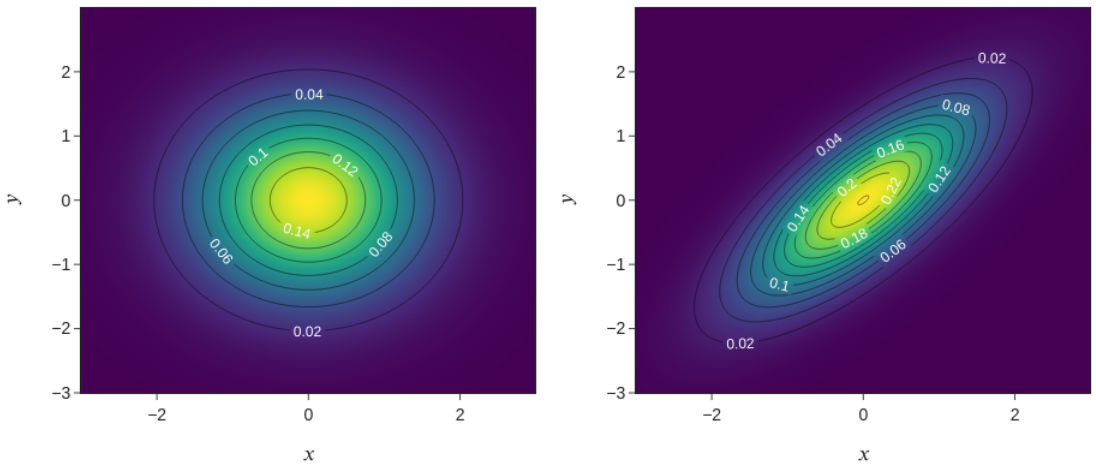

Figure 2.7: PDF of the uncorrelated 2-dimensional Normal distribution (left) and the correlated 2-dimensional Normal (right).

In contrast to variance and expectation, the entropy does not return a vector/matrix, but a scalar, since it only depends on the PMF/PDF that maps to a scalar. The entropy of a multivariate probability distribution is called the joint entropy $H(X) = H(X_1, X_2, ..., X_n)$.

$$H(X_1, ..., X_n) = -\sum_{x_i} ... \sum_{x_n} P(X_1 = x_1, ..., X_n = x_n) \log\big(P(X_1 = x_1, ..., X_n = x_n)\big) \quad (2.29)$$

The joint differential entropy is defined as.

$$H(X_1, ..., X_n) = -\int f(x_1, ..., x_n) \log\big(f(x_1, ..., x_n)\big) dx_1 \ ...dx_n \quad (2.30)$$

2.2.5 Approximation of Parameters

Most of the time, and in this thesis, it is of interest to find the true values of an unknown parameter by sampling the parameter. This works due to a fundamental law in probability theory that makes sampling-based methods so powerful. The strong law of large numbers relates the mean of a sequence of numbers with the expectation of a random variable. Assuming a sequence of samples is drawn independently from the same distribution, this sequence can be denoted as $X_1, X_2, ...X_n$. The strong law of number states that mean over the samples approaches the true mean μ as the number of samples goes to infinity. For a better illustration of this, see figure 2.8 that displays how the $\bar{X}$ of a dice roll evolves with an increasing number of samples.

$$\frac{X_1 + ... + X_n}{n} \rightarrow \mu, \text{for } n \rightarrow \infty \quad (2.31)$$

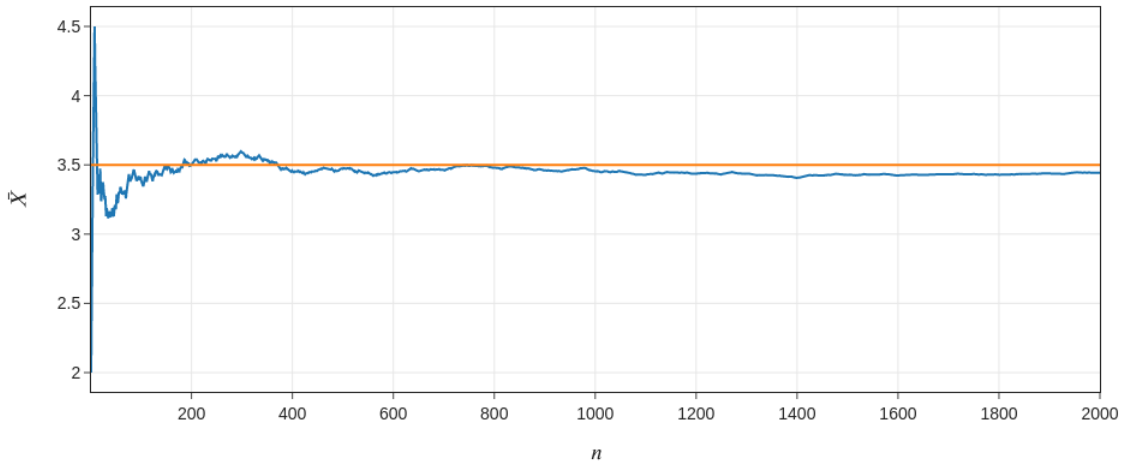

Figure 2.8: Sample mean $\bar{X}$ of the dice roll experiment for increasing sample sizes n. Expectation of the dice roll (orange).

This is very powerful. Knowing this, we can use the sample mean $\bar{X}$ and sample variance $\bar{\text{Var}}$ as an approximation for the true parameters.

$$\bar{X} = \frac{1}{n_{\text{samples}}} \sum x_i \tag{2.32}$$

$$\bar{\text{Var}}(X) = \frac{1}{n_{\text{samples}} - 1} \sum (x_i - \bar{x})^2 \tag{2.33}$$

Being able to make this approximation is critical when it is intractable to analytically solve for the true parameters, which will be the case for the distributions the studied algorithms will create.

Often one does not access a sample set of fixed size. Real-time systems that capture data using sensors regularly have a growing set of samples. One can note from the definitions of the sample mean and variance/covariance that the time complexity of the evaluation of these increases with the size of the sample set. The complexity states a problem for such real-time systems where an approximation of the sample mean and covariance might have to be accessible anytime and not just after the sampling process concludes (if it ever concludes). It is nonetheless possible to overcome this challenge by using an online update of the parameters that updates the current estimation of the mean $\bar{x_n}$ or $\mathbf{Cov}_n$ covariance at timestep n given a new observation x_{n+1} [DH07].

$$\bar{x}_{n+1} = \frac{1}{n-1}(n\bar{x}_n + x_{n+1}) \tag{2.34}$$

Given the updated mean we can then find update the covariance as:

$$\mathbf{Cov}_{n+1} = \frac{n}{1+n}\mathbf{Cov}_n + \frac{n}{(1+n)^2}(x_{n+1} - \bar{x}_{n+1})^T(x_{n+1} - \bar{x}_{n+1}) \tag{2.35}$$

2.2.6 Markov Chains

To study how a value that underlies randomness evolves one usually models it as a sequence of random variables $X_1, X_2, ..., X_n$, often the value X_n is dependent on a previous value X_{n-1} (Markov property). This transition from one state to the next can then be described using probabilities. A sequence of random numbers over which a transitioning probability is defined governs how the sequence evolves is called a stochastic process or a Markov chain [Ros97].

For a discrete sample space, these transition probabilities can be represented using a transition matrix $\boldsymbol{P}$.

$$\boldsymbol{P}_{ij} = P(X_t = x_j | X_{t-1} = x_i) \tag{2.36}$$

The entries $\boldsymbol{P}_{ij}$ specify the probability of transiting from state x_i to x_j. Under the laws of probability, the sum along the rows of Matrix $\boldsymbol{P}$ equal 1. Given an initial probability distribution P_0 over the starting state of the sequence, the probability of being in any particular state at time $t = n$ can be calculated:

$$P(t = n) = P(t = 0)\boldsymbol{P}^n \tag{2.37}$$

A Markov chain itself has a stationary distribution $\boldsymbol{\pi}$. This distribution is defined over the sample space and reflects the proportions of times being in a particular state when evolving

the Markov chain for an infinite amount of times. So given a transition matrix $\boldsymbol{P}$ of an irreducible Markov chain, there exists a stationary distribution $\boldsymbol{\pi}$ which fulfills the following:

$$\boldsymbol{\pi}^T \boldsymbol{P} = \boldsymbol{\pi}^T \tag{2.38}$$

This vector $\boldsymbol{\pi}$ furthermore corresponds to a unique Eigenvector of $\boldsymbol{P}$ that has an Eigenvalue of 1.

The above mechanism can be transferred to continuous state spaces as well. The transition is then characterized by a transition density $p(x, y)$, describing the transition probabilities from x to y ($x, y \in W$, the state space) and the stationary distribution satisfies the following interval [GRS95].

$$\pi(y) = \int_W \pi(x) p(x, y) dx \tag{2.39}$$

Markov chains and their stationary distributions are important to understand. They make up the foundation of a whole class of algorithms MCMC, allowing the sampling from probability distributions where direct sampling is complex. This will further be elaborated on later.

2.2.7 Kernel Density Estimation

Given a set of samples S from an unknown distribution, a task of interest is to find out what kind of distribution this unknown distribution may be. The most apparent approach to this can be choosing a suitable parametric distribution (uniform, normal, exponential, and others) and fitting its parameters (for example, using Maximum Likelihood Estimation). Although many distributions are defined, it is not straightforward to choose any particular distribution. There often might not be any parametric distribution reflecting the true underlying distribution. Just using any parametric distribution would result in a distortion of facts. This is the case in a scenario such as sampling from a constraint manifold. Because the feasible set consists of various disconnected components of different shapes and dimensions, it is impossible to find a parametric representation of the probability distribution. Non-parametric models can be used in such cases, one of which is Kernel Density estimation (KDE) [Han09].
The basic principle of Kernel Density Estimation is the approximation of the probability density at a point x using the density of previously sampled points in the neighborhood of x. A general estimator for a continuous random variable and a set of samples $\{x_1, x_2, .., x_n\}$ can be defined as:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} k\left(\frac{x_i - x}{h}\right) \tag{2.40}$$

k is a kernel function, that satisfies $k(u) = \int_{-\infty}^{\infty} k(u) du = 1$. A variety of kernels is defined (Uniform, Epanechnicov, Gaussian, and more) that govern how the neighborhood is being interpreted or, more precisely, a weighing of the samples according to their impact on the local probability density. A uniform kernel weighs all samples within a certain range surrounding x equally—the weights decay with the distance to x given a Gaussian kernel. A frequent choice for a kernel function is the Epanechincov kernel (see figure 2.9). The Epanechnicov kernel is often called the optimal kernel for favorable properties regarding the asymptotic

mean integrated square error, a measure of the estimation precision (which will not further be discussed here, but details can be found [Han09]).

$$k_{\text{epanechnikov}}(u) = \frac{3}{4}(1 - u^2)\mathbf{1}(|u| \leq 1) \tag{2.41}$$

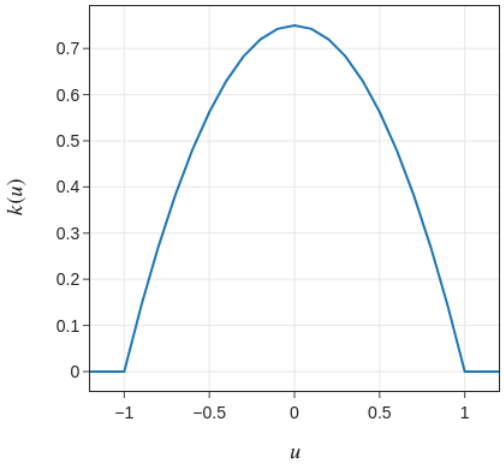

Figure 2.9: $k(u)$ of the Epanechnikov-Kernel

h is the bandwidth that reflects how wide the neighborhood is. $\mathbf{1}(|u| \leq 1)$ is the indicator function that returns 1 if the condition is met and 0 otherwise. A large h might result in a very smooth probability density surface at the cost of not accounting for local variations. On the other hand, choosing h too small might result in a very bumpy surface that also does not reflect the true underlying densities. The choice of the bandwidth is therefore of significant importance (see figure 2.10 for an illustration of the effects of h). Any prior knowledge should be included in the choice of h. In $\mathbb{R}^2$ and $\mathbb{R}^3$ it is also possible to inspect the achieved probability density surfaces. One can then find a bandwidth that makes a good trade-off between the above-explained scenarios using trial-and-error. Furthermore, rules-of-thumb like Silverman's rule propose a value for h given the sample standard deviation σ.

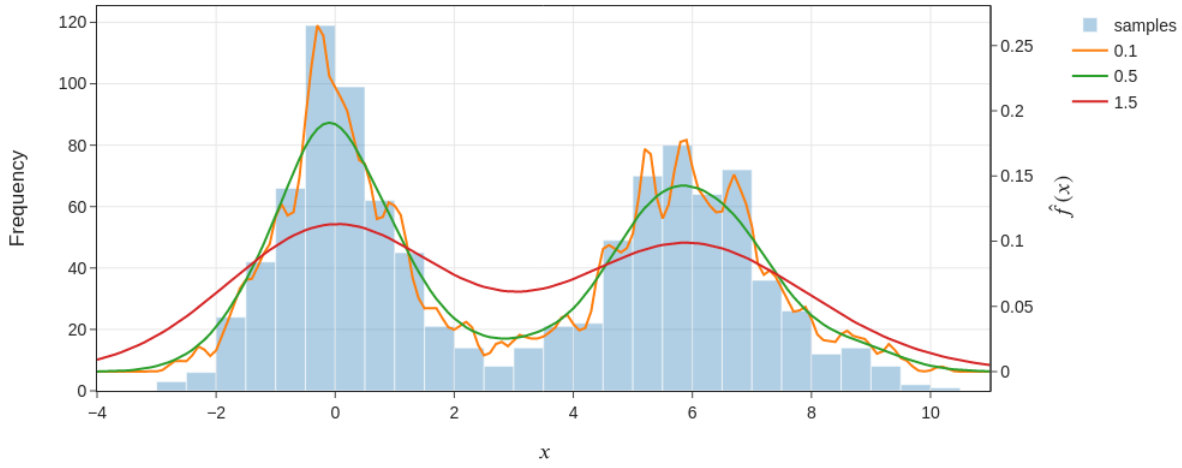

Figure 2.10: Effect of the bandwidth on a bimodal distribution

The kernel density estimator $\hat{f}(x)$ for $x \in \mathbb{R}^q$ can also be found for multivariate sets of samples and is defined as follows.

$$\hat{f}(x) = \frac{1}{n|H|} \sum_{i=1}^{n} K\left(H - 1(x_i - x)\right) \tag{2.42}$$

H is a vector of dimension q which elements represent the bandwidth along the specified axis.

$$|H| = h_1 h_2 ... h_q \tag{2.43}$$

$K(u)$ is a multivariate kernel.

$$K(u) = k(u_1)k(u_2)...k(u_3) \tag{2.44}$$

As previously mentioned, an advantage of non-parametric models is that they can model almost arbitrary distributions. A drawback nonetheless is that running inference on a non-parametric model has increased computational effort with growing sample sizes as all samples will be considered candidates being in the neighborhood. Therefore, a lot of research has been done to find ways of scaling kernel density estimators better to higher dimensions and larger sample sizes, such as the use of binning and an application of the Fourier transform [OCRR14].

2.2.8 Entropy Estimation

Much research has been done to solve the entropy of various parametric distributions analytically [LR78]. There are a few problems in terms of the distributions that will be analyzed in this thesis. First, most of the research has been done for univariate distributions. Second, as stated above, it is almost impossible to find a parametric representation for the sampling distribution over a constraint manifold. We do not have any prior knowledge about the

manifold, and the manifold will consist of disconnected parts of complicated shapes. It will therefore be not feasible to solve the entropy analytically. A solution to this is the estimation of the entropy using Plug-In estimates [BDGM97]. These evaluate a KDE on the sample set. A particular plug-in estimator that is used in this thesis is the cross-validation estimate, which does a leave-one-out evaluation of the KDE at the sample-locations x_i.

$$\hat{H}(X) = -\frac{1}{n}\sum_{i=1}^{n} \ln \ \hat{f}_{n,i}(x_i) \tag{2.45}$$

Given n samples, $\hat{f}_{n,i}$ corresponds to the estimate of the probability at the sample location i using a KDE trained on all samples except sample x_i. Leaving out the sample x_i is done to decrease the bias. This can easily be done since training and evaluation of the KDE go hand in hand and do not mean that a tremendous computational effort comes with the training.

2.2.9 Coverage

Uniformity or entropy is a powerful metric for describing how evenly a set of points is distributed. Uniformity in terms of a sample sets entropy $H(S)$ can be seen as making a statement about the local uniformity. It does not tell if the samples are distributed over the entire feasible set $\mathcal{X}$. Take, for example, the two distributions in figure 2.11 it is obvious that if the uniformity would be evaluated using the estimated entropy, both distributions can be considered uniform. Although distribution a is distributed over all of $\mathcal{X}$, distribution b is only distributed over a subset. Therefore, coverage is introduced as the property that S should cover the feasible space with a small covering radius r_c. This means

$$\forall z \in \mathcal{X} : \exists x_s \in S : \ |z - x_s| \leq r_c \ .$$

To measure coverage a reference set of n_{ref} points $R = \left\{x_{\text{ref},i} \in \mathcal{X} : i \in \{1, ..., n_{\text{ref}}\}\right\}$ that are uniformly distributed over the entire feasible set will be compared to the sample set. The average distance from to the sample set is then evaluated.

$$\begin{aligned} \text{ADTS}(R, S) &= \sum_{i=1}^{n_{\text{ref}}} d(x_{\text{ref},i}) \\ d(x_{\text{ref},i}) &= \min_j ||x_{\text{ref},i} - x_{s,j}|| \end{aligned} \tag{2.46}$$

Above will further be referred to as ADTS (Average Distance To Sample Set) If the distance from R to S is small then coverage is large, and it can be concluded that the algorithm generated samples close to most points in the reference. In contrast, if ADTS is large then we can conclude that there are subspaces in the feasible set that the algorithm has not well sampled. Take for example the 4 points in fig 2.11 as a reference. Clearly, the ADTS to distribution b is much greater than the ADTS to distribution a.

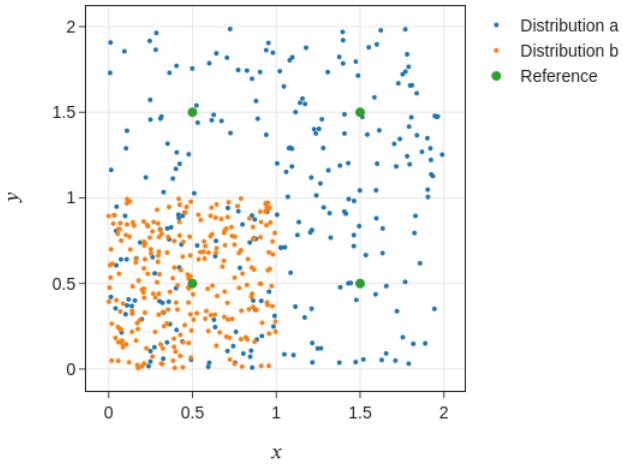

Figure 2.11: Two uniform distributions with different coverage.

2.3 Sampling Methods

Before introducing the algorithms that are used for sampling from constraint manifolds (section 3.2) an introduction of general sampling methods will be given in this section. Sampling is a powerful tool as it can be used in different contexts, for example, the approximation of intractable or analytically not solvable integrals, estimating some parameter, or exploring an unknown space. It is always related to the computational generation of random numbers (mostly pseudo-random), which is a field of research in itself. For this thesis, it will be assumed that we have the computational capacity to uniformly draw numbers between 0 and 1 which can, in turn, be abstracted to generate random numbers of more complicated structures. Sampling the uniform distribution is at the core of most later-described algorithms. It is therefore described first—subsequent rejection sampling and MCMC, especially its most general case of the Metropolis-Hastings algorithms will be presented. Eventually, RRT, a motion planning algorithm used in robotics, will be introduced. The algorithms described in this section can be used to sample from inequality-constrained spaces. The challenges that arise with the presence of equality constraints increase significantly how these algorithms have to be adapted, which will be the content of the section 3.2.

2.3.1 Sampling the Uniform Distribution

Given the capacity to generate random numbers between 0 and 1 a general framework to directly sampling a target probability distribution is finding a function $f(X) = Y$, with $X \sim U(0, 1)$ and Y a random variable following the target distribution. For arbitrary uniform distributions, this results in f being a function that shifts and stretches or contracts X according to the upper bound b_{low} and lower bounds b_{up} of the target uniform distribution of the random variable $Y = \{y \in \mathbb{R} : b_{\text{low}} \leq y \leq b_{\text{up}}\}$ or $Y \sim U(b_{\text{low}}, b_{\text{up}})$.

$$f(x) = x\ (b_{\text{low}} - b_{\text{up}}) + b_{\text{low}} = y \tag{2.47}$$

This framework is called the inverse transformation method [Dev86] and is applied for various univariate probability distributions like the exponential distribution. The function f then takes the form of the inverse of the cumulative distribution of Y. Applying the inverse transformation method is easy for exponential random variables. If Y is not an exponential random variable but normally distributed, other algorithms can be used, such as box-muller (This will not be explored further but left to the reader [BM58]).

This method of sampling arbitrary uniform distributions is chosen for sampling unconstrained but bounded spaces. For spaces in $\mathbb{R}^n$ with $n > 1$, the individual elements of the target vector can be uniformly sampled from within the bounds over the corresponding dimension. A target random vector $X \in \mathbb{R}^n$ that is bounded by b_{low} and b_{up} has the following distribution.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \sim \begin{bmatrix} U(b_{\text{low},1}, b_{\text{up},1}) \\ U(b_{\text{low},2}, b_{\text{up},2}) \\ \vdots \\ U(b_{\text{low},n}, b_{\text{up},n}) \end{bmatrix} \tag{2.48}$$

This will ensure that all points in the space are equally like to be sampled, and no correlation patterns between the different dimensions exist.

2.3.2 Rejection Sampling

Not all distributions can be sampled directly. Rejection sampling is a method trying to overcome this difficulty and is capable of sampling fairly complex distributions. The overall functionality of rejection sampling is having a proposal distribution $q(x)$ and the target distribution $p(x)$ [Bis06]. The proposal distribution is a distribution that is easily sampled. On the other hand, the target distribution probability density function can be queried at any particular point x. The proposal distributions density has to be scaled to assure $kq(x) \geq p(x)$. Intuitively this can be imagined as the scaled proposal distribution fully covering the target distribution (see figure 2.12 for an illustration).

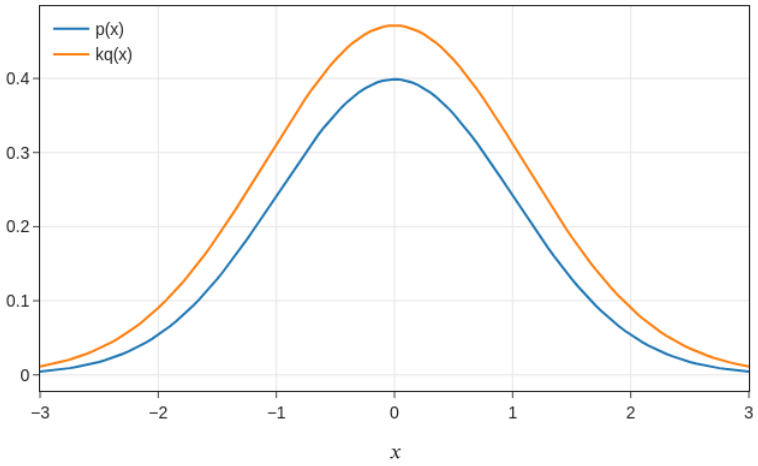

Figure 2.12: Scaled proposal distribution $kq(x)$ and target distribution $p(x)$

Given the PDF $p(x)$, an easy to sample proposal distribution $q(x)$ and its scaled density function $kq(x)$ the rejection sampling algorithm can be iterated after the procedure described in (Algorithm 1).

Algorithm 1 Rejection Sampling

Input: $n \in \mathbf{N},\ k \in \mathbb{R},\ q(x),\ p(x)$

$i \leftarrow 1$
while $i \leq n$ **do**
    $x_i \sim q(x)$
    $u \sim U(0, 1)$
    if $u < \frac{p(x_i)}{kq(x_i)}$ **then**
        Accept x_i
        $i \leftarrow i + 1$
    else
        Reject x_i
    end if
end while

The algorithm will create a set of samples that resemble the target distribution. It comes at the cost of potentially rejecting many samples and thus increasing computational effort. The amount of rejected samples will increase if the $kq(x)$ is much larger than $p(x)$ over a large space of x.

