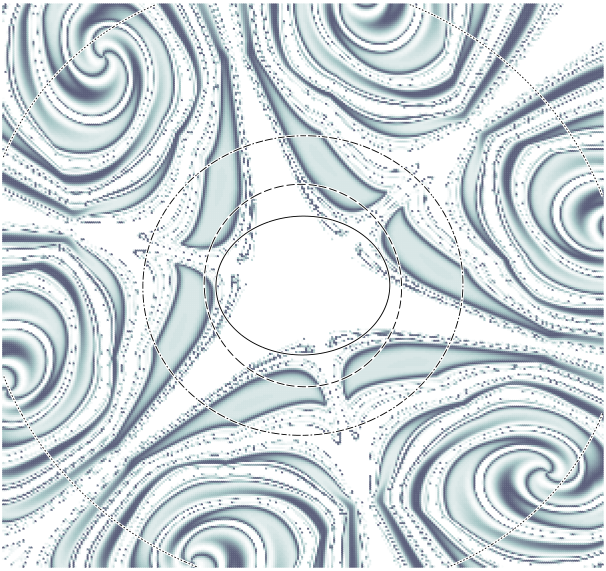
AN EFFICIENT RANDOMIZED APPROXIMATION ALGORITHM FOR VOLUME ESTIMATION AND DESIGN CENTERING



JOSEFINE ASMUS

2017

AN EFFICIENT RANDOMIZED APPROXIMATION ALGORITHM FOR VOLUME ESTIMATION AND DESIGN CENTERING

A dissertation submitted to

TECHNISCHE UNIVERSITÄT DRESDEN
FAKULTÄT INFORMATIK

for the degree of

Doctor rerum naturalium
(Dr. rer. nat.)

presented by

JOSEFINE ASMUS

Dipl. Biomath.
born on September 19th, 1986
in Rostock, Germany

Dresden
2017

Day of submission:
February 17th 2017

Day of defense:
April 28th 2017

Examination committee:
Prof. Dr. Raimund Dachselt (chair of the committee)
Technische Universität Dresden, Dresden, Germany
Prof. Dr. Ivo F. Sbalzarini (reviewer)
Technische Universität Dresden, Dresden, Germany
Prof. Dr. Dr. Fabian J. Theis (reviewer)
Technische Universität München, München, Germany
Dr. Dmitrij Schlesinger
Technische Universität Dresden, Dresden, Germany
Prof. Dr. Jeronimo Castrillon
Technische Universität Dresden, Dresden, Germany
Dr. Christian L. Müller
Flatiron Institute, Simons Foundation, New York City, NY, USA

Abstract

The design of systems or models that work robustly under uncertainty and environmental fluctuations is a key challenge in both engineering and science. This is formalized in the design centering problem, defined as finding a design that fulfills given specifications and has a high probability of still doing so if the system parameters or the specifications randomly fluctuate. Design centering is often accompanied by the problem of quantifying the robustness of a system. Here we present a novel adaptive statistical method to simultaneously address both problems. Our method, L_p-Adaptation, is inspired by how robustness evolves in biological systems and by randomized schemes for convex volume computation. It is able to address both problems in the general, non-convex case and at low computational cost. In this thesis, we describe the concepts of the algorithm and detail its steps. We then test it on known benchmarks, and demonstrate its real-world applicability in electronic and biological systems. In all cases, the present method outperforms the previous state of the art. This enables re-formulating optimization problems in engineering and biology as design centering problems, taking global system robustness into account.

ZUSAMMENFASSUNG

Die Konstruktion von Systemen oder Modellen, welche unter Unsicherheit und Umweltschwankungen robust arbeiten, ist eine zentrale Herausforderung sowohl im Ingenieurwesen als auch in den Naturwissenschaften. Dies ist im Design-Zentrierungsproblem formalisiert als das Finden eines Designs, welches vorgegebene Spezifikationen erfüllt und dies mit einer hohen Wahrscheinlichkeit auch noch tut, wenn die Systemparameter oder die Spezifikationen zufällig schwanken. Das Finden des Zentrums wird oft durch das Problem der Quantifizierung der Robustheit eines Systems begleitet. Hier stellen wir eine neue adaptive statistische Methode vor, um beide Probleme gleichzeitig zu lösen. Unsere Methode, L_p-Adaptation, ist durch Robustheit in biologischen Systemen und durch randomisierte Lösungen für konvexe Volumenberechnung inspiriert. L_p-Adaptation ist in der Lage, beide Probleme im allgemeinen, nicht-konvexen Fall und bei niedrigen Rechenkosten zu lösen. In dieser Arbeit beschreiben wir die Konzepte des Algorithmus und seine einzelnen Schritte. Wir testen ihn dann anhand bekannter Vergleichsfälle und zeigen seine Anwendbarkeit in elektronischen und biologischen Systemen. In allen Fällen übertrifft das vorliegende Verfahren den bisherigen Stand der Technik. Dies ermöglicht die Umformulierung von Optimierungsproblemen im Ingenieurwesen und in der Biologie als Design-Zentrierungsprobleme unter Berücksichtigung der globalen Robustheit des Systems.

Für meine Eltern

ACKNOWLEDGEMENTS

First of all I want to thank my supervisor Ivo Sbalzarini for letting me become a MOSAICAN and for his support throughout the last four years. Ivo, thank you for being so motivating and positive, and thank you for giving me lots of freedom but also for guiding me.

I want to thank Christian Müller for introducing me to the topics of design centering and volume approximation, for uncountable Skype-calls, for his infectious excitement and in general for being a true inspiration. Furthermore I want to thank Fabian Theis for kindly agreeing on reviewing my thesis.

It was a great pleasure of working with Lucas Wetzel and Daniel Platz and I am excited to find out where our project will lead us.

I want to thank all former and current members of the MOSAIC group I had the pleasure to interact with. Thank you for lively discussions, inspirations, tasty dinners, lots of fun and hard work. Special thanks go to Yaser, Bevan, and Alex, without whom this thesis would not have been the same.

I am grateful to Ivo, Christian and the Center for Advancing Electronics Dresden (cfaed) for making it possible that I could visit Christian in NYC at the Courant Institute at NYU in 2013 and at the Simons Foundation in 2015. These visits were an awesome boost for my science and on top of that I had the chance to explore this amazing city. These visits would not have been possible without the help of my parents, Klaus and the Thiemer family, thank you all!

I want to thank my family for their unconditional love and steady support. I could not have done this without you! Above all, I want to thank Emilia, for making my days brighter and keeping me sane.

CONTENTS

ABSTRACT i

ZUSAMMENFASSUNG iii

ACKNOWLEDGEMENTS v

TABLE OF CONTENTS I

NOMENCLATURE V

1 INTRODUCTION 1

1.1 MOTIVATION 1

1.2 PRIOR WORK 2

1.2.1 Previous approaches to design centering 3
1.2.2 Previous approaches to volume estimation 4

1.3 PROBLEM STATEMENT 5

1.4 CONTRIBUTION 6

2 PRELIMINARIES 9

2.1 INTRODUCTION TO MARKOV CHAIN MONTE CARLO (MCMC) 9

2.1.1 Metropolis-Hastings algorithm 12
2.1.2 Adaptive Markov-Chain Monte Carlo 14
2.1.3 Advantages and disadvantages of adaptive MCMC . 17

2.2 BAYESIAN INFERENCE 18

2.2.1 Approximate Bayesian Computation (ABC) 19
2.2.2 Using ABC for model selection 22

2.3 GAUSSIAN ADAPTATION 24

2.4 COVARIANCE MATRIX ADAPTATION EVOLUTION STRATEGY 26

2.4.1 Adaptation of the mean 27
2.4.2 Adaptation of the covariance matrix 27
2.4.3 Step size control 30

2.5 CONVEX VOLUME ESTIMATION 31

2.5.1 Rounding 31
2.5.2 Multiphase MCMC (subdivision and sampling) . . . 31

3 L_p-ADAPTATION, AN EFFICIENT ALGORITHM UNITING APPROXIMATE DESIGN CENTERING AND VOLUME ESTIMATION 35

3.1 CONCEPTS OF L_p-ADAPTATION 38

3.1.1 Maximizing robustness 38
3.1.2 Exploiting the hitting probability 40

3.2 STEPS OF L_p-ADAPTATION 41
3.2.1 Initialization 43
3.2.2 Sampling 45
3.2.3 Evaluation 45
3.2.4 Adaptation of the proposal 47
3.2.5 Adapting the hitting probability 52
3.3 AVERAGING OF COVARIANCE MATRICES 59
3.3.1 Example 60
3.4 BEHAVIOR OF L_p-BALLS IN DIFFERENT DIMENSIONS 63
3.5 EXTRACTING INFORMATION FROM GENERATED SAMPLES AND FROM THE EVOLVING PROPOSAL DISTRIBUTION 70
3.5.1 Correlations and sensitivity of parameters 70
3.5.2 Shape of the feasible region 71
3.6 CONCLUSION 76
4 VALIDATION 79
4.1 ILLUSTRATION IN TWO DIMENSIONS 79
4.2 SYNTHETIC L_p-BALLS 84
4.3 CONCLUSION 94
5 APPLICATION 95
5.1 DESIGNING A SWITCHED CAPACITOR FILTER 95
5.2 QUANTIFYING BASIN STABILITY OF SYNCHRONIZED STATES IN MUTUALLY DELAY-COUPLED OSCILLATORS 100

5.2.1 The model 101
5.2.2 The Kuramoto order parameter 101
5.2.3 Space transformation 103
5.2.4 Membership oracle 105
5.2.5 Results 106

5.3 COMPARING ROBUSTNESS OF TWO BACTERIAL TWO COMPONENT SYSTEMS 110

5.3.1 Ordinary differential equations models 110
5.3.2 Membership oracles 113
5.3.3 Results 117

5.4 CONCLUSION 125

6 CONCLUSIONS AND FUTURE WORK 127

APPENDICES 131

A MATLAB SOURCE CODE 133

B ADDITIONAL FIGURES FOR SECTION 4.2 137

C ADDITIONAL FIGURES FOR SECTION 5.3 145

BIBLIOGRAPHY 153

CURRICULUM VITAE 163

PUBLICATIONS 164

DECLARATION OF AUTHORSHIP 165

NOMENCLATURE

Here, we list used acronyms and symbols. Vectors and matrices are printed in bold face. Other variables may appear.

Acronyms

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
5D	five-dimensional
10D	ten-dimensional
20D	20-dimensional
50D	50-dimensional
90D	90-dimensional
ABC-SMC	sequential Monte Carlo ABC algorithm
ABC	Approximate Bayesian Computation
AFOSM	Advanced First-Order Second Moment
AM	adaptive Metropolis
AP	Adaptive Proposal
CADE	Constraint Adaptation by Differential Evolution
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CMA	Covariance Matrix Adaptation

DPLL	digital phase-locked loop
ES	Evolution Strategy
GaA	Gaussian Adaptation
HK	histidine kinase
i.i.d.	independent and identically distributed
LF	loop filter
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MH	Metropolis-Hastings
PCM	pulse code modulation
PD	phase detector
PLL	phase-locked loop
RC	Resistor Capacitor
RR	response regulator
SCPCM	switched capacitor pulse code modulation
SC	Switched Capacitor
TCS	two-component system
VCO	voltage controlled oscillator

Greek Characters

α	acceptance criterion
α_c	parameter
α_μ	parameter
β	learning rate
δ	relative change
ϵ	tolerance, error threshold
η	standard normal distributed vector
Γ	Gamma function
λ	population size (candidate solutions), Eigenvalues
μ	accepted candidate solutions, number of feasible points
ω	frequency
ψ	average phase

ρ	correlation coefficient
σ	standard deviation
σ^2	variance
$\sigma^{(g)}$	step size at generation g, overall scale of the sampling distribution
τ	time delay
θ	parameter vector, phase of an oscillator
$\dot{\theta}$	instantaneous frequency
θ'	point in transformed phase space (rotated, reduced, translated)
$\theta^\star$	candidate parameter vector, point in rotated phase space

Latin Characters

A	feasible region, convex body
B	ball in a rounding algorithm, Beta function
$\mathbf{B}$	matrix
b	ratio of two volume approximations
$\mathbf{b}$	vector of binary variables
$\mathbb{C}$	coupling matrix
$\mathbf{C}$	covariance matrix, transformation matrix
c_1	weight
c_m	weight factor for adaptation of mean
c_c	backward time horizon of the evolution path $\mathbf{p}_c$
c_μ	learning rate
cond	condition number
Cov	covariance
c_σ	backward time horizon of the evolution path $\mathbf{p}_\sigma$
c_T	threshold
d	distance function
det	determinant
d_F	Foerstner distance
d_r	relative difference

d_σ	damping parameter
$\mathbb{E}$	expectation
e	Euler-Mascheroni constant
F	intrinsic frequency
f	probability distribution, function, signal frequency
f_a	sampling frequency of a filter
F_c	filter cutoff frequency
f_c	contraction factor
f_e	expansion factor
g	generation, iteration
$\tilde{G}$	generalized Gamma distribution
H	transfer function
$\mathbf{I}_n$	n-dimensional identity matrix
K	rounded body, coupling strength
$L_p^n(r)$	L_p-ball in n dimensions with radius r
$L_p^n(r, \mathbf{C})$	L_p-ball in n dimensions with radius r and transformation matrix $\mathbf{C}$
$\mathcal{L}_p^n$	set of all L_p-balls in n dimensions where p is the type of ball
$\mathbf{m}$	design center, mean, center of the L_p-proposal
$\mathcal{M}$	set of candidate models
$\mathbb{N}$	set of natural numbers
$\mathcal{N}$	Normal (Gaussian) distribution
n	dimension of the space
N	number of oscillators, number of elements
N_C	weighting factor
N_m	weighting factor
N_T	weighting factor
P	target hitting probability
$\mathbf{p}_c$	evolution path
P_{emp}	empirical hitting probability
p_f	p-norm of feasible L_p-ball
p_p	p-norm of L_p-proposal

Pr	probability
$\mathbf{p}_\sigma$	evolution path
$\mathbf{P}$	transition probability matrix, vector of target hitting probabilities
$p(x)$	target distribution
$\mathbf{Q}$	matrix
$\mathbf{q}$	vector
$q(x)$	probability, proposal distribution
$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of positive real numbers
r	radius, step size
$\bar{r}$	averaged r
$\mathcal{S}_+^{n\times n}$	set of symmetric positive definite matrices
S	state space, input signal
s	size of interval
T	transformation
t	time
V_{BB}	volume of the axes-aligned bounding box
V_L	volume of the Loewner ellipsoid
vol	volume
$\widetilde{\mathrm{vol}}$	approximated volume
w	weight
$\mathbf{X}$	vector of sampled points
$\mathbf{x}$	parameter vector
y	observed data
$\mathbf{y}$	real random vector
$y^\star$	dataset
$\tilde{y}$	unobserved data
z	generalized order parameter
z_K	Kuramoto order parameter

Special symbols

$[\cdot]$	concentration
$\in$	set membership
∞	infinity
$\cap$	intersection
$\mathcal{O}$	complexity function, “big-O notation”
$\mathcal{O}^\star$	complexity function, “big-O notation”, logarithmic factors are omitted
$\prod$	product notation
$\subseteq$	sub-set
$\sum$	summation notation

CHAPTER

ONE

INTRODUCTION

1.1 MOTIVATION

Design centering is a long-standing and central problem in systems engineering and model inference. It is concerned with determining design parameters of a system or model that guarantee operation within given specifications and are robust against random variations. While *design optimization* aims to determine the design that *best* fulfills (one aspect of) the specifications, *design centering* wants to find the design that meets the specifications most *robustly*. Traditionally, this problem has been considered in electronic circuit engineering (Graeb, 2007), where a typical task is to determine the nominal values of electronic components (e.g., resistances, capacitances, etc.) such that the circuit fulfills some specifications and is robust against manufacturing tolerances in the components. Examples of specifications in electronic circuits are frequency response, harmonic distortion, energy consumption, and manufacturing cost. Recently, related ideas have also entered the field of synthetic biology with the aim of robustly designing novel synthetic biological circuits (Barnes et al., 2011; Woods et al., 2016). Any criterion that can be verified for a given design can be used as a specification.

In order to be robust against perturbations, the specifications cannot be defined too narrowly. This implies that there are usually many designs that fulfill the specifications. The size or volume of the set of all these *feasible* designs is an intuitive measure for the robustness with which the specifications can be fulfilled. Robustness is therefore related to the probability that the design still fulfills the same specifications when the design parameters randomly vary, or the specifications fluctuate. Quantifying this robustness requires estimating the size or volume of the set of all feasible designs.

Volume estimation and design centering in the most general form only assume that a given design can be evaluated through a procedure (referred to as "oracle" (Grötschel et al., 1988)) that checks whether the design fulfills the specifications, or not. In this general setting, design centering and volume estimation are hard problems. Exhaustively determining the set of all feasible designs requires exponentially many design trials in the number of design parameters. Since typical systems or circuits have tens to hundreds of design parameters, testing all possible combinations is prohibitive. It is hence intuitive that exact solutions to design centering are NP-hard (Puchalski et al., 2006), i.e., they cannot be efficiently determined on a deterministic computer. Less intuitively, it is also NP-hard to determine the exact volume of a high-dimensional set using a deterministic algorithm (Bárány and Füredi, 1987; Khachiyan, 1989), even if the set is convex. Efficient approaches to design centering and volume estimation are hence always approximate. However, even though volume estimation is closely related to design centering, previous approximate approaches have considered them separately.

1.2 PRIOR WORK

We therefore separately review previous approaches to design centering and volume approximation.

1.2.1 Previous approaches to design centering

Previous approaches to design centering can be classified into geometrical and statistical approaches (Sapatnekar et al., 1994). Geometrical approaches use simple geometric bodies to approximate the feasible region, which is usually assumed to be convex (Schwencker et al., 2000). Examples of geometrical approaches include Simplical Approximation (Director and Hachtel, 1977; Vaidya, 1989), which approximates the boundary of the feasible region by adaptation of a convex polytope. Due to the curse of dimensionality, however, Simplical Approximation becomes unpractical in dimensions $n > 8$ (Soin and Spence, 1980; Harnisch et al., 1997). Suggested improvements to relax the convexity requirement instead assume differentiability of the specifications (Vidigal and Director, 1982), which cannot be guaranteed in black-box problems. Another example of a geometrical approach is Ellipsoidal Approximation (Abdel-Malek and Hassan, 1991), which finds the ellipsoid of largest volume that still completely fits into the feasible region. All endpoints of the ellipsoidal axes and the center of the ellipsoid need to be feasible. While Ellipsoidal Approximation does not strictly require convexity of the feasible region, its approximation properties strongly depend on it. A third example of a geometrical approach is the polytope method (Sapatnekar et al., 1994), which also uses a convex polytope to approximate the feasible region, but then finds the design center by either inscribing the largest Hessian ellipsoid or by using a convex programming approach. The latter approach, however, requires an explicit probabilistic model of the variations in the design parameters, which is usually not available in practice.

Statistical approaches approximate the feasible region by Monte Carlo sampling. Since exhaustive sampling is not feasible in high dimensions, the key ingredient of statistical methods is to find a smart sampling proposal, and concentrate on informative regions. The methods then sample points from this proposal and evaluate the specifications for these points to decide if they are feasible. The ratio of feasible to infeasible points sampled then provides information about the robustness of a design (Gu and Roychowdhury, 2010). Constraint adaptation by Differential Evolution (CADE) (Storn, 1999) is a classical statistical design centering method based on Differential Evolution (Storn and Price, 1997). It assumes the feasible region to be convex and starts from a population of initial points. To find those points, the specifications (constraints) are first relaxed and then tightened successively back to the original ones. After the original specifications are met, the mean of all points (which have to be feasible) is used as an approximation of the design center. Another representative statistical approach is the Advanced First-Order Second Moment (AFOSM) method (Seifi et al., 1999). It samples candidate points from L_p-balls in order to estimate the yield (i.e., the ratio of feasible to infeasible points) and approximate the feasible region. Which L_p-norm to use is directly related to the assumed statistical distribution of the random perturbations. The proposal L_p-balls are adapted to maximize their volume while still being completely contained within the feasible region. This therefore does not allow estimating the total volume of the feasible region. A third example of a statistical method is the Center of Gravity Method (Soin and Spence, 1980). In each iteration, it computes the center of gravity of the feasible samples and of the infeasible samples. The design center is then moved toward the center of the feasible points and away from the center of the infeasible ones. The Momentum-Based Center of Gravity Method (Tan and Ibrahim, 1999) extends this idea to include information from the past *two* iterations.

1.2.2 Previous approaches to volume estimation

Volume computation is an important problem in many areas, e.g. software engineering, computer graphics, economics, and statistics (Liu et al., 2007). Deterministic methods for volume computation of convex polytopes use for example triangular methods or signed decomposition methods (Büeler et al., 2000). The former decompose the polytope into simplices whose volumes are easily computed and summed (Büeler et al., 2000). The latter decompose the polytope into signed simplices such that the signed sum of their volumes is the volume of the polytope (Büeler et al., 2000). However, it has been shown that deterministically computing the volume is NP-hard (Dyer and Frieze, 1988; Khachiyan, 1988, 1989), even for convex bodies.

Using a randomized algorithm, the volume of a convex body can be approximated to arbitrary precision in polynomial time (Dyer et al., 1991). Over the years, Dyer, Frieze, and Kannan's breakthrough-algorithm (with a theoretical complexity of $\mathcal{O}^\star(n^{23})$ oracle calls) has been improved in a sequence of papers until Lovasz and Vempala's $\mathcal{O}^\star(n^4)$-algorithm[1]. The main concepts of randomized volume approximation algorithms, Rounding and Multiphase Markov chain Monte Carlo, are further explained in Section 2.5.

1.3 PROBLEM STATEMENT

We consider the design (or parameter) space to be $\mathbb{R}^n$, i.e., the n-dimensional vector space of real numbers. The region (subspace) of the parameter space that contains all parameter vectors for which the system meets or exceeds the specifications is called the *feasible region* $A \subset \mathbb{R}^n$. We denote the total volume of the feasible region by $\mathrm{vol}(A)$, defined as the integral of the uniform density over A. This volume is a natural measure for the total amount of feasible designs available and can be used to compare and choose between different designs or competing models (Hafner et al., 2009). Moreover, the overall shape and orientation of the feasible region contains information about correlations between design parameters, which can be exploited for model reduction and to guide experimental verification of a design.

Depending on the available side-information about the design specifications, different operational definitions of the *design center* $\mathbf{m} \in A$ exist, including the *nominal design center*, the *worst-case design center*, and the *process design center* (Seifi et al., 1999). For instance, in the example of manufacturing an electronic circuit from components with known manufacturing tolerances, the design center maximizes the production yield. Here, we follow the general statistical definition of the design center (Kjellström and Taxen, 1981) and seek among all points (parameter vector) $\mathbf{x} \in A$ the design center $\mathbf{m} \in A$ that represents the mean of a probability distribution $q(\mathbf{x})$ of maximal volume covering the feasible region A with a given *target hitting probability* P. For convex feasible regions, using the uniform prob-

---

[1] the asterisk in the order notation indicates that logarithmic factors in n are omitted

ability distribution over A and $P = 1$, the design center coincides with the geometric center of the feasible region, which historically inspired the terminology.

When encountering a new problem one usually has no knowledge of the shape of its feasible region and wants to estimate different properties of it. If we know that the region is convex, good methods for both design centering and volume approximations are known. In real-world problems, however, we usually cannot guarantee convexity of the feasible region, but still want to get approximations for its design center or its volume. For this, a more general framework is needed that does not make any assumptions about the feasible region and can ideally be used for a broad range of applications.

1.4 CONTRIBUTION

Here, we jointly consider the problems of design centering and volume estimation in their most general form. We present an approximate statistical method that unites the two problems under the same framework. We also present an efficient computational algorithm, called L_p-Adaptation, for practical application of this new framework. Our contribution is hence twofold: a conceptual framework that unites design centering and robustness estimation, and a computationally efficient randomized approximation algorithm for it.

The proposed conceptual framework exposes several links and trade-offs between design centering and volume estimation. It is inspired by the observation that robust designs are a hallmark of biological systems, such as cell signaling networks, blood vasculature networks, and food chains (Kitano, 2004). Biological systems have to be robust against fluctuations, as otherwise they would likely not survive in a changing environment. It has been observed that the robustness of biological networks is related to the volume of the set of feasible parameters (Dayarian et al., 2009; von Dassow et al., 2000). This is the same definition of robustness we use for engineering systems. Nature has hence found a way of approximating both design centering and volume estimation through self-organization and natural selection. This succession of design alteration and design selection is akin to bio-inspired optimization algorithms, such as evolution strategies (Beyer and Schwefel, 2002) and genetic algorithms (Whitley, 1994), with the important difference that not optimization is the goal, but design centering and volume estimation. In our framework, design selection is hence done by checking whether the specifications are fulfilled. Feasible designs then undergo random alterations with the specific aim of exploring the space of all feasible designs as broadly and efficiently as possible.

Efficient and broad exploration of feasible designs is the core of the L_p-Adaptation algorithm. Following the biological inspiration, the algorithm is based on stochastic sampling of designs together with a consistent way of converting the explored samples to an estimate of the robustness and the design center. L_p-Adaptation is computationally efficient, reaching or outperforming the previous state of the art, as we show in this thesis. Most importantly, however, L_p-Adaptation is based on the joint consideration of the two problems and therefore relaxes the limiting assumptions previous approaches needed to make about either the convexity or smoothness of the set of feasible designs, or the correlations between parameters. L_p-Adaptation provides the first computationally efficient and versatile method for approximately solving general, oracle-based and non-convex design centering and volume estimation problems.

CHAPTER

TWO

PRELIMINARIES

In this chapter, we introduce important preliminaries for this thesis. First, we give a short introduction to Markov chain Monte Carlo methods and Bayesian inference, specifically to *Approximate Bayesian Computation.* Our method for design centering and volume approximation, L_p-Adaptation, which we will describe in Chapter 3, is a statistical method based on concepts of *Gaussian Adaptation* (GaA) (Kjellström and Taxen, 1981) and *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) (Hansen and Ostermeier, 1996). Therefore, we then review Kjellström's GaA and give an overview of CMA-ES. We will conclude with a summary of convex volume estimation.

2.1 INTRODUCTION TO MARKOV CHAIN MONTE CARLO (MCMC)

Markov chain Monte Carlo (MCMC) methods are sampling algorithms that are often used to solve multi-dimensional integrals and optimization problems in machine learning, physics, statistics and decision analysis, when no direct analytical method for sampling is available (Andrieu et al., 2003). Generating unbiased samples from a probability distribution is a prerequisite for inferring knowledge about the distribution.

Monte Carlo (MC) is a synonym for learning about probability models by simulating them, e.g. wherever a random process can be simulated, probabilities and expectations can be calculated by averaging over the simulations (Geyer, 1998). The core idea of MC simulation is to use independent and identically distributed (i.i.d.) samples from the target density $p(x)$ to approximate the target density (Andrieu et al., 2003). When the drawn samples are independent, the law of large numbers ensures that the approximation can be made as accurate as desired by increasing the number of samples (Gilks et al., 1996). If $p(x)$ is not straightforward to sample from, rejection sampling, importance sampling, or MCMC methods can be used (Andrieu et al., 2003).

A **Markov chain** is a sequence of random variables $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots$ taking values in an arbitrary state space S and satisfying the Markov property that the next state only depends on the present state, independent of the past:

$$\Pr(\mathbf{x}^{(t+1)} = j | \mathbf{x}^{(t)} = i, \cdots, \mathbf{x}^{(0)} = i_0) = \Pr(\mathbf{x}^{(t+1)} = j | \mathbf{x}^{(t)} = i), \quad (2.1)$$

where $i_0, \cdots, i_{t-1}, i, j \in S$ and $\Pr(j|i)$ is the transition probability of reaching state j from state i. This transition probability satisfies the two conditions:

$$\Pr(j|i) \geq 0, \qquad i, j \in S \quad (2.2)$$

$$\sum_{j \in S} \Pr(j|i) = 1, \qquad \forall i \in S \quad (2.3)$$

and is stationary if the conditional distribution $\mathbf{x}^{(t+1)}$ given $\mathbf{x}^{(t)}$ does not depend on t. The joint distribution of a Markov chain is determined by the initial distribution (the marginal distribution of $\mathbf{x}^{(0)}$) and the transition probability distribution (the conditional distribution of $\mathbf{x}^{(t+1)}$ given $\mathbf{x}^{(t)}$) (Geyer, 2011).

If the state space is finite, then the initial distribution can be associated with a vector $\pi(\mathbf{x}^{(0)})$, and the transition probabilities can be associated with a matrix $\mathbf{P}$ having elements a_{ij} defined by

$$a_{ij} = \Pr(\mathbf{x}^{(t+1)} = j | \mathbf{x}^{(t)} = i). \tag{2.4}$$

For a countable infinite state space, we can think of an infinite vector and matrix. If the state space is uncountable, we must think of the initial distribution as an unconditional probability distribution and the transition probability distribution as a conditional probability distribution (Geyer, 2011).

A Markov chain is **stationary** (time-homogeneous) if the evolution of the chain in the state space only depends on the current state of the chain and a fixed transition matrix.

A transition matrix is **irreducible**, if, for any state of the Markov chain, there is a positive probability of visiting all other states. A transition matrix is **aperiodic** if the chain does not get trapped in cycles.

If the transition matrix is irreducible and aperiodic, then the Markov chain will converge to the invariant/stationary distribution for any starting point. To ensure that a particular $p(x)$ is the desired invariant distribution, the **detailed balance** condition

$$p(i)a_{ij} = p(j)a_{ji}, \tag{2.5}$$

is sufficient, but not necessary (Andrieu et al., 2003). An MCMC sampler that satisfies detailed balance ensures that the stationary distribution of this chain is the target distribution. An example for an irreducible and aperiodic Markov chain that converges to a stationary distribution is the Markov chain with three states and the transition matrix

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}.$$

Figure 2.1 depicts the transition graph. Choosing the probability vector

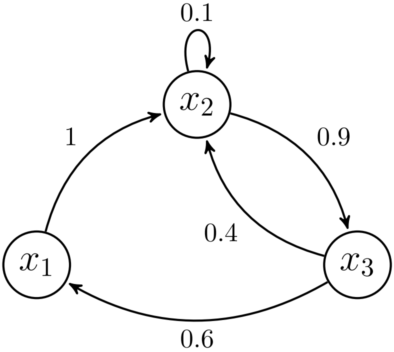


Figure 2.1: Transition graph for the Markov chain example with state space $S = \{x_1, x_2, x_3\}$.

for the initial state $\pi(\mathbf{x}^{(0)}) = [0.5, 0.2, 0.3]$, then

$$\pi(\mathbf{x}^{(0)})\mathbf{P} = [0.18, 0.64, 0.18]\,.$$

After t iterations (multiplications by $\mathbf{P}$), the product $\pi(\mathbf{x}^{(0)})\mathbf{P}^t$ converges to $[0.2213, 0.4098, 0.3689]$. Independent of the initial distribution $\pi(\mathbf{x}^{(0)})$, this Markov chain will converge to this stationary distribution. We now describe the most common MCMC algorithm, the Metropolis-Hastings algorithm.

2.1.1 METROPOLIS-HASTINGS ALGORITHM

The Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) is a simple and versatile MCMC method. It constructs a Markov chain on state space S that is ergodic and stationary with respect to the target probability density $p(x)$. The MH algorithm simulates samples from a proposal distribution. See Algorithm 1 for details of a generic MH algorithm. At first, the starting value is chosen, usually by sampling from the prior distribution. Then, the algorithm continues by looping through three components:

Algorithm 1: Metropolis-Hastings algorithm

```
Choose proposal distribution $q(x)$
Choose starting value $\mathbf{x}^{(0)} := (x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})$
for iteration $t \leftarrow 0, 1, \cdots$ do
    Sample point $\mathbf{x}^{cand}$ from $q(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)})$
    $\mathbf{x}^{(t+1)} \leftarrow \begin{cases} \mathbf{x}^{cand} & \text{with probability } \alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand}) \\ \mathbf{x}^{(t)} & \text{with probability } 1 - \alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand}) \end{cases}$
    where $\alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand}) = \min\left(1, \frac{p(\mathbf{x}^{cand}) q(\mathbf{x}^{(t)} | \mathbf{x}^{cand})}{p(\mathbf{x}^{(t)}) q(\mathbf{x}^{cand} | \mathbf{x}^{(t)})}\right)$
```

1. Generate a candidate (or proposal) solution (or sample) $\mathbf{x}^{cand}$ from the proposal distribution $q(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)})$

2. Use the acceptance criterion $\alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand})$ to compute the acceptance probability based on the proposal distribution and the full joint density $p(\cdot)$

3. Accept the candidate solution with probability α, or reject it with probability $1 - \alpha$

The proposal distribution can either be symmetric or asymmetric. It is symmetric if $q(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}) = q(\mathbf{x}^{(t-1)} | \mathbf{x}^{(t)})$. Often used symmetric proposals are the Gaussian distribution or the Uniform distribution centered at the current state of the chain. For example, using the isotropic multivariate Gaussian distribution, a candidate solution would be sampled according to $\mathbf{x}^{cand} \sim \mathbf{x}^{(t)} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$, where $\mathbf{I}_n$ is the n-dimensional identity matrix and σ^2 is the variance. Algorithms of this form (with a symmetric proposal) are called **random walk Metropolis algorithm**. A symmetric proposal distribution leads to a simplified acceptance function

$$\alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand}) = \min\left(1, \frac{p(\mathbf{x}^{cand}) q(\mathbf{x}^{(t)} | \mathbf{x}^{cand})}{p(\mathbf{x}^{(t)}) q(\mathbf{x}^{cand} | \mathbf{x}^{(t)})}\right) = \left(1, \frac{p(\mathbf{x}^{cand})}{p(\mathbf{x}^{(t)})}\right). \quad (2.6)$$

The ratio $\frac{p(\mathbf{x}^{cand})}{p(\mathbf{x}^{(t)})}$ ensures that the sampler is more likely to visit high probability areas of the joint density. In other words, as put by Sherlock et al. (2010), “uphill” proposals are always accepted, whereas “downhill” proposals are accepted with a probability equal to the relative “heights” of the posterior at the proposed and current values. Finding a good proposal for a particular problem can be difficult. Assuming one would sample the proposed moves from some fixed symmetric distribution, e.g., $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_n)$, then an important issue becomes how to scale the proposal, e.g., how to choose σ. If the variance is set too small, almost all proposed values will be accepted and the chain will be slow in exploring the space. Contrary, if the variance is chosen too high, the proposed moves will nearly all end up in low-probability areas of the target distribution and thus be rejected. In both cases, the convergence rate of the algorithm will be slow, and the chain should not be used for inference. Figures 2.2 and 2.3 illustrate this problem in a 2D example. To avoid the often manual search for improved proposal distributions, adaptive MCMC methods have been introduced.

2.1.2 Adaptive Markov-Chain Monte Carlo

As shown in the previous section, it is crucial to find the right associated parameters for an MCMC method. Adaptive MCMC methods have been developed to avoid manual tuning. They aim to learn parameters while running and automatically adapt the proposal distribution accordingly. The historically first of these methods, the Adaptive Proposal (AP) algorithm (Haario et al., 1999), uses a Gaussian proposal distribution $\mathcal{N}(\mathbf{x}^{(t)}, r^2\mathbf{C}^{(t)})$ centered at the current state $\mathbf{x}^{(t)}$ with a covariance matrix $\mathbf{C}$ calculated from a *fixed* number of previous iterations and a scaling factor r. Its successor, the adaptive Metropolis (AM) (Haario et al., 2001) algorithm, uses the *full history* of the process for adaptation. AM is non-Markovian, but has correct ergodic properties (Haario et al., 2001). AM automatically adapts the covariance matrix $\mathbf{C}$, as is summarized in Algorithm 2. Two conditions ensure convergence of an adaptive MCMC algorithm to its target distribution: diminishing (vanishing) adaptation and containment (Roberts and Rosenthal, 2007). Vanishing adaptation ensures that the covariance depends less and less on recently visited samples (Andrieu and Thoms, 2008). Containment (bounded convergence) requires that the time to stationarity of a transition kernel remains bounded in probability as $t \to \infty$. With increasing iteration count t of the AM algorithm, the adaptation of the covariance vanishes because of the chosen

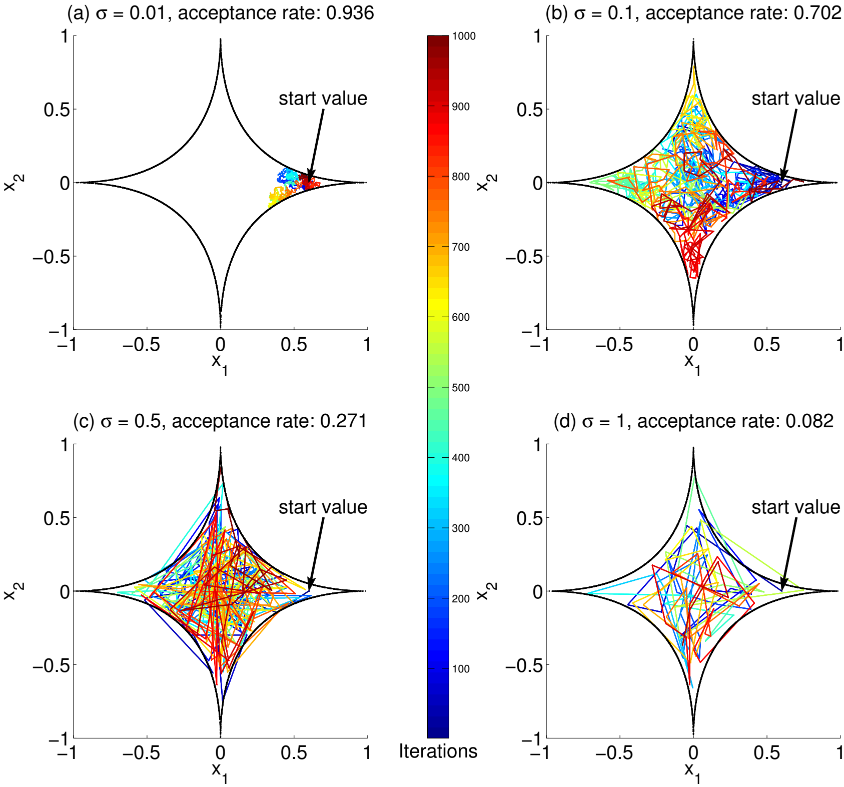
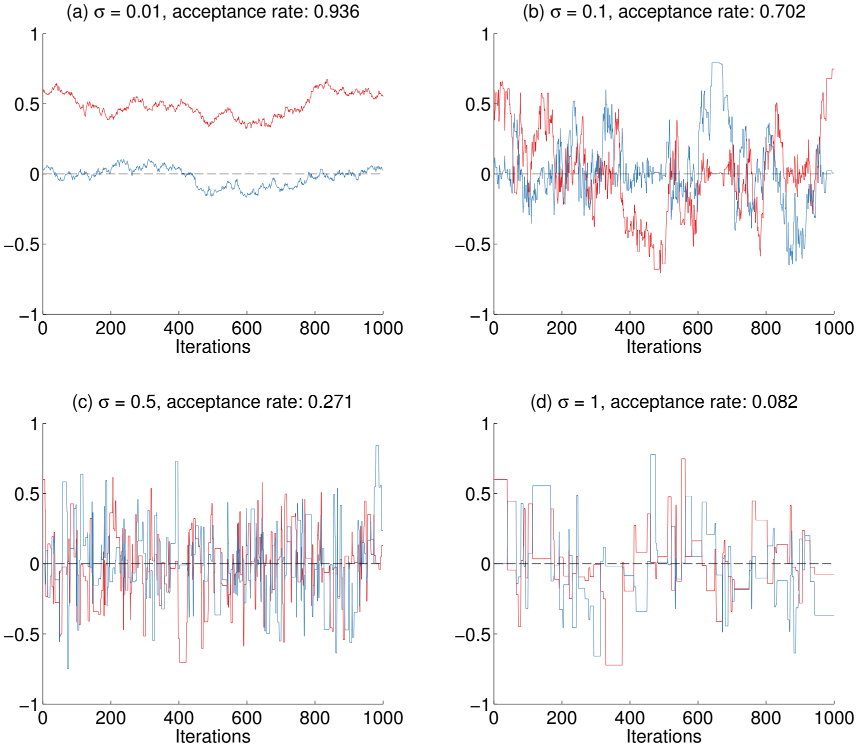


Figure 2.2: The Metropolis-Hastings (MH) algorithm is sensitive to the choice of proposal distribution. The region we want to sample from is the $L_{0.5}$*-ball in two dimensions; the outline is shown in black. In all four examples, we start the chain at* $x = [0.6, 0]$*, use a Gaussian proposal, and let the MH algorithm run for 1000 iterations. Only the standard deviation parameter* σ *of the Gaussian distribution differs. We show the trajectories of obtained samples, where colors indicate the number of iterations. (a) The chosen* σ *is too small, and the chain is far from exploring the entire space. (b) The chain is exploring the entire space, but the samples are highly correlated. (c) With a good choice of* σ*, the chain is mixing well and provides independent samples that approximate the target well. (d) The chosen* σ *is too large, the chain is not moving much because most of the proposed moves are rejected.*


(a) σ = 0.01, acceptance rate: 0.936

(b) σ = 0.1, acceptance rate: 0.702

(c) σ = 0.5, acceptance rate: 0.271

(d) σ = 1, acceptance rate: 0.082

Figure 2.3: Trace plots show how well a Markov chain is mixing. The target region is the two-dimensional $L_{0.5}$-ball, for visualization see Figure 2.2. The trace plot of X_1 is shown in red, of X_2 in blue. The four subplots differ only in the choice of σ. (a) A σ too small leads to a Markov chain that is poorly mixing because most of the proposed steps are accepted. (b) The chain is exploring the whole space, but the samples are highly correlated. (c) With a good choice of σ, the chain is mixing well and provides samples that approximatethe target well. (d) A σ too high leads to a small acceptance rate and thus also to a poor mixing.

