universität wien

MASTER'S THESIS

Title of the Master's Thesis

Optimization of Advanced Markov Chain Monte Carlo Methods with Applications to Biological Systems

submitted by

Richard D. Paul, BSc

in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Vienna, 2022

Degree programme code as it appears on the student record sheet:	UA 066 910
Degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Supervisor:	Univ.-Prof. Mag. Dr. Christoph Dellago

Abstract

Zusammenfassung

Acknowledgements

Contents

Nomenclature

Chapter 1

Introduction

The task of *inferring* non-observable *parameters* from the observable outcome of an experiment is a crucial and ever recurring task in the natural and engineering sciences. Many complex systems of interest can be modeled very accurately as dynamical systems. However, the parametrization of such models determines their behaviour significantly. As such, the search for sets of parameters, that allow to recover and explain real world observations as closely as possible is decisive to gain practical benefit from such models. In particular, such *model calibration* goes beyond the pure understanding of the system, but is crucial for prediction and simulation of possible future outcomes. In the light of the Corona pandemic, epidemiological models have become well-known to the general public as a tool to predict the spread of the disease. Within these models, the basic reproductive number R_0 is a key parameter controlling the dynamics of an outbreak, which however has to be *estimated* from the *data*, i.e. the number of infections per some fixed time interval.

Dynamical models, which describe interactions between different *species*, are omnipresent in the life sciences [26]. Thinking of the species as nodes and their interactions as edges in a network, it is quite intuitive to refer to these models as *network models*. Their list ranges from food webs [68], describing large scale ecological networks, over epidemiological models [3, 92] down to microbial [32] and biochemical reaction networks [67, 102], which aid in understanding and describing the inner workings of cells and organisms. These models find applications in solving many of the urgent problems of our century, such as preserving biodiversity [98], developing climate-friendly biotechnology [5] and curing diseases [22, 63]. A particular characteristic of network models, is the rise of linear constraints from modeling assumptions. For example, such constraints may stem from prior considerations on the nominal "direction" of the dynamics[1] or when mass balances are introduced over the species, deeming some parameter values infeasible a priori. The vast increments in computational power over the last decades has given rise to ever larger and more complex systems, up to the point where models nowadays are designed to capture the whole human metabolism [91]. However, this rise in system size is reflected in an ever growing number of parameters, which have to be inferred in order to make use of the models.

Bayesian inference is a statistical framework to perform inference based on a Bayesian paradigm and has also profited from the rise of computational power [53]. Unlike classical, frequentist statistics, Bayesian inference does not assume the existence of a "true", but unknown parameter value, but treats the parameters themselves as random variables, which are more

[1] When for example modeling the interactions in a predator-prey system, it seems very unlikely for a rabbit to spawn from the guts of a fox, whereas foxes seems quite likely to consume rabbits. So a negative consumption rate of rabbits is not meaningful.

likely to take values that *explain* the data well. Inference of the parameters is then performed by considering features of the *posterior distribution* of the parameters given the data, where typical quantities of interest are expectations and quantiles. The generality of the Bayesian paradigm allows Bayesian inference not only to take into account complex relationships between parameters, heterogeneous, noisy or non-informative data as well as prior domain knowledge within a conceptually simple framework, but makes it particularly well suited to capture uncertainties in the parameters. Clearly, this comes at the cost of being confronted with arbitrarily shaped distributions, over which one requires to compute expectations, i.e. perform integration. It is the power of Markov chains, unleashed by the Metropolis-Hastings algorithm [37], which makes Bayesian inference applicable in practice. The numerical simulation of particularly constructed Markov chains yields samples from the posterior distribution, which are then used to estimate expectations numerically.

However, the asymptotic nature of Markov chain convergence can become obstructive for large scale problems as the algorithm might take days, weeks or even months to converge. Paired with the growing need for fast and robust parameter estimation for ever larger problems, this has led to the development of new Markov chain Monte Carlo methods continuing unabated throughout the last decades [52]. Extensions and adaptations range from increasingly sophisticated proposal algorithms [31, 59, 52, 64] over approximate and adaptive samplers [19, 18, 58, 12, 48, 25, 33] to multilevel and surrogate techniques [18, 95, 64, 66, 25, 33]. The question of optimal hyperparametrization of the algorithms in use has yet gained relatively little attention. The findings of this work, however, suggest that a good hyperparametrization of the algorithm is crucial for statistical efficiency and, thus, for accelerated convergence. A commonly used approach is acceptance rate tuning, where one aims to achieve a particular target acceptance rate of the Metropolis criterion. This approach is based on theoretical results, stating that for independent multivariate distributions the optimal statistical efficiency is achieved at an acceptance rate of 0.234 when using a Gaussian random walk sampler [72, 73, 105]. However, the independence assumption is not true in general and, in particular, does not hold for linearly constrained domains, which are often encountered in Bayesian inference for biological network models. Thus, it remains an open question, whether acceptance rate tuning can still achieve optimal sampling efficiency in such settings.

A particular issue with the hyperparametrization of the Metropolis-Hastings algorithm stems from tuning often being performed manually due to the lack of reliable tuning algorithms. Since Bayesian inference is supposed to be a powerful tool for applied scientists, manual tuning is undesirable, not only consuming precious time, but also requiring the practitioner to deal with the algorithmic aspects of the method. *Bayesian optimization* is a particular discipline of optimizing black box functions, which has gained popularity for hyperparametrization tasks [84, 94, 40]. In particular, Thompson Sampling [93] is a Bayesian optimization technique that not only somewhat mimics human behaviour when facing black box problems, but also has strong theoretical convergence and performance guarantees. As such, Bayesian optimization methods have been advertised as "taking the human out of the loop" [80]. Clearly, this allows the practitioner to concentrate on her work and makes efficient sampling more accessible.

1.1 Contribution

In its core, this work deals with the question of hyperparametrization of the Metropolis-Hastings algorithm for linearly constrained distributions with expensive densities. We present theoretical considerations, why the "classical" result for tuning the Metropolis-Hastings algorithm might not hold true for our particular problem. Our hypotheses are tested in an extensive, empirical

study on small example problems from the field of Bayesian ^{13}C metabolic flux analysis [87], that convey typical characteristics encountered in real world problems. The results of this experimental study indeed suggest, that statistical efficiency can be largely improved, when tuning w.r.t. the expected squared jump distance [65] instead of the acceptance rate.

Motivated by these observations, we present a Thompson Sampling-based tuning algorithm, which is optimized to work well with Markov chain Monte Carlo estimates of the form

$$\frac{1}{N}\sum_{i=1}^{N} f(x_i) \approx \int f(x)\pi(x)\mathrm{d}x \tag{1.1}$$

by aggregating data in preliminary steps, which speeds up internally required matrix inversions. We present results achieved with this tuning algorithm and assess its performance against the results of the extensive, aforementioned study.

In order to perform the experiments presented throughout this thesis and to accelerate development cycles, the open source Python library `hopsy`[2] was implemented. It provides bindings of the existing high-performance C++ library `HOPS`[3] to Python, allowing to use its power through a simple Python API. In an effort to make `hopsy` a lasting contribution to the open source and research software community, care was taken to meet quality standards for research software engineering.

1.2 Related Work

In [72], the optimal scaling for the Gaussian random walk sampling a multivariate d-dimensional distribution with independent and identical marginal distributions was proven to be the scaling at which the acceptance rate of the Metropolis-Hastings algorithm is 0.234 in the asymptotic limit of $d \to \infty$. Due to the assumptions of independent marginals, this result is restricted to a rather small class of problems. In inference for dynamic models, parameters may interact in complex manners and correlate, in which case the above assumptions do not hold true for the the posterior distribution of the parameters. The technique, that [72] apply to prove their result and, thus, the approach to determine optimal scaling via the acceptance rate, are not applicable to more general problems.

The expected squared jump distance is a quantity well-known to be directly related to the autocorrelation of consecutive samples from the Markov chain [65] and has been used as an alternative tuning approach, as maximizing the expected squared jump distance minimizes autocorrelation and thus increases the statistical efficiency of the samples. [65] present an importance sampling based technique to determine an optimal stepsize.

Instead of considering the tuning of the Markov chain as a preliminary step, adaptive algorithms try to learn good proposal distributions on the go. Although the adaptive nature makes the algorithm non-Markovian in general, such a process will nevertheless converge towards the correct distribution asymptotically, if the adaptation vanishes throughout time [34, 4].

In fact, [65] and [106] consider their tuning approaches as an adaptive Markov chain, which at some point stops the tuning and continues sampling with the best parameter settings found until that point. As such, any combination of preliminary tuning and consecutive sampling can be considered an adaptive Markov chain, if the tuning is either guaranteed to converge or stops within finitely many iterations.

[3] `https://github.com/modsim/hopsy`

[3] `https://github.com/modsim/hops`

1.3 Structure

This thesis starts with an introduction into Bayesian inference in Chapter 2, where we introduce the posterior distribution and typical quantities of interest as well as how they may be estimated given samples from the posterior distribution. We then consider how Bayesian inference may be applied to dynamical systems and what pitfalls this might introduce.

In Chapter 3, we introduce metabolic networks as a particular class of biological network models, where linear constraints arise naturally and the cost to simulate a network can be high. After sketching some of the key modeling assumptions and techniques, we come back to the question of inference and highlight the connections to the Bayesian inference framework for dynamical models discussed previously.

Having introduced the network problem, which we aim to solve, we turn towards a quick introduction into the theory behind Markov chains and introduce the Metropolis-Hastings algorithm in Chapter 4. After stating the most important results regarding uniqueness and asymptotic convergence of and to the target distribution, we survey techniques for diagnosing convergence in practice.

Facing the particular problem of sampling distributions with linearly constrained support, in Chapter 5, we emphasize the importance of using proposal algorithms specifically designed to work well in such settings. We name and describe the most common algorithms as well as the common precondition technique of polytope rounding.

Starting from Chapter 6, we delve into the topic of hyperparametrization of Markov chain Monte Carlo methods. After introducing the effective sample size, one of the most common measures of statistical efficiency for Markov chain Monte Carlo methods, we describe its relation to expected squared jump distance, which is much easier to estimate. For completeness, we state the theoretical result about optimal tuning using the acceptance rate from [72], before discussing caveats and possible pitfalls of this method in combination with linearly constrained distributions.

In Chapter 7, we introduce some basics of Bayesian optimization and discuss Gaussian processes in particular, which are the tool of choice for Bayesian optimization on continuous domains. Further, we state how estimates of the form as in (1.1) can be aggregated in preliminary steps, allowing to speed up matrix inversions related to evaluating Gaussian processes. Once this is established, we state the Thompson Sampling-based tuning algorithm, which is a core contribution of this work.

Before turning to the experimental results, we give a short presentation of the open source software library `hopsy` in Chapter 8, focusing on motivation, main design principles, its capabilities and good practices from software research engineering used throughout the development.

Finally, in Chapter 9, we present the results of our extensive, experimental study conducted on on a number of problems, ranging from toy examples over synthetic models to small real-world examples. We then turn towards the proposed tuning algorithm from Chapter 7 and validate it on few small test problems, before comparing its performance with the results of the extensive study.

The results as well as possible further improvements and research directions are discussed in Chapter 10, before concluding the thesis in Chapter 11.

Chapter 2

Bayesian Inference

Given a model of a data generating process, equipped with unknown *parameters*, and some observed *data*, an *inference* task aims at deriving information on the parameters from the data. In *Bayesian inference*, the quality of parameter estimates is assessed using the *posterior distribution* $\pi(\theta \mid D)$ of the parameters θ given the data D. Using Bayes' theorem, it is given as

$$\pi(\theta \mid D) = \frac{\mathcal{L}(D \mid \theta)\, p(\theta)}{Z}$$

as a product of likelihood $\mathcal{L}(D \mid \theta)$ and the *prior* $p(\theta)$ and normalized by

$$Z = \int_\Theta \mathcal{L}(D \mid \theta)\, p(\theta)\, \mathrm{d}\theta, \tag{2.1}$$

which is known as the *evidence*. As such, the parameters themselves are considered random variables distributed with law conditional on the observed data. Intuitively, the probability measure on the parameters can be interpreted as a measure of plausibility of the parameters in explaining the data.

The prior distribution $p(\theta)$ is used to encode prior knowledge on the parameters. For example, constraints on the parameters can be encoded by setting the prior to be the uniform distribution over the constrained region. The likelihood $\mathcal{L}(D \mid \theta)$ is a statistical model, that describes how the data would have been distributed, if the given parameter θ were the *true* value. However, very much unlike classical inference there is no assumption on the existence of a true value in Bayesian statistics. Thus, the likelihood can be rather seen as a measure, that encodes how well a particular parameter *explains* the data. A common choice for the likelihood function is the Gaussian distribution, where the data is assumed to be normally distributed around some function $f(\theta)$ of the parameter, i.e.

$$\mathcal{L}(D \mid \theta) = \sqrt{\det(2\pi\Sigma)}^{-1} \exp\left\{ -\frac{1}{2} \| D - f(\theta) \|^2_{\Sigma^{-1}} \right\}. \tag{2.2}$$

Finally, the evidence Z is the overall probability of observing the data. From a mathematical perspective, the evidence is crucial in order to make π a well-defined distribution. In practice, however, the integral (2.1) can be high-dimensional and thus it may be infeasible to solve analytically. Fortunately, Markov chain Monte Carlo methods allow to draw samples from the posterior distribution without having to know the evidence, as we will see in Chapter 4. Using the Law of Large Numbers, expectations w.r.t. the posterior distribution are then estimated numerically. Typical expectations, which are interesting for inference are the expectation

$$\mu_\theta = \mathbb{E}[\theta] = \int_\Theta \theta\, \pi(\theta \mid D)\, \mathrm{d}\theta,$$

or the variance of the parameters

$$\sigma_\theta = \mathbb{E}[\theta^2] - \mathbb{E}[\theta]^2 = \int_\Theta \theta^2 \, \pi(\theta \mid D) \, \mathrm{d}\theta \; - \mu_\theta.$$

2.1 Credible Intervals & High Density Regions

Treating the parameters as random variables, makes Bayesian inference a perfect candidate for dealing with *uncertainty*. In the presence of noisy or incomplete data, a large range of different parameters might have similar posterior density and are thus considered equally likely. While in classical, frequentist inference, such information is given by the means of confidence intervals, Bayesian statistics use *credible intervals*. Given some level $\alpha \in [0, 1]$, an α-credible interval is an interval $[a, b]$ in which an α fraction of the density is located. Formally, we write

$$\int_a^b \pi(\theta \mid D) \, \mathrm{d}\theta = \alpha.$$

Usually credible intervals are defined in a symmetric manner, such that they have equally "heavy" tails, i.e. a and b are chosen s.t.

$$\int_{-\infty}^a \pi(\theta \mid D) \, \mathrm{d}\theta = \int_b^{-\infty} \pi(\theta \mid D) \, \mathrm{d}\theta = \frac{1-\alpha}{2}.$$

As the integrals cannot be solved analytically, the i-th marginal credible interval is computed numerically by sorting the generated samples from the posterior distribution along the i-th dimension and taking the $\lfloor n(1-\alpha)/2 \rfloor$-th and $\lceil n(1+\alpha)/2 \rceil$-th elements as lower and upper bounds, i.e.

$$[a_i, b_i] = [\theta_{\lfloor n(1-\alpha)/2 \rfloor}, \theta_{\lceil n(1+\alpha)/2 \rceil}]$$

assuming $\theta_{1,i} \leq \theta_{2,i} \leq \ldots \leq \theta_{n,i}$.

However, in the presence of multimodal distributions, credible intervals may cover low-probability regions as illustrated in Figure 2.1. An alternative approach are *high density regions*. Following [41], an α-high density region R is a subset of the parameter space, s.t.

$$\pi(\theta \mid D) > p_\alpha, \quad \forall \, \theta \in R$$

and where p_α is the largest value, s.t.

$$\int_R \pi(\theta \mid D) \, \mathrm{d}\theta \geq \alpha.$$

Given samples $\theta_1, \ldots, \theta_n$ from the posterior distribution, the value p_α can be estimated by sorting their posterior densities, s.t. $\pi(\theta_1 \mid D) \leq \ldots \leq \pi(\theta_n \mid D)$, and taking

$$\hat{p}_\alpha := \pi(\theta_i \mid D), \quad \text{with } i = \lfloor (1-\alpha)n \rfloor$$

as an estimator. For $n \to \infty$ the estimator $\hat{p}_\alpha \to p_\alpha$ [41]. The parameters with density larger than $\hat{p}_\alpha$ then form a set of samples from the high density regions. For the one-dimensional marginal distributions, high density regions can be easily computed as unions of intervals, where consecutive samples, sorted along the respective dimension, with $\theta_i \in \hat{R} := \{\theta : \pi(\theta \mid D) > \hat{p}_\alpha\} \subseteq \theta_1, \ldots, \theta_n$ form part of the same interval.

Unlike confidence intervals, credible intervals and high density regions are uniquely defined w.r.t. the posterior distribution π. Marginal credible intervals can be computed exactly, if the

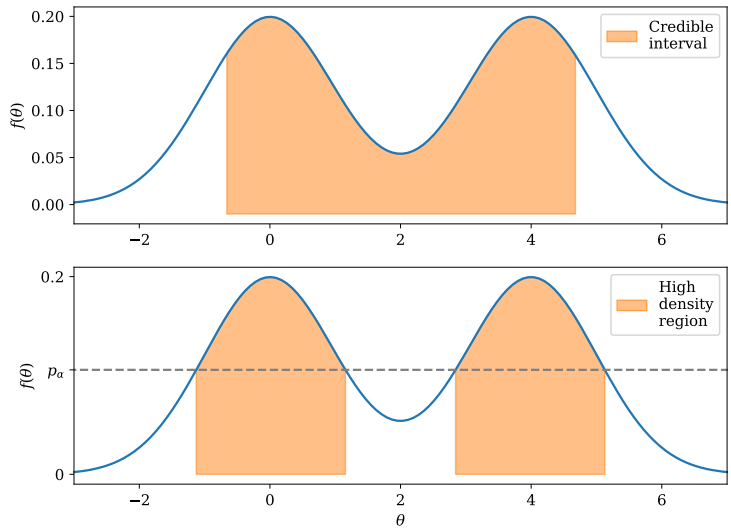


Figure 2.1: Comparison between a credible interval and high-density regions for a Gaussian mixture distribution, both computed for $\alpha = 0.75$. Unlike the credible interval, the high density region does not comprise the low density valley around $\theta = 2$.

marginal posterior distributions can be computed exactly. If the posterior distribution can only be approximated via numerical sampling, they can be numerically estimated using the samples. Nevertheless, there is no structural approximation error underlying this estimation and instead the estimation using samples will converge asymptotically towards the exact credible intervals.

Just as confidence intervals, credible intervals depend strongly on the data. However, by not assuming that there exists such thing as a unique, true, but unknown parameter, credible intervals do not risk not including this parameter. Instead, credible intervals have to be interpreted differently. An α-credible interval contains a random parameter taken from the posterior with probability α. For an α-high density region an even stronger statement can be made: A random parameter taken from the posterior will be contained in an α-high density region with probability α and, given it is contained, will be able to explain the data better than another randomly drawn parameter with probability $1 - \alpha$.

2.2 Marginal Posterior Densities

Another, visual approach to explore posterior distributions is by plotting the marginal distributions of the numerically obtained samples in form of histograms. The marginal posterior distribution w.r.t. to some subset θ^+ of the parameter set θ is defined as

$$\pi(\theta^+ \mid D) = \int \pi(\theta^- \mid D)\, \mathrm{d}\theta^-,$$

where $\theta^- = \theta \setminus \theta^+$. Although this integral may be high-dimensional, it can be easily computed from a given sample of θ by just omitting the part of each sample corresponding to θ^-.

In contrast to credible intervals or high-density regions, which try to summarize key features of the distribution within a few statistics, visual analysis allows to perceive features, which

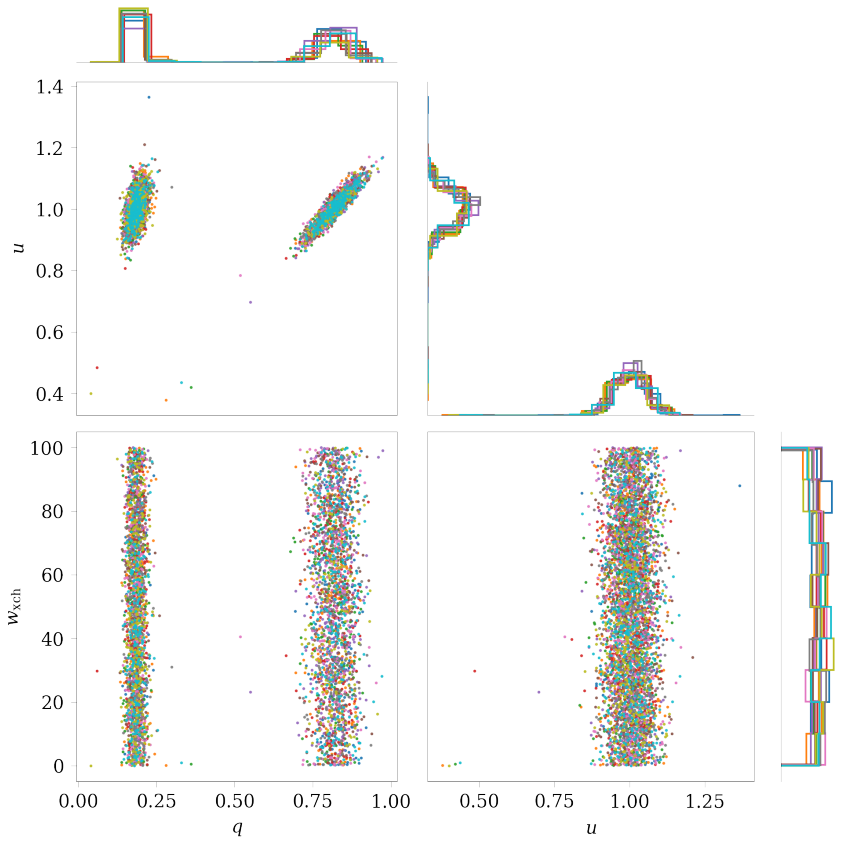


Figure 2.2: Marginal posterior densities of the STAT-2-ni model described in Chapter 9 evaluated using 10 parallel Markov chains. The w_{xch} parameter is visibly non-identifiable while the q parameter admits two distinct modes.

are not so easily characterised quantitatively, like e.g. skewness or multimodality. Obviously, plotting high-dimensional distributions is not feasible in practice, thus such visual approaches have to be limited to low-dimensional marginals, typically considering only 1- and 2-D marginal distributions. An example of such a plot is given in Figure 2.2.

2.3 Inference for Dynamical Models

In the natural science, many systems of interest can be modeled as dynamic systems given as a set of ordinary differential equations

$$\frac{\mathrm{d}X}{\mathrm{d}t} = u(X, t, \theta) \tag{2.3}$$

Popular examples from the life sciences are the Lotka-Volterra model for predator-prey interactions or the SIR model for spread of infectious diseases. In such models, parameters typically control the interaction between species. For example, the R_0 value in the SIR model controls the infectiousness of a disease. Such interactions cannot be measured directly, but have to be induced from collected data, e.g. the number of new infections per unit of time, known as the *incidence*. Clearly, the parameters and the data live on different spaces and, thus, a mapping from the parameter space to the data space is needed, in order to evaluate the quality of a parameter estimate in the light of the data. For dynamic models, such a mapping is obtained

from a *forward simulation* of the model, which is performed by numerical integration of the dynamic equations (2.3). The forward simulation then yields a trajectory of $X(t)$, which in case of the SIR model gives the numbers of susceptible, infected and recovered persons in the population throughout time. The number of new infections per unit of time can then be computed by taking the appropriate differences on the number of infected persons. We refer to this incidence, which was computed based on the forward simulations, as *simulated data*. In order to find a parameter θ, which "minimizes the distance" between observed and simulated data, a *loss* on the data is imposed. We denote the procedure of computing the simulated data from θ as $f(\theta)$. A commonly chosen loss is the *sum of squared residuals*, which is defined as

$$\ell_{\mathrm{SSR}}(\theta) = \frac{1}{2}\| D - f(\theta)\|^2_{\Sigma^{-1}}. \tag{2.4}$$

Here, Σ is assumed to be a symmetric positive definite matrix containing measurement errors of the data D. Note, that every evaluation of the loss of a particular parameter θ requires the whole procedure of forward integration of the dynamic equations and computation of simulated data. This effort introduces potentially significant, additional costs to the numerical treatment of posterior distributions of dynamic models.

Once given such a loss function, a likelihood model can be simply defined as

$$\mathcal{L}(\theta \mid D) = \exp\{-\ell_{\mathrm{SSR}}(\theta)\} = \exp\left\{-\frac{1}{2}\| D - f(\theta)\|^2_{\Sigma^{-1}}\right\}, \tag{2.5}$$

which is just the unnormalized density of a multivariate Gaussian distribution as already given in (2.2). Hence, using the sum of squared residuals as a loss function is equivalent to assuming the data being normally distributed around some "true", unknown mean. Again note, that the additional, computational costs introduced from the loss evaluation will also enter the likelihood. Furthermore, observe that the parameters θ are not normally distributed. Instead, the shape of the likelihood as a function of the parameters can be highly complexly shaped, admitting multiple modes.

In order to make the above problem formulation amenable to the Bayesian inference framework introduced previously, it remains to define a prior on the parameter values. Thinking back about the SIR model, it is clear that the R_0 parameter has to be non-negative. Clearly, this insight is absolutely independent of any data observation and thus clearly known *a priori*. To reflect such knowledge in the prior, we may e.g. just choose $p(\theta) = \mathbb{1}_{\theta \geq 1}$, which defines an *improper* prior, since it is not integrable. However, such priors may still be permissible, if the posterior distribution remains integrable. If integrability is unclear, setting defensive upper bounds on the parameters will make the distribution definitely integrable w.r.t. the Lebesgue measure (which is the standard case). This clearly comes at the cost of truncating the parameter space and thus possibly excluding valid parameter estimates. Thus, bounds should be well justified. The constrained space of admissiable solutions is then also referred to as the *feasible space*.

Studying dynamical systems in their equilibrium can lead to explicit dependencies between parameters [89]. In particular, if the dynamic equations were linear, then parameters may depend upon another linearly. In this case, the dimension of the parameter space may decrease, but the remaining parameters will live on a linearly constrained space, a *polytope*. How such linearly constrained spaces arise, will be demonstrated in detail in Chapter 3.

Although bounds are, as demonstrated, easily included into the posterior distribution, they introduce new numerical challenges, as we will discuss in more detail in Chapter 5. In particular, algorithms used to sample the posterior distribution may get stuck, if they are close to the borders of the feasible parameter space. Further, evaluating the likelihood using parameters

from outside the feasible space, might lead to undefined expressions, if for example the logarithm of a positive parameter has to be computed. Thus, making sure to first check the feasibility of the parameter value and then evaluating the likelihood is an important guideline in practice.

Finally, another issue which may exacerbate Bayesian inference for dynamic models are *non-identifiabilities*. Informally put, a parameter is considered non-identifiable, if the data does not help to identify its value. For example, coming back once more to our example of an infectious disease, the color of shoes a particular person wears is unlikely to influence their infection risk. Thus, if we consider the color of shoes as a parameter, it will be non-identifiable. Although obvious in this example, detecting non-identifiable parameters in dynamic models a priori is a challenging task [17]. Since the posterior density is invariant under changes in a non-identifiable parameter, the density might not be integrable in presence of non-identifiabilities. As mentioned before, bounding parameters can overcome this issue, nevertheless, non-identifiabilities can make the problem of sampling the posterior distribution much harder, since parameters may live on largely differing scales. We will demonstrate and discuss this in more detail in Chapter 10.

In summary, Bayesian inference can be applied in a straight-forward manner for parameter estimation of dynamic models. However, the following issues can make sampling of the posterior distribution much harder:

- Additional costs from forward simulations, which may require solution of dynamic equations.
- Constrained parameter spaces, impeding the usage of regular sampling algorithms.
- Non-identifiabilities, making the posterior non-integrable or again impeding the usage of regular sampling algorithms.

Chapter 3

Biological Network Modeling

Their list ranges from food webs [68], describing large scale ecological networks, over epidemiological models [3, 92] down to microbial [32] and biochemical reaction networks [67, 102], which aid in understanding and describing the inner workings of cells and organisms.

Networks are ubiquitous in the computational life sciences [26]. From large-scale ecosystems and food webs [68] to infection models [3, 92] down to molecular reaction and signaling networks [6, 67], they serve as a powerful modeling tool to represent and analyze complex relationships and interactions between the respective elements of the network. All of the aforementioned models are dynamic models, meaning that the behaviour of the respective system over time is described by a set of (ordinary) differential equations. These equations also define the interactions of the network's elements and thus the network's topology.

Throughout this chapter, we will introduce *metabolic network* models as an example of a particular class of dynamical biological network models, which admit two particular characteristics that hamper straight-forward Markov chain Monte Carlo sampling for Bayesian inference:

- a costly to evaluate forward simulation and
- a complexly shaped, linearly constrained domain.

After discussing the modeling technique, we consider the estimation of unknown parameters of such models, which mainly consist of the reaction rates, commonly called *fluxes*.

3.1 Metabolic Flux Analysis

A *metabolic network* is a model of an organism's metabolism. It describes what metabolites, like glucose or O_2 are, are turned over into which other metabolites and by what reaction. Further, also transport effects over compartment boundaries or cell walls may be described by what then is referred to under the more general term of a *flux*. The compartments and metabolites then form the nodes of the networks, which are usually referred to as *pools*. The construction of metabolic networks is itself a laborious task and many disciplines within the system's biology deal with their setup [67, 90]. An example network used for characterization of the carbon-nitrogen metabolism in *Mycobacterium tuberculosis* [10] is depicted in Figure 3.1.

However, even once given a metabolic network, this network still only depicts the potential of the organism, but not necessarily what actually happens within a cell when interacting within its environment. *Metabolic flux analysis*, in style of the other Omics fields also referred to as *fluxomics*, deals with determining the actual rates at which reactions happen within an

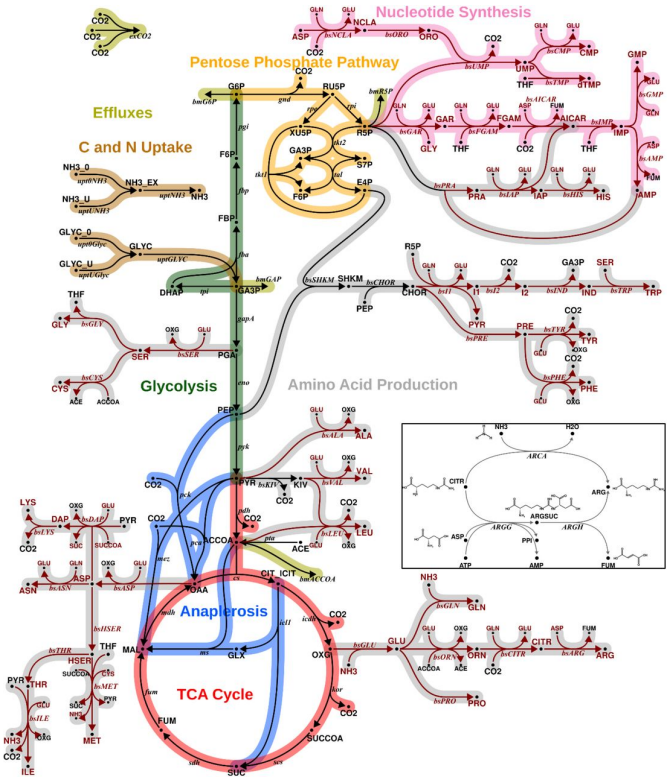


Figure 3.1: Model of the central carbon and nitrogen metabolism of *Mycobacterium tuberculosis* [10].

organism. Unfortunately, most intracellular fluxes cannot be measured directly. *^{13}C-metabolic flux analysis* is a particular technique which infers the fluxes by using isotope labeling data, typically using ^{13}C isotope labels, though other isotopes have also been used [10]. Unlike other techniques such as flux balance analysis [62], it does not rely on simplifying assumption, but instead tries to gain insights on the fluxes from the given data. Recently, Bayesian metabolic flux analysis has arised as a an application of Bayesian inference to metabolic flux analysis [85, 87].

3.2 Feasible Flux Space

Throughout this section, we will introduce ^{13}C-metabolic flux analysis by means of a simple example organism defined by the following reaction equations.

$$\begin{aligned} u &: S \to M \\ x &: M \to P \\ y &: M \to Q \\ z &: M \to R \end{aligned} \tag{3.1}$$

For simplicity, this example omits non-linear reactions[1], which would be introduced by considering reactions, that synthesize various reactants into a product. For a more specialized treatment, we refer the interested reader to [102]. The equations read as reaction u converting a unit of

[1]The term non-linear might seem inadequate here, but becomes clear when considering, that such reactions introduce non-linear differential equations in (3.4) when simulating isotope labeling experiments.

the compound S into a unit of the compound A and so on. The relations between reactions and compounds given by the reaction equations can be visualized in form of a reaction network as given in Figure 3.2. In order to treat the reaction network mathematically, the information displayed in the reaction network can also be represented as a matrix, the *stoichiometric matrix*

$$\hat{\mathcal{S}} = \begin{array}{c} \\ S \\ M \\ P \\ Q \\ R \end{array} \begin{array}{c} \begin{array}{cccc} u & x & y & z \end{array} \\ \begin{pmatrix} -1 & & & \\ 1 & -1 & -1 & -1 \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \end{array}.$$

The stoichiometric matrix stores the stoichiometric coefficients of every reaction, which in our example are all 1. In fact, the stoichiometric matrix is the adjacency matrix of the Levi graph of the reaction network, which in the general case is a hypergraph[2]. The stoichiometric matrix can now be used to state the dynamics of the reaction network. Given a flux vector $\nu = (u, x, y, z)$, the change of metabolites over time is given as

$$\frac{\mathrm{d}X}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}(S, M, P, Q, R)^\top = \hat{\mathcal{S}}\nu,$$

which expanded gives us

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}t}S &= -u, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}M &= u - x - y - z, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}P &= x, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}Q &= y, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}R &= z. \end{aligned} \tag{3.2}$$

In order to restrict the space of possible solutions ν, attention is often restricted to solutions which lead to a steady state within the organism's metabolism. The biological motivation behind this assumption is the concept of *homeostasis*, in which an organism is assumed to maintain an internal steady state [102]. Mathematically, the steady state assumption reads as

$$\frac{\mathrm{d}X}{\mathrm{d}t} = \hat{\mathcal{S}}\nu = 0,$$

which however considering the above system of equations (3.2) only allows for the trivial solution $\nu = 0$. The problem here are obviously the boundary metabolites S, P, Q and R, whereas for the metabolite pool M, the assumption may be read as *"what goes in, goes out"*. In order to fix the boundary metabolite pools, they are assumed to be infinite, meaning they can neither deplete nor overflow. Formally, this assumption is represented by dropping the respective equations from (3.2) and thus from $\hat{\mathcal{S}}$. The new stoichiometric matrix with missing boundary metabolites is referred to as *reduced stoichiometric matrix*

$$\mathcal{S} = \begin{array}{c} \\ M \end{array} \begin{array}{c} \begin{array}{cccc} u & x & y & z \end{array} \\ \begin{pmatrix} 1 & -1 & -1 & -1 \end{pmatrix} \end{array}$$

and thus the steady state condition becomes

$$u = x + y + z.$$

[2]If a reaction merges or splits compounds into several other compounds, the edges of the corresponding reaction network become hyperedges, connecting all the involved compounds.