Rejection Sampling over Constrained Spaces

First, let's look at the case of an inequality constrained space $\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0,\ b_{\text{low}} \leq x \leq b_{\text{up}}\}$. In order use rejection sampling we have to choose the proposal distribution $p(x)$ and the constant k. The target distribution $q(x)$ has support over $\mathcal{X}$. We can choose $p(x)$, $q(x)$ and k as:

$$q(x) \sim U(b_{\text{low}}, b_{\text{up}})$$
$$q(x) = \text{constant}$$

Since the target distribution is uniform over the feasible set:

$$p(x) = \begin{cases} \text{constant}, & \text{if } x \in \mathcal{X} \\ 0, & \text{otherwise} \end{cases}$$

As a proposal, we sample the unconstrained space uniformly. Its density is constant over the space. The target distribution is also constant over the feasible set. The constant k is unknown and depends on the volume of the feasible set compared to the whole unconstrained space. Since the density functions for proposal and target distributions are constants we can conclude that:

$$\exists k \in \mathbb{R} :\ kq(x) = p(x)\ \forall x \in \mathcal{X}\ . \tag{2.49}$$

This simplifies the rejection algorithm since the following holds:

$$\frac{p(x)}{kq(y)} = \begin{cases} 1, & \text{if } x \in \mathcal{X} \\ 0, & \text{otherwise} \end{cases}$$

Algorithm 2 Inequality Constrained Rejection Sampling

Input: $n \in \mathbf{N},\ b_{\text{low}},\ b_{\text{up}},\ g(x)$

```
i ← 1
while i ≤ n do
    x_i ~ U(b_low, b_up)
    if g(x_i) ≤ 0 then
        Accept x_i
        i ← i + 1
    else
        Reject x_i
    end if
end while
```

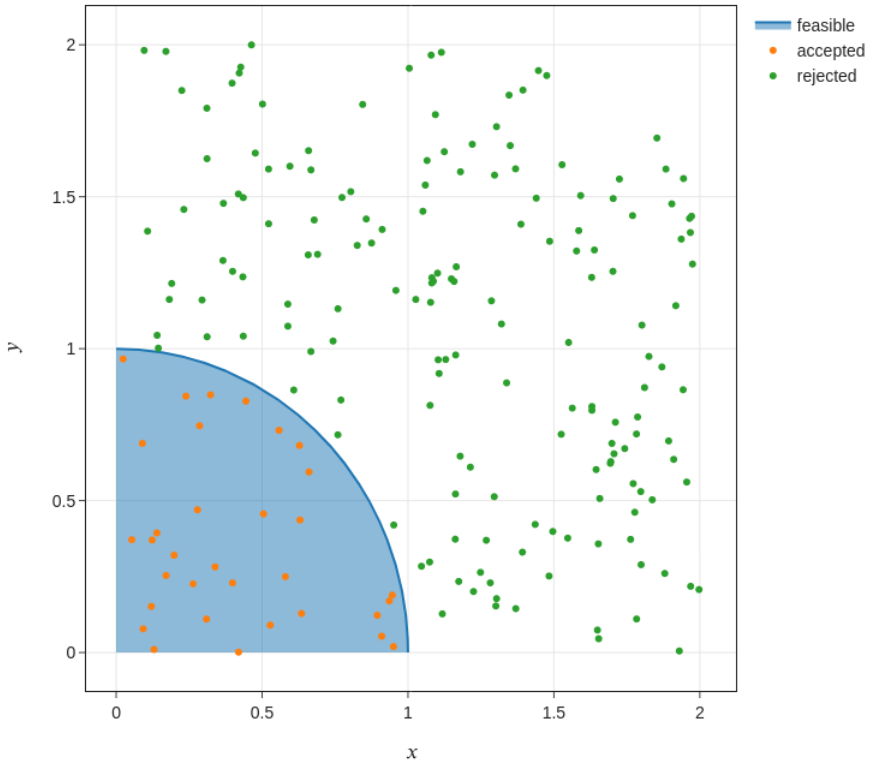

Figure 2.13: Example of the application of rejection sampling on an inequality constrained problem.

The above simplification is a nice property since it is only needed to sample the proposal distribution and check the constraints. The sampling of the uniform number usually used in the accept/reject step can be skipped. Figure 2.12 shows a feasible set within a section of the unit circle and how all uniform samples outside the feasible set are rejected.

As in the example, this works very well for spaces where the feasible set takes up a large proportion of the whole volume of the space. However, it will become problematic if the proportion is tiny, and most samples will be rejected.

Let us now consider the case of the equality constrained space in $\mathbb{R}^n$. The feasible set is now located on a constraint manifold embedded in a lower dimension $k < n$. Given that a sample lands on the manifold, it will still be accepted by the rejection sampling algorithm. A problem that arises is that the probability of sampling an exact point is practically 0, all samples are rejected. The rejection of all samples can further be understood when acknowledging that the relative volume of the manifold is 0.
Some solutions to is relaxation, thus allowing some constraint violations which comes at the obvious cost of a sample set of less quality.

2.3.3 Markov Chain Monte Carlo

Markov-Chain-Monte-Carlo (MCMC) is a class of algorithms that construct Markov-chains to generate samples from a distribution that is difficult to sample from directly or using rejection sampling. The transition matrix describing the dynamics of such a chain is designed so that its stationary distribution resembles the target distribution. Therefore, rejection sampling can also be considered a particular case of the MCMC algorithm where the transition probabilities do not depend on the current state.

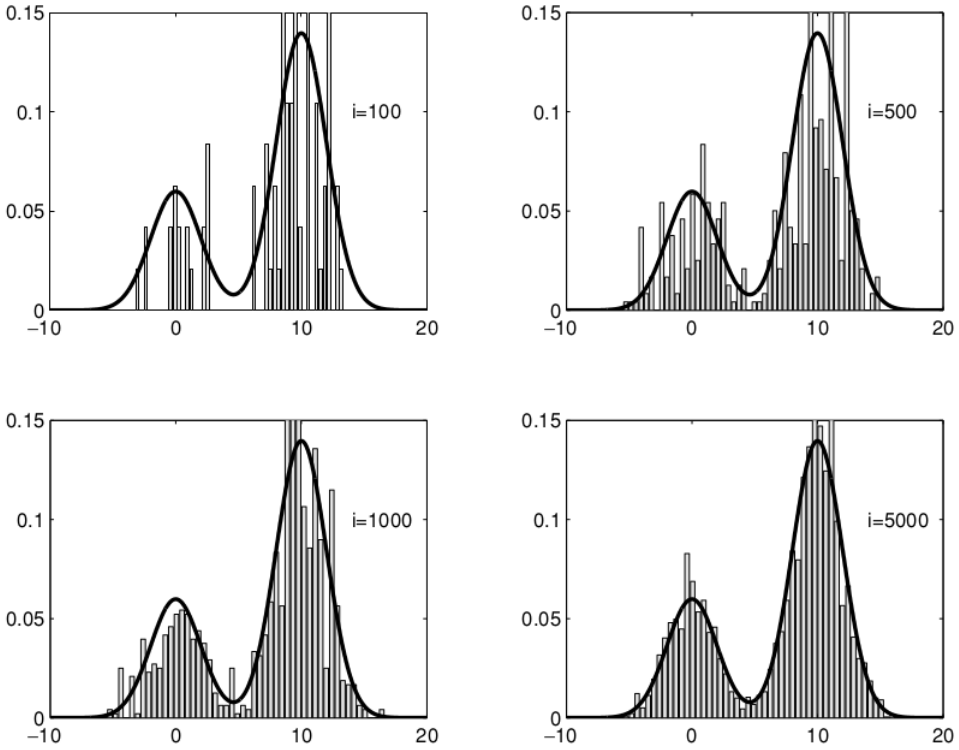

Figure 2.14: Sampling distribution generated at different timesteps using MCMC [AdFDJ03]

Metropolis-Hastings Algorithm

A variety of MCMC algorithms exist, such as Hamiltonian-Monte-Carlo, NUTS, Gibbs-Sampling. The Metropolis-Hastings (MH) algorithm is understood as a foundation to all these algorithms. These algorithms are extensions of Metropolis-Hastings that address particular issues like a large proportion of rejected samples or the inclusion of jacobian information to improve sampling properties [AdFDJ03].

Similar to rejection sampling, we have a proposal distribution $q(x)$, a target distribution $p(x)$, and a criterion by which a proposed sample is rejected or accepted. The major difference to rejection sampling (or why rejection would be assumed a variant of the MH algorithm) is that the proposal distribution is conditional on the latest accepted sample $q(x_i|x_{i-1})$. The procedure that the MH algorithm follows can be taken from Algorithm 3. The Markov chain that is constructed during the application of the MH-Algorithm is then just the sequence of

accepted samples. Furthermore, it is not just the proposed sample that is dependent on the current state/latest accepted sample, but also its acceptance probability.

$$A(x_i, x_{i-1}) = \frac{p(x_i)q(x_{i-1}|x_i)}{p(x_{i-1})q(x_i|x_{i-1})} \tag{2.50}$$

The term 2.50 can be understood as a comparison of the likelihood of taking a step to the proposed state and taking a step from the proposed state back to the current state. If the former is more likely than the latter the sample is accepted, otherwise, the sample will be accepted proportional to the quotient of the likelihoods.

Algorithm 3 Metropolis-Hastings Algorithm

Input: $n \in \mathbf{N},\ q(x),\ p(x)$

$i \leftarrow 1$
while $i \leq n$ **do**
  $x_i \sim q(x_i \mid x_{i-1})$
  $u \sim U(0,1)$
  if $u < \min\left(1, \frac{p(x_i)q(x_{i-1}|x_i)}{p(x_{i-1})q(x_i|x_{i-1})}\right)$ **then**
    Accept x_i
    $i \leftarrow i + 1$
  else
    Reject x_i
  end if
end while

Grid-Walk

The MH algorithm can be used to explore unknown spaces. Grid-Walk is an MH Algorithm with particular choices for the proposal distributions $q(x)$ [Vem05]. It has to be distinguished between the application of Grid-Walk on an inequality constrained space and a constraint manifold. For it to be utilized on a constraint manifold, specific algorithm extensions have to be made. These are constructing a tangent space and the projection onto the manifold. The basic idea of Grid-Walk and its application to inequality-constrained spaces will be introduced here. Its extension will follow in later sections of this document.

Grid-Walk with Rejection

Given an $x_{i-1} \in \mathcal{X}$ feasible set we define the proposal distribution the following:

$$q(x_i|x_{i-1}) \sim U\left(x_{i-1} - \frac{w_{\mathrm{GW}}}{2}, x_{i-1} + \frac{w_{\mathrm{GW}}}{2}\right)$$

Above corresponds to uniformly sampling the neighborhood of the current state x_{i-1}. The neighborhood is a n-cube with edge length w_{GW} and origin at x_{i-1}. w_{GW} is also referred to as the neighborhood width [Vem05]. Similar to the acceptance criteria established for rejection sampling on constrained spaces, we again find that $p(x)$ is constant over the feasible set and zero otherwise, the proposal probability is constant. The term 2.50 therefore simplifies to:

$$A(x_i, x_{i-1}) = \begin{cases} 1, & \text{if } x_i \in \mathcal{X} \\ 0, & \text{otherwise} \end{cases}$$

The above simplifications result in a random walk and a rejection sampling over the neighborhood of the current state. Figure 2.15 displays the first three samples drawn and their neighborhoods. Figure 2.16 displays how a chain would potentially evolve in $\mathbb{R}^2$ for 100 iterations.

Algorithm 4 Grid-Walk with Rejection

Input: $n \in \mathbf{N},\ q(x),\ p(x),\ g(x),\ x_0 \in \mathcal{X},\ w_{\text{GW}} \in$

$i \leftarrow 1$
while $i \leq n$ **do**
  $x_i \sim U(x_{i-1} - \frac{w_{\text{GW}}}{2}, x_{i-1} + \frac{w_{\text{GW}}}{2})$
  if $g(x_i) \leq 0$ **then**
    Accept x_i
    $i \leftarrow i + 1$
  else
    Reject x_i
  end if
end while

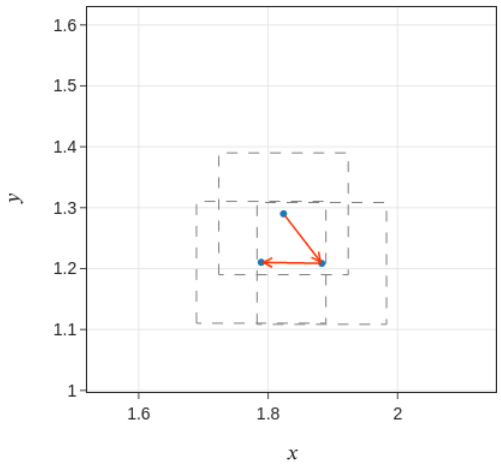

Figure 2.15: Three samples drawn using Grid-Walk. Neighborhoods in dashed-grey.

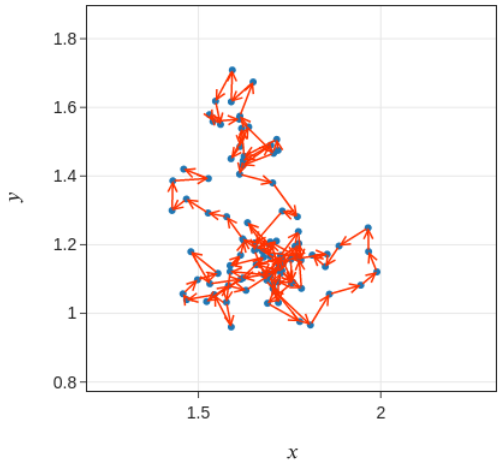

Figure 2.16: Evolution of a Grid-Walk Markov-Chain over 100 iterations.

Grid-Walk with Projection

A modification of above-described version of Grid-Walk is the projection of proposed samples that are infeasible. As a general definition of this algorithm, we will not denote any specific projection method, and the choice of projection method will result in different sample sets. For this thesis though, biased optimization will be the method used for projection.
Due to the projection, all samples are known to be feasible and therefore accepted with:

$$A(x_i, x_{i-1}) = 1$$

Therefore, the rejection step is removed and replaced with a projection step. Figure 5 display how an infeasible sample is projected onto the boundary of the feasible set.

Algorithm 5 Grid-Walk with Projection

Input: $n \in \mathbf{N},\ q(x),\ p(x),\ g(x),\ x_0 \in \mathcal{X},\ w_{\text{GW}} \in \mathbb{R}, PROJECTION(x, g(x))$

1: $i \leftarrow 1$
2: **while** $i \leq n$ **do**
3: $\quad x_i \sim U(x_{i-1} - + \frac{w_{\text{GW}}}{2}, x_{i-1} + \frac{w_{\text{GW}}}{2})$
4: $\quad$ **if** $g(x_i) > 0$ **then**
5: $\quad\quad x_i \leftarrow PROJECTION(x_i, g(x))$
6: $\quad$ **end if**
7: $\quad i \leftarrow i + 1$
8: **end while**

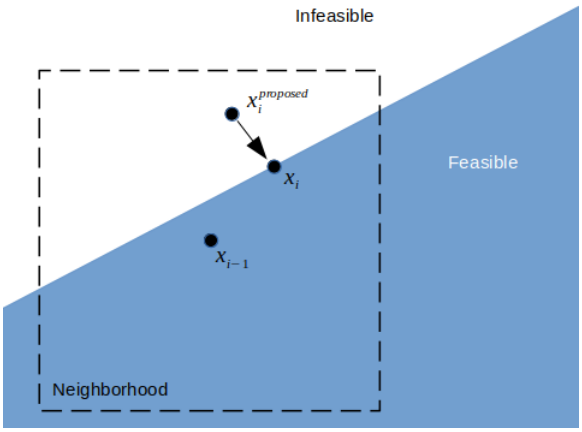

Figure 2.17: Grid-Walk with projection.

2.3.4 Rapidly-Exploring Random Tree

Another method for sample-based planning is RRT, a data structure introduced by Lavelle [Lav98]. It can be seen as a general sampling method for exploring unkown spaces. Compared to famous predecessors such as randomized potential field methods and PRM, RRT addresses some drawbacks of these methods. Some of them are its ease and general functionality, which does not depend heavily on a proper and tedious design of a potential function. The basic principle of RRT is the sampling of random states from the state space or configuration space to grow a tree and explore it for obstacles and valid configurations.

The procedure by which the RRT G is constructed over the configuration space (constrained by $g(x)$) starts by finding a feasible location x_{init} that will then become the root of the tree. Random samples x_{rand} from the configuration space are drawn, the node nearest to x_{rand} is then chosen ($NEAREST\_VERTEX$) (in this theses the metric to be used will be the euclidean distance), and a step with given size α is made in the direction pointing towards the previously draw random sample (NEW_CONF). The new location x_i is then checked for feasibility and added to the tree if feasible ($G.add\_vertex$ and $G.add\_edge$).

The uniform sampling paired with the nearest neighbor search and the generation of a new step between these points results in the RRT being biased towards exploring unknown subspaces of the configuration space and making up the "rapidness" of the exploration. This can easily be understood by studying the figure 2.18 algorithms which shows a sequence of the trees' state during exploration and how the tree very rapidly evolves to regions far from the root.

Algorithm 6 Build RRT

Input: $n \in \mathbf{N},\ b_{\text{low}},\ b_{\text{up}},\ g(x),\ x_{\text{init}},\ \alpha, G$

$i \leftarrow 1$
$G.init(x_{\text{init}})$
while $i \leq n$ **do**
    $x_{\text{rand}} \sim U(b_{\text{low}},\ b_{\text{up}})$
    $x_{\text{nearest}} \leftarrow NEAREST\_VERTEX(x_{\text{rand}}, G)$
    $x_i \leftarrow NEW\_CONF(x_{\text{nearest}}, \alpha)$
    if $g(x_i) \leq 0$ *and* $b_{\text{low}} \leq x_i \leq b_{\text{up}}$ **then**
        $G.add\_vertex(x_i)$
        $G.add\_edge(x_{\text{nearest}}, x_i)$
        $i \leftarrow i + 1$
    else
        Reject x_i
    end if
end while

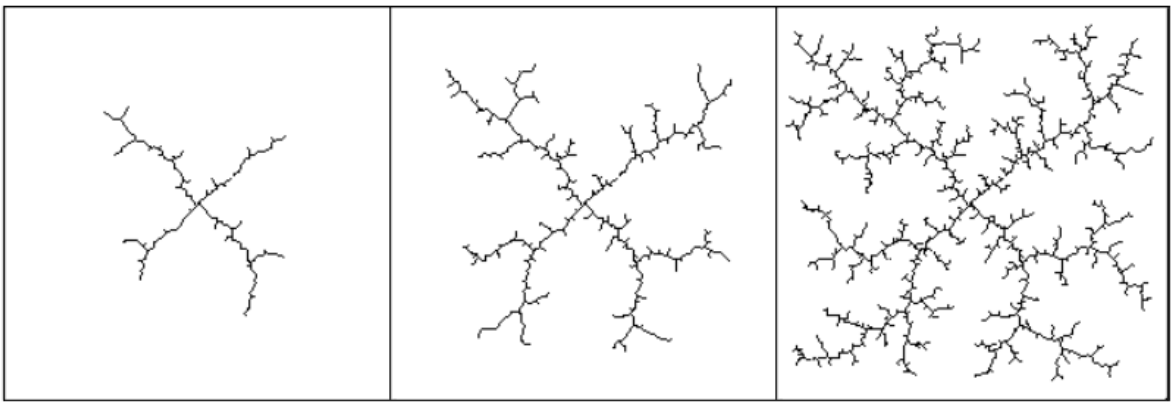

Figure 2.18: Evolution of an RRT in $\mathbb{R}^2$ [LaV].

2.4 Optimization

Various kinds of optimization problems exist. These problems could be the maximization of flow through a network, the traveling salesman problem, or even tasks like portfolio optimization in financial mathematics. All scientific fields apply optimization. Optimality principles are furthermore a way of describing states like the formation of atoms to some equilibrium state with minimum energy. Although diverse in application, there is a general problem description of an optimization problem [Tou21].

$$\min_{x \in \mathbb{R}^n} f(x) \text{ s.t. } g(x) \leq 0,\ h(x) = 0\ . \tag{2.51}$$

A function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is being optimized, such that it adheres to inequality constraints $g(x) : \mathbb{R}^n \to \mathbb{R}^m$ and the equality constraints $h(x) : \mathbb{R}^n \to \mathbb{R}^l$. These are the most basic

descriptions of an optimization problem. Solving such a problem begins with the design of the goal (or cost) function and its constraints such that they reflect the problems underlying dynamics and behavior as well as being in a form that can be solved using known approaches. Intuitively this corresponds to the goal function representing a surface consisting of feasible and infeasible region. Solving this problem corresponds to maneuvering over the surface to find its deepest valley (see figure 2.19) or deepest point in the feasible region. It can further be distinguished between local and global minimum. The former just being any valley on the surface that locally represents an optimum, the latter being the unique valley that minimizes the goal function globally. This is an important distinction since one often wants to find the global optimum. The presence of local optima nonetheless gives rise to certain challenges of solving for the global optimum, such as getting stuck in a local optimum.

The structure of how a problem is defined gives rise to a simple classification of optimization problems that all have corresponding suitable algorithms.

- Constrained vs. Unconstrained
- Linear vs. Non-linear
- Convex vs. Non-convex

2.4.1 Algorithms

The above characteristics of an optimization problem have important implications for solving the problem. Some characteristics like convex problems are in general favorable since the existence of a single optimum is ensured. Some algorithms, like the simplex algorithm, can only be used with linear problems. Therefore, the choice of the algorithm that shall solve a particular problem should take all these characteristics into account. In the most simple case, such as finding the minima of a parabola, one can even find the solution analytically. Since most optimization problems are of higher dimensions, numerical methods are used. Below is a brief description of a variety of algorithms.

Black-Box

These methods only require that the goal function $f(x)$ and the constraints can be queried for any x. Genetic algorithms and Bayesian optimization are examples of black-box algorithms based on sampling the goal function and using the so gained knowledge about the goal function to sample promising regions.

Gradient Methods

For gradient methods, it is necessary to have access to the gradient $\nabla f(x)$ or to compute it numerically. The basic idea behind gradient methods is to walk against the direction of the gradient and then converge in an optimum for specified criteria.

$$x_{i+1} = x_i - \alpha \nabla f(x_i) \tag{2.52}$$

In 2.52 $\alpha \in \mathbb{R}$ is a step size parameter that specifies how large of a step is taking. In a most basic form of gradient descent, is α assigned some fixed value. Various methods exist

to choose an optimal α and include the handling constraints.

A further extension of this approach is Newton Method that given the Hessian of the goal function $\nabla^2 f(x)$ provides the newton direction. The newton direction describes the above update as a step in the direction where the second order taylor approximation of $f(x_i)$ has a minimum [Tou21].

$$x_{i+1} = x_i - \nabla^2 f(x_i)^{-1} \nabla f(x_i) \tag{2.53}$$

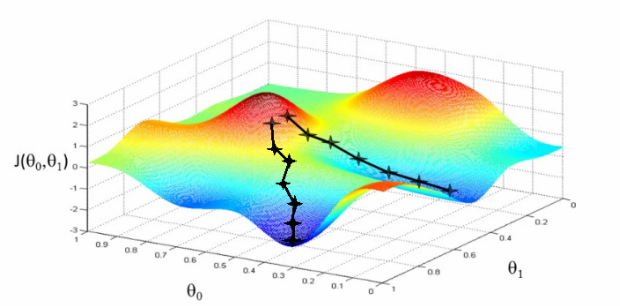

Figure 2.19: Example of two trajectories realized using gradient descent [Vry].

Constrained Optimization

Many applications have to respect constraints. Often the global optimum of the unconstrained problem may not lie in the feasible region anymore. Above algorithms therefore have to be adapted to cater for this situation. This adaptation often means that we find an alternative representation of the problem like transforming the constrained problem into a sequence of unconstrained problems. The algorithms then generally go through a loop of solving an unconstrained problem and using the solution as the starting point for the next unconstrained problem until it converges to the actual solution of the constrained problem[Tou21].

Interior Point methods for example transform the constrained problem into a sequence unconstrained problem by introducing a barrier term. This barrier term relates to the inequality constraints violation and ensures that the found solution stays within the feasible region. The barrier term usually corresponds to the negative logarithm of $g(x)$. The objective can therefore be written as:

$$\min_x f(x) - \mu \sum_i \log\big(-g_i(x)\big) \tag{2.54}$$

Above problem is solved for a fixed μ. It will then be decreased and the previously attained solution used as an initial guess for solving the next problem. This method requires a feasible point as a starting point which can be hindrance since finding a feasible point is not trivial for problems with a complicated set of constraints.

Squared Penalty methods also solve a sequence of unconstrained optimization problems. In this method a penalty term is added to the goal function.

$$\min_x f(x) + \mu \sum_i [g_i(x) > 0] g_i(x)^2 + \nu \sum_j h_j(x) \tag{2.55}$$

Here $[g_i(x) > 0]$ corresponds to the indicator function that evaluates to 1 if $g_i(x) > 0$ and 0 otherwise. This algorithm also goes through a loop of solving instances of the above described problem. The penalty parameters μ and ν are being increase after solving a problem. In contrast to aboves interior point methods do Squared Penalty methods not require a feasible starting point this comes at the cost of the found solution generally having some violation of the constraints.