Algorithm 2: The AM algorithm

```
Input  : Initial $\mathbf{x}^{(0)}$, $\mathbf{m}^{(0)}$, $\mathbf{C}^{(0)}$, $r$, number of iterations $K$
Output : Unbiased sample $\mathbf{x}^{(0)}, \cdots, \mathbf{x}^{(K)}$ from target distribution
         $p(\mathbf{x})$
for iteration $t \leftarrow 0, 1, \cdots K-1$ do
    Sample point $\mathbf{x}^{cand}$ from $\mathcal{N}\left(\mathbf{x}^{(t)}, r^2\mathbf{C}^{(t)}\right)$
    $\mathbf{x}^{(t+1)} \leftarrow \begin{cases} \mathbf{x}^{cand} & \text{with probability } \alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand}) \\ \mathbf{x}^{(t)} & \text{with probability } 1-\alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand}) \end{cases}$
    where $\alpha(\mathbf{x}^{(t)}, \mathbf{x}^{cand}) = \min\left(1, \frac{p(\mathbf{x}^{cand})}{p(\mathbf{x}^{(t)})}\right)$
    update running mean and covariance:
    $\mathbf{m}^{(t+1)} = \mathbf{m}^{(t)} + \frac{1}{t+1}\left(\mathbf{x}^{(t+1)} - \mathbf{m}^{(t)}\right)$
    $\mathbf{C}^{(t+1)} = \mathbf{C}^{(t)} + \frac{1}{t+1}\left(\left(\mathbf{x}^{(t+1)} - \mathbf{m}^{(t)}\right)\left(\mathbf{x}^{(t+1)} - \mathbf{m}^{(t)}\right)^T - \mathbf{C}^{(t)}\right)$
```

pre-factor of $\frac{1}{t+1}$. This vanishing adaptation is essential to prove ergodicity for the AM algorithm.

2.1.3 Advantages and disadvantages of adaptive MCMC

The choice of a proposal distribution for a non-adaptive MCMC method is crucial to its success (e.g. rapid convergence) (Rosenthal et al., 2011). The search for the right proposal distribution can be demanding because it is often done manually and especially in higher dimensions this is difficult. Adaptive MCMC methods overcome this difficulty by automatically learning a good proposal. Using adaptive proposals, the stationarity of the target $p(x)$ may not always be preserved, but stationarity can be guaranteed if adaptations are designed to satisfy certain conditions, see section 2.1.2. Unfortunately, for most practical applications it is impossible to verify those conditions directly (Yang, 2016). When using adaptive MCMC methods in practice usually no guarantees of convergence can be given (Yang, 2016). To circumvent this problem, it is possible to first use an adaptive MCMC algorithm to learn the proposal distribution and then use a non-adaptive MCMC algorithm with this proposal for which convergence is guaranteed.

2.2 BAYESIAN INFERENCE

MCMC methods are widely applied in statistical inference, especially in Bayesian inference. Statistical inference uses sampling to draw conclusions about quantities that are not observed, based on data that can have underlying variations. According to Casella (2008), there are three approaches to statistics: the Frequentists, who see sampling as infinite, such that experiments are repeatable and decision rules can be sharp. The Bayesians who treat unknown quantities probabilistically and say that the state of the world can always be updated. The Likelihoodists who do single sample inference based on maximizing the likelihood function, and who are Bayesians (but don't know it). The following information about Bayesian inference is adapted from Gelman et al. (2014).

In Bayesian inference, the aim is to compute and use the full posterior probability distribution over a set of random variables by updating prior information with new data. Bayesian data analysis can be divided into three steps

1. Setting up a full probability model
2. Conditioning on observed data
3. Evaluating the fit of the model and the implications of the posterior distribution

Probability statements are the Bayesian statistical conclusions that are made about parameters θ, or unobserved data $\tilde{y}$. These probability statements are conditional on the observed data y, and in our notation are written as $p(\theta|y)$ or $p(\tilde{y}|y)$.

Bayes' theorem is stated as

$$p(\theta|y) = \frac{p(\theta)p(y|\theta)}{p(y)}, \tag{2.7}$$

where $p(\theta)$ is the prior distribution, $p(\theta|y)$ is the posterior distribution and, $p(y)$ the marginal distribution of y, i.e., the probability of observing y without prior knowledge. For continuous θ, $p(y) = \int_{\theta} p(\theta)p(y|\theta)d\theta$. The sampling distribution $p(y|\theta)$ (also called likelihood) is the probability of observing y, given θ. Omitting the normalization factor $p(y)$, which, assuming fixed y, is a constant, gives the unnormalized posterior density $p(\theta|y) \propto p(\theta)p(y|\theta)$.

2.2.1 Approximate Bayesian Computation (ABC)

If likelihood functions are computationally intractable or too costly to evaluate, Approximate Bayesian Computation (ABC) methods can be used (Toni et al., 2009). There, the calculation of the likelihood is replaced by a comparison between the observed and simulated data. As a conceptual overview, Figure 2.4 (Sunnåker et al., 2013) shows a visualization of the method.

ABC methods can generate samples from a distribution that is hoped to be close to the real posterior distribution of interest. Since ABC methods do not require the computation of a likelihood function, they can be used to estimate posterior distributions of parameters for simulation-based models. The general procedure is as follows (Toni et al., 2009):

1. Sample a candidate parameter vector $\theta^\star$ from some proposal distribution $p(\theta)$
2. Simulate a dataset $y^\star$ from the model described by a conditional probability distribution $f(y|\theta^\star)$.
3. Compare the simulated dataset, $y^\star$, with the experimental data, y_0, using a distance function, d, and tolerance ϵ; if $d(y_0, y^\star) \leq \epsilon$, accept $\theta^\star$. The tolerance $\epsilon \geq 0$ is the desired level of agreement between y_0 and $y^\star$.

The resulting sample of parameters is from a distribution $p(\theta|d(y_0, y^\star) \leq \epsilon)$ and will be a good approximation of the posterior distribution $p(\theta|y)$ if ϵ is small enough (Toni et al., 2009). The distance function d is based on a given metric, the most popular choice is the Euclidean distance, and determines the level of discrepancy between $y^\star$ and y_0 (Sunnåker et al.,

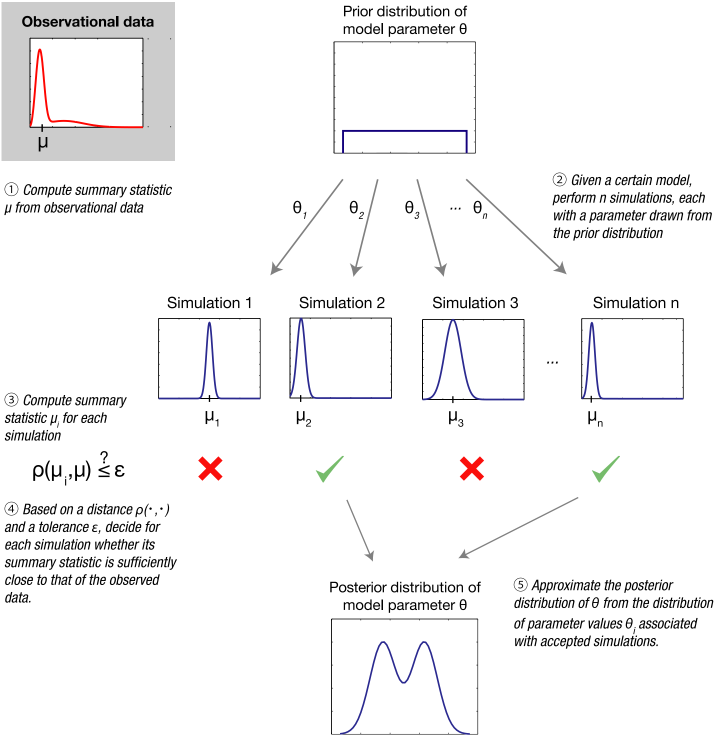

Observational data
μ
① Compute summary statistic μ from observational data
Prior distribution of model parameter θ
θ1 θ2 θ3 ... θn
② Given a certain model, perform n simulations, each with a parameter drawn from the prior distribution
Simulation 1
Simulation 2
Simulation 3
Simulation n
...
μ1 μ2 μ3 μn
③ Compute summary statistic μi for each simulation
ρ(μi,μ) ≤? ε
✗ ✓ ✗ ✓
④ Based on a distance ρ(·,·) and a tolerance ε, decide for each simulation whether its summary statistic is sufficiently close to that of the observed data.
Posterior distribution of model parameter θ
⑤ Approximate the posterior distribution of θ from the distribution of parameter values θi associated with accepted simulations.

Figure 2.4: Parameter estimation by Approximate Bayesian computation: a conceptual overview. (Source: Sunnåker et al. (2013))

2013). The often hard to define distance function between the datasets $d(y_0, y^\star)$ can be replaced by a distance specified on a set of m summary statistics (Toni et al., 2009), where $s(y_0)$ are the summary statistics of the experimental data and $s(y^\star)$ of the simulated data. Using a weighted Euclidean distance

$$d(s(y_0), s(y^\star)) = \sqrt{\sum_{i=1}^{m} \left(\frac{s_i(y_0) - s_i(y^\star)}{\sigma_i} \right)^2}, \tag{2.8}$$

where σ_i is an estimate of the prior predictive standard deviation of the ith summary statistic, normalizes the summaries, such that the distance is not dominated by one summary (Prangle et al., 2016).

The ABC rejection algorithm (Pritchard et al., 1999) is the most basic form of ABC, see Algorithm 3.

Algorithm 3: ABC rejection

```
for $i \leftarrow 1$ to $N$ do
    Sample $\theta^\star$ from $p(\theta)$
    Simulate a dataset $y^\star$ from $f(y|\theta^\star)$
    if $d(y_0, y^\star) \leq \epsilon$ then
        accept $\theta^\star$
    else
        reject
```

If the prior distribution is very different from the posterior distribution, the acceptance rate is low. To circumvent this disadvantage, one way is to use sequential importance sampling, which does not directly sample from the posterior, but from a series of intermediate distributions until the distance to the posterior is less than a specified ϵ_T. The resulting sequential Monte Carlo ABC algorithm, known as ABC-SMC (Toni et al., 2009; Barnes et al., 2011) proceeds as described in Algorithms 4 and 5. The tolerated distance between the simulated and the experimental data set, ϵ, equals ∞ at the start and is lowered successively during the iterations, until it equals the defined final value ϵ_T. With every new ϵ, N data sets with a distance less than ϵ to the experimental data, and the corresponding parameter vectors, are obtained (see Algorithm 5). The new ϵ is calculated automatically, only the final value, ϵ_T, has to be specified.

Algorithm 4: ABC-SMC

```
Input  : final value $\epsilon_T$
Output : $N$ samples $\theta^1, \theta^2, \cdots, \theta^N$ from the posterior
         $p(\theta | d(y^\star, y_0) \leq \epsilon_T)$
Initialize $\epsilon = \infty$
Set iteration indicator $t = 0$
while $\epsilon > \epsilon_T$ do
    for $i \leftarrow 1$ to $N$ do
        $[\theta^{\star\star}, y^\star] \leftarrow$ getDataset($t$,$\epsilon$,$i$)
        Set $\theta_t^i = \theta^{\star\star}$
        Set $d_t^i = d(y^\star, y_0)$
        Calculate weights $w_t^i = \begin{cases} 1 & \text{if } t == 0 \\ \frac{p(\theta_t^i)}{\sum_{j=1}^{N} w_{t-1}^j K_t(\theta_t^i | \theta_{t-1}^j)} & \text{else} \end{cases}$
    Normalize the weights
    Determine $\epsilon$ such that $\Pr(d_t \leq \epsilon) = 0.9$
    $t = t + 1$
```

2.2.2 Using ABC for model selection

If the model is not known, or a set of candidate models $\mathcal{M}$ is available, models can be ranked by looking at their marginal posterior distributions given data y_0. To estimate this marginal posterior distribution of a model, $p(m|y_0)$, a joint space of model indicators, $m = 1, 2, \cdots, |\mathcal{M}|$, and corresponding model parameters, θ, is defined (Toni et al., 2009). Bayesian inference on the joint space now allows for model selection. The ABC rejection algorithm with model selection proceeds as described in Algorithm 6 (Grelaud et al., 2009).

Algorithm 5: Get a dataset $y^\star$ that is ϵ-close to the experimental dataset y_0 and the corresponding parameters $\theta^{\star\star}$

```
$[\theta^{\star\star}, y^\star] \leftarrow$ getDataset($t$,$\epsilon$,$i$)
if $t$==0 then
    repeat
        repeat
            Sample $\theta^{\star\star}$ independently from $p(\theta)$
        until $p(\theta^{\star\star}) \neq 0$
        Simulate a candidate dataset $y^\star$ from $f(y|\theta^{\star\star})$
    until $d(y_0, y^\star) \leq \epsilon$
else
    repeat
        repeat
            Sample $\theta^\star$ from previous iteration $\theta^i_{t-1}$ with weights $w_{t-1}$.
            Perturb $\theta^\star$ to get $\theta^{\star\star} \sim K_t(\theta|\theta^\star)$ where $K_t$ is the perturbation kernel
        until $p(\theta^{\star\star}) \neq 0$
        Simulate a candidate dataset $y^\star$ from $f(y|\theta^{\star\star})$
    until $d(y_0, y^\star) \leq \epsilon$
```

Algorithm 6: ABC rejection for model selection

```
repeat
    Sample $m^\star$ from prior $p(m)$
    Sample $\theta^\star$ from $p(\theta|m^\star)$
    Simulate a dataset $y^\star$ from $f(y|\theta^\star, m^\star)$
    if $d(y_0, y^\star) \leq \epsilon$ then
        accept $(m^\star, \theta^\star)$
    else
        reject
until $N$ samples have been accepted
```

The marginal posterior distribution, $p(m = m'|y_0)$, is approximated by

$$p(m = m'|y_0) = \frac{\text{\# accepted parameter vectors for model } m'}{N}. \tag{2.9}$$

ABC-SMC can also be used for model selection in a similar way.

2.3 GAUSSIAN ADAPTATION

Gaussian Adaptation (GaA) (Kjellström and Taxen, 1981) is a stochastic search heuristic originally developed for electrical network design. It can be used for design centering (Müller and Sbalzarini, 2011), black-box sampling (Müller and Sbalzarini, 2010a), and optimization (Müller and Sbalzarini, 2010b). In optimization, the goal is to minimize an objective function $f : \mathbb{R}^n \mapsto \mathbb{R}$. In each iteration g, a new candidate solution $\mathbf{x}^{(g)} \in \mathbb{R}^n$ is generated by sampling from a multivariate Gaussian proposal distribution, $\mathcal{N}(\mathbf{m}^{(g-1)}, \mathbf{C}^{(g-1)})$, with mean $\mathbf{m}^{(g-1)}$ and $n \times n$ covariance matrix $\mathbf{C}^{(g-1)}$. $\mathbf{C}^{(g-1)}$ is decomposed:

$$\mathbf{C}^{(g-1)} = r^2(\mathbf{Q}^{(g-1)})(\mathbf{Q}^{(g-1)})^T, \tag{2.10}$$

where r is the scalar step size and $\mathbf{Q}^{(g-1)}$ is the normalized square root of $\mathbf{C}^{(g-1)}$.

If $f(\mathbf{x}^{(g)})$ is below a threshold c_T, it is used to update the mean $\mathbf{m}^{(g)}$ according to

$$\mathbf{m}^{(g)} = \left(1 - \frac{1}{N_m}\right)\mathbf{m}^{(g-1)} + \frac{1}{N_m}\mathbf{x}^{(g)}, \tag{2.11}$$

and the covariance matrix $\mathbf{C}^{(g)}$ according to

$$\mathbf{C}^{(g)} = (1 - \frac{1}{N_C})\mathbf{C}^{(g-1)} + \frac{1}{N_C}(\mathbf{x}^{(g)} - \mathbf{m}^{(g-1)})(\mathbf{x}^{(g)} - \mathbf{m}^{(g-1)})^T, \tag{2.12}$$

where the weighting factors N_m and N_C control the influence of the accepted sample on the mean and the covariance, respectively.

The acceptance threshold c_T is adapted according to

$$c_T^{(g)} = \left(1 - \frac{1}{N_T}\right) c_T^{(g-1)} + \frac{1}{N_T} f(\mathbf{x}^{(g)}), \tag{2.13}$$

where N_T is a weighting factor. The step size r is then increased by the expansion factor f_e:

$$r^{(g)} = f_e r^{(g-1)}, \qquad f_e > 1. \tag{2.14}$$

If $f(\mathbf{x}^{(g)}) \geq c_T$, the mean, the covariance matrix, and the acceptance threshold are not updated and the step size r is decreased by the contraction factor f_c:

$$r^{(g)} = f_c r^{(g-1)}, \qquad f_c < 1. \tag{2.15}$$

Kjellström introduced a mathematical equivalent, but numerically more robust update rule as an alternative to Eq. 2.12, where he uses the incremental change of the covariance matrix

$$\mathbf{\Delta C}^{(g)} = (1 - \frac{1}{N_C})\mathbf{I}^{(g-1)} + \frac{1}{N_C}(\eta^{(g-1)}\eta^{(g-1)})^T, \tag{2.16}$$

to update

$$\mathbf{\Delta Q}^{(g)} = (\mathbf{\Delta C}^{(g)})^{\frac{1}{2}}, \tag{2.17}$$

where $\eta^{(g-1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The square root $\mathbf{Q}^{(g)}$ of the covariance matrix is then updated according to:

$$\mathbf{Q}^{(g)} = \mathbf{Q}^{(g-1)} \mathbf{\Delta Q}^{(g+1)}. \tag{2.18}$$

$\mathbf{Q}^{(g)}$ is normalized such that $\det(\mathbf{Q}^{(g)}) = 1$. Thus, the volume of the covariance is only controlled by $r^{(g)}$.

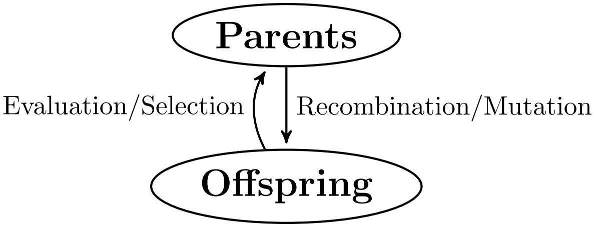


Figure 2.5: The general procedure of an evolution strategy: through recombination and mutation, a population of parents is creating offspring. After evaluation, the parents of the new generation are selected. This generation cycle is continued until some termination criterion is met.

2.4 COVARIANCE MATRIX ADAPTATION EVOLUTION STRATEGY

An evolution strategy (ES) (Beyer and Schwefel, 2002) is a stochastic optimization technique based on concepts from biological evolution, where mutation, recombination, and selection are applied to a population of individuals (candidate solutions, samples) in order to evolve a fitter generation (better solutions); for visualization see Figure 2.5.

In the Covariance Matrix Adaptation (CMA) Evolution Strategy (Hansen, 2016), a population is sampled from a multivariate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ with mean $\mathbf{m} \in \mathbb{R}^n$ and covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$. In each iteration (generation) new candidate solutions are sampled from this distribution and then used to adapt the distribution such that the whole population moves towards a region in parameter space that has better fitness values, e.g., smaller objective function in a minimization task.

For generation $g = 0, 1, 2, \cdots$, new candidate solutions are sampled according to

$$\mathbf{x}_k^{(g+1)} \sim \mathbf{m}^{(m)} + \sigma^{(g)} \mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)}), \text{ for } k = 1, \cdots, \lambda, \tag{2.19}$$

where $\sigma^{(g)} \in \mathbb{R}_+$ is the step size at generation g, and λ is the population size (the number of offspring). After the sampling step, the mean, the covariance matrix, and the step size are adapted.

2.4.1 Adaptation of the mean

The objective function is evaluated for all λ candidate solutions. The samples are then sorted, and the weighted average of the μ best candidate solutions is used as the new mean:

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^{(g+1)}, \tag{2.20}$$

with

$$\sum_{i=1}^{\mu} w_i = 1, w_1 \geq w_2 \geq \cdots \geq w_\mu > 0, \tag{2.21}$$

where w_i are positive weights for recombination.

2.4.2 Adaptation of the covariance matrix

The covariance matrix adapts to the underlying objective function by learning the pairwise dependences between variables in order to accelerate convergence. The covariance matrix adaptation consists of three subprocedures: rank-μ-update, rank-one-update, and cumulation.

Rank-μ-update

Assuming the population contains enough information to estimate a covariance matrix from it reliably, the samples from (2.19) can be used to estimate the covariance matrix with the empirical covariance:

$$\mathbf{C}_{emp}^{(g+1)} = \frac{1}{\lambda - 1} \sum_{i=1}^{\lambda} \left(\mathbf{x}_i^{(g+1)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} \mathbf{x}_j^{(g+1)} \right) \left(\mathbf{x}_i^{(g+1)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} \mathbf{x}_j^{(g+1)} \right)^T. \tag{2.22}$$

$\mathbf{C}_{emp}^{(g+1)}$ estimates the distribution variance within the sampled points. Instead of using the mean of the actually realized samples as the reference value (see (2.22)), the true mean $\mathbf{m}^{(g)}$ of the sampled distribution can be used and weights w_i can be included:

$$\mathbf{C}_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i \left(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}\right)\left(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}\right)^T. \tag{2.23}$$

$\mathbf{C}_{\mu}^{(g+1)}$ is an estimator for the distribution of selected steps. Sampling from $\mathbf{C}_{\mu}^{(g+1)}$ usually reproduces selected steps. Combining this information with the previous covariance matrix leads to

$$\begin{aligned}\mathbf{C}^{(g+1)} &= (1-c_\mu)\mathbf{C}^{(g)} + c_\mu \frac{1}{\sigma^{(g)^2}} \mathbf{C}_{\mu}^{(g+1)} \\ &= (1-c_\mu)\mathbf{C}^{(g)} + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}^{(g+1)} \mathbf{y}_{i:\lambda}^{(g+1)^T},\end{aligned} \tag{2.24}$$

where $c_\mu \leq 1$ is the learning rate and $\mathbf{y}_{i:\lambda}^{(g+1)} = (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})/\sigma^{(g)}$. This update was introduced by Hansen et al. (2003) and is called rank-μ-update because it learns information in μ directions by adding the variances of the μ selected points into the matrix.

RANK-ONE-UPDATE

In contrast to the rank-μ-update, which uses all selected steps from a single generation, the rank-one-update uses a single selected step $\mathbf{y}$ only. This adaptation increases the likelihood for previously successful steps to appear again. The rank-one update for the covariance matrix is

$$\mathbf{C}^{(g+1)} = (1-c_1)\mathbf{C}^{(g)} + c_1 \mathbf{y}\mathbf{y}^T. \tag{2.25}$$

Cumulation

The evolution path is the sequence of successive steps taken over the past generations. It can be expressed as a sum of consecutive steps, e.g., the evolution path of three steps of the distribution mean is:

$$\frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} + \frac{\mathbf{m}^{(g)} - \mathbf{m}^{(g-1)}}{\sigma^{(g-1)}} + \frac{\mathbf{m}^{(g-1)} - \mathbf{m}^{(g-2)}}{\sigma^{(g-2)}}. \tag{2.26}$$

The evolution path $\mathbf{p}_c$ is constructed using exponential smoothing (Hansen, 2016) and starts with $\mathbf{p}_c^{(0)} = \mathbf{0}$.

$$\mathbf{p}_c^{(g+1)} = (1 - c_c)\mathbf{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}, \tag{2.27}$$

where $c_c \leq 1$ and $1/c_c$ is the backward time horizon of the evolution path $\mathbf{p}_c$ and $\sqrt{c_c(2 - c_c)\mu_{\text{eff}}}$ is the normalization constant for $\mathbf{p}_c$. The rank-one update for the covariance matrix via the evolution path equals (Hansen and Ostermeier, 1996):

$$\mathbf{C}^{(g+1)} = (1 - c_1)\mathbf{C}^{(g)} + c_1 \mathbf{p}_c^{(g+1)} \mathbf{p}_c^{(g+1)^T}. \tag{2.28}$$

Combining the updates

The final CMA update then reads:

$$\mathbf{C}^{(g+1)} = (1 - c_1 - c_\mu)\mathbf{C}^{(g)} + c_1 \underbrace{\mathbf{p}_c^{(g+1)} \mathbf{p}_c^{(g+1)^T}}_{\text{rank-one update}} \underbrace{+ c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}^{(g+1)} \left(\mathbf{y}_{i:\lambda}^{(g+1)}\right)^T}_{\text{rank-}\mu\text{ update}}, \tag{2.29}$$

where $c_1, c_\mu \in \mathbb{R}_+$ are the weights for the updates and $c_1 + c_\mu \leq 1$.

2.4.3 STEP SIZE CONTROL

The optimal step size cannot be known a priori. The evolution path is hence used to control the overall scale $\sigma^{(g)}$ of the sampling distribution. The length of the evolution path is used for step-size control. A short evolution path, where single steps cancel each other if steps are anti-correlated, should lead to a decrease in step size, whereas a long evolution path, where single steps point in similar directions, should lead to an increased step size. With an optimal step size, steps are (approximately) perpendicular in expectation, i.e., uncorrelated (Hansen, 2016). The decision whether the evolution path is long or short is made after comparing the length of the path with its expected length under random selection because under random selection consecutive steps are independent and thus uncorrelated. The evolution path $\mathbf{p}_\sigma$ is constructed similarly to equation (2.27):

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{(g)^{-\frac{1}{2}}} \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}, \quad (2.30)$$

where the initial evolution path for the step size $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$, $c_\sigma < 1$ is the backward time horizon of the evolution path $\mathbf{p}_\sigma$ and $\sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}$ is a normalization constant. The transformation $\mathbf{C}^{(g)^{-\frac{1}{2}}}$ re-scales the current step $\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}$, where $\mathbf{C}^{(g)^{-\frac{1}{2}}}$ is found through Eigen-decomposition of $\mathbf{C}^{(g)}$. The expected length of a standard normal random vector $\mathbb{E}\,||\mathcal{N}(\mathbf{0}, \mathbf{I})|| \approx \sqrt{n}$ is the expected length of the evolution path $\mathbf{p}_\sigma^{(g+1)}$. If $||\mathbf{p}_\sigma^{(g+1)}||$ equals $\mathbb{E}\,||\mathcal{N}(\mathbf{0}, \mathbf{I})||$ then the step size $\sigma^{(g)}$ is unchanged, if it is larger, the step size is increased and if it is smaller, the step size is decreased, see the complete update:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_\sigma^{(g+1)}||}{\mathbb{E}\,||\mathcal{N}(\mathbf{0}, \mathbf{I})||} - 1\right)\right), \quad (2.31)$$

where $d_\sigma \approx 1$ is a damping parameter

2.5 CONVEX VOLUME ESTIMATION

Volume computation is an important problem in science and engineering. The computational cost of computing the volume of a convex body $A \in \mathbb{R}^n$, where A is determined by a membership oracle (Grötschel et al., 1988), is exponential in n (Bárány and Füredi, 1987). A membership oracle is a black box that checks whether a point $x \in \mathbb{R}^n$ is in A or not. Dyer and Frieze (1988) and Khachiyan (1988, 1989) showed that deterministically computing the volume is NP-hard. Using a randomized algorithm, the volume of a convex body can be approximated to arbitrary precision in polynomial time (Dyer et al., 1991). Dyer, Frieze, and Kannan's algorithm requires $\mathcal{O}^\star(n^{23})$ oracle calls[1] and has been improved in a sequence of papers, for an overview see Simonovits (2003). The fastest known algorithm for convex volume estimation needs $\mathcal{O}^\star(n^4)$ oracle calls[1] (Lovász and Vempala, 2006). We now explain the main concepts of randomized volume approximation algorithms: Rounding and Multiphase MCMC.

2.5.1 ROUNDING

In a rounding algorithm (sandwiching), an affine transformation T is applied to the convex body A, such that $T(A)$ contains the unit ball $B(0,1)$ and is contained in the ball with radius m, $B(0,m)$:

$$B(0,1) \subseteq T(A) \subseteq B(0,m), \tag{2.32}$$

where $B(x,R)$ is the ball centered at x with radius R. The rounded body, $T(A)$, is referred to as K from here on.

2.5.2 MULTIPHASE MCMC (SUBDIVISION AND SAMPLING)

After rounding, a series of, for example, $l = \lceil n \log_2 m \rceil$ (Ge and Ma, 2015), concentric balls $B_i = B(0, r_i)$ is constructed and placed between $B(0,1)$

---

[1] the asterisk in the order notation indicates that logarithmic factors in n are omitted

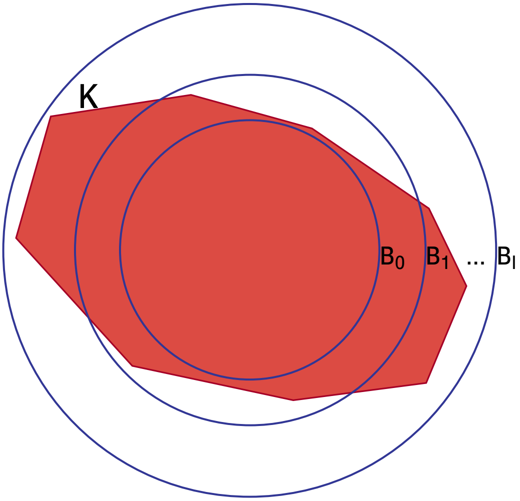


Figure 2.6: Multiphase Monte Carlo: To estimate the volume of the convex body K, an increasing sequence of concentric balls $B_0 \subseteq B_1 \subseteq \cdots \subseteq B_l$ is selected such that $B_0 \subseteq K \subseteq B_l$.

and $B(0,m)$, where

$$r_i = 2^{\frac{i}{n}}, \ \text{ for } i = 0, \cdots, l. \tag{2.33}$$

For an illustration see Figure 2.6. Set

$$K_i = B(0, r_i) \cap K, \tag{2.34}$$

then

$$K_0 = B(0,1) \ \text{ and } K_l = B(0,m) \cap K = K. \tag{2.35}$$

The balls are chosen such that the volume ratio of two consecutive balls is bounded by a constant and thus we also have their intersection with K bounded:

$$\frac{\text{vol}(K_{i+1})}{\text{vol}(K_i)} \leq \frac{\text{vol}(B(0, r_{i+1}))}{\text{vol}(B(0, r_i))} = 2. \tag{2.36}$$

The volume of K is then approximated by

$$\text{vol}(K) \approx \text{vol}(K_l) \approx \frac{\text{vol}(K_l)}{\text{vol}(K_{l-1})} \times \cdots \times \frac{\text{vol}(K_1)}{\text{vol}(K_0)} \times \text{vol}(K_0). \tag{2.37}$$

To estimate the volume ratios, N random points are sampled from K_{i+1} and then the number of points in K_i, c_i, is counted, thus

$$\frac{\mathrm{vol}(K_{i+1})}{\mathrm{vol}(K_i)} = \frac{N}{c_i}. \tag{2.38}$$

To sample uniformly from K_{i+1}, random walk methods are used, such as ball-walks, grid walks or hit-and-run methods. Hit-and-run yields the fastest algorithms today.

Lovász and Vempala (2006) were able to design an $\mathcal{O}^{\star}(n^4)$ algorithm by replacing the sequence of K_i by a sequence of log-concave functions $f_0 \leq f_1 \leq \cdots \leq f_m$ and calculating the ratios of integrals $(\int f_{i-1})/(\int f_i)$. There are still some difficulties in applying the described volume estimation algorithms, as well as a lack of practical implementations (Ge and Ma, 2015). The reasons for this are complex oracles, time-consuming oracle queries, and a very large constant prefactor in the theoretical complexity (Ge and Ma, 2015; Lovász, 1999). This leads to algorithms that are "almost infeasible even in low dimensions" (Ge and Ma, 2015).

CHAPTER

THREE

L_p-ADAPTATION, AN EFFICIENT ALGORITHM UNITING APPROXIMATE DESIGN CENTERING AND VOLUME ESTIMATION

Equipped with the background from Chapter 2, we now introduce our statistical method, L_p-Adaptation.

L_p-Adaptation unites approximate design centering and volume estimation. It iteratively samples the parameter space using the uniform density over L_p-balls as proposal distribution $q(\mathbf{x})$. L_p-balls are a good choice because it is known how to uniformly sample from L_p-balls and how to determine their volume. The choice of L_p-ball reflects assumptions about the noise performance of the parameters of the system. For example, the L_2-ball assumes a Gaussian perturbation prior, L_∞ the worst-case scenario.

An L_p-ball of radius $r > 0$ is defined as

$$L_p^n(r) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_p \leq r\}, \tag{3.1}$$

where

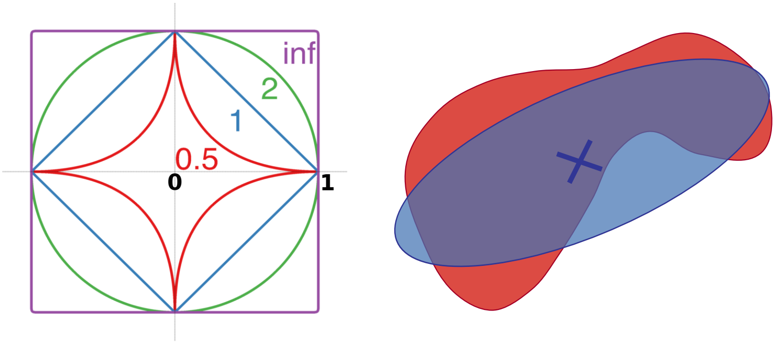
$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}} \tag{3.2}$$

and $p > 0$. L_p-Adaptation successively adapts the proposal distribution to the unknown feasible region A by adapting position, orientation, and aspect ratio of the L_p-balls based on the sampling history. This is inspired by adaptation concepts introduced in bio-inspired optimization (Kjellström and Taxen, 1981; Hansen and Ostermeier, 2001) and leads to better sampling efficiency and approximation quality, especially in feasible regions of high aspect ratio. While strong non-convexity will deteriorate sampling efficiency, our approach is not limited to convex feasible regions, since it can dynamically adapt to different areas of a non-convex feasible region.

The dynamic affine adaptation of the L_p-ball is based on the concept of Gaussian Adaptation (GaA) (Kjellström and Taxen, 1981), which continuously adapts the mean and the covariance matrix (describing correlations and scaling between the parameters) of a Gaussian proposal based on previous sampling success. GaA is a classical optimization heuristic that is linked to maximum-entropy estimation (Müller and Sbalzarini, 2010b) and provides a unifying algorithmic framework for optimization and sampling (Müller and Sbalzarini, 2010a). Based on this analogy, GaA has previously been extended to design centering (Müller and Sbalzarini, 2011). The use of a Gaussian proposal, however, becomes unfavorable for feasible regions of non-elliptic shape. For a more detailed description of GaA, see Section 2.3.

Efficient design centering and robust volume approximation become possible in the same framework by combining the adaptation concept of GaA with the use of L_p-balls (Seifi et al., 1999) as non-Gaussian proposals and with a schedule for changing (decreasing or increasing) target hitting probabilities, where the hitting probability is the probability that a sample from the proposal distribution is feasible. Some examples of unit L_p-balls in two dimensions are illustrated in Figure 3.1a, and for an illustration of a feasible region and a sampling proposal, see Figure 3.1b.

