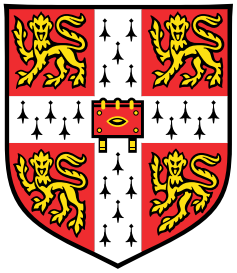
Advanced Bayesian Monte Carlo Methods for Inference and Control



Torben Sell

Department of Pure Mathematics and Mathematical Statistics
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Darwin College

February 2021

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text.

Parts of this thesis are published or under review for publication. The work presented in Chapter 3 is my contribution to the publication Goldman et al. (2020), of which Jacob V. Goldman and I are joint first authors and equally contributed to the publication, with advice from Sumeetpal S. Singh. Chapter 4 is my own work and pending submission for publication in future. Chapters 5 and 6 are currently under review for publication as Sell and Singh (2020). The work presented in this thesis are my contributions to these collaborations (unless stated otherwise in the text) and the work in this thesis benefitted from the guidance of my supervisor Sumeetpal S. Singh.

Torben Sell
February 2021

Abstract

Advanced Bayesian Monte Carlo Methods for Inference and Control

Torben Sell

Monte Carlo methods are are an ubiquitous tool in modern statistics. Under the Bayesian paradigm, they are used for estimating otherwise intractable integrals arising when integrating a function h with respect to a posterior distribution π. This thesis discusses several aspects of such Monte Carlo methods.

The first discussion evolves around the problem of sampling from only almost everywhere differentiable distributions, a class of distributions which includes all log-concave posteriors. A new sampling method based on a second-order diffusion process is proposed, new theoretical results are proved, and extensive numerical illustrations elucidate the benefits and weaknesses of various methods applicable in these settings.

In high-dimensional settings, one can exploit local structures of inverse problems to parallelise computations. This will be explored in both fully localisable problems, and problems where conditional independence of variables given some others holds only approximately. This thesis proposes two algorithms using parallelisation techniques, and shows their empirical performance on two localisable imaging problems.

Another problem arises when defining function space priors over high-dimensional domains. The commonly used Karhunen-Loève priors suffer from bad dimensional scaling: they require an orthogonal basis of the function space, which can often be obtained as a product of one-dimensional basis functions. This leads to the number of parameters growing exponentially in the dimension d of the function domain. The trace-class neural network prior, proposed in this thesis, scales more favourably in the dimension of the function's domain. This prior is a Bayesian neural network prior, where each weight and bias has an independent Gaussian prior, but with a key difference to existing Bayesian neural network priors: the

variances decrease in the width of the network, such that the variances form a summable sequence and the infinite width limit neural network is well defined. As is shown in this thesis, the resulting posterior of the unknown function is amenable to sampling using Hilbert space Markov chain Monte Carlo methods. These sampling methods are favoured because they are stable under mesh-refinement, in the sense that the acceptance probability does not shrink to 0 as more parameters are introduced to better approximate the well-defined infinite limit. Both numerical illustrations and theoretical results show that these priors are competitive and have distinct advantages over other function space priors.

These different function space priors are then used in stochastic control. To this end, a suitable likelihood for continuous value functions in a Bayesian approach to reinforcement learning is defined. This thesis proves that it can be used in conjunction with both the classical Karhunen-Loève prior and the proposed trace-class neural network prior. Numerical examples compare the resulting posteriors, and illustrate the new prior's performance and dimension robustness.

Für Mama und Papa

Acknowledgements

Thanks firstly to every person who is fighting for a more equal world, it is your work that makes me believe in humanity and motivated me to finish this dissertation. At the same time, I humbly note my own privileges, and promise to use them to push for a more egalitarian society.

I would like to thank my supervisor, Sumeet, for his support and for teaching me many aspects of research and work in academia. Further thanks to Jacob for a very enjoyable collaboration, and to Alix, Sam, and Sam for inspiring reading group meetings. Thanks to the CCIMI and the DPMMS, especially to Carola, Rachel, Tessa, and Josh, for creating the vibrant research atmosphere which I enjoyed so much. It is the CCIMI's generous studentship that allowed me to write a dissertation that some describe as 'very applied' and others as 'pretty theoretic'. I like to think that it contains the best of both worlds, and I hope the examiners agree. Thanks to my extended office mates; were it not for Maggie, Tamara, Tom, Oliver, Derek, and Ben and the various coffee breaks with them, I might have never finished this thesis. I hope that you will keep my rants about capitalism in mind; may your profits always be smaller than your contributions to society.