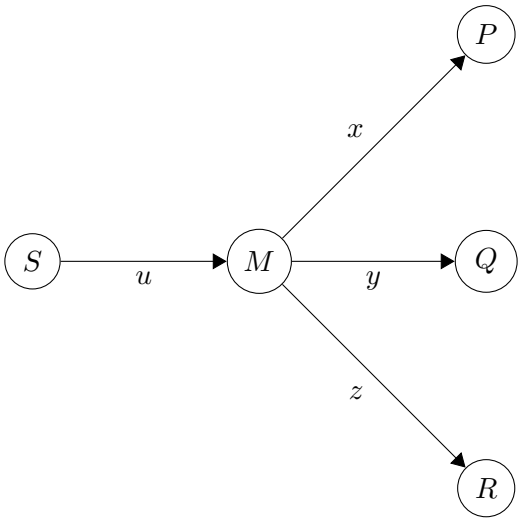


Figure 3.2: Reaction network of the system given by (3.1).

Often prior knowledge may exist on the direction in which a reaction happens. Although we stated the reactions given in (3.1) as nominally directed, they do not have to be in general. Thermodynamic considerations may however support assuming non-reversibility of reactions. For our example model, we shall assume all reactions to only happen in the nominal direction, requiring $\nu \geq 0$. Once we now know, for example, u, x and y, we could compute

$$z = u - (x + y)$$

and thus convert the equality constraint into an equivalent inequality constraint as

$$x + y \leq u.$$

In real experiments, the uptake rate u of the substrate S is often directly measurable and, if so, the other rates may be given relatively to the uptake, s.t. we arrive at the inequality constraint

$$x + y \leq 1$$

Mathematically, removing fluxes, which may be recovered as a linear combination of the remaining fluxes, correspond to computing a base of the flux space defined by $\mathcal{S}\nu = 0$. Although arbitrary bases may be chosen here, e.g. orthogonal bases, it is common to choose a subset of fluxes to form a basis, which are then referred to as the *free fluxes* ν_{free}. So, given a basis matrix K and possibly a shift vector p, we compute the fluxes $\nu = K\nu_{\text{free}} + p$ from the free fluxes. For our example model, we have

$$K = \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{pmatrix} 1 & \\ & 1 \\ -1 & -1 \end{pmatrix} \end{array} \quad \text{and} \quad p = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Taking all these inequality constraints now into consideration, we arrive at a constrained space of feasible fluxes

$$\mathcal{P} = \{\nu_{\text{free}} : A\nu_{\text{free}} \leq b\}, \tag{3.3}$$

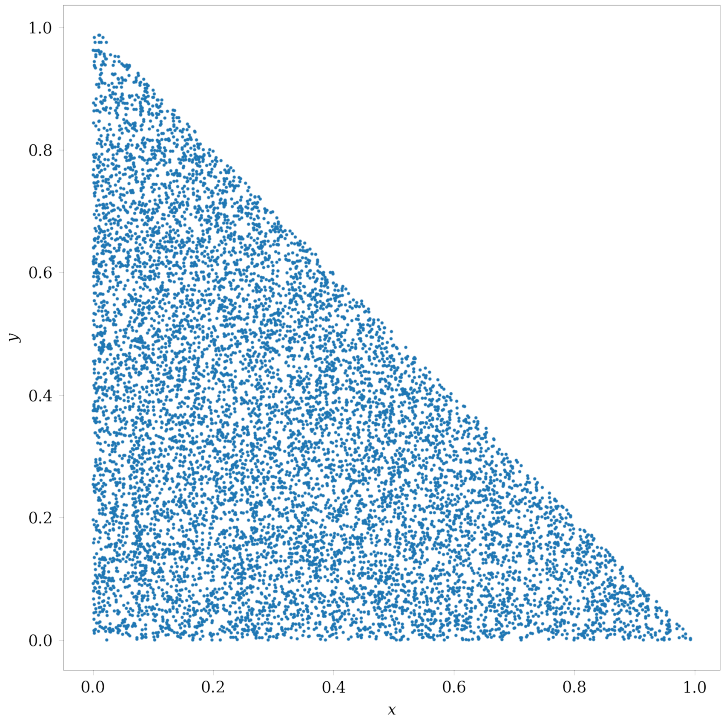


Figure 3.3: Uniform sampling of the feasible flux space $\mathcal{P}$ (3.3) of the simple reaction network in Figure 3.2, which is just a two-dimensional simplex.

with

$$A\nu_{\text{free}} = \begin{pmatrix} 1 & 1 \\ -1 & \\ & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = b,$$

the *feasible flux space*.

Although we have gained some information about the fluxes already, the feasible flux space still consists of uncountably many possible solutions. To further shrink the set of meaningful solutions, more data is acquired by performing isotope labeling experiments. Incorporating this data into computational procedures requires further modeling work, which is introduced in the upcoming section.

3.2.1 Bidirectional Fluxes

In the above example, we assumed all reactions to be irreversible and to only happen in the nominal direction. As we briefly mentioned, reactions however can be bidirectional, meaning that two adjacent pools exchange material continuously. However, modeling a bidirectional reaction by a single flux does not incorporate this information, but instead only accounts for the net direction of the flux. Thus, a bidirectional flux v may be considered as two unidirectional fluxes $v^{\rightarrow}$ and $v^{\leftarrow}$ with opposite nominal directions. However, as the two resulting fluxes cannot be observed independently from each other, another transformation is introduced, which separate the bidirectional flux into a *net flux* $v_{\text{net}} = v^{\rightarrow} - v^{\leftarrow}$ and *exchange flux* $v_{\text{xch}} = \min\{v^{\rightarrow}, v^{\leftarrow}\}$ [102].

Bidirectionality assumptions can have major impacts on the estimated net values, which are the real quantities of interest. At the same time, exchange fluxes are often non-identifiable

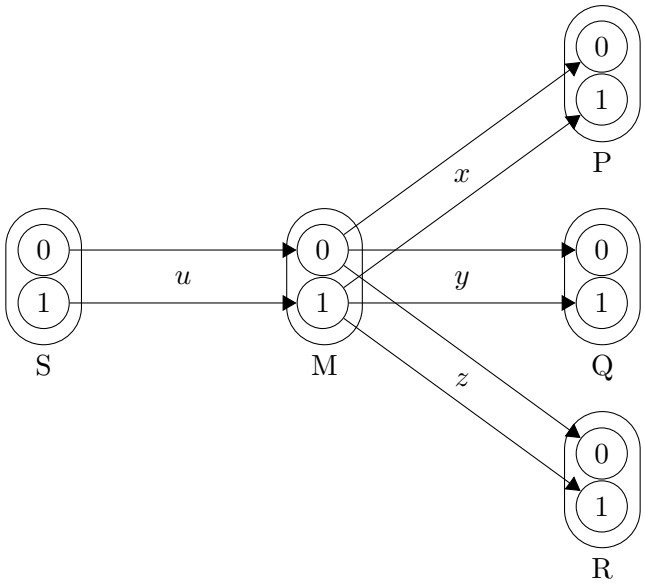


Figure 3.4: Exemplary atom transition network of the reaction network given by (3.1), where it is assumed that every metabolite consists of two carbon atoms.

and extensive computational analyses can become overly expensive, as every exchange flux increments the dimensionality of the problem [88].

3.3 Isotope labeling Experiments & Atom Transition Networks

In isotope labeling experiments, isotopically labeled substrate is fed into the organism. By measuring the isotopic enrichment of the organism's metabolites, data is collected which a meaningful flux solution should be able to explain. In order to simulate the outcome of an isotope labeling experiment, the metabolic model has to be extended to incorporate atom transitions. An atom transition network represents what atoms of the reactants get mapped to which atoms of the product for every reaction from the metabolic model. As usually only specific elements, most commonly carbon atoms, get isotopically labelled, the atom transition network only has to correctly represent those transitions. In practice, carbon atoms usually get labelled using the stable ^{13}C isotope.

Getting back to our example metabolic model, we will assume that all metabolites consist of two carbon atoms. Within this model, every metabolite can now occur in 4 different states: having no label, having either the first or the second atom labelled or having both of them labelled. These different states of the same molecule are called *isotopomers*. We denote the isotopomer fractions of a given metabolite using binary indices, e.g. m_{00} denotes the fraction of unlabelled metabolites M, m_{10} the fraction of metabolites labelled at the first atom, and so on. Given the atom transition network from Figure 3.4, the change of the isotopomer fractions of the metabolite pool M is given by

$$|M| \frac{\mathrm{d}m_{ij}}{\mathrm{d}t} = u\, s_{ij} - (x\, p_{ij} + y\, q_{ij} + z\, r_{ij}), \quad \text{for } i, j = 0, 1 \tag{3.4}$$

and where $|M|$ is supposed to denote the total concentration of metabolite M within the metabolism, called the *pool size*. Note, that pool sizes cannot be measured for all metabolites.

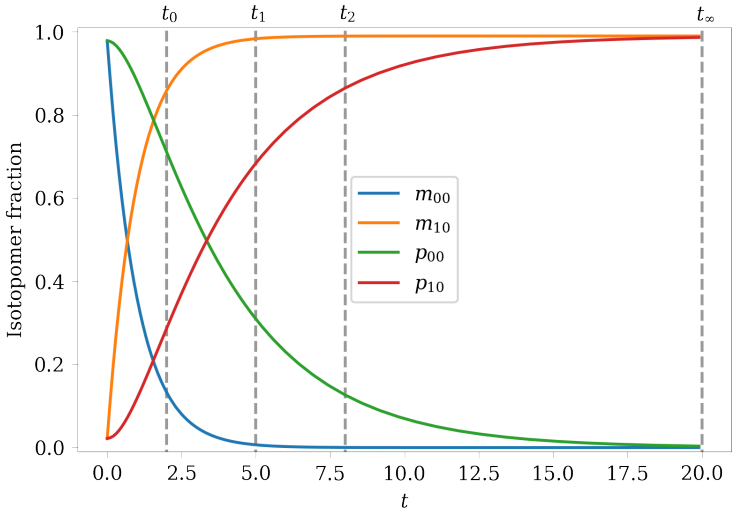


Figure 3.5: Isotopomer fractions over time for the example network defined in (3.1), using the atom transition network in Figure 3.4. The substrate was labelled with $s_{10} = 1$ and the fluxes were set to $(x, y) = (0.3, 0.2)$. The pool sizes were all set to 1. Isotope labeling measurements were taken at the indicated timestamps t_0, t_1, t_2 and t_∞.

Thus, they may enter the problem of determining fluxes as additional free parameters, which also have to be inferred. Similarly, dynamic equations can be deduced for all other metabolites, resulting in an *isotope labeling system*. The initial labeling of the substrate is part of the experimental setup of an isotope labeling experiment and is, therefore, given. As before, the substrate pool is assumed to be infinitely large, thus new isotopes may enter the system continuously. In contrast, the initial labeling of the other metabolites is usually assumed to be completely unlabelled and gets enriched with isotopes throughout time. Given a set of fluxes and in addition also the pool sizes, the isotopomer fractions can now be simulated over time. An example trajectory of the isotopomer fractions over time is given in Figure 3.5.

The number of isotopomer fractions for every metabolite scales exponentially with 2^n in the number of (carbon) atoms n in the metabolite. The total number of fraction, which have to computed, is thus the sum over the number of isotopomer fractions for all metabolites of interest. Although the number of carbon atoms in typical metabolites is usually not greater than 7, the number of isotopomer fractions becomes 2^{n+m}, when for example two metabolites with n, m carbon atoms respectively enter a reaction. Luckily, the number of isotopomer fractions does still not grow exponentially in the number of carbon atoms modeled within the whole system, as the reaction network is usually sparse and reactions comprise just a few metabolites. Nevertheless, system sizes may rise fast as the overall number of reactions increases [100]. Modern solvers apply network reductions, which remove parts of network, that do not influence the metabolites of interest [99]. In particular, from an inference perspective, we are only interested in the metabolites, for which we have observed data.

Nevertheless, solving the dynamical system established by the isotopomer fractions is typically considered expensive when doing inference of fluxes. Instead, another steady state assumption, this time on the isotopic enrichments, is imposed. To put it simply, the idea behind such an assumption is that the isotopically labeled material will propagate through the network until it reaches some equilibrium configuration. The existence of such a steady state for the isotope labeling systems was proven under very weak conditions [104], which usually hold in practice.

Formally, the isotopical steady state assumptions sets the change of isotopomer fractions 3.4 throughout time to 0, i.e.

$$\frac{\mathrm{d}m_{ij}}{\mathrm{d}t} = 0 \quad \Longrightarrow \quad u\, s_{ij} - (x\, p_{ij} + y\, q_{ij} + z\, r_{ij}) = 0, \quad \text{for } i, j = 0, 1, \tag{3.5}$$

which results in a (in general non-linear)[3] system of algebraic equations, which can be solved much more efficiently than the dynamic isotopic labeling systems [103]. Note, that in this case, no pool sizes are required, which may effectively reduce the number of unknown parameters. If isotopic steady state is assumed, the isotope labeling system is commonly referred to as *stationary*, consequently, it is termed *instationary* in the more general case, when differential equations are involved.

3.4 Flux Estimation

Of course, the fluxes cannot be known a priori, as determining them is the goal we try to achieve. However, throughout the course of an isotope labeling experiment, isotopomer fractions can be measured at different points in time using nuclear magnetic resonance (NMR) or mass spectroscopy (MS). Given such measurements and some adequate loss function as introduced in Chapter 2, the quality of a *flux estimate*, i.e. an arbitrary guess of free fluxes from the feasible flux space, can be quantized by a scalar value. If the underlying problem requires simulation of a instationary isotope labeling experiment, the parameter vector is also required to include the pool sizes. For generality, we, thus, denote the full parameter vector as θ. Since the pool sizes are usually of little interest and their existence as parameters is a mere technical artifact, we still refer to θ as a flux estimate. Typically, the loss function employed is the sum of squared residuals

$$\ell_{\mathrm{SSR}}(\nu_{\mathrm{free}}) = \frac{1}{2}\big\|\eta - f(\theta)\big\|^2_{\Sigma^{-1}}, \quad \text{with } \Sigma^{-1} = \mathrm{diag}(1/\sigma^2), \tag{3.6}$$

where $\eta = (\eta_1, \ldots, \eta_n)^\top$ are the measured isotopomer fractions with their respective measurement errors $\sigma = (\sigma_1, \ldots, \sigma_n)^\top$ and $f(\theta)$ is a condensed notation for simulating an isotope labeling experiment for the metabolic model at hand using some flux estimate θ. We assume here, that the output of the forward simulation $f(\theta)$ aligns correctly with the given measurements η.

Clearly, this loss function paired with the transformation introduced in Section 2.3 yields a likelihood model. In order to make the flux estimation problem amenable to Bayesian inference, it remains to choose an appropriate prior. Again, as discussed before, the prior can be used to restrict the space of feasible solutions. A natural choice is thus the uniform distribution on the feasible flux space $\mathcal{P}$, which then has unnormalized density $p(\theta) = \mathbb{1}_{\mathcal{P}}(\theta)$. Equipped with likelihood and prior, the unnormalized posterior density of a flux estimation problem becomes

$$\pi(\theta\,|\eta) = \frac{1}{2}\big\|\eta - f(\theta)\big\|^2_{\Sigma^{-1}}\, \mathbb{1}_{\mathcal{P}}(\theta). \tag{3.7}$$

Although conceptually straight-forward, this problem specification admits the same practical issues discussed before in Section 2.3. In particular, posterior distributions of fluxes have been observed to admit interesting shapes and being in fact far from normally distributed as we show in Figure 3.6.

[3] As mentioned before, any reaction consisting of multiple metabolites being synthesized into a product would introduce non-linear terms here.

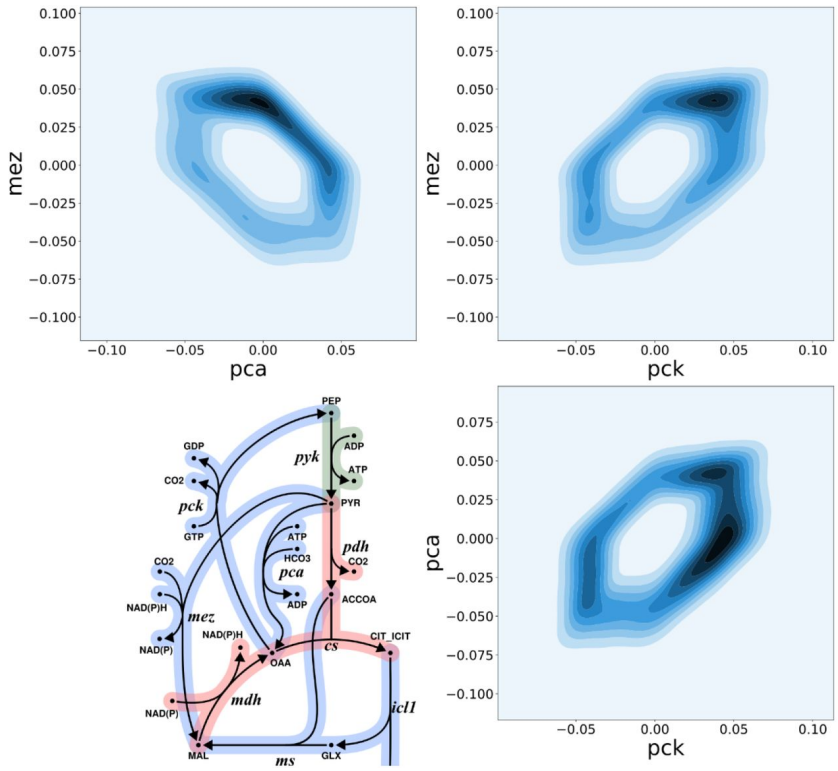


Figure 3.6: Joint marginal posterior distribution of fluxes taken from [10].

Chapter 4

Markov Chain Monte Carlo Methods

Markov chain Monte Carlo (MCMC) methods are pseudo-random numerical simulations of Markov chains, a class of randomized processes that converge towards a unique equilibrium distributions under mild conditions. Markov chains are easily constructed and, using the Metropolis-Hastings algorithm, can be used to generate samples from arbitrary distributions at the cost of correlations between consecutive samples. Thus, they are one of the computational working horses in Bayesian statistics, as they allow to produce samples from arbitrarily complex posterior distributions. These samples can then be used to estimate quantities of interest as mentioned in Chapter 2, which is a form of numerical integration procedure also known as Monte Carlo integration.

The main caveat of Markov chain Monte Carlo methods is the aforementioned issue of correlated samples, which reduces the statistical efficiency of the samples, thus requiring to produce many. As we will see throughout this chapter, not all Markov chains are equivalent, even if they converge towards the same equilibrium distribution. Thus, much work has been put into developing efficient algorithms [19, 20, 18, 52].

Throughout this chapter, we will start by introducing the basic definitions of Markov chains and state the most important convergence results, which are needed to successfully employ them for sampling arbitrary distributions. In Section 4.2, we introduce the famous Metropolis-Hastings algorithm. Given a pointwise evaluable, unnormalized probability density, the Metropolis-Hastings algorithm allows to construct Markov chains, which produce samples from the desired distribution defined by the given density. As we will see, the Metropolis-Hastings algorithm may rather be seen as a meta-algorithm, allowing for great freedom in designing proposal algorithms, which will all converge asymptotically. As simulations only run finitely long, assessing the speed of convergence and the approximation error at the end of the simulation are crucial to obtain reliable simulation results. Some diagnostics will be considered at the end of this chapter.

4.1 General State Discrete Time Markov Chains

A *Markov chain* is a stochastic process, where the probability of transitioning from the current state into another state depends only on the current state. This property is also often referred to by saying that the process has *no memory*, as any previously visited states have no influence on the evolution of the chain in the future. The chain's states live in some general state space $\mathcal{S}$, which together with its Borel-σ-algebra $\mathcal{B}_\mathcal{S}$ forms a measurable space $(\mathcal{S}, \mathcal{B}_\mathcal{S})$. The behaviour of a chain is almost exclusively controlled by the transition probabilities, which are

summarized in what is called a *transition kernel*. Formally, a transition kernel is defined as map $\mathcal{K} : \mathcal{S} \times \mathcal{B}_{\mathcal{S}} \to [0, 1]$, for which

- for any given state $x \in \mathcal{S}$, $\mathcal{K}(x, \cdot)$ is a probability measure on $\mathcal{B}_{\mathcal{S}}$, so
$$\mathcal{K}(x, \mathcal{S}) = \int_{\mathcal{S}} \mathcal{K}(x, \mathrm{d}y) = 1,$$
- for any given measurable set $A \in \mathcal{B}_{\mathcal{S}}$, $\mathcal{K}(\,\cdot\,, A)$ is a non-negative measurable function on $\mathcal{S}$.

Equipped with this definition, a general state discrete time Markov chain is a countable sequence of random variables $X_1, X_2, \ldots \in \mathcal{S}$ with initial distribution μ and transition kernel $\mathcal{K}$ obeying the following two properties:

- The starting point X_1 is distributed with law μ, i.e.
$$\mathbb{P}(X_1 \in A) = \mu(A) = \int_A \mu(\mathrm{d}x), \quad \forall A \in \mathcal{B}_{\mathcal{S}}\,.$$
- For $X_i = x$, the law of X_{i+1} conditional on X_i is given by the transition kernel $\mathcal{K}$ as
$$\mathbb{P}(X_{i+1} \in A | X_i = x) = \mathcal{K}(x, A) = \int_A \mathcal{K}(x, \mathrm{d}y)$$
and is independent of $X_1, \ldots, X_{i-1}$, i.e.
$$\mathbb{P}(X_{i+1} \in A | X_i = x, X_{i-1} = x_{i-1}, \ldots, X_1 = x_1) = \mathbb{P}(X_{i+1} \in A | X_i = x).$$

The particular property of Markov chain's needed to become applicable for Bayesian inference is convergence towards an *invariant distribution*, which gives the probability to observe the chain within a particular state. In order to admit convergence, the chain needs to be *aperiodic, irreducible* and *positive recurrent*. We will anticipate the following technical results by giving an intuitive overview over their implications:

Aperiodicity A Markov chain is called *aperiodic*, if it admits no periodicity in visiting disjoint subsets of the state space. An example for a periodic chain is the random walk on the natural numbers, where the chain may only visit its two neighbouring states. As such, if the chain starts on an odd number, it may only visit even numbers on odd steps afterwards and odd numbers on even numbers of steps. Thus, the state space is partitioned into two disjoint subsets, which are visited alternatingly.

Irreducibility A Markov chain, that cannot get stuck in some subset of the state space, is called *irreducible*. That is, any subset of the state space is reachable from any other subset of the state space. Together with aperiodicity, irreduciblity forms an important requirement for the convergence of the chain towards the desired distribution.

Positive Recurrence A Markov chain, where the expected time to return to a particular state – called the return time – is finite for any state, is called *positive recurrent*. Positive recurrence is an important requirement for the invariant distribution being well-defined as a finite measure. As we will see, ensuring positive recurrence is less an algorithmic issue, as in the setting of Bayesian inference the invariant distribution is given by the practitioner. The practitioner, thus, has to ensure that their invariant distribution is well-defined in admitting an integrable density function.

4.1.1 Basic Properties & Convergence

The following definitions and results are taken from [57] and [74]. Since in Bayesian inference, the invariant distribution is usually already given in form of the posterior, we start off by discussing the properties required to guarantee convergence of the chain towards its invariant distribution, if it exists. Afterwards, we will consider what could go wrong, such that we end up constructing a Markov chain, which does not admit an invariant distribution. In particular, the accidental construction of such Markov chains in Bayesian inference links to the posterior density not being integrable.

A measure π is called an *invariant measure*, if

$$\pi(A) = \int_{\mathcal{S}} \pi(\mathrm{d}x)\, \mathcal{K}(x, A) = (\pi\, \mathcal{K})(A), \quad \forall A \in \mathcal{B}_{\mathcal{S}}\,. \tag{4.1}$$

It is called an *invariant distribution*, if further

$$\pi(S) = 1.$$

The two most important results on invariant distributions for Markov chains are those of existence and convergence. Both results rely on the concepts of irreducibility and (positive) recurrence. Before introducing them, we quickly define the n-step transition kernel recursively as

$$\begin{aligned} \mathcal{K}^n(x, A) &= \int_{\mathcal{S}} \mathcal{K}^{n-1}(x, \mathrm{d}y)\, \mathcal{K}(y, A) \\ \mathcal{K}^0(x, A) &= \delta_x(A) \end{aligned} \tag{4.2}$$

Formally, a Markov chain is called *φ-irreducible* [74], if there exists a measure φ, s.t.

$$\varphi(A) > 0 \implies \exists n : \mathcal{K}^n(x, A) > 0, \quad \forall x \in \mathcal{S},\ \forall A \in \mathcal{B}_{\mathcal{S}}\,.$$

The measure φ can thus be understood as indicating the reachability of a set A from any other state. A simple example of an φ-irreducible chain is a Gaussian random walk on the real number line, defined as

$$x_{i+1} = x_i + \xi, \quad \text{with } \xi \sim \mathcal{N}(0, \sigma^2),$$

where $\varphi = \lambda$ is the Lesbesgue-measure, as any set with positive Lebesgue-measure may be reached by the process.

A Markov chain is called *periodic* with periodicity p [74], if there exists a partition $\{\mathcal{S}_i\}_{i=1}^p$ of the state space $\mathcal{S}$, s.t. $\mathcal{K}(x, S_{i+1}) = 1$ for all $x \in \mathcal{S}_i$ with $i = 1, \ldots, p-1$ and $\mathcal{K}(x, S_1) = 1$ for all $x \in \mathcal{S}_p$. A Markov chain, which is not periodic is, thus, called *aperiodic*. Intuitively speaking, a periodic chain moves cyclically from one of the partitioning sets to another.

Equipped with these definitions, we can now state the convergence theorem:

Theorem 4.1.1 ([74, Theorem 4])**.** *Given a φ-irreducible, aperiodic Markov chain with transition kernel $\mathcal{K}$ and invariant distribution π, both living on a measurable space $(\mathcal{S}, \mathcal{A})$, where $\mathcal{A}$ is countably generated from $\mathcal{S}$, then for π-almost every $x \in \mathcal{S}$*

$$\lim_{n\to\infty} \mathcal{K}(x, A) = \pi(A), \quad \forall A \in \mathcal{A}\,.$$

As the authors in [74] note, the σ-algebra $\mathcal{A}$ being countably generated is a weak condition and is e.g. fulfilled for $\mathbb{R}^d$ by the Borel-σ-algebra $\mathcal{B}_{\mathbb{R}^d}$, which can be generated from the set of intervals in $\mathbb{R}^d$ with rational endpoints.

At this point, it is not clear, whether the invariant distribution is unique and if chains might not converge towards a "wrong" distribution. However, as we will see in the next Section 4.1.2, invariant distributions, if they exist, are indeed unique.

One important consequence, that we can already draw from Theorem 4.1.1, is the independence of the starting distribution, as long as the actual starting point comes from some non-zero measure set under the invariant distribution. This observation is of practical relevance, as it allows us to choose starting points almost arbitrarily without affecting the convergence of the algorithm, which we will introduce in Section 4.2.

4.1.2 Invariant Distributions & Detailed Balance

Invariant distributions do not exist for every Markov chain. A simple example is the symmetric random walk on $\mathbb{Z}$, which in fact does not admit an invariant distribution [60]. Intuitively, we may have guessed this, as the chain might visit any integer equally likely in the long run. And indeed, the invariant measure of the symmetric random walk on $\mathbb{Z}$ is the uniform measure $\pi(x) = C$ for all $x \in \mathbb{Z}$ and C being an arbitrary constant. As $\sum_{x\in\mathbb{Z}} \pi(x) = \infty$, this measure cannot be normalized to form a probability measure. In other words, there exists no uniform distribution on $\mathbb{Z}$. With this example, we already anticipated a few of the key results of the existence of invariant distributions. Before stating it, we formally define the concept of positive recurrence.

A φ-irreducible Markov chain is said to be *recurrent*, if for any set $A \in \mathcal{B}_\mathcal{S}$ the expected number of returns to this set is infinite. Further, it is said to be *positive recurrent*, if in addition the expected return time is finite. That is, $X_1, X_2, \ldots$ is recurrent, if

$$\mathbb{E}[\eta_A | X_1 \in A] = \infty, \quad \forall A \in \mathcal{B}_\mathcal{S},$$

where $\eta_A := \sum_{n\geq 1} \mathbb{1}_A(X_n)$ is the number of returns to the set $A \in \mathcal{B}_\mathcal{S}$, and positive recurrent, if in addition

$$\mathbb{E}[\tau_A | X_1 \in A] = m_A < \infty, \quad m_A \in \mathbb{N},\ \forall A \in \mathcal{B}_\mathcal{S},$$

where $\tau_A := \inf\{n \geq 1 : X_n \in A\}$ is the return time to the set $A \in \mathcal{B}_\mathcal{S}$. Although (positive) recurrence is defined only locally, it does indeed apply to the whole state space, if the chain is φ-irreducible [57].

For Markov chains on countable state spaces, we can now state *Kac's theorem*:

Theorem 4.1.2 (**Kac's theorem**, adapted from [60, Theorem 1.7.7])**.** *Given an irreducible Markov chain on a countable state space $\mathcal{S}$, the chain is positive recurrent if and only if it admits an invariant distribution π. In this case,*

$$m_x = \frac{1}{\pi(x)}.$$

Intuitively speaking, Kac's theorem states, that the probability to observe any state in the chain is proportional to the frequency, with which the state is visited. Similar results exist for the uncountable state space and can be found in [57, Theorem 10.4.9, 10.4.10] and are omitted here, as they require the introduction of more technical details, which would go beyond the scope of this work.

As the setting of Bayesian inference, for which we are introducing Markov chains here, usually already defines the distributions, for which we want our chain to be invariant, the existence is not a problem in general. Nevertheless, it is important to keep in mind, that Markov chains can be constructed, which admit no invariant distribution, but instead only admit an invariant measure, which does not need to be finite. The density function of this measure will, thus, not be integrable. Indeed, the random walk on $\mathbb{Z}$ is exactly such a Markov chain.

The following theorem tackles the question of existence and uniqueness of an invariant measure:

Theorem 4.1.3 ([57, Theorem 10.0.1])**.** *A recurrent Markov chain admits a unique (up to constant multiples) invariant measure.*

Having certainty about the existence and uniqueness of invariant distributions, we now turn our attention towards a sufficient condition for invariance of a distribution under a transition kernel.

A Markov chain with transition kernel $\mathcal{K}$ is called *reversible*, if

$$\int_A \pi(\mathrm{d}x)\,\mathcal{K}(x,B) = \int_B \pi(\mathrm{d}x)\,\mathcal{K}(x,A), \quad \forall A,B \in \mathcal{B}_{\mathcal{S}} \tag{4.3}$$

(4.3) is also known as the *detailed balance equation* and its importance becomes clear from the following lemma.

Lemma 4.1.1. *If a Markov chain with transition kernel $\mathcal{K}$ is reversible w.r.t. some probability measure π, then π is the invariant distribution of the Markov chain.*

Proof. Let $\mathcal{K}$ be reversible w.r.t. π, then for any $A \in \mathcal{B}_{\mathcal{S}}$

$$\begin{aligned}\int_{\mathcal{S}} \pi(\mathrm{d}x)\,\mathcal{K}(x,A) &= \int_A \pi(\mathrm{d}x)\,\underbrace{\mathcal{K}(x,\mathcal{S})}_{=1,\ \forall x} \\ &= \int_A \pi(\mathrm{d}x) \\ &= \pi(A),\end{aligned}$$

but

$$\int_{\mathcal{S}} \pi(\mathrm{d}x)\,\mathcal{K}(x,A) = (\pi\,\mathcal{K})(A)$$

just as in (4.1), so clearly π is an invariant measure under $\mathcal{K}$. □

Thus, if we are given a probability measure π, it suffices to construct a transition kernel, which is reversible w.r.t. the given probability measure, to ensure that this chain will indeed have π as its invariant distributions. Obviously, still care needs to be taken to ensure aperiodicity and irreducibility to guarantee convergence of the chains simulation towards π. However, as we will see in the upcoming section, constructing reducible or periodic chains is rather artificial and most transition kernels commonly used in practice do indeed converge.

4.2 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm was first presented by Nicolas Metropolis, Arianna & Marshall Rosenbluth and Augusta & Edward Teller in 1953 [56] as a method to generate physical

Algorithm 1: Metropolis-Hastings algorithm

Data: $n \geq 1, \pi, q, x_1$ with $\pi(x_1) > 0$
Result: $x_1, \ldots, x_n \sim \pi$

```
for i = 1, ..., n do
    draw y ~ q(x_i, ·)
    draw u ~ U(0, 1)
    if u ≤ α(x_i, y) then
        x_{i+1} = y
    else
        x_{i+1} = x_i
    end
end
```

states of a system distributed by their Boltzmann distribution. Later Hastings generalized their idea to construct Markov chains, which converge towards any desired distribution [37].

For the remainder of this chapter, we will restrict our attention to chains in $\mathbb{R}^d$ for $d \in \mathbb{N}^+$ and assume all probability measures to be integrable w.r.t. the respective d-dimensional Lebesgue measure. Given a *target distribution* π, from which we want to generate samples, the Metropolis-Hastings algorithm proposes a new state y based on the current state x_i using some *proposal distribution* $q(x_i, \cdot)$. Abusing notation, we shall denote the respective densities with $\pi(\cdot)$ and $q(x_i, \cdot)$. The new state is accepted with probability

$$\alpha(x_i, y) = \min\left\{1, \frac{\pi(y)\, q(y, x_i)}{\pi(x_i)\, q(x_i, y)}\right\}. \tag{4.4}$$

in which case the chain transitions to $x_{i+1} = y$. The chain remains put with $x_{i+1} = x_i$, if the move was rejected. (4.4) is also commonly known as *Metropolis filter* or *Metropolis criterion*. The algorithm is summarized in Algorithm 1.