If L_p-Adaptation runs until it has reached a stationary state (stationary random process), i.e., the statistics of the proposal distribution, such as



(a) **(b)**

Figure 3.1: (a) Illustrations of some unit L_p-balls in two dimensions: $p \to \infty$: the L_∞-ball (the rectangle), $p = 2$: the L_2-ball (the circle), $p = 1$: the L_1-ball (the diamond), and $p = 0.5$: the $L_{0.5}$-ball (the star). (b) Samples are randomly drawn from the proposal distribution (blue), which is an affine transformation of an L_p-ball. The hitting probability P is the probability that the sample lies inside the feasible region A (red). For uniform sampling, it is given by the overlap area between the proposal and the feasible region.

hitting probability and mean volume, do no longer change on average, although the samples still stochastically fluctuate. Averaging over the last evaluations then provides the final estimates of the design center and volume of the feasible region. Since the process converges in distribution, i.e., the random process becomes stationary, averaging is meaningful.

The design center is approximated by the mean of the current L_p-ball, and the volume estimate is of the form

$$\widetilde{\mathrm{vol}}(A) \approx P \cdot \mathrm{vol}(L_p^n(r)), \tag{3.3}$$

where P is the hitting probability,

$$\mathrm{vol}(L_p^n(r)) = \frac{(2r \cdot \Gamma(1 + \frac{1}{p}))^n}{\Gamma(1 + \frac{n}{p})} \tag{3.4}$$

is the volume of the n-dimensional L_p-ball with radius r, and

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{3.5}$$

is the Gamma function.

In the remainder of this chapter, we first explain the concepts of L_p-Adaptation. Then we detail its steps and describe how to average over covariance matrices. This is followed by a description on the behavior of L_p-balls and a demonstration on which information we can extract from the generated samples and from the evolution of the proposal distribution. We end this chapter with a short conclusion.

3.1 CONCEPTS OF L_p-ADAPTATION

In this section we introduce the two main concepts of L_p-Adaptation: maximizing robustness and exploiting the hitting probability.

3.1.1 MAXIMIZING ROBUSTNESS

L_p-Adaptation can be interpreted as a synthetic evolutionary process that tries to maximize the robustness, rather than the fitness, of the underlying system. Robustness is measured in terms of the volume $\mathrm{vol}(L_p^n(r))$ of an n-dimensional L_p-ball with radius r, of which a certain fraction P (i.e., the target hitting probability) overlaps with the feasible region A.

Using the information gained during each iteration, the algorithm adapts the proposal to find the largest proposal with a given hitting probability. This is illustrated in Figure 3.2, showing two proposals that have the same hitting probability, but differ in size.

Let us denote by $\mathcal{L}_p^n = \{p, \mathbf{m}, \mathbf{C}\}$ the set of all L_p-balls where p is the type of ball, $\mathbf{m} \in \mathbb{R}^n$ the center of the L_p-ball, and $\mathbf{C} \in \mathcal{S}_+^{n\times n}$ is a symmetric positive-definite (covariance) matrix defining the affine map for transformation of the L_p-ball. L_p-Adaptation then seeks to maximize the following

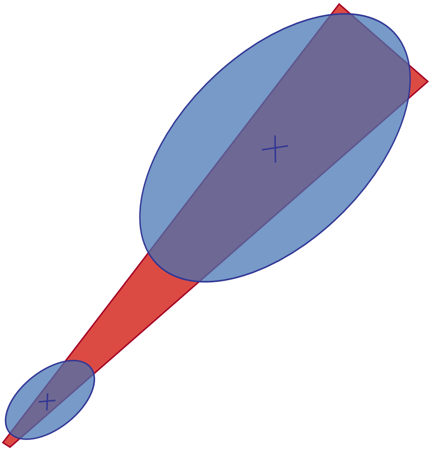


Figure 3.2: The two proposal distributions (blue) have the same hitting probability over the red feasible region. The adaptation scheme increases the size (volume) of the proposal under constant hitting probability.

robustness criterion:

$$\begin{aligned} \max_{L_p^n \in \mathcal{L}_p^n} \quad & \mathrm{vol}(L_p^n) \\ \text{s.t.} \quad & \forall \mathbf{x} \in L_p^n \\ & \Pr(\mathbf{x} \in A) = P \\ & \mathbf{m} \in A. \end{aligned} \tag{3.6}$$

The volume of an L_p-ball is completely determined by the volume of the unit L_p-ball (with zero mean and n-dimensional identity matrix $\mathbf{I}_n$) and the determinant of the matrix $\mathbf{C}$. Thus, the robustness criterion can be rewritten as a non-convex log-det maximization problem:

$$\begin{aligned} \max_{L_p^n \in \mathcal{L}_p^n} \quad & \log\det \mathbf{C} \\ \text{s.t.} \quad & \forall \mathbf{x} \in L_p^n \\ & \Pr(\mathbf{x} \in A) = P \\ & \mathbf{m} \in A. \end{aligned} \tag{3.7}$$

This objective function provides a natural non-convex extension of the maximum inscribed ellipsoid method of Seifi et al. (1999). For instance, if A is

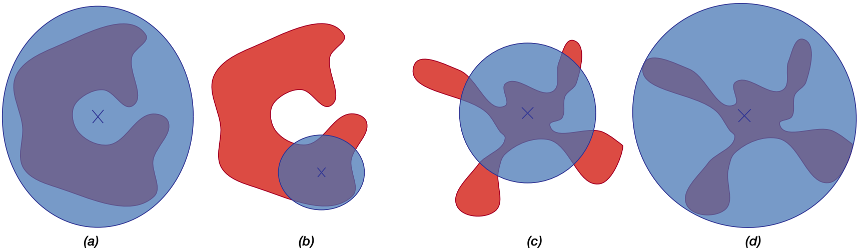


Figure 3.3: Illustration of the effect of different hitting probabilities on a feasible region. (a) Low hitting probabilities may lead to infeasible design centers. (b) Increasing the hitting probability (e.g. shrinking the proposal) leads to a feasible design center. (c) For volume estimation, high hitting probabilities may under-estimate the volume. (d) Decreasing the hitting probability leads to better volume approximation.

a convex polyhedron with known parameterization and $P = 1$, then Problem (3.7) is a convex problem that can be efficiently solved using interior point methods. However, in the general case, no efficient algorithms exist to solve Problem (3.7). L_p-Adaptation approximately solves this problem by using a synthetic evolutionary process, as described in Section 3.2.

3.1.2 Exploiting the hitting probability

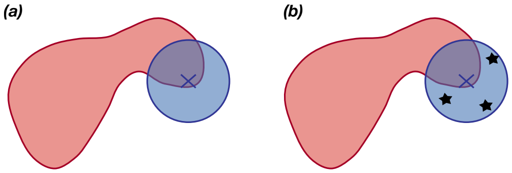
An important feature of the sampling and adaptation process is the ability to control the *target hitting probability* P, i.e., the probability of hitting the feasible region A with a sample. This allows L_p-Adaptation to provide simultaneous design center and volume estimates.

The hitting probability must be neither too low nor too high. Low hitting probabilities lead to low sampling efficiencies. High hitting probabilities result in slower adaptation to the feasible region, which may prevent exploring remote parts of the feasible region. This trade-off is illustrated in Fig. 3.3. For a Gaussian proposal and a convex feasible region, a hitting probability of 1/e (i.e., the inverse of the Euler-Mascheroni constant) is information-theoretically optimal (Kjellström and Taxen, 1981). When sampling uniformly from L_p-balls over non-convex regions, however, no such result is available. We hence dynamically adapt the hitting probability depending on the task at hand, starting from 1/e as an initial value. For design centering, the hitting probability in non-convex regions cannot be too low, as this could lead to an infeasible design center (Fig. 3.3a). It is hence successively increased in order to drive the process toward a feasible design center (Fig. 3.3b). For volume approximation, the hitting probability must not be too high, as this would miss some parts of the feasible region (Fig. 3.3c). In this case, it is hence successively lowered until the volume estimate no longer changes (Fig. 3.3d), leading to a better volume approximation. This strategy is similar to state-of-the-art multi-phase Monte Carlo methods (see Section 2.5) for approximate convex volume estimation (Simonovits, 2003; Vempala, 2010).

3.2 STEPS OF L_p-ADAPTATION

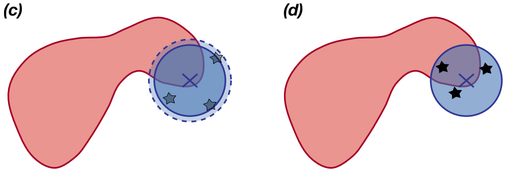
L_p-Adaptation draws samples uniformly from an L_p-ball as detailed in Algorithm 7, and iteratively adapts position, orientation, and aspect ratio of the L_p-ball, see Figure 3.4. This is done by adapting the mean and covariance matrix of an affine mapping applied to the ball prior to sampling. For improved sampling and adaptation efficiency, L_p-Adaptation uses, at each iteration, an adaptive multi-sample strategy (Hansen, 2008) that is considered state-of-the-art in bio-inspired optimization (Hansen and Ostermeier, 2001).

L_p-Adaptation consists of the following four steps: **Initialization**, **Sampling**, **Evaluation**, and **Adaptation**, which are repeated in iterations until a stopping criterion is fulfilled (e.g., a maximal number of evaluations of the specifications is reached). The L_p-Adaptation algorithm requires two inputs: (1) a feasible starting point, and (2) a membership oracle that checks whether any given point is feasible or not by evaluating the specifications. All other algorithm parameters have default values as given in Algorithm 8 and do not necessarily need to be set by the user. Below, we discuss the individual steps of the algorithm in detail.



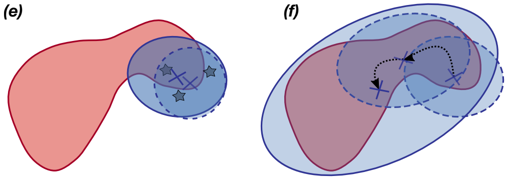
start with feasible point

sample new points ★, evaluate
→ all points infeasible



adapt step size

sample new points ★, evaluate
→ some point feasible



adapt step size, transformation matrix and mean

sample new points, evaluate, adapt

Figure 3.4: Schematic illustration of the algorithmic procedure of adapting the proposal distribution (blue ellipse) to the feasible region (red). The cross represents the mean of the proposal distribution. (a) The algorithm requires a feasible point so start with. This feasible point is set as the mean (center) of the initial proposal distribution. The initial proposal ball is isotropic with a radius of 1. (b,d) New points are drawn from the proposal and evaluated against the specifications by querying the membership oracle. (c) If all points are infeasible, the ball radius is reduced in order to increase the probability of sampling a feasible point next. The shape of the proposal remains unchanged. (e) If at least one point is feasible, the location, shape, and radius of the proposal are adapted by moving the mean in the direction of the center of all feasible points, increasing the radius, and adapting the affine transformation to include information about the distribution of feasible points. (f) At the end of the process, the proposal will have the largest possible volume for the given target hitting probability. Now, the mean can be used as a design center, and the volume can be approximated from the determinant of the proposal.

Algorithm 7: Uniformly sampling from a unit L_p-ball, see Section 4.1 in Ref. (Calafiore et al., 1998)

```
Input  : dimension $n$, $p > 0$
Output : real random vector $\mathbf{y}$ uniformly distributed in $L_p^n(1)$
    1. Sample $n$ real scalars i.i.d. from the generalized Gamma distribution $\xi_i \sim \tilde{G}(\frac{1}{p}, p)$.
    2. Construct a vector $\mathbf{x} \in \mathbb{R}^n$ with components $x_i = s_i \xi_i$, where $s_i$ are independent random signs.
    3. Compute $z = w^{1/n}$, where $w$ is a random variable uniformly distributed in the interval $[0, 1]$.
    4. Return $\mathbf{y} = z \frac{\mathbf{x}}{\|\mathbf{x}\|_p}$, where $\|\mathbf{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$.
```

3.2.1 Initialization

The goal of initialization is to determine the initial shape of the proposal distribution. Therefore, a user-provided initial feasible point is used as the mean of the initial proposal distribution. Since the proposal is an L_P-ball, it is completely determined by:

1. the norm $p > 0$,
2. the ball radius $r \in \mathbb{R}^+$, and
3. an affine transformation matrix $\mathbf{C} = r^2(\mathbf{Q})(\mathbf{Q})^T$, where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\det \mathbf{Q} = 1$.

The norm p is given by the user or set to its default value $p = 2$ and never changes throughout the algorithm. $\mathbf{Q}$ and r are dynamically adapted, usually starting from the initial values $\mathbf{Q} = \mathbf{I}$ and $r = 1$. Prior knowledge about the shape of the feasible region can be incorporated in $\mathbf{C}$.

The sample size per iteration $\lambda \in \mathbb{N}^+$ and the target hitting probability $P \in [0, 1]$ are initialized automatically to $P = \frac{1}{\mathrm{e}}$ (information-theoretic

Algorithm 8: L_p-Adaptation for approximate design centering and volume estimation with L_p-balls

```
Input      : Initial feasible point $\mathbf{m}^{(0)} \in \mathbb{R}^n$
             membership oracle $f : \mathbb{R}^n \to \{0,1\}$ (check specifications)
Output     : Design center $\mathbf{m}^{(K)}$, $\mathbf{Q}^{(K)}$, radius $r^{(K)}$, empirical hitting
             probability $P_{\mathrm{emp}}^{(K)}$
Initialize : transformation matrix $\mathbf{C}^{(0)} \in \mathbb{R}^{n \times n}$, default: $\mathbf{I}$
             rank-one update vector $\mathbf{p_c} \in \mathbb{R}^n, \mathbf{p_c} = \mathbf{0}$
             p-norm $p > 0$, default: $p = 2$
             population size $\lambda \in \mathbb{N}^+$, default: $4 + \lfloor 3\log(n) \rfloor$
             weighting factor $c_m \in [0,1]$, default: $1/(en)$
             learning constant for rank-one update $c_p$, default: $\frac{1}{\sqrt{n}}$
             window size for $P_{\mathrm{emp}}$ $w \in \mathbb{N}^+$, such that $\frac{w}{\lambda} \in \mathbb{N}^+$,
             default: $\lfloor \frac{300}{\lambda} \rfloor \lambda$
             vector of target hitting probabilities $\mathbf{P}$, where
             $P_{l,l=1 \cdot m} \in [0,1]$, default: $0.35$
             # of iterations $\mathbf{K}$, where $K_{l,l=1,\cdots,m} \in \mathbb{N}^+$, $K_0 = 0$, such
             that $K_l - K_{l-1}$ iterations with $P_l$, default: $\lfloor \frac{1000}{\lambda} \rfloor n$

BDDB ← $\mathbf{C}^{(0)}$      /* Eigen-decomposition, B orthogonal, D diagonal
                                 with elements sorted ascendingly */
Q ← BD
$\mathbf{Q}^{(0)} \leftarrow \frac{1}{(\det \mathbf{Q})^{1/n}} \mathbf{Q}$      // normalize Q, such that $\det \mathbf{Q} = 1$
$r^{(0)} \leftarrow \sqrt[2n]{\det \mathbf{C}^{(0)}}$      // $\mathbf{C} = r^2 \mathbf{Q}\mathbf{Q}^T$, all volume information in $r$
for $l \leftarrow 1$ to $m$ do
    set learning rate $\beta \in [0,1]$, default: $\frac{0.6}{(n+1.3)^2 + P_l \lambda}$
    $f_e \leftarrow 1 + \beta(1 - P_l)$      // expansion factor
    $f_c \leftarrow 1 - \beta P_l$      // contraction factor
    for $g \leftarrow K_{l-1} + 1$ to $K_l$ do
        /* sample $\lambda$ points, uniformly from $L_p$-ball with radius $r^{(g)}$,
           centered at $\mathbf{m}^{(g)}$, deformed by $\mathbf{C}^{(g)}$ and evaluate them      */
        $[\mathbf{X}, \mathbf{b}, \mu] \leftarrow$ SampleEval($\lambda$, $p$, $n$, $\mathbf{m}^{(g-1)}$, $r^{(g-1)}$, $\mathbf{Q}^{(g-1)}$, $f$)
        $[r^{(g)}, \mathbf{m}^{(g)}, \mathbf{C}^{(g)}, \mathbf{Q}^{(g)}, \mathbf{p_c}^{(g)}] \leftarrow$ Adaptation($\lambda$, $\mu$, $f_e$, $f_c$, $c_m$,
            $c_p$, $\mathbf{m}^{(g-1)}$, $r^{(g-1)}$, $\mathbf{Q}^{(g-1)}$, $\mathbf{C}^{(g-1)}$, $\mathbf{p_c}^{(g-1)}$, $\mathbf{X}$, $\mathbf{b}$)
        $P_{\mathrm{emp}}^{(g)} = \frac{\#\text{ of feasibe points in } \min((g-K_{l-1})\lambda, w) \text{ previous evaluations}}{(\min((g-K_{l-1})\lambda, w))}$
```

Algorithm 9: Sample λ points from proposal and evaluate them

```
   $[\mathbf{X}, \mathbf{b}, \mu] \leftarrow$ SampleEval($\lambda$, $p$, $n$, $\mathbf{m}$, $r$, $\mathbf{Q}$, $f$)
1  for $j \leftarrow 1$ to $\lambda$ do
2      Sample $\mathbf{y} \sim L_p^n(1)$  // sample uniformly from $L_p$-ball with radius 1, see Algorithm 7
3      $\mathbf{X}_{:,j} \leftarrow \mathbf{m} + r(\mathbf{Qy})$  // affine transform of samples
4      Evaluate $b_j = f(\mathbf{X}_{:,j})$  // if $\mathbf{X}_{:,j}$ feasible, $\mathbf{b}_j = 1$, else $\mathbf{b}_j = 0$
5  count number of feasible points $\mu$
```

optimum for Gaussian proposals (Kjellström and Taxen, 1981)) and $\lambda = 4 + \lfloor 3 * \log(n) \rfloor$ (default for CMA-ES (Hansen and Ostermeier, 1996)). If the algorithm should adaptively change the target hitting probability as described in Section 3.1.2, the user can provide a vector **P** of target hitting probabilities. **P** should then be increasing for a design centering task and decreasing for a volume approximation task, see Section 3.2.5.

3.2.2 SAMPLING

Sampling generates candidate solutions by drawing λ random numbers uniformly distributed in the proposal L_p-ball. This is efficiently done by first sampling points in the unit L_p-ball in n dimensions, as detailed in Algorithm 7, and then transforming the samples using the affine map $\mathbf{Q}$ and scaling them to radius r. This is detailed in Algorithm 9, Line 3, and it samples λ points from the current, adapted proposal distribution.

3.2.3 EVALUATION

For each of the λ points, the algorithm needs to evaluate whether it is feasible or not. For this binary decision, the provided membership oracle is queried for all points, internally checking them against the specifications, see Algorithm 9, Line 4. The number of feasible points is called μ.

Algorithm 10: Adapt radius r, mean $\mathbf{m}$, matrices $\mathbf{C}$ and $\mathbf{Q}$, and $\mathbf{p_c}$

```
$[r^{(g)}, \mathbf{m}^{(g)}, \mathbf{C}^{(g)}, \mathbf{Q}^{(g)}, \mathbf{p_c}^{(g)}] \leftarrow$ Adaptation($\lambda$, $\mu$, $f_e$, $f_c$, $c_m$, $c_p$,
$\mathbf{m}^{(g-1)}$, $r^{(g-1)}$, $\mathbf{Q}^{(g-1)}$, $\mathbf{C}^{(g-1)}$, $\mathbf{p_c}^{(g-1)}$, $\mathbf{X}$, $\mathbf{b}$)

Adapt $r^{(g)} \leftarrow f_e^{\mu} \cdot f_c^{\lambda-\mu} r^{(g-1)}$    // adapt ball radius
if $\mu > 0$ then
    Adapt $\mathbf{m}^{(g)} \leftarrow (1-c_m)\mathbf{m}^{(g-1)} + c_m \cdot \frac{1}{\mu}(\mathbf{X} \cdot \mathbf{b})$    // adapt mean
    set scalars $\alpha_j \geq 0$, for $j = 0, \ldots, \mu$    // $\alpha_j$ normalize input entries for matrix C
    $\mathbf{p_c} \leftarrow (1-c_p)\mathbf{p_c} + \sqrt{c_p(2-c_p)}\alpha_0(\mathbf{m}^{(g)} - \mathbf{m}^{(g-1)})$    // rank-one update
    $\mathbf{C}_\mu \leftarrow \sum_{j=1}^{\lambda} \mathbf{b}_j \frac{1}{\mu} \alpha_j^2 (\mathbf{X}_{:,j}^{(g)} - \mathbf{m}^{(g-1)})(\mathbf{X}_{:,j}^{(g)} - \mathbf{m}^{(g-1)})^T$    // use all feasible points for rank-$\mu$ update
    compute learning rates $c_1 \in [0,1]$ for rank-1 update, and $c_\mu \in [0,1]$ for rank-$\mu$ update, see equations (3.15) and (3.16)
    Adapt $\mathbf{C}^{(g)} \leftarrow (1-c_1-c_\mu)\mathbf{C}^{(g-1)} + c_1 \mathbf{p_c} \mathbf{p_c}^T + c_\mu \mathbf{C}_\mu$    // adapt affine transformation matrix
    $\mathbf{BDDB} \leftarrow \mathbf{C}^{(g)}$    // Eigen-decomposition
    $\mathbf{Q} \leftarrow \mathbf{BD}$
    $\mathbf{Q}^{(g)} \leftarrow \frac{1}{(\det \mathbf{Q})^{1/n}} \mathbf{Q}$    // normalize $\mathbf{Q}$, such that $\det \mathbf{Q} = 1$
else
    $\mathbf{m}^{(g)} = \mathbf{m}^{(g-1)}$
    $\mathbf{p_c}^{(g)} = (1-c_p)\mathbf{p_c}^{(g-1)}$
    $\mathbf{Q}^{(g)} = \mathbf{Q}^{(g-1)}$
    $\mathbf{C}^{(g)} = \mathbf{C}^{(g-1)}$
```

3.2.4 Adaptation of the proposal

Using the information gained during Evaluation, the algorithm adapts the proposal toward finding the largest proposal with the given hitting probability. In order to work toward larger sizes, $\mathbf{Q}$ and r are adapted in each iteration such that the hitting probability remains roughly constant. Radius, mean, and covariance of the affine map are adapted as detailed below.

3.2.4.1 radius

The proposal radius r is adapted as shown in Algorithm 10, Line 1, where $f_e > 1$ is an expansion factor and $f_c < 1$ is a contraction factor (see Section 3.2). The factors f_e and f_c are chosen such that the hitting probability remains constant under stationary conditions, which means that the volume of the proposal distribution

$$\det(\mathbf{C}) = r^{2n} \det(\mathbf{Q}\mathbf{Q}^T), \tag{3.8}$$

does not change (Kjellström and Taxen, 1981). For each feasible point, the radius is increased by a factor f_e, and for each infeasible point it is decreased by a factor f_c. Overall, this changes the proposal volume by factors of f_e^{2n} and f_c^{2n}, respectively. Assume there are S feasible and F infeasible points at stationarity. Since the total volume must not change, this leads to the condition

$$\prod_{i=1}^{S}(f_e)^{2n} \prod_{i=1}^{F}(f_c)^{2n} = 1. \tag{3.9}$$

Since the effective, empirical hitting probability, P_{emp}, at stationarity is equal to the target hitting probability, i.e., the algorithm has converged, and assuming f_e and f_c to be near 1, we get:

$$f_e = 1 + \beta(1 - P) \tag{3.10}$$

$$f_c = 1 - \beta P, \tag{3.11}$$

where P is the target hitting probability, and the learning rate $\beta \in [0,1]$ regulates the speed of adaptation. A β that is too large leads to oscillations in the radius; a β that is too small leads to slow adaptation. We use the default value

$$\beta = \frac{0.6}{(n+1.3)^2 + P\lambda}. \tag{3.12}$$

3.2.4.2 MEAN

If at least one feasible point was sampled, i.e., $\mu > 0$, the mean of the proposal is adapted according to:

$$\mathbf{m} \leftarrow (1-c_m)\mathbf{m} + c_m \frac{1}{\mu}\mathbf{X}\mathbf{b}\,, \tag{3.13}$$

where $\mathbf{X}_{:,j}$ is the j-th sampled point, and b_j is a binary variable indicating the feasibility of $\mathbf{X}_{:,j}$ (1: feasible, 0: infeasible). Therefore $\frac{1}{\mu}\mathbf{X}\mathbf{b}$ is the center of mass of the μ feasible points found, see Algorithm 10 Line 3.

The parameter $0 \leq c_m \leq 1$ defines how far the mean is moved in the direction of the center of mass of the new feasible points. If $c_m = 0$, the mean is not adapted at all, if $c_m = 1$, the mean immediately jumps to the center of mass of the μ new feasible points. We propose to use $c_m = 1/(\mathrm{e}n)$ (Müller and Sbalzarini, 2010a). It is also possible to make c_m depend on the number of feasible points per iteration, as $c_m = \min(\frac{\mu}{\mathrm{e}n}, 0.5)$, such that the mean moves faster if more feasible points are found.

3.2.4.3 COVARIANCE

If at least one feasible point was sampled, the covariance of the affine mapping of the proposal is adapted according to:

$$\mathbf{C} \leftarrow (1-c_1-c_\mu)\mathbf{C} + c_1 \mathbf{p_c}\mathbf{p_c}^T + c_\mu \mathbf{C}_\mu\,, \tag{3.14}$$

where the second term is a rank-one update proportional to the difference between the new and the old mean. The third term is a rank-μ update, adding μ linearly independent directions into the matrix, i.e., the information in the μ directions learned from the feasible points. The details of these updates are as previously described (see Section 2.4 and Hansen (2008)). The learning rates $c_1 \in [0,1]$ and $c_\mu \in [0,1]$ determine the weights of the rank-one update and the rank-μ update, respectively. Their sum must be smaller than 1. As suggested previously (Hansen, 2008), we set

$$c_1 = \alpha_c \frac{0.2}{(n+1.3)^2 + \mu}, \tag{3.15}$$

and

$$c_\mu = 0.2\alpha_c \frac{\mu - 2 + \frac{1}{\mu}}{(n+2)^2 + \alpha_\mu \mu}, \tag{3.16}$$

where $\alpha_c = 3$ and $\alpha_\mu = 0.2$ were found in a parameter study, see Section 3.2.4.4. Both c_1 and c_μ depend on μ, the number of feasible points in one iteration, such that larger μ give more importance to the rank-μ update.

3.2.4.4 PARAMETER STUDY FOR α_c AND α_μ

To decide for the parameters α_c and α_μ, we perform a grid search over $\alpha_c = [1,2,3,4,5,6,7,8,9,10] \times \alpha_\mu = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]$. For each point of this 2D grid we check how well the volume of different feasible regions is approximated over five separate runs with an L_1-ball and an L_2-ball as proposal distribution. The feasible regions are L_p-balls with p-norm $0.5, 1, 2$, and ∞, each isotropic and stretched along $(n-1)$ axes such that the longest axis is ten times longer than the shortest one, and the lengths of the axes are logarithmically spaced. Every test case is checked in dimensions $2, 5, 10, 20$, and 50. This yields 80 test cases for each combination of a_c and a_μ, where each test case is repeated five times. In Figure 3.5, the results of this parameter study are visualized. The color code depicts how well the volume was approximated, the smaller the error value, the better. The error is quantified by the relative difference

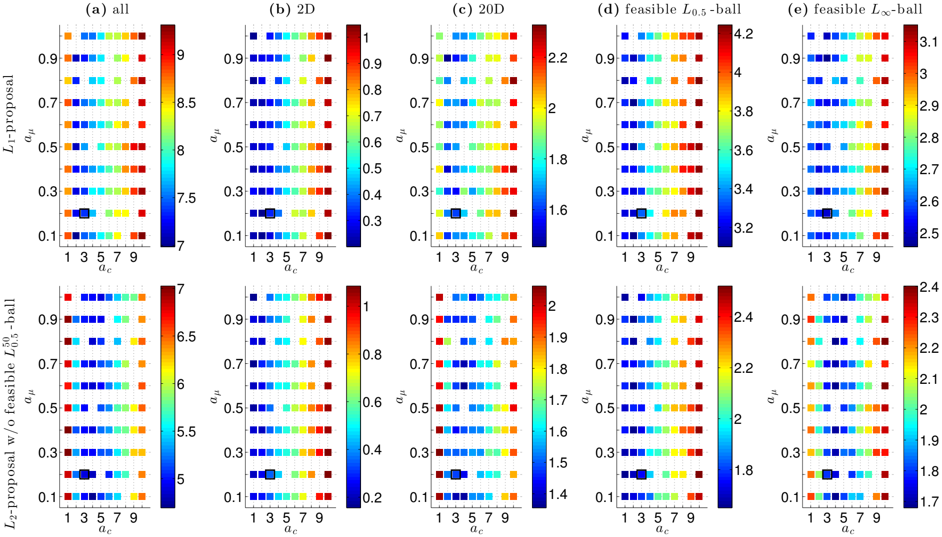


Figure 3.5: Grid search for a_c and a_μ: for each combination of a_c and a_μ, we tested how well L_p-Adaptation approximates the volume with an L_1-proposal (top row) and an L_2-proposal (bottom row). The chosen values $a_c = 3$ and $a_\mu = 0.2$ (black square) ensure low errors for all tested cases. The color code indicates the sum of the relative differences in volume approximation (the lower the value, the better): (a) Sum of all test cases. (b) Sum over all test cases in two dimensions. (c) Sum over all test cases in 20 dimensions. (d) Sum over all test cases where the feasible region is an $L_{0.5}$-ball. (e) Sum of all test cases where the feasible region is an L_∞-ball.

n	target hitting probabilites	number of iterations
2	$(0.35, 0.15)$	$K_{\max}\left(\frac{1}{2}, \frac{1}{2}\right)$
5	$(0.35, 0.15, 0.06)$	$K_{\max}\left(\frac{2}{5}, \frac{3}{10}, \frac{3}{10}\right)$
10	$(0.35, 0.15, 0.06, 0.03)$	$K_{\max}\left(\frac{1}{3}, \frac{2}{9}, \frac{2}{9}, \frac{2}{9}\right)$
20	$(0.35, 0.15, 0.06, 0.03, 0.01)$	$K_{\max}\left(\frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{2}{8}, \frac{2}{8}\right)$
50	$(0.35, 0.15, 0.06, 0.03, 0.01, 0.005, 0.002)$	$K_{\max}\left(\frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}\right)$

Table 3.1: Used target hitting probabilities and their corresponding number of iterations for the test cases in different dimensions n that were used for a parameter study to find a_c and a_μ. $K_{max} = n \cdot 10^4$.

in volume approximation:

$$d_r = \frac{|\frac{1}{5}\sum_{k=1}^{5} \widetilde{\mathrm{vol}}_k(A) - \mathrm{vol}(A)|}{\mathrm{vol}(A)}, \tag{3.17}$$

where $\widetilde{\mathrm{vol}}_k(A)$ is the approximated volume of run k and $\mathrm{vol}(A)$ is the exact volume which is known for these simple test bodies. Every run has a maximal number of iterations of $K_{\max} = n \cdot 10^4$, and a schedule of changing hitting probabilities $\mathbf{P}$ and corresponding numbers of iterations for each hitting probability as summarized in Table 3.1.

The plots in Figure 3.5 show the sums of the errors for different cases: the top row shows the results when an L_1-ball is used as the proposal distribution, the bottom row when an L_2-ball is used. Subplots 3.5a show the sum over all test cases, Subplots 3.5b the sum over all cases in two dimensions, and Subplots 3.5c the sum over all cases in 20 dimensions. In Subplots 3.5d, all cases where the feasible region is an $L_{0.5}$-ball are summed and in Subplots 3.5e all cases where the feasible regions is an L_∞-ball. We exclude the test cases of all $L_{0.5}^{50}$- balls when running L_p-Adaptation with an L_2-proposal, because the volume could not be approximated with the used hitting probabilities. If we would include this high error into the sum, there would be no difference anymore for the different a_c and a_μ. The choice of a_c influences the error much more than the choice of a_μ. The chosen values $a_c = 3$ and $a_\mu = 0.2$ ensure low errors for all tested cases.

3.2.5 Adapting the hitting probability

Introducing a schedule for changing the hitting probability makes it possible to find a better design center or to obtain a better volume approximation. The sequence of target hitting probabilities can be given as an input to the algorithm.

It should start from a value around $\frac{1}{\mathrm{e}}$ in order to first learn the location and rough shape of the feasible region. This information is then used to perform "warm starts" with the subsequent target hitting probabilities by changing the target hitting probability according to a **fixed, predefined schedule**. In a warm start we do not re-initialize but keep all information we learned about the feasible region (e.g. r, $\mathbf{m}$ and $\mathbf{C}$). Alternatively, a **variable schedule** can be used when *lowering* the hitting probability for volume estimation, see Algorithm 11.

For the fixed schedule, the user defines in advance after how many function evaluations the hitting probability is changed, and to what values. For the variable schedule, the hitting probability is decreased to predefined values whenever the process is converging. This is decided by looking at the relative changes of the radius r, the empirical hitting probability P_{emp}, and the volume of the axes-aligned bounding box V_{BB} and the Loewner ellipsoid V_L of all feasible points, see Algorithm 12. The Loewner ellipsoid (Gruber, 2011) of a set of points is the unique minimal-volume ellipsoid that contains this set of points. If r, P, V_{BB}, and V_L are not changing anymore, we assume that the proposal distribution "has seen" everything of the feasible region that is possible to see with the current hitting probability. Then, the target hitting probability is lowered. If reducing the hitting probability does not lead to a larger estimated volume, we assume that the proposal has covered the entire feasible region and stop the algorithm.

We now explain the variable schedule in more detail on Figures 3.6 and 3.7. They show example trajectories used for the decision when to lower the hitting probability. In the examples, the feasible regions are a stretched L_2^5-ball (Figure 3.6) and a stretched $L_{0.5}^5$-ball (Figure 3.7), where $(n-1)$ axes are stretched such that the longest one is ten times longer than the shortest

Algorithm 11: L_p-adaptation for volume estimation with L_p-balls **(variable schedule)**

```
Input      : Initial feasible point $\mathbf{m}^{(0)} \in \mathbb{R}^n$
             membership oracle $f : \mathbb{R}^n \to \{0,1\}$ (check specifications)
Output     : Design center $\mathbf{m}^{(K)}$, $\mathbf{Q}^{(K)}$, radius $r^{(K)}$, empirical hitting
             probability $P_{\text{emp}}^{(K)}$
Initialize : $\mathbf{C}^{(0)} \in \mathbb{R}^{n \times n}$, $\mathbf{p_c} \in \mathbb{R}^n$, $p > 0$, $\lambda \in \mathbb{N}^+$, $c_m, c_p \in [0,1]$,
             $w \in \mathbb{N}^+$, vector of $m$ decreasing target hitting
             probabilities $\mathbf{P}$, # of iterations $K_{\max}$, $K_0 = 0$
             for default values see Algorithm 8
BDDB ← $\mathbf{C}^{(0)}$   /* Eigen-decomposition, B orthogonal, D diagonal with elements sorted ascendingly */
$\mathbf{Q} \leftarrow \mathbf{BD}$
$\mathbf{Q}^{(0)} \leftarrow \frac{1}{(\det \mathbf{Q})^{1/n}} \mathbf{Q}$   // normalize Q, such that $\det \mathbf{Q} = 1$
$r^{(0)} \leftarrow \sqrt[2n]{\det \mathbf{C}^{(0)}}$   // $\mathbf{C} = r^2 \mathbf{Q}\mathbf{Q}^T$, all volume information in $r$
$l = 0$, $g = 0$   // count changes of $P$ and iteration count
repeat
    l = l+1
    set learning rate $\beta \in [0,1]$, default: $\frac{0.6}{(n+1.3)^2 + P_l \lambda}$
    $f_e \leftarrow 1 + \beta(1 - P_l)$   // expansion factor
    $f_c \leftarrow 1 - \beta P_l$   // contraction factor
    repeat
        g = g+1
        $[\mathbf{X}, \mathbf{b}, \mu] \leftarrow$ SampleEval($\lambda$, $p$, $n$, $\mathbf{m}^{(g-1)}$, $r^{(g-1)}$, $\mathbf{Q}^{(g-1)}$, $f$)
        $[r^{(g)}, \mathbf{m}^{(g)}, \mathbf{C}^{(g)}, \mathbf{Q}^{(g)}, \mathbf{p_c}^{(g)}] \leftarrow$ Adaptation($\lambda$, $\mu$, $f_e$, $f_c$, $c_m$, $c_p$, $\mathbf{m}^{(g-1)}$, $r^{(g-1)}$, $\mathbf{Q}^{(g-1)}$, $\mathbf{C}^{(g-1)}$, $\mathbf{p_c}^{(g-1)}$, $\mathbf{X}$, $\mathbf{b}$)
        $P_{\text{emp}}^{(g)} = \frac{\text{\# of feasibe points in } \min((g-K_{l-1})\lambda, w) \text{ previous evaluations}}{(\min((g-K_{l-1})\lambda, w))}$
        $[z_1, V_l] \leftarrow$ CheckConvergence1(all feasible points found so far, trajectory of radius obtained with $P_l$, trajectory of $P_{\text{emp}}$ obtained with $P_l$)
    until $z_1 == 1$ or $g = K_{max}$
    if $l > 2$ then
        $z_2 \leftarrow$ CheckConvergence2($V_{l-1}, V_l$)
    $K_l = g$
until $z_2 == 1$ or $l == m$ or $g == K_{max}$
```

Algorithm 12: Check if process has converged

```
$[z_1, V_l] \leftarrow$ CheckConvergence1(all feasible points found so far, trajectory of radius $r$ obtained with $P_l$, trajectory of $P_{\text{emp}}$ obtained with $P_l$, $p$, $n$)
Initialize: error thresholds $\epsilon_r$, $\epsilon_P$, $\epsilon_L$, $\epsilon_B$
$z_1 = 0$, $V_l = []$
if trajectories long enough then
    calculate $\delta_r$, the relative change of $r$
    calculate $\delta_P$, the relative change of $P_{\text{emp}}$
    if $\delta_r < \epsilon_r$ and $\delta_P < \epsilon_P$ then
        calculate $\delta_L$, the relative change of volumes of Loewner ellipsoids from accepted samples
        calculate $\delta_B$, the relative change of volumes of axes-aligned bounding boxes from accepted samples
        if $\delta_L < \epsilon_L$ and $\delta_B < \epsilon_B$ then
            $z_1 = 1$
            $V_l = P_{\text{emp}}^{(K)} \cdot \text{vol}(L_p^n(r))$
```

Algorithm 13: Check if volume approximation changed

```
$z_2 \leftarrow$ CheckConvergence2($V_{l-1}, V_l$)
Initialize: error threshold $\epsilon_V$
calculate $\delta_V$, the relative change of $V_{l-1}$ and $V_l$
if $\delta_V < \epsilon_V$ then
    $z_2 = 1$
else
    $z_2 = 0$
```

one and scaled such that the determinant of the transformation matrix $\det(\mathbf{C}) = 1$. Both figures show one example run with an L_2-proposal.