Moving away from mathematical influencers, my housemates Oli, Nico, Stef, Cynthia, Lucía, and Soda were a continuous and invaluable source of support. Thank you for all the late night chats, amazing dinners, dancing on our grand piano, home-made pizza, for helping me to improve as a person, and for bringing me tea when I cried in my room. Thanks to Anna and Lenn for giving me a warm home in Münster to write up my thesis, and to Kris for sharing his wisdom that some rosemary always has to end up on the floor.

Darwin College provided me with a family away from home, and including all names would make these acknowledgments longer than Das Kapital. I would like to name and thank a few regardless. Thanks to Tim Milner for being a kind, approachable, and helpful tutor, to John Dix for being a supportive, generous, and mindful person, and to Derek Scott for being the greatest porter one can imagine. To the DCSA Executive Committee 2018/19 for their

fantastic work which overall made my role as President enjoyable, to the choir for musical and other stimulation, and to the bar for fostering intellectual conversations over a delicious Becks. To Chris who reliably was up for a pint, to Jezebel and her heart of gold, to Jenaid for showing me the meaning of the sun, and to Chiara for asking me so many questions about Markov chains that I, in the end, believed I knew what I was talking about. Thanks to the Nomnomnoms for cooking delightful dinners seasoned with wholesomeness. Luís, thank you for your honest advice in both DCSA and friendship matters. Jyothi, thank you for being available whenever I needed emotional support and for lovely walks back home after Darbar. Lotti, thank you for engaging discussions on how to increase integers properly and for being a fantastic and reliable friend. Aleix, you always cheered up everyone around you, I wish we could've spent more time together.

To the CUCrC sailors for accepting me in the team, and for distracting me from the tiresome PhD life, though I could've lived without learning the difference between a Hawking and a Tomahawk. To the CUYC sailors, for making me feel part of the team, and for joining in with passionate Bella Ciaos. Thanks to Jenny for our rebellious trip to London and for her support regardless of which university's stash I was wearing, to Miha for introducing me to the Little Prince, and to Ronja for waking and fostering my love of plants.

Many people outside Cambridge kept me sane and grounded, and I am grateful for their ongoing support despite the physical distance separating us. Thanks to Larissa for our decade old friendship, to Konstantin for always drinking half of my wine, and to Rylan for the best transatlantic friendship I can imagine. The Wasserwacht Lübeck and its affiliates have been one of my strongest support groups since I joined their training in 2005; a big thank you goes out to all of you, and to Eva, Hajo, and Patrick in particular. Moritz, I will never forget your crazy actions, though I wish you never went on that one final drive. Thanks to Runa, for making me the proudest godfather, and to Jonte, Claudia, and Carsten for making me feel part of their family whenever I visit them. To Bärbel and Okke for being an invaluable support through sad times, and for deep conversations over whiskey and wine.

I am immensely grateful for having the sea as a home wherever in the world I am, and I hope that the sound of seagulls and waves will keep me company for many years to come. I love my family for supporting me in every way possible. Kerstin, thank you for being an inspiration and idol, now as much as you were ten years ago. Henrik, thank you for being such a non-judgemental and loving brother, and always being there for me. You two are my sister and brother as much as you are best friends to me. Silke and Georg, I couldn't find the words to thank you appropriately, so let the thesis dedication show you my infinite love.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Roman Symbols