Augmented Lagrangian methods solve following unconstrained optimization problem [Tou14]:

$$\min_x f(x) + \mu \sum_i [g_i(x) > 0 \vee \lambda_i > 0] g_i(x)^2 + \sum_i \lambda_i g_i(x) + \nu \sum_j h_j(x)^2 + \sum_j \kappa_j h_j(x) \tag{2.56}$$

As the name suggest we are adding a lagrange term. The problem is solved intially for $\kappa_j = 0$ and $\lambda_i = 0$. In the next iteration κ_j and λ_i are chosen such that they compensate for the penalty in the previous iteration. Especially for equality constrained optimization problems does this approach have the advantage of achieving exact constraints, unlike in squared penalty methods. ν and μ can further be kept constant throughout the algorithms runtime and the in- or decrement of the κ_j and λ_i update does not have to be chosen manually, but is estimated optimal as described above which further increases the efficiency and robustness of this approach.

Other

Besides black-box and gradient methods, other methods exist, such as the simplex algorithm, which can be applied to solve linear problems by exploiting the simplex structure of the feasible set and traversing the edges of the simplex to find the minimum.

2.4.2 Biased optimization

Most of the approaches mentioned above need an initial starting point x_0 that must be feasible. Finding such a point $x_0 \in \mathcal{X}$ can be a not so straightforward task given that the constraints slice the configuration space in various unknown components. Finding a feasible point can be represented as an optimization problem, and solving such is called phase 1 optimization. A general phase 1 problem that assumes no cost $f = 0$ has the following form.

$$\min_{x \in \mathbb{R}^n} f \text{ s.t. } g(x) \leq 0,\ h(x) = 0\ . \tag{2.57}$$

In general, phase 1 optimization precedes solving the actual optimization problem. The task of sampling from a constraint manifold can be reformulated as sampling solutions to a phase 1 optimization problem. Biased optimization is the approach used in this thesis. Given a bias (also referred to as seed) x_{bias}, biased optimization tries to find a point x that minimizes the distance to the bias x_{bias} while not violating any constraints.

$$\min_{x \in \mathbb{R}^n} ||x - x_{\text{bias}}||^2 \text{ s.t. } g(x) \leq 0,\ h(x) = 0\ . \tag{2.58}$$

Indeed a variety of metrics could be used to compare the distance between x and x_{bias}. The euclidean norm is a good choice for its intuitive understanding in low dimensions and its derivative easily being found, making the approach available to gradient-based algorithms.

3 Methods

3.1 Objectives

The objective of this thesis is the study of algorithms that sample from constraint manifolds $\mathcal{X}$:

$$\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0,\ h(x) = 0\}.$$

Specifically we want to sample $S \subset \mathcal{X}$ with low computational cost that is diverse and covers $\mathcal{X}$ well. Following measures will be evaluated as a reflection of these properties, uniformity, coverage and computational efficiency are measured:

- Entropy H (Specifically $\hat{H}$ an estimate of H using the a KDE $\hat{f}(S)$), as a measure of uniformity. Large Entropy being better.
- Variance $\mathrm{Var}(Y)$ where $Y = \{y_i \in \mathbb{R} : \hat{f}(x_i) = y_i \text{ for all } x_i \in S \ \ i \in [1; n_{\text{samples}}]\}$. The variance of the kernel density estimates for the generated samples. Low variance being better.
- $\mathrm{ADTS}(R, S)$ the average minimum distance from a uniform reference set R to S as a measure of coverage. Low being better.
- AIPS the average number of iterations to generate a sample from $\mathcal{X}$ as a measure of computational efficiency. Less being better.

The algorithms i.i.d.-Biased-Sampler, RRT-on-Tangent, Grid-Walk-on-Tangent that are used for sampling from constrained manifolds will be introduced in the coming section. A composite sampler that merges i.i.d.-Biased-Sampler and RRT-on-Tangent or Grid-Walk-on-Tangent will be proposed as a novel algorithm, as well as extensions like a filter, rejection sampling and bandit-sampler.

3.2 Constrained Sampling

The previous sections have given an overview of some basic methods applied for sampling tasks/state exploration, such as RRT, MCMC, and rejection sampling. These discussions were limited to constrained spaces with inequalities $g(x)$. We recall that it is impossible to apply these algorithms in the previously described form on constraint manifolds due to the manifold surface not having any volume and the probability of sampling an exact value is 0. These algorithms have to be adapted. Since the presence of equality constraints $h(x)$ and thereby a feasible set on a manifold are the major problem statement of this thesis and most relevant for the eventual applications of algorithms in real-world cases we will introduce how the use of tangent spaces and projection are applied to RRT and Grid-Walk

in order to facilitate sampling from constraint manifolds. In the following, two broad classes of samplers for constraint manifolds are introduced i.i.d.-like (3.2.1) and Markov-Chain-like samplers (3.2.2). These two classes can be combined to form composite samplers (3.2.3) that address certain difficulties in achieving high coverage and uniformity. Eventually, further functionality like filtering of samples, constraint handling, and the approximation of the size of the manifold are being discussed 3.3.

3.2.1 i.i.d.-Biased-Sampler

The general task of a sampler is to generate a sequence of random variables. The generated samples form an i.i.d.-like sampler are independent of each other. Opposing this is Markov Chain-like samplers where the generated sample is dependent on the previously drawn sample (discussed in the next section). Such an i.i.d. sampler is the uniform sampler over the configuration space that generates a sequence of random configurations all drawn from the same distribution $X \sim U(b_{\text{low}}, b_{\text{up}})$. These samples will not lie on the manifold, and we have to project the sample onto the manifold using biased optimization. The resulting feasible samples are still independent of each other. $BIASED\_OPTIMIZATION$ solves the following problem:

$$\min_{x \in \mathbb{R}^n} ||x - x_{\text{bias}}||^2 \text{ s.t. } g(x) \leq 0,\ h(x) = 0\ .$$

Algorithm 7 i.i.d. Biased Sampler

Input: $n \in \mathbf{N},\ b_{\text{low}},\ b_{\text{up}},\ g(x),\ h(x),$

1: $i \leftarrow 1$
2: **while** $i \leq n$ **do**
3: $\quad x_{\text{bias}} \sim U(b_{\text{low}},\ b_{\text{up}})$
4: $\quad x_i \leftarrow BIASED\_OPTIMIZATION(x_{\text{bias}},\ g(x),\ h(x))$
5: $\quad i \leftarrow i + 1$
6: **end while**

Two significant drawbacks can arise from this approach. The first drawback stems from the fact that if the manifold stretches through a small part of the configuration space and projecting a point very far from the manifold results in higher computational efforts. The second drawback arises from the manifold not stretching symmetrical through the configuration space, which will likely result in non-uniform sampling distributions (this is more closely studied in 4.2).

3.2.2 Markov-Chain-Like-Samplers

A Markov-Chain-Like-Sampler evolves around the idea of generating a sequence of random numbers where the next sample to drawn depends upon the preceding sample as has been introduced in section 2.3.3. This temporal dependency leads to an autocorrelation of generated samples. The resulting sample set S shall mirror our target distribution the uniform distribution. Such approaches are Metropolis-Hastings/Grid walk(2.3.3) and RRT (2.3.4) that have previously been introduced to inequality-constrained spaces. As mentioned before, some problems arise using i.i.d.-Like-Samplers, such as the i.i.d.-Biased-Sampler. Computational efforts and non-uniformity of the resulting distribution of samples are expected drawbacks.

Grid-Walk-on-Tangent and RRT-on-Tangent are meant to specifically address these issues by incorporating knowledge that has been gained during the sampling process. For example, a sample $x_i \in \mathcal{X}$ has been found. By knowing that location x_i is on the manifold, we can infer that any point near x_i is more likely to be on the manifold and feasible or at least not too far from the manifold. The i.i.d.-Sampler disregards this knowledge, and sampling any point as a seed to the succeeding biased optimization problem has equally likely chances of being a good candidate. Grid-Walk and RRT incorporate this knowledge and can be classified as a Markov-Chain-Like-Sampler. We recall that the Markov property states that a state only depends upon its previous state, transferring this to our sampling problem, the next sample x_{i+1} only depends upon the current state. For Grid-Walk-on-Tangent, the current state can easily be defined as the most recent drawn sample x_i, just as has been described in 2.3.3. On the other hand, the current state in RRT-on-Tangent encapsulates all vertices and edges that have been added to the tree so far.

The overall structure of a Markov-Chain-Like-Sampler for constraint manifolds can be described as follows. A seed or initial feasible point x_0 has to be found in the initial step. This can be done by sampling the uniform distribution over the configuration space and using biased optimization to project it onto the manifold. The tangent space of the manifold is then used to generate more seeds for a biased optimization problem that projects the points onto the manifold. The use of tangent spaces and projection is the central idea that distinguishes Markov-Chain-Like-Samplers for inequality constrained spaces and constraint manifolds.

Grid-Walk on Tangent

Similarly, like Grid-Walk on inequality constrained spaces, the Grid-Walk on the tangent has a predefined local neighborhood of width w_{GW} from which to generate a sample near the current sample x_{i-1}. It differs though how the neighborhood is defined. As seen from 9 the algorithms starts by finding an initial starting point x_0, which can be done by drawing a uniform sample over the configuration space and projecting it onto the manifold using biased optimization (see figure 3.1). It would then go on to find the tangent at this point and generate a uniform sample x_{rand} over a bounded subset of the tangent space with origin (this corresponds to a rectangular space of width w) at the previously found point x_{i-1}. The algorithm would take this x_{rand} as a seed to the biased optimizer and generate a sample x_i on the manifold and iterate this procedure of finding a tangent and optimization till the specified number of samples n are drawn.

In above algorithm $FIND\_TANGENT$ solves the following problem (as described in 2.1.2):

$$\begin{bmatrix} \boldsymbol{J} \\ \boldsymbol{\Theta^T} \end{bmatrix} \boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{I} \end{bmatrix} \tag{3.1}$$

Here $\boldsymbol{J}$ is the jacobian of $h(x_{i-1})$ and $\boldsymbol{\Theta} \in \mathbb{R}^{n,m}$ an orthonormal basis of the tangent space at x_{i-1}.

$SAMPLE\_TANGENT$ then uses this $\boldsymbol{\Theta}$ to map a $u \in \mathbb{R}^m : u \sim U(-\frac{w_{\text{GW}}}{2}, \frac{w_{\text{GW}}}{2})$ to the configuration space:

Algorithm 8 Grid Walk on the Tangent Space

Input: $n \in \mathbf{N},\ b_{\text{low}},\ b_{\text{up}},\ g(x),\ h(x),\ w_{\text{GW}}$

$i \leftarrow 0$
$x_{\text{rand}} \sim U(b_{\text{low}},\ b_{\text{up}})$
$x_0 \leftarrow BIASED\_OPTIMIZATION(x_{\text{rand}},\ g(x),\ h(x))$
$i \leftarrow i + 1$
while $i < n$ **do**
    $\mathbf{\Theta} \leftarrow FIND\_TANGENT(x_{i-1}, h(x))$
    $x_{\text{rand}} \leftarrow SAMPLE\_TANGENT(x_{i-1}, \mathbf{\Theta}, w_{\text{GW}})$
    $x_i \leftarrow BIASED\_OPTIMIZATION(x_{\text{rand}},\ g(x),\ h(x))$
    $i \leftarrow i + 1$
end while

$$x_{\text{rand}} = x_{i-1} + \mathbf{\Theta} u \tag{3.2}$$

This procedure will overcome the general problems attached to i.i.d.-Samplers having large computational cost by staying close to the manifold. Nevertheless, this approach faces problems when the manifold consists of various disconnected components and the Markov-Chain gets stuck on one component, not exploring the other. Something similar can be imagined in $\mathbb{R}^2$ when the manifold consists of two circles, one fully enclosing the other. If the first sample lies on the outer circle the inner circle will not be sampled since the tangent space is located outside the outer circle, with the optimizer converging to a point on the outer circle again. Below figure illustrates how we can walk on the manifold using Grid-Walk-on-Tangent.

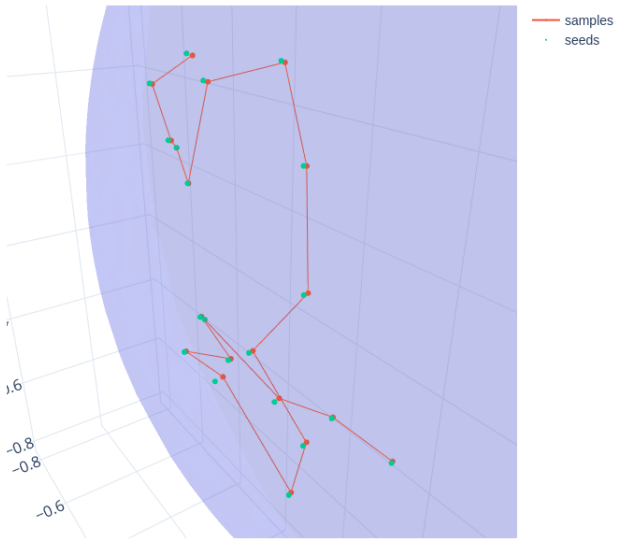

Figure 3.1: Example of an Grid-Walk on the tangent of a sphere and its projections

RRT on Tangent

Sampling constraint manifolds using RRT also requires first finding the tangent space, then sampling the tangent space and projecting onto the manifold. The tangent space is not recomputed on every iteration during the algorithm's runtime but only once after an initial feasible point on the manifold is found. This point will become the root of the RRT. The RRT is then grown on the tangent space. The vertices of the RRT on the tangent are projected onto the manifold (see figure 3.2).

Algorithm 9 RRT on the Tangent Space

Input: $n \in \mathbf{N},\ b_{\text{low}},\ b_{\text{up}},\ g(x),\ h(x),\ w_{\text{RRT}},\ G,\ \alpha,\ x_0$

$i \leftarrow 0$
$x_{\text{rand}} \sim U(b_{\text{low}},\ b_{\text{up}})$
$x_i \leftarrow BIASED\_OPTIMIZATION(x_{\text{rand}},\ g(x),\ h(x))$
$G.init(x_0)$
$\mathbf{\Theta} \leftarrow FIND\_TANGENT(x_0, h(x))$
$i \leftarrow i + 1$
while $i < n$ **do**
    $x_{\text{rand}}^{\text{tangent}} \leftarrow SAMPLE\_TANGENT(x_0, \mathbf{\Theta}, w_{\text{RRT}})$
    $x_{\text{nearest}}^{\text{tangent}} \leftarrow NEAREST\_VERTEX(x_{\text{rand}}^{\text{tangent}}, G)$
    $x_i^{\text{tangent}} \leftarrow NEW\_CONF(x_{\text{nearest}}^{\text{tangent}}, \alpha)$
    $G.add\_vertex(x_i^{\text{tangent}})$
    $G.add\_edge(x_i^{\text{tangent}}, x_{\text{nearest}}^{\text{tangent}})$
    $x_{\text{bias}} \leftarrow x_i^{\text{tangent}}$
    $x_i \leftarrow BIASED\_OPTIMIZATION(x_{\text{bias}},\ g(x),\ h(x))$
    $i \leftarrow i + 1$
end while

One can imagine that this approach works very well if the manifold is relatively flat. The RRT on the tangent will grow to the edges of its local bounds in a structured manner while the fixed stepsize α maintains some balance in the density at which vertices appear on the tangent. If the manifold is rather flat we hope the projected samples being similarly distributed as the vertices on the tangent. It has to be kept in mind that the manifold's curvature plays an important role. If the manifold is somewhat bent than flat, the tangent space is not a good approximation of the manifold with growing distance from the root. The optimization problem will likely be solved in more iterations since the distance between the tangent and the manifold grows. This would furthermore lead to a distortion of the distribution over the manifold. Another drawback is defining the area over which the RRT is grown upfront, which is problematic if the manifold is relatively tiny compared to the RRT bounded area. Especially if the manifold is disconnected, we expect many samples to be projected onto the edge of the manifold, also further distorting the distribution.

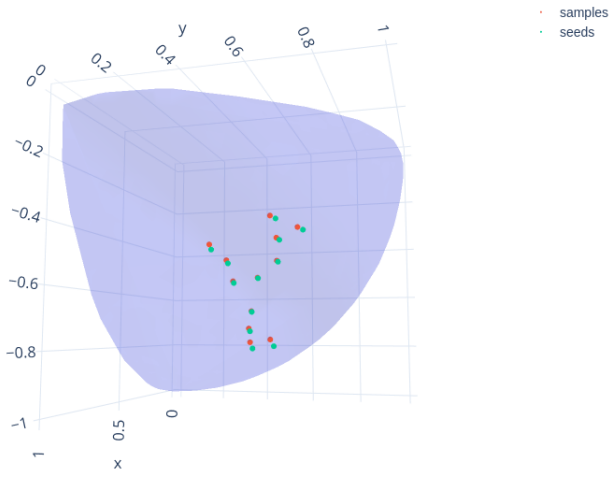

Figure 3.2: Example of an RRT on the tangent of a sphere and its projections

3.2.3 Composite Samplers

Markov-Chain-Like and i.i.d.-Like-Samplers can be combined to produce a composite sampler. Combining both should be the general case since they complement each other very well. Markov-Chain-based samplers are likely to generate a uniform distribution over a subset of the feasible space with a lack of coverage. In contrast, i.i.d.-Samplers generate samples likely to cover the feasible space well but lack uniformity. Therefore, a composite sampler consists of a global sampling component and a local sampling component. Within the global sampling stage, n_{global} seeds for the local sampling are being found, which can be done using an i.i.d.-Sampler with biased optimization (i.i.d.-Biased-Sampler). These seeds are used as the root of the RRT or the initial starting point of a Grid-Walk $x_{0,i} \forall\ i \in \{0, ..., n_{\text{local}}\}$. This procedure can be thought of as running n_{local} Markov-Chains. Given that the i.i.d.-Sampler generates a set of samples that cover the feasible space well and the local sampling on each chain generates a set that is locally uniform we hope that the union of the samples is again of high uniformity and coverage even if the feasible manifold is disconnected and has various parts.

Handling disconnected manifolds is a challenge described earlier. This challenge is meant to be overcome using composite samplers since each component has a Markov-Chain-Sampler. Still, other challenges arise. New parameters are introduced, which choice will influence uniformity and coverage. Sampling too many points in the global sampling stage may ensure that seeds are present on all individual manifold components, resulting in greater computational efforts. It furthermore may not lead to good uniformity since the local sampler like Grid-Walk would need a reasonably large amount of samples to explore the manifold and achieve uniformity. On the other hand, sampling to few seeds on the global stage may lead

Algorithm 10 Composite Sampler

Input: $n_{\text{local}}, n_{\text{global}} \in \mathbf{N},\ b_{\text{low}},\ b_{\text{up}},\ g(x),\ h(x),\ LOCAL\_SAMPLER,$

1: $i \leftarrow 0$
2: **while** $i < n_{\text{global}}$ **do**
3: $\quad x_{\text{rand}} \sim U(b_{\text{low}},\ b_{\text{up}})$
4: $\quad x_{0,i} \leftarrow BIASED\_OPTIMIZATION(x_{\text{rand}},\ g(x),\ h(x))$
5: $\quad LOCAL\_SAMPLER(x_{0,i}, n_{\text{local}})$
6: $\quad i \leftarrow i + 1$
7: **end while**

to a scenario where not all individual parts of the manifold are reached.

Furthermore, it has to be kept in mind that the multiple Markov-Chains may also distort the distribution. Consider, for example, two RRTs grown on the same manifold in a way that they overlap, resulting in higher density in regions where the RRTs overlap compared to the regions where they do not overlap. Some approaches to tackle these drawbacks will be introduced in the coming section.

3.3 Extension

Next to combining samplers, a few other steps can be integrated into the sampling approach to increase uniformity and coverage. These steps are heuristics that shall post-process a sample set to become more uniform or decide which chain to sample next and if this sample shall be rejected or accepted.

3.3.1 Filter

The intention of filtering a set of samples is to remove samples such that the resulting set is more uniform than the original. The heuristic by which a sample is removed is based on the nearest neighbor distance. If a set of samples is uniformly distributed it can be easily imagined that the distance of each point to its closest neighbor shall be somewhat similar for all points. By the same logic, one can see that if points are not uniformly distributed, the nearest neighbor distance might be widespread. Points lying in densely populated regions of the sampling space will have minimal nearest neighbor distances, while those in sparsely populated regions have large nearest neighbor distances. One can try to dilute the densely populated regions by removing points to balance these discrepancies.

The dilution of dense subspaces is essentially what the filter operation does. It takes in a set of samples S, tests each point for its nearest neighbor distance, and randomly selects the point itself or the nearest neighbor if a predefined threshold β is not exceeded. The algorithm starts by finding the nearest neighbor for each sample and storing the tuple $p_k : (x_i, x_j)$ in a set L. Next to this, we keep track of the samples that have been removed using the set $REMOVED$. The algorithm then works by iterating over pairs l in L. First $l.isin(REMOVED)$ checks if any samples of the current tuple have already been removed and if the distance threshold β is not reached. Any of the two elements of the tuple is then removed with equal probability.

Finally $L.get\_all()$ returns all unique samples S_{unique} in the remaining tuples of L.

Algorithm 11 Filter

Input: S, β

$L \leftarrow FIND\_NEAREST\_NEIGHBORS(S)$
$REMOVED.init()$
for each $l \in L$ **do**
    if *not* $l.isin(REMOVED)$ *and* $l.distance() < \beta$ **then**
        $x_i \leftarrow l.remove()$
        $REMOVED.insert(x_i)$
    end if
end for
$X_{\text{filtered}} \leftarrow L.get\_all()$

This filtering operation can be computationally costly for large sample sets S since it requires the calculation of a distance matrix $D \in \mathbb{R}^{n_{\text{samples}}, n_{\text{samples}}}$. Therefore, it is most suitable when applied to the seeds generated in the global sampling phase of a composite sampler. The filtering of seeds is vital because the individual Markov-Chains do not know the state of the other chains. Two chains with seeds next to one another would then later in the union of all samples result in a high density near the seeds. It can therefore be expected that if we compare two sets of seeds, one is uniformly distributed over the feasible set, the other is not, that the uniformly distributed set of seeds would lead to a more uniform distribution once the local sampling stage is finished.

3.3.2 Rejection Sampling

A problem to be expected when projecting points onto a constraint manifold is that the resulting distribution of projected points may be non-uniform because projected points aggregate on the edges of the manifold. This kind of scenario is usually the case when a combination of equality and inequality constraints is present. The equality constraints create the manifold, and the inequality constraints slice the manifold into disconnected components. This phenomenon of an accumulation of projected points in certain subspaces (specifically where the inequalities are met) of the manifold can be treated differently by handling the constraints. In general, both equality constraints and inequality constraints are integrated into the biased optimization problem. This can be split into two subproblems. First, the optimization problem is reformulated as an optimization problem that disregards the inequality constraints. The solution to this problem does not necessarily have to be feasible. Therefore, the next problem to solve is a reject-accept step on the solution of the previous optimization problem. Feasible samples are accepted, and samples that violate any inequality constraint are rejected. Mathematically this can be interpreted as treating the feasible set as an open set instead of a closed set that includes the edges. Points that accumulate on the edges are rejected.

3.3.3 Bandit Sampler

The bandit sampler is an extension of the composite sampler. If the composite samplers local chains are ran in parallel and not one after the other we can have a decision on which chain to expand in the next iteration. The bandit sampler then implements a logic to choose which chain to sample next to achieve uniformity. The naming bandit-sampler is derived from the multi-armed bandit in gambling, where one faces the problem of choosing the arm that maximizes the reward. The algorithm tries to solve the following problem. Imagine a constraint manifold consisting of two separate components of varying size. A composite sampler that globally generates two samples, each lying on a different manifold component, is then used. The two resulting chains or trees would generate samples from its respective manifold component. Since there is no prior knowledge of how these two components differ, the most sensible thing is to sample each chain for the same amount of time. Sampling both manifolds for the same amount will create an imbalance in the uniformity of the overall distribution since the samples on the small component have to be much denser situated on the manifold than the large component. A natural way to solve this issue would be to sample each component in proportion to its size. Since we usually do not know the number of disconnected components and their size, we have to approximate the size of a component as the chains progress. The bandit algorithm addresses this problem by approximating the size of a component and adjusting the probabilities of expanding a chain accordingly.

Several assumptions are made for this algorithm to be functional.

1. A single seed or equal amounts of seeds are placed on each disconnected manifold component.
2. The local sampling algorithm that expands the Markov-Chain generates a uniform distribution over the manifold component.
3. The disconnected manifold components are roughly rectangular.

Assumption one should hold since otherwise, the density of components with multiple seeds would be greater than that of a component with single or fewer seeds. As introduced in 2.2.3 we know that the covariance **Cov** of uniformly distributed samples over a rectangular space can be related to the width of the sides of the rectangle. So assuming a disconnected manifold component is rectangular. We are given its **Cov**, we can establish a function $f : \mathbb{R}^{n,n} \to \mathbb{R}^k$ that maps $f(\mathbf{Cov}) = d_w$ to a vector of the widths d_w of the rectangle (k corresponding to the dimensionality of the manifold), which in turn can be used to approximate the volume of the rectangular component.

$$V = \prod_{i=1}^{i=k} d_{w,i}. \tag{3.3}$$

The general algorithm is therefore composed of an initial sampling stage where each chain is sampled n_{init} times to generate some samples and an initial estimate of the covariance matrices. Algorithm 12 describes the procedure succeeding the initial sampling stage. Given the covariance matrices $\mathbf{Cov}_i$, we then approximate the size of each component and find a multinominal distribution over the chains that would result in the same density of points on each chain. The chain to be expanded is sampled from the multinominal distribution. After

each expansion, the size of the component is updated and the probabilities of the multinominal. The covariance and mean needed for this estimation are online updated as described in 2.28.