Given the above acceptance probability, the rejection probability is obviously $1 - \alpha(x_i, y)$ and, thus, the transition kernel defined by Algorithm 1 can be given as

$$\mathcal{K}(x, A) = \int_A q(x, y)\, \alpha(x, y)\, \mathrm{d}y + \mathbb{1}_A(x)\, r(x) \tag{4.5}$$

with

$$r(x) = 1 - \int q(x, y)\, \alpha(x, y)\, \mathrm{d}y\,.$$

Using this transition kernel, we can show that the Metropolis filter indeed creates a reversible Markov chain. That is, we aim to show

$$\int_A \pi(x)\, \mathcal{K}(x, B)\, \mathrm{d}x = \int_B \pi(y)\, \mathcal{K}(y, A)\, \mathrm{d}y,$$

which holds, if

$$\int_A \int_B \pi(x)\, q(x, y)\, \alpha(x, y)\, \mathrm{d}x\, \mathrm{d}y = \int_A \int_B \pi(y)\, q(y, x)\, \alpha(y, x)\, \mathrm{d}y\, \mathrm{d}x$$

and

$$\int_{A\times B} \pi(x)\delta_y(x)\, r(x)\mathrm{d}x\, \mathrm{d}y = \int_{A\times B} \pi(y)\delta_x(y)\, r(y)\mathrm{d}y\, \mathrm{d}x,$$

which becomes clear from expanding the detailed balance condition (4.3) for (4.5) and where we used

$$\mathbb{1}_A(x) = \int_A \delta_y(x)\, \mathrm{d}y.$$

Now the latter equality is trivially true, as either $x = y$, in which case the the integrands may be interchanged arbitrarily, or $x \neq y$, in which case both sides vanish. The former equality holds, since

$$\begin{aligned}
\pi(x)\, q(x,y)\, \alpha(x,y) &= \pi(x)\, q(x,y) \min\left\{1, \frac{\pi(y)\, q(y,x)}{\pi(x)\, q(x,y)}\right\} \\
&= \min\left\{\pi(x)\, q(x,y), \pi(y)\, q(y,x)\right\} \\
&= \min\left\{\frac{\pi(x)\, q(x,y)}{\pi(y)\, q(y,x)}, 1\right\} \pi(y)\, q(y,x) \\
&= \min\left\{\frac{\pi(x)\, q(x,y)}{\pi(y)\, q(y,x)}, 1\right\} \pi(y)\, q(y,x) = \pi(y)\, q(y,x)\, \alpha(y,x).
\end{aligned}$$

Thus, the Metropolis-Hastings algorithm indeed constructs a chain with the desired invariant distribution. We refer to the resulting chain as the *Metropolis chain*.

As is stated in Theorem 4.1.1, the convergence of our algorithm towards the correct invariant distribution is only guaranteed for φ-irreducible and aperiodic chains. Fortunately, the Metropolis-Hastings chain will be π-irreducible and and aperiodic under very mild conditions on the target and proposal distribution.

Lemma 4.2.1. *Let π be a bounded probability measure and let $\operatorname{supp}\pi := \{x \in \mathbb{R}^d : \pi(x) > 0\}$ be path-connected. Further, let the proposal distribution $q(x, \cdot\,)$ be s.t. $\exists\, \epsilon > 0$ for every $x \in \operatorname{supp}\pi$ and*

$$q(x,y) > 0, \quad \forall\, y \in \mathcal{B}(x,\epsilon),$$

where $\mathcal{B}(x,\epsilon) := \{y \in \mathcal{S}^d : \| x - y \|_2 < \epsilon\}$. The Markov chain with proposal distribution q and transition kernel as in (4.5) is π-irreducible and aperiodic.

Proof. C.f. Lemma A.1 in the appendix. □

To put it simply, if the support of the target distribution is connected and there exists a small ball around every state, that can be reached by the proposal distribution and has positive density under the target distribution, then the chain can jump through a finite sequence of balls from any one state to any other within the support of π. This result emphasizes the vast flexibility of the Metropolis-Hastings algorithm in developing proposal algorithms, which will all converge correctly towards the desired target distribution. Although the resulting chains all share the same invariant distribution, it is important to notice, that they are not the same chains. This simple observation is a key to the main contribution of this work, as chains may indeed vary in efficiency, that is, they may converge at different rates. This issue will be discussed in some more detail in Chapter 6. A number of general proposal algorithms as well as algorithms more

specialized for the linearly constrained case will be introduced in discussed in some more detail in Chapter 5.

Now that the correctness of the Metropolis-Hastings algorithm is established, it is easy to see from the Metropolis filter as in (4.4), that the algorithm will indeed also work, if the target distribution's density π is unnormalized. Let $\pi = \frac{\hat{\pi}}{Z}$ with $\hat{\pi}$ being an unnormalized density and Z the respective normalization constant. It is obvious from looking at Algorithm 1, that the target density will be computed only within the Metropolis filter. So, for an unnormalized density, we obviously have, that

$$\alpha(x_i, y) = \min\left\{1, \frac{\pi(y)\, q(y, x_i)}{\pi(x_i)\, q(x_i, y)}\right\} = \min\left\{1, \underbrace{\frac{Z}{Z}}_{=1} \frac{\hat{\pi}(y)\, q(y, x_i)}{\hat{\pi}(x_i)\, q(x_i, y)}\right\} = \min\left\{1, \frac{\hat{\pi}(y)\, q(y, x_i)}{\hat{\pi}(x_i)\, q(x_i, y)}\right\}.$$

This is another great advantage of the Metropolis-Hastings algorithm as computing the normalization constant requires the solution of the possibly high-dimensional integral

$$Z = \int_{\mathcal{S}} \hat{\pi}(x)\, \mathrm{d}x$$

and is thus infeasible in general.

4.2.1 Convergence Diagnostics

The main drawback of Markov chain Monte Carlo methods is the truncation error introduced from finite simulations. In general, the rate of convergence cannot be computed for arbitrary Markov chains. There, in practice, convergence has to be estimated. One of the most commonly used techniques is the *potential scale reduction factor* originally introduced in [28].

The main idea of the potential scale reduction factor is to compare the distribution of states produced by parallel Metropolis chains, where the starting points where taken from an *overdispersed* starting distribution, meaning that the variance of the starting distribution should be larger than that of the target distribution. This overdispersion then guarantees, that the variance of the combined samples from the parallel chains will approach the true variance of the target distribution from above.

Let x_{ij} be the jth sample from the ith chain from a total of m chains with n samples each, where the mean of a chain is

$$\hat{\mu}_i = \frac{1}{n} \sum_{j=1}^{n} x_i\, j$$

and the combined mean over all chains is

$$\hat{\mu} = \frac{1}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} x_i\, j.$$

Then, in equilibrium, the mean of each chain should match the mean over the combined samples and, thus, their variance, called the *between-sequence variance*

$$B/n := \frac{1}{m-1} \sum_{i=1}^{m} (\hat{\mu}_i - \hat{\mu}),$$

should decay and become zero. Further, the average *within-sequence variance* is defined as

$$W := \frac{1}{m(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - \hat{\mu}_i)^2$$

and, as is claimed in [28], the weighted average

$$\sigma_+^2 := \frac{n-1}{n} W + \frac{B}{n}$$

is an unbiased estimator of the true variance of the samples σ^2. However, in the case of overdispersed starting points, this estimate will approach the true variance from above. Now if σ^2 was known, one could simply compute the ratio σ_+^2/σ^2 and assess convergence once this ratio is sufficiently close to 1. Clearly, σ^2 cannot be known in general and must be estimated. Since $\sigma_+^2 \gtrsim W$ for large n, a good candidate would be $\sigma_+ 2/W$ or

$$\hat{R} := \sqrt{\frac{\sigma_+^2}{W}},$$

which indeed has been proposed as a variant of the *potential scale reduction factor* in the literature [27, 97]. Originally, a further correction factor $\hat{V} := \sigma_+^2 + B/(mn)$ had been proposed to account for variability in $\hat{\mu}_i$ [28].

As is mentioned in [13], overdispersion of starting points is an important requirement for the diagnostic to work well as otherwise σ_+^2 may underestimate the variance of the target distribution and thus diagnose convergence prematurely. One approach suggested in [13] is to detect modes of the distribution using optimization techniques and to generate starting points from a mixture distribution with components centered around the found modes. For target distributions whose support has finite Lebesgue-measure, uniform samples from this support can also be considered an overdispersed starting distribution.

However, once samples suitable for inference have been found, the overdispersion might introduce a strong bias in the estimates. That is, recalling Kac's theorem (Theorem 4.1.2), the probability to observe a state is proportional to the frequency with which that particular state is visited. In finite simulations, a single visit to a state with low density under the target distribution, might vastly overrepresent this state in relation to its probability.

As a remedy, it is typically recommended to discard the first half of the obtained samples as a so-called *burn-in* [13]. Metaphorically speaking, the burn-in period is the time we have to wait until the Metropolis chain *warmed up*. In situations, where samples are costly to obtain, discarding many samples might seem wasteful. In such a case, *starting hot* seems a reasonable approach, which means starting from the modes, as these are already more likely to lay in some high-probability region and thus decrease the risk of overrepresenting low-probability regions. By Theorem 4.1.1, the starting point may indeed be chosen arbitrarily without affecting the convergence properties of the chain as long as the starting point comes from the support of the target's density. However, starting in the modes of the distribution risks choosing an *underdispersed* starting distribution, in which case the aforementioned convergence diagnostics may be deceiving.

Chapter 5

Sampling on Convex Polytopes

A *convex polytope* in $\mathbb{R}^d$ is the set of all points in $\mathbb{R}^d$ for which a linear inequality of the form $Ax \leq b$ holds, that is

$$\mathcal{P} := \left\{ x \in \mathbb{R}^d : Ax \leq b \right\}$$

with $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. It was already outlined in Chapter 3 how such linear constraints can arise when modeling systems of interest, usually determining the space of meaningful or feasible solutions. When sampling arbitrary target distributions π but restricted to a convex polytope $\mathcal{P}$, we can easily adapt all of the formalism of the previous Chapter 4 by considering the constrained target density

$$\hat{\pi}(x) := \frac{\mathbb{1}_{\mathcal{P}}(x)\,\pi(x)}{Z_{\hat{\pi}}}, \quad \forall x \in \mathbb{R}^d \quad \text{with} \quad Z_{\hat{\pi}} := \int_{\mathcal{S}} \mathbb{1}_{\mathcal{P}}(x)\,\pi(x)\mathrm{d}x$$

where $\mathbb{1}_{\mathcal{P}}$ is the indicator on the convex polytope to which we want to restrict π. Clearly $\mathbb{1}_{\mathcal{P}} \cdot \pi$ is integrable if π is, making $Z_{\hat{\pi}}$ and $\hat{\pi}$ well-defined. In fact, if $\mathcal{P}$ has finite Lebesgue-measure, π only needs to be bounded on $\mathcal{P}$ for $\hat{\pi}$ to be integrable, since

$$\int_{\mathcal{S}} \mathbb{1}_{\mathcal{P}}(x)\,\pi(x)\mathrm{d}x = \int_{\mathcal{P}} \pi(x)\mathrm{d}x \leq \int_{\mathcal{P}} C\mathrm{d}x = \lambda(\mathcal{P})C < \infty \quad \text{with} \quad \pi(x) \leq C,\, \forall x \in \mathcal{P}\,.$$

These observations make $\hat{\pi}$ amenable to Markov chain Monte Carlo sampling and the Metropolis-Hastings algorithm under the condition that the starting point lies in the polytope and has non-zero density under the target distribution, i.e. $\pi(x) \neq 0$. The necessity of this condition can be directly derived from the acceptance criterion $\alpha(x, y)$ defined in (4.4), which is undefined for all x with $\pi(x) = 0$.

Given the condition of a starting point in $\mathcal{P}$, any algorithm which works for the unconstrained problem theoretically also works in the constrained setting. In practice however, naively applying methods for the unconstrained problem turns out to be suboptimal as moves outside of the polytope may be suggested, which always have to be rejected. The phenomenon is illustrated in Example 1 and can make high-dimensional problems practically infeasible if no countermeasures are taken. Nevertheless, some general proposal distributions will be discussed in this section before turning to more specialized algorithms which are tailored to work well in linearly constrained spaces.

Example 1. *Let $\mathcal{P} = \{x \in \mathbb{R}^d : x \geq \mathbf{0}\}$ and $x_0 = \mathbf{0}$, where $\mathbf{0} = (0, \ldots, 0)^\top \in \mathbb{R}^d$. Further let $q(x, \cdot) = \mathcal{U}_{\mathcal{B}(x,r)}$ be the uniform distribution on a ball centered at x with radius r. The probability to draw a proposal $y \sim q(x_0, \cdot)$ in $\mathcal{P}$ is $\mathbb{P}(y \in \mathcal{P}) = 2^{-d}$ which rapidly converges to zero as $d \to \infty$. Thus the expected number of draws we need to find our way back into the polytope is 2^d, which is prohibitive in high-dimensional spaces.*

5.1 Proposal Distributions

As discussed in the previous Chapter 4 the construction of a proposal distribution, which is periodic or reducible is rather artificial. This opens up a vast space of possible candidate distributions that define valid transition kernels with the desired properties. At the same time, the choice of an appropriate proposal distribution is one of the major challenges when using the Metropolis-Hastings algorithm. In practice, the theoretical asymptotic results stated before are of little worth, if the limiting behaviour can only be observed after weeks, months or years of simulation. The development of fast converging samplers is thus crucial for practical application of Metropolis-Hastings, especially in Bayesian inference, where the evaluations of the likelihood may require costly forward simulations, and has found widespread attention [18, 19, 20].

5.1.1 Isotropic Gaussian Random Walk

One of the most prominent and widely studied proposal distributions is the isotropic Gaussian random walk [72, 73, 105], where $y \sim q(x, \cdot) = \mathcal{N}(x, \hat{s}^2 I)$ with $\hat{s} = s\, d^{-1/2}$ following arguments by [72] and where s is some tuneable scaling parameter and d the dimension. For a constant positive definite covariance $\Sigma = LL^\top$ with L being its Cholesky factor, it holds that $Ly \sim \mathcal{N}(x, \hat{s}^2\Sigma)$. We refer to L as a *preconditioner*. Possible choices for L can for example be a rounding transformation of the polytope (as discussed in more detail in Section 5.2) or a (prior) estimate of the target distributions covariance $\bar{\Sigma}_\pi = LL^\top$.

In general, the covariance does not have to be constant, but may vary as a function of x. In that case, the ratio of proposal densities as required in the Metropolis filter (4.4) is

$$\frac{q(y,x)}{q(x,y)} = \frac{\sqrt{\det(2\pi\Sigma_y)}^{-1} \exp\left\{-\frac{1}{2}\| x-y \|^2_{\Sigma_y}\right\}}{\sqrt{\det(2\pi\Sigma_x)}^{-1} \exp\left\{-\frac{1}{2}\| x-y \|^2_{\Sigma_x}\right\}} = \sqrt{\frac{\det(\Sigma_x)}{\det(\Sigma_y)}} \frac{\exp\left\{-\frac{1}{2}\| x-y \|^2_{\Sigma_y}\right\}}{\exp\left\{-\frac{1}{2}\| x-y \|^2_{\Sigma_x}\right\}}, \tag{5.1}$$

where Σ_x and Σ_y are the proposal covariances at x and y respectively. Clearly, for a constant proposal covariance, the Gaussian random walk becomes a symmetric proposal distribution and thus $q(y,x)/\,q(x,y) = 1$.

5.1.2 Ball Walk

A conceptually very simple proposal distribution is the uniform distribution $\mathcal{U}_{\mathcal{B}(x,s)}$ from a ball

$$\mathcal{B}(x,s) = \left\{ z \in \mathbb{R}^d : \|z - x\|_2 \leq s \right\}$$

of radius s centered at the current state x. Similarly to the Gaussian random walk, we can apply a preconditioning transformation on the proposal $y \sim \mathcal{U}_{\mathcal{B}}(x,s)$ such that $Ly \sim \mathcal{U}_{\mathcal{E}(x,s,M^\top)}$ with $M = LL^\top$ is a uniformly drawn point from the ellipse

$$\mathcal{E}(x,s,M) = \left\{ z \in \mathbb{R}^d : \|z - x\|_M \leq s \right\} \quad \text{with} \quad \|z\|_M = \sqrt{z^\top M z}.$$

Throughout this work, we only considered the Ball walk with constant s and M, in which case $q(y,x)/\,q(x,y) = 1$.

5.1.3 Hit & Run

According to [83], the Hit & Run algorithm was first proposed by [9] and independently by [82] and is a general proposal algorithm, which allows to sample many, especially truncated, distributions. The steps are as follows:

1. Draw a direction $r \sim q_D$
2. Define the function $S(\gamma) := x + \gamma\, r$ and its image set $S := \mathcal{S} \cap \{\gamma \in \mathbb{R} : S(\gamma)\}$ in $\mathcal{S}$
3. Draw γ from the distribution defined by the density $q_S(\gamma) = \dfrac{q_S(x + \gamma\, r \mid x)}{\int_S q_S(x + \eta\, r \mid x)\, \mathrm{d}\eta}$
4. Propose $y = x + \gamma\, r$

The two most common choices for q_D are the uniform distribution on a d-dimensional unit hypersphere (also known as Hypersphere Directions Hit & Run) and a uniform distribution on the set of canonical basis vectors $e_1, \ldots, e_d$. The latter choice is also known as Coordinate (Directions) Hit & Run. Both methods can be preconditioned by some matrix $M = LL^\top$ by using the direction Lr in the above scheme. In this case, the directions are either drawn from the surface of the d-dimensional ellipsoid defined by M or as one of the vectors from the Eigenbasis of M .

In the convex polytope case, the line intersects S with the polytope border $\partial\mathcal{P}$ at most at two points $S(\ell)$ and $S(u)$. If $\mathcal{P}$ is bounded in all directions, then exactly two intersections of S and $\partial\mathcal{P}$ exist. Considering

$$Ay \leq b \quad \Longrightarrow \quad A(x + \gamma r) \leq b \quad \Longrightarrow \quad \gamma \leq \frac{(b - Ax)_i}{(Ar)_i}, \quad \forall i = 1, \ldots, d : (Ar)_i \neq 0,$$

ℓ and u can be computed directly as the minimum and maximum elements of the right-hand side of the last inequality, i.e.

$$\ell = \max\left\{ \frac{(b - Ax)_i}{(Ar)_i} : i = 1, \ldots, d \wedge (Ar)_i \neq 0 \right\} \leq 0,$$
$$u = \min\left\{ \frac{(b - Ax)_i}{(Ar)_i} : i = 1, \ldots, d \wedge (Ar)_i \neq 0 \right\} \geq 0.$$

Note, that we can safely neglect any entries, where $(Ar)_i = 0$, as this means, that the direction r is parallel to the ith constraint and thus moving along r cannot violate it. If the polytope is unbounded in the direction r, then u will be positive infinity and likewise ℓ will be negative infinity, if the polytope is unbounded in direction $-r$. We can be detect unboundedness of $\mathcal{P}$ in direction r, when none of the well-defined entries of $\frac{b-Ax}{Ar}$ has positive sign. Unboundedness in direction $-r$ works symmetrically by considering negative signs in $\frac{b-Ax}{Ar}$

In general, the one-dimensional density $q_S(\gamma)$ is only well-defined, if $q_S(x + \gamma\, r \mid x)$ is integrable on S. For a bounded polytope $\mathcal{P}$, the line S will always have finite Lebesgue measure, thus yielding a well-defined proposal density. In the unbounded case, more care has to be taken in choosing an appropriate density q_S. Throughout this work, we used a truncated univariate normal distribution $\mathcal{N}_S(x, \sigma_S^2)$ on the line S.

5.1.4 Dikin Walk

The Dikin walk sampler uses locally computed metrics in order to precondition the proposal at every step such that it reduces the risk of proposing moves outside the polytope. Its proposal distribution uses the *Dikin ellipsoid*

$$\mathcal{E}_{\mathcal{D}}(x) := \{z \in \mathbb{R}^d : \| x - z\|_{M_{\mathcal{D}}(x)} \leq s\}$$

at the current state x to obtain a proposal distribution, which has most of its density mass located inside the polytope. Given the system matrix $A \in \mathbb{R}^{m \times d}$ and the vector of boundaries $b \in \mathbb{R}^m$ defining the polytope $\mathcal{P}$, the ellipse defining matrix at state x is computed as

$$M_{\mathcal{D}}(x) = \sum_{i=1}^{m} \frac{a_i a_i^\top}{b_i - a_i^\top x},$$

where $a_i^\top$ denotes the ith row of A.

The Dikin ellipsoid can either be used as the covariance ellipse of a multivariate normal distribution, thus becoming a special case of the Gaussian random walk sampler, or as the support of a uniform distribution, in which case it can be considered as a special case of the Ball walk. Within this thesis, we only considered the Gaussian Dikin walk. In this case, clearly (5.1) holds for the ratio of proposal densities at x and y.

5.1.5 Adaptive Metropolis

The Adaptive Metropolis (AM) algorithm was originally proposed by [34] and is a variant of the Gaussian Random Walk, where the covariance $\bar{\Sigma}_t$ of the proposal distribution $q(x_{t-1}, \ldots, x_0, \cdot) = \mathcal{N}(x, s^2\, \bar{\Sigma}_t)$ at time t is *learned* from the chains history $(x_0, x_1, \ldots, x_{t-1})$, i.e.

$$\bar{\Sigma}_t = \begin{cases} \Sigma_0 & \text{if } t < t_0 \\ \mathrm{Cov}(x_0, \ldots, x_{t-1}) + \varepsilon I & \text{if } t \geq t_0 \end{cases}$$

where

$$\mathrm{Cov}(x_0, \ldots, x_{t-1}) = \frac{s}{t-1} \sum_{i=1}^{t-1} (x_i x_i^\top - \bar{x}\bar{x}^\top)^2 \quad \text{with} \quad \bar{x} = \frac{1}{t-1} \sum_{i=1}^{t-1} x_i,$$

$\varepsilon \in \mathbb{R}$ and $I \in \mathbb{R}^{d \times d}$ is the d-dimensional identity. t_0 is a given length of some initial period, where the constant covariance Σ_0 is used. This chain is obviously not Markovian anymore, since the transition kernel keeps memory about previous states. However, Haario et al. [34] show that as the chain runs the covariance converges

$$\lim_{t \to \infty} \bar{\Sigma}_t = s^2\, \Sigma_\pi + \varepsilon I$$

where Σ_π is the true variance of the target distribution. As a consequence, the whole chain can be shown to converge towards the correct target distribution.

As an adaption of the algorithm by Haario et al. [34] to the constrained setting, we set the initial covariance $\Sigma_0 = d^{-1/2}\, E_{MV}$ and replace the regularization term εI with $\varepsilon\, E_{MV}$, where E_{MV} is the positive definite matrix defining the maximum volume ellipsoid, which is described in some more detail in Section 5.2.

5.1.6 Constrained Simplified Manifold Metropolis Adjusted Langevin Algorithm

Unlike the previously introduced proposal algorithms, the simplified manifold Metropolis adjusted Langevin algorithm (SmMALA) uses local curvature and gradient information about the target density π to precondition proposals [31]. More precisely, the SmMALA proposal distribution is given as $q(x, \cdot) = \mathcal{N}(\mu(x), s^2\,\mathcal{I}^{-1}(x))$ centered around the *drifted* state $\mu(x) = x + \frac{s}{2}\mathcal{I}^{-1}(x)\nabla \log \pi(x)$. Here, $\mathcal{I}(x)$ is the *expected Fisher information matrix*, defined as

$$\mathcal{I}(x) = \mathbb{E}\left[\left(\frac{\mathrm{d}}{\mathrm{d}x}\log \pi(x \mid D)\right)^2 \middle| x\right].$$

For a posterior density as in (3.7), the Fisher information may be computed as

$$\mathcal{I}(x) = J_f(x)^\top \Sigma^{-1} J_f(x)$$

which requires computation of the Jacobian $J_f(x) = \frac{\mathrm{d}f(z)}{\mathrm{d}z}\big|_x$ of f. Recall, that for inference of parameters of dynamical systems, f may require numerical solution of a system of ordinary differential equations. Computing its Jacobian $J_f(x)$ can, thus, be expensive. Intuitively speaking, using the Fisher information as a metric, the SmMALA algorithm will move further along directions, where less gain of information is expected. As such, it adapts its proposal to move far along "flat" directions of the target distribution, while moving less in more "steep" directions.

However, the SmMALA algorithm is a general proposal algorithm developed for unconstrained problems. An adaptation to the linearly constrained case was proposed in [85], termed as the Constrained SmMALA (CSmMALA). The CSmMALA proposal introduces a new metric $M(x) = \omega\,\mathcal{I}(x) + (1-\omega)\,M_{\mathcal{D}}(x)$, where $M_{\mathcal{D}}(x)$ is the matrix defining the Dikin ellipse at x and $\omega \in [0, 1]$ is a user-defined parameter, the *Fisher weight*. The proposal distribution of the CSmMALA is then given as $q(x, \cdot) = \mathcal{N}(\mu(x), s^2\,M^{-1}(x))$ centered around the *drifted* state $\mu(x) = x + \frac{s}{2}M^{-1}(x)\nabla \log \pi(x)$. Again, as with the Dikin walk and the Adaptive Metropolis, the proposal density ratio can be computed according to (5.1).

5.2 Preconditioning & Rounding

As described in [44, 51], the time many sampling algorithms take to reach convergence, usually referred to *mixing time*, depends on the *sandwiching ratio* R/r, which is the ratio of the radii of the smallest containing and the largest contained ball of the polytope. Intuitively speaking, if the polytope is nearly isotropic and the sandwiching ratio is small, then an isotropic random walk (as the Hit-And-Run, BallWalk or Gaussian proposal algorithm with identity covariance) will work better than if the polytope is highly non-isotropic and e.g. highly stretched along some dimensions. The problem is depictured in Figure 5.1. Analytical results capturing this phenomenon exist in [44, 51].

Polytope rounding is a way of preconditioning the sampling problem by bringing the polytope into a nearly isotropic configuration and thus decreasing the sandwiching ratio [86]. The problem is solved by computing the maximum volume ellipsoid E_{MV} using the algorithm devised in [107] and then computing its inverse Cholesky factor to use it as a rounding transformation [36]. Consider the radius 1 maximum volume ellipsoid $\mathcal{E}_{MV} = \{x : x^\top E_{MV}\, x = 1\}$, by applying the transformation $L^\top$ with $LL^\top = E_{MV}$ we obtain

$$L^\top \circ \mathcal{E}_{MV} = \{\underbrace{L^\top x}_{=:\,y} : x^\top E_{MV}\, x = 1\} = \{y : y^\top L^{-1}\, E_{MV}\, L^{-\top} y = 1\} = \{y : y^\top y = 1\} = \mathcal{B}(0, 1),$$

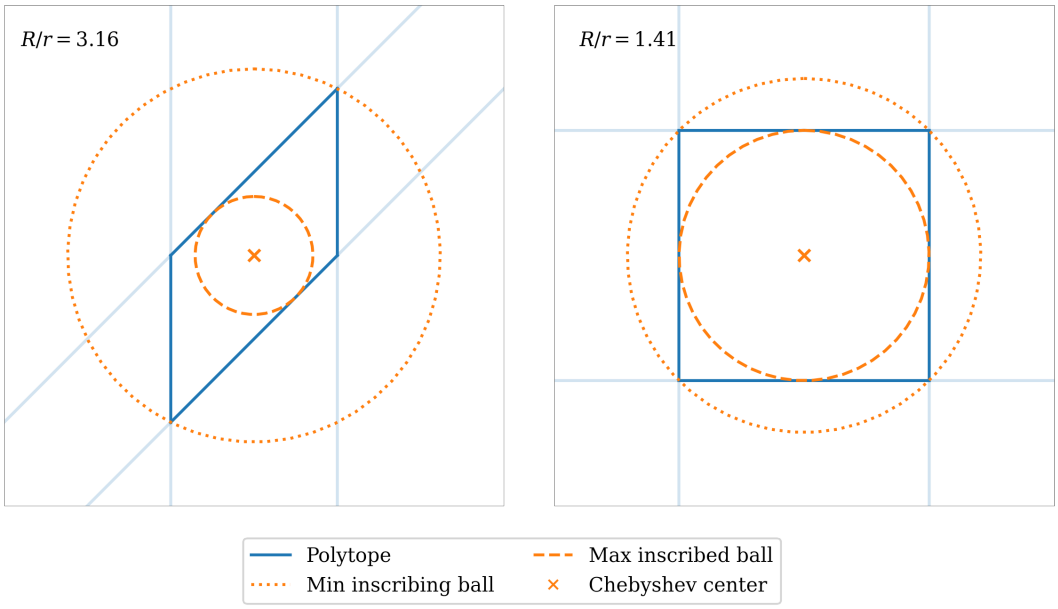


Figure 5.1: Sandwiching ratio of a parallelogram before and after rounding by depicturing the minimal inscribing and the maximal inscribed ball.

the unit ball. Thus, the inverse Cholesky factor of the maximum volume ellipsoid is a meaningful rounding transformation. Given the original polytope $\mathcal{P} = \{x : A\,x \leq b\}$, some linear transformation T and possibly a shift s, we arrive at the transformed polytope

$$\mathcal{P}' := T \circ \mathcal{P} = \{\underbrace{T\,x + s}_{=:\,y} : A\,x \leq b\} = \{y : \underbrace{A\,T^{-1}}_{=:\,A'} y \leq b + \underbrace{A T^{-1} s}_{=:\,b'}\} = \{y : A'\,y \leq b'\}.$$

Sampling n states can then be performed directly in the transformed polytope, requiring a single rounding transformation to begin with and a back transformation for every final state as $x = T^{-1}(y - s)$.

Chapter 6

Tuning Proposal Distributions

The theoretically guaranteed convergence of the Metropolis-Hastings algorithm as outlined in Chapter 4 allows for great freedom in designing proposal distributions that meet the theoretical assumptions of the convergence theorems. From the asymptotic perspective, an algorithm taking minutes is qualitatively the same as an algorithm taking centuries. In practice however, mere asymptotic convergence may obviously not be enough to make the Metropolis-Hastings algorithm feasible. Therefore, much of the work on Markov chain Monte Carlo simulations and the Metropolis-Hastings algorithm has focused on developing more clever proposal algorithms [52], among which Differential Evolution-MCMC (DE-MCMC) [59], the Metropolis adjusted Langevin algorithm (MALA) [31, 75] and Hamiltonian Monte Carlo (HMC) [23, 39] are considered to be some of the most efficient ones.

Apart from the design of ever more sophisticated proposal algorithms, determining optimal hyperparameters for these algorithms provides another lever to increase efficiency of the Metropolis-Hastings algorithm. Hyperparameter tuning is of great interest, not only because it may allow to improve convergence rates and efficiency significantly, but also because meaningful comparisons between different classes of proposal algorithms should consider the peak performance of every algorithm using (close to) optimal hyperparameters as well as the effort needed to identify such hyperparameters. The latter aspect is strongly influenced by design principles of *algorithm engineering* [77]. In algorithm engineering, algorithm design does not follow only theoretical guidelines, but takes practical feasibility into account, too, by performing extensive benchmarks on real-world problems. As a result, simple and straightforward algorithms often turn out to be more efficient in practice than more sophisticated algorithms with better theoretical performance bounds [77]. Focusing on experimentally assessed performance may not only penalize complex algorithms, where tuning effort might be high, but also allows to take characteristics of particular problem classes into account, where analytical treatment might be infeasible.

Remark. *As an example, in Bayesian ^{13}C Metabolic Flux Analysis, where computing the posterior density of feasible parameter values requires costly forward simulations, HMC is considered infeasible as it requires many very expensive evaluations of the posterior density's gradient [50]. Although a well-tuned HMC algorithm is considered capable of producing samples with very low autocorrelation at every step [50], the computational costs of doing so might be much larger than those of using a less sophisticated algorithm, which produces much more samples in the same time.*

Throughout this work, the consideration of practical feasibility will be the main guideline for designing and comparing hyperparameter tuning methods. It will also reappear in Section 6.2,

where consideration of time costs is emphasized in settings, where simulation steps may live on different time scales, thus shifting optimality criteria. Note, that theoretical considerations are nevertheless important in algorithm engineering, e.g. when deriving suitable performance measures.

The major challenge of hyperparameter tuning for Markov chain Monte Carlo methods is that, in general, efficiency of a particular hyperparameter choice cannot be computed a priori, but has to be estimated. Since testing hyperparameters exhaustively brings little benefit, if the tuning effort exceeds the effort, which would have been needed to sample the problem until convergence using an arbitrary step size, tuning can only happen in a heuristic manner using short, preliminary sampling runs. However, in order to perform tuning, we first have to formalize the notions of performance and efficiency. Thus, we start this chapter by introducing the *effective sample size* as one of the most prominent measures of statistical efficiency and identify its practical drawbacks for hyperparameter tuning. Afterwards, we briefly state the classical results for acceptance rate tuning and reason, why these results may not be applicable to the more general problems dicsussed within this thesis. This leads to discussion of the *expected squared jump distance* as an alternative tuning target, due to its close relation to the effective sample size. In Section 6.2, possible effects of a linearly constrained domain paired with costly evaluations of the target distribution's density are considered.

6.1 Efficiency Criteria

From a theoretical point of view, the main criterion for efficiency of a Markov chain is the *mixing time*, which tells us how many steps the chain will take until it is only some ϵ "far away" from its invariant distribution, where the metric used is the total variation distance. Formally, the mixing time is defined [49] as

$$t_{\text{mix}}(\epsilon) := \min\{t \geq 1 : \max_{x \in \mathcal{S}} \max_{A \subseteq \mathcal{S}} |\mathcal{K}^t(x, A) - \pi(A)| < \epsilon\}.$$

In practice, the total variation distance cannot be computed numerically without already having samples from the target distribution at hand, making it unsuitable as an efficiency criterion for hyperparameter tuning.

Since in Bayesian inference, we often try to estimate expectations, a common approach to assess efficiency in practice, is considering the statistical efficiency of the samples by means of their correlations. In particular, high statistical efficiency leads to faster convergence of the estimate towards its true value, as the Markov chain central limit theorem shows Theorem 6.1.1. Before stating it, we introduce the notion of statistical efficiency in the upcoming section.