$\mathcal{A}$	action space
$a(\cdot,\cdot)$	acceptance probability
$b_i^{(l)}$	bias of a NN, superscript indicates the layer, subscript the nodes it influences
B_t	Brownian motion
C	covariance operator
C	covariance matrix
$\mathcal{D}$	differential operator
d	dimension of X
$Exp(\lambda)$	exponential distribution with rate λ
$\mathfrak{F}$	flip operator
f	probability density function
$Geom$	Geometric distribution
$\mathcal{H}$	Hilbert space
h	test function which is to be integrated
$H(x)$	blur operator (confusion with the Hamiltonian is not possible given the contexts)
$H(x,v)$	Hamiltonian of the augmented target, $H(x,v) = -\log f(x,v)$

$\hat{I}$ Monte Carlo estimate of an integral I

$\mathbb{I}\{A\}$ indicator function of the set A

I integral of interest

$K(x, dy)$ Markov transition kernel

$k(x, y)$ pdf of a Markov transition kernel K

ℓ log of the likelihood function

ℓ^2 space of square summable sequences

$\mathcal{L}$ likelihood function

L two-dimensional Laplacian operator on a square grid, a matrix

$L^2(\mathcal{X}, \mathbb{R})$ space of square integrable functions from $\mathcal{X} \subseteq \mathbb{R}^d$ to $\mathbb{R}$

$\mathbb{N}$ integers

$\mathcal{N}$ normal distribution

$N^{(l)}$ number of nodes in layer l of a NN

$p(\cdot)$ probability of an event (it will be clear from the context under which distribution)

$\mathbb{Q}$ rational numbers

$Q(x, dy)$ Markov transition kernel (in proposal distributions)

$\mathbb{R}$ real numbers

$\mathfrak{R}$ reflection operator

$\mathcal{U}(A)$ uniform distribution over a set A

$U(z)$ potential of a distribution, $U(z) = -\log f(z)$

U^λ MYE smoothed potential

v value function

$w_{i,j}^{(l)}$ weight of a NN, superscript indicates the layer, subscripts the nodes it connects

$\mathcal{X}$	domain of X
X	random variable
x	realisation of a random variable
X_A	all variables in block A of X
X_{-A}	all variables of X but those in block A
X_i	i-th coordinate of X
X_{-i}	all coordinates of X but the i-th
$\mathcal{Y}$	domain of observations
y	observation, data
Z	normalisation constant

Greek Symbols

δ	in algorithms: step size
δ_x	Dirac measure, located at x
λ	tightness parameter of the MYE
μ	generic distribution
μ	mean of distribution (in Chapter 3 only)
μ	policy (in Chapter 6 only)
Ω	precision matrix, inverse of the covariance matrix
π	posterior distribution
π_0	prior distribution
Φ	cdf of a standard normal distribution
ϕ	pdf of a standard normal distribution
φ	basis function (in the context of KL expansions), activation function (in the context of NNs)

π^λ MYE smoothed posterior

ρ rate function for an inhomogeneous Poisson process

σ^2 variance, especially for normal distributions

τ event time

θ parameters of a NN, $\theta = (w, b)$, i.e. the collection of all weights and biases

Acronyms / Abbreviations

BNN Bayesian neural network

BPS bouncy-particle sampler

BRL Bayesian reinforcement learning

CDF cumulative distribution function

DA delayed-acceptance MCMC algorithm

GP Gaussian process

iPP inhomogeneous Poisson process

MAP maximum a posteriori estimate

MCMC Markov chain Monte Carlo

MH Metropolis-Hastings

MwG Metropolis-within-Gibbs

MYE Moreau-Yosida Envelope

MY-UULA Moreau-Yosida unadjusted Langevin algorithm

MY-UULA Moreau-Yosida unadjusted underdamped Langevin algorithm

NN neural network

paraDA parallelised delayed acceptance MCMC algorithm

paraMwG parallelised (and blocked) Metropolis-within-Gibbs

pCNL preconditioned Crank-Nicolson Langevin algorithm

pCN preconditioned Crank-Nicolson algorithm

PDE partial differential equation

pdf probability density function

PDMP piece-wise deterministic Markov process

pMALA proximal Metropolis-adjusted Langevin algorithm

RWMH random walk Metropolis-Hastings algorithm

SDE stochastic differential equation

SK-ROCK stochastic second kind orthogonal Runge-Kutta-Chebyshev method

TV total variation

ZZS zig-zag sampler

Chapter 1

Introduction

The mathematical analysis of data, interpreting it, and the extraction of useful information from it, are key aspects of modern day statistics. The ultimate goal is often to infer something about an unknown quantity, and further to quantify the (un)certainty of the inference objective.