It has to be kept in mind that this algorithm is susceptible to previous assumptions. A very curved and non-rectangular manifold would result in the eigenvalue decomposition not giving a good volume approximation. Multiple chains on the same component would result in this component having a multiple of the density as if a single chain would just sample it. Clustering the chains would be one approach to overcome this challenge.

Algorithm 12 Bandit Sampler

Input: $n_{\text{chains}}, n_{\text{samples}} \in \mathbf{N}$, b_{low}, b_{up}, $g(x)$, $h(x)$, $LOCAL\_SAMPLER$, $\mathbf{Cov}_i$, chains

1: $i \leftarrow 0$
2: **while** $i < n_{\text{samples}}$ **do**
3: $A \leftarrow APPROXIMATE\_SIZE(\text{chains})$
4: $u \leftarrow CHOOSE\_NEXT\_CHAIN(A, \text{chains})$
5: $x_i \leftarrow LOCAL\_SAMPLER(\text{chains}[u], b_{\text{low}}, b_{\text{up}}, g(x), h(x))$
6: $UPDATE\_MEAN(\text{chains}[u], x_i)$
7: $UPDATE\_COVARIANCE(\text{chains}[u], x_i, \mu_c)$
8: $i \leftarrow i + 1$
9: **end while**

$APPROXIMATE\_SIZE$ is not trivial since we do generally not have the simple situation where the component is a box which edges align with the edges of the configuration space, it would not be difficult to find the size of the box. The length of each edge could then just be found using the variance along that direction. Realistically, we do not know anything about the shape, but we assume that it would be roughly rectangular and of lower dimension than the configuration space. If this rectangle is not aligned with the axis of the configuration space we have a problem with the previous approach of directly applying the transformation. The covariance matrix can have off-diagonal values implying non-uniformity along the axis. We, therefore, try to find a transformation of our samples such that their axes align with the edges of the rectangular component. In other words, we want to find the principal components. The principal components are found using the eigendecomposition of the **Cov** [And03].

$$\mathbf{Cov} = Q\Lambda Q^{-1} \tag{3.4}$$

From this we gain the matrices $Q, \Lambda \in \mathbb{R}^n$. The formers columns correspond to the i-th eigenvector q_i and latters i-th entry along the diagonal to the i-th eigenvalue λ_i. λ_i furthermore corresponds to the variance Var_i in the direction of q_i. Using the definition of the variance of a uniform distribution (equation 2.22), we can find the w the length of the interval of support:

$$\lambda = \sigma^2 = \frac{1}{12}w^2 \Rightarrow w = 12\lambda^{\frac{1}{2}} \tag{3.5}$$

Given that the manifold of interest is of dimension k and $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ we can then approximate the volume A.

$$A = \prod_{i=1}^{i=k} w_i = 12 \prod_{i=1}^{i=k} \lambda_i^{\frac{1}{2}} \tag{3.6}$$

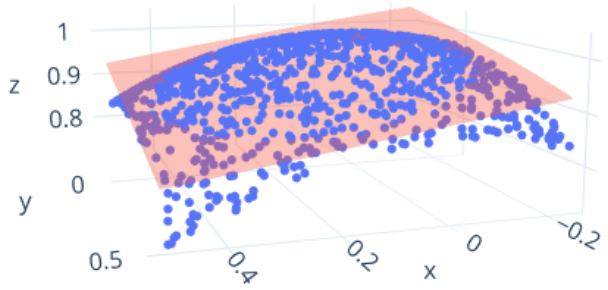

Figure 3.3: Uniform samples over a curved manifold (blue) and corresponding rectangular surface to approximate the size for.

In figure 3.3 uniform samples over a curved manifold with boundary are displayed. Approximating its size can be imagined as projecting the samples onto the red plane, finding its edges' length, and computing the area.

After having the size of the chains' manifold components estimated, the probability of sampling any of the chains will be updated with $CHOOSE\_NEXT\_CHAIN$. Large components are more likely to be sampled. We therefore define the discrete random variable X that can take values $x \in \{1, ..., n_{\text{chains}}\}$. The probability of sampling chain i next is then:

$$P(X = x_i) = \frac{A_i}{\sum_{i=1}^{i=n_{\text{chains}}} A_i} \tag{3.7}$$

If the estimated sizes A_i would be the true size, then for $n_{\text{total}]}$ the total amount of samples distributed among the chains and using 3.9 the expectation of the density ρ_i of samples per manifold chain would be:

$$\mathbb{E}(\rho_i) = \frac{\mathbb{E}(n_{\text{total}} P(X = x_i))}{A_i} = \frac{\mathbb{E}(n_{\text{total}})}{\sum_{i=1}^{i=n_{\text{chains}}} A_i} = \mathbb{E}(\rho) \tag{3.8}$$

If further the assumption holds that the samples are distributed uniformly over the individual manifold components and that there is only one chain on each component, we can further conclude that the probability density f_i over manifold component i would be constant and uniform:

$$f_i = \rho = \text{const.} \tag{3.9}$$

3.3.4 AIPS

Computational efficiency is an important aspect when designing and choosing an algorithm for a specific problem. The above-described algorithms differ in how sampling from the

feasible set $\mathfrak{X}$ is approached and undoubtedly differ in their computational efforts. The average iterations per sample (AIPS) is a metric that describes these computational efforts. It is based on the fact that all algorithms studied in this thesis have in common that they use biased optimization. An iterative method solves the biased optimization problem, and the number of iterations n_{iter} reflects how computationally expensive solving this problem was. AIPS is an aggregate measure over the set:

$$N_{\text{iter}} = \left\{ n_{\text{iter},i} \in \mathbf{N} : i \in \{1, 2, ..., n_{\text{opts}}\} \right\} \tag{3.10}$$

Here n_{opts} stands for the number of times an optimization problem was solved during the runtime of the algorithm. AIPS will then be estimated the following:

$$\text{AIPS} = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{iter},i}} n_{\text{iter},i} \tag{3.11}$$

Because rejection can be applied it is important to notice that $n_{\text{samples}} \leq n_{\text{opts}}$.

4 Experiments

The purpose of this thesis is to study various algorithms that have been introduced previously. These algorithms are:

- i.i.d.-Biased-Sampler
- RRT-on-Tangent (will be referred to as RRT from here on)
- Grid-Walk-on-Tangent (will be referred to as Grid-Walk from here on)
- Composite-Sampler

This shall be done both qualitative and quantitatively by systematically applying the algorithms in their various forms in different scenarios. The performance of a specific algorithm is expected to be influenced by the following factors:

- The location and shape of the manifold
- The compactness of the manifolds
- Algorithm-specific parameter choices

Furthermore, the experiments are varied in the dimension of the configuration space. Experiments in lower dimensions are conducted so that visual representations can be found and an intuitive understanding of the arising phenomena is gained that then can be abstracted to larger dimensions.

The experiments can be grouped into:

1. Linear manifold in $\mathbb{R}$
2. Linear manifold in $\mathbb{R}^2$
3. Sphere manifold in $\mathbb{R}^3$
   a) Centered and connected
   b) Off-Center and connected
   c) Off-Center and disconnected
4. Robotics Examples
   a) 3-DOF end-effector
   b) 7-DOF end-effector

4.1 Linear Manifold in $\mathbb{R}$

Purpose

We mean to study how samples are distributed along a line using RRT and Grid-Walk. Particular focus is put on how samples accumulate near the borders of the feasible space. Furthermore, will we look at how these compare between the application of Rejection Sampling and Projection.

Setup

The configuration space are the elements in $\mathbb{R}$. The feasible space $\mathcal{X}$ corresponds to the unit line segment.

$$\mathcal{X} = \{x \in \mathbb{R} : 0 \leq x \leq 1\} \tag{4.1}$$

An initial seed x_0 is sampled uniformly over $\mathcal{X}$ and expanded using RRT and Grid-Walk for varying parameters. The Neighborhood width w_{GW} parameter for Grid-Walk is varied. The neighborhood size is also varied for RRT. We specifically vary a parameter b that describes how far the region over which the RRT is built up extends over the edges of the feasible set (for example, if $b = 0$ the sampling region for the RRT is precisely the unit line segment if $b = 0.1$ the sampling region is $X_{\text{sampling}} = \{x \in \mathbb{R} : -0.9 \leq x \leq 1.1\}$).

Each algorithm-rejection/projection-parameter combination creates a set S of size $n_{\text{samples}} = 1e6$.

Cellular Sampler

A theoretical approach is furthermore applied in this setting. Therefore, a discrete system referred to as cellular sampler (inspired by cellular automata [Wei]) is introduced that may reflect a simplified version of the problem for which some properties can be analytically solved, such as the transition probabilities and the stationary distribution. This is meant to complement the previous empirical experiments.

A cellular sampler is designed as follows. The feasible set is split into n_{cell} equally sized cells and we introduce a discrete Markov-Process with a state-space X_{cell}.

$$X_{\text{cell}} = \{x \in \mathbb{N} : 0 \leq x \leq n_{\text{cell}} - 1\} \tag{4.2}$$

The state $x(t) = m$ represents that a sample has been generated at time t in cell m. Given this definition of a state, we can further specify a transition matrix $\boldsymbol{P} \in \mathbb{R}^{n_{\text{cell}}, n_{\text{cell}}}$ and find its stationary distribution $\mathbf{P}_{\text{stat}} \in \mathbb{R}^{n_{\text{cell}}}$.
The transitions matrices for Grid-Walk and RRT with Rejection and Projection are defined below.

Grid-Walk Projection

$$\boldsymbol{P}_{i,j}^{\text{Proj}} = p(i,j) = \begin{cases} \frac{1}{w}, & \text{if } d_{i,j} \leq k \text{ and } j \notin \{0, n_{\text{cell}} - 1\} \\ \frac{1+k-d_{\text{edge}}}{w}, & \text{if } d_{i,j} \leq k \text{ and } j \in \{0, n_{\text{edge}} - 1\} \\ 0, & \text{otherwise} \end{cases}$$

Grid-Walk Rejection

$$\boldsymbol{P}_{i,j}^{\text{Rej}} = p(i,j) = \begin{cases} \frac{1}{w}, & \text{if } d_{i,j} \leq k \text{ and } d_{\text{edge}} > k \\ \frac{1}{w-k+d_{\text{edge}}}, & \text{if } d_{i,j} \leq k \text{ and } d_{\text{edge}} \leq k \\ 0, & \text{otherwise} \end{cases}$$

The Grid-Walk algorithm is characterized by a neighborhood width w_{GW} which is discretized as well such that $w_{\text{GW}} = 2k + 1$, such that $k \in \mathbb{N}$ is the amount of cells to the left or right of the current state that are still part of its neighborhood. Furthermore we define $d_{\text{edge}} \in \mathbb{N}$, the distance of the current state to the nearest edge in cells such that $d_{\text{edge}} = min(i, n - i - 1)$. Then $d_{i,j}$ is the distance in cells between i and j such that $d_{i,j} = |i - j|$.
The resulting transition matricies are displayed in figures 4.2 and 4.1.

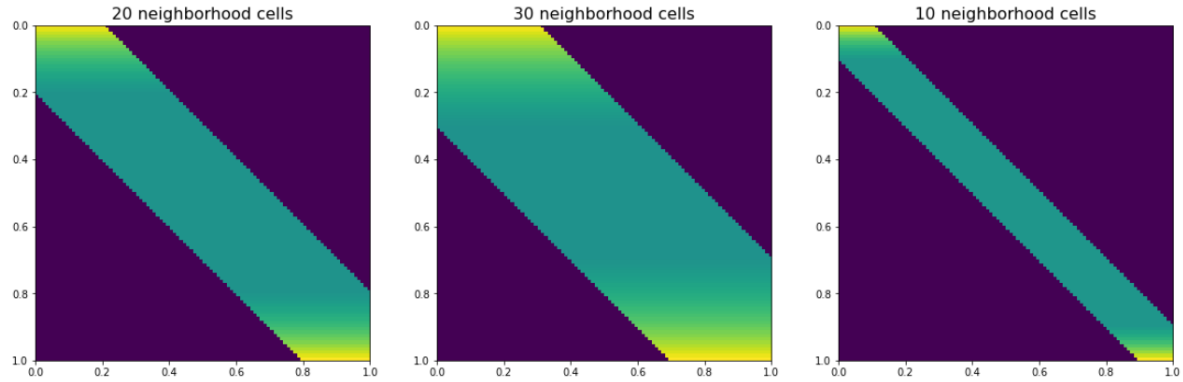

Figure 4.1: Transition matrix of the discrete Grid-Walk with rejection for various w_{GW}.

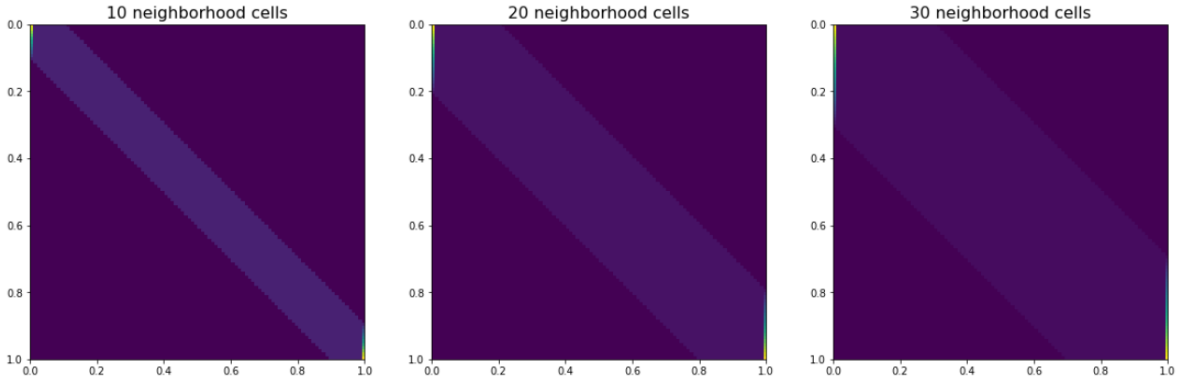

Figure 4.2: Transition matrix of the discrete Grid-Walk with projection for various w_{GW}. Notice large transition probabilities $p(i,j)$ in the first and last column.

Grid-Walk Compact Manifold

$$\boldsymbol{P}_{i,j}^{\text{Closed}} = p(i,j) = \begin{cases} \frac{1}{w}, & \text{if } d_{i,j} \leq k \text{ or } k - d_{\text{edge}} > (n_{\text{cell}} - 1) - j \\ 0, & \text{otherwise} \end{cases}$$

Another transition matrix is defined that represents a closed manifold such as a circle. In this scenario with $n_{\text{cells}} = 10$, there is a transition probability to jump from one end of the unit line to the other whereby no boundary is realized. This may give some intuition about the sampling distribution over a closed surface. The transition matrix looks the following.

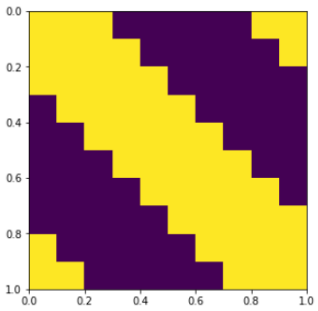

Figure 4.3: Transition matrix of a the unit line without bounds

RRT

A further simplification is made for the transition matrices of RRT. It is assumed that the RRT has grown just to the point where a vertice is present in each cell. Therefore, the initial phase where the RRT grows to the borders is neglected, which is reasonable since for a large number of samples, this phase will become less critical for the stationary distribution. For this thesis, it will only be observed that the stepsize α is equivalent to the size of a single cell. The transition probabilities for varying b can then be defined as follows:

RRT projection

$$\boldsymbol{P}_{i,j}^{\text{Proj}} = p(i,j) = p(j) = \begin{cases} \frac{b+1}{n+2b}, & \text{if } j \in \{0, n-1\} \\ \frac{1}{n+2b}, & \text{otherwise} \end{cases}$$

RRT Rejection

$$\boldsymbol{P}_{i,j}^{\text{Rej}} = p(i,j) = p(j) = \begin{cases} \frac{1}{2(n-1)}, & \text{if } j \in \{0, n-1\} \\ \frac{1}{n-1}, & \text{otherwise} \end{cases}$$

Evaluation

KDE is used to estimate the probability density for each attained set of samples along with the feasible set and averaged for each algorithm variation to approximate the expected probability density $\hat{f}$. The resulting $\hat{f}$ are compared among the parameter variations to get an understanding of their influence on uniformity and coverage. A visual display of the attained PDFs will be used.

The transition matrices for each algorithm will be analyzed for any patterns and differences between projection and rejection. The stationary distributions $\boldsymbol{\pi}$ are studied and compared to the previously empirically and for its continuous counterpart achieved $\hat{f}$ for similarities.

4.2 Linear Manifold in $\mathbb{R}^2$

Purpose

This scenario aims to study i.i.d.-Biased-Sampler as a mean to project a sample onto the feasible set. We try to find out if samples are projected in a structured manner and to gain some intuition about how the structure of the manifold and the configuration space may influence resulting distributions.

Setup

The configuration space is in $\mathbb{R}^2$. A linear equality constraint characterizes the feasible set. The resulting manifold is a line in the lower half.

$$\mathcal{X} = \{x \in \mathbb{R}^2 : -2 \leq x_i \leq 2; -\frac{2}{3}x_1 + \frac{4}{3} = x_2\} \tag{4.8}$$

i.i.d.-Biased-Sampler is then used to generate a set of seeds $X_{\text{seeds}} = \{x_{\text{seeds},i} \in \mathbb{R}^2 : i \in [1; n_{\text{samples}}]\}$ and a set of samples S, such that $x_{\text{seeds},i}$ acts as the bias for the optimization problem that converged to $x_{\text{samples},i} \in S$.

Evaluation

A parametric form of the equality constraint is found, and KDE is applied over the segment that belongs to the feasible set. This estimate of the PDF along the feasible set describes how frequent certain parts of the feasible set have been sampled and it will be studied which parts are sampled the least and the most. Furthermore, we will visualize the lines from $x_{\text{seeds},i}$ to $x_{\text{samples},i}$ as a reflection of which trajectory the sampled had to take.

4.3 Sphere Manifold in $\mathbb{R}^3$

Since we expect that the location and compactness of the manifold have a significant influence on the applied algorithms, the 3-Sphere scenario is divided into three sub scenarios. In the first, a selection of algorithms is studied for a closed sphere manifold that lies centered in the configuration space. The second scenario places the closed sphere in the corner of the configuration space, and the third furthermore divides the sphere into 8 different manifold

parts by introducing quadratic inequality constraints. A summary of the used parameters can be taken from the appendix.

4.3.1 Centered and Connected

Purpose

i.i.d.-Biased-Sampler, RRT, and Grid-Walk are compared in an idealized setting. This scenario is furthermore meant to establish a baseline for the following scenarios where the sphere is off-center and study the effect of the manifold in the center of the configuration space.

Setup

For this scenario the feasible set $\mathcal{X}$ can be defined the following:

$$\mathcal{X} = \{x \in \mathbb{R}^3 : -2 \leq x_i \leq 2; x_1^2 + x_2^2 + x_3^2 = 1\} \tag{4.9}$$

The configuration space is, therefore, a cube, and the feasible set is a sphere of unit radius (see figure 4.4).
i.i.d.-Biased-Sampler, RRT with multiple seeds (composite sampler), and Grid-Walk with a single seed are used to sample the above described feasible space.

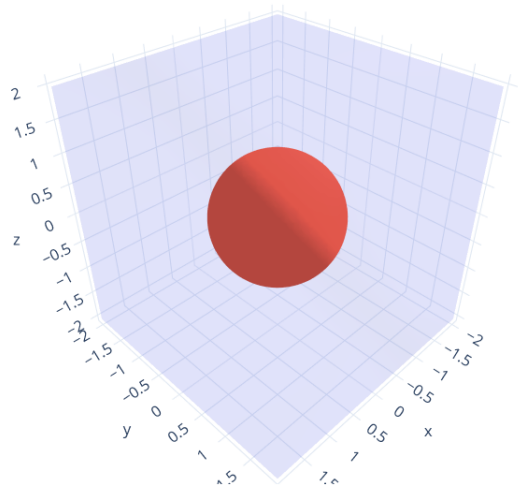

Figure 4.4: Center-Connected: Configuration space and feasible manifold

4.3.2 Off-Center and Connected

Purpose

The closed sphere is moved to a corner of the configuration space to study how an asymmetrical setup would influence the resulting distributions. In this scenario, the filter (11) is introduced to make conclusions about how the filter affects uniformity and coverage for the studied algorithms. Furthermore will Grid-Walk be applied for multiple seeds simultaneously.

Setup

For this scenario the feasible set $\mathcal{X}$ can be defined the following:

$$\begin{aligned} \mathcal{X} &= \{x \in \mathbb{R}^3 : b_{\text{low}} \leq x \leq b_{\text{up}}; x_1^2 + x_2^2 + x_3^2 = 1\} \\ b_{\text{low}} &= \begin{bmatrix} -3 \\ -2 \\ -4 \end{bmatrix}, \; b_{\text{up}} = \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix} \end{aligned} \tag{4.10}$$

The configuration space is now a cuboid with edges of length 7, 6 and 5.It is displayed in figure 4.5.

i.i.d.-Biased-Sampler, RRT and Grid-Walk with and without a filtering of the initially sampled seeds are used in this scenario.

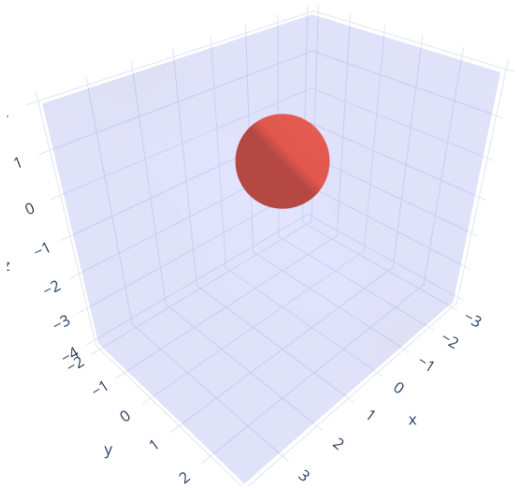

Figure 4.5: Off-Center-Connected: Configuration space and feasible manifold

4.3.3 Off-Center and Disconnected

Purpose

Previously we have been concerned with a closed manifold. Generally we are also interested in disconnected manifolds. In robotics application we often encounter such manifolds due to obstacles in the configuration space and other constraints on the joints. Therefore, the sphere is cut into various parts so that a more realistic situation can be studied. In this scenario, the bandit approach is furthermore introduced. Overall we want to use the results from this set of experiments to make statements about if a manifold is closed affects the performance of the algorithms and if the introduction of constraint handling and the bandit approach may solve any upcoming challenges.

Setup

The configuration space and overall shape of the sphere are the same as above. A set of quadratic inequalities $g_l(x)$ are nonetheless introduced.

$$\mathcal{X} = \{x \in \mathbb{R}^3 : b_{\text{low}} \leq x \leq b_{\text{up}};\ x_1^2 + x_2^2 + x_3^2 = 1;\ g_l(x) \leq 0\ \forall\ l \in \{1,2,3\}\} \tag{4.11}$$

$$b_{\text{low}} = \begin{bmatrix} -3 \\ -2 \\ -4 \end{bmatrix},\ b_{\text{up}} = \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix} \tag{4.12}$$

$$\begin{aligned} g_1 &= -5x_2^2 - x_3 + 1.2 \\ g_2 &= -5x_3^2 - x_2 + 1.2 \\ g_3 &= -100x_1^2 - x_3 + 2 \end{aligned} \tag{4.13}$$

As can be taken from the figure 4.6 the resulting manifold is split into 8 triangular parts of different sizes and varying distances between the parts.

Again we apply variants of RRT and Grid-Walk, all of which apply a filter operation. The variants applied are using Rejection, Projection or Rejection combined with a Bandit-Sampler.

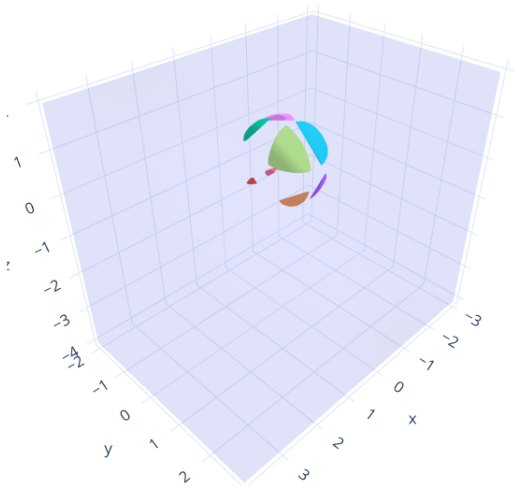

Figure 4.6: Off-Center-Disconnected: Configuration space and feasible manifold

4.3.4 Evaluation

As stated before it is of interest to compare the resulting sampling distributions for uniformity, coverage and computational efficiency.