6.1.1 Effective Sample Size

When trying to estimate a statistic using a sample $(x_1, x_2, \ldots, x_n)$ drawn from some unknown distribution π, natural questions are, how representative are the samples of the true underlying distribution π and how many samples have to be drawn until one can be sure, that the estimation error is somewhat small. One way to quantify answers to these questions is to consider the standard deviation in the estimator. Consider the sample average

$$\hat{\mu} := \frac{1}{n} \sum_{i=1}^{n} x_i$$

as an estimator of the true mean $\mu := \mathbb{E}[x]$ of our random variable $x \sim \pi$, then $\mathrm{SE} := \sqrt{\mathrm{Var}[\hat{\mu}]}$ is known as the *standard error of the mean* and can be computed as

$$\begin{aligned}
\mathrm{Var}[\hat{\mu}] &= \mathbb{E}[\hat{\mu}^2] - \mathbb{E}[\hat{\mu}]^2 \\
&= \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)^2\right] - \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} x_i\right]^2 \\
&= \frac{1}{n^2}\,\mathbb{E}\left[\left(\sum_{i=1}^{n} x_i\right)^2\right] - \frac{1}{n^2}\left(\sum_{i=1}^{n} \underbrace{\mathbb{E}[x_i]}_{=\mu}\right)^2 \\
&= \frac{1}{n^2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}[x_i x_j] - n^2\mu^2\right) \\
&= \frac{1}{n^2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \Big(\mathbb{E}[x_i x_j] - \mu^2\Big)\right) \\
&= \frac{1}{n^2}\left(\sum_{i=1}^{n} \underbrace{\Big(\mathbb{E}[x_i^2] - \mu^2\Big)}_{=\sigma^2} + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} \Big(\mathbb{E}[x_i x_j] - \mu^2\Big)\right) \\
&= \frac{\sigma^2}{n^2}\left(n + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} \frac{\mathbb{E}[x_i x_j] - \mu^2}{\sigma^2}\right)
\end{aligned}$$

Now since $x_i \sim \pi$ for $i = 1, \ldots, n$, we have that $\mathbb{E}[x_i x_j] = \mathbb{E}[x_0 x_{j-i}] = \mathbb{E}[x_0 x_k]$ for $k := j - i$. Furthermore, the quantity

$$\rho_{ij} := \frac{\mathbb{E}[x_i x_j] - \mu^2}{\sigma^2} = \frac{\mathbb{E}[x_i x_{i+k}] - \mu^2}{\sigma^2} = \frac{\mathbb{E}[x_0 x_k] - \mu^2}{\sigma^2} =: \rho_k$$

is well known as the autocorrelation coefficient at lag k. Note that the above identity holds, because we assume all x_i to have the same law π, therefore the autocorrelation at some given lag is time-invariant. Using the identity, we obtain

$$\begin{aligned}
\mathrm{Var}[\hat{\mu}] &= \frac{\sigma^2}{n^2}\left(n + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} \rho_{ij}\right) \\
&= \frac{\sigma^2}{n^2}\left(n + 2\sum_{k=1}^{n}\sum_{i=1}^{n-k} \rho_k\right) \\
&= \frac{\sigma^2}{n^2}\left(n + 2\sum_{k=1}^{n}(n-k)\rho_k\right) \\
&= \frac{\sigma^2}{n}\left(1 + 2\sum_{k=1}^{n}\left(1 - \frac{k}{n}\right)\rho_k\right)
\end{aligned}$$

This already suffices to motivate large sample as the standard error of the mean estimator will only go down as $\mathcal{O}(\sqrt{n}^{-1})$. Since the $(1-k/n)$ factor in the sum does only increase linearly with n and the autocorrelation coefficient is assumed to be decaying exponentially [29], the factor $(1 - k/n)$ is usually neglected. This can also be further motivated by the fact that neglecting it does only lead to an overestimation of the standard error. Thus, we define the *integrated*

autocorrelation time as

$$\tau := 1 + 2 \sum_{k \geq 1} \rho_k \tag{6.1}$$

and can now write the standard error as

$$\text{SE} = \sqrt{\text{Var}[\hat{\mu}]} = \frac{\sigma\sqrt{\tau}}{\sqrt{n}} = \frac{\sigma}{\sqrt{n_{\text{eff}}}}$$

where we defined the *effective sample size* $n_{\text{eff}} := n/\tau$. An intuitive take on the integrated autocorrelation time is to consider it as the time (in MCMC steps) one has to wait until consecutive samples become uncorrelated. In the same sense, the effective sample size is then the number of uncorrelated samples obtained from the correlated draws.

Note that for uncorrelated samples, the autocorrelation coefficients are $\rho_{ij} = \delta_{ij}$ and thus $\tau = 1$. The standard error then becomes the well known uncorrelated standard error

$$\text{SE} = \frac{\sigma}{\sqrt{n}}.$$

Using all of the above introduced definitions, we can now state the Markov chain central limit theorem (MC-CLT).

Theorem 6.1.1 (**Markov chain central limit theorem**, adapted from [30] and Theorem 27, [71])**.** *For a ϕ-irreducible, aperiodic and reversible Markov chain $(x_1, x_2, \ldots) \in \mathcal{S}^{\mathbb{N}}$ with invariant distribution π, and measurable $h : \mathcal{S} \to \mathbb{R}$, it holds that*

$$\frac{1}{n} \sum_{i=1}^{n} h(x_i) \sim \mathcal{N}(\mu_h, \sigma_h^2 \tau_h n^{-1})$$

for $\sigma_h^2 = \mathbb{E}[(x - \mu)^2] < \infty$ with $\mu_h = \mathbb{E}[h(x)]$ and $\tau_h = 1 + 2\sigma_h^{-2} \sum_{k \geq 1} \mathbb{E}[h(x_0)h(x_k)] - \mu_h$.

The MC-CLT emphasizes, that there exist two levers for improving convergence of estimates: increasing the sample size n or decreasing its autocorrelation τ.

Estimating the Effective Sample Size

In practice, estimating the effective sample size is a non-trivial task due to the infinite sum in the integrated autocorrelation time. Further, clearly the autocorrelation coefficients $\rho_k, k \geq 1$ themselves have to be estimated, which requires at least some nk samples. Hence, autocorrelation terms at higher lags can only be estimated, if sufficient samples are available. Furthermore, the estimation error at higher lags will always be larger than those for smaller lags. Apart from this, autocorrelation coefficients can be computed efficiently using the fast Fourier transformation algorithm. This becomes obvious from considering the autocorrelation coefficients' estimator

$$\hat{\sigma}^2 \hat{\rho}_k + \hat{\mu}^2 = \sum_{i=1}^{n} x_i x_{i+k},$$

which is just the convolution of a discrete signal with itself. Thus, the convolution theorem [81] holds and

$$\hat{\sigma}^2 \hat{\rho}_k + \hat{\mu}^2 = \mathcal{F}^{-1}\{\mathcal{F}\{x\} \cdot \mathcal{F}\{x\}\},$$

which can thus be computed in $\mathcal{O}(n \log n)$.

Given the estimated autocorrelation coefficients, it remains to decide how and when to truncate the infinite sum from (6.1). One of the most commonly used[1,2] estimators is the *initial monotone sequence estimator* from [29]. Defining $p_i := \hat{\rho}_{2i} + \hat{\rho}_{2i+1}$, an *initial positive sequence* is obtained by considering all p_i until the first of them becomes less or equal 0. So the initial positive sequence is given as $p_1, \ldots, p_m$ for $m := \max\{i \geq 1 : p_j > 0, j = 1, \ldots, i\}$. The initial monotone sequence p_m is then obtained by choosing $p_{\mathrm{m},i} = \min\{p_1, \ldots, p_i\}$ for $i = 1, \ldots, m$. The integrated autocorrelation time can then be estimated by taking the sum of the sequence

$$\hat{\tau} := 1 + 2 \sum_{i=1}^{n/2} p_{\mathrm{m},i} \tag{6.2}$$

As [29] states, the initial monotone sequence estimator is not consistent, i.e. there is no guarantee, that the estimate (6.2) will converge towards the true value. However, they prove that in fact the sequence $\rho_{2i} + \rho_{2i+1}$ is a strictly decreasing, strictly convex, positive sequence, which motivates the previously introduced estimator. In fact, they argue, that in numerical simulations the autocorrelation at high lag will be mostly dominated by noise, for which the estimator tries to correct.

6.1.2 Acceptance Rate

The *acceptance rate* of the Metropolis-Hastings algorithm is the expected value of the Metropolis criterion, formally given as

$$\alpha := \int \alpha(x, y)\, \pi(x)\, \mathcal{K}(x, y)\, \mathrm{d}\, x\, \mathrm{d}\, y\,. \tag{6.3}$$

In practice, the acceptance rate is straightforwardly estimated by counting the number of accepted moves and dividing it by the total number of steps taken so far. Intuitively, a high acceptance rate might seem desirable, as rejected proposals always lead to perfectly correlated samples and do not let the Metropolis chain move at all. Indeed, the most efficient sampling technique, which is i.i.d. sampling of the posterior distribution, is a Metropolis chain that yields acceptance rate 1. Clearly, as we considered before, i.i.d. sampling of the posterior is not feasible in general, and we have to use more simple proposal distributions, that only act locally. In this settings and if the we assume that our target distribution's density is Lipschitz-continuous, the acceptance rate will converge against 1, if the steps of the chain are very small, which prevents the chain from exploring the state space [73]. At the same time, the acceptance rate is also assumed to converge against 0 for large step sizes. The latter can informally be explained by considering that any Lebesgue-integrable function converges almost everywhere against zero (cf. Lemma A.2 in the appendix). For very large step sizes, most proposed moves will therefore lead into low density regions and $\pi(x_1)/\pi(x_0)$ will be small. Thus, for symmetrical proposals, the acceptance probability will also be small. These considerations suggest, that there exists a trade-off between the two effects, which both effectively obstruct fast exploration of the state space and, thus, convergence.

Indeed, there exist widely used, theoretical results for simple target distributions of the form

$$\pi(x) = \prod_{i=1}^{d} f(x_i) \tag{6.4}$$

[1] `https://mc-stan.org/docs/2_29/reference-manual/effective-sample-size.html`
[2] `https://arviz-devs.github.io/arviz/api/generated/arviz.ess.html`

for some $f : \mathbb{R} \to \mathbb{R}$. In particular, as [72] prove, the isotropic Gaussian random walk achieves optimal efficiency at an acceptance rate of 0.234 in the asymptotic limit of $d \to \infty$. In practice, the asymptotic statement is not a problem, as the optimal acceptance rate converges quickly and already for $d = 5$ is achieved at about 0.27 [73]. However, clearly the form of the target distribution (6.4) is an issue, as it does not allow for any kind of correlations between any x_i and x_j, $i \neq j$. Clearly, when doing Bayesian inference for dynamic models, correlations cannot be ruled out, thus, it is unclear a priori, if tuning according to the acceptance rate will achieve optimal efficiency. Also note, that in the presence of a linearly constrained domain, any constraint including more than two coordinates will already introduce dependence, thus breaking one of the key assumptions of the result.

Further, the result is limited to the isotropic Gaussian random walk only. Although similar results exist for the Metropolis adjusted Langevin algorithm, stating an optimal acceptance rate at 0.574 [73], they do not exist for every algorithm. As it was shown in Example 1, in the case of linear constraints, algorithms which do not constrain their support to the polytope are likely to get stuck in high dimensions. It appears reasonable to assume, that such algorithms should be outperformed by the more sophisticated algorithms, which guarantee to remain inside the polytope or at least bound the probability to propose infeasible moves. However, for the latter algorithms, no optimal acceptance rates have been found and, thus, the acceptance rate results becomes again inapplicable. In summary, the optimal acceptance rate results are constrained to few particular problem and proposal algorithm combinations.

6.1.3 Expected Squared Jump Distance

Clearly, the requirement of many samples in order to get stable estimates of (6.2) disqualifies the effective sample size as a target for hyperparameter tuning using short, preliminary runs. However, a closely related and easier to estimate statistic, is the *expected squared k-jump distance*

$$k\text{-ESJD} := \mathbb{E}\big[(x_k - x_0)^2\big].$$

By expanding the squared term and using linearity of expectation, its relation to the autocorrelation and thus the effective sample size becomes obvious, i.e.

$$\begin{aligned} k\text{-ESJD} &= \mathbb{E}\big[x_k^2\big] - 2\,\mathbb{E}\big[x_k x_0\big] + \mathbb{E}\big[x_0^2\big] \\ &= \mathbb{E}\big[x^2\big] - 2\,\mathbb{E}\big[x_k x_0\big] + \mathbb{E}\big[x^2\big] \\ &= 2\Big(\mathbb{E}\big[x^2\big] - \mu^2 - \big(\mathbb{E}\big[x_k x_0\big] - \mu^2\big)\Big) \\ &= 2(\sigma^2 - \sigma^2\rho_k) \\ &= 2\sigma^2(1 - \rho_k) \end{aligned} \tag{6.5}$$

for $k \geq 1$ and where the chain was assumed to be in equilibrium [65]. For the remainder of this work, we will call the 1-ESJD just simply the expected squared jump distance and abbreviate it with ESJD. Note that σ^2 is a constant and thus maximizing the expected squared jump distance will minimize the k-th autocorrelation lag as well as the integrated autocorrelation time τ. Pairs of autocorrelation lags decrease over time [29], i.e. $\rho_k + \rho_{k+1} \geq \rho_{k+2i} + \rho_{k+2i+1}$ for all $k \geq 1$. Therefore, it is advisable to focus on the autocorrelations at small lags or respectively on the expected squared jump distance at few jumps. Nevertheless, a high expected squared jump distance only controls ρ_1 and, thus, if $\rho_k \approx \rho_1$ for $k > 1$, autocorrelation may still be large. The

identity from (6.5) also allows to optimize multiple autocorrelation coefficients at once, since

$$\sum_{k=1}^{K} \mathbb{E}\big[(x_k - x_0)^2\big] = \sum_{k=1}^{K} 2\sigma^2(1-\rho_k) = 2\sigma^2\left(\sum_{k=1}^{K} 1 - \sum_{k=1}^{K}\rho_k\right) = 2\sigma^2\left(K - \sum_{k=1}^{K}\rho_k\right).$$

This allows fine control over autocorrelation at multiple lags, however at the cost of increased computational effort and less robust estimation at higher lags, if the number of samples remains constant.

A different but quite illustrative perspective on the expected squared jump distance can be obtained by considering that $x_1 \sim \mathcal{K}(x_0, \cdot)$, where $\mathcal{K}$ is the Metropolis-Hastings kernel as in (4.5). Using the law of total expectation, we obtain

$$\mathbb{E}\big[(x_1 - x_0)^2\big] = \mathbb{E}\Big[\,\mathbb{E}_{\mathcal{K}}\big[(x_1 - x_0)^2 | x_0\big]\Big]$$

and now taking the definition of the Metropolis-Hastings kernel from (4.5) we can compute the inner expectation as

$$\begin{aligned}
\mathbb{E}_{\mathcal{K}}\big[(x_1 - x_0)^2 | x_0\big] &= \int_{\mathcal{S}} (x_1 - x_0)^2 \, \mathcal{K}(x_0, x_1) \, \mathrm{d}x_1 \\
&= \int_{\mathcal{S}} (x_1 - x_0)^2 q(x_0, x_1) \, \alpha(x_0, x_1) \, \mathrm{d}x_1 + \int_{\mathcal{S}} \underbrace{(x_1 - x_0)^2 \delta_{x_1}(x_0)}_{=0} r(x_0) \, \mathrm{d}x_1 \\
&= \int_{\mathcal{S}} (x_1 - x_0)^2 q(x_0, x_1) \, \alpha(x_0, x_1) \, \mathrm{d}x_1 \\
&= \mathbb{E}_q\big[(x_1 - x_0)^2 \, \alpha(x_0, x_1) | x_0\big]
\end{aligned}$$

and so we can rewrite the expected squared jump distance as

$$\mathbb{E}\big[(x_1 - x_0)^2\big] = \mathbb{E}\Big[\,\mathbb{E}_q\big[(x_1 - x_0)^2 \, \alpha(x_0, x_1) | x_0\big]\Big].$$

This reformulation allows some intuitive interpretation on the shape of the expected squared jump distance as a function of the step size s. Typically, the target distribution's density is assumed to be Lipschitz continuous and, thus, the acceptance rate is assumed to converge against 1 for small step sizes [73]. At the same time the *proposed squared jump distance* $(x_0 - x_1)^2$ clearly decreases, if the step size decreases and, thus, the expected squared jump distance converges against 0.

As we discussed previously, the acceptance rate is also assumed to converge against 0 for large step sizes. However, the proposed jump distance clearly increases as the step size increases. In general, the proposed squared jump distance can be assumed to be increasing as $\mathcal{O}(s^2)$, thus the acceptance rate has to decay at rate faster than $\mathcal{O}(s^{-2})$ for the expected squared jump distance to converge towards zero for large step sizes. Now for unconstrained proposals on a polytope with finite Lebesgue-measure, this holds. However, in fact, for constrained proposals, where the proposed squared jump distance is bounded by the diameter of the polytope, this does not hold in general. The experimental results presented in Section 9.1.3 support these observations.

Estimating the Expected Squared Jump Distance

We define the *estimated expected squared k-jump distance* given the samples $x_1, \ldots, x_n$ as

$$\widehat{k\text{-ESJD}} = \sum_{i=1}^{n-k} \|x_{i+k} - x_i\|^2$$

Estimating the expected squared k-jump distance is considerably simpler than estimating the effective sample size, as the number of high order autocorrelations we include is simply controlled by k as (6.5) implies.

6.2 Influence of Computational Costs on Efficiency Criteria

In practice, considering the wall-clock runtime is crucial to determine a problem practically feasible. Therefore, when using Markov chain Monte Carlo methods for sampling target distributions, which are computationally expensive to evaluate, considering computational costs already during tuning is advisable. An expensive proposal algorithm may create almost perfectly uncorrelated samples, but if it takes hours in doing so, a sampler which creates an abundancy of strongly correlated samples in a matter of minutes might end up creating a higher number of effective samples per second. The quantity of interest is thus the number of effective samples per second

$$n_{\text{eff},c} := \frac{n_{\text{eff}}}{s}$$

Another phenomenon considering time costs may arise when the infeasibility of a proposed state can be determined quickly, making rejection less expensive than acceptance. As discussed in Chapter 3 and Chapter 5, states outside the polytope represent biologically infeasible parameter sets and determining whether a state is inside or outside $\mathcal{P}$ can be done by one matrix-vector multiplication. Computing the likelihood as in (2.2) is considerably more expensive as it requires a full forward simulation of the model.

We define r_f to be the rejection rate of proposed states inside the polytope and let $r_{\mathcal{P}}$ be the rejection rate of proposed states outside the polytope. Now by considering c_f to be the cost of computing the likelihood (and therefore the forward simulation) and $c_{\mathcal{P}}$ to be the cost of computing the indicator on the polytope, we obtain the cost per sample as

$$c := (\alpha + r_f)\, c_f + c_{\mathcal{P}}\,.$$

Intuitively speaking, any proposal inside the polytope requires computation of the likelihood, whereas proposals outside the polytope can be rejected quickly once they have been deemed infeasible. Considering $n_{\text{eff},c}$, the inverse cost acts as a penalty on the effective samples size somewhat proportional to the acceptance rate α. Since the acceptance rate is an overall decreasing function of the step size s, this observations implies that larger step sizes, which produce more proposals outside the polytope, may be favorable over smaller step sizes when taking time costs into account.

Remark. *The described phenomenon of reduced costs per sample when proposals can be deemed infeasible for violating polytope constraints does not apply, if the sampler does not propose moves outside the polytope at all. This holds true especially for truncated proposal distributions as those which can be sampled using the Hit & Run algorithm described in Section 5.1.*

Chapter 7

Bayesian Optimization

After having determined suitable tuning targets for hyperparametrization of the Metropolis-Hastings algorithm in Chapter 6, we now turn our attention to the practical task of optimizing the given targets. As we already discussed in Chapter 6, the fundamental problem of evaluating tuning targets lies in the fact, that they cannot be computed a priori as function of the step size, but have to be estimated from samples. Such estimation will not only be subject to stochastic noise, but is also possibly expensive, if the sampling of the target distribution is already expensive. Preliminary attempts throughout the work of this thesis suggested that these distinct features impose great problems to common optimization algorithms.

Bayesian optimization is an optimization technique suited to compute extrema of possibly costly black box functions using the Bayesian paradigm. Unlike most classical optimization approaches, Bayesian optimization is well capable of dealing with stochastic noise and functions, which are not necessarily explicitly computable, but which instead may only be estimated. That is, assume we want to optimize some function $f(x)$, which cannot be computed exactly, but where an estimator $\hat{f}(x)$ is available for every $x \in \mathcal{X}$. Then we estimate

$$\arg\max_{x \in \mathcal{X}} f(x) \approx \arg\max_{x \in \mathcal{X}} \hat{f}(x).$$

Computing the estimator $\hat{f}(x)$ might be computationally expensive, thus, brute-force approaches like grid searches, where the parameter space $\mathcal{X}$ is sampled regularly, may be prohibitive and, instead, we are interested in minimizing the amount of function evaluations necessary to reach the optimum. Bayesian optimization algorithms have shown to be particularly efficient methods for optimizing such costly black box functions [96]. Throughout this chapter, we will consider f to be the black box function, we want to optimize, regardless of whether it is an estimator to some underlying function or an exactly evaluable function.

In a nutshell, Bayesian optimization methods assume a probability distribution over the possible function shapes $\mathbb{P}(f)$, which gets updated as data is collected in form of pointwise evaluations of f. That is, given we have observed data $D_t = \left\{(x_i, f(x_i))\right\}_{i=1}^{t}$ throughout t iterations, the posterior belief over the possible function shapes at time t is

$$\mathbb{P}(f \mid D_t) \propto \mathbb{P}(D_t \mid f)\, \mathbb{P}(f).$$

The posterior at each iteration is then used to determine where to evaluate the target function (or its estimate) next. The exact procedure, that maps from the posterior belief to the next parameter x_{t+1} , which is to be evaluated, is given by the *acquisition function* $u(\,\cdot \mid D_t)$ at time t. More precisely, we let

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} u(x \mid D_t).$$

Whereas the overall framework, as given in Algorithm 2, remains the same, the different designs of acquisition functions are what virtually distinguish Bayesian optimization algorithms. Throughout this work, we consider only *Thompson sampling* [93], which is a conceptually simple method, that has proven not to only work well in practice, but which also admits theoretically optimal regret bounds [1, 2]. Thompson sampling works by sampling the acquisition function from the posterior distribution $\mathbb{P}(f \mid D_t)$. The distribution over f is modelled using a *Gaussian process*, a stochastic process, that generalizes Gaussian distributions to functions.

Algorithm 2: Bayesian optimization algorithm [80]

```
D_0 = ∅
for t = 1, 2, ... do
    x_t = arg max_{x∈X} u(x | D_{t-1})
    y   = f(x_t)
    D_t = D_{t-1} ∪ {x_t, y}
end
```

In the upcoming sections, we start off by giving a short introduction to Gaussian processes and Thompson sampling, before we introduce the main contribution of this work, where we use the described Bayesian optimization techniques to tune hyperparameters of proposal distributions for the Metropolis-Hastings algorithm by optimizing the targets introduced in Chapter 6.

7.1 Gaussian Process Regression

Gaussian processes extend multivariate Gaussian distributions to functions and, thus, infinitely large sets of random variables. More precisely, a Gaussian process is an uncountable collection of random variables, for which any finite sample follows a multivariate Gaussian distribution [70]. Just as multivariate Gaussian distributions, a Gaussian process is defined by its mean and covariance, which however – unlike the finite case – are now considered to be functions $\mu(x)$ and $\kappa(x, x')$ respectively. We write

$$g \sim \mathcal{GP}(\mu, \kappa),$$

to state that g is a Gaussian process. In practice, Gaussian processes are sampled at n discrete locations $(x_1, \ldots, x_n)$, at which then – as by above definition – the function's values follow a multivariate normal distribution

$$(y_1, \ldots, y_n)^\top \sim \mathcal{N}(\mu', \Sigma).$$

The mean μ' and covariance Σ of the finite sample are given by the mean and covariance functions of the Gaussian process as

$$\mu'_i = \mu(x_i) \quad \text{and} \quad \sigma^2_{ij} = \kappa(x_i, x_j).$$

For convenience, we shall also write $\mu' = \mu(x)$ and $\Sigma = \kappa(x, x')$ with x, x' being vectors. By the additivity of a Gaussian random variable with respect to its mean, we have that $y = z + m(x)$ for some $z \sim \mathcal{N}(0, \Sigma)$, indicating, that the behaviour of the process is mainly defined by the covariance function κ. One of the most commonly used covariance functions, is the squared exponential function

$$\kappa(x, x') = \alpha^2 \exp\left\{ -\frac{\|x - x'\|_2^2}{\ell^2} \right\}, \tag{7.1}$$

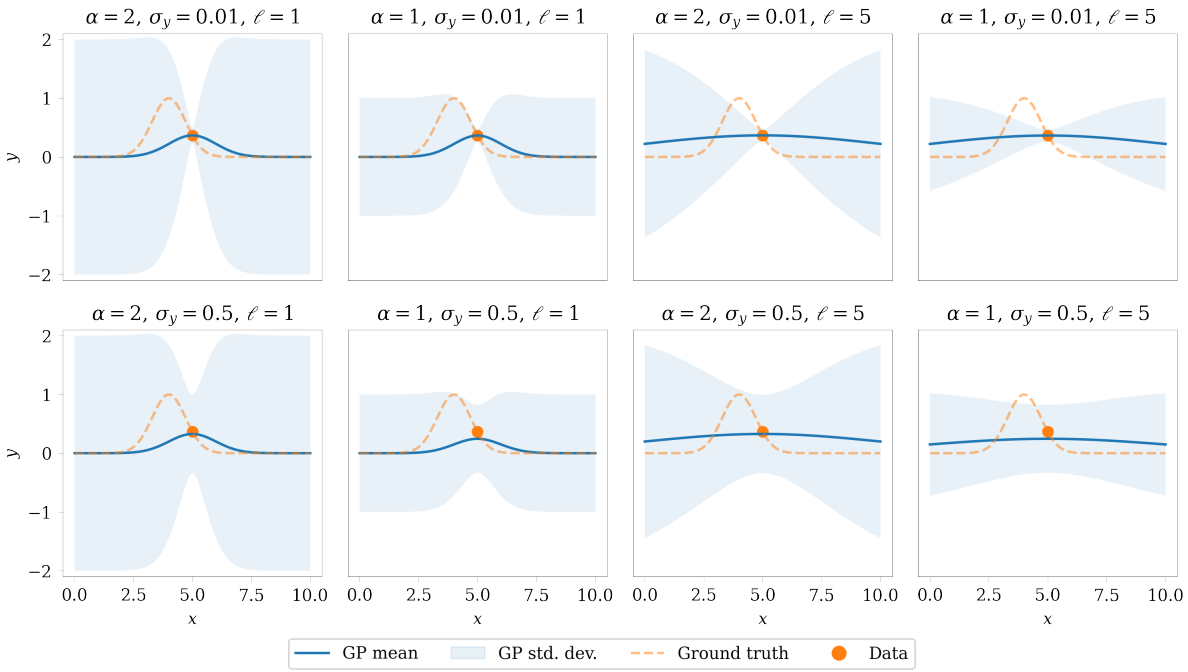


Figure 7.1: Effect of different hyperparameter choices α, ℓ for a Gaussian process with a squared exponential kernels (7.1) and observation noise σ_y.

which results in a smooth process [24]. The squared exponential function comes with two hyperparameters, namely a scaling factor α, which controls the magnitude of the covariance, and a length scale ℓ, which controls how quickly the influence of the value $g(x)$ over the value $g(x')$ decays as x and x' get further apart. A visualization of some squared exponential kernels and the resulting Gaussian processes using different hyperparameters is given in Figure 7.1.

7.1.1 Posterior Gaussian Processes & Regression

Gaussian processes are commonly applied to solve regression problems for rather arbitrarily shaped functions. In order words, given a set of observations $D = \{x_i, y_i\}$ from some unknown data generating process, how do we choose the mean and covariance functions, s.t.

- the mean of the Gaussian process passes the observations closely, that is $\mu(x_i) \approx y_i$,
- the variance in $g(x)$ is high, when far from any observation, and small, if close to an observation?

Luckily, the statistic nature of Gaussian processes allows to use the concept of conditional probability to solve these questions. That is, consider a joint vector $y = (y_1, y_2)^\top$ to be a finite sample from a Gaussian process $\mathcal{GP}(\mu, \kappa)$ at the locations $x = (x_1, x_2)^\top$. Then clearly, as we already considered above,

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim \mathcal{N}(\mu', \Sigma) = \mathcal{N}\left(\begin{pmatrix} \mu'_1 \\ \mu'_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \right)$$

with $\mu' = \mu(x)$ and $\Sigma = \kappa(x, x)$. Now consider, that y_1 have already been observed, then the distribution of y_2 becomes a Gaussian distribution conditional on y_1, which can be computed

explicitly as

$$y_2|y_1 \sim \mathcal{N}\big(\underbrace{\mu_2' + \Sigma_{21}\Sigma_{11}^{-1}(y-\mu_1')}_{=:\hat{\mu}'}, \underbrace{\Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}}_{=:\hat{\Sigma}}\big),$$

which is the standard formula for conditioning multivariate Gaussian distributions [8]. Using this conditional Gaussian distribution, we can produce finite samples from the conditional Gaussian process, which is then also referred to as the *posterior Gaussian process*

$$g|D \sim \mathcal{GP}(\mu_D, \kappa_D)$$

with $D = \{x_D, y_D\}$ and

$$\begin{aligned} \mu_D(x) &= \mu(x) + \kappa(x, x_D)\Sigma_D^{-1}\big(y_D - \mu(x_D)\big), \\ \kappa_D(x, x') &= \kappa(x, x') - \kappa(x, x_D)\Sigma_D^{-1}\,\kappa(x_D, x'), \end{aligned} \tag{7.2}$$

where the data covariance $\Sigma_D = \kappa(x_D, x_D)$ is constant given the data and, thus, may be precomputed. If the subvector y_1 is empty, we arrive again at

$$g \sim \mathcal{GP}(\mu, \kappa),$$

what is then accordingly called the *prior Gaussian process*.

Often, observations y may come only with additional noise, s.t. $y = f(x) + \epsilon$, which in the simplest case is assumed to be i.i.d. with $\epsilon \sim \mathcal{N}(0, \operatorname{diag}(\sigma_y^2))$. In this case, the data covariance becomes $\Sigma_D = \kappa(x_D, x_D) + \operatorname{diag}(\sigma_y^2)$ [70]. However, also in the case of noise free measurements, it is meaningful to add some small ϵ to the diagonal, as otherwise the posterior variance at some x_D, which has already been observed, becomes 0 since

$$\kappa_D(x_D, x_D) = \kappa(x_D, x_D) - \kappa(x_D, x_D)\,\kappa(x_D, x_D)^{-1}\,\kappa(x_D, x_D) = \kappa(x_D, x_D) - \kappa(x_D, x_D) = 0.$$

Clearly, the variance for a point, where we *know* the value of the process, becoming zero reflects exactly this knowledge. However, when working with Gaussian processes in a numerical setting, the resulting covariance matrix will not be positive definite anymore due to the diagonal zero entry. Therefore, standard approaches to sample normal distributions, which usually rely on computing Cholesky factors of the covariance matrix, will cease working.

7.1.2 Pre-Aggregating Data

A major caveat of Gaussian processes is the inversion of Σ_D required in (7.2), which costs $\mathcal{O}(n^3)$ for n observations $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. In general, Gaussian processes are not restricted to unique observations at x_i, that is, we may well have $x_i = x_j$ for some indices $i \neq j$, where $y_i \neq y_j$. However, observing the same location x_i multiple times, although possibly bringing some information gain, increases the size of the data and thus the dimensions of the data covariance Σ_D, slowing down computations as more data is acquired. If our data y_i is the result of some estimator based on the realisation of some underlying random variable $\theta_i = (\theta_{i1}, \ldots, \theta_{im})^\top \sim \pi_i^m$ with density depending on x_i. That is

$$y_i = \hat{f}(x_i) = \frac{1}{m}\sum_{k=1}^{m} h(\theta_{ik})$$

which is an estimator for

$$f(x_i) = \int h(\theta)\pi_i(\theta)\mathrm{d}\theta, \tag{7.3}$$

then for two observations y_i, y_j with $x_i = x_j$, we have that the average of both observations

$$\bar{y} := \frac{y_i + y_j}{2} = \frac{1}{2m}\sum_{k=1}^{m} h(\theta_{ik}) + \frac{1}{2m}\sum_{k=1}^{m} h(\theta_{jk}) = \frac{1}{2m}\sum_{k=1}^{2m} h(\theta_k) \tag{7.4}$$

is still an estimator of the exact target (7.3) as $\pi_i = \pi_j$, so

$$(\theta_i, \theta_j)^\top = (\theta_1, \ldots, \theta_{2m})^\top \sim (\pi_i^m, \pi_j^m)^\top = \pi_i^{2m} = \pi_j^{2m}.$$

The aggregated estimator from (7.4) can be expanded to update m_1 observations of the same location x_i with m_2 new observations at the same location. In that case, the aggregated estimator after $m = m_1 + m_2$ observations of x_i becomes

$$\begin{aligned}
\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i &= \frac{1}{m}\left(\sum_{i=1}^{m_1} y_i + \sum_{i=m_1+1}^{m_2} y_i\right) \\
&= \frac{m_1}{m}\frac{1}{m_1}\sum_{i=1}^{m_1} y_i + \frac{m_2}{m}\frac{1}{m_2}\sum_{i=m_1+1}^{m} y_i \\
&= \frac{m_1\bar{y}_1}{m} + \frac{m_2\bar{y}_2}{m}.
\end{aligned} \tag{7.5}$$

If further the error of y_i is estimated from parallel evaluations of $\hat{f}(x_i)$ as the variance in the estimators, they can be updated in a similar fashion. That is, let σ_1^2 and σ_2^2 be such estimates over observations of size m_1 and m_2 with mean $\bar{y}_1$ and $\bar{y}_2$ respectively, then the full variance σ^2 over all $m = m_1 + m_2$ observations can be computed as

$$\sigma^2 = \frac{m_1(\sigma_1^2 + \bar{y}_1^2)}{m} + \frac{m_2(\sigma_2^2 + \bar{y}_2^2)}{m} - \bar{y}^2 \tag{7.6}$$

with $\bar{y}$ as above.

Thus, if applying Gaussian process regression to such an estimator, we propose pre-aggregating observations following this procedure, when observing the same locations is likely, e.g. because the Gaussian process is evaluated at constant, pre-defined grid points. This will bound the dimensions of the data covariance Σ_D by the number of grid points used, therefore, making the runtime of the algorithm more controllable.

7.2 Thompson Sampling

Algorithm 3: General Thompson sampling algorithm

```
D_0 = ∅
for t = 1, 2, ... do
    draw u ~ GP(μ_{D_{t-1}}(x), κ_{D_{t-1}}(x, x))
    x_t = arg max_{x∈X} u(x)
    y_t = f(x_t)
    D_t = D_{t-1} ∪ {x_t, y_t}
end
```

Thompson Sampling was first introduced in 1933 [93] by William R. Thompson in order to minimize the number patients receiving inferior treatment, when two competing treatments

with yet unknown success rates exist. This kind of problem is nowadays also commonly known as a *multi-armed bandit problem*, referring to the slot machines – called *one-armed bandit* – in a gambling hall robbing the gambler's money [76]. The problem consists of having to decide, what bandit to play next, assuming that not all bandits have the same success probability. Thus, the gambler has to *explore* the bandits, which means playing them in order to obtain samples of their success probability, while at the same time *exploiting* this information to maximize the winnings. This underlying structure of both problems of having to balance resources between exploration and exploitation is commonly known as an *exploration-exploitation dilemma* [7].

Although Thompson Sampling has been around for already quite a while, it has gained increasing popularity in recent years [76], after its efficiency was first observed empirically [16, 79] and later its optimal expected regret upper bound could be proven for some problems[1, 2].

Conceptually, Thompson Sampling is a very simple algorithm, its details given in Algorithm 3. It starts off by assuming a distribution over the possible function shapes of the a priori unknown target function. For continuous target functions, this distribution is modelled using the Gaussian processes introduced in the preceding section. Given such a distribution, a sample is drawn from it and where this sample is maximized, the target function is evaluated. The acquired data is then used to update the posterior distribution according to (7.2) and the procedure begins anew. An exemplary optimization trajectory using Thompson sampling is visualized in Figure 7.2.

7.3 Tuning Proposal Distributions with Thompson Sampling

As mentioned before in Chapter 6, hyperparameter tuning of the Metropolis-Hastings algorithm is in general not feasible without performing preliminary numerical simulations to roughly estimate suitable efficiency measures. Clearly, these preliminary estimates themselves however are subject to estimation errors, especially since estimates over not yet converged samples might be strongly influenced by local effects. This possibly large estimation error makes Bayesian optimization a good candidate in finding optima over such hard to estimate functions. Nevertheless, estimates might improve as the chain "warms up", i.e. it starts to move within the higher density regions. As such, hyperparameter tuning of the Metropolis-Hastings algorithm is not only an exploration problem, but inherently also admits an exploitative nature in that testing good hyperparameters will move the chain faster to equilibrium and, thus, also improve estimates for other hyperparameter sets.