Throughout this thesis the object of interest is some unknown random variable X defined on a domain $\mathcal{X}$, and the goal is to infer information about this variable from some data $y \in \mathcal{Y}$. Mathematically, this can be modelled using probability distributions, which are from now on referred to simply as distributions.

This thesis uses the Bayesian framework, where a *prior distribution* π_0 summarises prior belief about X, and the *likelihood function* $\mathcal{L}(y|x)$ re-weights the prior proportional to 'how much more likely' possible realisations x of X are in comparison to others, jointly giving rise to the *posterior distribution*. This is justified by Bayes' Theorem, named after Reverend Thomas Bayes who developed an early version thereof (Bayes, 1763) which was later generalised by Pierre Laplace (Hald, 1998). The posterior measure is defined by

$$\pi(dx) = \frac{1}{Z}\mathcal{L}(y|x)\pi_0(dx), \tag{1.1}$$

where $Z = \int \mathcal{L}(y|x)\pi_0(dx)$ is the normalising constant, ensuring that π is indeed a probability distribution. Notably, the posterior is only well defined if $0 < Z < \infty$, a condition that is non-trivial especially for infinite-dimensional spaces as we will discuss in Chapter 5. Intuitively, this condition is satisfied whenever the likelihood A) is integrable with respect to the prior, giving a finite normalisation constant, and B) does not rule out all sets with positive prior mass, such that $Z > 0$.

Inference, as well as decision making and quantification of uncertainty, is based on this posterior distribution, and often requires integration of functions with respect to these posterior measures,

$$I = \int h(x)\pi(dx). \tag{1.2}$$

To give an explicit example, let X be the highest water level during a flood in a coastal town. The data y consists of observations obtained throughout the year and in previous flooding situations. The function $h(x)$ corresponds to the economic cost of the damage which arises if the water level reaches x. The integral (1.2) then describes the expected cost of a coming flood. This impacts policy making: a local government can decide to improve dams and other countermeasures, thereby reducing the probability of extensive flooding and thus also the overall expected cost.

Generally, integrals of the form (1.2) are not analytically tractable, and need to be approximated using Monte Carlo estimates. Given a set of samples $\{x_i\}_{i=1}^N$ from the posterior distribution, a Monte Carlo estimate to (1.2) is given by

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} h(x_i). \tag{1.3}$$

This estimator converges to the true integral $\hat{I}_N \to I$ with the Monte Carlo error decreasing as $O(N^{-1/2})$ independent of the dimensionality of the space $\mathcal{X}$: if $h(x) = 1$ for all x, then

$$\hat{I}_N - I \to \mathcal{N}\left(0, \frac{\sigma_\pi^2}{N}\right),$$

where σ_π^2 is the variance of the distribution π, which is assumed to be finite throughout this thesis. This error decay stands in contrast to the error decay of numerical integration schemes discretising the domain which usually depends on the dimension of the space $\mathcal{X}$; this is one of the main reasons for the popularity of Monte Carlo methods.

The intractable integration problem (1.2) has thus been reduced to a sampling problem to obtain estimates (1.3).

Using pseudo-random numbers, sampling from simple distributions such as uniform or Gaussian distributions is possible. For more complicated distributions, some kind of importance sampling (Au and Beck, 1999; Glynn and Iglehart, 1989; Neal, 2001) might

be an option, but for the vast majority of sampling problems, Markov chain Monte Carlo methods are the go-to tool. These methods, discussed extensively in Chapter 2, rely on the construction of a Markov chain to obtain correlated samples, but achieve the same error rate of $O(N^{-1/2})$ if the constructed Markov chain is ergodic (Hastings, 1970).