As described in Algorithm 12, the relative change of step size r and the relative change of the empirical hitting probability P_{emp} are monitored. This is done by averaging the trajectories over two equisized, subsequent intervals (red and orange rectangles in Figures 3.6 and 3.7) and comparing them. Let's call the averages of two equisized, subsequent intervals of a trajectory m_1 and m_2. If

$$\delta = \frac{|m_1 - m_2|}{\frac{1}{2}(m_1 + m_2)} < \epsilon \tag{3.18}$$

for the hitting probability and the radius, then the same is checked for the upper bounds of the volume: the volume of the Loewner ellipsoid (blue) and the volume of the axes-aligned bounding box (green). The size of the averaging intervals increases with lower hitting probability because with a smaller hitting probability more samples are needed to get the same information (obtain a similar number of new feasible points). We choose the size of the intervals to be

$$s = \min\left(\frac{30n}{P}, 2000\right). \tag{3.19}$$

We cap the size to 2000 to avoid very large intervals. The thresholds ϵ chosen for the two examples are $\epsilon_r = \epsilon_P = \epsilon_L = \epsilon_B = 0.005$. The smaller the ϵ, the more equal the averages m_1 and m_2 and the more likely that the run of L_p-Adaptation converged. Monitoring the empirical hitting probability and the radius are more important than monitoring the outer approximations. The outer approximations are more of an indication that there is no huge part missing from the feasible region that could have been "seen" with the current proposal. Before lowering the target hitting probability P, the volume of the feasible region is estimated by

$$\widetilde{\text{vol}}(A) = P_{\text{emp}} \cdot \text{vol}(L_p^n(\bar{r})), \tag{3.20}$$

where $\bar{r}$ is the averaged step size over an interval with size

$$s = \frac{1}{2}(K_l - K_{l-1}). \tag{3.21}$$

P	$\widetilde{\text{vol}}(A)$	relative error
0.35	$1.637 \cdot 10^{-4}$	0.4198
0.25	$2.080 \cdot 10^{-4}$	0.2629
0.15	$2.305 \cdot 10^{-4}$	0.1832
0.1	$2.580 \cdot 10^{-4}$	0.0859
0.05	$2.743 \cdot 10^{-4}$	0.0278
0.03	$3.133 \cdot 10^{-4}$	0.1103
0.01	$3.117 \cdot 10^{-4}$	0.1047

Table 3.2: Volume approximation of an $L_{0.5}^5$-ball with an L_2^5-proposal, and its relative error, during one run of L_p-Adaptation with a variable schedule of decreasing target hitting probabilities P, see also Figure 3.7. The true volume of the feasible region is $2.8219 \cdot 10^{-4}$.

$K_l - K_{l-1}$ is the number of iterations done with the current target hitting probability P_l. Starting with the second volume estimate of the feasible region, the relative change δ_V (see Equation (3.18)) is calculated, see Algorithm 13. The algorithm stops, if δ_V is below the threshold ϵ_V (in the two examples $\epsilon_V = 0.01$). The threshold ϵ_V defines how equal two successive volume approximations have to be before the algorithm stops. If ϵ_V is too high, L_p-Adaptation may stop too early and underestimate the volume, whereas if it is too low, L_p-Adaptation may not stop (not before the maximal number of evaluations is reached) due to the fluctuating nature of the process. When choosing ϵ_V, one should have this trade-off in mind. We find a choice of $\epsilon_V \in [0.005, 0.05]$ to work well in practice.

The relative errors of the volume approximation of the L_2^5-ball are 0.0004, 0.0157, and 0.0225 with a target hitting probability P of 0.35, 0.25, and 0.15, respectively. At the end of the run, the relative error of the volume approximation of the $L_{0.5}^5$-ball is 0.1047 with a target hitting probability P of 0.01. For the volume approximations with the other P and corresponding errors, see Table 3.2.

In our examples, using an L_2^5-proposal to approximate the volume of the $L_{0.5}^5$-ball (Figure 3.7) with the variable schedule needs about six times more oracle evaluations than approximating the volume of the L_2^2-ball (Figure 3.6). In Figure 3.6d the volume approximation of the feasible region does not change with decreasing P. The initial proposal distribution

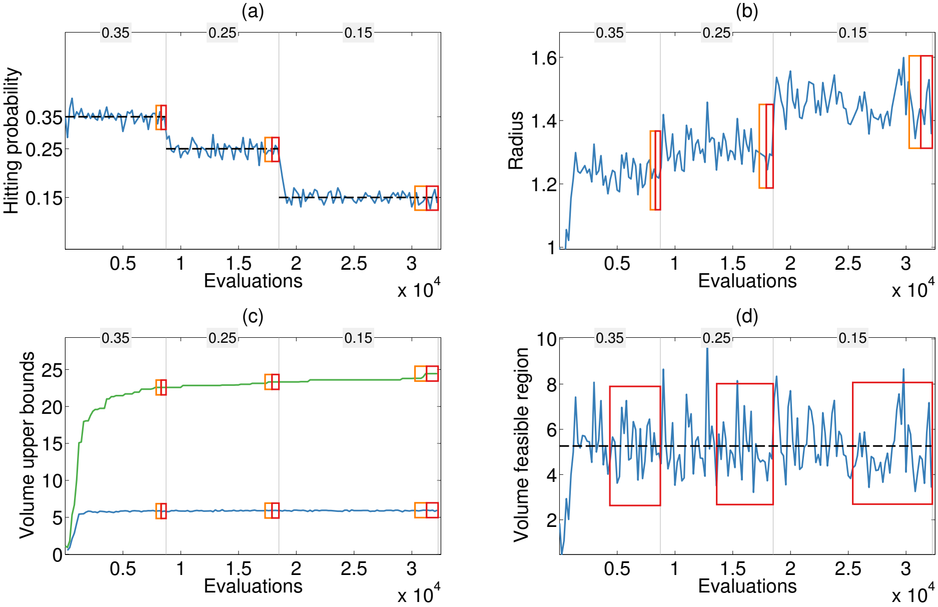


Figure 3.6: Example trajectories relevant for adaptation of the hitting probability. The feasible region is a stretched L_2-ball in five dimensions. In each plot, the different phases with changing target hitting probabilities (as shown at the top of each plot) are separated by gray vertical lines and the rectangles indicate intervals where values are averaged and compared for adaptation. (a) The target hitting probability (black dashed line) and the evolution of the empirical hitting probability (blue). (b) The radius r of the proposal. (c) Volumes of the outer approximations Loewner ellipsoid (blue) and axes-aligned bounding box (green). (d) The approximated volume of the feasible region. (a-c) are used to decide when to lower the hitting probability, see Algorithm 12, (d) is used to decide when to stop L_p-Adaptation if run with a variable schedule, see Algorithm 13.

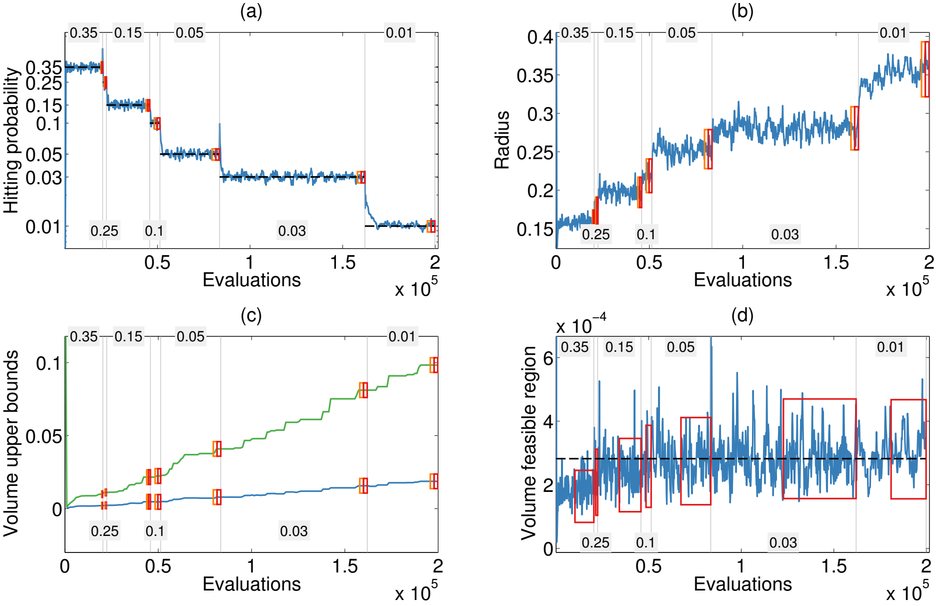


Figure 3.7: Example trajectories relevant for adaptation of the hitting probability. The feasible region is a stretched $L_{0.5}$-ball in five dimensions. In each plot, the different phases with changing target hitting probabilities (as shown at the top and the bottom of each plot, respectively) are separated by gray vertical lines and the rectangles indicate intervals where values are averaged and compared for adaptation. (a) The target hitting probability (black dashed line) and the evolution of the empirical hitting probability (blue). (b) The radius r of the proposal. (c) Volumes of the outer approximations Loewner ellipsoid (blue) and axes-aligned bounding box (green). (d) The approximated volume of the feasible region. (a-c) are used to decide when to lower the hitting probability, see Algorithm 12, (d) is used to decide when to stop L_p-Adaptation if run with a variable schedule, see Algorithm 13.

is isotropic and does not match the stretched L_2^5-ball, therefore the approximated volume increases until the proposal has learned the shape of the feasible region by around 2000 evaluations and then only fluctuates around this value. In comparison, when looking at the volume approximation of the $L_{0.5}^5$-ball (Figure 3.7d), we see that a hitting probability of $P = 0.35$ is not enough for the L_2^2-proposal to cover the $L_{0.5}^5$-ball and that the volume approximation is increasing with decreasing hitting probabilities.

In general, a better volume approximation can be achieved using a proposal that is big enough to cover most of the volume of the feasible region and by averaging over separate runs.

3.3 Averaging of Covariance Matrices

Since the process of adapting the proposal distribution to the location and shape of the underlying feasible region is only stationary in distribution, we want to average over the last iterations to get a value for the location (mean) and the shape (covariance) of the feasible region. Naively averaging the covariance matrices component-wise will not preserve the directions, see Figure 3.8.

How to average over covariance matrices has for example been studied in EEG signal classification (Barachant et al., 2013; Yger et al., 2015). There, an average covariance matrix is found by solving the optimization problem

$$\min_{\mathbf{C}} \sum_{k=1}^{m} d(\mathbf{C}_k, \mathbf{C}), \tag{3.22}$$

where $\mathbf{C}_{k,k=1,\cdots,m}$ are the covariance matrices that should be averaged, $\mathbf{C}$ is the resulting averaged covariance matrix and d is a distance function. Examples for distance functions are the Euclidean distance, the LogEuclidean distance, and the Riemannian metric (Yger et al., 2015).

Since we can assume that L_p-Adaptation has converged to a stationary distribution (only then averaging is meaningful), we can use Eigenvalues and Eigenvectors as a more natural way of averaging covariance matrices. We require that the Eigenspaces of two successive covariance matrices do

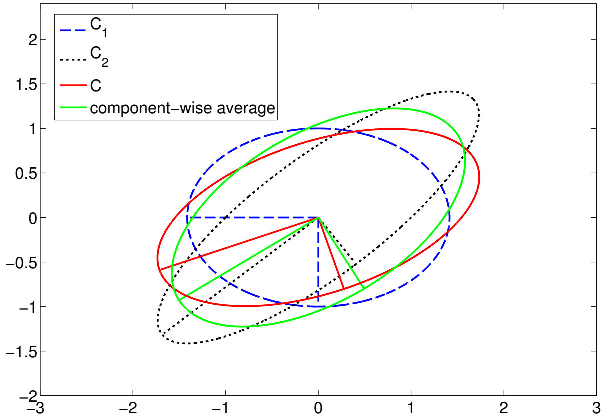


Figure 3.8: The average covariance matrix $\mathbf{C}$ *(red ellipse) of the two covariance matrices* $\mathbf{C}_1$ *(blue ellipse) and* $\mathbf{C}_2$ *(black ellipse). The green ellipse shows the component-wise average of* $\mathbf{C_1}$ *and* $\mathbf{C_2}$*. Lines represent their corresponding Eigenvectors.*

not change much. The variance between two successive covariance matrices is tolerable if the directions of one covariance matrix can be uniquely mapped onto the directions of the other.

Under these assumptions, which are trivially fulfilled for L_p-Adaptation, averaging is then realized by Eigen-decomposition of the m covariance matrices to be averaged, where we identify consecutive Eigenspaces and Eigenvalues before averaging them. The detailed algorithm is described in Algorithm 14 and the MATLAB source code can be found in Appendix A.1. Below, we illustrate the procedure in a simple example.

3.3.1 EXAMPLE

Here we show a simple 2D example of how to average over two covariance matrices $\mathbf{C}_1 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ and $\mathbf{C}_2 = \begin{pmatrix} 3 & 2 \\ 2 & 2 \end{pmatrix}$ according to the presented procedure, we first get the Eigen-decompositions $\mathbf{BDB}^{-1} \leftarrow \mathbf{C}$ of both matrices, where columns of $\mathbf{B}$ are

Algorithm 14: Average covariance matrix

```
Input  : Covariance matrices $\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_m \in \mathbb{R}^{n \times n}$
Output : Average covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$
$\mathbf{B}_{\text{old}}\mathbf{D}_{\text{old}}\mathbf{B}_{\text{old}}^{-1} \leftarrow \mathbf{C}_1$   /* Eigen-decomposition, columns of $\mathbf{B}_{\text{old}}$ are Eigenvectors of $\mathbf{C}_1$, diagonal elements of $\mathbf{D}_{\text{old}}$ are corresponding Eigenvalues. */
$\mathbf{D}_{\Sigma} = \mathbf{D}_{\text{old}}$
$\mathbf{B}_{\Sigma} = \mathbf{B}_{\text{old}}$
for $i \leftarrow 2$ to $m$ do
    set vector $\mathbf{w}$ to $\mathbf{0}$
    $\mathbf{B}\mathbf{D}\mathbf{B}^{-1} \leftarrow \mathbf{C}_i$   /* Eigen-decomposition */
    for $j \leftarrow 1$ to $n$ do
        $\mathbf{q} \leftarrow \mathbf{B}(:, j)$
        $\mathbf{v} \leftarrow \mathbf{q}^T \mathbf{B}_{\text{old}}$
        $\text{idx}_{max} \leftarrow \arg\max_{k=1 \cdots N}(|v_k|)$   /* get index of previous Eigenvector that q is most alike to */
        $\mathbf{w}(j) \leftarrow \text{idx}_{max}$   /* get order of vectors */
        $\mathbf{D}_{\Sigma}(\text{idx}_{max}, \text{idx}_{max}) \leftarrow \mathbf{D}_{\Sigma}(\text{idx}_{max}, \text{idx}_{max}) + \mathbf{D}(j, j)$   /* add Eigenvalues */
        $a \leftarrow \mathbf{q}^T \mathbf{B}_{\text{old}}(:, \text{idx}_{max})$
        /* check that vectors are pointing in same direction */
        if $a < 0$ then
            $\mathbf{p} = (-1) \cdot \mathbf{q}$
        else
            $\mathbf{p} = \mathbf{q}$
        $\mathbf{B}(:, j) \leftarrow \mathbf{p}$
        $\mathbf{B}_{\Sigma}(:, \text{idx}_{max}) = \mathbf{B}_{\Sigma}(:, \text{idx}_{max}) + \mathbf{p}$   /* add Eigenvectors */
    check that mapping of Eigenvectors is bijective
    $\mathbf{B}_{\text{old}} \leftarrow \mathbf{B}(:, \mathbf{w})$
$\mathbf{D} \leftarrow \frac{1}{m}\mathbf{D}_{\Sigma}$
normalize each Eigenvector in $\mathbf{B}_{\Sigma}$
$\mathbf{C} \leftarrow \mathbf{B}_{\Sigma}\mathbf{D}\mathbf{B}_{\Sigma}^{-1}$   /* get averaged covariance */
$\mathbf{C} \leftarrow \frac{1}{2}(\mathbf{C} + \mathbf{C}^T)$   /* symmetrize it */
```

Eigenvectors of **C** and diagonal elements of **D** are the corresponding Eigenvalues:

$$\mathbf{B}_{\text{old}}\mathbf{D}_{\text{old}}\mathbf{B}_{\text{old}}^{-1} \leftarrow \mathbf{C}_1$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \leftarrow \mathbf{C}_1$$

$$\mathbf{B}\mathbf{D}\mathbf{B}^{-1} \leftarrow \mathbf{C}_2$$

$$\begin{pmatrix} 0.6154 & -0.7882 \\ -0.7882 & -0.6154 \end{pmatrix}\begin{pmatrix} 0.4384 & 0 \\ 0 & 4.5616 \end{pmatrix}\begin{pmatrix} 0.6154 & -0.7882 \\ -0.7882 & -0.6154 \end{pmatrix} \leftarrow \mathbf{C}_2$$

We continue by having a look at the first Eigenvector of $\mathbf{C}_2$

$$\mathbf{q} = \mathbf{B}(:,1) = \begin{pmatrix} 0.6154 \\ -0.7882 \end{pmatrix},$$

to see which Eigenvector of C_1 it best maps to.

We know that the smaller the angle between two vectors $\mathbf{q}_1$ and $\mathbf{q}_2$, the larger their absolute scalar product $|\mathbf{q}_1 \cdot \mathbf{q}_2|$ (if they are orthogonal it equals 0, and if they are pointing in the same or opposite direction, it equals 1). We thus calculate

$$\mathbf{v} = \mathbf{q}^T\mathbf{B}_{\text{old}} = \begin{pmatrix} 0.6154 & -0.7882 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} -0.7882 & 0.6154 \end{pmatrix}.$$

Calculating v tells us which direction of $\mathbf{C}_1$ best maps to q and since $|v_1| > |v_2|$, we map $(-1) \cdot \mathbf{q}$ onto the first Eigenvector of $\mathbf{C}_1$ (To make the vectors point into the same direction, we multiplied $\mathbf{q}$ by (-1)). Checking the same for $\mathbf{q} = \mathbf{B}(:,2)$, the second Eigenvector of $\mathbf{C}_2$, it gets multiplied by (-1) and mapped to the second Eigenvector of $\mathbf{C}_1$.

The matrix of the sums of Eigenvectors now equals

$$\mathbf{B}_\Sigma = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \begin{pmatrix} -0.6154 & 0.7882 \\ 0.7882 & 0.6154 \end{pmatrix} = \begin{pmatrix} -0.6154 & 1.7882 \\ 1.7882 & 0.6154 \end{pmatrix}.$$

The matrix with the sums of Eigenvalues in the diagonal equals

$$\mathbf{D}_\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} + \begin{pmatrix} 0.4384 & 0 \\ 0 & 4.5616 \end{pmatrix} = \begin{pmatrix} 1.4384 & 0 \\ 0 & 6.5616 \end{pmatrix}.$$

To get the averaged covariance matrix, we normalize the summed Eigenvector to unit length. The averaged Eigenvectors then equal

$$\mathbf{B}_m = \begin{pmatrix} -0.3254 & 0.9456 \\ 0.9456 & 0.3254 \end{pmatrix},$$

and the averaged Eigenvalues equal

$$\mathbf{D}_m = \frac{1}{2}\mathbf{D}_\Sigma = \begin{pmatrix} 0.7192 & 0 \\ 0 & 3.2808 \end{pmatrix}.$$

This leads to the averaged covariance matrix:

$$\mathbf{C} = \mathbf{B}_m \mathbf{D}_m \mathbf{B}_m^{-1} = \begin{pmatrix} 3.00 & 0.7882 \\ 0.7882 & 0.9905 \end{pmatrix}.$$

The matrices $\mathbf{C}_1$, $\mathbf{C}_2$, and their average $\mathbf{C}$, are illustrated in Figure 3.8. In summary, we average covariance matrices by averaging their Eigenvectors and Eigenvalues and then reconstructing the average matrix from this average Eigen-decompostion. This guarantees that the directions are preserved.

3.4 BEHAVIOR OF L_p-BALLS IN DIFFERENT DIMENSIONS

Since we are using L_p-balls as proposal distributions, we want to comprehend their behavior.

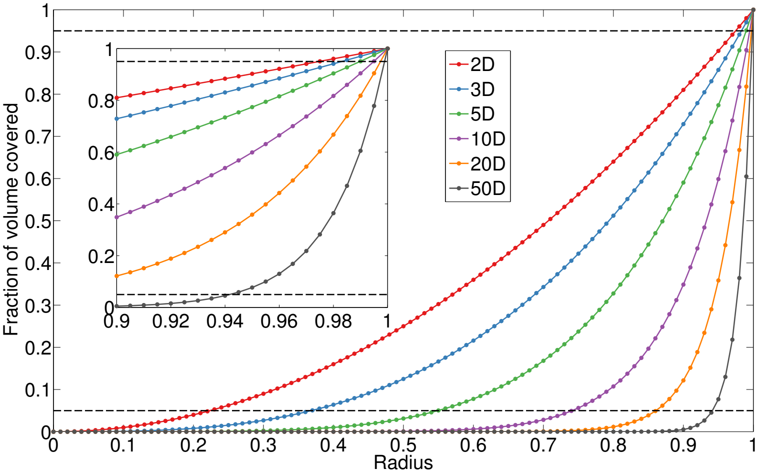


Figure 3.9: The fraction of the volume of the $L_p(1)$*-ball that is covered by itself depending on its radius* r *is shown for dimensions* $n = 2, 3, 5, 10, 20,$ *and* 50.

As dimension increases, an increasing proportion of the volume of an L_p-ball is found near its surface, see Figure 3.9. In this figure, we shown how much of the volume of an L_p-ball is covered by itself depending on its radius. In 50 dimensions, 95% of the volume is in the region where $r \geq \sqrt[n]{0.05} = 0.94$, whereas in two dimensions it is where $r \geq 0.22$.

To understand the behavior of L_p-balls better and to get an insight into how to choose target hitting probabilities, we now look at how much volume of an $L_{p_f}(1)$-ball is covered by L_{p_p}-balls of different radii, where $p_f \neq p_p$. In Figure 3.10 we show how the volume of an $L_{0.5}$-ball (top row) is covered by an L_1-ball, L_2-ball, and an L_∞-ball, and how the volume of an L_1-ball (bottom row) is covered by an $L_{0.5}$-ball, L_2-ball, and an L_∞-ball. In Figure 3.11 we show how the volume of an L_2-ball (top row) is covered by an $L_{0.5}$-ball, L_1-ball, and an L_∞-ball, and how the volume of an L_∞-ball (bottom row) is covered by an $L_{0.5}$-ball, L_1-ball, and an L_2-ball. In each subplot, this is shown for dimensions $n = 2, 3, 5, 10, 20,$ and 50. All curves are sigmoidal and get steeper with increasing dimensions.

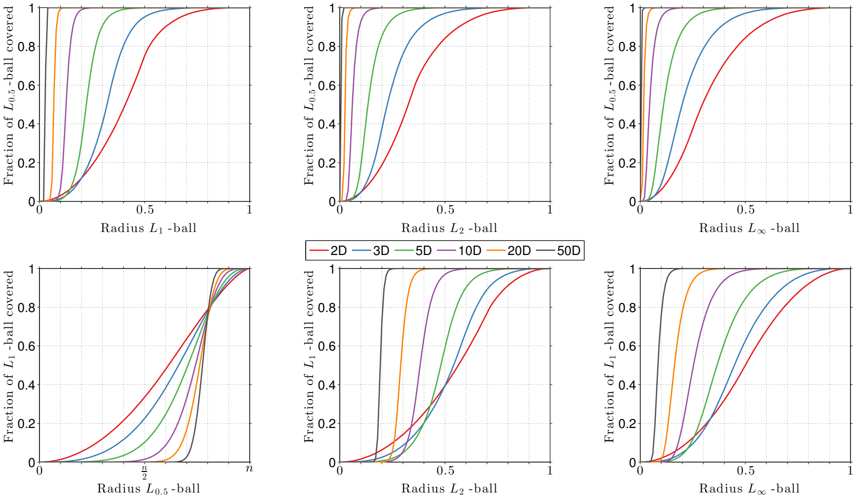


Figure 3.10: The fraction of the volume of the $L_{0.5}(1)$*-ball (top row) and* $L_1(1)$*-ball (bottom row) that is covered by different* $L_{p_p}(r)$*-balls (columns) depending on their radii in dimensions* $n = 2, 3, 5, 10, 20,$ *and* 50*. The radii are plotted until the* $L_{p_p}(r)$*-ball completely encloses the* $L_{p_f}(1)$*-ball. If* $p_f \leq p_p$*, this occurs when* $r = 1$*, if* $p_f = 1$ *and* $p_p = 0.5$*, this occurs when* $r = n$*.*

Most of the volume is then in a shell of the proposal, which gets thinner with increasing dimension.

If p_p is greater than p_f, the proposal L_{p_p}-ball with radius 1 covers the feasible $L_{p_f}(1)$-ball completely. With increasing dimension, the radius of the L_{p_p}-ball that is needed to cover almost all of the L_{p_f}-ball, gets smaller. Equally, with increasing p_p, the radius of the L_{p_p}-ball required to cover the same fraction is smaller.

If $p_p < p_f$, the proposal needs to have a larger radius than the feasible L_{p_f}-ball in order to cover it completely. How large the radius has to be is visualized in Figure 3.12 for 2D. If the radius of the L_∞-ball is set to r_∞, than the Euclidean distance to the corners, which is the radius of the

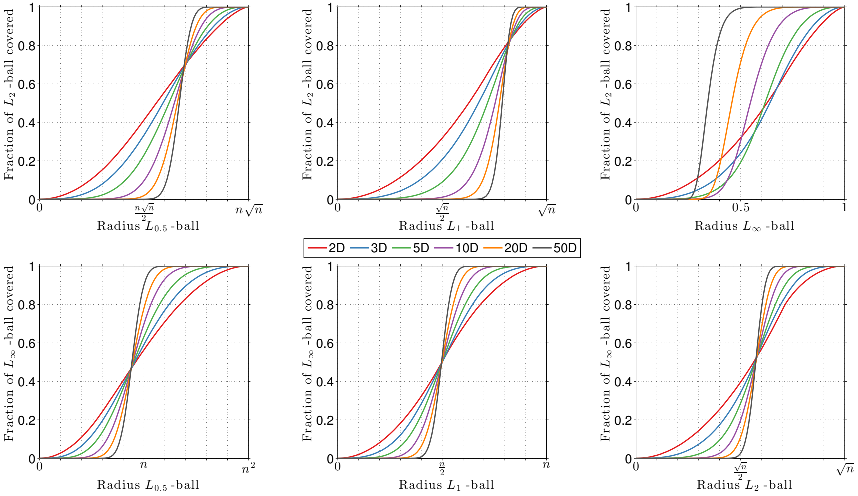


Figure 3.11: The fraction of the volume of the $L_2(1)$*-ball (top row) and* $L_\infty(1)$*-ball (bottom row) that is covered by different* $L_{p_p}(r)$*-balls (columns) depending on their radii in dimensions* $n = 2, 3, 5, 10, 20,$ *and* 50*. The radii are plotted until the* $L_{p_p}(r)$*-ball completely encloses the* $L_{p_f}(1)$*-ball. If* $p_f \leq p_p$*, this occurs when* $r = 1$*. For the cases where* $p_f > p_p$*, see Figure 3.12.*

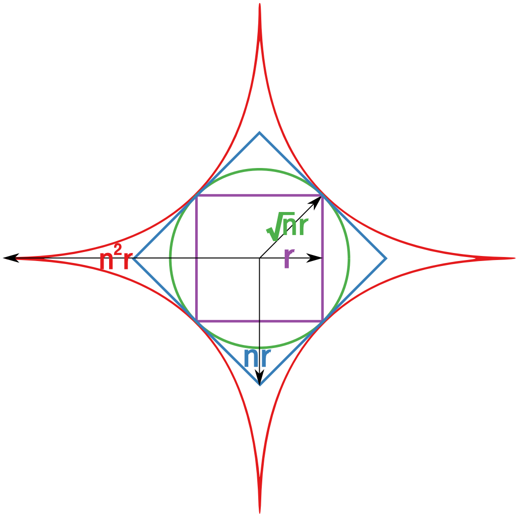


Figure 3.12: The $L_\infty(r)$*-ball (purple) is completely enclosed by the* $L_2(\sqrt{n}r)$*-ball (green), the* $L_1(nr)$*-ball (blue), and the* $L_{0.5}(n^2r)$*-ball (red).*

enclosing L_2-ball is

$$r_2 = \sqrt{\left(\sum_{i=1}^{n} |r_\infty|^2\right)} = \sqrt{n}r_\infty, \tag{3.23}$$

the radius of the enclosing L_1-ball is

$$r_1 = \sum_{i=1}^{n} |r_\infty| = nr_\infty, \tag{3.24}$$

and the radius of the enclosing $L_{0.5}$-ball is

$$r_{0.5} = \left(\sum_{i=1}^{n} \sqrt{|r_\infty|}\right)^2 = n^2 r_\infty. \tag{3.25}$$

If $p_p < p_f$, all lines for the different dimensions and a specific p_p,p_f-combination, intersect at one point, which means that the proposals cover the same fraction of the feasible L_p-balls when they have the same relative radius.

The fraction of an L_{p_p}-ball overlapping with an L_{p_f}-ball is the probability of drawing a point inside the latter when sampling from the former. Assuming the L_{p_f}-ball is completely enclosed by the L_{p_p}-ball, we can calculate this fraction by

$$\Pr\Big(\mathbf{x} \in L^n_{p_f}(r_f) | \mathbf{x} \sim L^n_{p_p}(r_p)\Big) = \frac{\mathrm{vol}(L^n_{p_f}(r_f))}{\mathrm{vol}(L^n_{p_p}(r_p))}. \tag{3.26}$$

For $p_f = 1$, $p_p = 2$, and $n = 50$ this value is:

$$\Pr\big(\mathbf{x} \in L^{50}_1(1) | \mathbf{x} \sim L^{50}_2(1)\big) = \frac{\mathrm{vol}(L^{50}_1(1))}{\mathrm{vol}(L^{50}_2(1))} = 2.1 \cdot 10^{-37}. \tag{3.27}$$

This hitting probability is too small to be used in L_p-Adaptation. But, as we can see in Figure 3.10, the radius of the proposal does not have to equal the feasible region's radius in order to cover the volume. A smaller radius leads to a higher hitting probability. In Table 3.3 we summarize how low the hitting probability needs to be such that the proposal L_p-ball covers 90% of the feasible L_p-ball's volume for dimensions $n = 2, 3, 5, 10, 20$, and 50. The higher the dimension, the lower the hitting probability has to be to cover the same percentage of the feasible region. Covering 90% of an $L^{50}_1(1)$-ball with an L^{50}_2-proposal is possible with a hitting probability of 0.0012.

To allow for better volume approximations, we lower the hitting probability in L_p-Adaptation such that the proposal covers more of the feasible region's volume. In this study we see, that the feasible region does not have to be completely enclosed to cover most of its volume. These theoretical insights can be used as a guideline for the user how to set the final target hitting probability when doing volume approximation with L_p-Adaptation.

	$n = 2$	$n = 3$	$n = 5$	$n = 10$	$n = 20$	$n = 50$
feasible $L_{0.5}(1)$-ball						
L_1-proposal	0.84	0.66	0.41	0.13	0.019	$4.1 \cdot 10^{-5}$
L_2-proposal	0.60	0.33	0.092	0.0031	$3.6 \cdot 10^{-6}$	$7.4 \cdot 10^{-16}$
L_∞-proposal	0.47	0.18	0.019	$2.2 \cdot 10^{-5}$	$3.3 \cdot 10^{-12}$	$6.2 \cdot 10^{-36}$
feasible $L_1(1)$-ball						
$L_{0.5}$-proposal	0.87	0.75	0.58	0.31	0.10	0.0057
L_2-proposal	0.92	0.81	0.59	0.26	0.057	0.0012
L_∞-proposal	0.75	0.48	0.16	0.0052	$1.7 \cdot 10^{-6}$	$1.2 \cdot 10^{-18}$
feasible $L_2(1)$-ball						
$L_{0.5}$-proposal	0.72	0.52	0.28	0.068	0.0049	$3.6 \cdot 10^{-6}$
L_1-proposal	0.92	0.83	0.68	0.42	0.18	0.019
L_∞-proposal	0.91	0.78	0.48	0.10	0.0020	$1.7 \cdot 10^{-9}$
feasible $L_\infty(1)$-ball						
$L_{0.5}$-proposal	0.57	0.32	0.097	0.0051	$1.6 \cdot 10^{-5}$	$1.1 \cdot 10^{-12}$
L_1-proposal	0.74	0.54	0.26	0.041	0.0011	$3.2 \cdot 10^{-8}$
L_2-proposal	0.92	0.78	0.53	0.18	0.022	$5.1 \cdot 10^{-5}$

Table 3.3: Hitting probability that is necessary to cover 90% *of a feasible* L_p*-ball with a proposal of different p-norm.*

3.5 EXTRACTING INFORMATION FROM GENERATED SAMPLES AND FROM THE EVOLVING PROPOSAL DISTRIBUTION

Using L_p-Adaptation, the volume approximation and the estimated design center are not the only information we can obtain about the feasible region. L_p-Adaptation approximates the feasible region with an adaptive proposal distribution. In this section we show, how the collected samples $\mathbf{X}$ and the evolving proposal distribution itself can be used to extract information about correlations and sensitivity of the design parameters, as well as about the shape of the feasible region.

3.5.1 CORRELATIONS AND SENSITIVITY OF PARAMETERS

From the sample covariance matrix of the accepted points, the Pearson correlation coefficient between parameters x_i and x_j can be estimated as:

$$\rho_{x_i,x_j} = \mathrm{Corr}(x_i, x_j) = \frac{\mathrm{Cov}[x_i, x_j]}{\sigma_{x_i}\sigma_{x_j}}, \tag{3.28}$$

where σ_{x_i} and σ_{x_j} are the standard deviations. The correlation coefficient is a dimensionless number between -1 and 1 and measures the linear dependences between x_i and x_j. If there is a perfect linear proportionality, it equals 1, and if there is a perfect linear anti-proportionality, it equals -1.

The Eigen-decomposition of the transformation matrix $\mathbf{C}$

$$\mathbf{BDB} \leftarrow \mathbf{C} \tag{3.29}$$

gives $\mathbf{D}$, a diagonal matrix of Eigenvalues λ_i in increasing order and $\mathbf{B}$, a matrix whose columns are the corresponding Eigenvectors. The Eigenvectors are also called principle directions. The most sensitive direction is the one with the smallest Eigenvalue, where sensitive means that in that direction a perturbation renders a feasible solution infeasible most easily. Such information can be useful for example when designing an electrical circuit where the design parameters are some properties of electrical components. Detecting the most sensitive direction could be followed by investing in components with smaller tolerance in those directions.

3.5.2 Shape of the feasible region

Information about the shape of the feasible region can be extracted from the evolution of the proposal distribution and the collected samples. We discuss six shape features that can be obtained.

3.5.2.1 Upper bounds on approximated volume

The Loewner ellipsoid (Gruber, 2011) of a set of points is the unique minimal volume ellipsoid that contains this set of points. To approximate this ellipsoid we use Anye Lis Matlab implementation of Khachiyans algorithm (Khachiyan, 1996). Looking at the volume of the Loewner ellipsoid and the axes-aligned bounding box of the set of all feasible points found during one run of L_p-Adaptation, gives upper bounds for the volume approximation of the feasible region. Analysis of its evolution over time shows how well the feasible region is explored. As long as the volume of the Loewner ellipsoid is increasing, new parts of the feasible region are still being found. These bounds provide additional global geometric insight about the feasible region, e.g., if a Loewner ellipsoid or an axis-aligned bounding box gives a tighter bound on the feasible region and thus approximates it better. Large variances of the Loewner ellipsoids and axes-aligned bounding-boxes of several runs suggest that the feasible region is highly non-convex or disconnected.

3.5.2.2 Distortion of feasible region

A measure of distortion is the condition number of the covariance matrix $\mathbf{C}$. The condition number $\mathrm{cond}(\mathbf{C}) \geq 1$ is the ratio of the largest to smallest value in the singular value decomposition of the matrix. It indicates how much a unit sphere is distorted under the transformation by the matrix: the larger the condition number, the longer and thinner the unit sphere becomes. A well-conditioned matrix has a small condition number, an ill-conditioned matrix a large condition number. This provides information about the sphericity or aspect ratio of the feasible region.

3.5.2.3 MARGINAL STATISTICS

A feasible region can be visualized by its univariate and bivariate marginal distributions, which can be plotted.
Let $X_1, X_2, \cdots, X_i, X_{i+1}, \cdots, X_n$ denote n continuous random variables, then the univariate marginal distribution of X_i is

$$f_{X_i}(x_i) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \cdots x_n) dx_1, \cdots, dx_{i-1}, dx_{i+1}, \cdots, dx_n \quad (3.30)$$

and the bivariate marginal distribution of X_1 and X_i is

$$f_{X_1 X_i}(x_1, x_i) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \cdots x_n) dx_2, \cdots, dx_{i-1}, dx_{i+1}, \cdots, dx_n. \quad (3.31)$$

From the collection of samples obtained by L_p-Adaptation we can extract the i-th component sample to obtain the marginal distribution of X_i. To decrease sampling bias, we discard the samples obtained at the beginning of a run and use only those obtained after the burn-in period (the phase where the Markov-chain did not reach a high-probability region yet). With the remaining samples we do a kernel density estimation to get a smooth estimate of the marginal distribution.

3.5.2.4 CONVEXITY AND CONNECTIVITY

Usually, we use several runs of L_p-Adaptation, each using a different starting point. We then check if the runs were exploring the same part of the feasible region, i.e., find indications if the region is non-convex or disconnected. We do this by comparing:

- the means and the transformation matrices of the proposals
- the sample means and the sample covariances
- the approximated volumes and
- the outer approximations, Loewner ellipsoids, and axes-aligned bounding boxes.

If the variation between the upper bounds for the volume of the separate runs is still significant when every individual run seems converged, this is a first indication that the proposals did not approximate the same part of the feasible region and that the feasible region is likely to be non-convex and/or disconnected. If the outer approximations of the separate runs do not overlap, one could start investigating the area in-between to investigate if the different parts of the feasible region are connected. This could be done by sampling points along a line between two feasible points from different parts of the feasible region found (Hafner, 2010).

3.5.2.5 FIND p IF FEASIBLE REGION IS CLOSE TO L_p-BALL

The sample covariance of a sample uniformly drawn from a transformed L_p-ball and the transformation matrix $\mathbf{C}$ are equal up to a pre-factor that depends on n and p.