The influence of local effects in estimation might introduce a strong bias when using preliminary simulations of the Metropolis chain. Consider as an example a Gaussian random walk on a standard Gaussian target starting "far away" from the mean. In such a case, large steps "up hill", i.e. in direction towards the mean, are likely to be accepted resulting in very large jump distances for large step sizes. However, once the chain arrives to the Gaussian peak, the large step size will mostly lead to proposals in very low probability regions and thus the chain might barely move as the Metropolis filter will mostly reject such moves. Hence, this example suggests that we are particularly interested in hyperparameters, that work well for warmed-up chains. A heuristical approach to heating up chains quickly, might be starting the Metropolis chain in some local optimum. Using multistart optimizations, it even might be feasible to gain multiple starting points and thus simulate parallel chains starting from different states. Since the convergence of the Metropolis chain is independent of its starting point as long as the starting point comes from the support of the target distribution, such an approach is indeed safe.

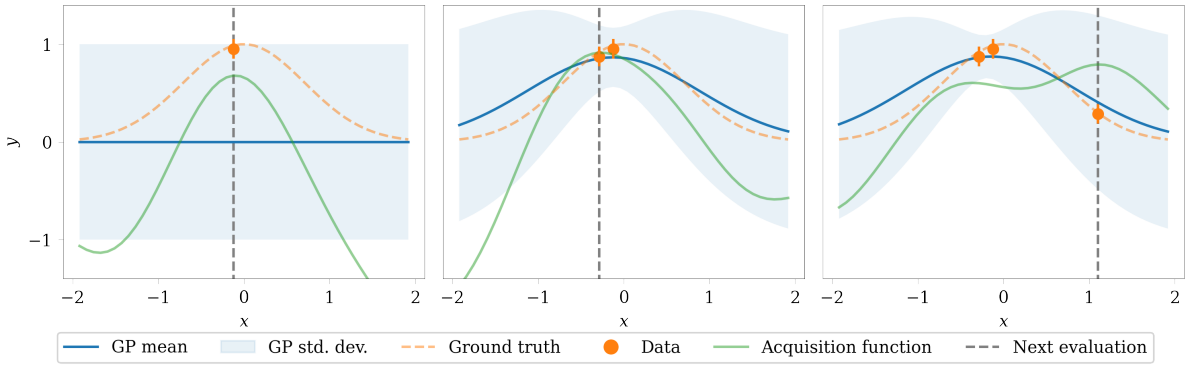


Figure 7.2: Thompson sampling trajectory for a simple toy problem of a squared exponential target $f(x) = \exp\{-x^2\} + \epsilon$ with Gaussian independent noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Nevertheless, even in multistart optimization optima might be overlooked. Good exploration of the state space by the Metropolis chain thus remains important even during the preliminary tuning phase, emphasizing the exploitative aspect of our tuning problem. These considerations led us to choose Thompson sampling as a suitable approach to perform hyperparameter tuning of proposal distributions for the Metropolis-Hastings algorithm.

7.3.1 The Algorithm

At its core, the Algorithm 4 follows Algorithm 3 quite closely, however three adaptations are introduced:

- Using the idea of data pre-aggregation and observing, that the expected squared jump distance as well as the acceptance rate are indeed expectations over $\pi\mathcal{K}$, we reduce the number of total observations by taking averages whenever a step size gets observed multiple times. This step aims at controlling the costs of the matrix inversion required in (7.2). The corresponding steps are given in Algorithm 4, line 20 to 28.

- Unlike other Gaussian process applications, which often assume i.i.d. errors in the data observation [70], one can do better in Markov chain Monte Carlo estimation by running parallel chains and estimate the variance of the estimation across the parallel chains. However, since only few samples are used for the preliminary estimations, the error estimate itself may suffer from large estimation errors. Thus, we smooth the error estimates across multiple step sizes. This approach is discussed in some more detail in Section 7.3.3. The corresponding steps are given in Algorithm 4, line 29 to 32.

- At last, a convergence criterion is introduced. That is, if the maximum of the posterior mean remains unchanged over some pre-defined number of iterations, then the algorithm is assumed to have converged. The corresponding steps are given in Algorithm 4, line 6 to 13.

Since most proposal algorithms presented in Chapter 5 only admit a single hyperparameter, namely the step size, and grids on multiple dimensions grow exponentially, we focused exclusively on step size tuning. Except of the quickly rising costs when applying the method to multiple hyperparameters it can be generalized trivially to higher dimensions. An approach to mitigate the drawbacks of a grid-based approach will be discussed in Section 10.1.

Algorithm 4: Thompson sampling MCMC tuning algorithm

Data: Search grid $s = (s_1, \dots, s_h) \ni s_i > 0, h \geq 1$, Markov chains $\mathcal{K} = (\mathcal{K}_1, \dots, \mathcal{K}_M), M \geq 1$, number of test samples $n \geq 1$, optimization iterations $N \geq 1$, iterations for convergence $N_0 \geq 1$

$D_0 = \emptyset$
$\mu_{D_0}(x) = \mu(x) = 0$
$\kappa_{D_0}(x, x') = \kappa(x, x') = \exp\{-\|x - x'\|_2^2\}$
$i^* = 1, k^* = 0$
for $t = 1, 2, \dots, N$ **do**
  if $i^* = \arg\max_{i=1,\dots,h} \mu_{D_{t-1}}(s)_i$ **then**
    $k^* = k^* + 1$
    if $k^* = N_0$ **then**
      break
    end
  else
    $k^* = 0$
  end
  draw $u = (u_1, \dots, u_h) \sim \mathcal{N}\left(\mu_{D_{t-1}}(s), \kappa_{D_{t-1}}(s)\right)$
  $i' = \arg\max_{i=1,\dots,h} u_i$
  for $j = 1, 2, \dots, m$ **do**
    draw $x_j = (x_{j1}, \dots, x_{jn}) \sim \mathcal{K}_i^n(x_0, \cdot \mid s_{i'})$
    $\hat{y}_j = 1/n \sum_{k=1}^n f(x_{jk})$
  end
  if $s_{i'} \notin D_t$ **then**
    $y_{i'} = 1/M \sum_{i=1}^M \hat{y}_i$
    $\sigma_{i'}^2 = 1/M \sum_{i=1}^M (\hat{y}_i - y_{i'})^2$
    $k_{i'} = 1$
  else
    update $y_{i'}$ as in (7.5) using $(\hat{y}_j)_{j=1}^M$
    update $\sigma_{i'}$ as in (7.6) using $(\hat{y}_j)_{j=1}^M$
    $k_{i'} = k_{i'} + 1$
  end
  $s' = (s : s \in D_{t-1} \cup \{s_{i'}, y_{i'}, \sigma_{i'}^2, k_{i'}\})$
  $\sigma^2 = (\sigma^2 : \sigma^2 \in D_{t-1} \cup \{s_{i'}, y_{i'}, \sigma_{i'}^2, k_{i'}\})$
  $W = k(s', s')$
  $\hat{\sigma}_i^2 = \frac{\sum_j w_{ij} \sigma_j^2}{\sum_j w_{ij}}$
  $D_t = D_{t-1} \cup \{s_{i'}, y_{i'}, \sigma_{i'}^2, \hat{\sigma}_{i'}^2, k_{i'}\}$
end

Clearly the most important parameter of Algorithm 4 is the test sample size n, which will be used within every round to assess the candidate step size's performance. As with the general question of how long a Metropolis chain should be simulated to reach convergence, there exists no clear guideline on how many samples to use. Typically, one will define a number of samples to be drawn before assessing convergence when doing regular sampling with the Metropolis-Hastings algorithm. If the Metropolis chain is not yet converged, more samples will be drawn and the process is repeated until either convergence is diagnosed or an upper limit of samples has been drawn. Practical experience presented the approach to take approximately $100\,000 \cdot d$ to $1\,000\,000 \cdot d$ samples with d being the dimension of the problem to be a meaningful guideline. Since tuning is supposed to happen a priori and in order to speed up the convergence of the chain, it is clear, that one wishes to achieve good tuning results using less samples.

The number of optimization iterations is closely related to the number of test samples. The product of iterations and test samples gives an upper bound on samples, which will be invested into sampling. Since multiple observations of the same step size will be aggregated, re-measuring the same step size multiple times is equivalent to measuring it once using more samples. Nevertheless, setting a higher number of optimization iterations while reducing the number of test samples per iteration, s.t. a constant upper bound of samples is maintained, allows for more exploration.

Another important parameter of Algorithm 4 is the number of parallel chains to be used. In order to get any kind of error estimate from computing cross-chain variances, a minimum of 2 parallel chains is required. However, more chains are clearly recommended to achieve more robust error estimates. In general, multiple chains can be parallelized trivially and thus little additional cost is to be expected, if the computer architecture has enough cores available. Within this work, we usually set the number of chains to 10.

Preliminary test runs as well as the results presented in Section 9.1.3 suggest using a log-uniform grid on the step size. Simply put, finding the right order of magnitude within the optimal step sizes lives is key in achieving higher statistical efficiency of the Metropolis chain's samples. Defining the search grid is left to the user. Clearly, if good prior knowledge exists on what step sizes might perform well, the grid should be chosen appropriately as a large search space will require more time to explore it.

Although there exist many possible choices for kernels of the Gaussian process used to model the distribution over the tuning target, we chose a squared exponential kernel, which may be considered a standard choice for smooth target functions [11, 24]. The same results, which motivated using a log-uniform grid also suggested to fix the kernel's length parameter to 1, meaning that observing the performance of a step size tells us something about the behaviour of the performance within a range of ± 1 order of magnitude. The scale parameter mostly influences the initial variance of the Gaussian process, which will be quickly dominated by the estimated variance of the observations. Thus, we heuristically also set it to 1.

Finally, the number of iterations N_0 to assess convergence has to be specified. Since the algorithm will not stop before having performed at least N_0 iterations, it forms a lower limit on the number of iterations. Considering that in the beginning the probability to draw a candidate step size is almost uniformly distributed, it is important to set N_0 with respect to the grid size, as for a large grid the probability to draw N_0 step sizes without much affecting the posterior mean while at the same time not having explored much of the space is high. A meaningful heuristic might be to choose N_0 as some multiple of the orders of magnitude covered by the grid.

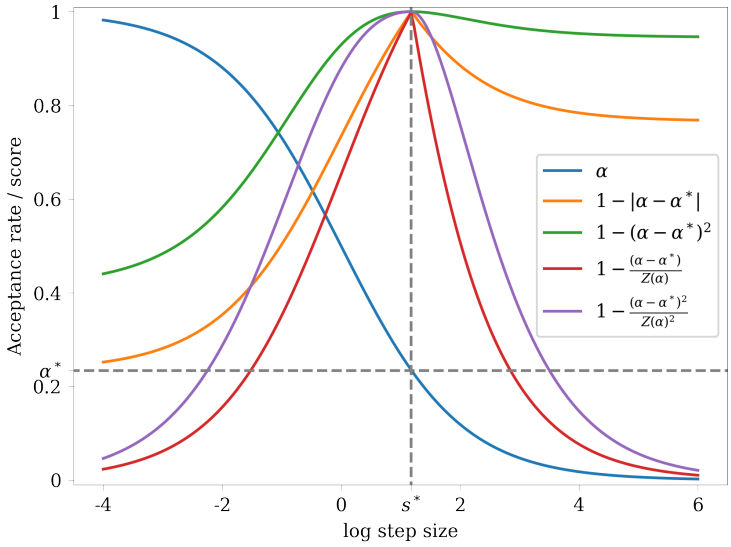


Figure 7.3: Transformations leading to the score function (7.7) employed in acceptance rate tuning. A decreasing logistic curve was used as a model of the acceptance rate as a function of the step size. We refer to $s \geq s^*$ as the "heavier" side under $1 - |\alpha - \alpha^*|^p$. We use a short hand notation for the denominator $Z(\alpha) = \alpha \mathbb{1}_{\alpha \leq \alpha^*} + (1 - \alpha)\mathbb{1}_{\alpha > \alpha^*}$

7.3.2 Tuning Targets

Following the considerations from Chapter 6, the expected squared jump distance is an appropriate target function for assessing quality of a particular hyperparameter choice. Since our goal is to increase the expected squared jump distance, no further transformations are required to plug this target into the Thompson sampling algorithm.

For particular problems acceptance rate tuning can be a good choice as a tuning target. In order to compare the effect of both targets, we were also interested in robust acceptance rate tuning. Typically, the acceptance rate is assumed to be a strictly decreasing function in the step size and thus the problem of finding the step size s^* at which a particular target acceptance rate α^* is achieved is commonly tackled by employing a binary search. That is, fixing some lower and upper bound on the step size, a fixed number of preliminary test samples is drawn using the step size splitting the search interval into two halves. The acceptance rate is then estimated using these samples. If the estimated acceptance rate is larger than the target acceptance rate, then the procedure is repeated on the right half of the split interval, otherwise it is repeated on the left half.

However, in doing so, we encountered severe issues of this approach getting stuck when the acceptance rates are very noisy, e.g. because to few test samples are used. That is, assume the step size 2 achieves a desired acceptance rate of 0.234 and a search interval $[1, 4]$ is used. Then the step size 2.5 will get tested first, but, due to high estimation errors, it might achieve an acceptance rate of 0.3, then clearly the search will continue on the interval $[2.5, 4]$. In such a case, the correct step size 2 can not be found anymore due to the deterministic manner of search. Thus, the naive binary search approach is not capable to handle the stochastic noise inherent to the problem.

Clearly, the Thompson Sampling approach stated in Algorithm 4 is not directly applicable

to intersection problems. Nevertheless, by observing, that $\alpha(s^*) = \alpha^*$ is equivalent to $|\alpha(s^*) - \alpha^*|^p = 0$ for any $1 \leq p < \infty$ and $|\alpha(s) - \alpha^*|^p \geq 0$, we can reformulate the intersection problem as an optimization problem. This reformulation yields an asymmetric peak at s^* as can be observed in Figure 7.3. Using Algorithm 4 would favor testing of step sizes on the "heavier" side of the peak, which is $s \leq s^*$ if $\alpha^* \geq 0.5$ and $s \geq s^*$ if $\alpha^* \leq 0.5$. Thus, we propose the following *score function*

$$\hat{\alpha}(s) = \begin{cases} 1 - \left(\frac{|\alpha(s)-\alpha^*|}{\alpha^*}\right)^p & \text{if } \alpha(s) \leq \alpha^* \\ 1 - \left(\frac{|\alpha(s)-\alpha^*|}{1-\alpha^*}\right)^p & \text{if } \alpha(s) > \alpha^* \end{cases}. \tag{7.7}$$

Note that the term in the nominator yields a peak at s^* with $\hat{\alpha}(s^*) = 1$, turning the problem into a maximization problem as required by Algorithm 4. The choice $p = 1$ is straightforward, however, for $p = 2$ the score function (7.7) becomes more smooth, which is supposed to work better when approximating the score function using a squared exponential kernel.

7.3.3 Error Estimation & Smoothing

As we mentioned earlier in the description of the tuning algorithm, instead of assuming some fixed error on the performance estimates, we can estimate the error itself by running parallel chains with the same hyperparameter set and computing the variance over the performance estimates. As such, we end up estimating the standard error of the performance estimate similar to the standard error in the mean presented in Section 6.1.1. However, these numerical estimates themselves are subject to estimation errors. Although the error will clearly converge towards zero in the limit of taking infinitely many samples, it can be arbitrarily large or small for finite sample sizes.

A particular phenomenon arises in the case of testing large step sizes. The probability of the Metropolis chain to leave the current state x can be given as

$$p_{\text{jump},s}(x) = \int_{\mathcal{S}} \alpha_s(x, y)\, q_s(x, \mathrm{d}y)$$

and the probability to leave x is thus geometrically distributed with success probability $p_{\text{jump},s}(x)$. Hence, in expectation it will take the chain $n_{\text{jump},s} = 1/\, p_{\text{jump},s}(x)$ attempts to finally leave x. As already mentioned in Section 6.1.3, the acceptance rate is typically assumed to converge against 0 for large step sizes. Thus, if $p_{\text{jump},s}$ decreases for increasing s, then clearly $n_{\text{jump},s}$ increases. So when performing short preliminary runs to estimate performance measures, it is likely that the chain remains put, if the number of test samples $n < n_{\text{jump},s}$. In particular, if this happens for all parallel chains, which may be the case especially if $n \ll n_{\text{jump},s}$, then the the variance in the performance estimates will be 0, which can lead to numerical issues as described at the end of Section 7.1.1.

As a remedy, we smooth the squared error by averaging it over all squared error estimates in an ℓ-neighborhood. This is implemented using a uniform ball kernel

$$k(s, s') = \begin{cases} 1, & \text{if } \| s - s' \|_2 \leq \ell, \\ 0, & \text{else,} \end{cases}$$

which is then used to compute a weight matrix W with $w_{ij} = k(s_i, s_j)$ using all observed data points. Given the vector of estimated squared erors σ^2, we compute the smoothed squared error

at x_i as

$$\hat{\sigma}_i^2 = \frac{\sum_j w_{ij}\sigma_j^2}{\sum_j w_{ij}}.$$

Chapter 8

Implementation

In order to realize the experiments done throughout the course of this thesis, the open source Python toolbox `hopsy` was developed. `hopsy` is a library offering Python bindings of the most important features of the high-performance polytope sampling library `HOPS`[1] [42], implemented in C++ using `pybind11`[2] [43]. Thus, `hopsy` allows to use the powerful C++ implementation via a comfortable and flexible Python interface, allowing for much faster implementation of experimental and visualization pipelines. `hopsy` is currently available for Python 3.7 to 3.10 for Windows and Linux via the Python package index[3].

Unlike the Python packages `dingo`[4] or `COBRApy`[5], which also claim to support polytope sampling, `hopsy` is not restricted to the analysis of metabolic network models. As such, it offers many more algorithms and is capable of sampling arbitrary truncated densities on the polytope. In particular, `COBRApy` only offers two sampling algorihtms, namely the Artifically Centered Hit-and-Run algorithm [46] and optGPSampler [55], of which for both no convergence has been proven, and `dingo` offers only a Multiphase Monte Carlo algorithm [15]. In both tools, the polytope can only be constructed from the metabolic model, which renders them awkward to use in other, unrelated fields. As such, `hopsy` follows a strict single-responsibility principle [54].

In `hopsy`, a sampling problem is defined by two main components:

- the polytope $\mathcal{P}$, defined by the left-hand side linear operator A and the bounds b in the polytope-defining inequality $A\,x \leq b$,
- a model, which defines an arbitrary, pointwise evaluable density.

Thus, it becomes responsibility of the user or third-party software to define the sampling problem. This problem can then be sampled with any of the provided algorithms:

- Adaptive Metropolis [34],
- Ball Walk [37],
- CSmMALA [85],
- Dikin Walk [45],

[1] `https://modsim.github.io/hops/doxygen/index.html`
[2] `https://pybind11.readthedocs.io/`
[3] `https://pypi.org/`
[4] `https://github.com/GeomScale/dingo`
[5] `https://opencobra.github.io/cobrapy/`

- Gaussian Random Walk,
- various Hit & Run algorithms [14].

As it was mentioned in Section 5.2, polytope sampling problems can be preconditioned by transforming the polytope into a more isotropic shape. `hopsy` offers convenience functions for performing rounding transformations using the polytope rounding library `PolyRound`[6] [86].

Since convergence diagnostics are an important component when doing Markov chain Monte Carlo simulations, we added convenience functions for computing effective sample size and potential scale reduction factors. Under the hood, these methods use the specialized `arviz`[7] library [47], which is a widespread tool for Markov chain convergence diagnostics also used by popular software packages like `PyMC3`[8], `stan`[9], `emcee`[10] and many others.

8.1 Best Practices

Research software often fails to meet common quality standards, impairing its sustainability, reproducibility and reusability [21]. In order to allow `hopsy` to become an useful, robust and easily accessible software tool for polytope sampling, we took care to follow common best practices in software development. In particular, `hopsy` development is setup on a corporate GitLab server, where versions and changes are tracked using `git`[11]. A public version of the source code of all releases is available on GitHub[12]. Further, a continuous integration pipeline has been installed, which automatically builds and tests `hopsy`, allowing to detect erroneous changes early during development.

Installation of software, which requires compilation, can often be tedious and was previously a hurdle to using `HOPS`. Thus, `hopsy` is continuously built within `manylinux` environments[13] which allow convenient distribution of pre-compiled and, thus, easy to install software packages.

To further facilitate usage, `hopsy` comes with a publicly available documentation and software reference[14]. Usage examples and detailed description of most of the software's components are provided.

8.2 Combining With Third-Party Software

Since the focus of this work lies on solving sampling problem for Bayesian ^{13}C metabolic flux analysis, combining `hopsy` with third-party software was an important design goal from the beginning on. In particular, the high-performance ^{13}C simulator `x3cflux` and its Python interface `x3cfluxpy` were used to simulate isotope labeling experiments as discussed in Section 3.1.

`hopsy` allows integration of third-party software by providing a simple interface. In order to sample the distribution defined by an arbitrary, positive density, a model in `hopsy` only has to provide a negative log-likelihood function. Thanks to the `pybind11` software, `hopsy` can take

[6] `https://pypi.org/project/PolyRound`
[7] `https://arviz-devs.github.io/arviz/`
[8] `https://docs.pymc.io/en/v3/`
[9] `https://mc-stan.org/`
[10] `https://emcee.readthedocs.io/en/stable/`
[11] `https://www.git-scm.com/`
[12] `https://github.com/modsim/hopsy`
[13] `https://github.com/pypa/manylinux`
[14] `https://modsim.github.io/hopsy`

an arbitrary Python object and try to treat it as a valid model by checking during runtime, if the passed object provides the required function. Further, the passed object does not need to be implemented in Python, but may itself be e.g. another C++ object providing appropriate Python bindings. Clearly, this allows to implement powerful simulators and sampling algorithms absolutely independent from each other and still combine them within Python.

Due to its flexibility, `hopsy` has already found application in other settings, such as starting point generation for optimization problems in chromatographic process design [78] and for soft matter simulations of magnetic filaments[15].

8.3 Usage Example

We present a short usage example in order to emphasize the simplicity of usage of `hopsy` in the following Figure 8.1.

```
import hopsy
import numpy as np
import matplotlib.pyplot as plt

problem = hopsy.Problem([[2, 1], [-1, 0], [0, -1]], [5, 0, 0],
↪  hopsy.Gaussian(dim=2))

mc = hopsy.MarkovChain(problem, starting_point = [.1, .1])
rng = hopsy.RandomNumberGenerator(42)

acceptance_rate, draws = hopsy.sample(mc, rng, n_samples = 5000)

plt.scatter(draws[:,:,0], draws[:,:,1], alpha=.2)
plt.ylim([-1, 4])
plt.xlim([-1, 4])
plt.show()
```

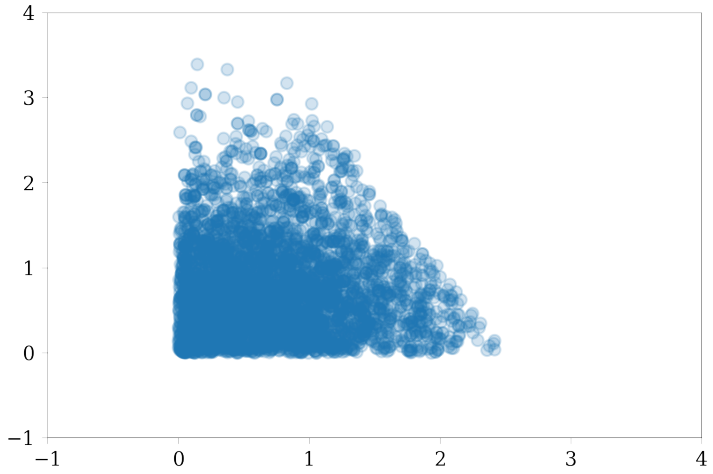


Figure 8.1: Code example of `hopsy` for sampling a truncated Gaussian distribution in a simplex.

[15] `https://github.com/ripaul/magnetic-filaments`

Chapter 9

Experimental Setup & Results

The two mains questions, we aim to answer by the means of numerical experiments throughout this chapter, are

- Does tuning w.r.t. the expected squared jump distance outperform tuning w.r.t. the acceptance rate in terms of statistical efficiency?
- Is Algorithm 4 able to find the optimal hyperparameters found using a brute-force approach?

All experiments were run on an Ubuntu 18.04.4 LTS system with 64 Intel® Xeon® CPU E5-2683 v4 cores with 2.10GHz each and 128GB memory. The experiments were setup as Jupyter notebooks[1] in `Python 3.8`[2] using the polytope sampling library `hopsy` and, where required, the high-performance ^{13}C simulator `x3cflux`[99]. In order to enhance reproducibility, all experiments were run within a `docker`[3] container, a light weight virtualization solution. The notebooks as well as the container configuration file used to setup the environment and produce the upcoming results are made publicly available[4].

9.1 Brute Force Optimal Step Sizes

As stated in Section 6.1.2 and Section 6.2, the restrictive assumptions on the target distribution required for optimal efficiency being achieved at some predefined acceptance rate suggest that optimal step sizes for sampling linearly constrained problems might yield smaller acceptance rates. The aim of this section is, thus, to check this hypothesis and validate empirically, that tuning w.r.t. the expected squared jump distance yields improved efficiency. Using a brute force approach, we study a range of different example problems, which convey two of the main characteristics often encountered in real-world problems: multimodality and non-identifiability.

Our experiments are setup as follows. Given a particular sampling problem and proposal algorithm, we fix a grid of step sizes, which we want to test, and then let m parallel Metropolis chains run for every step sizes. Chains are then either run until convergence is diagnosed or an upper limit of samples has been drawn. Note that this is not a good hyperparametrization approach at all, since we are solving the question for good hyperparameters in a hen-egg fashion, by already running chains until convergence. However, we will then use the obtained results as a

[1] `https://jupyter.org/`
[2] `https://www.python.org/`
[3] `https://www.docker.com/`
[4] `https://github.com/ripaul/mscthesis`

ground truth to evaluate Algorithm 4. The samples obtained from the brute force experiments are used estimate the following quantities and statistics:

Potential scale reduction factor Monitoring the convergence of the chain is important in order to determine robustness of the estimates. A high potential scale reduction factor suggests that the chains have not well mixed and thus estimates using their samples might still be strongly influenced by local effects (cf. Section 4.2.1). We expect the potential scale reduction factor to be closer to 1, the more efficient a particular step size choice is.

Efficiency As described in Section 6.1, the effective sample size is one of the most common measures to assess the statistical efficiency and, thus, the convergence of estimates based on the obtained samples. High effective sample sizes suggest, that consecutive samples show little correlation and are thus "more independent" from each other. Since our chains might drawn different numbers of samples in total, we consider the relative effective sample size normalized by the number of drawn samples. We refer to this quantity as the *efficiency* $r_{\text{eff}} = \tau^{-1}$, where τ is the integrated autocorrelation time. The efficiency will be our ground truth defining measure of optimality, meaning that we will consider step sizes which optimize r_{eff} as the optimal step size. As suggested in [31], the efficiency is computed separately for every dimension and the minimal effective sample size is considered. This approach thus requires good statistical efficiency for all dimensions. In order to consider time costs as well, we consider the efficiency per second, which reports how many uncorrelated samples the chain produces per second.

Expected squared jump distance The expected squared k-jump distance directly relates to the autocorrelation coefficient of the samples at lag k and is therefore assumed to be a good tuning target in order to increase the effective sample size, while at the same time being much easier to estimate. Within this experiment, we consider only $k = 1$, to which we simply refer to as the expected squared jump distance (ESJD). Strong agreement of the optima of the expected squared jump distance and efficiency indicate, that the expected squared jump distance is indeed a meaningful tuning target.

Acceptance rate The acceptance rate is the classical text book approach for step size tuning. Typically, the acceptance rate is assumed to be a decreasing function, approaching 1 as the step size decreases and 0 as it increases. We conjecture that for the more complex problems within this experiment, the acceptance rate will achieve the value 0.234 (or 0.574) at a suboptimal step size w.r.t. the efficiency. We denote the target acceptance rate as α^*, which takes value 0.574 for the CSmMALA algorithm and 0.234 for all other algorithms.

Average time per sample In Section 6.2, we suggested an argument how the presence of linear constraints might give raise to varying costs per sample depending on the step size. Further, we outlined how this phenomenon could influence a shift in optimality, when considering the number of effective samples per fixed unit of time. Thus, we keep track of the average time per sample as a function of the step size in order to check our hypothesis.

9.1.1 Example Problems

The following example problems were constructed from the *Spiralus* model [101]. The network topology is visualized in Figure 9.1. Although the *Spiralus* is a toy ^{13}C model, it was designed to incorporate many of important features of real metabolic network models [102], such as

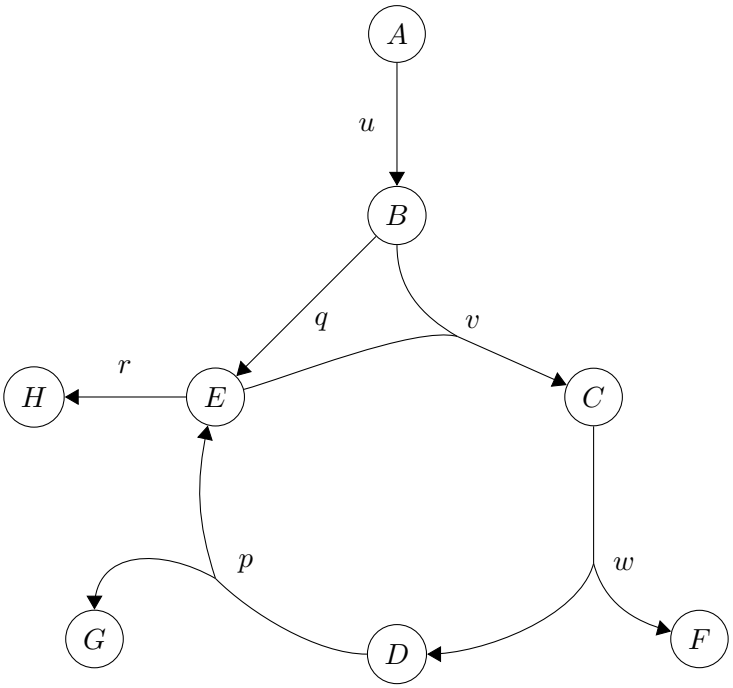


Figure 9.1: Spiralus network model as presented in [102]. This model usually admits two free fluxes, which in our case are u and q.

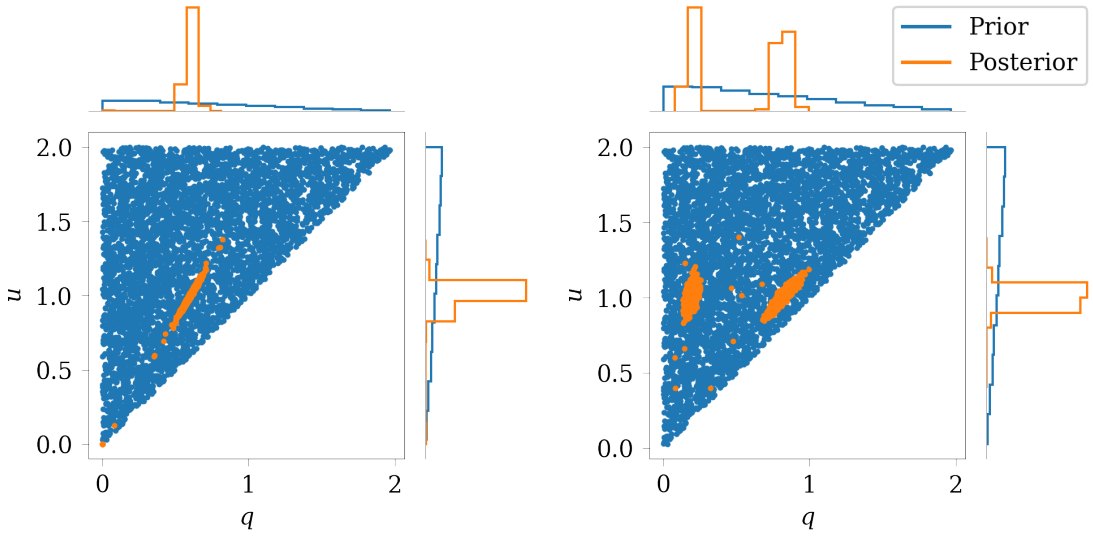
admitting cycles and non-linear reactions. In its stationary variant, the Spiralus model admits 2 free parameters. Synthetic measurement data was produced in order to obtain an unimodal and a bimodal parameter estimation problem. Slight modifications on the model were introduced, in order to further introduce non-identifiable parameters. For validation, we also considered a practically unconstrained standard Gaussian in 20 dimensions. We continue by presenting every problem in brief before discussing the experimental setup.

Standard Gaussian (Gauss) For the Gaussian model, we expect the optima of effective sample size and expected squared jump distance to agree with the step size, that achieves α^*. An example plot of the 2-dimensional marginal posteriors is given in Figure A.1 in the appendix.

Stationary unimodal Spiralus (STAT-1) The stationary base model defines a two-dimensional parameter estimation problem with two positivity constraints on the free parameters, i.e. fluxes, u, q. A third constraint halves the first quadrant with $q \leq u$. The single mode of the posterior distribution can be found at $(u^*, q^*) = (1, 0.6)$.

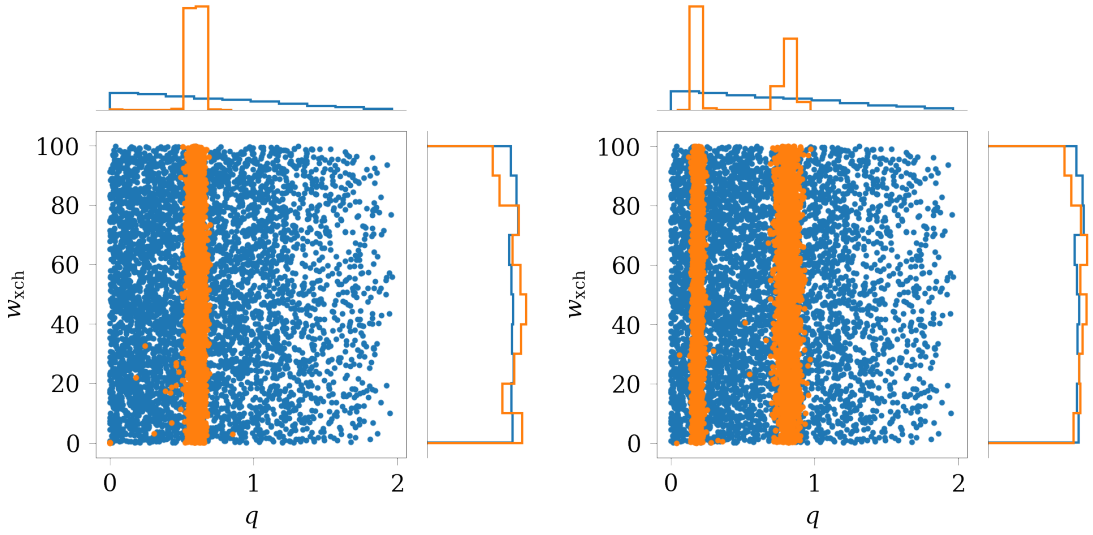
Stationary unimodal Spiralus with non-identifiability (STAT-1-ni) By releasing the fixed directionality on w an exchange flux is introduced, which accounts for the reversibility of the corresponding reaction. This increments the dimensionality of the problem to 3. In the particular problem at hand, the introduced exchange flux is non-identifiable (cf. Figure 9.2c). Since non-identifiabilities may often lead to uniform distributions, which are only well-defined for domains with finite Lebesgue measure, we apply box constraints, s.t. $0 \leq w_{\text{xch}} \leq 1000$.