Each algorithm-scenario-experiment pair is repeated for $n_{\text{runs}} = 100$ times. An individual run generates $n_{\text{samples}} = 5000$ sample. For each so achieved set of samples $S_i^{\text{algo,scenario}}$ for all $i \in \{1, 2, ..., n_{\text{runs}}\}$ a KDE is used to evaluated for the entropy H, variance Var, ADTS and AIPS

Coverage in terms of ADTS is evaluated as been described here 2.2.9. The reference set R of uniformly distributed points is created by reparameterizing the sphere and directly sampling the manifold uniformly. In presence of inequality constraints and disconnected manifolds a rejection sampling is applied to discard all infeasible samples. The reference set of points has same size as the sample sets.

Deserno describes how this reference set can be sampled defining the spherical coordinates of a sphere with unit radius [Des04]:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \sin\theta\ \cos\phi \\ \sin\theta\ \sin\phi \\ \cos\theta \end{bmatrix} \tag{4.14}$$

Here $\theta \in [0; \pi]$ is the polar angle and $\phi \in [0; 2\pi]$ is the azimuthal angle. Then we sample a

$x_3 \sim U(-1, 1)$ and a $\phi \sim U(0, 2\pi)$ and find the cartesian coordinates of our random point as:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} (1 - x_3^2)^{\frac{1}{2}} \cos\phi \\ (1 - x_3^2)^{\frac{1}{2}} \sin\phi \\ x_3 \end{bmatrix} \tag{4.15}$$

Next to the above metrics will the experiments be evaluated visually by plotting the probability density surface and/or the samples and seeds for a choice of experiments.

4.4 Robotics Examples

Next to the evaluation of the algorithms on the above-developed test scenarios, it is important to study their applicability on actual robotics examples. Two scenarios will therefore be introduced in this section. These scenarios both include a robot manipulator whose endeffectors x_{endeff} shall be positioned at a certain goal position x_{goal} subject to obstacles in the environment and the physical constraints of the design of the robot manipulator. The goal position is $x_{\text{endeff}} = f(\theta)$ with $\theta \in \mathbb{R}^{n_{\text{joints}}}$ the rotational angle of the j-th joint. The feasible which reflects all combinations of θ_j that realize that $x_{\text{endeff}} = x_{\text{goal}}$ can therefore be defined as:

$$\mathcal{X} = \{\theta \in \mathbb{R}^{n_{\text{joints}}} : -\pi \leq \theta \leq \pi, f(\theta) = x_{\text{goal}}, g(\theta) \leq 0\} \tag{4.16}$$

The function $g(\theta) : \mathbb{R}^{n_{\text{joints}}} \to \mathbb{R}$. Evaluates $g > 0$ if any of the robot parts touch the obstacle or another robot part. It guarantees the pose is physically feasible and no collisions. To study robotics examples we will evaluate a 3-DOF and a 7-DOF robot manipulator.

4.4.1 3-DOF

Purpose

First, a lower-dimensional robotics problem will be studied. This problem is similar to the above scenarios since the configuration space is in $\mathbb{R}^3$. The feasible set $\mathcal{X}$ is on a 1-d manifold and the feasible set can be visualized as a curve through three dimensional space and therefore easier analyzed than any higher-dimensional problems. The purpose of this experiment is therefore to use the above-described algorithms and find if the algorithms perform similarly in a rather real-world robotics example than above and to further find challenges in a concrete robotics example and what is causing such. Further the resulting manifolds will be visualized to gain some intuition of its shape. A summary of the used parameters can be taken from the appendix.

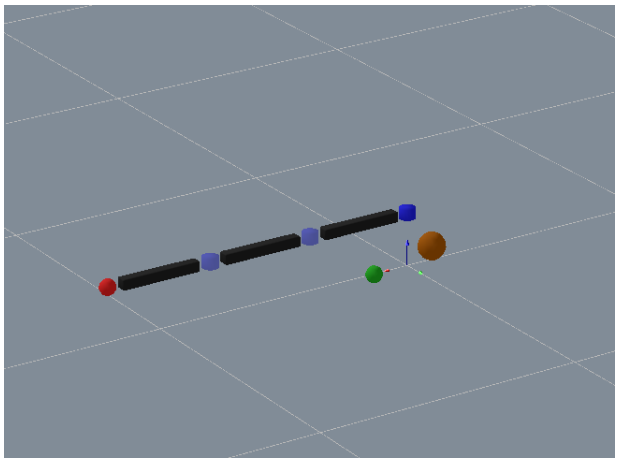

Figure 4.7: Setting of the 3-DOF robotics scenario. Obstacle (brown), target(green) and end-effector(red)

Setup

The initial environment looks like in image 4.7. The robot is attached to the ground and its part can move in a plane by rotation of the joints (blue). Three rigid body parts are attached to one another and the floor with a rotational joint. The blue ball is the target or goal position that the end-effector (red) shall be positioned at a brown ball is an obstacle. i.i.d.-Biased-Sampler, RRT with and without filter, Grid-Walk with and without filter are applied to generate $n_{\text{samples}} = 5000$ on the feasible manifold.

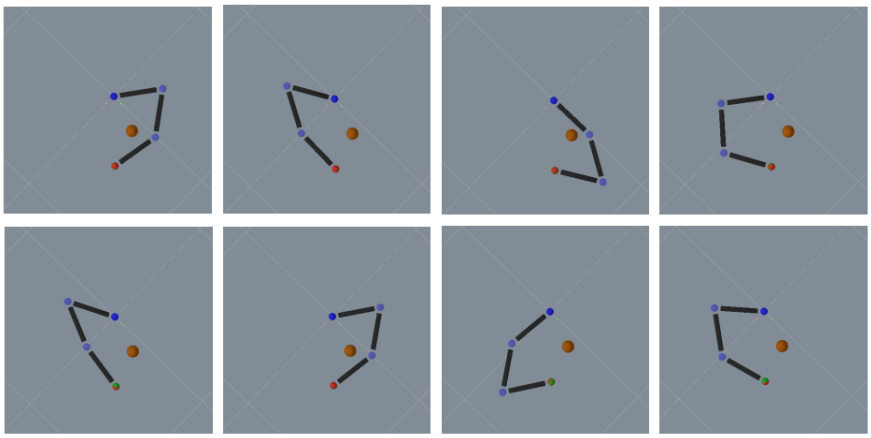

Figure 4.8: Various feasible configurations

4.4.2 7-DOF

Purpose

The purpose of this is the application on the model of an actual robot used in an industrial setting. A focus will be put on how higher dimensionality may influence the sampling set S.

Setup

The problem setting is shown in figure 4.9. The robot displayed is a model of an industrial Kuka robot with 7 rotational joints. The goal and end-effector position are represented as a yellow cube. The grey ball again serves as an obstacle.

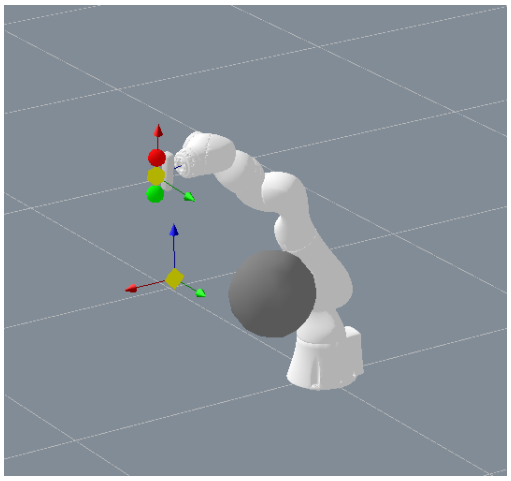

Figure 4.9: Setting of the 7-DOF robotics scenario. Obstacle (grey), target and end-effector(yellow)

As in the 3-DOF example i.i.d.-Biased-Sampler, RRT with and without filer, Grid-Walk with and without filter are applied. Since the configuration space is in $\mathbb{R}^7$ and we constraint the position a 4-d feasible manifold is present. A much larger amount of samples $n_{\text{samples}} = 100.000$ has to be drawn such that the estimated metrics are meaningful.

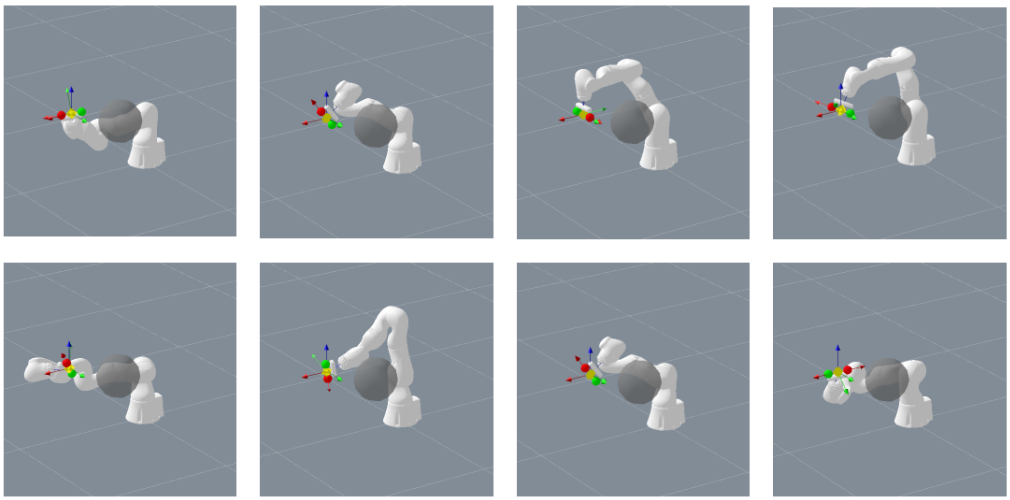

Figure 4.10: 7-DOF: Various feasible configurations.

4.4.3 Evaluation

As before we will evaluate the algorithms for uniformity by estimating their entropy H and variance Var of the KDE estimates $\hat{f}$. The $ADTS$ will be used as a measure of the coverage of the sample distributions. As described in 2.2.9 a uniform set R of points serving as reference is needed. Since we do not have a parametric description of the feasible manifolds we cannot apply rejection sampling to find a reference set of points as has been done in the previous experiments. Instead of a rejection sampled reference, we will create the reference set $R \subseteq S_{\text{all}}$ from the sample sets $S_a : a \in \{\text{RRT}, \text{GridWalk}, ...\}$.

$$S_{\text{all}} = \bigcup_a S_a \tag{4.17}$$

First, the union of all samples sets S_{all} will be created. The points in this set will be binned in n-cubes of edge length l. From each bin, a sample is then drawn. The set that is realized by binning and sampling from within the bins will first of all be reduced in size compared S_{all} and return a set that we assume to cover the feasible set well and be uniform. We choose l as follows

$$l_{\text{3-DOF}} = 0.025,\ l_{\text{7-DOF}} = 0.5$$

Choosing l like this results in the reference datasets approximately achieving the size of the sample sets.

Next to the qualitative evaluation, the sample sets will be subsampled and their poses displayed. These can then be analyzed for differences among the different sample sets.

4.5 Implementation

The experiments have been implemented in c++ it includes the above-introduced sampling algorithms as well as all supporting structures that facilitate the experiments, such as constraints and objective functions. Biased optimization has been realized using NLopt [Joh] to implement an augmented lagrangian method [CGT91].

The simulation environment for the robotics examples and capacities to solve the optimization problems have been provided as part of the rai package [Tou] and have been integrated with the code of the experiment module. Python has been used to implement the evaluation of the attained results, such as the estimation of the entropy and coverage. Most of the figures in this thesis have also been created using python and plotly. A web app has furthermore been written in dash and plotly for the interactive visualization of the results.

5 Results

The results achieved from the previously introduced set of experiments are described in this section. Visual representations of the results have been built both on an aggregate level over a set of experiment runs and for the defined metrics of coverage, uniformity, and computational performance. The resulting manifolds are also visualized and enable very detailed analysis. An in-depth discussion of these results and the significant findings and implications will follow after this section in 6.

5.1 Linear Manifold in $\mathbb{R}$

This experiment consisted of the application of RRT and Grid-Walk on a line in $\mathbb{R}$ with a feasible set $\mathcal{X} = \{x \in \mathbb{R} : 0 \leq x \leq 1\}$. The focus of this set of experiments was the behavior of the individual algorithms, and the results are therefore introduced per algorithm without any comparisons between the algorithms.

5.1.1 Grid-Walk

The neighborhood size w_{GW}, which describes the space from within which a new sample is drawn on each iteration, is the parameter that has to be chosen previously. A large neighborhood width w_{GW} would correspond to large jumps on the chain. A very small neighborhood would result in a chain that evolves slower—deciding if a sample is projected into the feasible set or if it shall just or rejection sampled is also of interest.

The estimated PDF $\hat{f}$ using KDE along the unit line is therefore displayed in figure 5.1 for $w_{\text{GW}} \in \{0.2, 0.4, 0.6\}$ and further divided for the use of projection (right) and rejection sampling (left). In the left figure, that displays $\hat{f}$ on a logarithmic scale for Grid-Walk using rejection sampling one can notice first of all that $\hat{f}$ varies for different values of w_{GW}. All curves show a rather flat trajectory over the central part of the feasible set. This roughly flat segment is wider and lower for decreasing w_{GW}. Near $x = 0$ does $\hat{f}$ increases linearly till it reaches the flat segment. Similarly does $\hat{f}$ decrease near the border of the feasible set at $x = 1$.

The first observation when studying the right figure 5.1, which corresponds to Grid-Walk using Projection is again that $\hat{f}$ varies with w_{GW}. The curve seems symmetric, and very distinct peaks can be seen at the borders. Two minima are located right next to the peaks on both sides. Moving further to the center, we notice a seemingly linear increase till another local maximum is reached. A slight dip is then followed by an almost flat line segment merging the two symmetrical segments of the curve. Regardless of the choice of w_{GW} does this structure (peak, down, up, dip, flat) occur. The peaks on the borders are greater for large w_{GW} while the flat segment is shorter and lower.

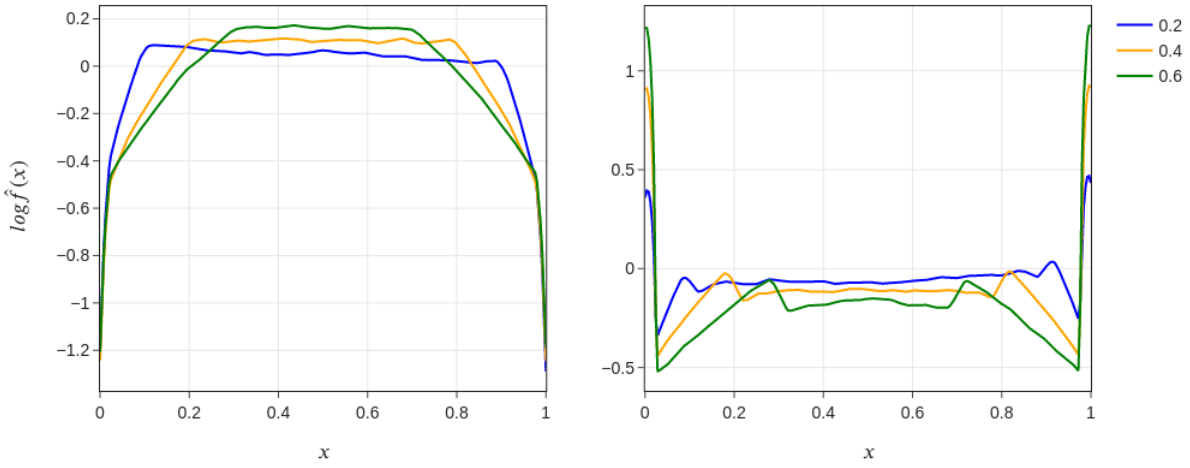

Figure 5.1: Expected $\hat{f(x)}$ for Grid-Walk over the unit line for various w. With rejection (left) and projection (right)

5.1.2 RRT

Figure 5.2 displays the log PDF $\hat{f}$ over the unit line when applying RRT with rejection (left) and projection (right). The parameter b corresponds to how far the configuration space extends over the edges of the feasible set. This parameter b has not been varied for RRT using rejection since it can be expected that it would not result in any variation of $\hat{f}$. The structure of $\hat{f}$ for RRT using rejection can be split into three segments, steep increase, slightly concave, steep decrease.

In contrast to using rejection, it can be expected that the parameter b somehow influences the resulting distributions. The curves in the right figure 5.2 therefore show different trajectories. The overall structure of the curves is peak, steep decrease, slightly concave, steep increase, peak. Peaks, as well as increase and decrease, seem to be symmetric. Considering the variation of b it can be noted that the curves corresponding to lower b show lower peaks on the borders but a higher located concave segment in the center. The inflection point where the steep in- and decrease meet the concave segment are at the same locations for all b.

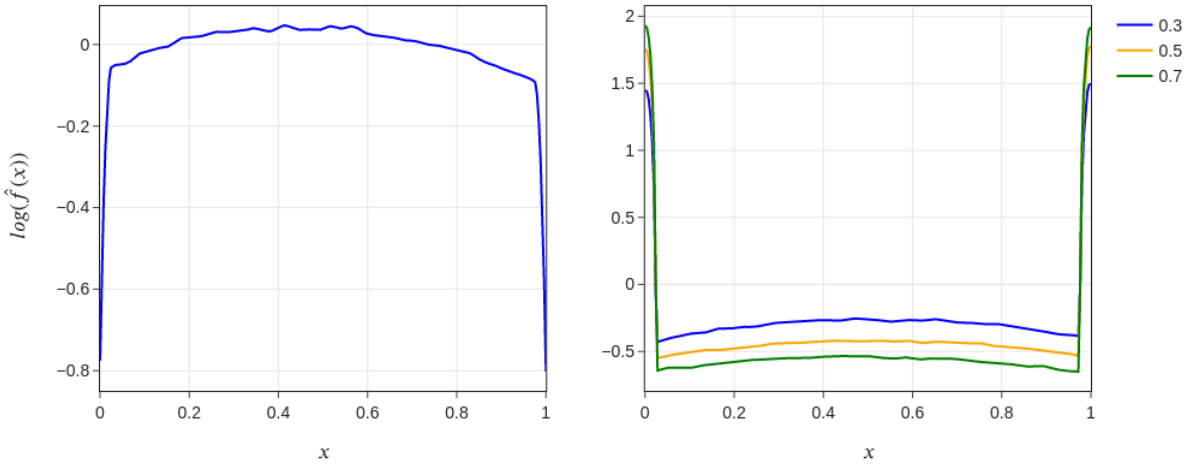

Figure 5.2: Expected $f(\hat{x})$ for RRT over the unit line for various b. With projection (left) and rejection (right)

5.1.3 Cellular Sampler

The results gained from the evaluation of the cellular samplers are meant to complement and verify the empirical results made before. Although the stationary distribution P_{stat} of the cellular samplers are discrete distributions it will be visualized by interpolation between the probability values, so that an easier comparison to figure 5.1 and 5.2 is possible.

Grid-Walk

Comparing figure 5.1 ($\hat{f}$ of Grid-Walk) and 5.3 (P_{stat} of Grid-Walk) similarities can be seen. Figure 5.3 furthermore shows the stationary distribution P_{stat} for various w_{GW}. The shape of the curves follow the same structure of slopes, dips and/or flat segments. A slight curvature can be noticed for Grid-Walk projection at the segments that follow/preceeds the maxima and the small dips.

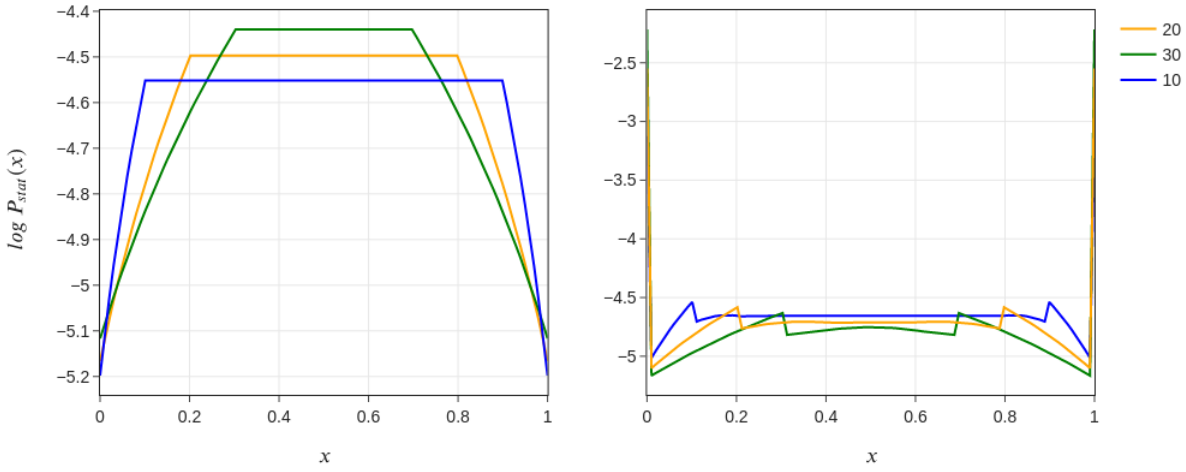

Figure 5.3: Stationary distribution of the discrete Grid-Walk with projection (left) and rejection (right)

The stationary distribution of the connected unit line that shall act as a discrete version of a compact 1-dimensional manifold resembles a uniform distribution as can be taken from figure 5.4.

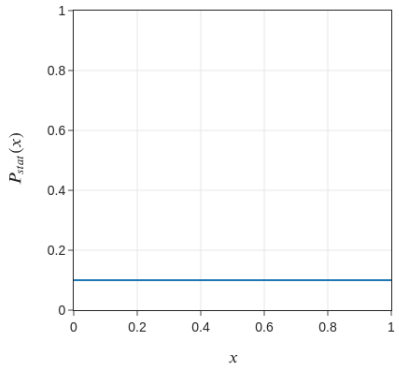

Figure 5.4: Stationary distribution of a closed discretized manifold

RRT

Again we see a very similar structure between the stationary distribution $P_{\text{stat}}(x)$ displayed in figure 5.5 and its empirically estimated counterpart (figure 5.2). Steep slopes mark the borders of the feasible set. Whilst $\hat{f}$ in figure 5.2 we notice a wide but slighty concave segment in the center, a flat and wide segment can be seen for the stationary distribution.

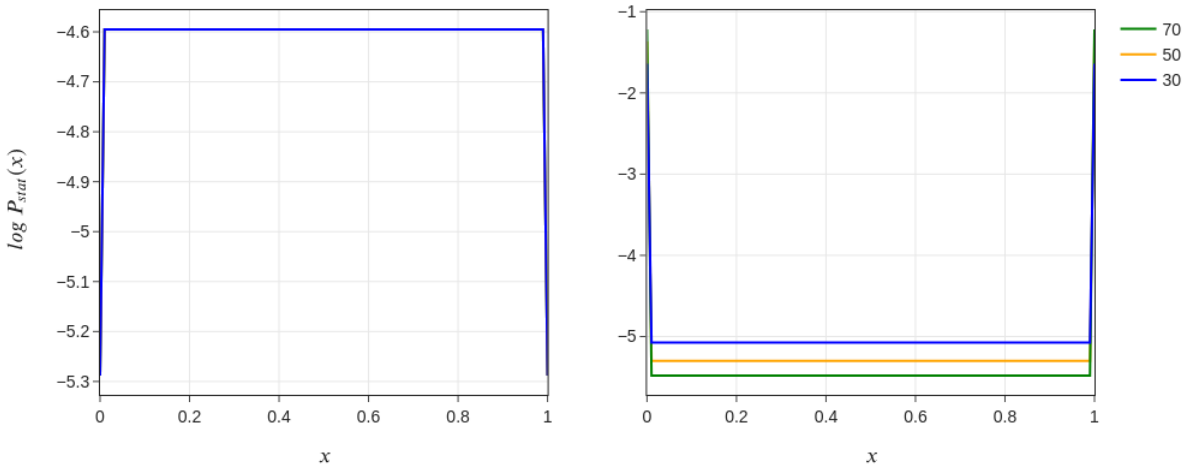

Figure 5.5: Stationary distribution of the discrete RRT with rejection (left) and projection (right)

5.2 Linear Manifold in $\mathbb{R}^2$

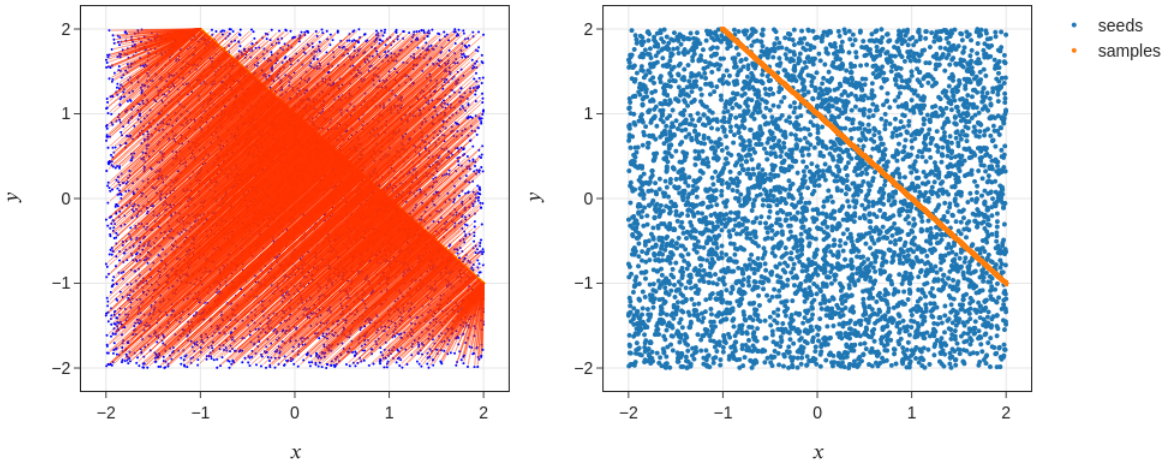

Figure 5.6: Seeds and projections onto a linear manifold using biased optimization

The i.i.d.-Biased-Sampler has been applied to a configuration space in $\mathbb{R}^2$ with a linear manifold in $\mathbb{R}$. Figure 5.6 shows both seeds and eventually generated samples and how the seeds have been projected onto the feasible set. It can be seen that the feasible set is a line that cuts through the upper right quadrant of the configuration space. The red lines connect

seeds and the feasible sample that solves the biased optimization problem. The configuration space is divided into different parts for easier description (figure 5.7). In parts 2 and 3, one can see that the edges connecting samples and seeds align parallel to one another, and all seem normal to the line representing the feasible set. In contrast, the edges in parts 1 and 4 are not parallel, but all point towards the same point where the feasible set meets the bounds of the configuration space. To study the distribution the PDF along the linear manifold is plotted in figure 5.8. Most striking seems that the borders of the feasible set achieve very distinct values compared to their immediate neighborhood. It can further be noticed that the PDF again peaks at the border of parts 2 and 3, but between the borders and this peak, a linear increase/decrease can be observed. Figure 5.9 displays $\hat{f}(x)$ and the feasible set $\mathcal{X}$ marked as a red square.