Lacko and Harman (2012) obtain the following proposition:
Let $\mathbf{X} \sim L_p^n(r)$, then

$$\textbf{(i)}\, \mathbb{E}[\mathbf{X}] = \mathbf{0}_n, \quad \text{and} \tag{3.32}$$

$$\textbf{(ii)}\, \mathrm{Cov}[\mathbf{X}] = \frac{B(\frac{3}{p}, \frac{n+p}{p})}{B(\frac{1}{p}, \frac{n+p+2}{p})} r^2 \mathbf{I}_n, \tag{3.33}$$

where

$$B(a,b) = \int_0^1 \nu^{a-1}(1-\nu)^{b-1} d\nu = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{3.34}$$

is the Beta function,

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{3.35}$$

is the Gamma function, and

$$\Gamma(z+1) = z\Gamma(z). \tag{3.36}$$

This leads to

$$\begin{aligned} \mathrm{Cov}[\mathbf{X}] &= \frac{\Gamma(\frac{3}{p})\Gamma(\frac{n+p}{p})}{\Gamma(\frac{3+n+p}{p})} \cdot \frac{\Gamma(\frac{1+n+p+2}{p})}{\Gamma(\frac{1}{p})\Gamma(\frac{n+p+2}{p})} r^2 \mathbf{I}_n & (3.37) \\ &= \frac{\Gamma(\frac{3}{p})\Gamma(\frac{n}{p}+1)}{\Gamma(\frac{1}{p})\Gamma(\frac{n+2}{p}+1)} r^2 \mathbf{I}_n. & (3.38) \end{aligned}$$

Applying Equation (3.36), it follows:

$$\mathrm{Cov}[\mathbf{X}] = \frac{n}{n+2} \cdot \frac{\Gamma(\frac{3}{p})\Gamma(\frac{n}{p})}{\Gamma(\frac{1}{p})\Gamma(\frac{n+2}{p})} r^2 \mathbf{I}_n. \tag{3.39}$$

If we now have a sample $\mathbf{X}$ from an L_p^n-ball with known radius r and transformation matrix $\mathbf{C}$

$$\mathbf{X} \sim L_p^n(r, \mathbf{C}), \tag{3.40}$$

then the sample covariance equals

$$\mathrm{Cov}[\mathbf{X}] = \frac{n}{n+2} \cdot \frac{\Gamma(\frac{3}{p})\Gamma(\frac{n}{p})}{\Gamma(\frac{1}{p})\Gamma(\frac{n+2}{p})} \cdot \mathbf{C}. \tag{3.41}$$

If the feasible region is close to an L_p-ball, this information can be used to choose a better proposal by finding the p with which the distance of the sample covariance and the transformation matrix $\mathbf{C}$ learned by L_p-Adaptation is minimized

$$\underset{p>0}{\arg\min} \quad d_F\left(\mathrm{Cov}[\mathbf{X}], \tfrac{n}{n+2} \cdot \tfrac{\Gamma(\frac{3}{p})\Gamma(\frac{n}{p})}{\Gamma(\frac{1}{p})\Gamma(\frac{n+2}{p})} \cdot \mathbf{C}\right), \tag{3.42}$$

where

$$d_F(\mathbf{A},\mathbf{B}) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(\mathbf{A},\mathbf{B})} \tag{3.43}$$

is the Foerstner distance (Förstner and Moonen, 1999) between two symmetric positive definite matrices $\mathbf{A}$ and $\mathbf{B}$, with Eigenvalues $\lambda_i(\mathbf{A},\mathbf{B})$ from $|\lambda\mathbf{A}-\mathbf{B}| = 0$. This tells to which L_p-ball the feasible region is most similar in shape.

3.5.2.6 Proposal changes with sequence of hitting probabilities

Besides giving better volume approximations and design centers, using a schedule of changing hitting probabilities also enables us to get more information about symmetries of the feasible region.

Assuming a fixed mean, looking at the ratio of two volumes approximated with different hitting probabilities, e.g.

$$b = \frac{\widetilde{\text{vol}}(A)_{P_1=0.35}}{\widetilde{\text{vol}}(A)_{P_2=0.85}}, \tag{3.44}$$

tells us how much volume of the larger proposal ($P_1 = 0.35$) was not included in the smaller proposal ($P_2 = 0.85$). If P_1 in the numerator is smaller or equal to P_2 in the denominator, $b \geq 1$. If $P_1 \neq P_2$ and $b = 1$, i.e., the volume approximations are identical, the used L_p-ball is a good approximation of the shape of the feasible region. We can be more certain about it being a good shape approximation if the difference of the target hitting probabilities is large enough. We found that using the default initial value of target hitting probability 0.35 to be a good choice for P_1 and a value above 0.65 a good chocie for P_2. If we use L_2-balls as proposals, the higher the value b, the more different the feasible region is from an ellipsoid, and b can be interpreted as a compactness measure.

Assuming the normalized transformation matrix $\mathbf{C}_n = \frac{\mathbf{C}}{\det(\mathbf{C})^{1/n}}$ is not changing with changing P, i.e., the proposals only differ in size, but not in shape, then a proposal mean **m**, that does not change with changing hitting probabilities, indicates that the feasible region is symmetric.

3.6 Conclusion

We have presented L_p-Adaptation, a statistical method that unites approximate design centering and volume estimation into a single framework. The method is based on using L_p-balls as proposals for sampling, which are dynamically adapted based on the previous samples in order to efficiently explore the feasible region. Maximizing the robustness (volume of the proposal) and the ability to control the hitting probability are the key concepts of L_p-Adaptation. Using a schedule of increasing or decreasing target hitting probabilities enables us to get better design centers and better volume approximations, respectively. The hitting probability in the design centering problem can be interpreted as the yield. If the shape of the feasible region and the proposal differ, a decreasing hitting probability ensures that more feasible volume can be 'seen' by the proposal, thus providing a better volume estimation. However, especially in higher dimensions ($\geq 50D$), this can require a very low hitting probability, which may lead to high computational costs.

The quality of the approximate solutions obtained from L_p-Adaptation depends on the unknown shape of the feasible region, the dimensionality of the problem, and the starting point. In practice, we thus recommend doing multiple L_p-Adaptation runs from different starting points found by initial optimization or brute-force sampling, and checking the quality of the resulting design centers and volume estimates.

If prior knowledge about the shape of the feasible region is available, it can be included in the initial covariance matrix. From there, the covariance matrix then automatically adapts to the feasible region. For an example, see Figure 3.13, where the feasible region is shaded in gray (its shape is similar to the example from Storn (1999) discussed in Section 4.1). It consists of a large rectangle with a thin attached arm. The initial proposal distribution (red circle) is isotropic and has its mean at the very tip of the thin arm. If we would fix the proposal to be isotropic (green circle), the chance of finding the large rectangle is much lower than if we allow

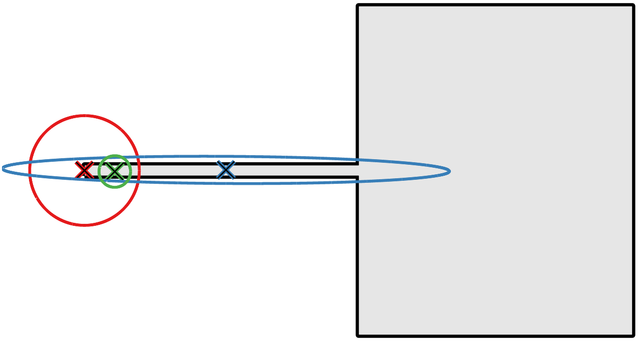


Figure 3.13: Illustration of covariance matrix adaptation. The feasible region is shaded in gray. It consists of a larger rectangle with a thin arm attached to it. If we start with an isotropic proposal distribution at the tip of this arm (red circle), L_p-Adaptation finds the larger rectangle by continuously adapting the proposal. The blue ellipse shows the adaptation to the thin arm. Fixing the proposal to be isotropic would shrink it, to maintain the target hitting probability (green circle), and make it impossible to adapt to the thin arm.

the proposal to adapt its shape (blue ellipse). We recommend to only fix the covariance matrix if one is interested in the yield of a specific point (e.g., a previously found design center) without affine covariance adaptation. When fixing the covariance matrix to be isotropic, the radius of the proposal associated with a specific target hitting probability P tells how far the design center can be perturbed in any direction and still remain feasible with probability P (see also Section 5.1, Figure 5.2, where we compare the robustness of three design centers using this method).

CHAPTER

FOUR

VALIDATION

We demonstrate the properties of L_p-Adaptation in some benchmark cases, where the correct answers are known. First we consider three simple 2D test cases from the literature. This is followed by basic L_p-balls in up to 90 dimensions.

4.1 ILLUSTRATION IN TWO DIMENSIONS

Figure 4.1 shows the behavior of L_p-Adaptation in a classic 2D test case from the literature (Storn, 1999). The feasible region is shaded in gray. It consists of a large rectangle ($x_2 \in [-10, 10]$ and $x_1 \in [5, 10]$) with a thin attached arm ($x_2 \in [-0.1, 0.1]$ and $x_1 \in [0, 5]$). The region is not convex, and the thin arm is deceiving for an algorithm. We start L_p-Adaptation from a feasible point at the very tip of this arm, at $(0, 0)$ (red cross in Fig. 4.1). The initial proposal distribution (red ellipse) is isotropic with a radius of 1, which does not allow the algorithm to "see" the large rectangle initially. After 355 function evaluations, however, the proposal distribution

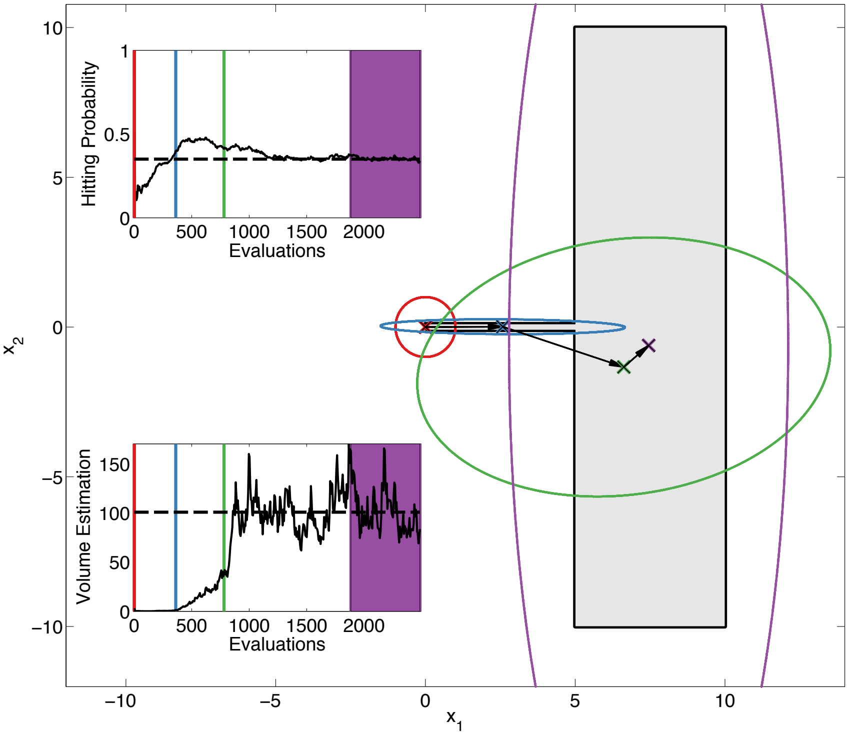


Figure 4.1: Illustration of L_p-Adaptation using the test case of Storn (1999). The feasible region (shaded gray) is non-convex and deceiving, consisting of a large rectangle with a thin arm attached to it. The algorithm is started from the tip of the arm, which is the hardest possible stating point (red). Even though it initially cannot "see" the large rectangle, the proposal distribution adapts to the shape of the arm (blue) within 355 evaluations and discovers the large rectangle after 775 evaluations (green). The evolution of the design center is shown by crosses, the L_2 proposal is shown as ellipses. Colors correspond to the evaluation numbers indicated. The two inset plots show the evolution of the hitting probability and the volume estimate versus the number of function evaluations. The dashed black lines indicate the target hitting probability and ground-truth volume, respectively. The final design center and volume (purple) are obtained by averaging over the last 600 evaluations.

has adapted to the shape of the thin arm (blue ellipse) and points in the large rectangle start to be found. This progressively moves the center of the proposal toward the large rectangle, which is reached after 775 evaluations (green ellipse). The algorithm is run until 2500 evaluations, when it has reached a stationary state (stationary random process). Averaging over the last 600 evaluations (purple areas in the inset figures) provides the final estimates of the design center and volume of the feasible region. Since the process converges in distribution, i.e., the random process becomes stationary, averaging is meaningful. The final volume estimate is 100.8 (true value: 101.0) and the design center is (7.45, -0.62). This compares well with the exact center of the large rectangle, which is (7.5, 0.0). Any point with $x_1 = 7.5$ and $x_2 \in [-7.5, 7.5]$ has maximum distance from any border of the feasible region.

Before the proposal distribution has adapted to the shape of the thin arm, the effective, empirical hitting probability is lower than the target hitting probability (dashed line in the inset figure). Thereafter, the hitting probability increases until the mean of the proposal distribution has moved into the large rectangle. After around 1200 evaluations the target hitting probability is reached and maintained. The lower inset plot shows how the algorithm “discovers” the large part of the feasible region. Around 775 evaluations (green line), the estimated volume sharply increases as the large rectangle becomes visible, and it finally fluctuates around the true value of 101.0 (dashed line).

Figure 4.2 shows two other classic 2D test cases from the literature: the “Handle” (Lasserre, 2014) (Fig. 4.2a) and the “Folium” (Henrion et al., 2009) (Fig. 4.2b). Again, the feasible regions are shaded in gray. The volume approximations obtained by different methods are shown in the panels below. L_p-Adaptation (red stars) uses an ellipsoidal proposal distribution ($p = 2$) and starts from ten random feasible points. The results are compared with those from uniform sampling (“bruteforce”, gray diamonds) and with the upper bounds provided by Loewner ellipsoids (Gruber, 2011) (blue circles) and the axes-aligned bounding box (green squares) of all feasible samples. The true volume is indicated by the dashed black line. Brute-force sampling provides the most accurate result, but is exponentially inefficient with dimension. After around 600 evaluations for the “Handle” and 150 evaluations for the “Folium”, the results from L_p-

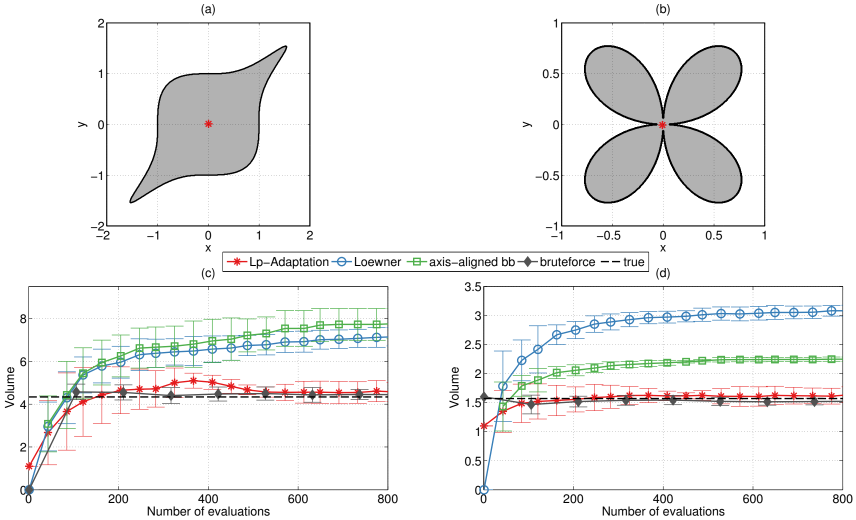


Figure 4.2: Two 2D test cases from the literature: (a) the "Handle" (Lasserre, 2014), given by the parametric equation $x^6 + y^6 - 1.925x^3y^3 \leq 1$*, and (b) the "Folium" (Henrion et al., 2009) parameterized by* $-(x^2 + y^2)^3 + 4x^2y^2 \geq 0$*. Both feasible regions (shaded gray) are non-convex. The red stars are the averaged design centers (of ten individual runs). (c,d) Average and standard deviation (over ten independent runs) of the volume estimates obtained with* L_p*-Adaptation (red stars) and brute-force exhaustive sampling (gray diamonds). Upper bounds are obtained from the Loewner ellipsoid (blue circles) and axes-aligned bounding box (green square) of the samples generated by* L_p*-Adaptation. The true volume is indicated by the dashed black line.*

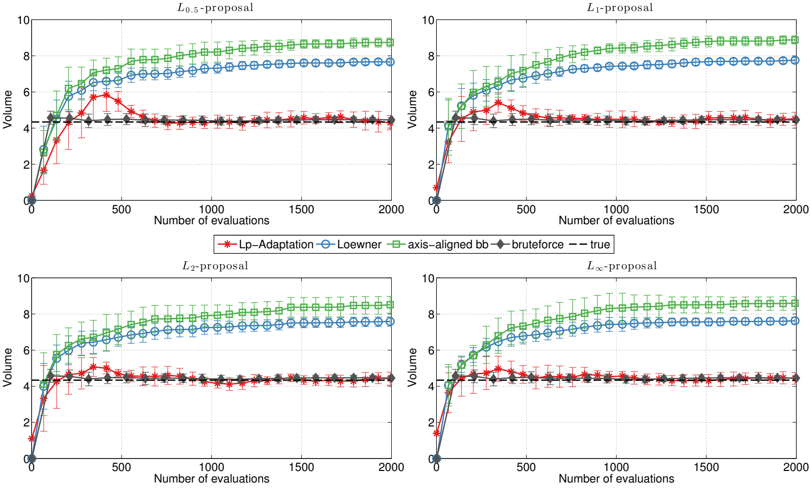


Figure 4.3: Average and standard deviation over ten independent runs of L_p-Adaptation to estimate the volume of the "Handle" ((Lasserre, 2014)). The results are compared with brute-force sampling. Two different upper bounds are obtained from the standard Loewner ellipsoid and the axis-aligned bounding box of all feasible points. The four panels use different L_p-balls as proposal distributions.

Adaptation are within the error bars from the correct volume. The final centers of the estimated volumes are shown by red dots in the upper panel of Figure 4.2. In both cases they are at the geometric center of the body. However, the geometric center of the estimated volume need not be a robust design center, as evident for the "Folium".

The large variances of the Loewner ellipsoids and the axes-aligned bounding boxes for the "Handle" indicate that the individual runs differ in how well they explore the outermost arms of the Handle. Figures 4.3 and 4.4 show the results with other L_p-balls and for higher numbers of function evaluations. Similar to what we observed for $p = 2$, L_p-Adaptation provides a good volume approximation in all cases. Not only the L_p-balls with $p = 2$, but also with $p = 0.5$, 1, and ∞ lead to good results. Initially the radius of all L_p-balls is set to 1. Therefore, the first volume approximation is larger the higher the chosen p-norm of the L_p-ball. As long as the volumes

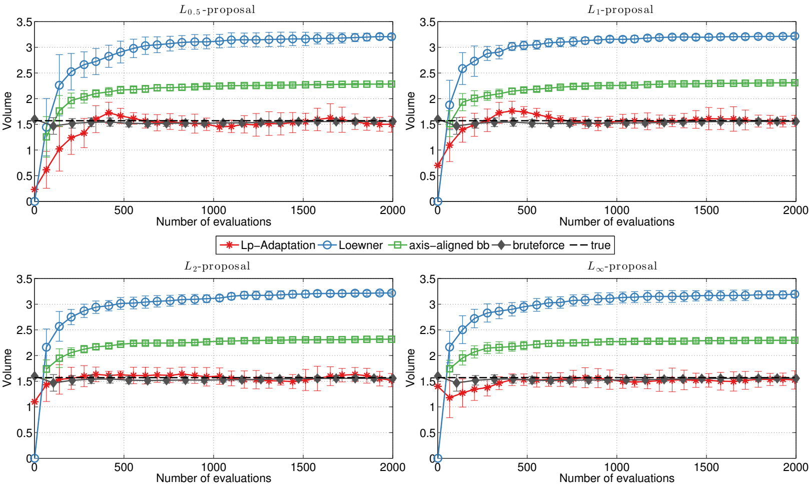


Figure 4.4: Average and standard deviation over ten independent runs of L_p-Adaptation to estimate the volume of the "Folium" (Henrion et al., 2009). The results are compared with brute-force sampling. Two different upper bounds are obtained from the standard Loewner ellipsoid and the axis-aligned bounding box of all feasible points. The four panels use different L_p-balls as proposal distributions.

of axis-aligned bounding box and Loewner ellipsoid are increasing, new points in the corners of the test bodies are found. We conclude that the choice of p-norm for the proposal is irrelevant in these low-dimensional examples and that the estimation accuracy of L_p-Adaptation is comparable with that of brute-force sampling.

4.2 SYNTHETIC L_p–BALLS

In order to test the algorithm also in higher dimensions, we consider synthetic L_p-balls as feasible regions. Unlike the fixed-dimensional test cases from the literature, these can be scaled to arbitrary dimension with the true result still known in all cases. We use L_p-Adaptation to estimate the volumes of these L_p-balls, which is particularly interesting when the true p and the proposal p do not match.

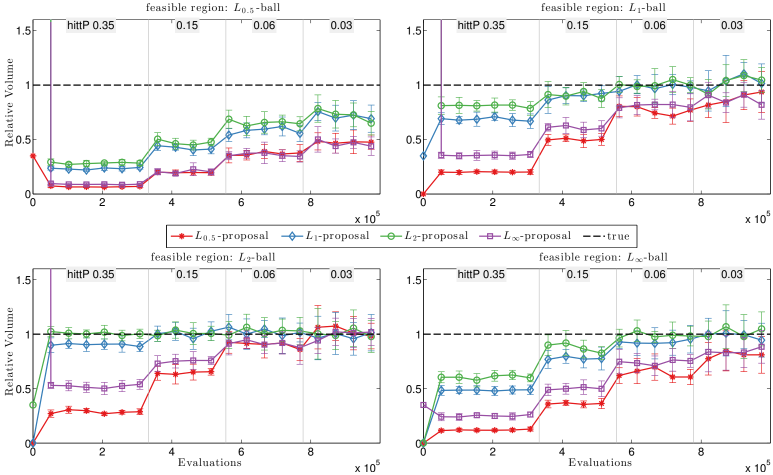


Figure 4.5: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume / true volume) of ten-dimensional unit L_p-balls ($p = 0.5, 1, 2, \infty$) approximated with proposal distributions of different p. The results are shown for decreasing target hitting probability, starting from the default initialization 0.35, as indicated at the top of each plot. The dashed line shows the true volume.

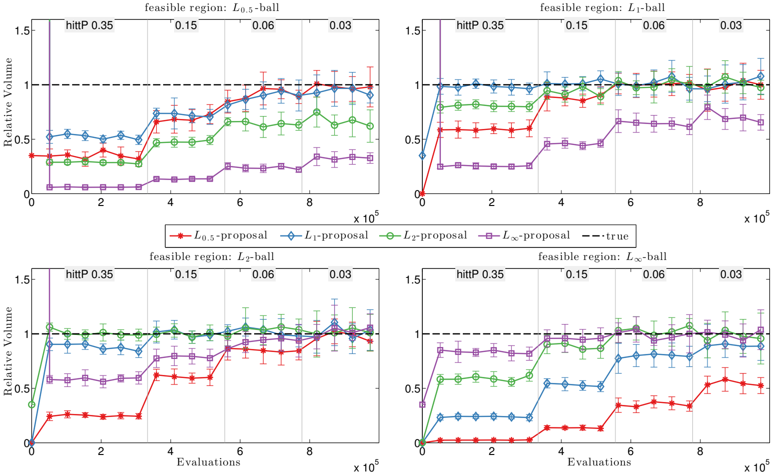


Figure 4.6: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume / true volume) of stretched ten-dimensional L_p-balls ($p = 0.5, 1, 2, \infty$) approximated with proposal distributions of different p. The feasible L_p-balls are stretched along $(n-1)$ axes such that the longest axis is $\sqrt{1000}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced. The results are shown for decreasing target hitting probability, starting from the default initialization 0.35, as indicated at the top of each plot. The dashed line shows the true volume.

Figure 4.5 shows the normalized estimated volumes for different feasible regions (feasible regions with p-norms 0.5, 1, 2, and ∞) when using different proposals (proposal p-norms 0.5 (red), 1 (blue), 2 (green), and ∞ (purple)) in 10 dimensions. The true volume is indicated by the dashed black line. All feasible regions are unit L_p-balls, e.g. their condition number is 1. In Figure 4.6, all feasible regions are stretched along $(n-1)$ axes such that the longest axis is $\sqrt{1000}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced. Therefore the condition number is 1000. The figures show the volume estimates versus the number of function evaluations. We start from the standard target hitting probability of 0.35 in order to first learn the shape of the feasible region. As indicated at the top of the plots, we then successively lower it to 0.15, 0.06, 0.03, and 0.01 in order to refine the volume estimate. Intuitively, one would assume that the results are best when the L_p-ball used as a proposal for sampling matches the true shape of the feasible region, i.e., if the two p values are identical. However, if the feasible region is isotropic, the L_2-proposal is always better than the other proposals (Figure 4.5). In the cases where the L_p-ball is stretched (Figure 4.6) the other proposals catch up and the 'true' p is actually better for $p=1$, $p=2$, and $p=\infty$, but with decreasing P, the estimation with an L_2-proposal (for L_1-ball and L_∞-ball) or L_1-proposal (for L_2-ball) is equally good. The volume estimation of the $L_{0.5}$-ball is best with an L_1-proposal, but with decreasing P, the $L_{0.5}$-proposal obtains equally good results. With increasing dimension (see Figure 4.7 for 20D), it becomes more prominent that using either an L_1-ball or an L_2-ball as proposal seems to be a good choice independent of the shape of the feasible region. In 50D the volumes of the $L_{0.5}$-ball and the L_∞-ball are not estimated well anymore by any proposal, see Figure B.1.

In 20D we also test GaA (Müller and Sbalzarini, 2010a) on all four anisotropic test bodies and the state-of-the-art convex volume estimation algorithm of Cousins and Vempala (Cousins and Vempala, 2016) on the convex anisotropic test cases ($p=1,2,\infty$), see Figure 4.7. Compared to GaA (orange) with fixed hitting probability 0.01, L_p-Adaptation works better in all cases. GaA consistently underestimates the volume except when the feasible region is an L_2-ball, which matches the shape of the Gaussian proposal. The convex volume estimation algorithm of Cousins and Vempala (Cousins and Vempala, 2016) (gray) equals or outperforms L_p-

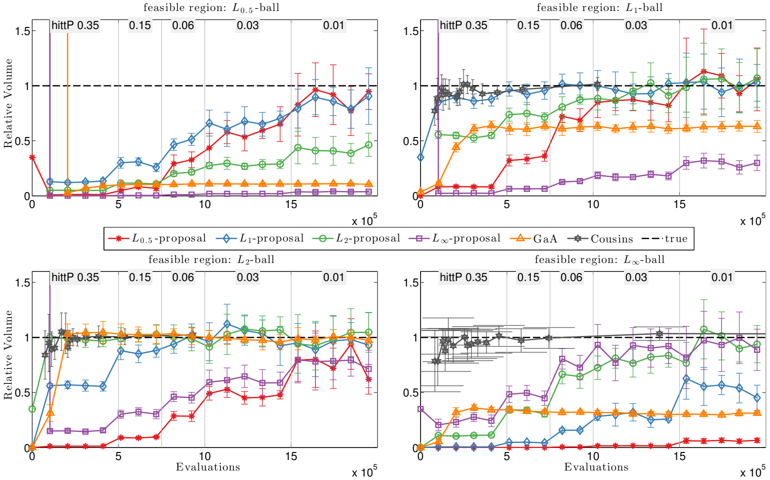


Figure 4.7: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume / true volume) of stretched 20-dimensional L_p-balls ($p = 0.5, 1, 2, \infty$) approximated with proposal distributions of different p. The feasible L_p-balls are stretched along $(n-1)$ axes such that the longest axis is $\sqrt{1000}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced. The results are shown for decreasing target hitting probability, starting from the default initialization 0.35, as indicated at the top of each plot. The dashed line shows the true volume. For comparison, we also show the results obtained with Gaussian Adaptation and Cousins' convex volume estimator (Cousins and Vempala, 2016).

Adaptation for convex L_p-balls. This is expected because the algorithm is specialized in convex bodies and has direct access to the underlying geometric description of the body (a polytope or an ellipsoid) whereas L_p-Adaptation requires no prior knowledge. Remarkably, L_p-Adaptation shows similar convergence for L_1-balls and L_2-balls when using the same bodies as proposals. We conclude that L_p-Adaptation performs better than Gaussian Adaptation and reaches the same accuracy as specialized algorithms for convex bodies, albeit without requiring convexity or prior knowledge about the shape of the feasible region.

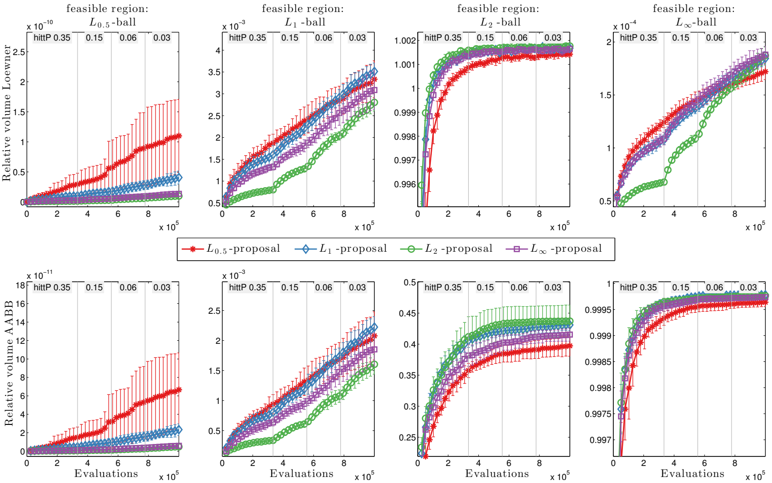


Figure 4.8: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) of isotropic ten-dimensional L_p-balls ($p = 0.5, 1, 2, \infty$) approximated with proposal distributions of different p.

By looking at the volumes of the outer approximations of the feasible region, i.e., the Loewner ellipsoids and the axis-aligned bounding boxes, we can see how explorative a proposal is. The higher the volume of the outer approximations, the better a feasible region is explored. In Figure 4.8 we show the relative volumes of the Loewner ellipsoid (top row) and the axis-aligned bounding box (bottom row) of isotropic 10-dimensional L_p-balls explored with different L_p-proposals. In 10D, explorativeness of the L_2-proposal is independent of the condition of the feasible region, see Figure B.6. For the other proposals it makes a difference, whether the feasible region is isotropic (see Figure 4.8) or stretched along the axes (see Figure 4.9). If the feasible region is a L_2-ball (isotropic or stretched), all proposals are equally good in exploring the feasible region such that the volume of the associated Loewner ellipsoid is close to 1, see Figure B.2. Similarly, if the feasible regions is a L_∞-ball (isotropic or stretched), the volume of the associated axis-aligned bounding box is close to 1, see Figure B.3. In the isotropic cases, the different proposals are equally explo-

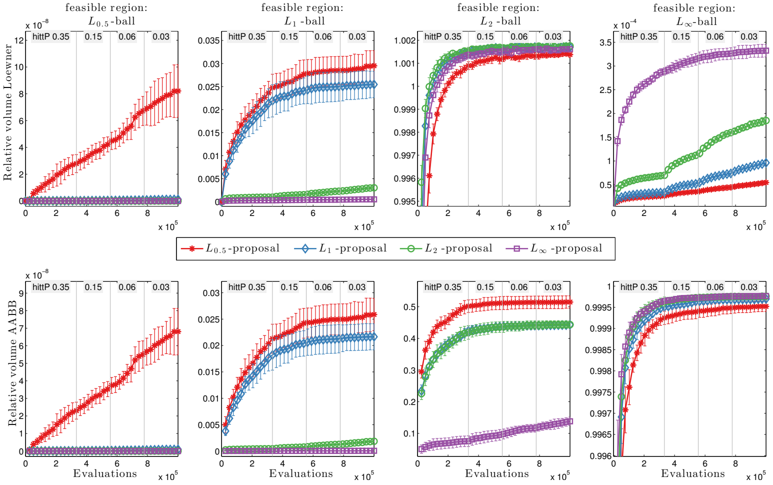


Figure 4.9: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) of stretched ten-dimensional L_p-balls ($p = 0.5, 1, 2, \infty$) approximated with proposal distributions of different p. The feasible L_p-balls are stretched along $(n-1)$ axes such that the longest axis is $\sqrt{1000}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced.

rative for the L_1-ball, L_2-ball, and L_∞-ball as feasible regions (Figure 4.8), whereas in the non-isotropic cases (Figure 4.9) the explorativeness differs depending on the proposals shape. In terms of explorativeness, using an L_∞-proposal is of advantage only if the feasible region is an L_∞-ball. In the other cases, the $L_{0.5}$-proposal reaches the highest volumes for both outer approximations (Figure 4.9). The pointy shape of an $L_{0.5}$-proposal is particularly well-suited for exploration of those cases. It seems like a contradiction, that the $L_{0.5}$-proposal is the most explorative proposal in those cases, but not among the best regarding the volume approximation. A possible reason for this could be that an $L_{0.5}$-proposal is more sensitive to small rotations than the other proposals.

In general, we see that an L_p-proposal with matching p can explore an L_p-ball better if the condition of the ball is higher. We suspect that a reason for this could be that with higher aspect ratios less dimensions have to be

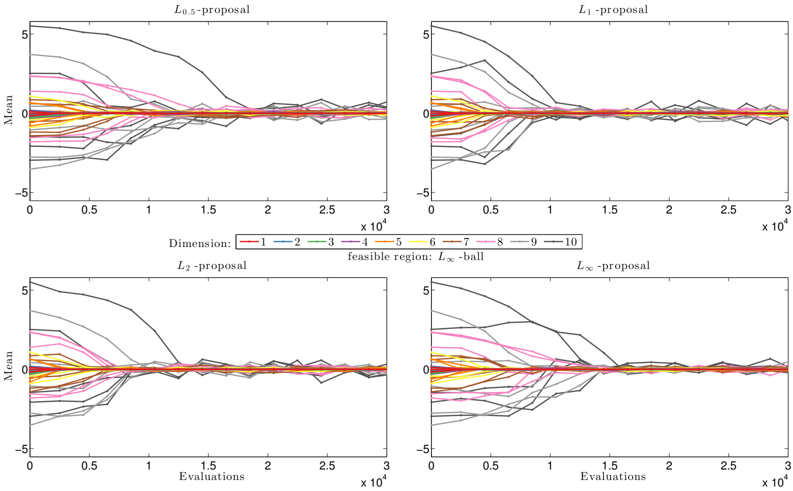


Figure 4.10: Trajectories of proposal means (of five independent runs), where the feasible region is a L_∞^{10}-ball that is stretched along $(n-1)$ axes such that the longest axis is $\sqrt{1000}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced. In each plot, a different proposal p is shown. The color code shows the components of the mean in ten dimensions.

found correctly since most of the mass of the feasible region is found along fewer dimensions.

$L_{0.5}$-proposals and L_1-proposals are more explorative for an isotropic L_∞-ball than for a stretched L_∞-ball, see Figures B.4 and B.5. L_2-proposals are equally explorative for L_∞-balls (see Figure B.6), independent of their condition. An L_∞-proposal is more explorative for isotropic L_p-balls (for $p = 0.5, 1, 2$) than if they are stretched, see Figure B.7. These are interesting observations, but to fully understand them, further theoretical investigations would be necessary.

In Figure 4.10 we show trajectories of five runs of the proposal means, where the feasible region is the stretched L_∞^{10}-ball. In Figure 4.11, we show the corresponding Eigenvalues of the normalized transformation matrix $\mathbf{C}$ of the proposal. Note that the runs are plotted until 30000 evaluations (all other plots are shown until the end of the runs at $n \cdot 10^5$ evaluations). After

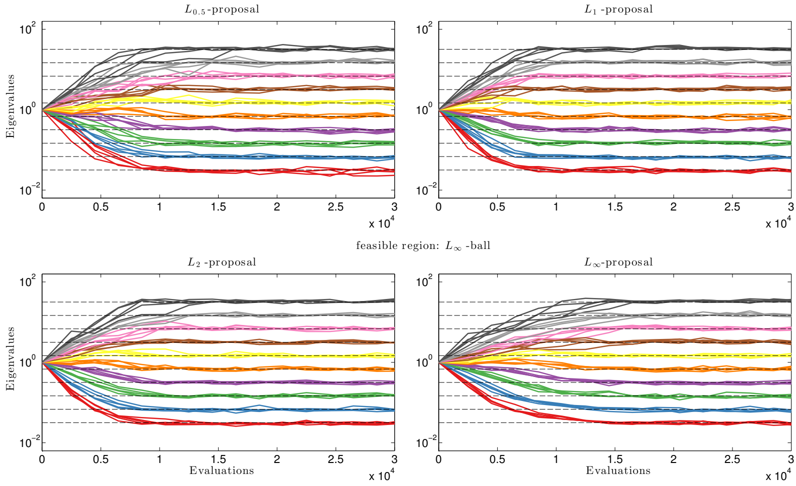


Figure 4.11: Eigenvalues of the normalized transformation matrix **C** *obtained from five runs of* L_p*-Adaptation, where the feasible region is a* L_∞^{10}*-ball, that is stretched along* $(n-1)$ *axes such that the longest axis is* $\sqrt{1000}$ *times longer than the shortest one, and the lengths of the axes are logarithmically spaced. In each plot, a different proposal* p *is shown. The black dashed lines show the true Eigenvalues.*

$2 \cdot 10^4$ evaluations, the means of all proposals are fluctuating around the true mean of $\mathbf{0}$. The means of the L_1-proposals and the L_2-proposals converge faster than the means of the $L_{0.5}$-ball and the L_∞-ball (Figure 4.10). After $1.5 \cdot 10^4$ evaluations all proposals have learned the rough shape of the stretched L_∞^{10}-ball (Figure 4.11).