While these computational sampling methods were introduced to the Bayesian statistics community in the middle of the 20^{th} century, they were hardly used by Bayesian statisticians prior to the 1990s due to the limited availability of computing power. Today, with faster and more powerful computers standing in every research institution, this bottleneck has been widened and Markov chain Monte Carlo methods are used daily by applied scientists around the world. An active research community continuously improves the performance of the methods and deepens the theoretical understandings thereof. An overview of the past, present, and future of Bayesian computational methods is given by Green et al. (2015).

This thesis builds on these recently developed works. We propose new methodology, provide further theoretical insights, and improve empirical understanding through multiple numerical experiments; all of this with a focus on sampling problems on high- and infinite-dimensional domains, where advanced Monte Carlo methods are essential for meaningful inference.

The thesis is organised as follows: Chapter 2 gives an overview of several Markov chain Monte Carlo algorithms. The first chapter with new contributions to the statistical community is Chapter 3, which discusses sampling from non-differentiable distributions, this chapter is part of the publication Goldman et al. (2020). Chapter 4 defines fully and approximately localisable Bayesian inverse problems, and proposes methodology for such problems which exploits the localised structure through parallel computing. Thereafter, we move to infinite-dimensional spaces, and discuss existing and new function space priors in Chapter 5. Chapter 6 formulates reinforcement learning for continuous value functions under the Bayesian paradigm by introducing a likelihood for stochastic control problems over function spaces. A modified version of Chapters 5 and 6 is submitted for publication as Sell and Singh (2020).

Chapter 2

Markov Chain Monte Carlo

This chapter introduces the foundation of the chapters to follow: Markov chain Monte Carlo methods. Not all these algorithms are confined to the Bayesian framework and work for generic probability distributions π. Discussing all existing Markov chain Monte Carlo methods is beyond the scope of this thesis, this chapter is thus restricted to the most popular ones and those needed for the work in later chapters. Some popular samplers omitted here include the elliptical slice sampler (Murray et al., 2010), which is a modification of the slice sampler (Neal, 2003), and hit-and-run samplers such as in Bélisle et al. (1993), to name a few.

This chapter is organised as follows: the first section summarises important definitions and theorems relating to discrete-time Markov chains, in particular the notions of stationary distributions and ergodic chains are introduced. Section 2.2 firstly explains how stochastic differential equations can be used for sampling, before giving concrete examples of these, namely the Langevin equations and Hamiltonian dynamics. In Section 2.3 we go back to discrete-time Markov chains and discuss reversible sampling algorithms. A particular focus of that section is the generic Metropolis-Hastings sampler, and some explicit variations thereof. To conclude this chapter, we look at piece-wise deterministic Markov processes and two examples of such processes in Section 2.4. Nothing in this chapter is novel work, instead it introduces the basic algorithms used in the remainder of this thesis, with advanced modifications discussed in the later chapters.

Throughout this chapter, the notation $p(X = dy)$ clarifies which random variable probabilities are calculated with respect to, wherever confusions are impossible, notations such

as $\mu(dy)$ are used to improve readability. The same holds for conditional distribution where $\mu(Y = dy|X = x)$ is simplified to $\mu(dy|x)$ where appropriate.

2.1 Discrete-Time Markov Chains

A *discrete-time Markov chain* on a state space $\mathcal{X}$ is a memoryless sequence of random variables $\{X_i\}_{i=1}^N$, with $X_i \in \mathcal{X}$ for all i and $N \in \mathbb{N} \cup \{\infty\}$, i.e.

$$p(X_k = dy|X_1 = x_1, \ldots, X_{k-1} = x_{k-1}) = p(X_k = dy|X_{k-1} = x_{k-1}),$$

for any $k \in \mathbb{N}$ assuming the first probability is non-zero.

A Markov chain is called *ergodic*, if for any initial distribution μ_0, the estimator (1.3) converges to the integral (1.2) almost surely for any sufficiently regular function h, and the ultimate goal of Markov chain Monte Carlo is to define algorithms that result in ergodic Markov chains. To establish ergodicity, we are relying on further concepts which we introduce now.