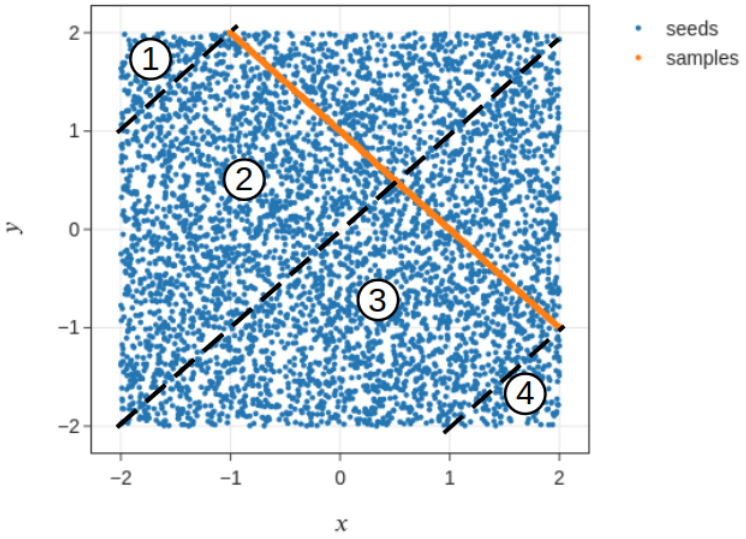

Figure 5.7: Labelled parts of the configuration space

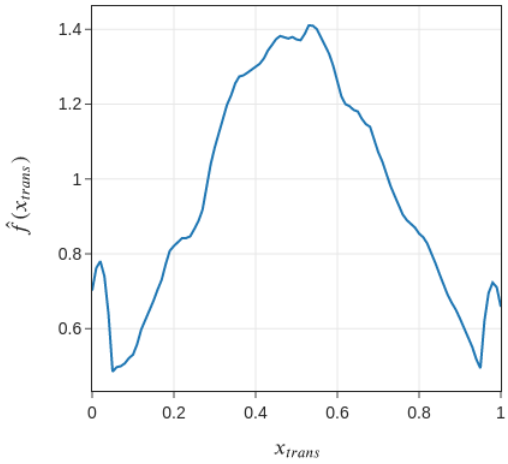

Figure 5.8: KDE estimate $\hat{f}$ over the feasible manifold

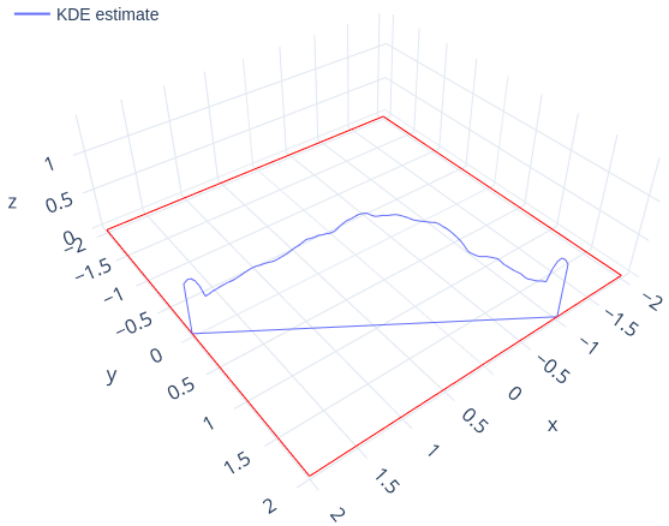

Figure 5.9: KDE estimate $\hat{f}$ in reference to the configuration space

5.3 Sphere in $\mathbb{R}^3$

A total of three scenarios are studied in this section. A sphere manifold in the center of a cube (the configuration space), the sphere moved towards a corner of the configuration space to break with the previous symmetry, and a sphere in the corner of the configuration space that is split into different triangular shaped parts. As described in 4.3 it is of interest to compare the algorithms within each scenario. We want to gain an understanding of what performance depends on. Furthermore, we want to understand how changes in the configuration space and the manifold itself change the behavior of the algorithms. Uniformity, coverage, and computational performance are observed via the metrics entropy $\hat{H}$, coverage in terms of ADTS , variance Var, and AIPS. For each scenario and algorithm, the distributions of these metrics are displayed using boxplots to gain insight into where the mean is located as well as quickly compare the spread of the values.

5.3.1 Center Connected

RRT, Grid-Walk (GW), and i.i.d.-Biased-Sampler (iid Biased) are compared for this scenario. In terms of ADTS, a clear distinction between RRT and the other algorithms can be seen in figure 5.10. Grid-Walk and i.i.d.-Biased-Sampler have distributions of ADTS located at lower values and not as widely spread as RRT. In terms of Var and entropy $\hat{H}$, there seems to be the same kind of pattern, such that RRT has the lowest values for entropy and the largest values for variance. Grid-Walk and i.i.d.-Biased-Sampler, on the other hand, show lower variance and higher entropy values. The distribution of entropy estimates is again very similar between Grid-Walk and i.i.d.-Biased-Sampler. Looking at the AIPS, a different picture arises. i.i.d.-Biased-Sampler seems an outlier with much larger values than RRT and Grid-Walk, which estimates distribute similarly. Table 5.1 can further be consulted for the mean and standard deviation of the metrics.

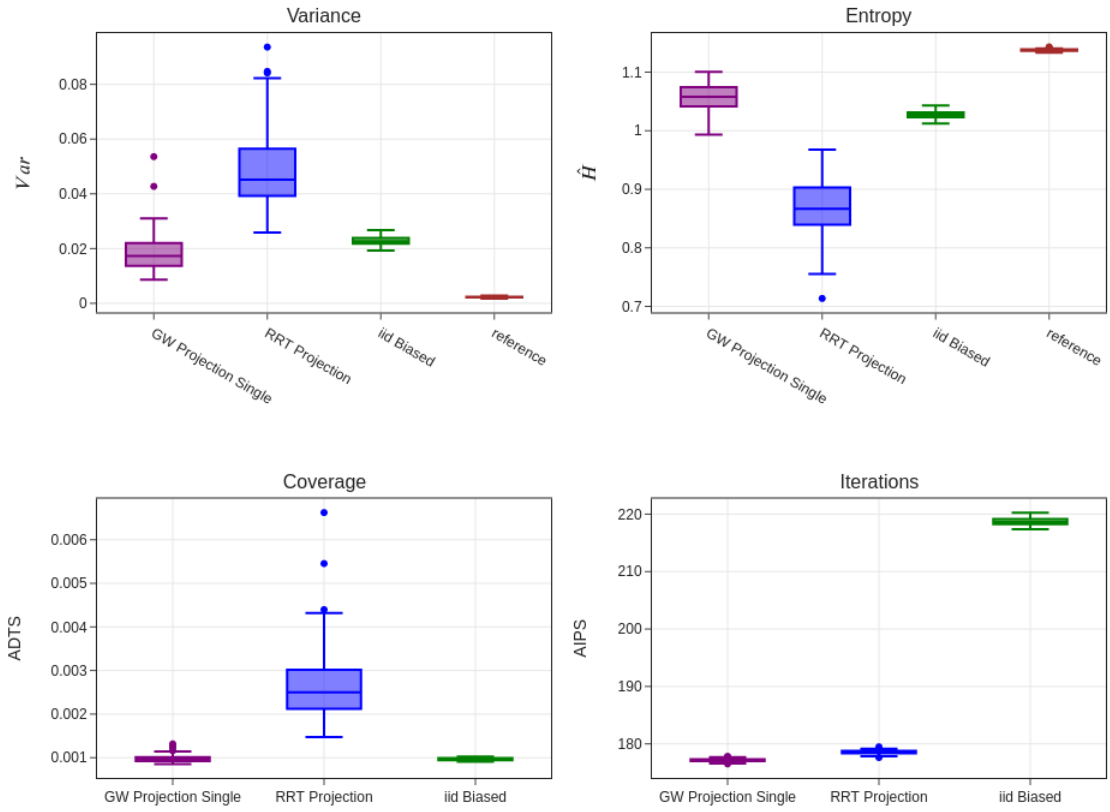

Figure 5.10: Center-Connected: Distributions of uniformity (entropy and variance), coverage and computational efficiency (iterations/AIPS).

	variance		entropy		coverage		iterations	
experiment	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
GW-Projection-Single	0.0185	0.0069	1.0562	0.0238	0.0010	0.0001	177	0.2651
RRT-Projection	0.0488	0.0139	0.8668	0.0439	0.0026	0.0008	179	0.3271
iid-Biased	0.0229	0.0016	1.0272	0.0062	0.0010	0.0000	219	0.6291
reference	0.0023	0.0002	1.1377	0.0018				

Table 5.1: Center-Connected: Evaluated metrics.

5.3.2 Off-Center Connected

The mean and standard deviations of the evaluated metrics can be found in table 5.2. A filter operation is introduced to the set of algorithms for this scenario. Grid-Walk is extended to use multiple seeds meaning multiple chains evolve simultaneously. These are again compared to i.i.d.-Biased-Sampler. In terms of uniformity, it can be seen in figure 5.11 that the application of Grid-Walk using multiple chains outperforms the other algorithms. Furthermore,

does the extension of a filter increase uniformity for both RRT and Grid-Walk. RRT without a filter and i.i.d.-Biased-Sampler appear to be the least performant. RRT nonetheless shows a much wider spread distribution. The i.i.d.-Biased-Sampler has less performance than Grid-Walk, but has a very narrow distribution.

In terms of coverage, three algorithms results appear to have very similar distributions. Grid-Walk-Multiple, Grid-Walk-Multiple-Filter, and i.i.d.-Biased-Sampler have lower ADTS values relative to RRT and RRT-Filter. RRT-Filter and RRT furthermore have much broader distributions than the previously mentioned. Evaluating how the algorithms compare computationally, it can be noted that most algorithms show similar performance. i.i.d.-Biased-Sampler is the only algorithm that needed more iterations to generate a sample on all experiment runs.

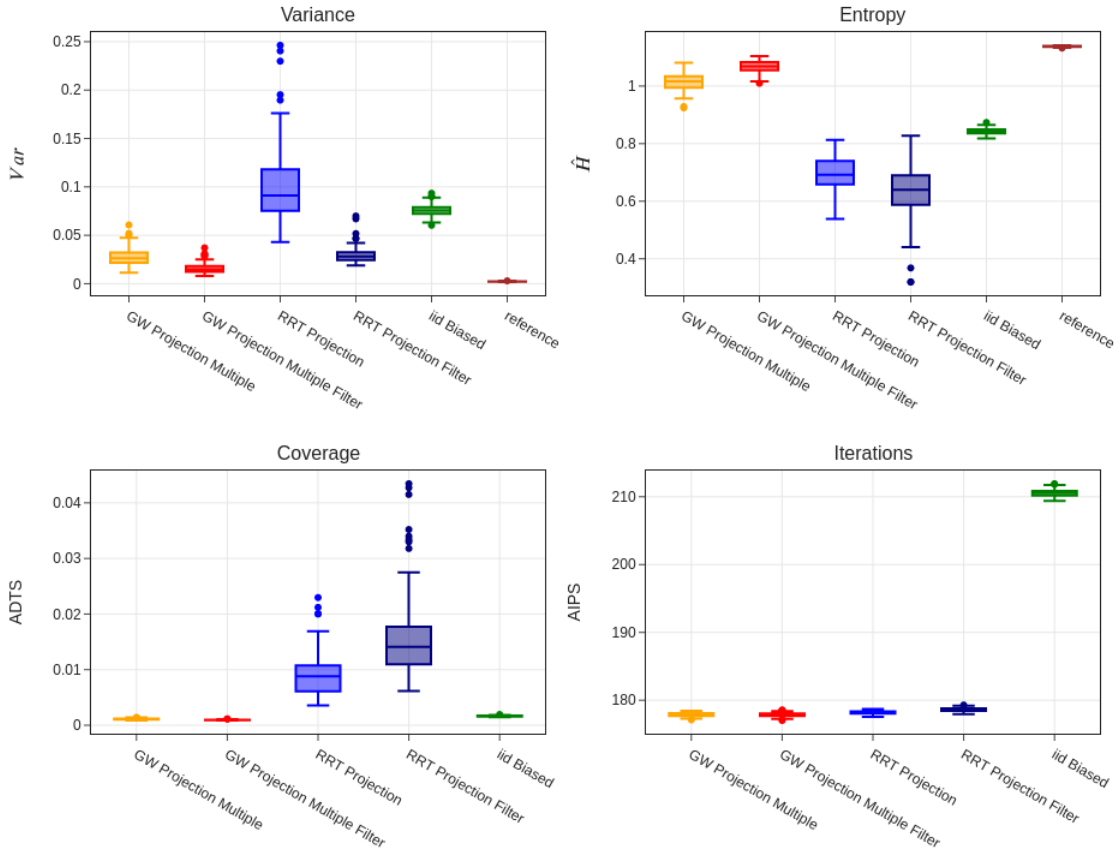

Figure 5.11: Off-Center-Connected: Distributions of uniformity (entropy and variance), coverage and computational efficiency (iterations/AIPS).

5.3.3 Off-Center Disconnected

As previously described, this scenario reflects a somewhat real-world situation since the feasible set is not only embedded on a manifold but also consists of individual components of

	variance		entropy		coverage		iterations	
experiment	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
GW-Projection-Multiple	0.0279	0.0091	1.0135	0.0306	0.0011	0.0001	177.89	0.27
GW-Projection-Multiple-Filter	0.0157	0.0054	1.0671	0.0211	0.0009	0.0001	177.86	0.28
RRT-Projection	0.1017	0.0395	0.6925	0.0600	0.0091	0.0039	178.17	0.25
RRT-Projection-Filter	0.0299	0.0084	0.6277	0.0888	0.0158	0.0077	178.62	0.26
iid-Biased	0.0761	0.0061	0.8426	0.0108	0.0017	0.0001	210.56	0.51
reference	0.0023	0.0003	1.1378	0.0019				

Table 5.2: Off-Center-Connected: Evaluated metrics.

different sizes and shapes. RRT and Grid-Walk are applied to this scenario with extensions such as Filter, Rejection, and Bandit sampling. Table 5.3 and Figure 5.12 show the distributions of the variances, entropies, ADTS, and AIPS for this scenario. The least uniform appearing algorithms apply the bandit approach for the selection of the chain to evolve on each iteration. On the other hand, the most performant seem both RRT and Grid-Walk using a filter. These are closely followed by the same algorithms extended using Rejection instead of Projection. In terms of uniformity, it can further be seen that the range in which the variance and entropy values spread differs quite substantially among the algorithms. Algorithms that, on average, perform worse also seem to have a much larger range of variance and entropy values. The best performing RRT and Grid-Walk version that uses projection and a filter seems to have a narrower distribution than their counterparts.

Comparing the algorithms for coverage Grid-Walk seems to perform better overall than RRT. Applying projection and filter again shows the best coverage. On the other hand, using rejection and the bandit approach cover the space less well. In terms of the distribution on an algorithm level, it can be noted that Grid-Walk-Projection-Filter, next to achieving overall best coverage, seems to have the most narrow distribution. The other versions of Grid-Walk and RRT show a wider spread distribution. Specifically, ADTS shows large values. Compared to Grid-Walk-Projection-Filter this algorithm's coverage distribution has a much longer tail.

Two distinct groups of algorithms can be established by studying the AIPS. Grid-Walk and RRT using projection need fewer iterations to generate a sample. All algorithms applying rejection instead of projection need more iterations to generate a sample.

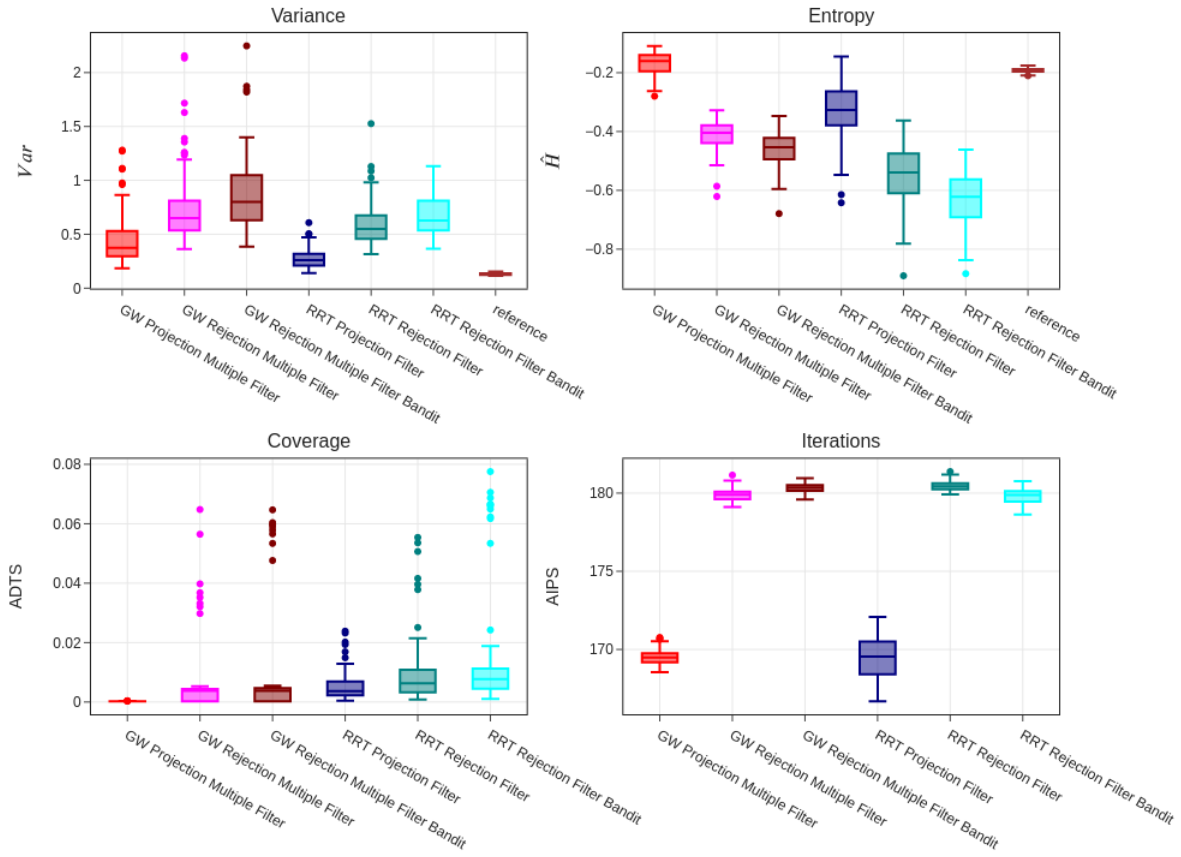

Figure 5.12: Off-Center-Disconnected: Distributions of uniformity (entropy and variance), coverage and computational efficiency (iterations/AIPS).

	variance		entropy		coverage		iterations	
experiment	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
GW-Projection-Multiple-Filter	0.45	0.23	-0.17	0.04	0.00024	0.00004	169.51	0.47
GW-Rejection-Multiple-Filter	0.74	0.33	-0.41	0.05	0.00540	0.01119	179.89	0.36
GW-Rejection-Multiple-Filter-Bandit	0.87	0.34	-0.46	0.06	0.00665	0.01517	180.32	0.28
RRT-Projection-Filter	0.27	0.08	-0.33	0.09	0.00534	0.00461	169.50	1.28
RRT-Rejection-Filter	0.58	0.19	-0.55	0.10	0.00947	0.01061	180.45	0.27
RRT-Rejection-Filter-Bandit	0.67	0.18	-0.63	0.08	0.01345	0.01805	179.79	0.48
reference	0.13	0.01	-0.19	0.01				

Table 5.3: Off-Center-Disconnected: Evaluated metrics.

5.4 Robotics Examples

The results from the experiments done on the 3-DOF and 7-DOF Robotics are presented in this section. The estimated variance Var, entropy H, coverage (in terms of ADTS), and AIPS

are displayed as tables in this section. i.i.d.-Biased-Sampler, RRT, RRT-Filter, Grid-Walk and Grid-Walk-Filter have been used on both robotics examples and it will briefly be pointed out which algorithms distinguish in terms of a specific metric. An elaboration of the causes of these differences will be done in section 6.4.

5.4.1 3-DOF

A few observations to make from table 5.4 are that the most uniform sample set has been generated by RRT using a filter as it achieved the best variance and entropy. i.i.d.-Biased-Sampler and plain RRT follow. Grid-Walk achieved the least uniformity. Generally, the variance and entropy do not differ by magnitudes among the algorithms.

In terms of ADTS, Grid-Walk and i.i.d.-Biased-Sampler generated S show less distance to the reference while RRT with and without filter seems more distant from the reference dataset.

Although i.i.d.-Biased-Sampler achieved good results in terms of coverage and uniformity it is much less computational performant with magnitudes more iterations needed to generate a single sample. Further points to note are that using a filter results in higher sample counts and that Grid-Walk has the smallest AIPS.

Algorithm	**Variance** Var	**Entropy** $\hat{H}$	**ADTS**	**AIPS**
i.i.d.-Biased-Sampler	0.182	-0.1	7.1e-5	341
RRT-filter	0.09	-0.13	1.4e-3	26
RRT	0.473	-0.11	1.9e-3	15
Grid-walk-filter	0.31	-0.19	4.1e-4	8
Grid-walk	0.2	-0.14	0.2e-4	5

Table 5.4: 3-DOF: Evaluated metrics for robotics example.

5.4.2 7-DOF

Table 5.5 summarises the metrics estimated from the experiments of the 7-DOF experiment. RRT-Filter and i.i.d.-Biased-Sampler rank best in terms of uniformity, with RRT having the lowest variance at Var = 0.09 and i.i.d.-Biased-Sampler having the highest entropy at $H = -0.1$ and is followed by RRT. The least uniform has been Grid-Walk. For RRT and Grid-Walk it can further be noticed that the variant using filter shows light improved variance and entropy. Compared to the 3-DOF experiments, it is also interesting to see that the differences among the algorithms are much greater in the 7-DOF scenario.

A similar pattern is observed for coverage, i.i.d.-Biased-Sampler also covers the feasible set best and is followed by RRT. This is confirmed when looking at figure 5.13 that shows the estimated cumulative density function (ECDF) of the distances from the reference to the sample sets. The opposite seems true for AIPS for which the ranking seems reversed. i.i.d.-Biased-Sampler has about nine times larger AIPS than RRT and 36 times as large as Grid-walk.

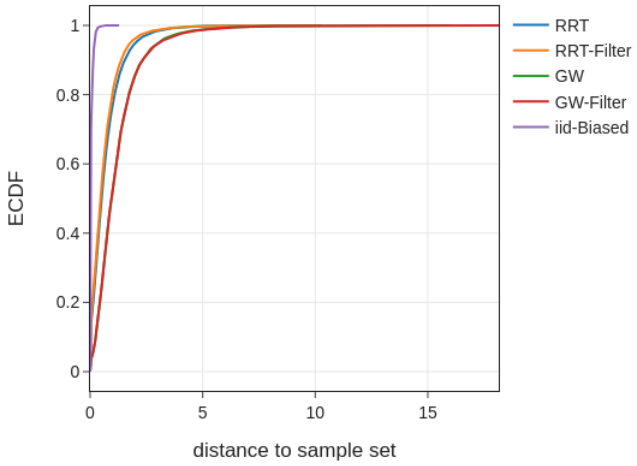

Figure 5.13: 7-DOF: ECDF of the distances to the sample set.