To test how many evaluations we need to reach a certain precision of the volume approximation, we use L_1-, and L_2-balls as feasible regions, both isotropic and stretched along the axes such that the longest axis is $\sqrt{1000}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced in $2, 5, 10, 20, 50, 70$, and 90 dimensions. For all cases we test an L_1-proposal and an L_2-proposal with a variable schedule of changing hitting probabilities. As shown in Figure 4.12, in all but one case (approximating an isotropic L_1^{90}-ball with L_1^{90}-proposal) the volume of the feasible regions can be approximated with a relative error of 0.1 with the used proposals. Approximating the L_2-ball is easier than approximating

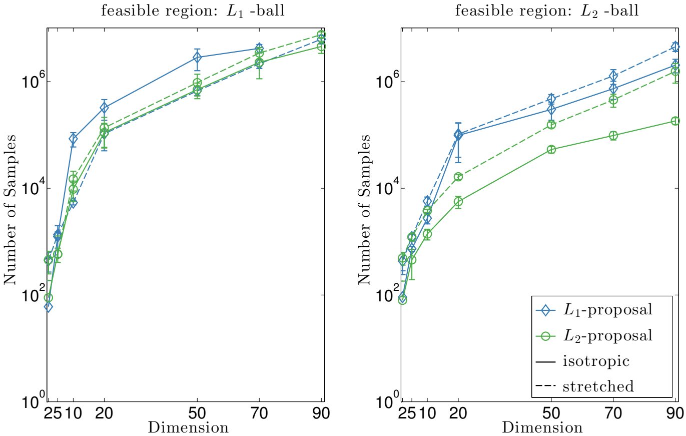


Figure 4.12: Number of samples required until relative error of the volume approximation $\frac{|\operatorname{vol}(A)-\widetilde{\operatorname{vol}}(A)|}{\operatorname{vol}(A)} \leq 0.1$*, where the feasible region* A *is an* L_1*-ball (left) and an* L_2*-ball (right), respectively, both isotropic (continuous line) and stretched along the axes (dashed line). The proposal distributions are* L_1*-balls (blue diamond) and* L_2*-balls (green circle).*

the L_1-ball. The L_2-proposal reaches this relative error faster than the L_1-proposal. The number of evaluations needed to reach this precision scales roughly polynomial with space dimension.

4.3 CONCLUSION

We have illustrated the behavior of L_p-Adaptation in several 2D examples and found that the choice of the p-norm for the proposal was irrelevant in low dimensional examples and that the accuracy is comparable to exhaustive sampling. Furthermore, we showed that the mean and the covariance converge to the correct shape. By testing L_1-balls and L_2-balls as feasible regions in up to 90D, we found that L_p-Adaptation scales polynomially with dimension and not exponentially (as is the case with exhaustive sampling).

Increasing the dimension, we saw interesting behavior of the proposal explorativeness depending on the p-norm of both the feasible region and the proposal, and the condition number of the feasible region. If the p-norm of feasible region and proposal match, the proposal is more explorative the higher the condition number gets. A possible explanation for this could be that the higher the condition number, the more mass of the feasible region is distribution in fewer dimensions and thus the problem becomes easier for a proposal of same shape. Further theoretical investigations could lead to a better understanding of the behavior. This could then be used for a guided optimization of the algorithmic parameters of L_p-Adaptation concerning the update of the transformation matrix $\mathbf{C}$ and the proposal's mean $\mathbf{m}$ depending on the p-norm of the proposal.

In our studies we see that using either the L_1- or the L_2-ball as a proposal is at least as good as using the shape of the underlying feasible region. For unknown feasible regions we therefore suggest to use the L_2-ball as a default proposal.

CHAPTER

FIVE

Application

Here we show the application of L_p-Adaptation in three examples. First we show how L_p-Adaptation can be used to design a switched capacitor filter and, second, how we can use it to quantify basin stability of synchronized states in mutually delay-coupled oscillators. In the third presented application, which is from systems biology, we compare the robustness of two bacterial two component systems.

5.1 Designing a Switched Capacitor Filter

Switched Capacitor (SC) filters are a modern replacement for Resistor Capacitor (RC) filters. They are well suited for integration on silicon chips, due to reduced sensitivity of their transfer function to manufacturing inaccuracies. While the basic design processes of SC and RC filters are similar, a key challenge in SC filter design is parasitic capacitance. Here, we consider a design scenario for an SC-based pulse code modulation (PCM) low-pass filter with parasitic capacitances, as introduced as a test case by

Storn (1999). The circuit diagram of this example is shown in Figure 5.1a. The transfer function of this SC-PCM filter is:

$$H(f) = \frac{V_{\text{out}}}{V_{\text{in}}} = H_1(f) \cdot H_2(w) \cdot H_3(w) \tag{5.1}$$

with

$$w = j \cdot \tan \frac{\pi f}{f_a} \tag{5.2}$$

and

$$j = \sqrt{-1}, \tag{5.3}$$

where f_a is the sampling frequency of the filter and f is the signal frequency. The chosen f_a for $H_2(w)$ is 128 kHz, and 32 kHz for $H_3(w)$, as in the original publication (Storn, 1999).

The transfer function of the analog RC lowpass pre-filter is:

$$H_1(f) = \frac{1}{1 + j \cdot 2\pi f \cdot R_0 \cdot C_0} = \frac{1}{1 + j \cdot f \cdot v_1}. \tag{5.4}$$

For the SC-PCM lowpass filter with parasitic capacitances, the transfer functions $H_2(w)$ and $H_3(w)$ can be written as (Storn, 1999)

$$H_2(w) = \frac{w^2 \left[\left(v_{32} - \frac{\gamma}{2}\right)\left(v_{12}\beta + \frac{\gamma}{2}\right) + v_{132}\frac{\gamma}{2}\right] + w\left[\alpha(v_{32} - v_{132}) + v_{12}\beta\alpha\right] + \alpha^2}{w^2 \left[\left(v_{32} + v_{532} - \frac{\gamma}{2}\right)\left(v_{12}\beta + \frac{\gamma}{2}\right) + v_{132}\frac{\gamma}{2}\right] + w\left[\alpha(v_{32} + v_{532} - v_{132}) + v_{12}\beta\alpha\right] + \alpha^2}$$

$$H_3(w) = \frac{w^2 \left[\left(v_{33} - \frac{\gamma}{2}\right)\left(v_{13}\beta + \frac{\gamma}{2}\right) + v_{133}\frac{\gamma}{2}\right] + w\left[\alpha(v_{33} - v_{133}) + v_{13}\beta\alpha\right] + \alpha^2}{w^2 \left[\left(v_{33} + v_{533} - \frac{\gamma}{2}\right)\left(v_{13}\beta + \frac{\gamma}{2}\right) + v_{133}\frac{\gamma}{2}\right] + w\left[\alpha(v_{33} + v_{533} - v_{133}) + v_{13}\beta\alpha\right] + \alpha^2},$$

where $\alpha = \left(1 + \frac{\gamma}{2}\right)$, $\beta = (1 + \epsilon)$ and the constants $\gamma \in [2.55\%, 10.5\%]$ and $\epsilon \in [0.1\%, 1\%]$ represent the parasitic effects. We set $\gamma = 5\%$ and $\epsilon = 0.5\%$, as in the original publication (Storn, 1999).

According to these expressions, the filter has nine design parameters: $\{v_1, v_{12}, v_{32}, v_{132}, v_{532}, v_{13}, v_{33}, v_{133}, v_{533}\}$, defining a 9-dimensional design space. The feasible region consists of all points in that space for which $|H(f)|$ lies within the specifications given by the solid black lines in Figure 5.1b. They define the gain, ringing, and sharpness characteristics of the filter. A filter fulfills the specifications if the magnitude of its transfer function, $|H(f)|$ is above $\{1.0, 1.0292, 1.0, 0.031623\}$ at frequencies

$\{0, 200, 3600, 4600\}$Hz and below $\{0.0, 0.97162, 0.94951, 0.90157, 0.0\}$ at frequencies $\{0, 300, 2400, 3000, 3400\}$Hz.

We start L_p-Adaptation from ten different initial feasible points found by parameter optimization using the Gaussian adaptation (GaA) algorithm (Kjellström and Taxen, 1981; Müller and Sbalzarini, 2010b). For this initial optimization, we allow v_1 to vary within the interval $[\mathrm{e}^{-9}, \mathrm{e}^3]$ and all other parameters in $[\mathrm{e}^{-3}, \mathrm{e}^3]$. The objective function is the sum of squared deviations between the realized $|H(f)|$ and the prescribed upper and lower specification boundaries of the transfer function across the frequency range f. The optimization starts from ten different points, sampled uniformly in the natural logarithm of the entire parameter domain.

Figure 5.1c shows four selected pairwise density contour plots of the feasible points obtained by the ten runs of L_p-Adaptation. A high density of feasible points is shown in red, low density in blue. The lines are curves of constant density. The black trajectory shows the evolution of the mean of the proposal distribution for the run that yielded the largest volume approximation. Individual iterations of the algorithm are shown by stars. The yellow star is the final design center found by this run. Because the feasible region in this example is not convex, different runs find different design centers in different parts of the feasible region. Two design centers found by Storn (1999) in the same part of the feasible region is indicated by the green diamond and the magenta circle.

We assess the robustness of the design centers found by L_p-Adaptation by comparing with the results reported by Storn (1999). Robustness is measured by the size (radius) of a hyper-cube or hyper-ellipsoid that contains a given fraction of feasible points as discussed in Section 3.6, see Figure 5.2. In all cases, the design centers found by L_p-Adaptation are more robust than the previous results, as indicated by the larger radii for all target hitting probabilities.

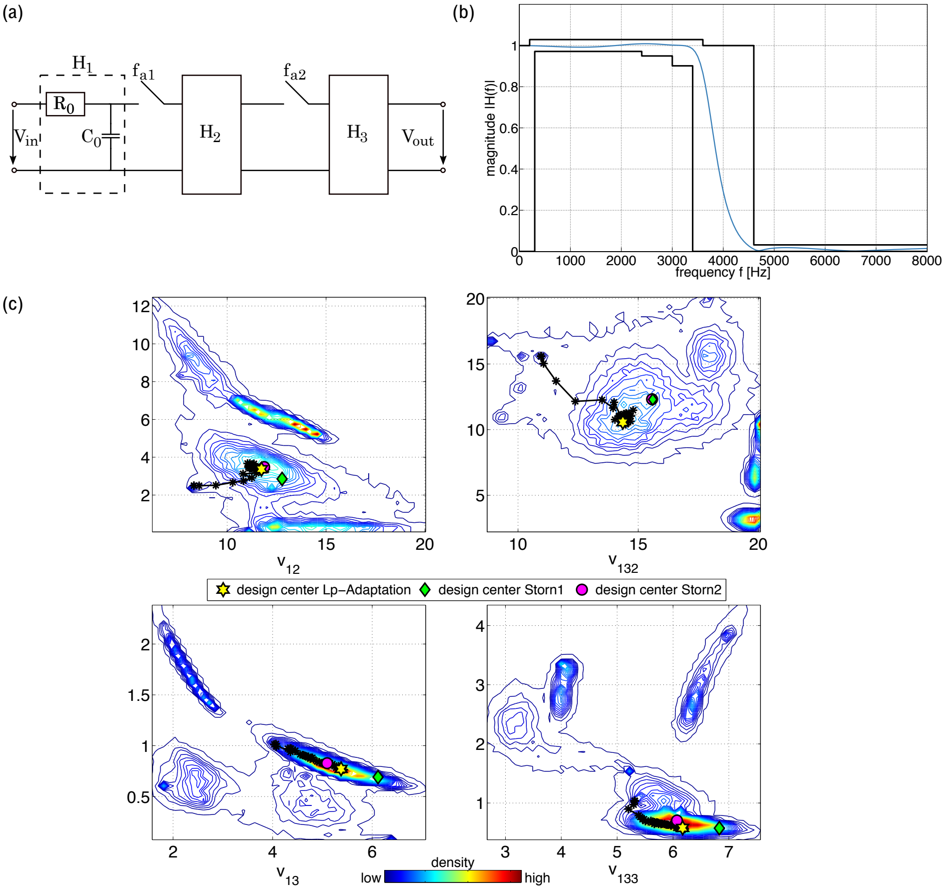


Figure 5.1: (a) Circuit diagram of the Switched Capacitor (SC) filter. (b) Specifications for the transfer function of the filter (black lines) and one example satisfying the constraints (blue line). (c) Pairwise density contour plots of the feasible points obtained by ten runs of L_p-Adaptation with different starting points in the nine-dimensional design space. The black trajectory show the evolution of the mean for the run that yielded the largest volume approximation. The yellow star is the final design center obtained by this run. Two design centers reported by Storn (1999) are indicated by the green diamond and the magenta circle.

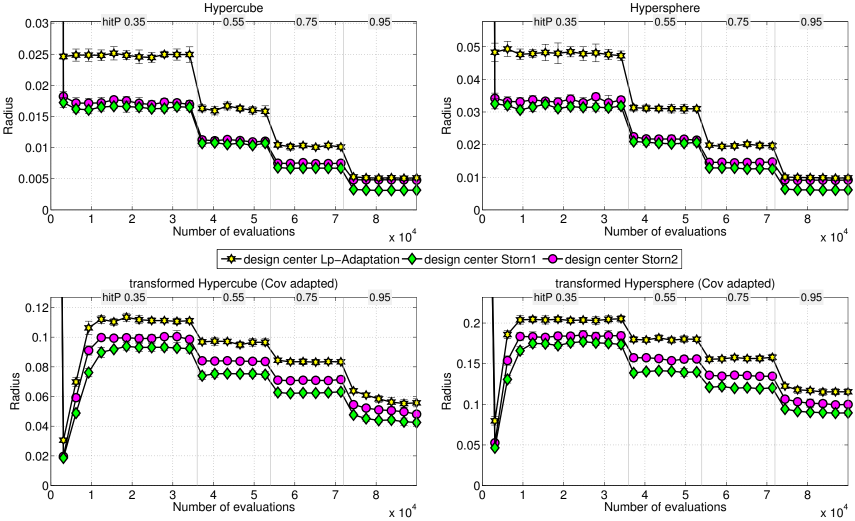


Figure 5.2: Comparison of the robustness of three different design centers, the larger the radius, the more robust the design centers: one found by L_p*-Adaptation (yellow star) and two reported by Storn (Storn, 1999) (green diamond and magenta circle). We run* L_p*-Adaptation for these different design centers with a fixed mean and a decreasing schedule of the hitting probability as shown at the top of each plot. In the top row we show the results for the runs with fixed covariances (identity matrix), in the bottom row we allow covariance adaptation. We test two proposals:* L_∞ *(left, hyper-cube) and* L_2 *(right, hyper-sphere), both with (bottom row) and without (top row) affine covariance adaptation. The plots show the radius of a hypercube/hypersphere such that approximately {0.35%, 0.55 %, 0.75%, 0.95%} of the points in this body are feasible, as indicated by the target hitting probabilities at the top of each plot.*

5.2 QUANTIFYING BASIN STABILITY OF SYNCHRONIZED STATES IN MUTUALLY DELAY-COUPLED OSCILLATORS

In this section we present ongoing work in a collaboration with Lucas Wetzel, Daniel Platz, Benjamin Friedrich and Alexandros Pollakis.

Phase-locked loops (PLLs) are electronic control systems (circuits) that are able to synchronize by evaluating mutual phase differences and adjusting their frequencies accordingly (Pollakis et al., 2014; Wetzel et al., 2017). As illustrated in Figure 5.3, a PLL consists of three main parts: a phase detector (PD), a loop filter (LF), and a voltage controlled oscillator (VCO). These components are organized in a loop. The output voltage of the PD represents the phase relation between the input signal and the feedback from the VCO. The LF dampens the high-frequency components, and the remaining low frequency components, i.e., the phase differences, are the input signal for the VCO (Pollakis et al., 2014).

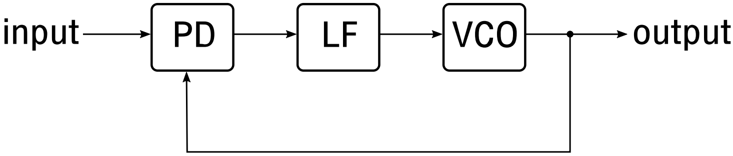


Figure 5.3: Schematic illustration of a phase-locked loop (PLL). A PLL consists of a phase detector (PD), a loop filter (LF), and a voltage controlled oscillator (VCO).

PLLs are widely used in electronic applications, such as radio, communications, and computers. We study systems of mutually coupled digital phase-locked loops (DPLLs). We want to analyze synchronized states of delay-coupled DPLL systems. Linear stability analysis allows to analyze how small perturbations affect synchronized states, but the dynamics of larger perturbations are not described. Here, we use L_p-Adaptation to study the effect of (larger) perturbations on synchronized states. The set of initial points in phase space from which the system converges to such a synchronized state, is called the basin of attraction of that state. The basin's volume can be seen as a measure of its stability (Menck et al., 2013). Menck et al. (2013) estimate the basin's volume in Watts-Strogatz networks of Rössler oscillators by uniformly sampling points from an a-priori defined subspace of the state space and counting the number of points that arrive at the synchronized state. Using L_p-Adaptation for a similar problem, we can estimate the basin's volume without any prior knowledge of its size because our method adaptively learns the size of the basin (i.e., the feasible region). In the remainder of this section, we first describe the phase model for the mutually coupled DPLLs and the Kuramoto order parameter, then we explain the membership oracle we use for L_p-Adaptation. We conclude this section with results we obtain from L_p-Adaptation.

5.2.1 The model

We use a phase model of coupled electronic clocks to study the dynamics of synchronization (Jörg et al., 2015; Wetzel et al., 2017). A system of N coupled oscillators can be described as follows:

$$\dot{\theta}_k(t) = \omega + \frac{K}{n_k}\sum_{l=1}^{N} c_{kl}\int_0^{\infty} du\, p(u)\, h(\theta_l(t-\tau-u) - \theta_k(t-u)), \tag{5.5}$$

where $\theta_k(t)$, with $k = 1, 2, \cdots, N$, corresponds to the phases of the individual oscillators at time t with the intrinsic frequency ω, the coupling strength K between the oscillators, and a 2π-periodic coupling function h. $\dot{\theta}_k(t)$ correspond to the instantaneous frequencies. The coupling matrix $\mathbb{C} = (c_{kl})$ with $c_{kl} \in \{0, 1\}$ describes the connections between the oscillators: if oscillator k has a connection from oscillator l, $c_{kl} = 1$. The total number of incoming connections of oscillator k is $n_k = \sum_{l=1}^{N} c_{kl}$. The filter kernel $p(u)$ describes the properties of the low-pass filter, and τ is the communication delay. We investigate homogeneous systems with constant and equal delays and identical intrinsic frequencies.

5.2.2 The Kuramoto order parameter

The real part $z_K(t)$ of the complex-valued Kuramoto order parameter (Kuramoto, 1975) is a measure of synchrony of a system of N coupled phase

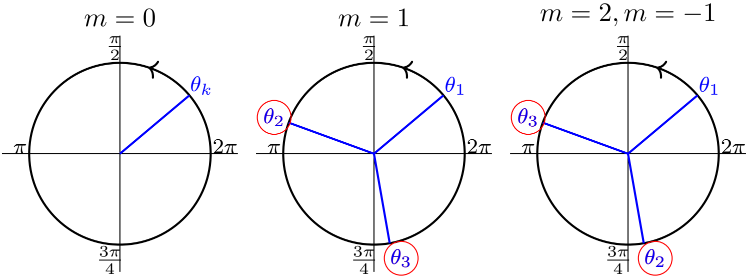


Figure 5.4: In a system of three mutually coupled oscillators, there are three different m-twist solutions with $z(t, m) = 1$. An m-twist solution is a synchronized state where all oscillators have a common frequency with constant phase relations between neighboring oscillators. In the unit circle, the state of oscillator k can be represented by its phase $\theta_k(t)$.

oscillators:

$$z_K(t)e^{i\psi(t)} = \frac{1}{N}\sum_{k=1}^{N} e^{i\theta_k(t)}, \tag{5.6}$$

where $\theta_k(t)$ with $k = 1, 2, \cdots, N$, correspond to the phases of the individual oscillators, $z_K(t) \in [0, 1]$ measures the coherence of the oscillator population, and $\psi(t)$ is the average phase (Acebrón et al., 2005). In synchronized states with no phase differences, the oscillators are completely synchronized and $z_K(t) = 1$ (Wetzel, 2012). Other synchronized states for which $z_K(t) = 0$ (Wetzel, 2012), can exist, see Eq. (5.6). Synchronized states are for example m-twist solutions with $m > 0$ (Peruani et al., 2010; Wiley et al., 2006), where all oscillators share a common frequency and where the phases are equally arranged in $[0, 2\pi)$ (the phase difference is identical between all adjacent pairs of oscillators) (Wetzel, 2012). We now consider such m-twist solutions only for systems of oscillators that have bidirectional nearest neighbor interactions in a 1D ring (periodic boundary conditions) configuration. Figure 5.4 visualizes the existing m-twist states in a system of three coupled oscillators. Since we also want to study m-twist solutions, we use a generalization of the Kuramoto order parameter that can distinguish between the different twist-states:

$$z(t,m)e^{i\psi(t)} = \frac{1}{N}\sum_{k=1}^{N} e^{i\left[\theta_k(t)+k\frac{2\pi m}{N}\right]}, \tag{5.7}$$

where m is the twist number, θ_k with $k = 1, 2, \cdots, N$, correspond to the phases of the individual oscillators, and N is the total number of oscillators.

For in-phase synchronization (0-twist), where all frequencies and phases are identical,

$$z(t,m)e^{i\psi(t)} = \frac{1}{N}\sum_{k=1}^{N} e^{i\theta_k(t)} \tag{5.8}$$

recovers the Kuramoto order parameter $z_K(t)$. We can use Eq. (5.7) to determine how close a set of phases is to an m-twist solution. In Figure 5.4 we show snapshots of the phases in the three possible m-twist configurations for $N = 3$ oscillators. Note that the 2-twist corresponds to the -1-twist, i.e., the order of the oscillators is reversed, see Figure 5.4. The integer $m \in 0, 1, \cdots, N-1$ denotes how many multiple of 2π accumulate when summing over all phase differences of adjacent oscillators in the ring of N oscillators (Wetzel, 2012).

5.2.3 Space transformation

Points on the main diagonal in phase space describe the states, where all oscillators have identical phases (0-twist). All other m-twist solutions are represented by parallel lines to this diagonal, see Figure 5.5a. We are interested in the volume of the basin of attraction, which can be a measure of its stability (Menck et al., 2013). With L_p-Adaptation a volume approximation can be obtained, but in order to allow L_p-Adaptation to converge, we look at a finite subspace of the phase space $[0, 2\pi]^N$. Since the coupling function h is 2π-periodic, this is sufficient. m-twist states are characterized by specific points in phase space (which is a snapshot at $t = t_1$ when the system is synchronized):

$$\theta_{k+1} = k\frac{2\pi m}{N}, \tag{5.9}$$

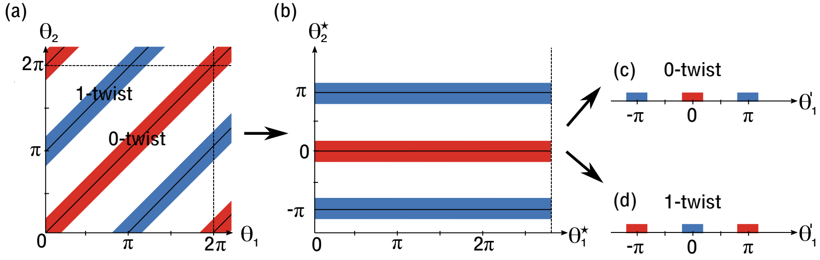


Figure 5.5: Transformation of phase space illustrated in 2D. (a) All m-twist solutions are parallel to the main diagonal in the phase space spanned by the oscillator phases $\theta_1, \cdots, \theta_n$. (b) The first oscillator θ_1 is rotated onto the main diagonal of the phase space. A point in this rotated phase space is called $\theta^\star$. We do not consider perturbations that affect all phases the same, thus we fix $\theta_1^\star = 0$ in the rotated phase space. We translate the reduced rotated phase space such that its origin coincides with the point that characterizes the m-twist to be studied: in 2D this can be (c) a point that characterizes the 0-twist or (d) a point that characterizes the 1-twist. A point in this transformed phase space is called θ'.

where $\frac{2\pi m}{N}$ defines the phase differences between neighboring oscillators.

We first rotate the phase space such that the phase of the first oscillator θ_1 is rotated onto the main diagonal of the N-dimensional phase space, see Figure 5.5b. We call a point in this rotated phase space $\theta^\star = (\theta_1^\star, \theta_2^\star, \cdots, \theta_N^\star)$. A perturbation in $\theta_1^\star$ direction means that the phases of all N oscillators are changed by the same amount. We now reduce the rotated phase space by fixing $\theta_1^\star = 0$ because we do not consider perturbations that affect all phases the same. We then translate the reduced rotated phase space such that its origin coincides with the specific point of the studied m-twist, see Figures 5.5c and 5.5d and consider perturbations around this origin in transformed coordinates. We call a point in this transformed phase space $\theta' = ({\theta'}_1, \cdots, {\theta'}_{N-1})$. This transformation of the phase space gives us a basin of attraction that is more connected and thus easier to approximate with L_p-Adaptation.

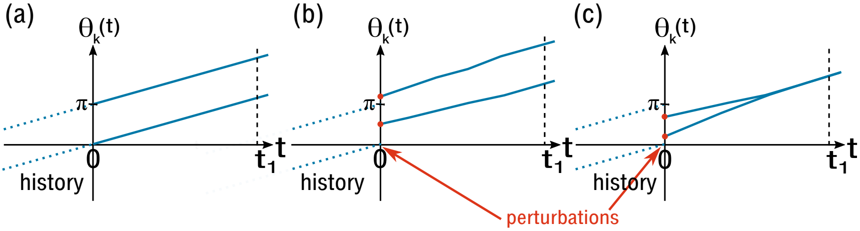


Figure 5.6: Schematic trace plots of the phases of the oscillators for $n = 2$. The oscillators are in a frequency synchronous state (1-twist) for $t < 0$. (a) If the phases are not perturbed, the phases are in the same synchronous state at $t = t_1$. (b,c) The phases are perturbed at $t = 0$. (b) At $t = t_1$ the phases are back in the same synchronized state, thus the oracle classifies this perturbation as a feasible point. (c) At $t = t_1$ the phases are not back in the synchronized state, the oracle classifies this perturbation as a infeasible point.

5.2.4 MEMBERSHIP ORACLE

L_p-Adaptation operates on the rotated and reduced phase space. We study the influence of different parameters on the basin of attraction of a stable synchronized state. Linear stability analysis is used to find those states and to approximate the simulation time. The membership oracle f decides whether a given perturbation vector $\theta' \in \mathbb{R}^{N-1}$ leads back to the synchronized state the system was in before. Then the perturbation belongs to the basin of attraction (the feasible region A). To do so, the phase model is simulated for the original phases θ_k, for $k = 1, \cdots, N$. We assume that the simulation time t_1 depends on the decay of the slowest perturbation mode. For an illustration of the system with two oscillators and the 1-twist as synchronized state, see Figure 5.6. Assuming we investigate the basin of attraction of an m-twist state that is stable, then, if the system is not perturbed at $t = 0$, it is still at the synchronized state at $t = t_1$, see Figure 5.6a, i.e., the oracle classifies a perturbation of $\mathbf{0}$ as a feasible point. The perturbation used to perturb the system at $t = 0$ is classified as a feasible point if at $t = t_1$ the phases are in the prior synchronized state again (Figure 5.6b). The perturbation is classified as infeasible, if the phases did not reach the same synchronized state at $t = t_1$ (Figure 5.6c). To decide whether the synchronized state (m-twist) was reached, we use the order parameter $z(t_1, m)$, see Equation (5.7).

5.2.5 Results

We want to study how the basins of attraction of stable synchronized states depend on different design parameter of a DPLL network. We are for example interested in the effects of the cutoff frequency F_c, the communication delay τ, and the coupling strength K. In low dimensions (numbers of oscillators), this can be studied by exhaustively sampling the phase space or its discretization. An example of how the basins of attraction change with those parameters for $N = 3$ and intrinsic frequency $F = 1$Hz is shown in Figure 5.7. Changing one parameter at a time, we observe that increasing K can lead to an increase in size of the basin of attraction. The delay τ influences whether an m-twist synchronized state will be stable or unstable, a larger τ leads to smaller decay rates.

Exhaustively sampling the space becomes computationally infeasible for an increasing number of oscillators. L_p-Adaptation can nevertheless be used to obtain a volume approximation of the basins of attraction. To study the cutoff frequency's influence on the size of the basin of attraction, see Figure 5.8, we run L_p-Adaptation with a fixed schedule of increasing hitting probabilities $\mathbf{P} = (0.35, 0.55, 0.75, 0.9)$ and a fixed initial mean $\mathbf{m}_0$ located at $\theta' = (0, 0)$, hence the feasible $\mathbf{m}_0$ is the start value for all ten runs. The higher the hitting probability, the smaller the volume approximation becomes, which mean that the shape of the feasible region is not a simple ellipsoid.

For $F_c = 0.1, 0.5, 1$, and 10, Figure 5.9 shows the corresponding transformed phase spaces with points (θ'_1, θ'_2). For each hitting probability, L_p-Adaptation obtains a transformation matrix $\mathbf{C}$. The ellipsoids shown in Figure 5.9 correspond to the averaged transformation matrices of ten runs. The DPLL system with a filter cutoff frequency of 0.5 has the largest volume of its basin of attraction among the studied systems. Comparing the ratios of the volumes obtained with a target hitting probability of 0.35 and of 0.9, we see that the basin of attraction is also most scattered for

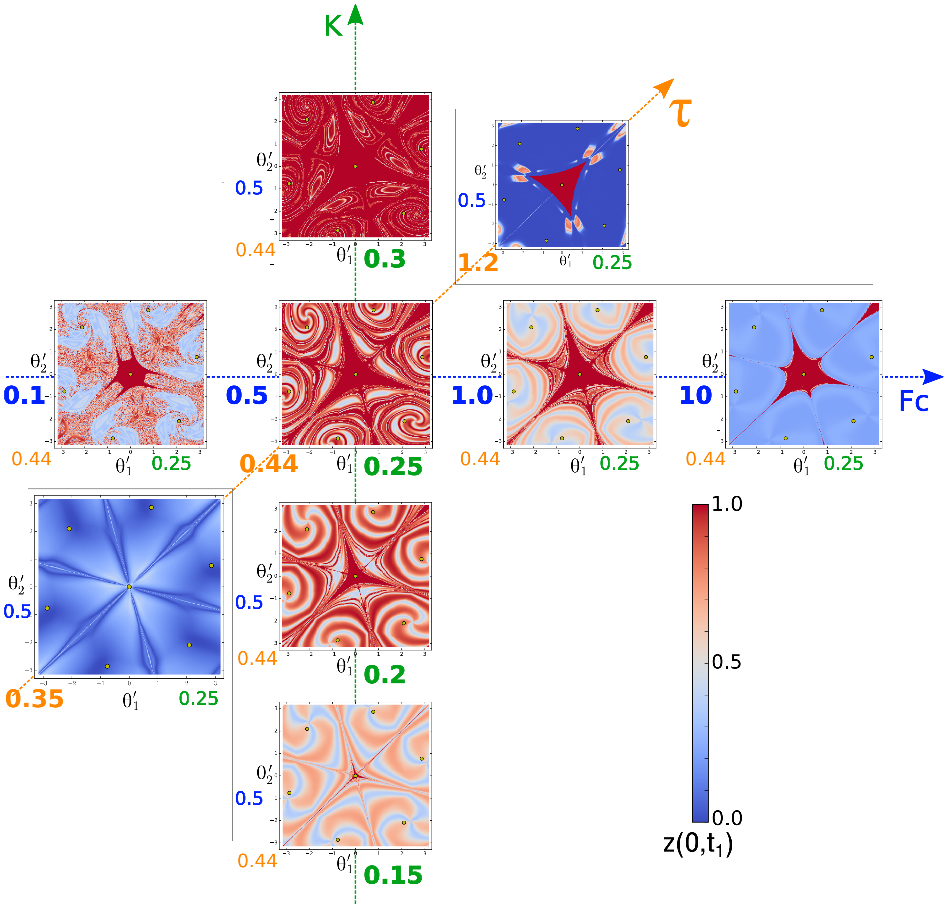


Figure 5.7: Basin of attraction depending on the parameters of a DPLL network of $N = 3$ coupled oscillators. The color code shows the value of the order parameter $z(0, t_1)$ at the end of the simulation. We show examples for changing coupling strength K (green), changing delay τ (orange), and changing filter cutoff frequency F_c (blue). The phase space is transformed, as described in Section 5.2.3. This figure was prepared in collaboration with Lucas Wetzel.

$F_c = 0.5$. We will investigate the influence of other design parameters in the near future.

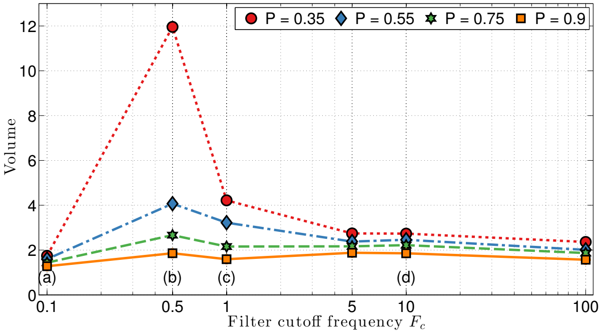


Figure 5.8: The volume of the basin of attraction of the synchronized state is a measure for its robustness. We plot the volume of the basin of attraction of the in-phase synchronized state (0-twist) in a 1D ring of n=3 DPLLs with nearest neighbor coupling, $\tau = 0.44$, $K = 0.25$, *and* $\omega = 1.215$. *A perturbation* θ' *is feasible if* $z \geq 0.995$. *Different lines correspond to different target hitting probabilities in* L_p*-Adaptation. For* $F_c = 0.1, 0.5, 1,$ *and* 10, *the basins of attraction are visualized in Figure 5.9.*

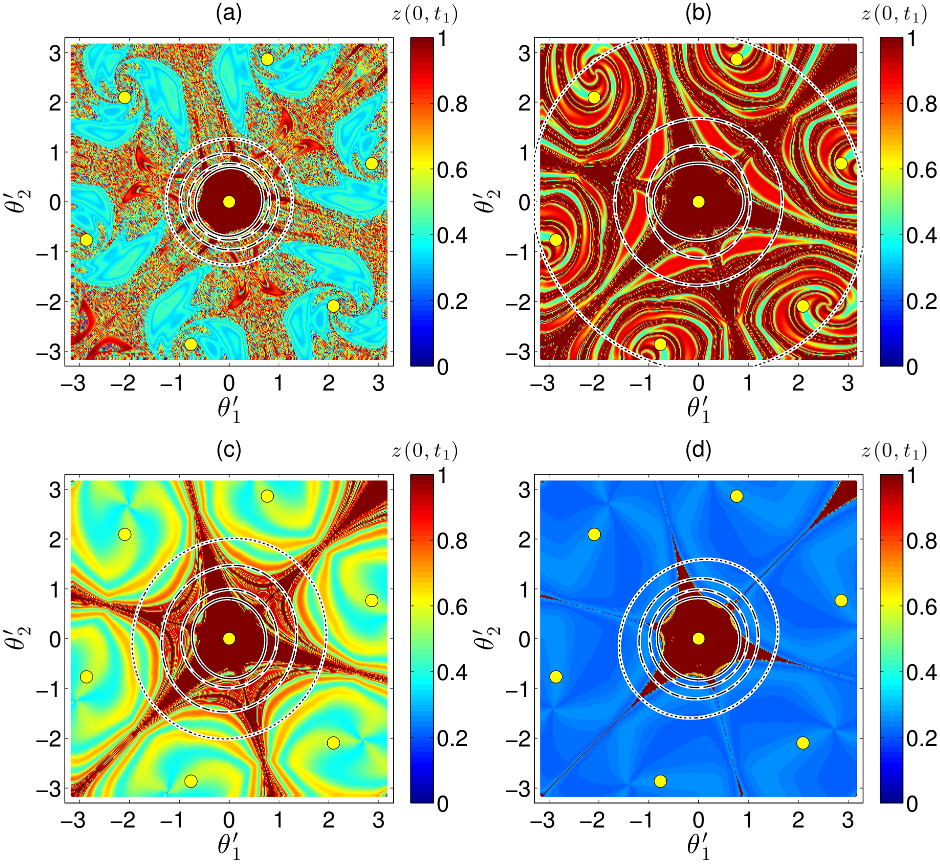


Figure 5.9: We study the basin of attraction of the in-phase synchronized state (0-twist) in a 1D ring of n=3 DPLLs with nearest neighbor coupling. The communication delay $\tau = 0.44s$*, and the coupling strength* $K = 0.25$*. The color code shows the order parameter* $z(0,t)$*, where* $z = 1$ *if the DPLLs are in-phase synchronized at time t, i.e., the perturbations decayed. A point in the transformed phase space* (θ'_1, θ'_2) *is feasible if* $z \geq 0.995$*. Ellipsoids (largest to smallest) show the results obtained by* L_p*-Adaptation with target hitting probabilities of* $0.35, 0.55, 0.75,$ *and* 0.9*, respectively. The systems differ in the filter cutoff frequency* F_c*: (a)* $F_c = 0.1$ *Hz, (b)* $F_c = 0.5$ *Hz, (c)* $F_c = 1$ *Hz, and (d)* $F_c = 10$ *Hz. Yellow dots show the different possible m-twist states. The brute-force results used for this figure were produced by Lucas Wetzel.*

5.3 Comparing Robustness of Two Bacterial Two Component Systems

As a third real-world example, we consider a network model from systems biology: the bacterial two-component system (TCS). The TCS signaling network allows bacteria to sense and respond to environmental changes. The TCS is an evolutionarily conserved stimulus-response mechanism found in all bacterial species, and to a lesser extent also in archaea and eukaryotes such as plants, molds, and yeast.

Stimulus sensing in the TCS relies on a sensor histidine kinase (HK), which auto-phosphorylates and, upon stimulus, passes the phosphate group on to a response regulator (RR) protein. As a response to the stimulus S, the phosphorylated RR regulates the transcription of target genes (Kim and Cho, 2006). TCS occur in nature in different flavors, all sharing the same working principle. Here, we compare two TCS differing in the number of phosphate binding domains on the HK. This includes the most commonly found TCS with two binding domains and an alternative form with four stimulus-binding domains. Following the terminology of Barnes et al. (2011), we call them the "orthodox system" and the "unorthodox system", respectively, as depicted in Figure 5.10.

We compare both systems' ability to robustly achieve different input-output characteristics by tuning their parameters (Barnes et al., 2011).