Given a state x, the mapping $x \mapsto K(x, dy)$ defines a probability measure in the second argument of the *Markov transition kernel* K. The name is justified as the kernel describes the conditional transition probabilities

$$p(X_k = dy|X_{k-1} = x_{k-1}) = K(x_{k-1}, dy).$$

Given this kernel and an initial distribution μ_0, all unconditional distributions $\mu_k(dy) = p(X_k = dy)$ of the chain can be calculated using

$$\mu_k(dy) = \int_{\mathcal{X}} \cdots \int_{\mathcal{X}} K(x_{k-1}, dy) K(x_{k-2}, dx_{k-1}) \cdots K(x_1, dx_2) \mu_0(dx_1).$$

A measure μ is called a *stationary measure* if

$$\mu(dy) = \int_{\mathcal{X}} K(x, dy) \mu(dx), \tag{2.1}$$

intuitively speaking, the measure does not change if the transition kernel is applied.

A kernel K is called *reversible* with respect to a measure μ if

$$\int_A K(x,B)\mu(dx) = \int_B K(x,A)\mu(dx) \tag{2.2}$$

for any μ-measurable sets A and B, that is, the probability of being in A and transitioning to B is as likely as being in B and transitioning to A. If the kernel K and the measure μ have densities k and f with respect to some base measure, the reversibility condition can be simplified to

$$f(x)k(x,y) = f(y)k(y,x), \tag{2.3}$$

which we refer to as *detailed balance condition*.

Reversibility is a desirable property, as it immediately implies stationarity of μ:

$$\mu(A) = \int_A \mu(dx) = \int_A K(x,\mathcal{X})\mu(dx) \overset{(2.2)}{=} \int_{\mathcal{X}} K(x,A)\mu(dx),$$

giving (2.1) as desired. In practice, reversibility is often easier to show, but is no requirement for stationarity. As we will see in the remainder of this chapter, reversible samplers are often easy to construct, but can suffer from poor mixing behaviour, which motivated the development of sampling algorithms based on non-reversible chains. Reversible samplers are discussed in Section 2.3, some non-reversible ones in Section 2.4.

To conclude this section, we state the desirable property of ergodicity. A set A is called *K-invariant* if it is measurable in the second argument of the kernel and furthermore $K(x,A) = 1$ for all $x \in A$. A measure μ is called *ergodic* if for any μ-measurable and K-invariant set A, it holds that $\mu(A) \in \{0,1\}$. Ergodicity of the measure μ implies ergodicity of the Markov chain induced by K by Birkhoff's theorem (Petersen, 1989).

In what follows, we will always first show that the constructed Markov chains have the desired stationary measure, and then, if possible, that the chain is also ergodic. For general ergodic theory, we refer the reader to e.g. Petersen (1989).

2.2 Sampling Using Stochastic Differential Equations

In this section, we discuss how diffusion processes, that is solutions to stochastic differential equations, can be used to derive MCMC samplers. This can be done in two different

ways: one can either simulate a diffusion process that has the desired target as its stationary distribution, or one can use the diffusion process to construct reversible MCMC samplers. We begin with a discussion of the former, and assume that the target distribution π admits a density which can be written as

$$f(x) \propto \exp(-U(x)), \tag{2.4}$$

where U is called the *potential* of π. Sometimes, it is useful to augment the target distribution by an auxiliary variable, which often introduces some notion of velocity or momentum. The discussion therefore continues by defining a variable z, where either $z = x$ or $z = (x, v)$. In the latter case, the potential of the augmented variable will be referred to as the *Hamiltonian* H,

$$f(x, v) \propto \exp(-H(x, v)). \tag{2.5}$$

The next paragraphs treat either case of z, and special cases are discussed in the following subsections.