Algorithm	**Variance** Var	**Entropy** $\hat{H}$	**ADTS**	**AIPS**
i.i.d.-Biased-Sampler	7.77e-5	3.67	0.039	187
RRT-filter	0.018	1.41	0.621	21
RRT	0.03	1.23	0.687	20
Grid-walk-filter	0.069	-0.42	1.19	5
Grid-walk	0.064	-0.41	1.18	5

Table 5.5: 7-DOF: Evaluated metrics for robotics example.

6 Discussion

In this section, a critical assessment of the previously shown results will be made. The discussion's purpose is to explain and interpret patterns that occur in the results within the context of the stated research questions. The learnings and insights taken from this discussion are further meant to be used in the further research and development of efficient algorithms for constrained sampling. The structure of this chapter will follow the structure of the experiments. First, the unit line experiment and linear manifold in $\mathbb{R}^2$ are discussed. It is expected that the discussion of these experiments will bring forth insights on the sampling distribution over the feasible set that can later be abstracted to higher dimensions and integrated into the discussion of the sphere experiments. Eventually, the robotics examples will be discussed, focusing on the practicality of the algorithms and their application in high-dimensional spaces.

6.1 Linear Manifold in $\mathbb{R}$

6.1.1 Grid-Walk

First, Grid-Walk with projection will be discussed. A first observation that can be made when inspecting the results of the unit line experiment is the bowl-like shape of $\hat{f}$ the PDF and changing of the shape with varying sizes of the w_{GW}. It seems that the choice of w_{GW} is influencing the resulting PDF. Conceptually this makes sense when thinking of the Grid-Walk algorithm as taking a random step in the neighborhood of the current state. When the chain approaches an inequality constraint (a boundary) it is evident that a random step outside of the feasible set is much more probable when the w_{GW} is chosen wide. When using a narrow w_{GW}, many more iterations are needed to initially reach the area near the edge where a projection follows a step outside $\mathcal{X}$. This leads to a second important observation that the edges are spaces of particular interest since the mechanism of projection leads to an accumulation of samples in these spaces, which naturally leads to imbalances and decreasing uniformity over the feasible set. These conclusions become very intuitive when looking at the transition probabilities of the cellular sampler for jumping to the border (figure 4.2), which show high transition probabilities (that differ from the rest of the transition matrix) towards the edges.

The minima right next to the peaks in the right figure 5.1 are also a result of the presence of boundaries/inequality constraints. These are sampled the least since the space for which these are candidates of being sampled next is much smaller than for any point in the center. The local maxima in the first and last third of the unit line can be attributed to this particular point being a candidate for all its neighboring points plus the increased probability on the edge that is part of its neighborhood. A general takeaway is that certain points on the manifold will eventually accumulate more samples. These points do not necessarily have to be edges. Points that accumulate samples furthermore reinforce non-uniformity in the neighborhood. These effects are only local, and a uniform distribution is achieved when applying

Grid-Walk with increased distance from these points.

Similar conclusions can be drawn for the application of Grid-Walk using rejection sampling. Again it seems that the $\hat{f}$ is dependent on the w_{GW}. The edges also seem to be special points of interest where $\hat{f}$ goes to 0. Again the space surrounding these points is non-uniform. The PDF near the edges increases moving towards the center a distance of $d = \frac{w_{\text{GW}}}{2}$ is overcome, which marks the start of the nearly flat segments. The imbalance of the PDF $\hat{f}$ is only local and does not affect that the center of the line is more uniform than the edge regions. Overall we can conclude that using rejection removes the peaks on the borders and that with decreasing w_{GW} we could also increase the width of the uniform flat segment in the center.

The closed unit-line has only been evaluated as a cellular sampler. An important takeaway from the resulting uniform stationary distribution P_{stat} (figure 5.4) is that it matters if a manifold is closed. Grid-Walk is a good candidate to apply on manifolds, of which we know that it is closed. The uniform distribution would eventually be achieved over $\mathcal{X}$.

6.1.2 RRT

The results for RRT applied on the unit line showed a wide and almost flat segments in the center for both rejection and projection. It will be concluded that the algorithm theoretically has good properties of creating a uniform distribution.
The slight curvature that has been noticed in figure 5.2 is attributed to the chain having evolved for just a finite amount of samples. In the initial phase of the RRT-growth, the tree expands rapidly towards the edges, but there is still a probability that a step is taken back towards the root which increases the density of points near the root and this slight curvature. Since in the discrete case (see figure 5.5) we neglect the starting growth of the tree and are only concerned with the stationary distribution P_{stat} and therefore do not see these effects in the figures. The fact that $\hat{f}$ decreases for the projection approach (right figure 5.2) on the center segment with greater b that exceeds the feasible set reflects that choosing b, the bounds of the RRT have to be chosen properly in order to increase uniformity.

Nonetheless, it has to be kept in mind that the unit line example studied here only works very well since the bounds of the RRT exceed the feasible set. If the RRTs bounds would not exceed the feasible set the space outside of the bounds could not be sampled. This would lead to the situation where the RRT would only locally explore the feasible set at the cost of less coverage.

6.2 Linear Manifold in $\mathbb{R}^2$

The line experiment in $\mathbb{R}^2$ was meant to investigate the projection behavior of a biased optimizer and how it influences the sampling distribution S when paired with a uniform sampler to generate initial seeds/biases over the configuration space. First of all, it has to be noted that the discussion and inferences made here are only applicable to the studied scenario of a linear manifold. For this kind of scenario, one can easily see in figure 5.6 that the biased optimizer finds the nearest point in $\mathcal{X}$. As just mentioned, this may not be true for non-linear manifolds where the resulting optimization problem is non-convex, and it is not guaranteed

to converge to the global optimum.

A further observation is that there is a structure behind the resulting PDF over $\mathcal{X}$. Points where the manifold meets the bounds (or, in other scenarios, inequality constraints) seem to be of special interest for discussion. The peak of $\hat{f}$ (figure 5.9) at $x_{\text{trans}} = 0.5$ will also be discussed further. Generally, it had been noticed that many edges between $x_{\text{seed},i}$ and sample $x_{\text{sample},i}$ are parallel to each other and normal to the manifold. This makes sense since this would resemble the shortest distance and therefore the global optimum for the biased optimization problem when using a Euclidean metric. Since our configuration space is bounded and the manifold is not aligned with the axis, there are subsets of points that projections do not follow that pattern. For these subsets like 1 and 4 (as marked in figure 5.7), a corresponding feasible sample that lies on a line normal to the manifold cannot be found. An implication is that all seeds in these parts are projected onto the point where the border of the configuration space meets the feasible set. The accumulation of samples in these points leads to greater $\hat{f}$.

We can conclude that $f(x)$ on the manifold depends upon the space normal to the manifold relative to the total volume of the configuration space. Exceptions are special points of interest, such as the borders of the configuration space f. The piecewise linear behavior of f (excluding the borders) can also be derived from this. The size of the space normal to the manifold linearly increases until the normal would intersect with the corners of the configuration space and then again decreases linearly.

A significant takeaway from this is that first of all $f(x)$ is non-uniform and that there are particular points where the $f(x)$ peaks. Overall f is a function of the configuration space and how the manifold is situated within this space.

6.3 Sphere Manifold in $\mathbb{R}^3$

6.3.1 Center Connected

i.i.d.-Biased-Sampler and Grid-Walk have the best performance in terms of coverage and uniformity. An explanation for Grid-Walk's good performance may be that the manifold is closed and that with large enough samples and a proper choice of the w_{GW} parameter the sphere will be equally covered. A phenomenon such as the accumulation of samples in a particular point or subspace, as has been observed in the line manifold in $\mathbb{R}^2$ is not present here since the manifold is fully encapsulated within the configuration space and no edges exist. This reflects the behavior that was previously studied using a discrete system of a circle (see 6.1).

Considering the previous experiment in $\mathbb{R}^2$ it might be irritating why i.i.d.-Biased-Sampler works similarly well. On second thought, it makes sense since the manifold to be sampled is symmetric and centered in the configuration space such that the surrounding free space is relatively equally distributed around the manifold. Surely there is more space towards the corners, which is also reflected in the $\hat{f}$ of i.i.d.-Biased-Sampler as can be seen in figure 6.1, that shows higher PDF facing the corners. This nonetheless has a minor impact.

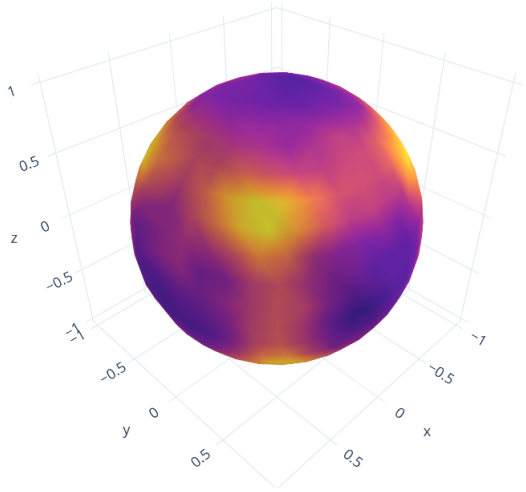

Figure 6.1: Center-Connected: KDE estimate $\hat{f}$ over the sphere manifold.

On the other hand, RRT covers the space less well and uniformly because the overall coverage and uniformity depend on the roots of the trees. If the roots are themselves not equally spaced over the manifold but lie near to each other, the RRTs can overlap (figure 6.2). If the roots have great distances between each other, there might be spaces of the feasible set that cannot be sampled since the individual trees bounds leave gaps in between. Overall it can be found that near the root of an individual RRT space is evenly sampled, but when considering the entire $\mathcal{X}$ distortions can appear due to overlapping trees or spaces not sampled at all.

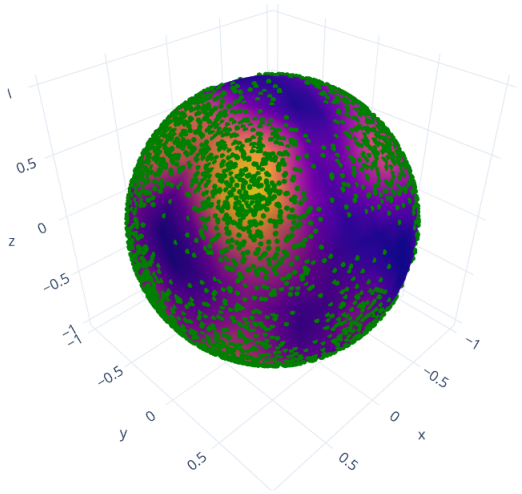

Figure 6.2: Center-Connected: Application of RRT. Notice that some spaces on the manifold are not sampled at all.

The differences in AIPS between i.i.d.-Biased-Sampler and the other algorithms can be derived from the seeds/biases that are sampled and then projected onto the manifold. Both for

RRT and Grid-walk these seeds are already near the manifold and a projection does not have to travel such a large distance. i.i.d.-Biased-Sampler samples uniformly over the whole configuration space and therefore more considerable distances to the manifold have to be overcome, resulting in more iterations. See figure 6.3 that displays seeds and samples generated during the application of a Grid-Walk. All seeds lie in close proximity to the manifold.

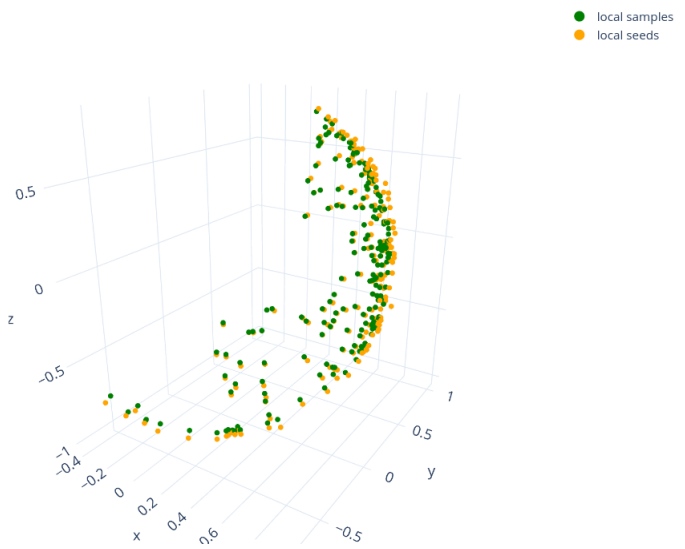

Figure 6.3: Seeds and resulting samples using Grid-Walk on the sphere manifold. The seeds are very near the samples.

6.3.2 Off-Center Connected

We have noted that i.i.d.-Biased-Sampler performance decreased in this scenario. This is explained using figure 6.4. It is clear that the distribution over the sphere is very distorted, and sample-hot-spots can be found, resulting from the sphere being located in a corner. The free space surrounding the manifold is not as evenly distributed as in the previous experiment. The consequence of the side of the sphere facing the large free spaces is sampled more often.

The just made conclusion can be transferred to applying the filter, which seems to improve the uniformity for Grid-Walk. Since the starting points of the Grid-Walk chains are sampled using the i.i.d.-Biased-Sampler we again achieve a somewhat distorted distribution of starting points that are clustered on spaces facing a large free space. In figure 6.5 the removed or filtered samples are displayed together with such that have been accepted. The accepted samples are much more evenly spread out over the sphere, resulting in a more uniform distribution that is eventually generated when evolving the markov chains.

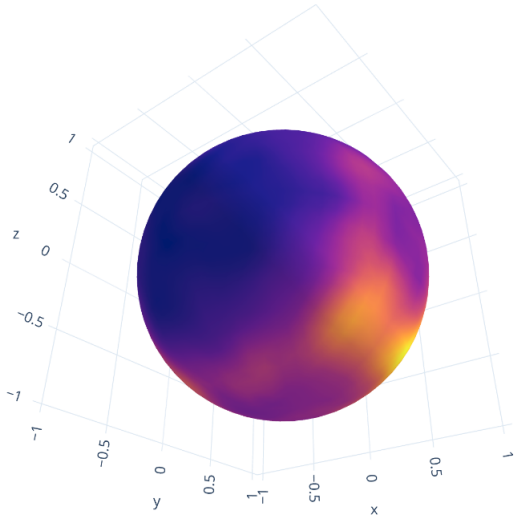

Figure 6.4: Off-Center-Connected: KDE estimate $\hat{f}$ over the sphere manifold. Notice the large blue parts that has low density.

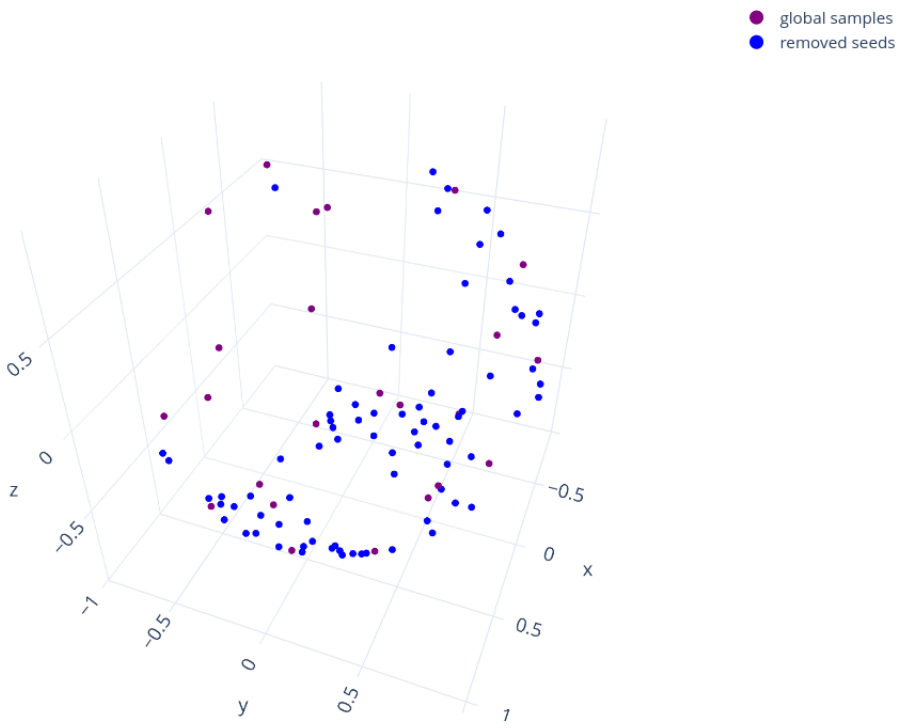

Figure 6.5: Off-Center-Connected: Accepted and removed global samples after application of filter.

Again a problem seems to occur with the application of RRT. The fact that RRT performs less well than Grid-Walk can be attributed to the parameter choices of RRT. Previous thoughts on

the overlapping of trees and not reachable subspaces of $\mathcal{X}$ will be discussed more thoroughly. Since it is necessary to bound the growth of each RRT, some subspaces of the feasible set do not fall within the bounds of any of the RRTs. Sampling these spaces then has a probability 0. This kind of situation can be reinforced when applying a filter. Some roots are filtered out, which increases the probability of creating spaces of the manifold that are not reachable. This, in turn, implies less uniform distributions and less coverage. Another situation that can occur for multiple RRT's grown simultaneously is when their roots are placed next to each other. Then the trees can overlap, resulting in an accumulation of samples in the overlapping spaces.

Theoretically, if one would know the shape of $\mathcal{X}$ the feasible manifold and choose the bounds w_{RRT} of the tree and the filter threshold β accordingly, fewer situations as above explained would occur. An alternative option would be to massively increase the bounds of the tree, hoping that the whole feasible set can at least be covered. However, this has limitations because the local seeds are sampled on the tangent space. The tangent space for the sphere will always be located on the outside of the sphere and assuming that a biased optimizer will always find the nearest feasible point a problem arises. The bottom half of the sphere (if the root of the RRT is located right on the top of the sphere) would not be sampled since the points of nearest distance all lie on the upper half of the sphere. This brings forth another problem that contributes to less uniformity when choosing large bounds of the RRT. As the seeds on the tangent move further away from the root the distance between tangent space and manifold also increases. This and the interplay with the sphere's curvature results in more dense sampling as we move away from the root of the RRT. In general, one does not have any prior knowledge of the manifold, and therefore the choice of these parameters is a delicate task. A take-away from this would be that RRTs performance is very sensitive to the choice of parameters and not as robust as Grid-Walk or i.i.d.-Biased-Sampler.

6.3.3 Off-Center Disconnected

In this scenario, two extensions were added to the algorithms, bandit-sampling and rejection-sampling, to address aspects like different sized manifolds and accumulation of samples on boundaries. The results nonetheless showed no improvement in terms of coverage and uniformity. RRT-Rejection and RRT-Rejection-Bandit, as well as Grid-Walk-Rejection and Grid-Walk-Rejection-Bandit did not achieve better coverage or uniformity than their counterpart using projection. In terms of coverage, it is clear that the edges of the manifolds are not covered well since samples are not projected onto the edge. It has been shown in the unit line experiment that this phenomenon can manifest for Grid-Walk when w_{GW} is chosen very wide. For RRT the same problems of spaces not sampled due to the chosen bounds w_{RRT} of the tree as described before also apply in the disconnected scenario. It seems like this may even be reinforced in the disconnected scenario using a filter. The filter's impact can be understood when thinking of a situation where the distance between two samples is less than the predefined threshold β. One sample lies near the boundary, the other in the center of the manifold. Since the filter chooses arbitrarily between these two samples, there is a 50 % chance of the sample in the center being removed, which could lead to a not reachable space in the center and less coverage (see the right figure 6.7). Furthermore, this would also decrease uniformity. The accepted sample near the border will become a root of the RRT. The feasible space that this RRT can cover is much smaller because of its location near the boundary and its tangent space extending over the boundary. If projection sampling

is used, many samples will be projected onto the boundary and if rejection sampling is used the feasible set that can be covered will be sampled much denser, both resulting in decreased uniformity. This makes RRT an algorithm that is very sensitive to the choices of parameters and extensions such as the filter.

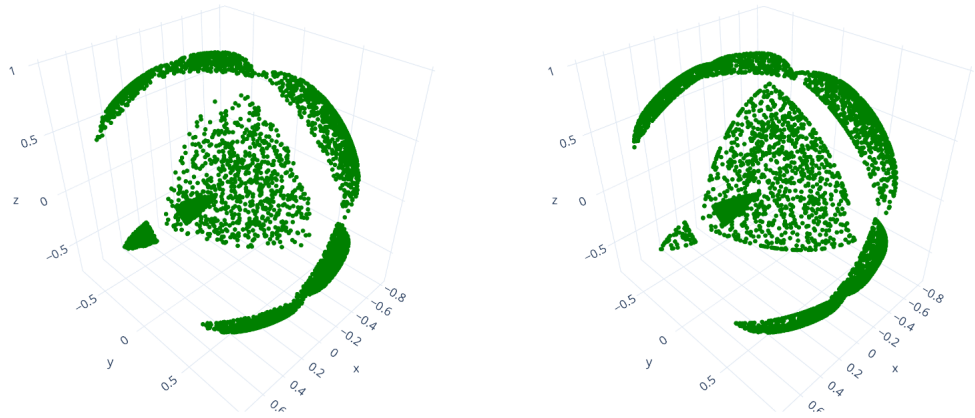

Figure 6.6: Off-Center-Disconnected: Using rejection sampling (left) and projection (right) using Grid-Walk.

Recall the previous unit line experiments that showed that using RRT-Rejection would result in a stationary distribution relatively uniform over most of the feasible set while projection results in non-uniformities near the boundary. That using rejection results in a more uniform distribution seems contradictory to the results achieved in this scenario where projection-based algorithms outperform rejection sampling extended algorithms. One has to keep in mind that the stationary distribution calculated reflects the shape of the distribution resulting from a Markov process of infinite steps. The experiments have nonetheless been run with a finite amount of samples. The effect of a finite number of samples can be seen in 6.9 which shows the projection of samples for a few RRTs on the boundary. The amount of times the RRT has been extended is reasonably small, and therefore only a few samples can be found on the boundary, which does not distort the uniformity as much as expected. This small proportion of projected samples on the boundary brings forth another critical consideration for designing a constrained sampling algorithm. Besides a uniform stationary distribution, uniformity should be guaranteed during the whole sampling process. In practice, the algorithm is run for a finite number of samples. If the distribution over the feasible set is very non-uniform for a few samples, it does not matter if its stationary distribution is perfectly uniform. Especially in industrial applications where the time an algorithms samples/runs may have economic implications.

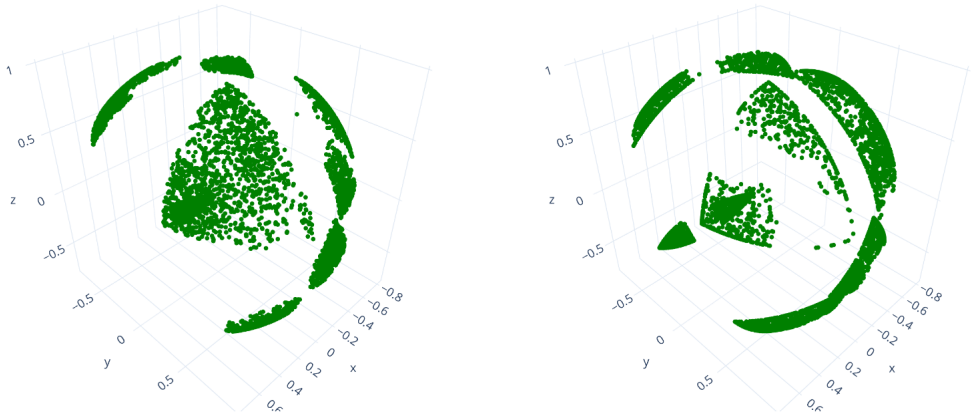

Figure 6.7: Off-Center-Disconnected: Rejection (left) and projection (right) using RRT.

A similar picture is displayed in figure 6.6 that shows Grid-Walk with rejection (left) and Grid-Walk with projection (right). The rejection managed to prevent the boundary sampling, but it still seems that projection-based Grid-Walk covers the triangular spaces better, especially the corners.

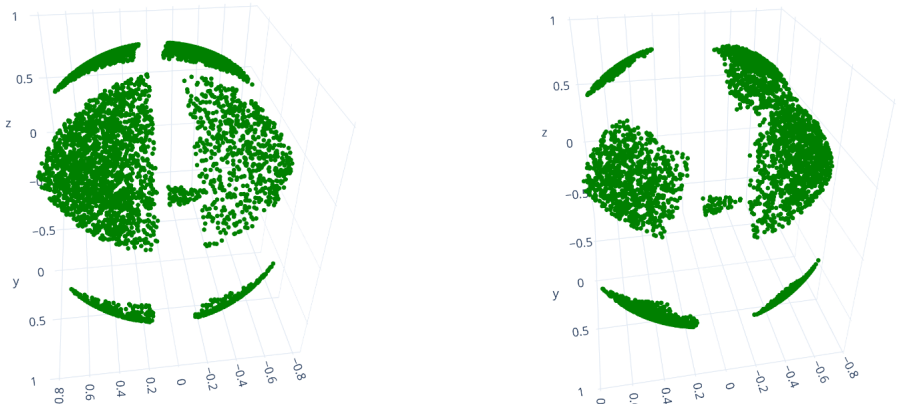

Figure 6.8: Off-Center-Disconnected: Application of Bandit-Sampling . Grid-Walk (left) and RRT (right).