5.3.1 Ordinary differential equations models

The ordinary differential equation models describing the dynamics of both systems are as follows:

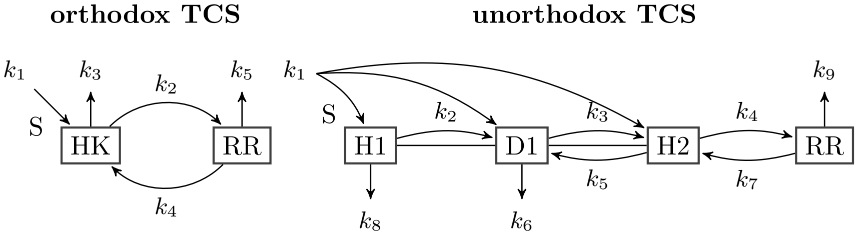


Figure 5.10: Two different bacterial two-component systems (TCS). Reactions involving phosphate groups are represented by arrows with the respective reaction rates indicated. Left: the "orthodox system", in which the histidine kinase (HK) auto-phosphorylates upon sensing a stimulus S *and passes the phosphate group on to the response regulator (RR) protein. The orthodox system has two phosphate binding domains. Right: the "unorthodox system", in which the HK has three binding domains: H1, D1, and H2. Together with the binding domain on the RR, this system has a total of four phosphate binding domains.*

Orthodox system

The model assumes that the total concentrations of the HK and RR proteins sum up to 1, hence:

$$\begin{aligned} HK + HK_p &= 1 \\ RR + RR_p &= 1, \end{aligned}$$

where a subscript p denotes the phosphorylated form of the molecule. This leads to the following model equations for the time dynamics of the concentrations $[\cdot]$ of all involved chemical species:

$$\begin{aligned} \frac{d[HK]}{dt} =& k_2(1-[HK])(1-[RR_p]) + k_3(1-[HK]) - k_4[HK][RR_p] \\ & - k_1[HK][\mathrm{S}] \\ \frac{d[RR_p]}{dt} =& k_2(1-[HK])(1-[RR_p]) - k_4[HK][RR_p] - k_5[RR_p]. \end{aligned}$$

UNORTHODOX SYSTEM

In the unorthodox TCS, HK has three phosphorylation sites: H1, D1, and H2. This leads to eight different phospho-species of HK, $HK_{1\dots 8}$, with the following phosphorylation states (1=phosphorylated, 0=unphosphorylated):

	H1	D1	H2
HK_1	1	1	1
HK_2	0	1	1
HK_3	1	0	1
HK_4	1	1	0
HK_5	0	0	1
HK_6	0	1	0
HK_7	1	0	0
HK_8	0	0	0

Again it is assumed that the total concentrations of HK and RR sum up to 1:

$$\sum_{i=1}^{8} HK_i = 1$$
$$RR + RR_p = 1.$$

This leads to the following model equations for the time dynamics of the concentrations $[\cdot]$ of all involved chemical species:

$$\begin{aligned}\frac{d[HK_1]}{dt} =& k_4[HK_4](1-[RR_p]) + k_6[HK_3] - k_7[HK_1][RR_p] + k_8[HK_2] \\ & - k_1[HK_1][\mathrm{S}] \\ \frac{d[HK_2]}{dt} =& k_4[HK_6](1-[RR_p]) + k_6[HK_5] - k_7[HK_2][RR_p] - k_8[HK_2] \\ & + k_1[HK_1][\mathrm{S}] - k_2[HK_2]\end{aligned}$$

$$\begin{aligned}
\frac{d[HK_3]}{dt} =& -k_3[HK_3] + k_4[HK_7](1-[RR_p]) + k_5[HK_4] - k_6[HK_3] \\
& - k_7[HK_3][RR_p] + k_8[HK_5] - k_1[HK_3][\mathrm{S}] + k_2[HK_2] \\
\frac{d[HK_4]}{dt} =& k_3[HK_3] - k_4[HK_4][1-RR_p] - k_5[HK_4] + k_6[HK_7] \\
& + k_7[HK_1][RR_p] + k_8[HK_6] - k_1[HK_4][\mathrm{S}] \\
\frac{d[HK_5]}{dt} =& -k_3[HK_5] + k_4(1-\sum_{i=1}^{7}[HK_i])(1-[RR_p]) + k_5[HK_6] \\
& - k_6[HK_5] - k_7[HK_5][RR_p] - k_8[HK_5] + k_1[HK_3][\mathrm{S}] \\
\frac{d[HK_6]}{dt} =& k_3[HK_5] - k_2[HK_6] - k_4[HK_6](1-[RR_p]) - k_5[HK_6] \\
& + k_6(1-\sum_{i=1}^{7}[HK_i]) + k_7[HK_2][RR_p] - k_8[HK_6] + k_1[HK_4][\mathrm{S}] \\
\frac{d[HK_7]}{dt} =& k_2[HK_6] - k_4[HK_7](1-[RR_p]) - k_6[HK_7] + k_7[HK_3][RR_p] \\
& + k_8(1-\sum_{i=1}^{7}[HK_i]) - k_1[HK_7][\mathrm{S}] \\
\frac{d[RR_p]}{dt} =& k_4(1-[RR_p])(1-[HK_1]-[HK_2]-[HK_3]-[HK_5]) \\
& - k_7[RR_p]([HK_1]+[HK_2]+[HK_3]+[HK_5]) - k_9[RR_p].
\end{aligned}$$

Note that in the original paper (Barnes et al., 2011) the equation for $\frac{d[HK_5]}{dt}$ has a typo. The first term should be $-k_3[HK_5]$, as shown above, and not $-k_3[HK_3]$.

5.3.2 Membership oracles

The membership oracle numerically solves the dynamic equations in order to decide whether a given vector $\mathbf{x} \in \mathbb{R}^n$ ($n = 5$ for the orthodox system, $n = 9$ for the unorthodox system) of kinetic rate constants leads to a reaction network that fulfills the specifications, or not. All simulations are done in MATLAB 8.2.0 (R2013b) using the "Systems Biology Toolbox 2" and the "SBPD Extension Package" (Schmidt and Jirstrand, 2006) with a time step size of 0.001 until final time 10. The specifications are given in terms of the input/output behavior of the network. The output is the concentration of RR_p in response to a specific input signal S over time points $T = \{t_k\}_{1\leq k\leq N}$ where $t_1 = 0$, $t_N = 10$, and $t_{k+1} = t_k + 0.001$. All variables are in the range $[0, 1000]$. We consider four different cases, corresponding to different network design goals and hence different membership oracles, as described below.

5.3.2.1 FAST RESPONSE

The "fast response" case aims to design a network that rapidly follows a sudden change of the input, see Figure 5.11a. The following input signal is considered, which has two sudden changes:

$$[\mathrm{S}] = \begin{cases} 1, & \text{if } t_k \in [2, 4] \\ 0, & \text{else.} \end{cases}$$

Desired output: The network fulfills the specifications if the maximum response of the output is reached within 0.1 time units after the pulse starts, and the minimum no more than 0.1 time units after the pulse ends. The specifications hence are:

$$\begin{aligned} t_{\max} &= \arg\max_k [RR_p](t_k) \\ t_{\min} &= \arg\min_k [RR_p](t_k > t_{\max}) \end{aligned}$$

where $\mathbf{x}$ is feasible iff $0 \leq t_{\max} - 2 \leq 0.1$ and $0 \leq t_{\min} - 4 \leq 0.1$.

5.3.2.2 STEADY OUTPUT

The "steady output" case aims to design a network that produces a stationary output upon a constant input signal of the same magnitude, see Fig-

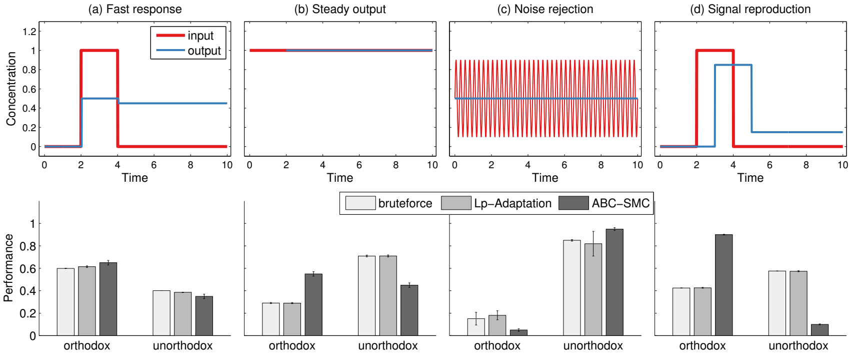


Figure 5.11: Design centering for four different desired input-output behaviors of the two TCS. Top row: Chemical concentrations as a function of time for the input signal S *(red line) and the output signal RR (blue line). For each case, one or several example outputs are shown that fulfill the specifications. Bottom row: The corresponding performances of the orthodox and unorthodox TCS as estimated by three different methods: brute-force sampling, L_p-Adaptation, and approximate Bayesian computation based on sequential Monte Carlo (ABC-SMC) (Barnes et al., 2011). Performance is quantified by the posterior probability for each model (ABC-SMC) or by the normalized volume (L_p-Adaptation and brute-force) (Hafner et al., 2009). (a) The "fast response" case requires the network output to react to sudden changes in the input with no more than 0.1 time units delay. (b) The "steady output" case requires the output to remain constant at the level of the input at most 2 time units after the input started. (c) The "noise rejection" case requires the output to remain constant at the mean of a rapidly fluctuating input. (d) The "signal reproduction" case requires the output to reproduce the shape of the input signal with no more than 20% magnitude error and no more than 1 time unit delay.*

ure 5.11b. We consider the constant input

$$[\mathrm{S}] = 1.$$

Desired output: The network fulfills the specifications if the output is steady at the same level at most 2 time units after the input appeared. The specifications are:

$$\epsilon = \frac{N_B}{N_2} \sum_{\substack{k \\ t_k \geq 2}} \left([RR_p](t_k) - 1\right)^2,$$

where N_2 is the number of elements in $T = \{t_k\}, t_k \geq 2$, and $N_B = 160$ is the number of elements used by Barnes (Barnes et al., 2011). The vector $\mathbf{x}$ is feasible iff $\epsilon \leq 0.01$.

5.3.2.3 NOISE REJECTION

The "noise rejection" case aims to design a network that removes high frequencies from the input signal, see Figure 5.11c. We consider the input signal:

$$[\mathrm{S}] = 0.5 + 0.4 \cdot \sin(8\pi t),$$

consisting of a constant (DC) part and an oscillatory (AC) signal. Desired output: The network fulfills the specifications if the output signal rejects the oscillatory (AC) part after at most 2 time units. The specifications hence are:

$$\epsilon = \frac{N_B}{N_2} \sum_{\substack{k \\ t_k \geq 2}} \left([RR_p](t_k) - 0.5\right)^2$$

with the same definitions as above. The vector $\mathbf{x}$ is feasible iff $\epsilon \leq 0.3$.

5.3.2.4 SIGNAL REPRODUCTION

The “signal reproduction” case aims to design a network where the output reproduces the input within a certain tolerance, see Figure 5.11d. We again consider as input the square pulse:

$$[\mathrm{S}] = \begin{cases} 1, & \text{if } t_k \in [2, 4] \\ 0, & \text{else.} \end{cases}$$

Desired output: The network fulfills the specifications if the output signal rises above 0.8 within 1 time unit after the pulse starts, and drops to below 0.2 within 1 time unit after the pulse ends. The specifications hence are:

$$\begin{aligned} t_{\max} &= \arg\max_k [RR_p](t_k) \\ t_{\min} &= \arg\min_k [RR_p](t_k > t_{\max}) \end{aligned}$$

where $\mathbf{x}$ is feasible iff $0 \leq t_{\max} - 2 \leq 1$ and $0 \leq t_{\min} - 4 \leq 1$ and $(1 - \max([RR_p])) < 0.2$ and $\min([RR_p](t_k > t_{\max})) < 0.2$.

5.3.3 RESULTS

Figure 5.11 shows four different desired input-output behaviors, as detailed above, along with the respective design specifications. In the lower panels, the corresponding performances of the orthodox and the unorthodox systems are compared. Performance is quantified by how well a system achieves the task compared to the other system, as computed with three different approaches: brute-force sampling, L_p-Adaptation, and approximate Bayesian computation based on sequential Monte Carlo (ABC-SMC) (Barnes et al., 2011). ABC-SMC compares the posterior probabilities of the two systems, where the posterior probability in ABC-SMC is the fraction of accepted samples for the given system compared to the total number of accepted samples for both systems. L_p-Adaptation and brute-force sampling compare the normalized volumes of the systems, see Section 2.2.2. The normalized volume is the fraction of the proposal volume for the given system compared to the total volume for both systems. In order to compare models of different dimensionality, volumes are normalized as $\sqrt[n]{\mathrm{vol}}$ (Hafner et al., 2009). The error bars for the brute-force sampling are obtained by bootstrapping ten samples of size $5 \cdot 10^6$ for the orthodox and of size $9 \cdot 10^6$ for the unorthodox system. The error bars for L_p-Adaptation show the standard deviation across ten runs, each with a sample size of $5 \cdot 10^4$ for the orthodox and $9 \cdot 10^4$ for the unorthodox system. The error bars for ABC-SMC show the variability in the marginal model posteriors over three runs (Barnes et al., 2011).

The "fast response" case requires the network to rapidly react to abrupt changes in the input stimulus S (Figure 5.11a). Here, all three approaches reach the same conclusion: both systems are able to achieve the desired input-output response, but the orthodox system outperforms the unorthodox system. Faithfully reproducing a constant input stimulus is the goal of the "steady output" case (Figure 5.11b). Here, L_p-Adaptation agrees with brute-force sampling that the unorthodox system can more robustly produce this behavior, whereas ABC-SMC prefers the orthodox system. The "noise rejection" case aims to design a network that reproduces a constant signal while rejecting high-frequency fluctuations about it (Figure 5.11c). This is much more robustly realized in the unorthodox system, as agreed by all three methods. Finally, the "signal reproduction" case is to design a TCS network where the output signal reproduces the shape of the input stimulus within specified tolerances (Figure 5.11d). Again, L_p-Adaptation agrees with brute-force sampling that the unorthodox system better achieves this behavior, whereas ABC-SMC considers the orthodox system better.

L_p-Adaptation can be used for model selection, since the volume of the feasible region is a measure for the robustness of the system. Our results agree with the exhaustive brute-force approach, but only require a fraction of the samples. Figure 5.12 shows the evolution of the estimated normalized volumes versus the number of function evaluations for the orthodox models in the top row and for the unorthodox models in the bottom row. The normalized volume is a measure for the robustness of the system, quantifying the subspace of rate constants for which the system fulfills the specifications. The volumes of the axes-aligned bounding box (green squares) and the Loewner ellipsoid (blue circles) of all feasible samples

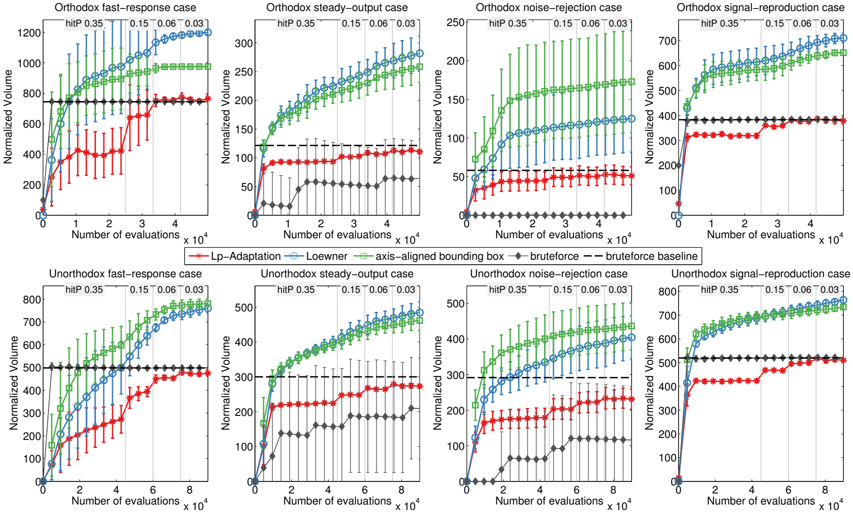


Figure 5.12: Averages and standard deviations of the normalized volume ($\sqrt[n]{\mathrm{vol}}$ (Hafner et al., 2009)) estimation of the feasible region over ten independent runs of L_p-Adaptation. We show the orthodox TCS (top panel) and the unorthodox TCS (bottom panel) using an L_2-ball proposal. The results for the L_1 proposal are visually indistinguishable and hence omitted. For comparison, the ground-truth baseline obtained by exhaustive brute-force sampling is shown as a dashed black line. Brute-force sampling using the same number of function evaluations as L_p-Adaptation is shown in grey. The upper bounds obtained from Loewner ellipsoids and axes-aligned bounding boxes of the L_p-Adaptation samples are in blue and green, respectively. Their large variance indicates that the feasible region is non-convex or disconnected.

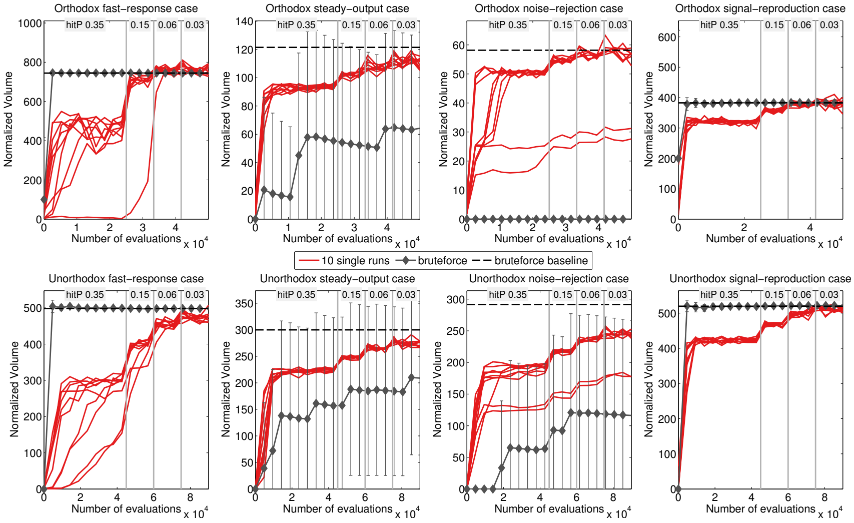


Figure 5.13: Normalized volume ($\sqrt[n]{\mathrm{vol}}$) estimation of the feasible regions of all four cases for both TCS models. The 10 independent runs of L_p-Adaptation using an L_2-ball proposal are shown as individual red lines in each case. The ground-truth baseline obtained by exhaustive brute-force sampling is shown as a dashed black line. Brute-force sampling using the same number of function evaluations as L_p-Adaptation is shown with gray diamonds and error bars (standard deviation over 10 brute-force runs). The schedule of reducing the target hitting probability is shown at the top of each plot.

are shown as upper bounds. An L_2-ball is used as proposal distribution. The hitting probability is dynamically reduced according to the schedule shown on top of each plot. The dashed black line shows the baseline result obtained by exhaustive brute-force sampling the orthodox system $7.5 \cdot 10^6$ times and the unorthodox system $1.5 \cdot 10^7$ times. The results of all individual runs and all eight cases are shown in Figure 5.13. L_p-Adaptation converges toward the baseline in all cases. In the noise-rejection cases, however, the different runs cover different parts of the non-convex feasible region and some of them do not converge to the baseline. This is manifested in the larger error bars in the unorthodox noise-rejection case in Figure 5.12, and clearly visible in the two families of curves in Figure 5.13. Nevertheless L_p-Adaptation performs much better than brute-force sampling using the same number of function evaluations (grey diamonds) and is able to sample the feasible region much more efficiently. Cases where brute-force sampling reaches the baseline faster are indicative of a feasible region that fills almost the entire space, as also confirmed by the larger normalized volumes in these cases (fast-response and signal-reproduction cases). The normalized volume of the entire parameter space is 1000 in this example. When the feasible region is significantly smaller than the whole space, L_p-Adaptation performs better and more reliably than brute-force sampling (steady-output and noise-rejection cases). The orthodox noise-rejection case has a particularly small feasible region. In this case, none of the 10 brute-force sampling runs finds any feasible solution, whereas L_p-Adaptation reaches the baseline in 8 out of 10 runs.

MARGINAL DISTRIBUTIONS

We provide additional figures of the marginal and pairwise distributions of parameters for the TCS in order to assess the space exploration of L_p-Adaptation. For all pairwise distribution plots, we used a less biased version of the set of all feasible points obtained in all runs by omitting the samples obtained during the burn-in period (the phase where the Markov-chain did not reach a high-probability region yet) and after this, we thin our samples by only using every 5th feasible point found. Figure 5.14 shows the joint pairwise distributions and the marginal densities of the feasible points of the unorthodox system for the noise-rejection case. Com-

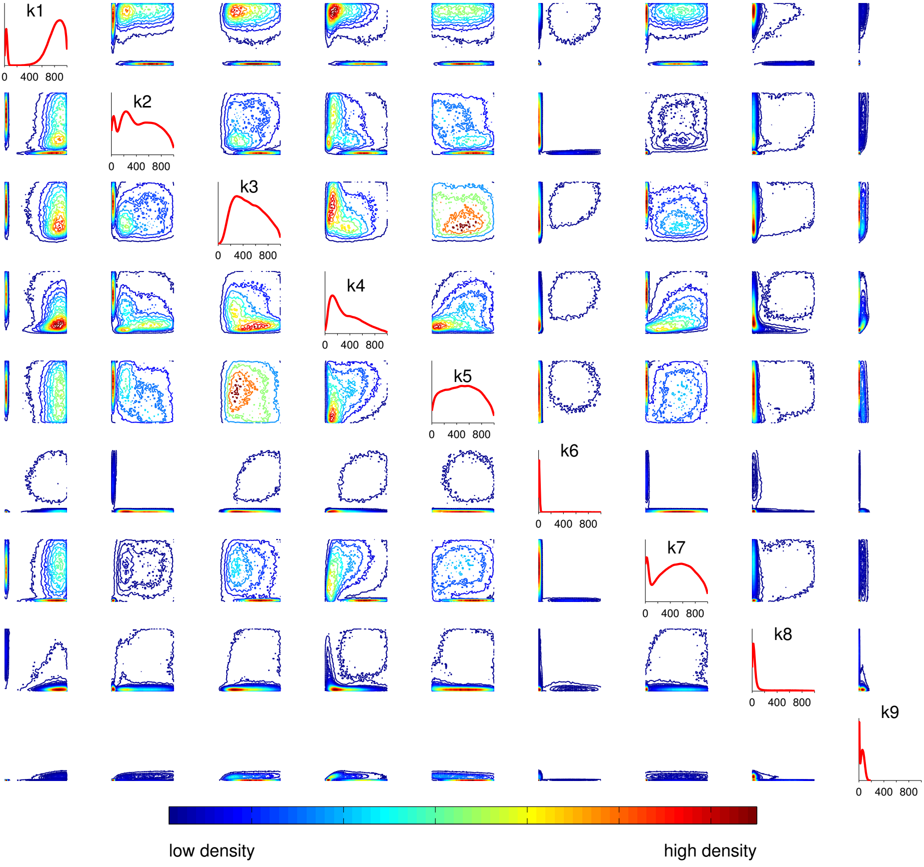


Figure 5.14: Marginal densities (diagonal) and joint pairwise distributions of feasible points for the unorthodox TCS in the noise-rejection case. Colors show the density of the feasible points obtained.

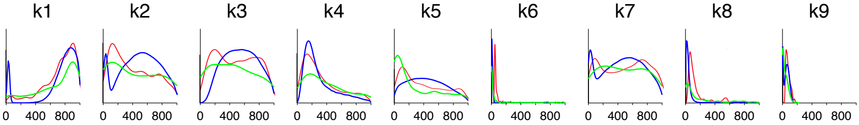


Figure 5.15: Comparison of the marginal densities for each of the nine design parameters of the unorthodox TCS in the noise-rejection case. Red: marginals obtained by approximate Bayesian computation based on sequential Monte Carlo (ABC-SMC) (Barnes et al., 2011); blue: marginals obtained by L_p-Adaptation; green: marginals from points obtained by uniformly sampling the entire parameter space (brute-force sampling).

paring the distributions with those shown in Barnes' appendix, Figure 5 of Ref. (Barnes et al., 2011), we see that our method explores the parameter space more comprehensively.

Figure 5.15 compares the marginal distributions obtained by ABC-SMC (Barnes et al., 2011), L_p-Adaptation, and brute-force sampling. Using brute-force sampling, 228 feasible points were found by sampling $1.5 \cdot 10^7$ points. L_p-Adaptation finds 159,140 feasible points by sampling $9 \cdot 10^5$ points in 10 runs, each with a different starting point. This indicates that L_p-Adaptation provides an efficient way of exploring feasible regions that are small compared to the total space. Using only 6% of the function evaluations of brute-force sampling, the number of feasible points found by L_p-Adaptation is almost 700-times higher than that of brute-force sampling, amounting to five orders of magnitude better sampling efficiency.

Figure 5.16 shows the joint pairwise distributions and the marginal densities of the feasible points of the orthodox system for the signal-reproduction case. This is the same case also shown by Barnes et al. (2011). Figure 5.17 again compares the marginal densities obtained by the three different methods. The conclusions are commensurate with those from the orthodox system.

For the sake of completeness we show the pairwise distributions and the marginal densities of the remaining cases in Appendix C.

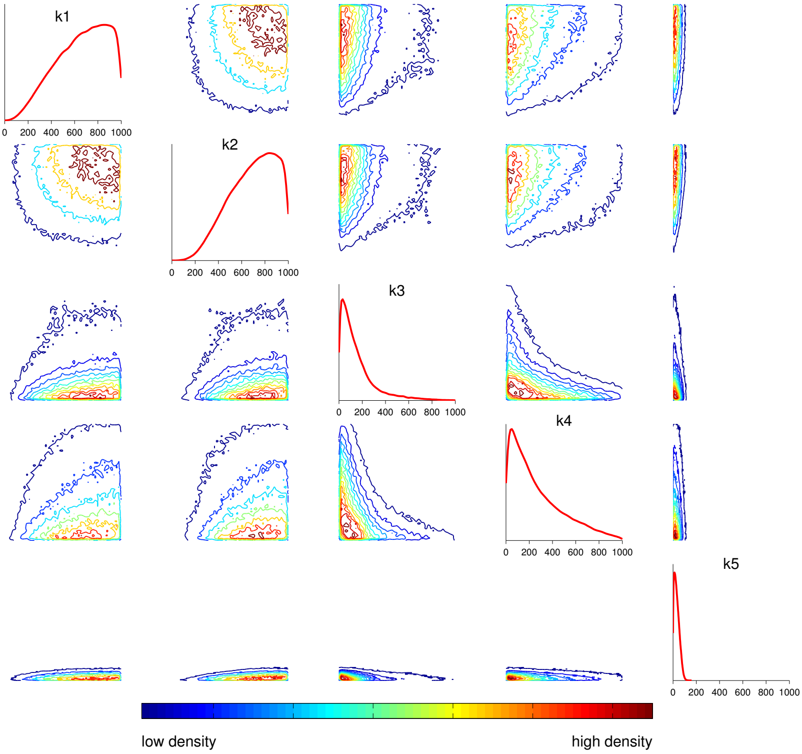


Figure 5.16: Marginal densities (diagonal) and joint pairwise distributions of feasible points of the orthodox TCS in the signal-reproduction case. Colors show the density of the feasible points obtained.

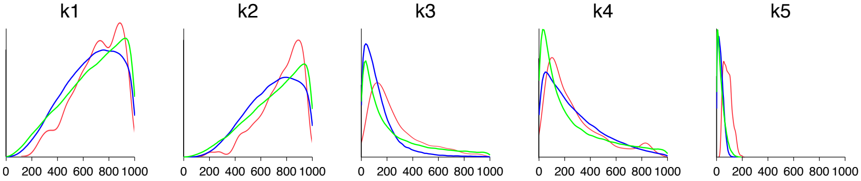


Figure 5.17: Comparison of the marginal densities for each of the five design parameters of the orthodox TCS in the signal-reproduction case. Red: marginals obtained by approximate Bayesian computation based on sequential Monte Carlo (ABC-SMC) (Barnes et al., 2011); blue: marginals obtained by L_p-Adaptation; green: marginals from points obtained by uniformly sampling the entire parameter space (brute-force sampling).

5.4 Conclusion

We presented the application of L_p-Adaptation in three examples. In most cases tested here, L_p-Adaptation produced approximations comparable to exhaustive sampling, albeit at a fraction of the computational cost. In the examples where the feasible region was small compared to the whole design space, orders of magnitude fewer design trials were needed by L_p-Adaptation than by exhaustive sampling. Taken together, these results show that L_p-Adaptation produces results that agree with the brute-force baseline, albeit at a much lower computational cost. This is true for both design centering and volume estimation.

In many cases, L_p-Adaptation also produced better-quality results than previous approaches. In the real-world example of the switched capacitor filter, for example, the design center found had a better robustness than the ones previously used as a benchmark. Furthermore, in all four biological network cases, L_p-Adaptation selected the same system as brute-force sampling, whereas previously published results deviated in two cases.

We therefore expect L_p-Adaptation to be of practical use, also because it is effectively parameter-free. All algorithm parameters have default settings that only need to be changed in exceptional cases. All results presented here were obtained using the default settings. The only requirement for using L_p-Adaptation is that a feasible starting point is known. This also makes L_p-Adaptation an ideal candidate for being used as a base sampler in other parameter exploration methods, which approximate the feasible region by concatenations of ellipsoids (Zamora-Sillero et al., 2011) or by approximate posterior distributions through the ABC methodology (Toni et al., 2009; Barnes et al., 2011).

CHAPTER

SIX

Conclusions and Future Work

We have presented a general framework that unites design centering and volume approximation. To efficiently explore a feasible region, the proposal distribution is dynamically adapted based on previous samples. The feasible region is defined by specifications that the design is required to fulfill. The design center is robust against fluctuations or perturbations in the design parameters and the specifications, with robustness quantified by the volume of the feasible region. The volume of the feasible region is a proxy for the number of feasible designs that exist, and hence provides an intuitive robustness measure. Both design centering and volume estimation are hard computational problems, which have so far been considered separately. To our knowledge, L_p-Adaptation is the first algorithm to provide approximate solutions to both problems simultaneously and hence unite them under a single framework.

We illustrated L_p-Adaptation in several 2D examples and tested it on L_p-balls in up to 90D as feasible regions with known ground truth. Finally, we have shown the applicability of L_p-Adaptation in three diverse real-world examples, indicating the broad range of possible applications.

Running L_p-Adaptation for problems with L_p-balls as feasible regions, we made interesting observations: If the p-norm of the proposal and the feasible region match, the proposal is more explorative the more the feasible region is stretched, i.e., the higher its condition number is. We think that an explanation for this could be that with higher condition numbers, more mass of the feasible region is distributed in fewer dimensions and thus easier to approximate for a proposal of same shape. To fully understand all of the L_p-balls behavior, further theoretical investigations would be needed. A better understanding of the behavior could then be used for a guided optimization of those algorithmic parameters that could directly be influenced by the proposal's shape (e.g. update of the transformation matrix and the proposal's mean).

In our studies we see that with our default algorithmic parameters, using either the L_1- or the L_2-ball as a proposal is at least as good as using the shape of the underlying feasible region. For unknown feasible regions we therefore suggest to use the L_2-ball as a default proposal.

A critical feature of L_p-Adaptation in its present form is the sequence of hitting probabilities that is used to steer the process between finding a robust design center and estimating the feasible volume. Successively decreasing the hitting probability allows more accurate volume estimation, yet renders an infeasible design center increasingly likely. In practice, we thus recommend doing multiple L_p-Adaptation runs from different starting points found by initial optimization or brute-force sampling, and to check the quality of the resulting design center and volume estimates. Furthermore, one can also extend the principal idea of learning position, scale, and orientation of the L_p-balls to other convex or quasi-convex bodies, such as closed convex polytopes, that better capture the shape of the feasible points derived from L_p-Adaptation. The only requirement for these alternative (quasi-)convex bodies is the ability to efficiently sample uniform points from them.

In the future, performance improvements could be realized by adding the possibility of fixing the transformation matrix $\mathbf{C}$ to be diagonal as done for CMA-ES (Ros and Hansen, 2008). For feasible regions whose main directions are aligned with the coordinate system, this could yield a speedup because it reduces the degrees of freedom of the transformation matrix to be learned by the proposal. Postponing the update of the transformation matrix until after a specific number of iterations (Ros and Hansen, 2008) and not updating it in every iteration, are other possibilities that could be further explored to achieve a better scaling behavior for L_p-Adaptation.

The quality of the approximate solutions obtained from L_p-Adaptation depends on the unknown shape of the feasible region, the dimensionality of the problem, and the starting point. No theoretical guarantees can be given. Rigorous analysis of the algorithmic complexity and solution quality of design centering and volume-approximation schemes hinges on the convexity of the feasible region (Simonovits, 2003; Lovász and Vempala, 2006), which is an unrealistic assumption in practice and not the intended application domain for L_p-Adaptation. Nevertheless, because L_p-Adaptation can be understood as a simultaneous rounding and volume-computation scheme (Simonovits, 2003), we expect the number of function evaluations that are required to reach a certain level of accuracy to scale polynomially with problem dimensionality. Our results for the synthetic L_p-balls point in the same direction. Considering arbitrary, non-convex feasible regions, future theoretical analysis of L_p-Adaptation might be possible in the PAC (Probably Approximately Correct) framework (Valiant, 2013), which has previously been successfully applied to biological and bio-inspired algorithms. This, however, is beyond the scope of our present work.

Notwithstanding these open questions, the benchmarks presented here advance the state of the art in general design centering and volume estimation. Versatile default parameters and the availability of open-source implementations render L_p-Adaptation practically useful, and we expect a number of engineering and biological problems, including novel designs of synthetic biological circuits (Woods et al., 2016; Hold et al., 2016), to benefit from a re-interpretation in the design centering framework.

The source code of L_p-Adaptation is freely available as a MATLAB toolbox on `http://mosaic.mpi-cbg.de` and via anonymous git from `https://github.com/Joe1909/LpAdaptation_Code`.

Appendices

APPENDIX

ONE

Matlab source code

Code A.1: Matlab *source code to average over covariance matrices*

```matlab
function [out_averageCov,warningCell]= averageCov(cellC)
%% Function to average over all covariance matrices found ...
    in cellC
%% Input:   cell of covariance matrices
%% Output:  struct out_averageCov with
%%          the average covariance  out_averageCov.C
%%          the asymmetric part     out_averageCov.C_asym
%%          a cell of warnings      out_averageCov.warningCell

dim = size(cellC{1},1);
numLast = size(cellC,1);
warningCell=cell(numLast,1);
warningCnt = 1;

% save average Eigenvalues and Eigenvectors
eigVal_m = nan(dim,1);
eigVec_m = nan(dim);

if numLast == 1
    %   use last covariance
    C = cellC{end};
    C_sym = C;
    C_asym = [];
    warningCell{warningCnt} = 'only one covariance';
else
    for k =1:numLast
        % get MC Covariance in each step
        C = cellC{end-numLast+k};
        % Eigen-decomposition
        [eigVec,eigVals] = eig(C);

        if k==1
            % to save average Eigenvalues and Eigenvectors
            eigVal_m = diag(eigVals);
            eigVec_m = eigVec;
            % Eigenvectors from previous iteration
            eigVec_old = eigVec;
        else
            check = zeros(dim,1);
            order = zeros(dim,1);
            for v=1:dim
                % for each Eigenvector, find corresponding ...
                    Eigenvector from
                % previous iteration
                q = eigVec(:,v);
                [¬,idx_max] = max(abs(q' * eigVec_old));

                check(idx_max) = check(idx_max) + 1;
                order(v) = idx_max;

                % add Eigenvalues
                diag_eigVals = diag(eigVals);
                eigVal_m(idx_max) = eigVal_m(idx_max) + ...
                    diag_eigVals(v);

                % check that vectors are pointing in same ...
                    direction
                projection = q' * eigVec_old(:,idx_max);
                if projection < 0
                    p = q * (-1);
                else
                    p = q;
                end
                eigVec(:,v) = p;
                % add Eigenvectors
                eigVec_m(:,idx_max)=eigVec_m(:,idx_max) + p;
            end
```

```
                if ¬isempty(find(check==0, 1))
                    error('Eigenvectors too different, not ...
                        clear which Eigenvectors to map to ...
                        which ');
                end
                % Eigenvectors to compare with in next iteration
                eigVec_old = eigVec(:,order);
            end
        end

        % divide by numLast --> get mean
        eigVec = eigVec_m./numLast;
        eigVals = eigVal_m./numLast;

        % normalize eigvectors
        for d=1:dim
            eigVec(:,d) = eigVec(:,d)./norm(eigVec(:,d));
        end
        % get covariance
        C_p = eigVec * diag(eigVals) * inv(eigVec);
        % symmetrize C
        C_sym = 0.5 * (C_p + C_p');
        % asymmetric part
        C_asym = 0.5 * (C_p - C_p');
    end

    out_averageCov.C = C_sym;
    out_averageCov.C_asym = C_asym;

    warningCell = warningCell(1:(warningCnt-1));
    out_averageCov.warningCell = warningCell;
```

APPENDIX

TWO

Additional figures for Section 4.2

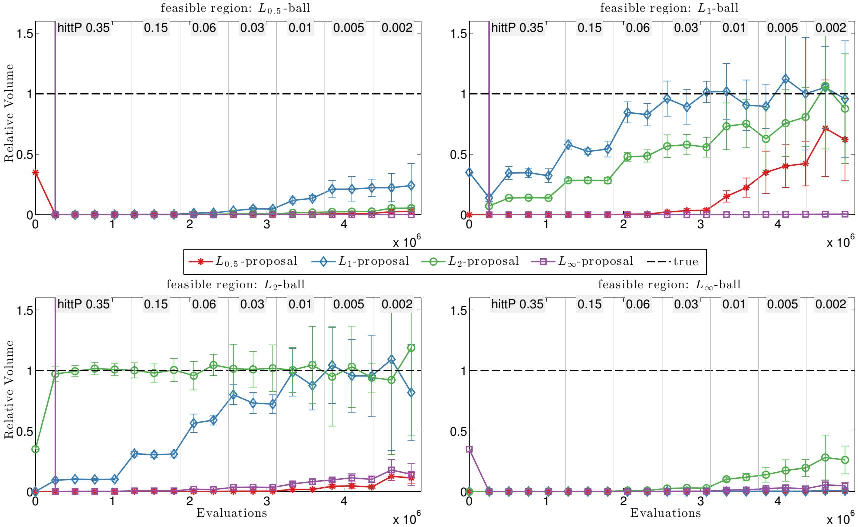


Figure B.1: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume / true volume) of stretched 50-dimensional L_p-balls ($p = 0.5, 1, 2, \infty$) approximated with proposal distributions of different p. The feasible L_p-balls are stretched along $(n-1)$ axes such that the longest axis is $\sqrt{1000}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced. The results are shown for decreasing target hitting probability, starting from the default initialization 0.35, as indicated at the top of each plot. The dashed line shows the true volume.