Stationary bimodal Spiralus (STAT-2) In order to obtain a bimodal posterior distribution, we omitted certain isotopomer fractions, s.t. there exist two parameter combinations with the same



(a) STAT-1 problem admitting a single mode on a triangular domain.

(b) STAT-2 problem admitting two modes on a triangular domain.



(c) STAT-1-ni problem admitting two modes and one non-identifiability.

(d) STAT-2-ni problem admitting two modes and one non-identifiability.

Figure 9.2: Marginal prior and posterior distributions of the example problems presented in Section 9.1.1. Prior is the uniform distribution on the polytope and, thus, visualizes the shape of the domain.

stationary solution of (3.5). This results in two distinct locally optimal solutions as depicted in Figure 9.2b. The dimension of the problem remains untouched and is therefore 2. As is visible in Figure 3.6, multimodalities and even more complex shapes are commonly encountered for real-world problems.

Stationary bimodal Spiralus with non-identifiability (STAT-2-ni) Combining STAT-2 and STAT-1-ni yields a bimodal model with another unidentifiable flux. As with STAT-1-ni, this approach adds another dimension to the problem, resulting again in 3 dimensions.

A graphical overview of the most important features of every example problem is given in Figure 9.2. The full corner plots can be found in Figures A.1 to A.5 in the appendix.

9.1.2 Markov Chain Setup

Every example problem was sampled using the following algorithms from Section 5.1:

- Ball walk,
- CSmMALA with Fisher weight 0.5,
- Dikin walk,
- Gaussian random walk,
- Gaussian Hit & Run.

For the Ball walk, Gaussian random walk and Gaussian Hit & Run sampler, mixing time depends on the sandwiching ratio of the polytope. Thus, these algorithms were also tested on rounded polytopes (cf. Section 5.2). Rounding transformation were computed using `PolyRound`.

This setup results in a total of 40 different combinations of proposal algorithms and problems. Every such combination was setup to run a total of 50 parallel Metropolis chains, of which always 10 were grouped together. Every such group shared a common seed, but used a different stream of the PCG-64 [61] random number generator. For more robust estimation, 5 different seeds were tested. Groups of chains were formed, since the potential scale reduction factor by design requires multiple chains to assess convergence.

Starting points of each chain were drawn randomly from the uniform distribution on the domain of the problems. This is required for robust estimation of the potential scale reduction factor $\hat{R}$ as we discussed in Section 4.2.1. Uniform samples were produced in a preliminary step. All chains were then run for $10\,000 \cdot d$ iterations, before assessing convergence using the $\hat{R}$ statistics. If $\hat{R} < 1.05$, the simulation ended assuming convergence of the chains. Otherwise, another $10\,000 \cdot d$ iterations of the Metropolis-Hastings algorithm were performed before assessing $\hat{R}$ again. This procedure was then repeated until either convergence was diagnosed or an upper limit of 50 outer iterations had been used. As a result, every problem was run for at most $510\,000 \cdot d$ iterations of the Metropolis-Hastings algorithm.

For every proposal-problem combination, an individual grid of step sizes which we want to test was constructed. Preliminary runs of the simulation with a simple, log-uniform grid of step sizes and using only few samples were used to decide what range of step sizes to test (data not shown). For every problem, the step size grid was setup to be more dense around the expected maximum of the final curve and more sparse when further away. In particular, step sizes were chosen with log-uniform spacing, which was reduced in the dense region. All step sizes were between 1e-5 and 1e3. This resulted in a total number of 13 085 individual sampling problems.

Simulation of all of these problems was parallelized on 60 cores of the aforementioned computer system and took around 300 hours in total.

9.1.3 Results

The results of the brute force experiment are presented in Figure 9.3 and Figure A.6 in the appendix. In Figure 9.3, we show the acceptance rate, potential scale reduction factor, expected squared jump distance and effective sample size as functions of the step size parameter. Figure A.6 shows similar plots, but considers computational costs and, thus, includes the effective sample size per second and expected squared jump distance per second. Since inference is biased for non-convergent samples, only samples from convergent runs should be considered reliable. We encode this information by using solid lines where $\hat{R} < 1.05$ and transparent lines else. Further, the target acceptance rate $\alpha^* = 0.234$ (0.574 for CSmMALA) is marked by a horizontal gray dotted line. The corresponding step size s^*_α achieving this acceptance rate is marked by vertical gray dotted line. All values, except the potential scale reduction factor $\hat{R}$ and the acceptance rate α, were normalized by the maximum value on the convergent step sizes.

Gauss As expected, for the Gaussian target distribution sampled with the Gaussian random walk sampler, optimal efficiency is indeed achieved where the acceptance rate $\alpha = \alpha^*$. This also holds in the case, where we used rounding, which is expected. Note, that for box constraints, the rounding transformation is actually the identity. However, rounding with `PolyRound` also scales the polytope s.t. the maximal contained ball has radius 1, which explains the change of scale of the optimal step size.

Similar results are indeed visible for most other samplers as well, which is not expected with regard to the literature. However, the Hit & Run algorithm seems to achieve higher efficiency at smaller step sizes and thus larger acceptance rate. Further, also the CSmMALA algorithm does not seem to achieve the theoretically optimal acceptance rate, but peaks at a slightly larger step size.

As we expected from the theoretical considerations in Section 6.1.3, the optima of effective sample size and expected squared jump distance do indeed agree. Overall, we conclude by observing that the Gaussian target is very well behaved and confirms many of the theoretical results.

STAT-1 Considering the unimodal Spiralus without non-identifiabilities, things start to become more interesting. None of the algorithms achieves optimal efficiency at s^*_α. However, for most proposals, there is still a high agreement between expected squared jump distance and the effective sample size. The Hit & Run sampler forms an exception and admits two peaks of the expected squared jump distance, both in the rounded and non-rounded setting. While one of the peaks agrees with the optimum of the effective sample size, the other one does not and instead achieves a comparably poor effective sample size. Indeed, such behaviour is in line with the theoretical considerations from Section 6.1.3, as minimizing the autocorrelation at lag 1 does still allow the autocorrelation at higher lags to be large. A possible remedy already discussed in Section 6.1.3 is tuning w.r.t. to the k-ESJD for $k \geq 1$, which is considered in Section 9.2.

STAT-1-ni Moving to the unimodal Spiralus with a non-identifiable parameter, we observe that the introduction of a non-identifiability has largely aggravated the sampling problem. As such, only CSmMALA and Dikin walk managed to reach convergence at all. For the Dikin walk, again good agreement between effective sample size and expected squared jump distance

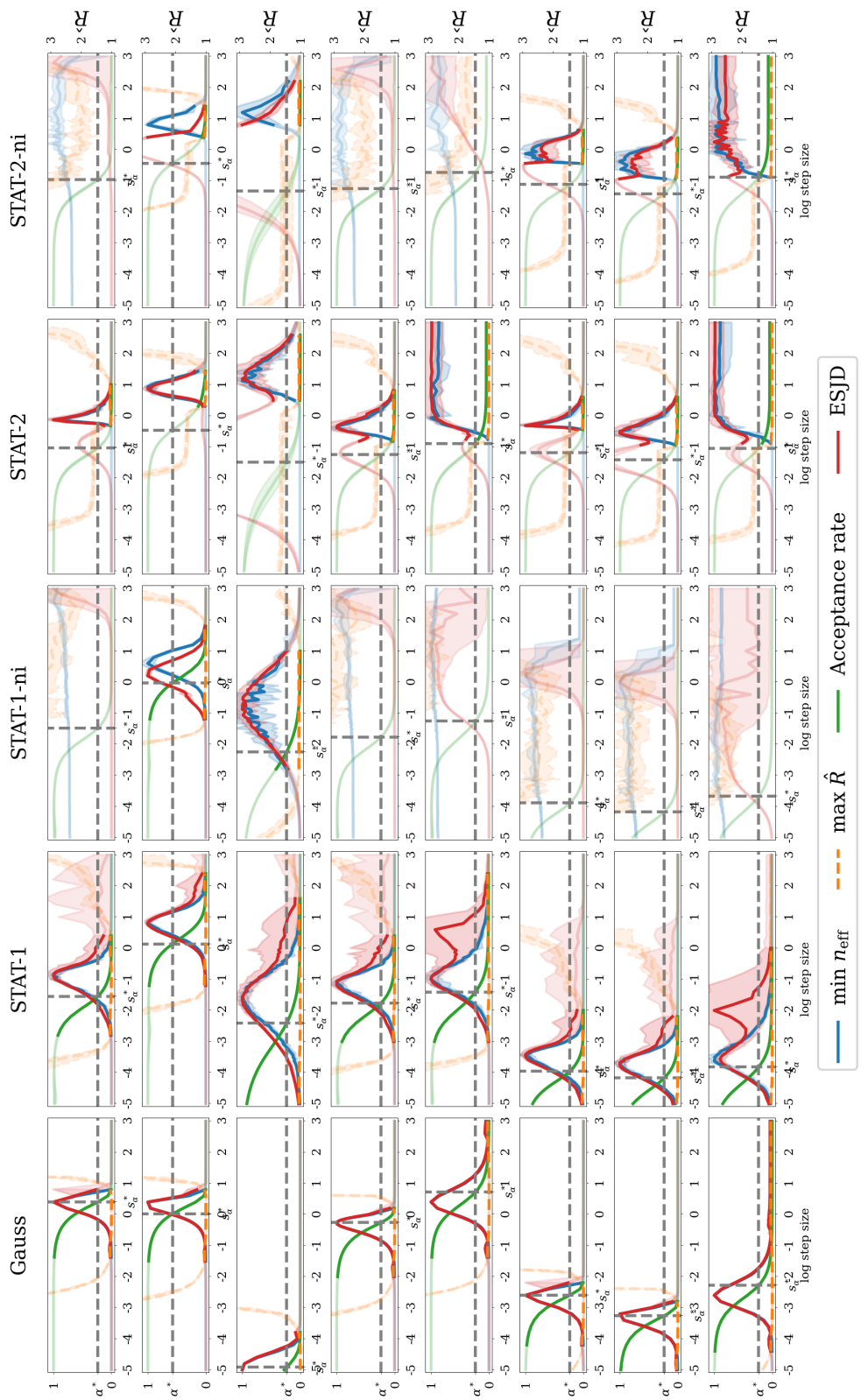


Figure 9.3: Results of the brute force experiment described in Section 9.1. For every proposal-problem combination, each step size from a grid on $[1e{-}5, 1e3]$ was run until convergence or a maximal number of $510\,000 \cdot d$ samples was used. Grey-dotted lines indicate the "optimal" acceptance rates 0.234 or 0.574 and the corresponding step size achieving this acceptance rate. Transparency indicates whether the simulation for the particular step size managed to converge ($\hat{R} < 1.05$). Error regions were computed as 5% and 95% quantiles across all 50 chains.

is observable, whereas s_α^* again achieves suboptimal efficiency. For CSmMALA, s_α^* achieves reasonable efficiency, while expected squared jump distance and effective sample size are slightly offset.

STAT-2 For the bimodal Spiralus without non-identifiabilities, we observe multiple locally optimal expected squared jump distances for most algorithms. Intuitively, this might be explained by considering, that within each mode, different step sizes work well, if the modes differ in shape. For all proposals, that admit such a bimodal expected squared jump distance, one of the modes did not achieve convergence. This implies, that the estimated expected squared jump distance might still be strongly influenced by local effects. In comparison, the effective sample size dos not admit multiple modes.

The Hit & Run sampler shows a particularly interesting effect, where effective sample size as well as expected squared jump distance plateau at a high level. Indeed, for the Hit & Run algorithm increments of the step size are expected to have decreasing influence on the proposal distribution, as the line distribution[5] will converge to a uniform distribution. The presence of a non-identifiable parameter might amplify this effect, as the target distribution becomes a uniform distribution along the corresponding dimension.

STAT-2-ni Similar to the unimodal Spiralus with non-identifibabilities, also the bimodal Spiralus appears to pose a hard sampling problem. However, rounding seems to be a successful approach, as the algorithms which work on a rounded space do achieve convergence this time. Similar to the STAT-2 problem, we observe multiple modes in the expected squared jump distances as well as the plateau effect of the Hit & Run algorithm.

Summary Clearly, the overall question which we want to answer empirically with this experiment is whether the expected squared jump distance is a better tuning target than achieving some pre-defined acceptance rate. As a summary, we present Figure 9.4 and Figure 9.5, where we report the effective sample sizes (or effective sample size per second) achieved by either the step sizes, that either achieve optimal expected squared jump distance or the desired acceptance rate. Again, we mark values, where the Markov chain corresponding to the step size did not converge, using transparency. Efficiency is reported as fraction of the maximal efficiency, which was achieved throughout all step sizes for the particular problem-proposal combination.

In Figure 9.4 we first consider the effective sample size without time costs. The results for the Gaussian target emphasize, what we described before. There, for most sampling algorithms, optimal efficiency is indeed achieved by both, optimizing the expected squared jump distance or achieving the respective pre-defined acceptance rate.

For the STAT-1 problem, the superiority of optimizing the expected squared jump distance is clearly evident, resulting in simulations being approximately twice as efficient. For the CSmMALA, the simulation even becomes four times as efficient.

While for the STAT-1-ni problem only few runs managed to converge, the expected squared jump distance again shows to be the better criterion for improving efficiency of simulations among those. Finally, for the bimodal problem STAT-2 and STAT-2-ni, none of the runs achieving $\alpha = \alpha^*$ did manage to reach convergence. Again, the expected squared jump distance turns out to be a better criterion for choosing efficient step sizes, resulting in more chains, that not only achieved convergence, but also close to optimal efficiency.

[5]Recall that the Hit & Run algorithm first chooses a random direction from the unit hypersphere and then moves along that direction using some univariate distribution along the ray. C.f. Section 5.1.3.

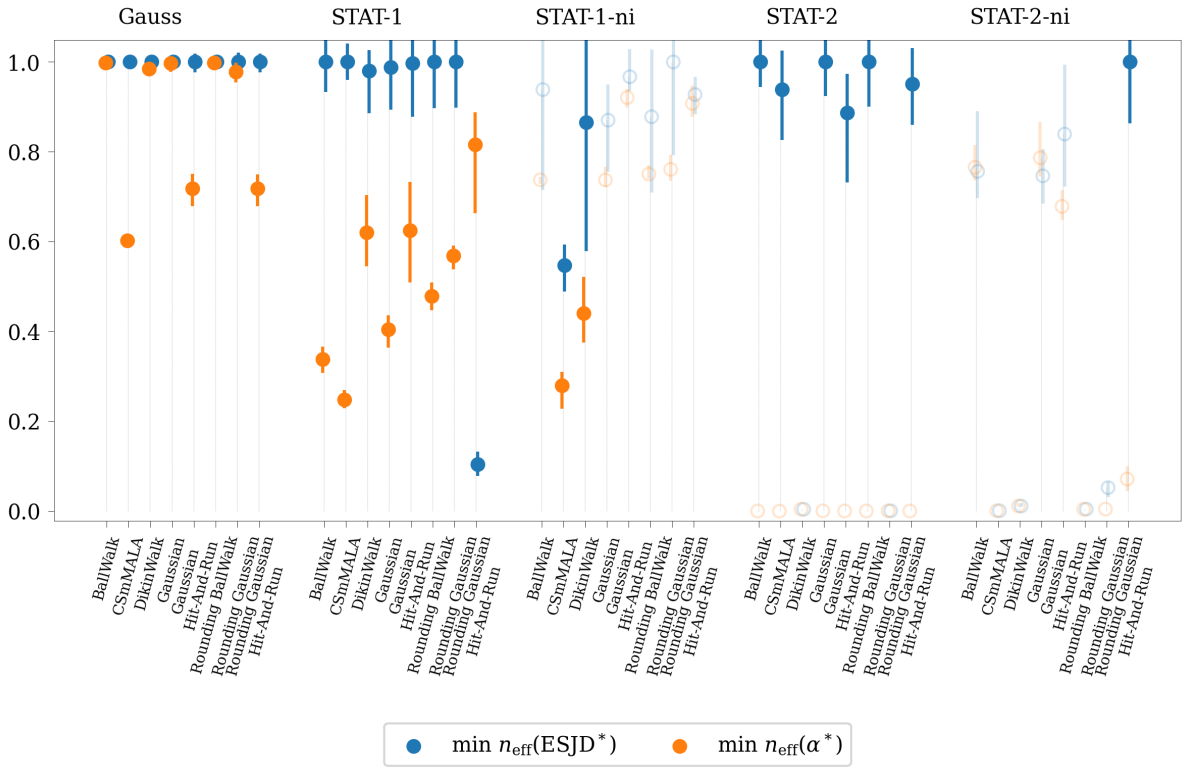


Figure 9.4: Efficiency achieved by the step size optimizing the expected squared jump distance and the step size achieving the desired acceptance rate of either 0.234 or 0.574. Efficiency is normalized against the mean optimal efficiency achieved for every particular problem-proposal combination. Transparency indicates non-convergence of the chain using the particular step size. Error bars were computed as 5% and 95% quantiles across all 50 chains.

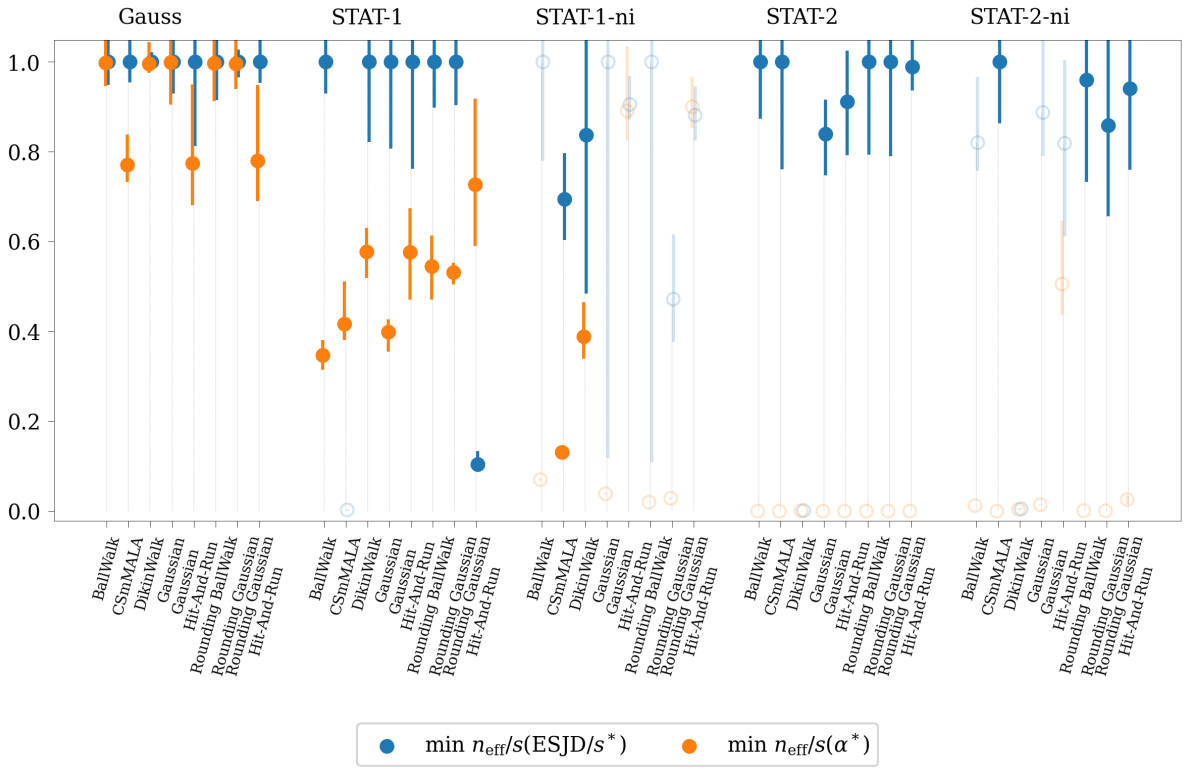


Figure 9.5: Efficiency per second achieved by the step size optimizing the expected squared jump distance and the step size achieving the desired acceptance rate of either 0.234 or 0.574. Efficiency per second is normalized against the mean optimal efficiency per second achieved for every particular problem-proposal combination. Transparency indicates non-convergence of the chain using the particular step size. Error bars were computed as 5% and 95% quantiles across all 50 chains.

A very similar picture is found when including time costs into the notion of efficiency. In Figure 9.5, we consider the achieved efficiency per second. As described in Section 6.2, we conjecture for such an approach to work better, if the evaluation of the target density which we are sampling admits cheap and expensive evaluations. In fact, most of the picture remains the same as in the previous plot, although the error bars of the achieved efficiency seem to increase. This is natural, considering that measurement of time costs on computer systems can be noisy, especially if these are shared within a research group. However, for the STAT-2-ni problem, considering time costs does indeed seem to improve the efficiency of the simulation.

In Section 6.2, we conjectured that the presence of a linearly constrained domain paired with a costly likelihood function might shift the optimal statistical efficiency per second towards larger step sizes. In particular, this was assumed to be a major reason, why acceptance rate tuning might be inferior to expected squared jump distance tuning. However, our results suggest, that time costs are not the major contributor of the observed discrepancy between acceptance and expected squared jump distance tuning, since the phenomenon becomes apparent even when neglecting them. Nevertheless, the assumption of varying time costs can indeed be observed in Figure A.7 in the appendix, although it does not introduce the strong shift in the optimal efficiency, that we expected (cf. Figure A.8 in the appendix).

In conclusion, based on the conducted benchmark, we answer the question of whether the expected squared jump distance yields more efficient step sizes than aiming for particular acceptance rates in the affirmative.

9.2 Thompson Sampling Tuning

Obviously, investing 300 hours to find the optimal step sizes of sampling 2 and 3 dimensional problems is not a reasonable tuning approach. Throughout this section, we present the tuning results achieved using Algorithm 4. We start off by comparing the tuning of the problems from Section 9.1.1 with the brute force results from previous Section 9.1.3. Here, we focus on how well the Thompson sampling's posterior of the tuning target agrees with the respective curves obtained from the brute force experiment and if similar efficiency would be recovered from the tuned step sizes. Afterwards, we consider the tuning of two larger metabolic network models, for which no ground truth is available. We perform tuning w.r.t. to the expected squared jump distance, the sum of expected squared jump distance at 1 and 5 jumps, which we abbreviate as 1,5-ESJD and the acceptance rate. In order to assess performance of the tuning, we sample the problems using the step sizes obtained from tuning until convergence. The resulting samples are used to compute the effective sample size and efficiency among the different tuning approaches will be compared.

9.2.1 Posterior Approximation

For the problems from Section 9.1.1, we considered how well the posteriors of the Thompson sampling tuning (Algorithm 4) approximate the results from the previous, brute force experiment. Further, we again compared the efficiencies achieved depending on the tuning target which was used. Experiments were conducted using $100 \cdot d$ test samples from 10 parallel chains per iteration and a total of 100 iterations. As with the brute force experiment, the log-uniform step size grid was set with endpoints $[1e{-}5, 1e3]$ and a total size of 81 . The number of constant mean iterations N_0 to assess convergence was set to 20. As such, the maximal amount of samples invested into tuning corresponds to the minimal amount of samples we invested in the brute force experiment. Further, tuning targets used were the acceptance rate, using the

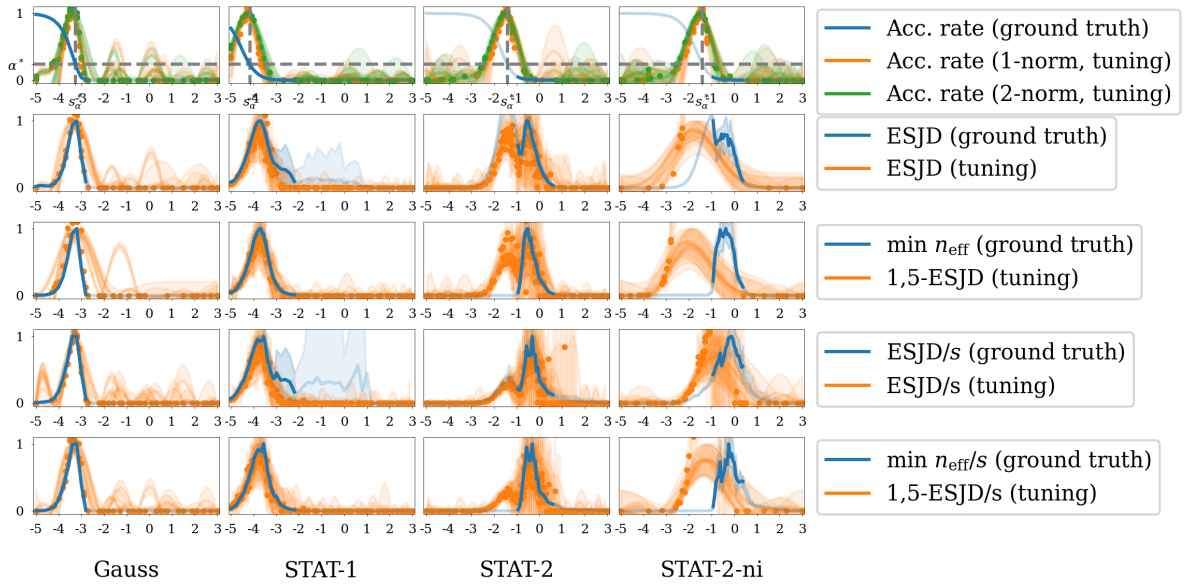


Figure 9.6: Thompson sampling posteriors for the Gaussian random walk compared with their respective ground truth. For acceptance rate tuning, posterior scores are plotted against the acceptance rate and target acceptance rate as well as the corresponding step size have been marked by gray dotted lines. STAT-1-ni model has been omitted here, since no reliable ground truth was found for it in the brute force experiment (cf. Figure 9.3).

score function introduced in Section 7.3.2 with $p \in \{1, 2\}$, expected squared jump distance at one jump and the sum of expected squared jump distances at 1 and 5 jumps (1,5-ESJD) as described in Section 6.1.3. Tuning w.r.t. the expected squared jump distances was performed with and without considering time costs. Again, every setup was repeated using 5 different random seeds. Finally, unlike the previous brute force results, chains were started in the modes of the distribution, which are known in this example. This approach follows, what was discussed in Section 7.3. Since we do not expect our chains to run until convergence and, thus, have no need for diagnosing convergence, the requirement of an overdispersed starting distribution can be dropped. Recall, that this approach is valid independently of the problem as long as the starting points have non-zero density under the target distribution and is, thus, no artificial simplification for the particular problem at hand. Instead, since we are interested in finding step sizes, which work good for "most of the time", starting in the modes avoids possibly strong biases in the tuning target while the chain moves through low probability regions. The experiment was again conducted in parallel on 60 cores of the computer system mentioned at the beginning of this chapter. The elapsed real time for this experiment was about 17 minutes.

In Figure 9.6, the tuning posteriors of the Gaussian sampler on a rounded domain are compared against the corresponding statistics computed in the brute force experiment. As no reliable ground truth was produced for the STAT-1-ni problem using the Gaussian sampler, we omitted it from this summary. The full results for all algorithms, problems and tuning targets can be found in Figures A.9 to A.13 in the appendix. Since the 1,5-ESJD was not part of the brute force experiment, we instead compare the corresponding tuning result with the effective sample size.

As we can see in Figure 9.6, finding the desired acceptance rate can be done reliably using the Thompson sampling algorithm. As expected, both acceptance rate scores using either $p = 1$ or $p = 2$ agree in their optimum. For the ESJD and 1,5-ESJD, the Thompson sampling tuning

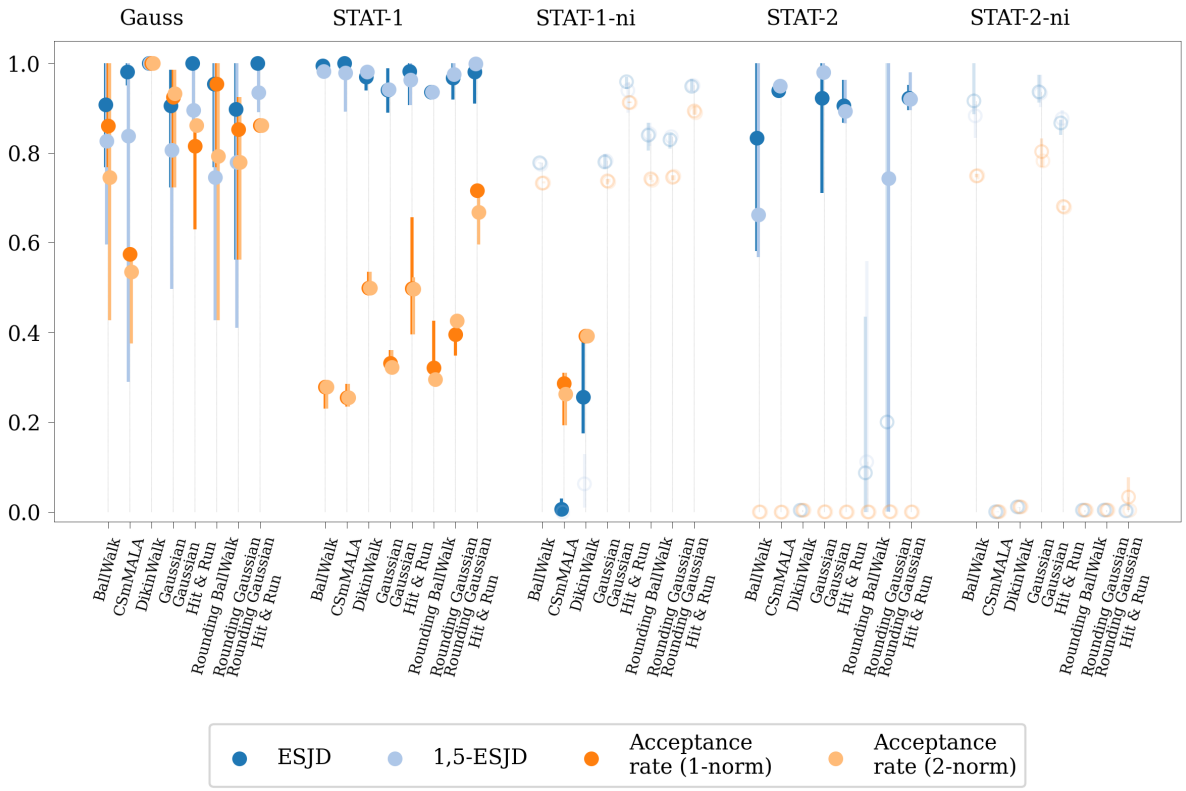


Figure 9.7: Efficiency achieved by Algorithm 4, tuning w.r.t. ESJD, 1,5-ESJD and acceptance rates. Acceptance rate score with 1- or 2-norm are w.r.t. to the choice of p in (7.7). Efficiency per second is normalized against the mean optimal efficiency per second achieved for every particular problem-proposal combination. Transparency indicates non-convergence of the chain using the particular step size. Error bars were computed as 5% and 95% quantiles across all 50 chains.

is well capable to recover much of the shape of the tuning target, if no non-identifiabilities are present. For the STAT-2-ni problem which comprises a non-identifiability, the Thompson sampling posteriors underestimate the correct stepsize. Nevertheless, the rough order of magnitude within which the optimal step sizes resides is detected, especially when considering time costs.

As we conjectured in the previous section, considering the 1,5-ESJD does indeed help to find the correct optimum of the effective sample size. That is, using the 1,5-ESJD, the estimated optimum shifts towards the optimum of the relative efficiency. However, there remains some mismatch, implying that considering higher lags will further improve to detect the correct optimum. Remarkably, considering time costs does an even better job in detecting the correct optimum corresponding to the optimum of the effective sample size.

In order to consider the results over all tuning problems, that were solved throughout this experiment, we summarize the performance of the different tuning targets in a similar fashion as previously for the brute force results. In Figure 9.8 and Figure 9.7, we present the efficiencies achieved by every tuning approach for every problem-proposal combination, with and without respect to time costs respectively. Overall, a similar picture can be drawn as for the brute force result. While there is more deviation between acceptance rate and expected squared jump distance tuning already for the Gaussian problem, tuning w.r.t. the expected squared jump distance achieves higher efficiencies than acceptance rate tuning. The STAT-2 problem in

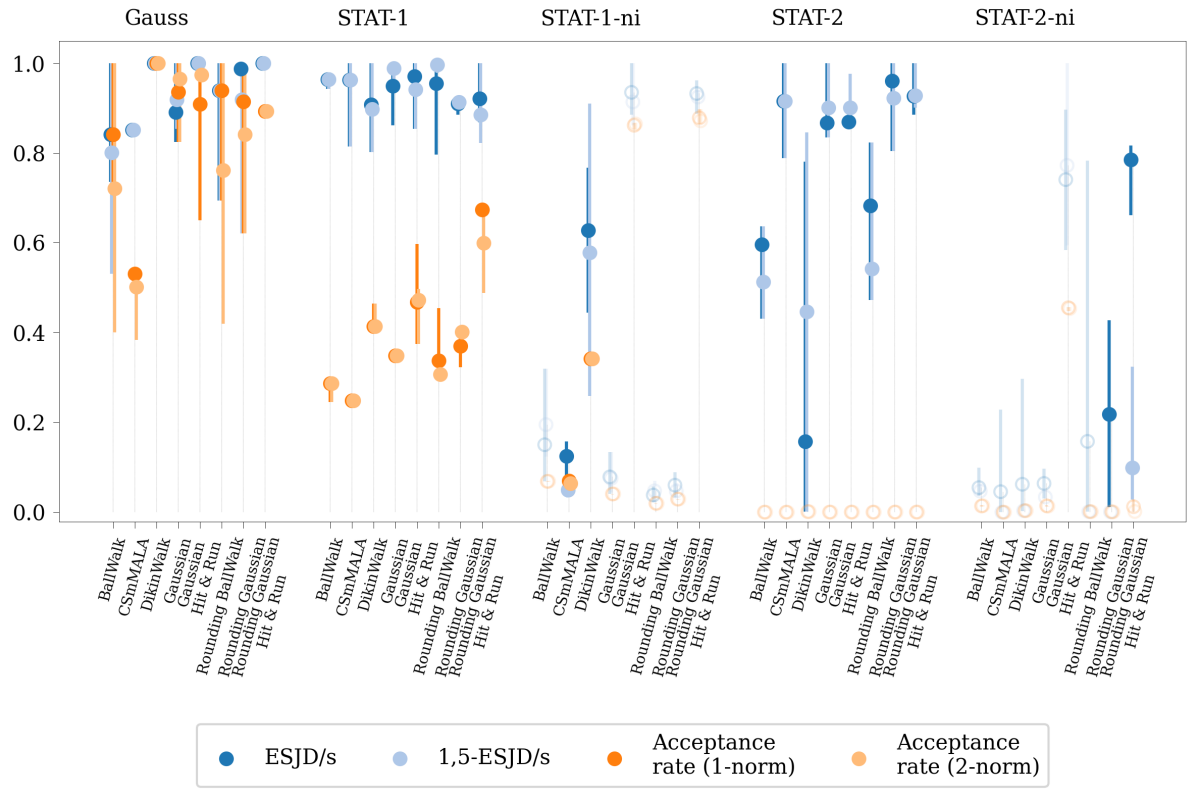


Figure 9.8: Efficiency per second achieved by Algorithm 4, tuning w.r.t. ESJD/s, 1,5-ESJD/s and acceptance rates. Acceptance rate score with 1- or 2-norm are w.r.t. to the choice of p in (7.7). Efficiency per second is normalized against the mean optimal efficiency per second achieved for every particular problem-proposal combination. Transparency indicates non-convergence of the chain using the particular step size. Error bars were computed as 5% and 95% quantiles across all 50 chains.

particular appears to be not solvable, if acceptance rate tuning is used. Although considering time costs seem favorable in Figure 9.6 for finding the optimal step size for the STAT-2 problem, it appears to achieve lower relative performance than the approch without time costs.