For the Bandit-Sampler variants, a situation such that only a single seed is placed on each disconnected manifold was challenging to realize. Analyzing figure 6.8 it can be seen that the left disconnected manifold is much denser sampled than the left because more chains sample this manifold. The distribution over a single disconnected manifold can still be uniform if aggregating over all manifolds results in non-uniformities.

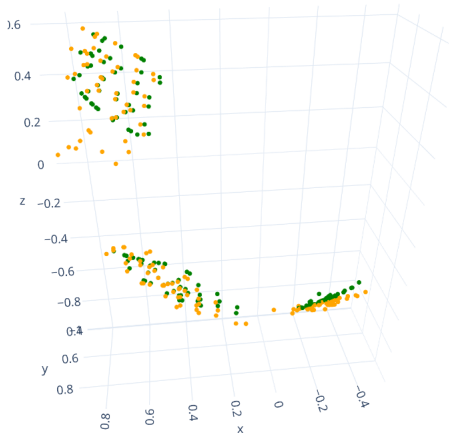

Figure 6.9: Off-Center-Disconnected: A detailed view of some samples generated using RRT.

Finally, the differences in the number of iterations needed to generate a sample (AIPS) between the algorithms shall be discussed. Clearly, rejection sampling extended algorithms needed more iterations to generate a sample. This is not a consequence of the seeds having more considerable distances to the manifold, as was the case with i.i.d.-Biased-Sampler, but rather with the fact that the proportion of samples rejected still undergo an optimization procedure which, of course, also contribute to the total amount of iterations and therefore also increasing the AIPS.

6.4 Robotics Examples

6.4.1 3-DOF

Many of the previously made observations and conclusions can be confirmed with the 3-DOF robotics experiments. As expected, the most computational efforts are needed for i.i.d.-Biased-Sampler, since the seeds are on average located further from the manifold. This also explains the different AIPS between Filter and non-Filter. In the 3-DOF scenario, more global samples have been generated when using a filter. This explains why Grid-Walk and RRT's variant using a filter resulted in a higher AIPS.

Also, previously explained phenomena of not reachable subspaces of the feasible set resulting from the combined use of Filter and RRT appear in the 3-DOF robotics example. See figure 6.10. It shows the sample sets S for i.i.d.-Biased-Sampler and RRT-filter, a gap in the manifold can be seen for RRT-Filter. This figure further explains why RRT then has such good uniformity. The samples on the manifold have distinct distances to another because a fixed stepsize is chosen for RRT, which leads to KDE estimating very similar values for these points. The distinct distances between the samples result from using RRT on a one-dimensional manifold. Given a root, it is predefined where vertices of the RRT may appear. The predefined vertice locations further strengthen the point that RRT is sensitive to its parameter choice. If the step size is too large, large empty spaces will appear between the samples. If the step size is too small, only a small proportion of the manifold is covered.

When relating the manifold structure and the robot's poses by comparing figures 6.10 and 6.11 we see that each disconnected manifold piece relates to a cluster of poses. All poses where the robot arm on the left correspond to a manifold, and all on the right correspond to the other manifold.

When further interpreting the images 6.11 it has to be kept in mind that these are only a subset of S. However, overall, all algorithms discovered the two clusters of poses, and only a few differences are noted. In figure 6.10 on the right we can notice a gap in the manifold, this can be attributed to the filter operation removing too many seeds, leaving part of the manifold uncovered by the RRTs. Similar irregularities can be seen in the other images, which are causes for the different results.

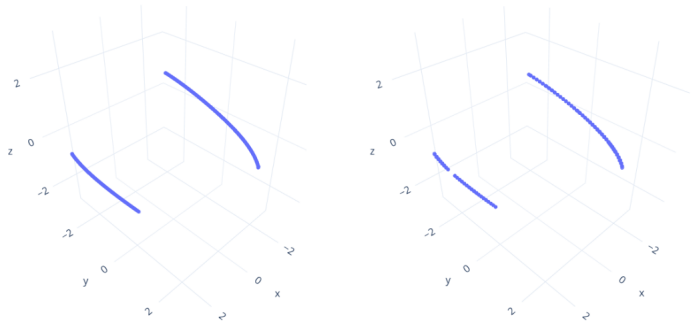

Figure 6.10: 3-DOF: Feasible manifolds. i.i.d.-Biased-Sampler (left) and RRT-Filter (right).

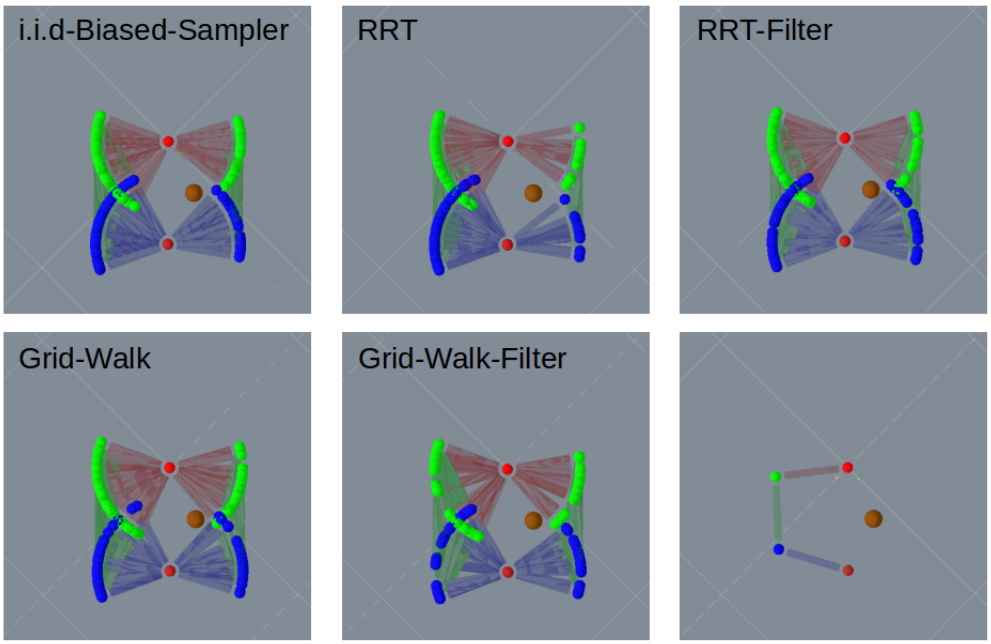

Figure 6.11: 3-DOF: 100 subsampled feasible configurations.

6.4.2 7-DOF

The results from the 7-DOF experiments also confirm that the distance from the seed to the manifolds plays a significant role in an algorithm's computational efficiency, making i.i.d.-Biased-Sampler inefficient in terms of AIPS compared to RRT and Grid-Walk.
For all other metrics, i.i.d.-Biased-Sampler ranked the best, which seems surprising since we have seen that both RRT and Grid-Walk performed better in the sphere experiments. From the visual exploration of the sphere experiments, it was seen that the stepsize α of RRT and window width w_{GW} of Grid-Walk were appropriately chosen such that the algorithms managed to travel over most of the manifold for the given amount of samples. This kind of analysis is not possible in $\mathbb{R}^7$, and we can only guess that the parameters were not chosen very well in this scenario.

Above mentioned parameter-related challenges brings forth a major advantage of i.i.d.-Biased-Sampler and disadvantage of RRT and Grid-Walk. When using i.i.d.-Biased-Sampler no parameters have to be chosen, making the algorithm much easier to use and robust, which is favorable. On the other hand, RRT or Grid-Walk may outperform i.i.d.-Biased-Sampler if parameters are chosen wisely. It is necessary, though, to choose the parameters right. A proper choice of the parameters for Grid-Walk and RRT is not straightforward, especially since one usually does not have any prior knowledge of where the feasible set is located, how many disconnected components it consists, and what shape they have.

Eventually the different realized poses shall be discussed (figure 6.12). It again has to be kept in mind that only a subset of S is shown in the images. Nonetheless, just from this visual inspection, it can be seen that all algorithms have sampled a variety of poses and captured a very similar spectrum of poses. However, a distinction to notice is that RRT and RRT-Filter seem to have discovered some more poses that place the end-effector between the obstacle and the goal. An explanation for this phenomenon may be that these poses are a disconnected manifold or part of it that is hard to reach because they lie in a corner or near the bounds of the configuration space, shielded by other manifolds or just very small. i.i.d.-Biased-Sampler has problems sampling these regions because samples are projected to larger, more central located manifolds. Instead, RRT could walk over the manifold to reach this for i.i.d.-Biased-Sampler hard-to-reach part of the manifold. We nonetheless do not see this reflected in the estimated coverage. This is possibly due to RRT not sampling manifold parts that i.i.d.-Biased-Sampler has sampled. The performance increase in coverage due to the poses that place the end-effector between obstacle and goal does not compensate for other regions of the manifold that RRT has not covered well.

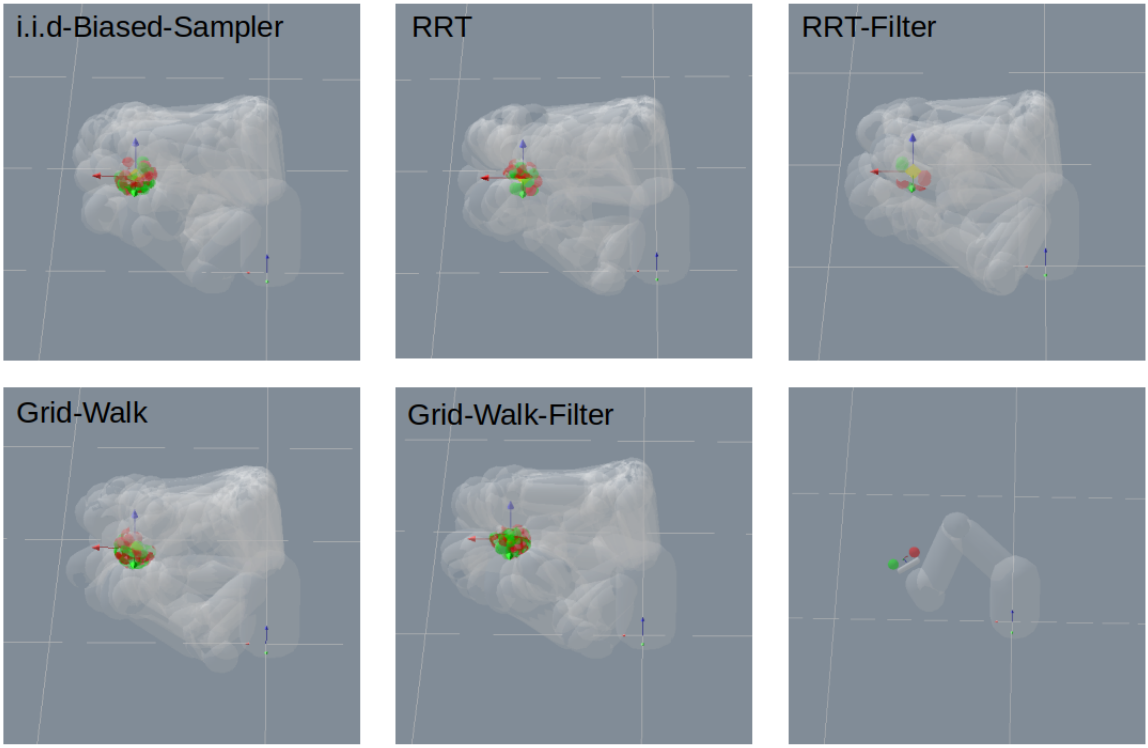

Figure 6.12: 7-DOF: 100 subsampled feasible configurations.

7 Conclusion

At the end of this document, the most important findings will be summarized in order to provide an overview. In addition, the procedure will be critically examined and an outlook will be given.

This thesis has studied various sampling algorithms that solve the problem of generating a set of samples $S \subset \mathcal{X}$ where $\mathcal{X}$ is the feasible set, a manifold that is defined by equality and inequality constraints. This manifold is embedded in the configuration space K and possibly consisting of many disconnected components of arbitrary shape. The algorithms studied are i.i.d-Biased-Sampler and the Markov-Chain-Samplers RRT(-on-Tangent) and Grid-Walk(-on-Tangent). All of these make use of biased optimization to project points from the configuration space K onto the manifold $\mathcal{X}$. A framework of metrics has been established to measure the uniformity (entropy H and variance Var of a set of KDE estimates), coverage (ADTS) of S and computational efficiency of the algorithm (AIPS). A variety of scenarios was then developed to study the algorithms behaviours both in an intuitive manner in a lower dimensional space as well as in higher dimensional robotic environments to reflect real world robotic problems and evaluate their performance in terms of above mentioned metrics.

7.1 Major Findings

A first observation is that it matters if a manifold is closed or with a boundary. For closed manifolds, it was shown that Grid-Walk approaches a stationary uniform distribution and is therefore a suitable and computationally efficient algorithm in such settings.

For manifolds with boundaries, it was then observed that the boundaries are spaces of special interest and that due to the projection behavior samples accumulate on the boundary which results in a non-uniform distribution on the boundary as well as its neighborhood. Theoretically simply rejecting the samples on the boundary would result in a more uniform distribution, but it was also found that in practice this comes at the cost of less coverage and that the accumulation of samples on the bounds is not noticeable when the amount of samples is small.

We furthermore discovered that for i.i.d.-Biased-Sampler coverage and uniformity is better if the feasible manifold lies in the center of the configuration space compared to when it lies in a corner. It therefore matters where in the configuration space the manifold is located. Since biased optimization can be perceived as a projection of sample x onto its nearest point $y \in \mathcal{X}$ a situation can occur in which parts of $\mathcal{X}$ that are shielded by other feasible manifolds or are not as exposed to a large proportion of the configuration space and may be sampled much less using i.i.d.-Biased-Sampler. Along this was also found that RRT and Grid-Walk outperform i.i.d.-Biased-Sampler in terms of computational efficiency because the points to

project onto the manifold are on average much closer to the manifold.

Furthermore, composite samplers were presented which combine a global sampling stage in which i.i.d.-Biased-Sampler is used to generate seeds/starting points for the local sampling stage where RRT or Grid-Walk are applied. The composite-sampler solves the problem that multiple disconnected manifolds are not covered well when only using a single instance/chain of RRT or Grid-Walk. When using a single instance of RRT or Grid-Walk it is not guaranteed that the Markov-Chain jumps between the manifolds. It was nevertheless found that uniformity suffers when after a global sampling stage the amount of Markov-Chains on a manifold differs between the disconnected manifolds. This is reinforced when the manifolds are of different sizes, such that small manifolds are sampled more densely. To solve the former issue a filter operation that removed seeds from spaces of high seed density given a distance threshold showed promising results when this threshold was chosen properly. The algorithms were further extended by introducing the Bandit-Sampler that approximates the individual manifold component sizes a chain is sampling and uses this to construct a probability distribution to sample the chain that will be evolved in the next iteration. This probability distribution corresponds to the approximated manifolds sizes and resembles that a chain sampling a large manifold is on average sampled more often. A major assumption taken was that the disconnected manifolds are each sampled by a single or the same amount of chains. It nonetheless turned out that this is in practice not the case and the bandit-sampler has to be further adjusted to cater for such situations.

When transferring the algorithms to robotics examples, we confirmed that RRT and Grid-Walk are computationally more efficient compared to i.i.d.-Biased-Sampler. Opposing the good results obtained previously using RRT and Grid-Walk it turned out that especially in high dimension and with no prior knowledge of the feasible set setting the parameters such as steps size α in RRT or neighborhood width w_{GW} in Grid-Walk becomes a challenging problem itself. This lead to the conclusion that i.i.d.-Biased-Sampler has the major advantage of being an easy to apply and robust approach to constrained sampling because no parameters have to be chosen.

7.2 Outlook

This thesis studied the sampling distributions over constraint manifolds generated using i.i.d-Biased-Sampler, RRT and Grid-Walk. The focus was put on a general understanding of the algorithms that can act as a reference for researchers to further design and study algorithms for the sampling from constraint manifolds. The examples and experiments studied in this thesis were therefore rather general except for some experiments from the robotic domain. A logical next step would thus be to further study and develop the algorithms and sampling distributions for other domain-specific tasks such as legged locomotion. There is also still a lot of theoretical foundation to be laid, the unit-line experiment could be modified for higher and/or maybe even arbitrary dimensions. One could then examine what the stationary distribution is on an edge or corner.

Some extensions that have been developed like the composite-sampler, the filter operation, and the bandit-sampler tackle important challenges faced within constrained sampling such

as many disconnected and differently sized manifolds. These extensions nonetheless showed limitations in their practical application. The bandit sampler suffers from multiple markov-chains sampling the same disconnected manifold resulting in high sampling density on this manifold. Designing a clustering algorithm to identify these chains that sample the same manifold and adjust their sampling frequency could solve this issue. A drawback of RRT-on-Tangent is that the RRT is locally bounded which can lead to less coverage if the feasible set exceeds the bounds. A solution to this could be to integrate AtlasRRT as a local sampler in the composite-sampler.

One of the major challenges in constrained sampling specifically for robotic tasks in high dimensional configuration spaces is that one does not have any prior knowledge about the shape of the feasible set. Not knowing the shape of the feasible set makes it significantly more difficult to choose parameters like RRT's stepsize α or Grid-Walks neighborhood width w properly. It would therefore be of great value to derive heuristics or other procedures by which these parameters could be chosen. This could be integrated into a sampler as a preceding step or dynamically adjusted during the sampling. Next to the further development of the algorithms, it would also be of benefit to gain more insights into the actual topology of the feasible sets in higher dimensions which could be done using persistent homology to identify qualitative features.

Annex

Below is a summary of the parameters used for the sphere and robotics experiments in section 4.3.1. The sphere scenarios Center Connected (CC), Off-Center-Connected (OCC) and Off-Center-Disconnected (OCDC).

Algorithm	**Scenario**	**n_runs**	**n_global**	**n_local**	**d_thresh**	**w_rrt**	**alpha**	**w_gw**
RRT-Filter	3-DOF	1	100	50	0.5	1.2	0.01	
GW-Filter	3-DOF	1	100	50	0.5			0.1
RRT	3-DOF	1	50	100		1	0.01	
GW	3-DOF	1	50	100	0.5			0.1
i.i.d.-Biased-Sampler	3-DOF	1	5,000					
RRT-Filter	7-DOF	1	1,000	100	0.5	1.2	0.01	
GW-Filter	7-DOF	1	1,000	100	0.5			0.05
RRT	7-DOF	1	1,000	100		1.2	0.01	
GW	7-DOF	1	1,000	100				0.05
i.i.d.-Biased-Sampler	7-DOF	1	100,000					
RRT-Projection	CC	100	100	50		0.5	0.01	
Gw-Projection	CC	100	1	5,000				0.5
i.i.d.-Biased-Sampler	CC	100	5,000					
Gw-Projection	OCC	100	100	50				0.5
Gw-Projection-filter	OCC	100	100	50	0.4			0.5
RRT-Projection	OCC	100	100	50		0.5	0.01	
RRT-Projection-Filter	OCC	100	100	50	0.4	0.5	0.01	
i.i.d.-Biased-Sampler	OCC	100	5,000					
Gw-Rejection-Filter	OCDC	100	100	50	0.4			0.5
Gw-Projection-filter	OCDC	100	100	50	0.4			0.5
RRT-Rejection-Filter	OCDC	100	100	50	0.4	0.5	0.01	
RRT-Projection-Filter	OCDC	100	100	50	0.4	0.5	0.01	
i.i.d.-Biased-Sampler	OCDC	100	5,000					
RRT-Rejection-Filter-Bandit	OCDC	100	100	50	0.4	0.5	0.01	
Gw-Rejection-Filter-Bandit	OCDC	100	100	50	0.4			0.5

Bibliography

[AdFDJ03] Andrieu, Christophe, Nando de Freitas, Arnaud Doucet Michael I. Jordan: *An Introduction to MCMC for Machine Learning.* Machine Learning, 50:5–43, 2003.

[And03] Anderson, T.W.: *An Introduction to Multivariate Statistical Analysis.* Wiley Series in Probability and Statistics. Wiley, 2003.

[BDGM97] Beirlant, Jan, E. Dudewicz, L. Gyor E.C. Meulen: *Nonparametric Entropy Estimation: An Overview.* International Journal of Mathematical and Statistical Sciences, 6, 01 1997.

[Bis06] Bishop, Christopher M.: *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[BM58] Box, George Edward Pelham Mervin Edgar Muller: *A note on the generation of random normal deviates.* The Annals of Mathematical Statistics, 29:610–611, 1958.

[BM20] Breiding, Paul Orlando Marigliano: *Random points on an algebraic manifold*, 2020.

[BSK11] Berenson, Dmitry, Siddhartha Srinivasa James Kuffner: *Task Space Regions: A framework for pose-constrained manipulation planning.* The International Journal of Robotics Research, 30(12):1435–1460, 2011.

[BSU12] Brubaker, Marcus, Mathieu Salzmann Raquel Urtasun: *A Family of MCMC Methods on Implicitly Defined Manifolds.* Lawrence, Neil D. Mark Girolami (): *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, 22 *Proceedings of Machine Learning Research*, 161–172, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.

[CGT91] Conn, Andrew, Nicholas Gould Philippe Toint: *A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds.* SIAM Journal on Numerical Analysis, 28, 04 1991.

[CT06] Cover, Thomas M. Joy A. Thomas: *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, July 2006.

[Des04] Deserno, Markus: *How to generate equidistributed points on the surface of a sphere*, 2004.

[Dev86] Devroye, Luc: *Non-Uniform Random Variate Generation(originally published with.* Springer-Verlag, 1986.

[DH07] Dasgupta, Sanjoy Daniel Hsu: *On-Line Estimation with the Multivariate Gaussian Distribution.* 278–292, 06 2007.

[Eis20] Eisermann, Michael: *Lecture notes in Grundlagen der Topologie*, October 2020.

[GRS95] Gilks, W.R., S. Richardson D. Spiegelhalter: *Markov Chain Monte Carlo in Practice.* Chapman & Hall/CRC Interdisciplinary Statistics. Taylor & Francis, 1995.

[Han09] Hansen, Bruce E.: *Lecture notes on Nonparametrics*, Spring 2009.

[HFC+02] Heckert, N., James Filliben, C Croarkin, B Hembree, William Guthrie, P Tobias J Prinz: *Handbook 151: NIST/SEMATECH e-Handbook of Statistical Methods*, 2002-11-01 00:11:00 2002.

[Joh] Johnson, Steven G.: *The NLopt nonlinear-optimization package.*

[JP11] Jaillet, Léonard Josep M. Porta: *Path Planning with Loop Closure Constraints Using an Atlas-Based RRT. ISRR*, 2011.

[Kap01] Kaplansky, I.: *Set Theory and Metric Spaces.* AMS Chelsea Publishing Series. AMS Chelsea Publishing, 2001.

[KMK18] Kingston, Zachary, Mark Moll Lydia E. Kavraki: *Sampling-Based Methods for Motion Planning with Constraints.* Annual Review of Control, Robotics, and Autonomous Systems, 1(1):159–185, 2018.

[LaV] LaValle, Steve: *About RRTs.*

[Lav98] Lavalle, Steven M.: *Rapidly-Exploring Random Trees: A New Tool for Path Planning.* , 1998.

[Lav06] Lavalle, Steven M.: *Planning Algorithms.* Cambridge University Press, 2006.

[LR78] Lazo, A.V. P. Rathie: *On the entropy of continuous probability distributions (Corresp.).* IEEE Transactions on Information Theory, 24(1):120–122, 1978.

[Moo65] Moore, Gordon E.: *Cramming more components onto integrated circuits.* Electronics, 38(8), April 1965.

[OCRR14] O'Brien, Travis, William Collins, Sara Rauscher Todd Ringler: *Reducing the computational cost of the ECF using a nuFFT: A fast and objective probability density estimation method.* Computational Statistics Data Analysis, 79, 11 2014.

[OHHDT21] Ortiz-Haro, Joaquim, Jung-Su Ha, Danny Driess Marc Toussaint: *Structured deep generative models for sampling on constraint manifolds in sequential manipulation. 5th Annual Conference on Robot Learning*, 2021.

[Rhe96] Rheinboldt, W.C.: *MANPAK: A set of algorithms for computations on implicitly defined manifolds.* Computers Mathematics with Applications, 32(12):15–28, 1996.

[Ros97] Ross, Sheldon M.: *Introduction to Probability Models.* Academic Press, San Diego, CA, USA, Sixth , 1997.

[Tou] Toussaint, Marc: *RAI Robotic AI.*

[Tou14] Toussaint, Marc: *A Novel Augmented Lagrangian Approach for Inequalities and Convergent Any-Time Non-Central Updates.* arXiv: Optimization and Control, 2014.

[Tou21] Toussaint, Marc: *Lecture notes on Optimization*, Winter 2020/21.

[Vem05] Vempala, Santosh: *Geometric random walks: a survey.* Combinatorial and Computational Geometry, 573–612, 2005.

[Vry] Vryniotis, Vasilis: *Tuning the learning rate in Gradient Descent.* `http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/`. Accessed: 2021-02-18.

[Wei] Weisstein, Eric: *Cellular Automaton.*