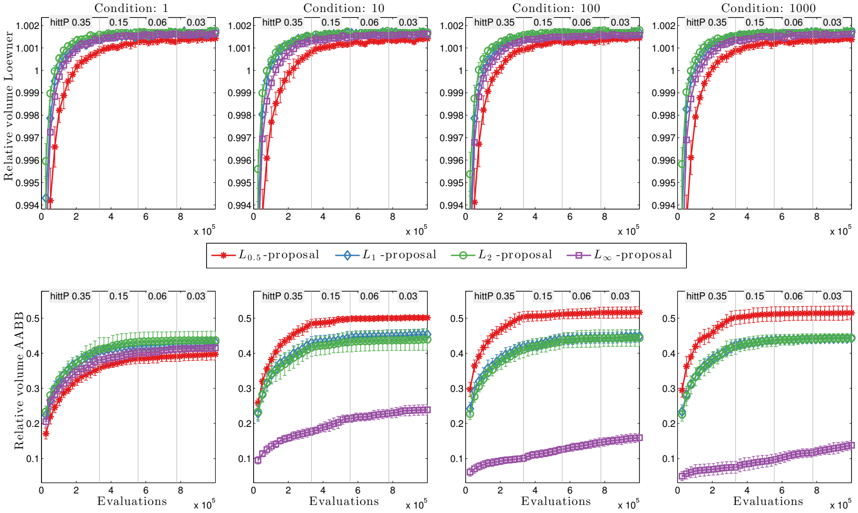


Figure B.2: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) for ten-dimensional L_2-balls approximated with proposal distributions of different p. The feasible L_2-balls are stretched along $(n-1)$ axes such that the longest axis is $\sqrt{Condition}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced.

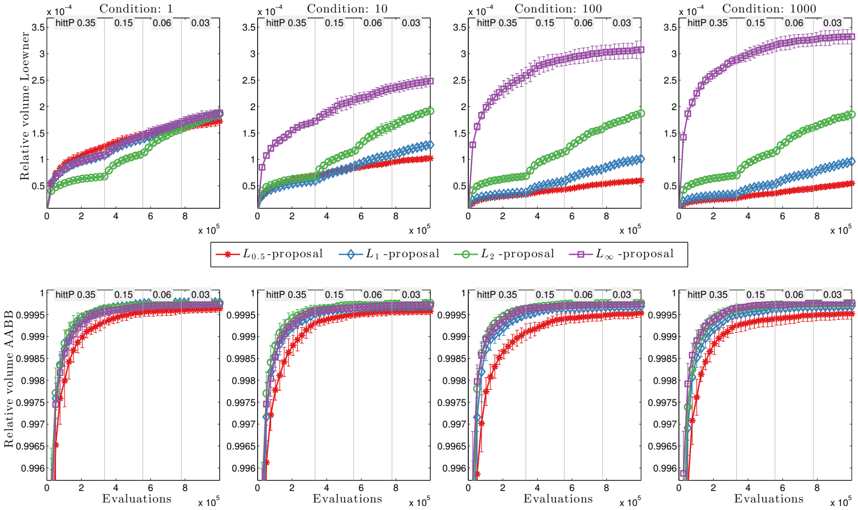


Figure B.3: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) for ten-dimensional L_∞-balls approximated with proposal distributions of different p. The feasible L_∞-balls are stretched along $(n-1)$ axes such that the longest axis is $\sqrt{Condition}$ times longer than the shortest one, and the lengths of the axes are logarithmically spaced.

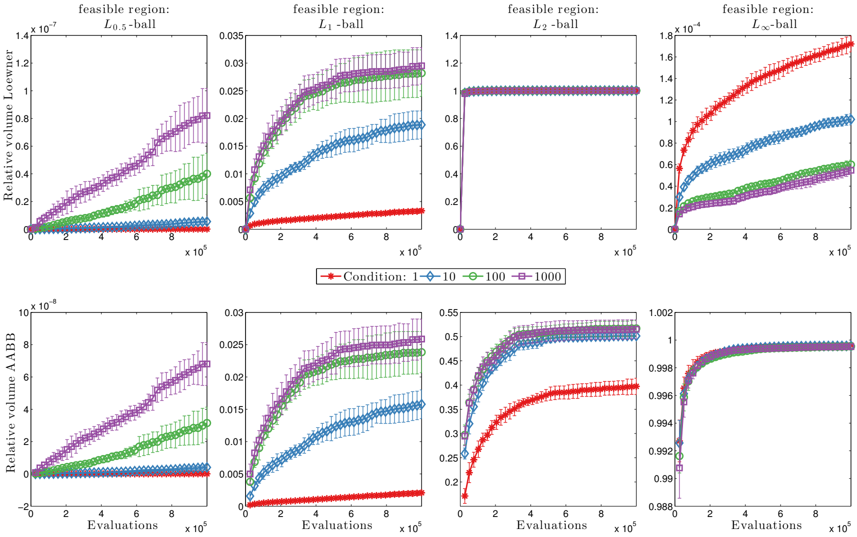


Figure B.4: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) for ten-dimensional L_p-balls of different condition approximated with $L_{0.5}$-proposals.

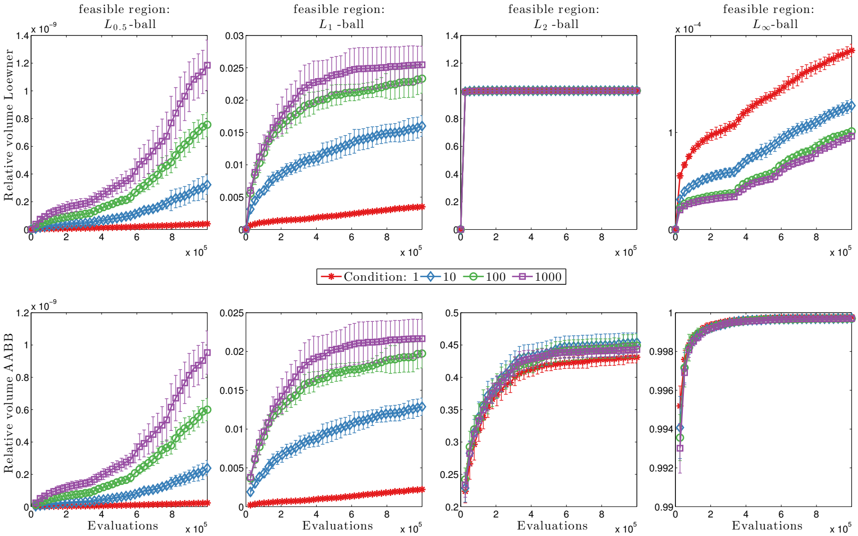


Figure B.5: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) for ten-dimensional L_p-balls of different condition approximated with L_1-proposals.

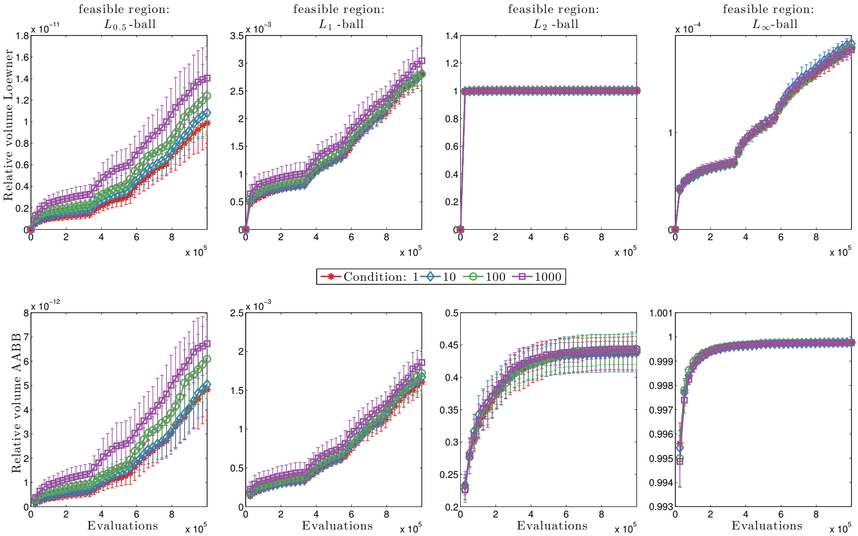


Figure B.6: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) for ten-dimensional L_p-balls of different condition approximated with L_2-proposals.

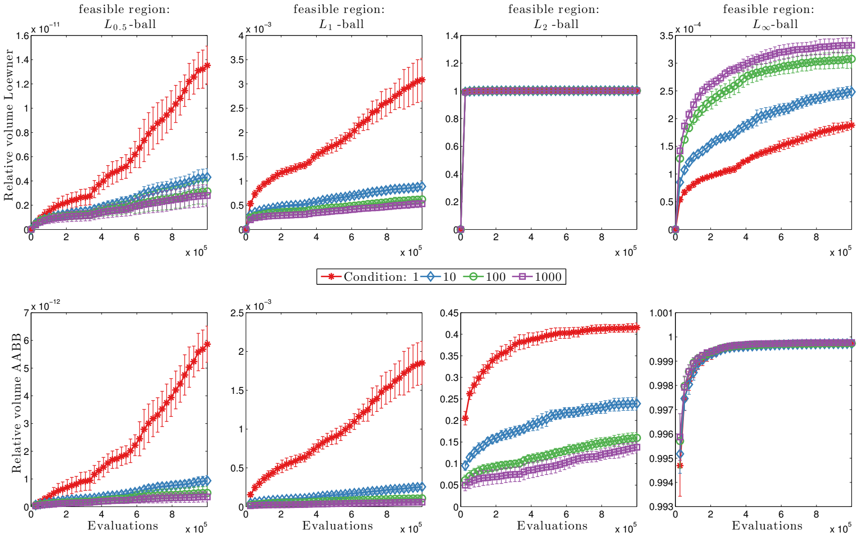


Figure B.7: Average and standard deviation (over ten independent runs) of the relative volume (estimated volume/true volume) of Loewner ellipsoids (top row) and axis-aligned bounding boxes (AABB, bottom row) for ten-dimensional L_p-balls of different condition approximated with L_∞-proposals.

APPENDIX

THREE

Additional figures for Section 5.3

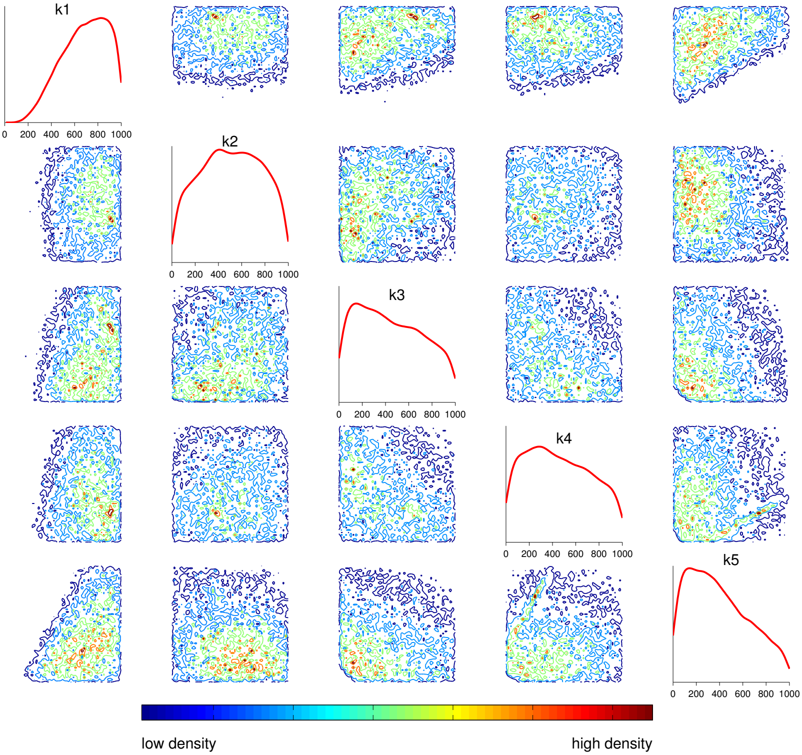


Figure C.1: Marginal densities (diagonal) and joint pairwise distributions of feasible points of the orthodox TCS in the fast response case.

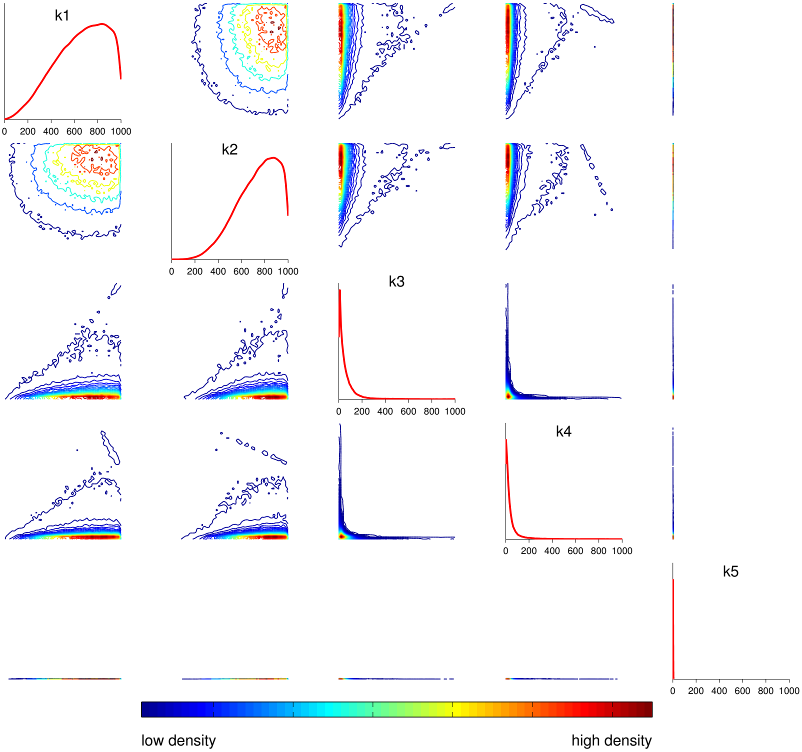


Figure C.2: Marginal densities (diagonal) and joint pairwise distributions of feasible points of the orthodox TCS in the steady output case.

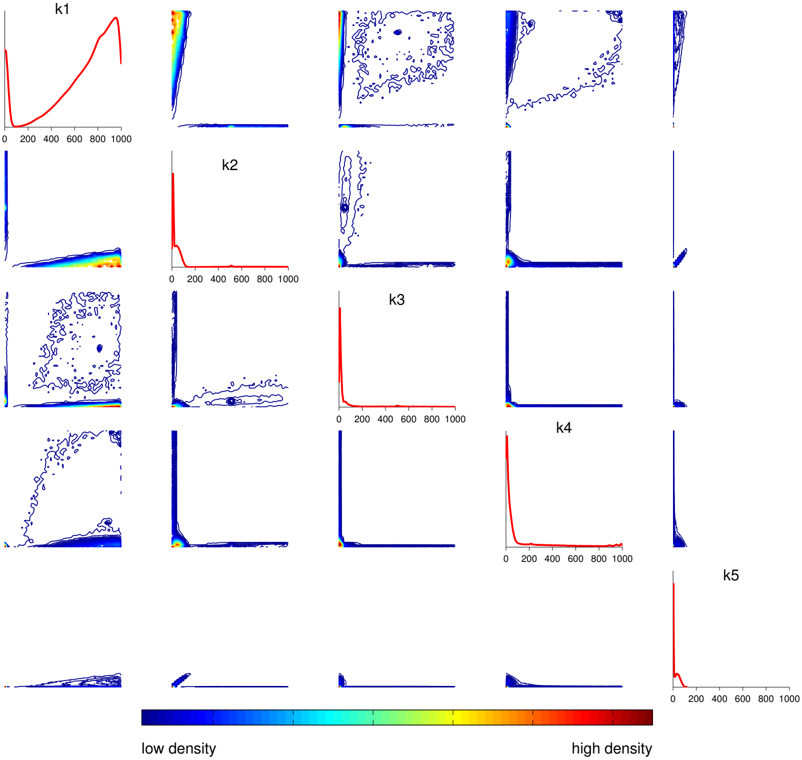


Figure C.3: Marginal densities (diagonal) and joint pairwise distributions of feasible points of the orthodox TCS in the noise rejection case.

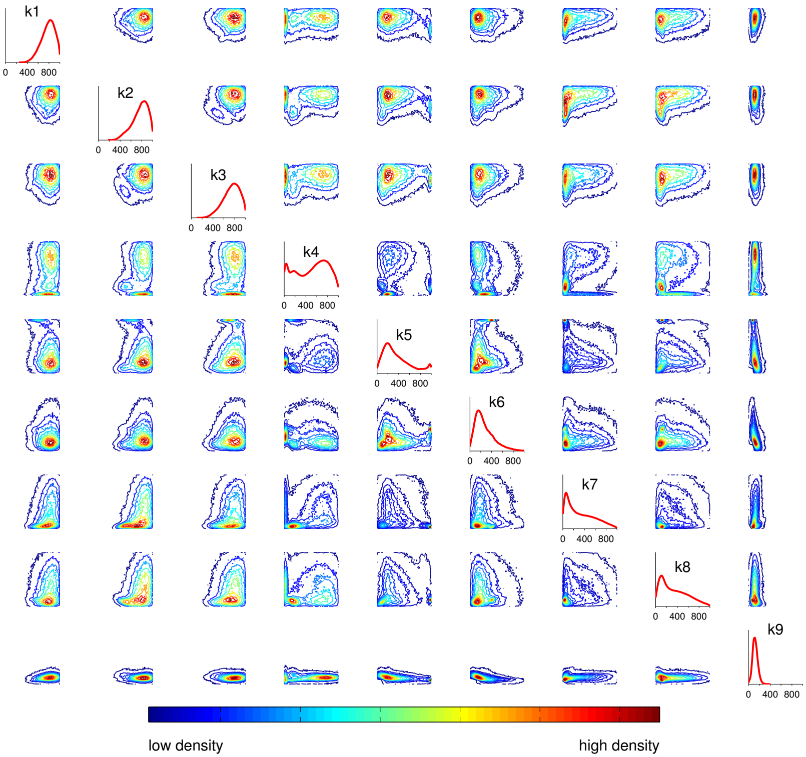


Figure C.4: Marginal densities (diagonal) and joint pairwise distributions of feasible points of the unorthodox TCS in the fast response case.

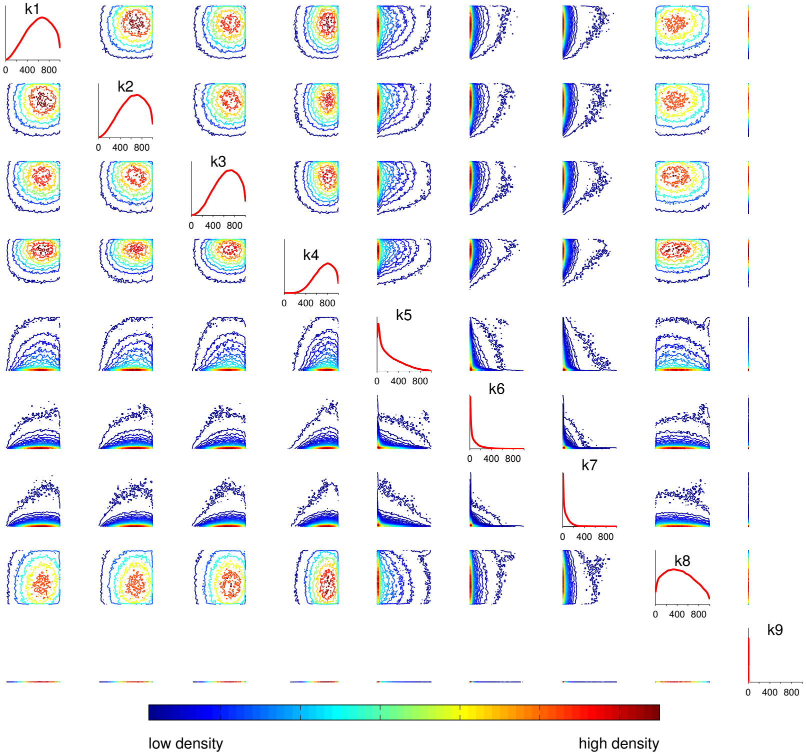


Figure C.5: Marginal densities (diagonal) and joint pairwise distributions of feasible points of the unorthodox TCS in the steady output case.

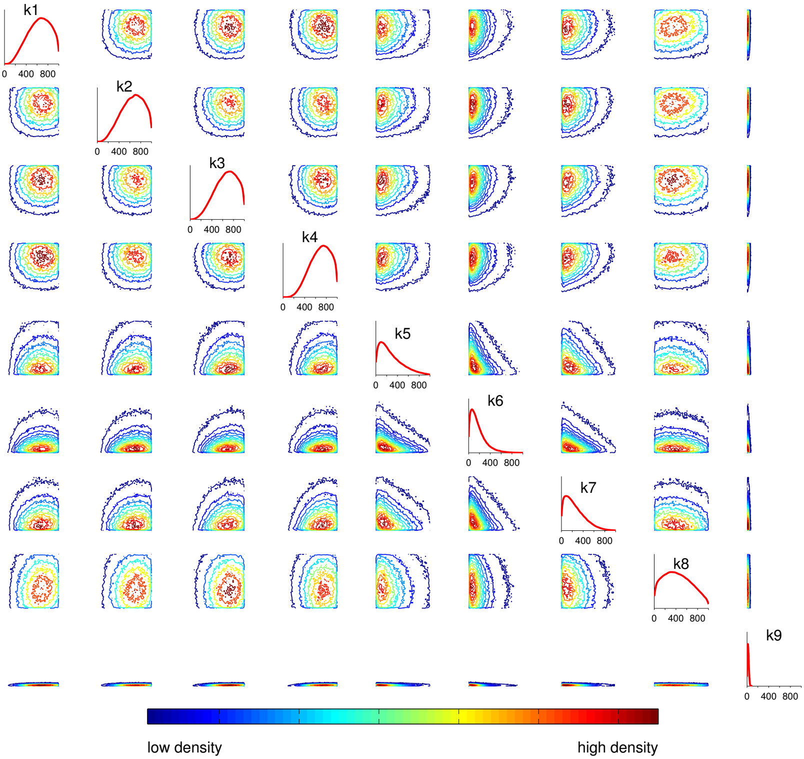


Figure C.6: Marginal densities (diagonal) and joint pairwise distributions of feasible points of the unorthodox TCS in the signal reproduction case.

BIBLIOGRAPHY

H. L. Abdel-Malek and A.-K. S. Hassan, “The ellipsoidal technique for design centering and region approximation,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 10, no. 8, pp. 1006–1014, 1991.

J. A. Acebrón, L. L. Bonilla, C. J. P. Vicente, F. Ritort, and R. Spigler, “The kuramoto model: A simple paradigm for synchronization phenomena,” *Reviews of modern physics*, vol. 77, no. 1, p. 137, 2005.

C. Andrieu and J. Thoms, “A tutorial on adaptive MCMC,” *Statistics and Computing*, vol. 18, no. 4, pp. 343–373, Dec. 2008.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, “An introduction to MCMC for machine learning,” *Machine Learning*, vol. 50, pp. 5–43, 2003.

A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, “Classification of covariance matrices using a Riemannian-based kernel for BCI applications,” *Neurocomputing*, vol. 112, pp. 172–178, 2013.

I. Bárány and Z. Füredi, “Computing the volume is difficult,” *Discrete & Computational Geometry*, vol. 2, no. 4, pp. 319–326, 1987.

C. P. Barnes, D. Silk, X. Sheng, and M. P. Stumpf, “Bayesian design of synthetic biological systems,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 37, pp. 15 190–15 195, 2011.

H.-G. Beyer and H.-P. Schwefel, “Evolution strategies–a comprehensive introduction,” *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

B. Büeler, A. Enge, and K. Fukuda, “Exact volume computation for polytopes: a practical study,” in *Polytopes—combinatorics and computation.* Springer, 2000, pp. 131–154.

G. Calafiore, F. Dabbene, and R. Tempo, “Uniform sample generation in lp balls for probabilistic robustness analysis,” in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 3. IEEE, 1998, pp. 3335–3340.

G. Casella, “Bayesians and frequentists, models, assumptions, and inference,” 2008, aCCP 37th Annual Meeting, Philadelphia, PA.

B. Cousins and S. Vempala, “A practical volume algorithm,” *Mathematical Programming Computation*, vol. 8, no. 2, pp. 133–160, 2016.

A. Dayarian, M. Chaves, E. D. Sontag, and A. M. Sengupta, “Shape, size, and robustness: feasible regions in the parameter space of biochemical networks,” *PLoS Comput Biol*, vol. 5, no. 1, p. e1000256, 2009.

S. W. Director and G. D. Hachtel, “The simplicial approximation approach to design centering,” *Circuits and Systems, IEEE Transactions on*, vol. 24, no. 7, pp. 363–372, 1977.

M. Dyer, A. Frieze, and R. Kannan, “A random polynomial-time algorithm for approximating the volume of convex bodies,” *Journal of the ACM (JACM)*, vol. 38, no. 1, pp. 1–17, 1991.

M. E. Dyer and A. M. Frieze, “On the complexity of computing the volume of a polyhedron,” *SIAM Journal on Computing*, vol. 17, no. 5, pp. 967–974, 1988.

W. Förstner and B. Moonen, “A metric for covariance matrices. Krumm, F. Schwarze, vs (hg.): Quo vadis geodesia…?, festschrift for Erik W. Grafarend on the occasion of his 60th birthday,” TR Dept. of Geodesy and Geoinformatics, Stuttgart University, Tech. Rep., 1999.

C. Ge and F. Ma, “A fast and practical method to estimate volumes of convex polytopes,” in *International Workshop on Frontiers in Algorithmics.* Springer, 2015, pp. 52–65.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis.* Chapman & Hall/CRC Boca Raton, FL, USA, 2014, vol. 2.

C. Geyer, "Introduction to Markov chain Monte Carlo," *Handbook of Markov Chain Monte Carlo*, pp. 3–48, 2011.

C. J. Geyer, "Markov chain Monte Carlo lecture notes," *Course notes, Spring Quarter*, 1998.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, "Introducing Markov chain Monte Carlo," *Markov chain Monte Carlo in practice*, vol. 1, p. 19, 1996.

H. E. Graeb, *Analog Design Centering and Sizing.* Springer, 2007.

A. Grelaud, C. P. Robert, and J.-M. Marin, "ABC methods for model choice in Gibbs random fields," *Comptes Rendus Mathematique*, vol. 347, no. 3, pp. 205–210, 2009.

M. Grötschel, L. Lovász, and A. Schrijver, "Geometric algorithms and combinatorial optimization," *Journal of the Operational Research Society*, no. 40, p. 797, 1988.

P. M. Gruber, "John and Loewner ellipsoids," *Discrete & Computational Geometry*, vol. 46, no. 4, pp. 776–788, 2011.

C. Gu and J. Roychowdhury, "Yield estimation by computing probabilistic hypervolumes," in *Extreme Statistics in Nanoscale Memory Design.* Springer, 2010, pp. 137–177.

H. Haario, E. Saksman, and J. Tamminen, "Adaptive proposal distribution for random walk Metropolis algorithm," *Comput. Stat.*, vol. 14, no. 3, pp. 375–395, 1999.

——, "An adaptive Metropolis algorithm," *Bernoulli*, vol. 7, no. 2, pp. 223–242, 2001.

M. Hafner, "Quantitative analysis of robustness in systems biology: Combining global and local approaches," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, 2010.

M. Hafner, H. Koeppl, M. Hasler, and A. Wagner, "'Glocal' robustness analysis and model discrimination for circadian oscillators," *PLoS Comput. Biol.*, vol. 5, no. 10, p. e1000534, 2009.

N. Hansen, "Adaptive encoding for optimization," 2008.

N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

N. Hansen, S. D. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, Spr 2003.

N. Hansen, "The CMA evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

N. Hansen and A. Ostermeier, "Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation," in *Proceedings of the 1996 IEEE Conference on Evolutionary Computation (ICEC '96)*, 1996, pp. 312–317.

T. Harnisch, J. Kunert, H. Toepfer, and H. Uhlmann, "Design centering methods for yield optimization of cryoelectronic circuits," *IEEE transactions on applied superconductivity*, vol. 7, no. 2, pp. 3434–3437, 1997.

W. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

D. Henrion, J. B. Lasserre, and C. Savorgnan, "Approximate volume and integration for basic semialgebraic sets," *SIAM review*, vol. 51, no. 4, pp. 722–743, 2009.

C. Hold, S. Billerbeck, and S. Panke, "Forward design of a complex enzyme cascade reaction," *Nature Communications*, vol. 7, 2016.

D. J. Jörg, A. Pollakis, L. Wetzel, M. Dropp, W. Rave, F. Jülicher, and G. Fettweis, "Synchronization of mutually coupled digital PLLs in massive MIMO systems," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1716–1721.

L. G. Khachiyan, "The problem of calculating the volume of a polyhedron is enumerably hard," *RUSSIAN MATHEMATICAL SURVEYS*, vol. 44, no. 3, pp. 199–200, MAY-JUN 1989.

——, "Rounding of polytopes in the real number model of computation," *Mathematics of Operations Research*, vol. 21, no. 2, pp. 307–320, 1996.

L. Khachiyan, "On the complexity of computing the volume of a polytope," *Izvestia Akad. Nauk SSSR, Engineering Cybernetics*, vol. 3, pp. 216–217, 1988.

J.-R. Kim and K.-H. Cho, "The multi-step phosphorelay mechanism of unorthodox two-component systems in E. coli realizes ultrasensitivity to stimuli while maintaining robustness to noises," *Computational biology and chemistry*, vol. 30, no. 6, pp. 438–444, 2006.

H. Kitano, "Biological robustness," *Nature Rev. Genetics*, vol. 5, no. 11, pp. 826–837, 2004.

G. Kjellström and L. Taxen, "Stochastic optimization in system design," *IEEE Trans. Circ. and Syst.*, vol. 28, no. 7, pp. 702–715, July 1981.

Y. Kuramoto, "Self-entrainment of a population of coupled non-linear oscillators," in *International symposium on mathematical problems in theoretical physics.* Springer, 1975, pp. 420–422.

V. Lacko and R. Harman, "A conditional distribution approach to uniform sampling on spheres and balls in lp spaces," *Metrika*, vol. 75, no. 7, pp. 939–951, 2012.

J. Lasserre, "Unit balls of constant volume: which one has optimal representation?" *arXiv preprint arXiv:1408.1324*, 2014.

S. Liu, J. Zhang, and B. Zhu, "Volume computation using a direct Monte Carlo method," in *International Computing and Combinatorics Conference.* Springer, 2007, pp. 198–209.

L. Lovász, "Hit-and-run mixes fast," *Mathematical Programming*, vol. 86, no. 3, pp. 443–461, 1999.

L. Lovász and S. Vempala, "Simulated annealing in convex bodies and an O*(N4) volume algorithm," *Journal of Computer and System Sciences*, vol. 72, no. 2, pp. 392–417, 2006.

P. J. Menck, J. Heitzig, N. Marwan, and J. Kurths, “How basin stability complements the linear-stability paradigm,” *Nature Physics*, vol. 9, no. 2, pp. 89–92, 2013.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *J. Chem. Phys.*, vol. 21, pp. 1087–1092, 1953.

C. L. Müller and I. F. Sbalzarini, “Gaussian Adaptation revisited — an entropic view on covariance matrix adaptation,” in *Proc. EvoStar*, ser. Lect. Notes Comput. Sci., vol. 6024. Istanbul, Turkey: Springer, April 2010, pp. 432–441.

——, “Gaussian Adaptation as a unifying framework for continuous black-box optimization and adaptive Monte Carlo sampling,” in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, July 18-23 2010, pp. 2594–2601.

——, “Gaussian Adaptation for robust design centering,” in *Evolutionary and deterministic methods for design, optimization and control, Proc. EuroGen*, C. Poloni, D. Quagliarella, J. Périaux, N. Gauger, and K. Giannakoglou, Eds. Capua, Italy: CIRA, ECCOMAS, ERCOFTAC, September 14–16 2011, pp. 736–742.

F. Peruani, E. M. Nicola, and L. G. Morelli, “Mobility induces global synchronization of oscillators in periodic extended systems,” *New Journal of Physics*, vol. 12, no. 9, p. 093029, 2010.

A. Pollakis, L. Wetzel, D. J. Jörg, W. Rave, G. Fettweis, and F. Jülicher, “Synchronization in networks of mutually delay-coupled phase-locked loops,” *New Journal of Physics*, vol. 16, no. 11, p. 113009, 2014.

D. Prangle *et al.*, “Adapting the ABC distance function,” *Bayesian Analysis*, 2016.

J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman, “Population growth of human Y chromosomes: A study of Y chromosome microsatellites,” *Molecular Biology and Evolution*, vol. 16, no. 12, pp. 1791–1798, 1999.

B. Puchalski, L. Zielinski, and J. Rutkowski, "Use of granular method to design centering," in *2006 IEEE International Symposium on Circuits and Systems*, May 2006, pp. 4 pp.–3989.

G. Roberts and J. Rosenthal, "Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms," *Journal of applied probability*, vol. 44, no. 2, pp. 458–475, 2007.

R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *International Conference on Parallel Problem Solving from Nature.* Springer, 2008, pp. 296–305.

J. S. Rosenthal *et al.*, "Optimal proposal distributions and adaptive MCMC," *Handbook of Markov Chain Monte Carlo*, pp. 93–112, 2011.

S. S. Sapatnekar, P. M. Vaidya, and S.-M. Kang, "Convexity-based algorithms for design centering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 12, pp. 1536–1549, 1994.

H. Schmidt and M. Jirstrand, "Systems biology toolbox for MATLAB: a computational platform for research in systems biology," *Bioinformatics*, vol. 22, no. 4, pp. 514–515, 2006.

R. Schwencker, F. Schenkel, H. Graeb, and K. Antreich, "The generalized boundary curve - a common method for automatic nominal design centering of analog circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '00. New York, NY, USA: ACM, 2000, pp. 42–47.

A. Seifi, K. Ponnambalam, and J. Vlach, "A unified approach to statistical design centering of integrated circuits with correlated parameters," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 46, no. 1, pp. 190–196, 1999.

C. Sherlock, P. Fearnhead, and G. O. Roberts, "The random walk Metropolis: linking theory and practice through a case study," *Statistical Science*, pp. 172–190, 2010.

M. Simonovits, "How to compute the volume in high dimension?" *Mathematical programming*, vol. 97, no. 1-2, pp. 337–374, 2003.

R. Soin and R. Spence, “Statistical exploration approach to design centring,” in *IEE Proceedings G (Electronic Circuits and Systems)*, vol. 127, no. 6. IET, 1980, pp. 260–269.

R. Storn, “System design by constraint adaptation and differential evolution,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 22–34, 1999.

R. Storn and K. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

M. Sunnåker, A. G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz, “Approximate Bayesian computation,” *PLoS Comput Biol*, vol. 9, no. 1, p. e1002803, 2013.

H. K. Tan and Y. Ibrahim, “Design centering using momentum based CoG,” *Engineering Optimization*, vol. 32, no. 1, pp. 79–100, 1999.

T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. H. Stumpf, “Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems,” *J. R. Soc. Interface*, vol. 6, no. 31, pp. 187–202, 2009.

P. M. Vaidya, “A new algorithm for minimizing convex functions over convex sets,” in *Foundations of Computer Science, 1989., 30th Annual Symposium on.* IEEE, 1989, pp. 338–343.

L. Valiant, *Probably Approximately Correct: Nature’s Algorithms for Learning and Prospering in a Complex World.* New York, NY, USA: Basic Books, Inc., 2013.

S. S. Vempala, “Recent progress and open problems in algorithmic convex geometry,” in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

L. M. Vidigal and S. W. Director, “A design centering algorithm for nonconvex regions of acceptability,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, no. 1, pp. 13–24, 1982.

G. von Dassow, E. Meir, E. M. Munro, and G. M. Odell, “The segment polarity network is a robust developmental module,” *Nature*, vol. 406, pp. 188–192, 2000.

L. Wetzel, “Effect of distributed delays in systems of coupled phase oscillators,” Ph.D. dissertation, Citeseer, 2012.

L. Wetzel, D. J. Jörg, A. Pollakis, W. Rave, G. Fettweis, and F. Jülicher, “Self-organized synchronization of digital phase-locked loops with delayed coupling in theory and experiment,” *PLoS ONE*, vol. 12, no. 2, p. e0171590, 2017.

D. Whitley, “A genetic algorithm tutorial,” *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

D. A. Wiley, S. H. Strogatz, and M. Girvan, “The size of the sync basin,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 16, no. 1, p. 015103, 2006.

M. L. Woods, M. Leon, R. Perez-Carrasco, and C. P. Barnes, “A statistical approach reveals designs for the most robust stochastic gene oscillators,” *ACS synthetic biology*, 2016.

J. Yang, “Convergence and efficiency of adaptive MCMC,” Ph.D. dissertation, University of Toronto, 2016.

F. Yger, F. Lotte, and M. Sugiyama, “Averaging covariance matrices for EEG signal classification based on the CSP: an empirical study,” in *Signal Processing Conference (EUSIPCO), 2015 23rd European.* IEEE, 2015, pp. 2721–2725.

E. Zamora-Sillero, M. Hafner, A. Ibig, J. Stelling, and A. Wagner, “Efficient characterization of high-dimensional parameter spaces for systems biology,” *BMC systems biology*, vol. 5, no. 1, p. 142, 2011.

Curriculum Vitae

Name: Josefine Asmus
Date of birth: September 19th, 1986
Place of birth: Rostock, Germany
Citizen of: Germany

2013 - 2017	Ph.D. studies in Computer Science at Dresden University of Technology, Dresden, Germany. Ph.D. project carried out at the Center for Systems Biology Dresden and the Max Planck Institute of Molecular Cell Biology and Genetics Advisor: Prof. Dr. Ivo F. Sbalzarini and Dr. Christian L. Müller (Flatiron Institute, Simons Foundation, New York City)
2009	Studies in Biomathematics at Massey University in Palmerston North, New Zealand
2006 - 2012	Diploma in Biomathematics at Ernst-Mortiz-Arndt-University Greifswald, Greifswald, Germany

PUBLICATIONS

J. Asmus, C. L. Müller, I. F. Sbalzarini. **Lp-Adaptation: Simultaneous Design Centering and Robustness Estimation of Electronic and Biological Systems**, accepted by Scientific Reports, 2017

J. Asmus, D. Borchmann, I. F. Sbalzarini, and D. Walther. **Towards an FCA-based Recommender System for Black-Box Optimization.** Proceedings of the International Workshop "What can FCA do for Artificial Intelligence?" (FCA4AI2014), 2014.

G. Hempel, A. Goens, J. Asmus, J. Castrillon, I. F. Sbalzarini. **Robust Mapping of Process Networks to Many-Core Systems using Bio-Inspired Design Centering** accepted at SCOPES17, 2017

DECLARATION OF AUTHORSHIP

This thesis is a presentation of my own original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature. The work was done at the Technische Universität Dresden, and the project was carried out at the Center for Systems Biology Dresden and the Max Planck Institute of Molecular Cell Biology and Genetics under the supervision of Prof. Dr. Ivo F. Sbalzarini. This project was co-supervised by Dr. Christian L. Müller from the Simons Foundation in New York City. I hereby declare that this thesis has not been submitted before to any institution for assessment purposes.

Dresden, February 17, 2017 Josefine Asmus