Finally, we remark that the hyperparameter choice of the Gaussian process in Algorithm 4 used to approximate the tuning target were indeed suitable for the problems at hand. Although we do observe that the optima of the tuning targets vary in curvature, the peak of the curves do indeed span over a range of 1 to 6 orders of magnitude of the step size. A length scale of 1 is, thus, an appropriate choice for the problems at hand.

Chapter 10

Discussion

As we can see from the results of Section 9.1, for the linearly constrained problems, the expected squared jump distance is a more reliable indicator for statistical efficiency than the acceptance rate, independently of the used proposal algorithm and across all the problems, which were considered. In particular for the bimodal, fully identifiable problem STAT-2, the step size achieving the target acceptance rate of either 0.234 or 0.574 (CSmMALA) was not able to converge within the given amount of samples. These results emphasize the importance of meeting the requirements of the theoretical results regarding optimal efficiency achieved at certain acceptance rates. This contradicts the common practice of tuning chains to acceptance rates between 0.4 and 0.2 regardless of the used proposal algorithm and problem. In particular, we observe that optimal efficiency is often achieved at acceptance rates even smaller than 0.2.

In Section 6.2, we discussed possible influences of the polytope on statistical efficiency. More precisely, we conjectured that large step sizes should be favoured over small step sizes, when considering efficiency per second, since rejections due to the polytope are computationally cheaper than rejections due to a poor likelihood. Although we observe indeed, that optimal efficiency is achieved at step sizes larger than those obtained from acceptance rate tuning, the influence of the polytope remains unclear, as we observe the phenomenon also when neglecting computational costs. Clearly, the key message of favoring expected squared jump distance tuning over acceptance rate tuning remains untouched by this. In fact, this observation raises the question, whether our results would also hold for more general, possibly unconstrained problems. Pinpointing the effect of the polytope on the other hand, could be investigated by well-constructed experimental studies, where one for example inflates the polytope such that its influence on the sampling problem should vanish.

Considering the proposals, which failed to produce a single convergent chain for the STAT-1-ni and STAT-2-ni problem, we may ask, whether the optimal step size could have been found outside the used search interval $[1e{-}5, 1e3]$. Remarkably however, for all convergent problems, we find the optimal step size just in the "valley", where $\hat{R}$ is minimized. Likewise, we observe such a valley also for the non-convergent examples. In fact, $\hat{R}$ seems to grow on both ends of the search interval. This indicates, that indeed, the search interval was well chosen, but not enough samples were invested into sampling the problem.

In Section 9.2, we demonstrated that Algorithm 4 is able to approximate the tuning target reasonably well while using only a fraction of the resources, which were invested to compute the ground truth. The step sizes identified by ESJD and 1,5-ESJD tuning do achieve higher statistical efficiency than those identified by acceptance rate tuning. For some problems efficiency is even improved by 300%, which can be directly translated in requiring only a third of the samples or time. For other problems, convergence could not be achieved using acceptance

rate tuning at all, but only using ESJD or 1,5-ESJD tuning. These results are an encouraging proof of concept, which motivates further improvement and benchmarking of the method on real world problems.

Overall, it turns out, that problems which are harder to sample are also harder to tune. This is clearly expected, since our tuning algorithm relies heavily on MCMC estimation. The Bayesian framework used for performing the optimization allows to use the posterior of an "insufficient" tuning procedure as prior for further tuning. However, deciding when tuning is "sufficient", is a non-trivial question. Using the introduced convergence criterion of Algorithm 4, one could deem tuning to be insufficient, if all tuning iterations where exhausted, implying that the maximum of the posterior mean did not converge. Clearly, such an approach would require human interaction to continue the procedure, as otherwise the tuning might run endlessly. Nevertheless, our algorithm minimizes this interaction to a bare "to continue or not to continue tuning" question.

A major question which remains open is the hyperparametrization of Algorithm 4. Clearly, we expect improved approximation of the tuning target by the Thompson sampling posterior from increasing the number of test samples used. However, reducing the number of test samples per iteration while reducing the overall number of iterations allows Algorithm 4 to perform more informed actions at every iteration. In an extreme case, one could update the posterior after every single new draw taken from the Metropolis chain. This clearly comes at the cost of having to update the posterior more often, but might help to converge within less total Metropolis steps. On the other hand, we could invest all of our resources in a single, uninformed iteration, which would obviously be equivalent to just draw a step size uniformly and run the simulation with it. Considering these two extrema, we presume that there exists an optimal balance between the two parameters.

10.1 Future Work

The promising results achieved with Algorithm 4, encourage further development of the procedure. The vast amount of work done on Gaussian process and Bayesian optimization opens up many possible approaches, of which we will discuss a few.

- In [65], an importance sampling strategy was presented to tune proposal distributions for the Metropolis-Hastings algorithm. In particular, samples across multiple iterations were reused by recomputing their density under a differently parametrized proposal distribution. Intuitively speaking, a given trajectory of a Markov chain was treated *as if* it had been produced by the proposal distribution using the current hyperparameter choice. Importance sampling allows to account for the bias introduced by such an erroneous assumption. This approach allows to update all estimates of the tuning target for every step size from a single batch of new samples. Such an approach could readily be introduced to Algorithm 4 to refine the data collected throughout the tuning procedure. Using this approach, one could hope to obtain improved estimates of the tuning target without increasing the number of test samples, which presumably dominates the costs of tuning.

- Within this work, we have focused on tuning of the step size parameter. Thus, all of our tuning problems were only one-dimensional. Some proposal algorithms, as e.g. CSmMALA [85], admit multiple hyperparameters, thus applying the procedure to higher-dimensional tuning problems is a desirable extension. However, using Algorithm 4 might be infeasible due to the grid, on which the posterior Gaussian process has to be evaluated. Clearly, the size of such a grid grows exponentially in the number of hyperparameters, which we

desire to tune. A possible remedy are random Fourier features [69], which approximate the posterior Gaussian process by the means of randomly drawn Fourier series. At the cost of introducing an approximation error, the randomly drawn functions have the benefit of actually being functions instead of being a finite sample of a function. Thus, these functions are amenable to common optimization techniques and do not require the definition of static grid points, on which the posterior is evaluated. The usage of random Fourier features for multidimensional Bayesian optimization was already proposed in [38].

- The different tuning targets, which we considered throughout this work, are all guaranteed to be non-negative. For the acceptance rate, we even know, that it cannot exceed a value of 1 and it is generally assumed to be non-increasing. However, none of this information is reflected in the Gaussian process, which we use to approximate the targets. One possible solution consists in transforming the Gaussian process into a log-Gaussian process by training the Gaussian process in Algorithm 4 in the log-space of the data. So sampling the Gaussian process will result in a sample in the log-space. Clearly, the resulting log-Gaussian process will take only non-negative values. Another approach consists in considering truncated Gaussian processes on a linearly constrained space. This allows to reflect knowledge on the monotonicity or curvature of the target, as a bound on the derivative of the target becomes a linear constraint on any finite sample thereof. However, it requires one to perform a constrained sampling problem, as no direct approach is available to sample truncated multivariate Gaussian distributions. Also, in a small preliminary survey the covariance matrices obtained from a posterior Gaussian process turned out to be very ill-conditioned, which hampers the sampling problem. The flexibility to include prior information into a truncated Gaussian process is, thus, another reason to work on specialized samplers for truncated problems and making them available to the community in form of high-performance software libraries.

- An issue, which we did not cover in much detail, is the selection of an appropriate search interval. From a Bayesian perspective, defining the search interval is equivalent to defining a prior. In our examples, an appropriate search interval was known thanks to the brute force experiment and many prior iterations. Clearly, the goal of automizing tuning procedures disagrees with performing such an extensive study for every problem encountered in practice. Thus, procedures to detect unsuitable priors is required. For the tuning targets, which we considered in this work, a possible approach consists in testing the bounds of the search interval first. Since small step sizes are commonly assumed to converge against an acceptance rate of 1 and large step sizes against 0, a search interval, where the lower bound is much smaller than 1 and the upper bound is much larger than 0 might be considered to small to include the optimal step size. In particular, such a diagnostic is assumed to also work when tuning w.r.t. the expected squared jump distance, since the expected squared jump distance will converge against 0 for small step sizes and thus high acceptance rate. This assumption is also supported by the experimental results from Section 9.1.3. Clearly, testing the expected squared jump distance at both bounds is not a suitable approach, since a suitable search interval may very well achieve an ESJD of 0 at both endpoints.

- Finally, note that the pre-aggregation introduced in Section 7.1.2 is actually not applicable to the acceptance rate score function (7.7), which we used for tuning the acceptance rate. In particular, due to stochastic noise, step sizes greater (or smaller) than the optimal step size may nevertheless under- or overestimate the true acceptance rate. Therefore, the pre-aggregation step takes averages over values which were differently scaled, meaning that the aggregated value will be subject to some average of the scalings α^* and $1 - \alpha^*$.

The results from Section 9.1.3 however suggest that this does not hamper acceptance rate tuning using Algorithm 4. Nevertheless, an alternative approach, were pre-aggregation becomes valid again, consists in computing the score function on the mean of the posterior Gaussian process, which would then be directly trained on the achieved acceptance rates. Since the acceptance rate is an expectation w.r.t. to the joint distribution $\pi \mathcal{K}$ (cf. (6.3)), pre-aggregation according to Section 7.1.2 does indeed become valid again.

Within this work, we did not benchmark our method against other possible optimization techniques. Partly, this was due to difficulties which were encountered when using publicly available optimization libraries and algorithms, as these algorithms seemed incapable of handling the stochastic noise in the evaluations of the tuning targets. However, methods which are able to deal with such noise exist, a particular one being the covariance matrix adaptation evolution strategy (CMA-ES) [35], which is an evolutionary algorithm, that, loosely spoken, draws candidate solutions from a multivariate normal distribution with covariance estimated from the currently observed populations. Considering Figure 9.3, such a method might be a good candidate, since the tuning targets look indeed somewhat Gaussian. Unlike Algorithm 4, this algorithm has no capabilities of including the uncertainty in the tuning target evaluated from parallel chains, but only requires matrix inversion of constant cost $\mathcal{O}(k^3)$, where k is the number of hyperparameters to tune.

Within this work, we only considered computationally costly problems. Another open questions is under what circumstances tuning with Algorithm 4 is meaningful. In particular, for inexpensive problems, the computational costs of tuning might be dominated by the matrix inversion within Algorithm 4, in which case a less sophisticated method might achieve better performance. Clearly, this question is also tightly coupled to the size our test grid and the overall number of iterations of the tuning algorithm, as these parameters effectively control the cost of the required matrix inversion.

As we discussed in Chapter 6, the problem of providing appropriate hyperparameters should be taken into consideration when developing and benchmarking novel proposal algorithms. We hope, that this work contributes to making Markov chain Monte Carlo simulation not only more feasible to the practitioners, but also allows to gain more insight about the performance of sampling algorithms. A natural approach would be a large scale study, comparing performances of multiple sampling algorithms with and without tuning on a number of real-world problems.

Adaptive sampling algorithms try to overcome the need for hyperparametrization by "learning" appropriate proposal distributions while samping. Clearly, such algorithms are not Markovian anymore, but can be proven to converge against the correct distribution, if the adaptation vanishes in the limit of taking infinitely many samples. An exhaustive comparison of different proposal algorithms should, thus, also compare performance of adaptive algorithms against tuned non-adaptive proposals. In fact, the transition from a priori tuning to adaptive algorithms is smooth, as any form of tuning which stops after a fixed period can be considered as an adaptive sampling algorithm.

Chapter 11

Conclusion

Throughout this thesis, we considered the problem of optimal hyperparametrization of proposal distributions for the Metropolis-Hastings algorithm, the working horse of Bayesian inference. In short, the two main issues of such hyperparameter tuning are the choice of a meaningful tuning target and its optimization using short, preliminary, possibly strongly biased simulations of the Metropolis chain. Within this work, we consider the expected squared jump distance as an alternative tuning target over the acceptance rate tuning, which is the commonly used approach in practice. In particular, the expected squared jump distance directly relates to the autocorrelation of the Metropolis chain and, thus, to the statistical efficiency of the generated sample, independently of the underlying problem and proposal algorithm used. Such a relation cannot be established for the acceptance rate in general. In particular, we considered problems of parameter estimation for dynamical systems with linearly constrained domain. We conjectured, that achieving the text book target acceptance rates will not optimize efficiency for such models due to the complex shape of the domain and the highly non-linear, potentially multimodal target distribution. The metabolic network models, which are used in Bayesian ^{13}C metabolic flux analysis and admit such linearly constrained domains combined with an arbitrarily shape target distribution, were used as example problems.

In an extensive, experimental study, we reaffirmed that for linearly constrained problems with arbitrary target distributions, optimizing the expected squared jump distance yields more efficient sampling than achieving particular, pre-defined acceptance rates, for which no theoretical guarantees of optimality exist. In particular, some of the problems treated turned out to be infeasible when choosing step sizes according to the acceptance rate.

Estimating efficiency metrics like the expected squared jump distance to find suitable hyperparameters is a hen-egg problem. Using short, preliminary simulations of the Metropolis chain for estimation introduce the risk of large estimation errors. However, investing too many resources into tuning may waste much of the time, one actually tries to save by using an efficient hyperparametrization. An iterative strategy, refining estimates, while allocating resources such that more promising hyperparameters are refined more often, is an exploration-exploitation trade-off. The Thompson sampling algorithm, commonly used in Bayesian optimization, is an efficient technique, to solve such problems. We, therefore, proposed a variant of the Thompson sampling algorithm to tackle the tuning of proposal scaling, which we refer to as step size tuning. Comparing a short, 17 minute tuning run using our algorithm against the results of a 300 hour brute force study, the Thompson sampling tuning algorithm was able to very well approximate the brute force results. In particular, the step sizes found by our algorithm achieved comparable statistical efficiencies to those, which were considered optimizing the respective tuning target in the brute force experiment.

Bibliography

[1] S. Agrawal and N. Goyal, *Analysis of thompson sampling for the multi-armed bandit problem*, in Proceedings of the 25th Annual Conference on Learning Theory, S. Mannor, N. Srebro, and R. C. Williamson, eds., vol. 23 of Proceedings of Machine Learning Research, Edinburgh Scotland, 25–27 Jun 2012, JMLR Workshop and Conference Proceedings, pp. 39.1–39.26.

[2] ——, *Further optimal regret bounds for thompson sampling*, 2012.

[3] M. Ala'raj, M. Majdalawieh, and N. Nizamuddin, *Modeling and forecasting of covid-19 using a hybrid dynamic model based on seird with arima corrections*, Infectious Disease Modelling, 6 (2021), pp. 98–111.

[4] C. Andrieu and J. Thoms, *A tutorial on adaptive mcmc*, Statistics and Computing, 18 (2008), pp. 343–373.

[5] S. Aslam, S. Ul Islam, K. A. Ganai, and E. R. Patro, *Impact of Biotechnology on the Climate Change*, Springer International Publishing, Cham, 2020, pp. 109–120.

[6] E. U. Azeloglu and R. Iyengar, *Signaling networks: information flow, computation, and decision making*, Cold Spring Harbor perspectives in biology, 7 (2015), pp. a005934–a005934. 25833842[pmid].

[7] O. Berger-Tal, J. Nathan, E. Meron, and D. Saltz, *The exploration-exploitation dilemma: A multidisciplinary framework*, PLOS ONE, 9 (2014), pp. 1–8.

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

[9] A. Boneh and A. Golan, *Constraints' redundancy and feasible region boundedness by random feasible point generator (rfpg)*, (1979).

[10] K. Borah, M. Beyss, Y. Xu, J. Barber, C. Costa, J. Newcombe, A. Theorell, M. J. Bailey, D. J. Beste, J. McFadden, and K. Nöh, *Bayesian multi-model-based 13c15n-metabolic flux analysis quantifies carbon-nitrogen metabolism in mycobacteria*, bioRxiv, (2022).

[11] E. Bradford, A. M. Schweidtmann, and A. Lapkin, *Efficient multiobjective optimization employing gaussian processes, spectral sampling and a genetic algorithm*, Journal of Global Optimization, 71 (2018), pp. 407–438.

[12] J. A. Brofos, M. Gabrié, M. A. Brubaker, and R. R. Lederman, *Adaptation of the independent metropolis-hastings sampler with normalizing flow proposals*, 2021.

[13] S. P. Brooks and A. Gelman, *General methods for monitoring convergence of iterative simulations*, Journal of Computational and Graphical Statistics, 7 (1998), pp. 434–455.

[14] C. J. P. Bélisle, H. E. Romeijn, and R. L. Smith, *Hit-and-run algorithms for generating multivariate distributions*, Mathematics of Operations Research, 18 (1993), pp. 255–266.

[15] A. Chalkis, V. Fisikopoulos, E. Tsigaridas, and H. Zafeiropoulos, *Geometric algorithms for sampling the flux space of metabolic networks*, in 37th International Symposium on Computational Geometry (SoCG 2021), K. Buchin and E. Colin de Verdière, eds., vol. 189 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2021, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 21:1–21:16.

[16] O. Chapelle and L. Li, *An empirical evaluation of thompson sampling*, in Advances in Neural Information Processing Systems, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds., vol. 24, Curran Associates, Inc., 2011.

[17] O.-T. Chis, J. R. Banga, and E. Balsa-Canto, *Structural identifiability of systems biology models: A critical comparison of methods*, PLOS ONE, 6 (2011), pp. 1–16.

[18] J. A. Christen and C. Fox, *Markov chain monte carlo using an approximation*, Journal of Computational and Graphical Statistics, 14 (2005), pp. 795–810.

[19] P. R. Conrad, Y. M. Marzouk, N. S. Pillai, and A. Smith, *Accelerating asymptotically exact mcmc for computationally intensive models via local approximations*, Journal of the American Statistical Association, 111 (2016), pp. 1591–1607.

[20] P. G. Constantine, C. Kent, and T. Bui-Thanh, *Accelerating markov chain monte carlo with active subspaces*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2779–A2805.

[21] S. Crouch, N. C. Hong, S. Hettrick, M. Jackson, A. Pawlik, S. Sufi, L. Carr, D. De Roure, C. Goble, and M. Parsons, *The software sustainability institute: Changing research software attitudes and practices*, Computing in Science Engineering, 15 (2013), pp. 74–80.

[22] G. de Anda-Jáuregui and E. Hernández-Lemus, *Computational oncology in the multi-omics era: State of the art*, Frontiers in Oncology, 10 (2020).

[23] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, *Hybrid monte carlo*, Physics Letters B, 195 (1987), pp. 216–222.

[24] D. K. Duvenaud, *Automatic model construction with Gaussian processes*, PhD thesis, University of Cambridge, 2014.

[25] Y. R. Efendiev, T. Y. Hou, and W. Luo, *Preconditioning markov chain monte carlo simulations using coarse-scale models*, SIAM J. Sci. Comput., 28 (2006), pp. 776–803.

[26] S. P. Ellner and J. Guckenheimer, *Dynamic Models in Biology*, Princeton University Press, 2006.

[27] A. Gelman, *Bayesian data analysis*, CRC Press, Boca Raton, 2014.

[28] A. Gelman and D. B. Rubin, *Inference from iterative simulation using multiple sequences*, Statistical Science, 7 (1992), pp. 457–472.

[29] C. J. Geyer, *Practical markov chain monte carlo*, Statistical Science, 7 (1992), pp. 473–483.

[30] ——, *Introduction to Markov Chain Monte Carlo*, CRC Press, 2011, ch. chapter1.

[31] M. Girolami and B. Calderhead, *Riemann manifold langevin and hamiltonian monte carlo methods*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73 (2011), pp. 123–214.

[32] K. Guseva, S. Darcy, E. Simon, L. V. Alteio, A. Montesinos-Navarro, and C. Kaiser, *From diversity to complexity: Microbial networks in soils*, Soil Biology and Biochemistry, 169 (2022), p. 108604.

[33] H. Haario, M. Laine, A. Mira, and E. Saksman, *Dram: Efficient adaptive mcmc*, Statistics and Computing, 16 (2006), pp. 339–354.

[34] H. Haario, E. Saksman, and J. Tamminen, *An adaptive metropolis algorithm*, Bernoulli, 7 (2001), pp. 223–242.

[35] N. Hansen, *The cma evolution strategy: A tutorial*, 2016.

[36] H. S. Haraldsdóttir, B. Cousins, I. Thiele, R. M. Fleming, and S. Vempala, *Chrr: coordinate hit-and-run with rounding for uniform sampling of constraint-based models*, Bioinformatics, 33 (2017), pp. 1741–1743.

[37] W. K. Hastings, *Monte carlo sampling methods using markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.

[38] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, *Predictive entropy search for efficient global optimization of black-box functions*, 2014.

[39] M. D. Hoffman and A. Gelman, *The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo*, J. Mach. Learn. Res., 15 (2014), pp. 1593–1623.

[40] F. Hutter, H. H. Hoos, and K. Leyton-Brown, *Sequential model-based optimization for general algorithm configuration*, in Proceedings of the 5th International Conference on Learning and Intelligent Optimization, LION'05, Berlin, Heidelberg, 2011, Springer-Verlag, pp. 507–523.

[41] R. J. Hyndman, *Computing and graphing highest density regions*, The American Statistician, 50 (1996), pp. 120–126.

[42] J. F. Jadebeck, A. Theorell, S. Leweke, and K. Nöh, *Hops: high-performance library for (non-) uniform sampling of convex-constrained models*, Bioinformatics, 37 (2021), pp. 1776–1777.

[43] W. Jakob, J. Rhinelander, and D. Moldovan, *pybind11 – seamless operability between c++11 and python*, 2017. https://github.com/pybind/pybind11.

[44] R. Kannan, L. M. Lovász, and M. Simonovits, *Random walks and an o*(n^5) volume algorithm for convex bodies*, Random Struct. Algorithms, 11 (1997), pp. 1–50.

[45] R. Kannan and H. Narayanan, *Random walks on polytopes and an affine interior point method for linear programming*, Mathematics of Operations Research, 37 (2012), pp. 1–20.

[46] D. E. Kaufman and R. L. Smith, *Direction choice for accelerated convergence in hit-and-run sampling*, Operations Research, 46 (1998), pp. 84–95.

[47] R. Kumar, C. Carroll, A. Hartikainen, and O. Martin, *Arviz a unified library for exploratory analysis of bayesian models in python*, Journal of Open Source Software, 4 (2019), p. 1143.

[48] A. Lazarus, D. Husmeier, and T. Papamarkou, *Multiphase mcmc sampling for parameter inference in nonlinear ordinary differential equations*, in Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, A. Storkey and F. Perez-Cruz, eds., vol. 84 of Proceedings of Machine Learning Research, PMLR, 09–11 Apr 2018, pp. 1252–1260.

[49] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*, American Mathematical Society, 2006.

[50] S. Leweke, *Markov chain monte carlo methods for bayesian inference and viable flux volume estimation of 13c metabolic network models*, Master's thesis, Universität Siegen, 2013. Universität Siegen, Masterarbeit, 2013.

[51] L. Lovász and S. Vempala, *Hit-and-run from a corner*, SIAM Journal on Computing, 35 (2006), pp. 985–1005.

[52] D. Luengo, L. Martino, M. Bugallo, V. Elvira, and S. Särkkä, *A survey of monte carlo methods for parameter estimation*, EURASIP Journal on Advances in Signal Processing, 2020 (2020).

[53] G. M. Martin, D. T. Frazier, and C. P. Robert, *Computing bayes: Bayesian computation from 1763 to the 21st century*, 2020.

[54] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall PTR, USA, 2003.

[55] W. Megchelenbrink, M. Huynen, and E. Marchiori, *optgpsampler: an improved tool for uniformly sampling the solution-space of genome-scale metabolic networks*, PloS one, 9 (2014), pp. e86587–e86587. 24551039[pmid].

[56] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, The Journal of Chemical Physics, 21 (1953), pp. 1087–1092.

[57] S. Meyn and R. Tweedie, *Markov Chains and Stochastic Stability*, Springer-Verlag, London, 1993.

[58] C. Müller, H. Diedam, T. Mrziglod, and A. Schuppert, *A neural network assisted metropolis adjusted langevin algorithm*, Monte Carlo Methods and Applications, 26 (2020), pp. 93–111.

[59] B. Nelson, M. J. Payne, and E. B. Ford, *Differential evolution mcmc: An algorithm for bayesian parameter estimation of multi-planet systems*, in American Astronomical Society Meeting Abstracts, vol. 217 of American Astronomical Society Meeting Abstracts, January 2011, p. 343.20.

[60] J. R. Norris, *Markov chains.*, Cambridge series in statistical and probabilistic mathematics, Cambridge University Press, 1998.

[61] M. E. O'Neill, *Pcg: A family of simple fast space-efficient statistically good algorithms for random number generation*, Tech. Rep. HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA, September 2014.

[62] J. D. Orth, I. Thiele, and B. Palsson, *What is flux balance analysis?*, Nature Biotechnology, 28 (2010), pp. 245–248.

[63] E. Özcan and T. Çakir, *Reconstructed metabolic network models predict flux-level metabolic reprogramming in glioblastoma*, Frontiers in Neuroscience, 10 (2016).

[64] M. D. Parno and Y. M. Marzouk, *Transport map accelerated markov chain monte carlo*, SIAM/ASA Journal on Uncertainty Quantification, 6 (2018), pp. 645–682.

[65] C. Pasarica and A. Gelman, *Adaptively scaling the metropolis algorithm using expected squared jumped distance*, tech. rep., 2010.

[66] B. Peherstorfer and Y. Marzouk, *A transport-based multifidelity preconditioner for markov chain monte carlo*, 2018.

[67] O. Perez-Garcia, G. Lear, and N. Singhal, *Metabolic network modeling of microbial interactions in natural and engineered environmental systems*, Frontiers in Microbiology, 7 (2016).

[68] S. R. Proulx, D. E. L. Promislow, and P. C. Phillips, *Network thinking in ecology and evolution*, Trends in Ecology & Evolution, 20 (2005), pp. 345–353. SPECIAL ISSUE: BUMPER BOOK REVIEW.

[69] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, in Advances in Neural Information Processing Systems, J. Platt, D. Koller, Y. Singer, and S. Roweis, eds., vol. 20, Curran Associates, Inc., 2008.

[70] C. E. Rasmussen, *Gaussian processes for machine learning*, MIT Press, 2006.

[71] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, Springer New York, 2004.

[72] G. O. Roberts, A. Gelman, and W. R. Gilks, *Weak convergence and optimal scaling of random walk metropolis algorithms*, The Annals of Applied Probability, 7 (1997), pp. 110–120.

[73] G. O. Roberts and J. S. Rosenthal, *Optimal scaling for various metropolis-hastings algorithms*, Statistical Science, 16 (2001), pp. 351–367.

[74] ——, *General state space markov chains and mcmc algorithms*, Probability Surveys, 1 (2004), pp. 20–71.

[75] G. O. Roberts and R. L. Tweedie, *Exponential convergence of langevin distributions and their discrete approximations*, Bernoulli, 2 (1996), pp. 341–363.

[76] D. Russo, B. V. Roy, A. Kazerouni, I. Osband, and Z. Wen, *A tutorial on thompson sampling*, 2020.

[77] P. Sanders, *Algorithm Engineering – An Attempt at a Definition*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 321–340.

[78] J. Schmölder and M. Kaspereit, *A modular framework for the modelling and optimization of advanced chromatographic processes*, Processes, 8 (2020), p. 65.

[79] S. L. Scott, *A modern bayesian look at the multi-armed bandit*, Applied Stochastic Models in Business and Industry, 26 (2010), pp. 639–658.

[80] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, *Taking the human out of the loop: A review of bayesian optimization*, Proceedings of the IEEE, 104 (2016), pp. 148–175.

[81] J. O. Smith, *Mathematics of the Discrete Fourier Transform (DFT)*, `http://ccrma.stanford.edu/~jos/mdft/`, accessed (28.04.2022). online book, 2007 edition.

[82] R. L. Smith, *Monte carlo techniques for generating random feasible solutions to mathematical programs*, (1980).

[83] ——, *Efficient monte carlo procedures for generating points uniformly distributed over bounded regions*, Operations Research, 32 (1984), pp. 1296–1308.

[84] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, in Advances in Neural Information Processing Systems, F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, eds., vol. 25, Curran Associates, Inc., 2012.

[85] A. Theorell, *Bayesian methods for data-driven characterization of cells*, PhD thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, 2019. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University 2020; Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2019.

[86] A. Theorell, J. F. Jadebeck, K. Nöh, and J. Stelling, *Polyround: polytope rounding for random sampling in metabolic networks*, Bioinformatics, (2021).

[87] A. Theorell, S. Leweke, W. Wiechert, and K. Nöh, *To be certain about the uncertainty: Bayesian statistics for 13c metabolic flux analysis*, Biotechnology and Bioengineering, 114 (2017), pp. 2668–2684.

[88] A. Theorell and K. Nöh, *Reversible jump mcmc for multi-model inference in metabolic flux analysis*, Bioinformatics, 36 (2019), pp. 232–240.

[89] A. Theorell and J. Stelling, *Metabolic networks, microbial consortia, and analogies to smart grids*, Proceedings of the IEEE, (2022), pp. 1–16.

[90] I. Thiele and B. Palsson, *A protocol for generating a high-quality genome-scale metabolic reconstruction*, Nature Protocols, 5 (2010), pp. 93–121.

[91] I. Thiele, S. Sahoo, A. Heinken, J. Hertel, L. Heirendt, M. K. Aurich, and R. M. Fleming, *Personalized whole-body models integrate metabolism, physiology, and the gut microbiome*, Molecular Systems Biology, 16 (2020), p. e8982.

[92] R. N. Thompson, *Epidemiological models are important tools for guiding covid-19 interventions*, BMC Medicine, 18 (2020), p. 152.

[93] W. R. Thompson, *On the likelihood that one unknown probability exceeds another in view of the evidence of two samples*, Biometrika, 25 (1933), pp. 285–294.

[94] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, *Auto-weka: Combined selection and hyperparameter optimization of classification algorithms*, in Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, New York, NY, USA, 2013, Association for Computing Machinery, pp. 847–855.

[95] M. Trias, A. Vecchio, and J. Veitch, *Delayed rejection schemes for efficient markov-chain monte-carlo sampling of multimodal distributions*, 2009.

[96] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon, *Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020*, in Proceedings of the NeurIPS 2020 Competition and Demonstration Track, H. J. Escalante and K. Hofmann, eds., vol. 133 of Proceedings of Machine Learning Research, PMLR, 06–12 Dec 2021, pp. 3–26.

[97] A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner, *Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of mcmc*, Bayesian Analysis, (2020).

[98] S. R. Weiskopf, Z. V. Harmáčková, C. G. Johnson, M. C. Londoño Murcia, B. W. Miller, B. J. E. Myers, L. Pereira, M. I. Arce-Plata, J. L. Blanchard, S. Ferrier, E. A. Fulton, M. Harfoot, F. Isbell, J. A. Johnson, A. S. Mori, E. Weng, and I. M. D. Rosa, *Increasing the uptake of ecological model results in policy decisions to improve biodiversity outcomes*, Environmental Modelling & Software, 149 (2022), p. 105318.

[99] M. Weitzel, K. Nöh, T. Dalman, S. Niedenführ, B. Stute, and W. Wiechert, *13cflux2—high-performance software suite for 13c-metabolic flux analysis*, Bioinformatics, 29 (2012), pp. 143–145.

[100] M. Weitzel, W. Wiechert, and K. Nöh, *The topology of metabolic isotope labeling networks*, BMC Bioinformatics, 8 (2007), p. 315.

[101] W. Wiechert, M. Möllney, N. Isermann, M. Wurzel, and A. A. de Graaf, *Bidirectional reaction steps in metabolic networks: Iii. explicit solution and analysis of isotopomer labeling systems*, Biotechnology and Bioengineering, 66 (1999), pp. 69–85.

[102] W. Wiechert, S. Niedenführ, and K. Nöh, *A primer to13c metabolic flux analysis*, in Fundamental Bioengineering, Wiley-VCH Verlag GmbH & Co. KGaA, oct 2015, pp. 97–142.

[103] W. Wiechert and K. Nöh, *Isotopically non-stationary metabolic flux analysis: complex yet highly informative*, Current Opinion in Biotechnology, 24 (2013), pp. 979–986. Chemical biotechnology • Pharmaceutical biotechnology.

[104] W. Wiechert and M. Wurzel, *Metabolic isotopomer labeling systems: Part i: global dynamic behavior*, Mathematical Biosciences, 169 (2001), pp. 173–205.

[105] J. Yang, G. O. Roberts, and J. S. Rosenthal, *Optimal scaling of random-walk metropolis algorithms on general target distributions*, Stochastic Processes and their Applications, 130 (2020), pp. 6094–6132.

[106] J. Yang and J. S. Rosenthal, *Automatically tuned general-purpose mcmc via new adaptive diagnostics*, Computational Statistics, 32 (2016), pp. 315–348.

[107] Y. Zhang and L. Gao, *On numerical solution of the maximum volume ellipsoid problem*, SIAM Journal on Optimization, 14 (2003), pp. 53–76.

List of Algorithms

List of Figures

Miscellaneous Figures & Tables

Chapter 9 – Experimental Setup & Results

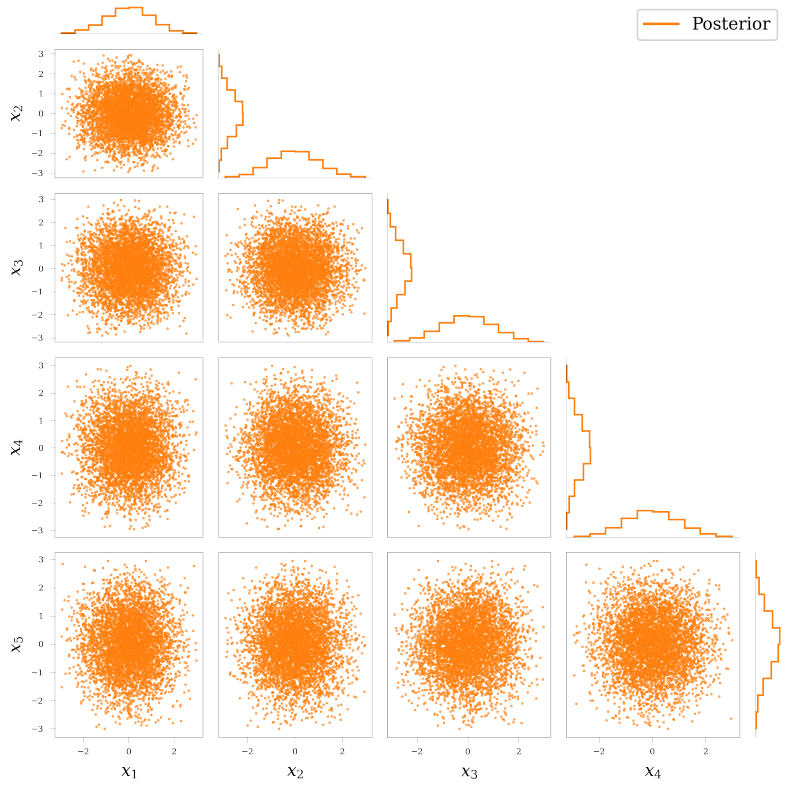


Figure A.1: Corner plot of the posterior distribution of the Gauss model in Section 9.1.1

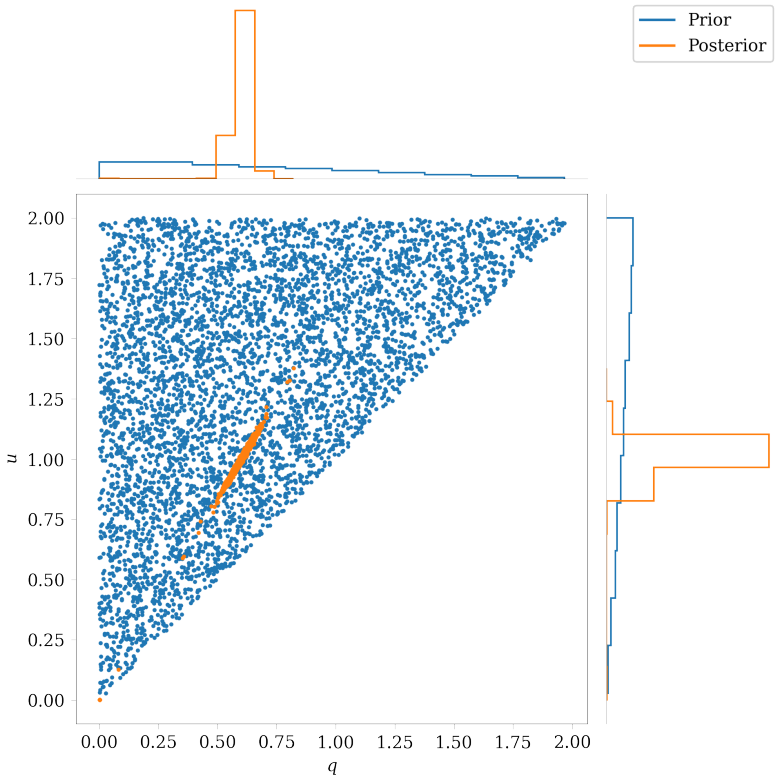


Figure A.2: Corner plot of the posterior distribution of the STAT-1 model in Section 9.1.1

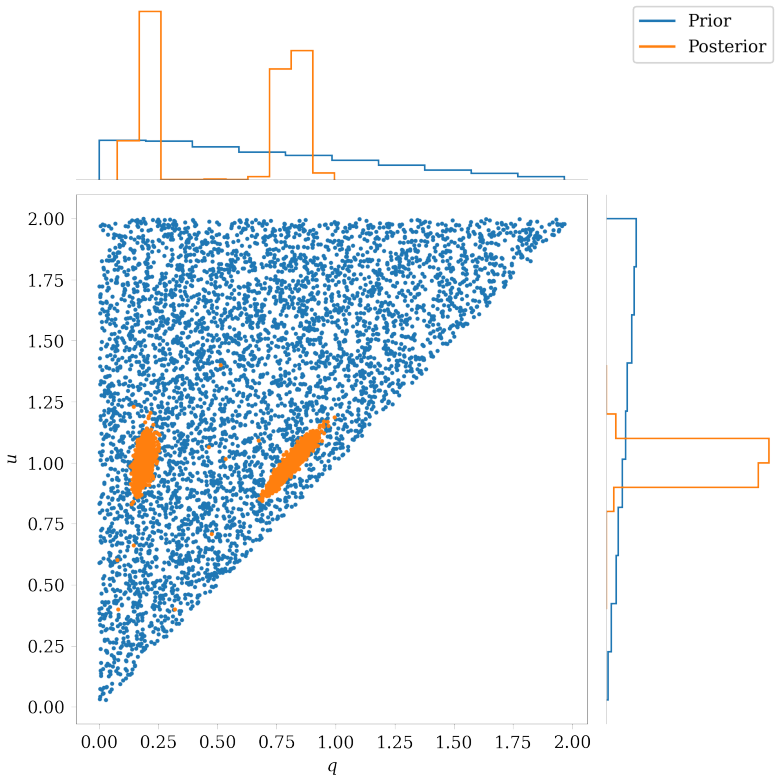


Figure A.3: Corner plot of the posterior distribution of the STAT-2 model in Section 9.1.1

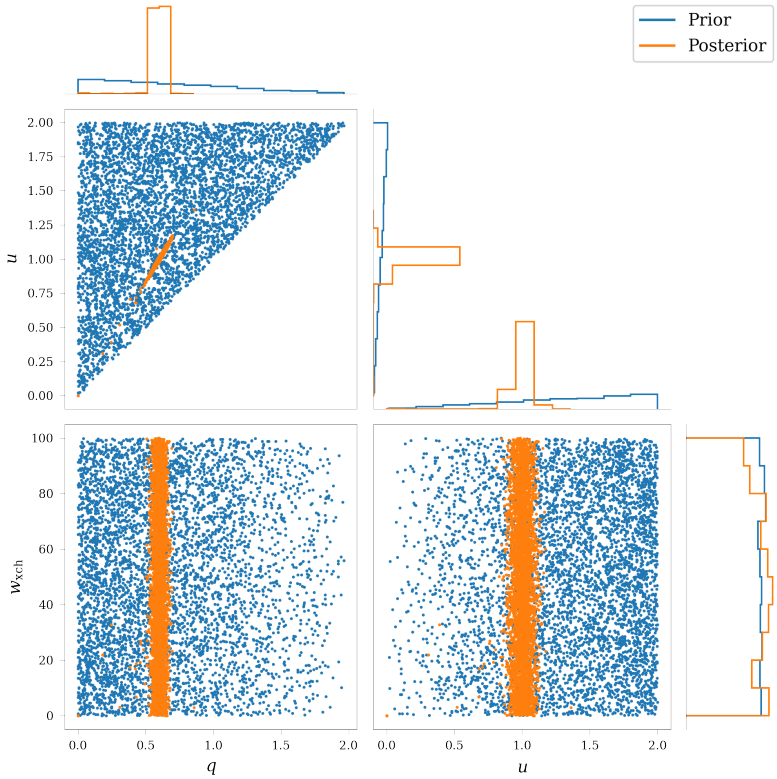


Figure A.4: Corner plot of the posterior distribution of the STAT-1-ni model in Section 9.1.1

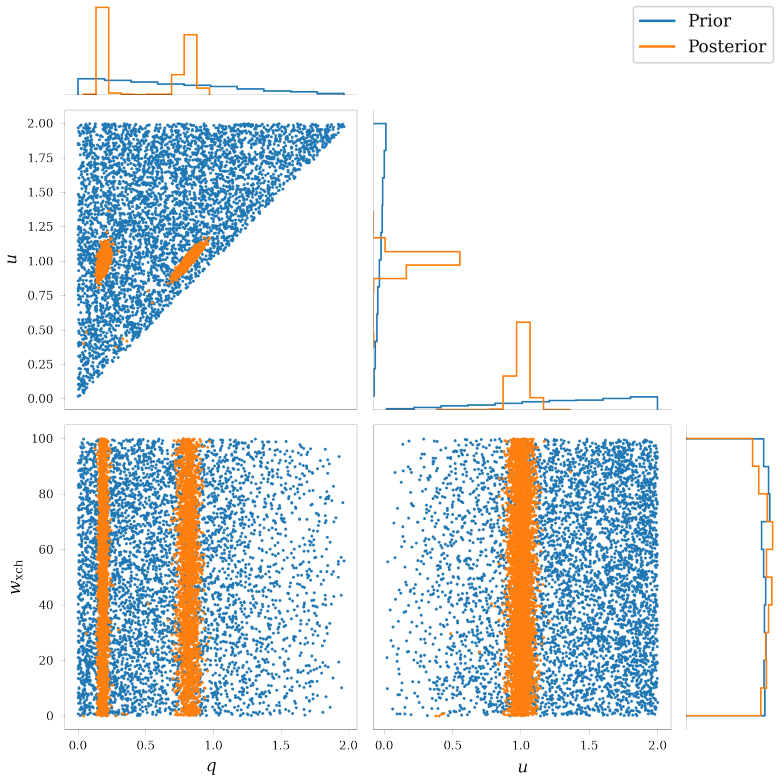


Figure A.5: Corner plot of the posterior distribution of the STAT-2-ni model in Section 9.1.1

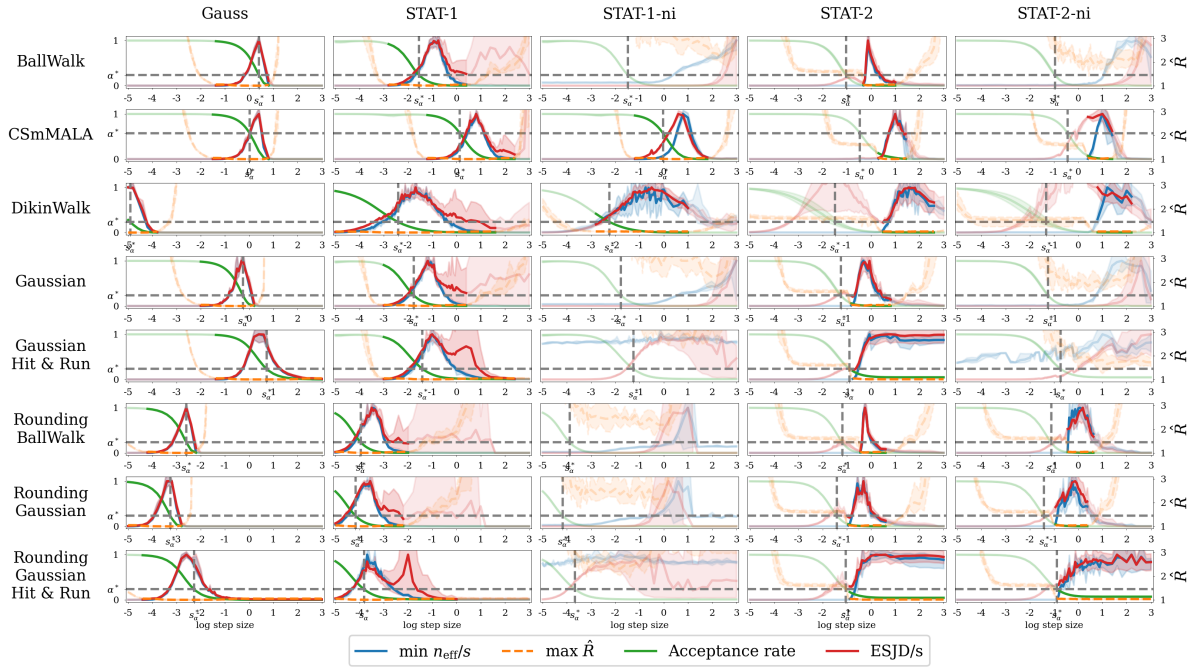


Figure A.6: Results of the brute force experiment described in Section 9.1 w.r.t. to time costs.

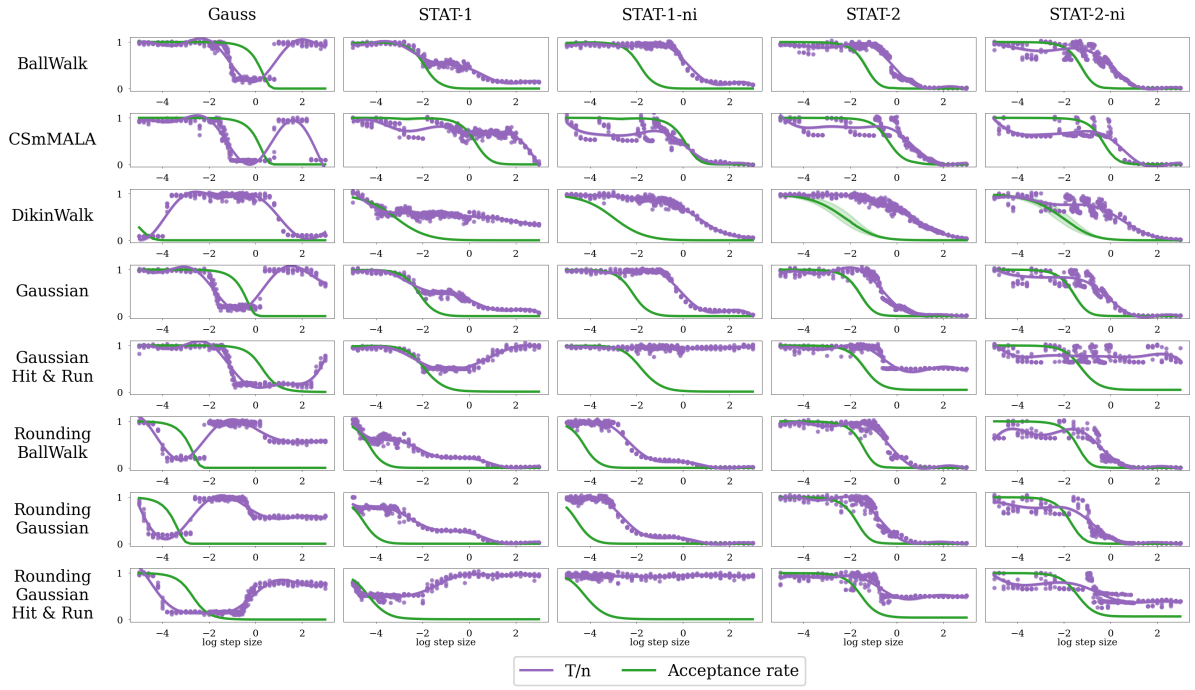


Figure A.7: Time per sample in comparison to the acceptance rate measured during the brute force experiment described in Section 9.1.

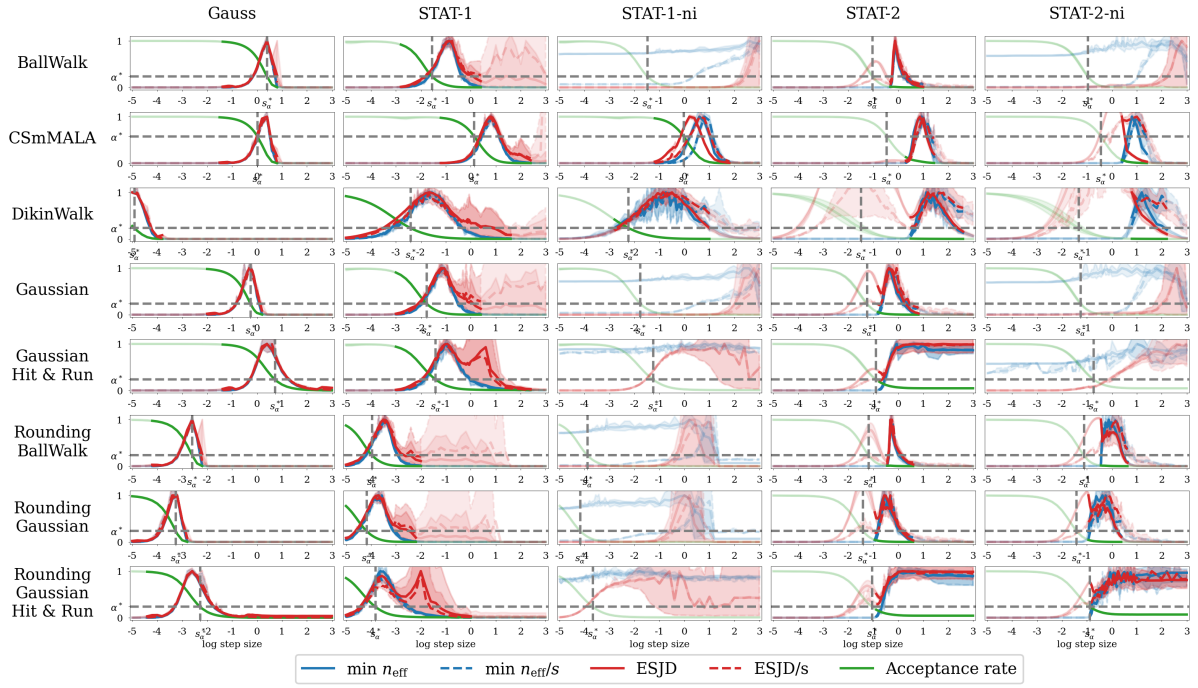


Figure A.8: Comparison of expected squared jump distance and effective sample size with and without considering time costs.

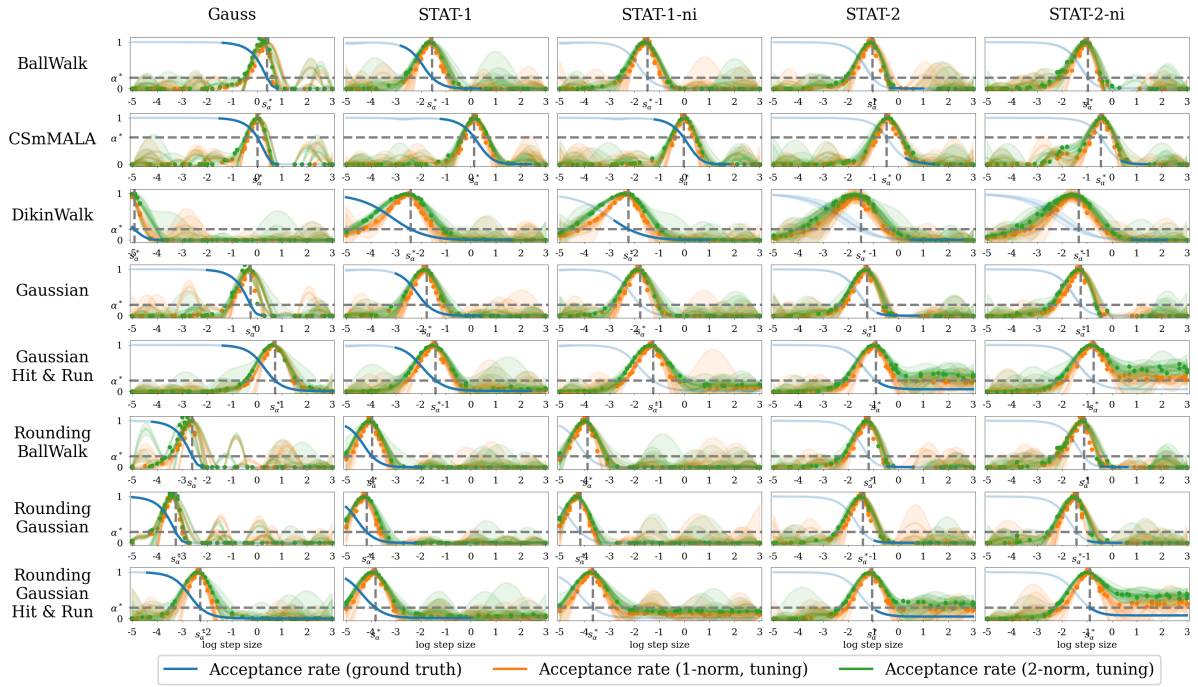


Figure A.9: Thompson sampling posterior approximations of the acceptance rate score functions used for tuning.

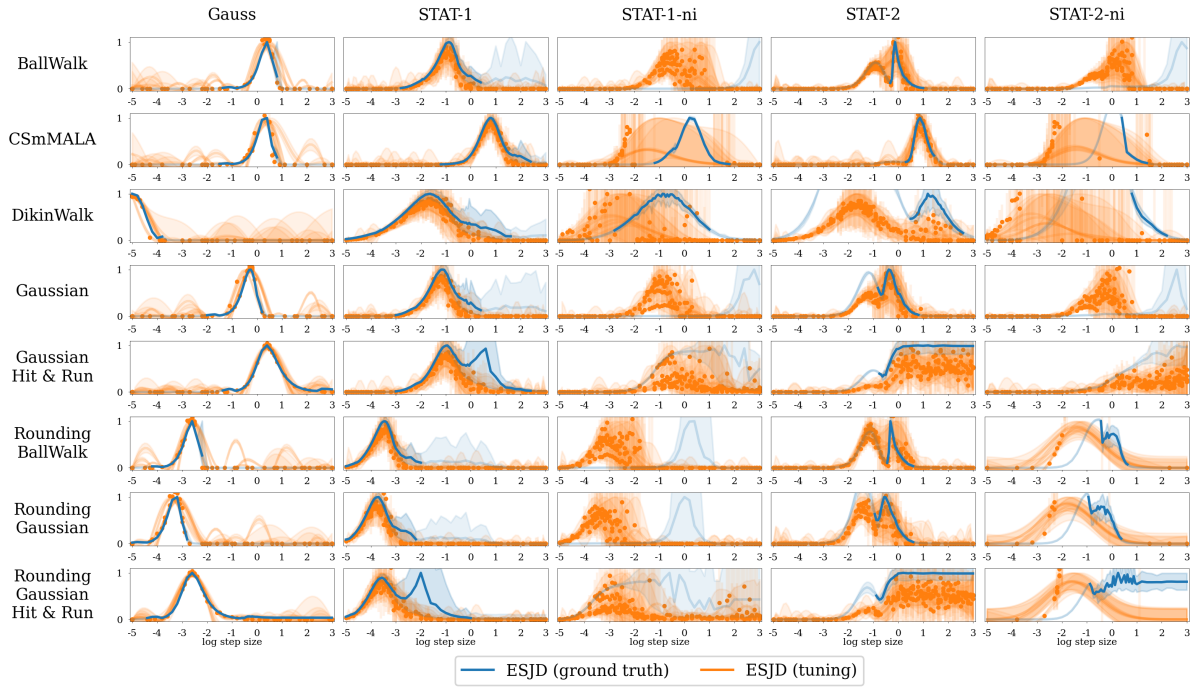


Figure A.10: Thompson sampling posterior approximations of the ESJD.

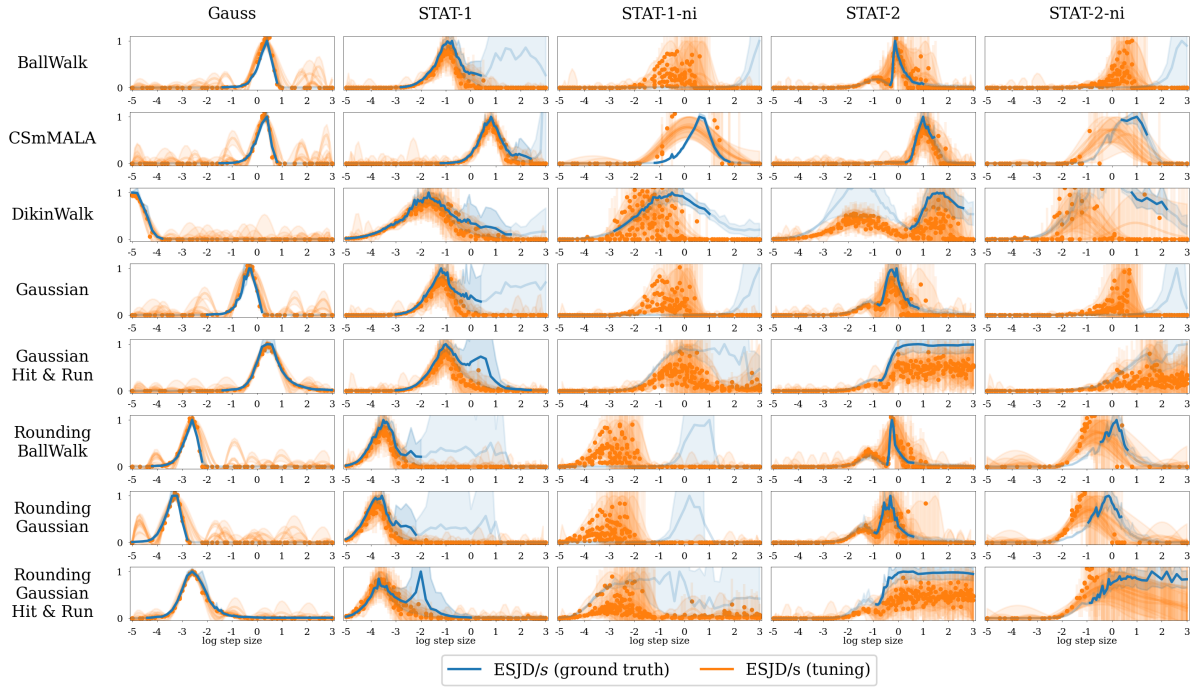


Figure A.11: Thompson sampling posterior approximations of the ESJD/s.

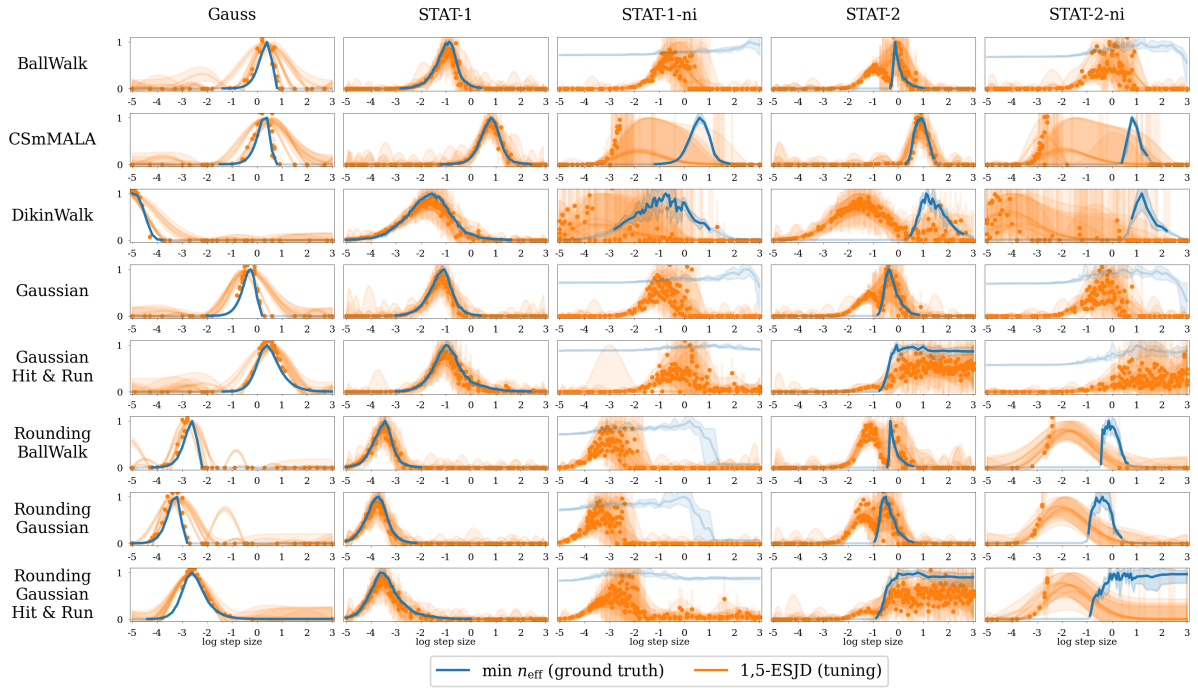


Figure A.12: Thompson sampling posterior approximations of the 1,5-ESJD.

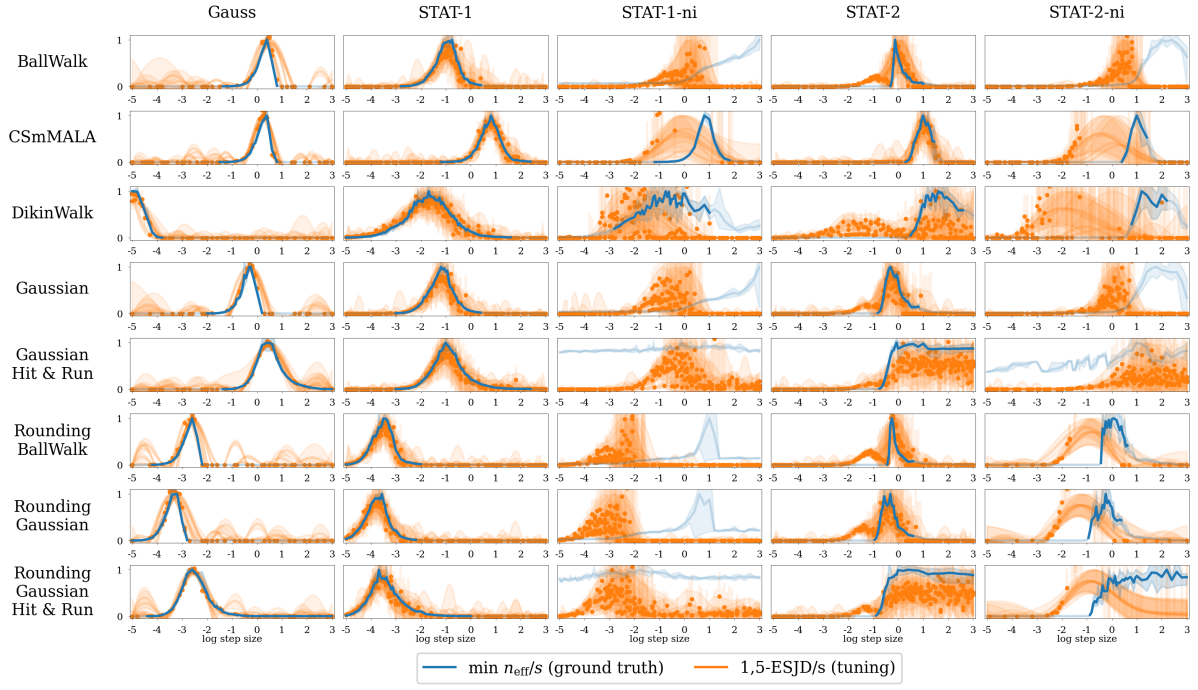


Figure A.13: Thompson sampling posterior approximations of the 1,5-ESJD/s.

Miscellaneous Proofs

Chapter 4 – Markov Chain Monte Carlo Methods

Lemma A.1 (Lemma 4.2.1)**.** *Let π be a bounded probability measure and let* $\operatorname{supp}\pi := \{x \in \mathbb{R}^d : \pi(x) > 0\}$ *be path-connected. Further, let the proposal distribution $q(x, \cdot)$ be s.t. $\exists\, \epsilon > 0$ for every $x \in \operatorname{supp}\pi$ and*

$$q(x, y) > 0, \quad \forall\, y \in \mathcal{B}(x, \epsilon),$$

where $\mathcal{B}(x, \epsilon) := \{y \in \mathcal{S}^d : \| x - y \|_2 < \epsilon\}$. The Markov chain with proposal distribution q and transition kernel as in (4.5) is π-irreducible and aperiodic.

Proof. **(i)** *Aperiodicity.* Assume the chain is periodic, then there exist at least two disjoint subsets $\mathcal{S}_1, \mathcal{S}_2$ s.t., assuming $x \in \mathcal{S}_1$ w.l.o.g., $\mathcal{K}(x, \mathcal{S}_2) = 1$ and $\mathcal{K}(x, \mathcal{S}_1) = 0$. Clearly, $x \in \mathcal{B}(x, \epsilon)$ and so by assumption $q(x, x) > 0$. Therefore, for any measurable set A containing x, $\mathcal{K}(x, A) > 0$, so especially also $\mathcal{K}(x, \mathcal{S}_1) > 0$, but this contradicts the initial assumption of periodicity of the chain.

(ii) *Irreducibility.* Let $y \in A$ for some $A \subseteq \operatorname{supp}\pi$, then by path-connectedness, there exists a continuous map $p : [0, 1] \to \operatorname{supp}\pi$ with $p(0) = x$ and $p(1) = y$ for any $x \in \operatorname{supp}\pi$. Thus, by continuity of p, there exists $x_{i+1} \in \{p(a) : a \in [0, 1]\}$ for every $i = 1, \dots, n$, s.t. $x_{i+1} \in \mathcal{B}(x_i, \epsilon_i)$ and $x_i \in \mathcal{B}(x_{i+1}, \epsilon_{i+1})$, where we set $x_1 = x$ and $x_n = y$. Given the sequence $(x_1, \dots, x_n)$, we now have that

$$\prod_{i=1}^{n-1} q(x_i, x_{i+1})\, \alpha(x_i, x_{i+1}) > 0,$$

since $\alpha(x_i, x_{i+1}) > 0$, if $\pi(x_{i+1})\, q(x_{i+1}, x_i) > 0$, which holds by construction of the sequence for every $i = 1, \dots, n-1$. Therefore

$$\begin{aligned}\mathcal{K}^n(x, A) = \mathcal{K}^n(x_1, A) &= \prod_{i=1}^{n-1} \int q(x_i, x_{i+1})\, \alpha(x_i, x_{i+1})\, \mathrm{d}x_{i+1} \\ &= \int \dots \int \prod_{i=1}^{n-1} q(x_i, x_{i+1})\, \alpha(x_i, x_{i+1})\, \mathrm{d}x_2 \dots \mathrm{d}x_n > 0\end{aligned}$$

for any and, thus, all $A \subseteq \operatorname{supp}\pi$. □

Chapter 6 – Tuning Proposal Distributions

Lemma A.2. *Let $f : \mathbb{R} \to \mathbb{R}$ be Lebesgue-integrable, then for a.e. $x \in \mathbb{R}$ we have*

$$\lim_{n \to \infty} f(x + n) \to 0,$$

where $n \in \mathbb{N}$.

Proof. Let $k \in \mathbb{Z}$, then by monotone convergence and translation invariance of the Lebesgue-measure it holds that

$$\begin{aligned}\int_k^{k+1} \sum_{n\geq 1} |f(x+n)|\lambda(\mathrm{d}x) &= \sum_{n\geq 1} \int_k^{k+1} |f(x+n)|\lambda(\mathrm{d}x) \\ &= \sum_{n\geq 1} \int_{k+n}^{k+n+1} |f(x)|\lambda(\mathrm{d}x) \\ &= \int_{k+1}^{\infty} |f(x)|\lambda(\mathrm{d}x) \qquad \leq \underbrace{\int_{\mathbb{R}} |f(x)|\lambda(\mathrm{d}x) < \infty}_{\text{by assumption}}\end{aligned}$$

and since any integrable function is bounded almost everywhere,

$$\sum_{n\geq 1} |f(x+n)| < \infty \quad \text{for a.e. } x \in [k, k+1].$$

Now it follows from absolute convergence that

$$\lim_{n\to\infty} |f(x+n)| = 0 \quad \text{for a.e. } x \in [k, k+1],$$

but by k being arbitrary it also holds for a.e. $x \in \mathbb{R}$. □