The continuous-time Markov processes of interest in this section can be written as stochastic differential equation (SDE) on $\mathbb{R}^m$ of the form

$$dZ_t = b(Z_t)dt + \sqrt{2D(Z_t)}dB_t, \tag{2.6}$$

where B_t is a m-dimensional Wiener process, and b and D are the location dependent drift and diffusion coefficients, respectively. If $D = 0$, the SDE has no stochastic part, and reduces to an ordinary differential equation. The following theorem, taken from and proved in Ma et al. (2015), states under which conditions the SDE (2.6) leaves the target distribution with density given by (2.5) invariant:

Theorem 1. *μ with density given by* (2.5) *is a stationary distribution of the dynamics* (2.6) *if, for all z, μ-almost surely, the drift can be written as*

$$b(z) = -[D(z) + Q(z)]\nabla H(z) + \Gamma(z) \tag{2.7}$$

for a positive semidefinite matrix $D(z)$, skew symmetric matrix Q, and Γ defined by

$$\Gamma_i(z) = \sum_{j=1}^{d} \frac{\partial}{\partial x_j}\Big(D_{ij}(z) + Q_{ij}(z)\Big).$$

If D is positive definite, or if ergodicity can be shown, μ is the unique stationary distribution.

It can further be shown that in fact all SDEs of form (2.6) that have π as their stationary distribution, permit the drift function to be written as in (2.7), see Ma et al. (2015). In what follows, we will consider different choices of z, D, and Q to sample from a target distribution μ.

2.2.1 The Langevin Equations

The *overdamped Langevin equation* is the special case of the generic (2.6) equation with $z = x$, constant $D(x) = D$, and $Q(x) = 0$, such that $b(x) = -\nabla U(x)$, resulting in

$$dX_t = -D\nabla H(X_t)dt + \sqrt{2D}dB_t, \tag{2.8}$$

where $U(x)$ is the potential given by (2.4).

The dynamics of the overdamped Langevin equation are characterised by reversible, and very diffusive behavior. A generic way to alleviate these backtracking tendencies over short time-scales is to introduce persistence in the trajectories via a notion of velocity. To this end, the target space $\mathcal{X}$ is augmented with $\mathbb{R}^d$, and on this space let v be a d-dimensional vector of velocities drawn from $\mathcal{N}(0, \sigma I_d)$. The augmented Hamiltonian of interest is $H(x, v) = U(x) + \|v\|_2^2/(2\sigma)$. Defining $z = (x, v) \in \mathbb{R}^{2d}$, and choosing

$$D = \begin{pmatrix} 0 & 0 \\ 0 & \gamma\sigma I_{d\times d} \end{pmatrix}, \quad Q = \begin{pmatrix} 0 & \sigma I_{d\times d} \\ -\sigma I_{d\times d} & 0 \end{pmatrix},$$

gives the *underdamped Langevin equations*

$$\begin{cases} dX_t &= \gamma V_t dt \\ dV_t &= -\nabla U(X_t)dt - \gamma\sigma V_t dt + \sqrt{2\sigma}dB_t \end{cases} \tag{2.9}$$

for $\sigma > 0$ and $\gamma > 0$ a speed parameter. This process has a stationary distribution with density proportional to $\exp(-H(x, v)) = \exp(-U(x) - \|v\|_2^2/(2\sigma))$, and the marginal distribution of x is the desired target distribution π with density $f(x) \propto \exp(-U(x))$.

ULA

To use the Langevin equations for sampling, one simply discretises the SDE. Using the Euler-Maruyama scheme to discretise equation (2.8) results in the *Unadjusted Langevin Algorithm* (ULA):

$$X_{t+1} = X_t - \delta D \nabla U(X_t) + \sqrt{2D}\xi_t, \tag{2.10}$$

where δ is a user-specified step size, and the ξ_t are drawn independently from a standard normal distribution. This algorithm thus constructs an ergodic Markov chain to generate samples that are approximately distributed according to the desired target distribution π. A discretisation bias is inevitable (Kloeden and Platen, 2013) but can be reduced by choosing a smaller step size, or it can be corrected for as we will discuss in Section 2.3.2. The simplest discretisation is the Euler-Maruyama discretisation which is used in Algorithm 1.

Algorithm 1 Unadjusted Langevin Algorithm using a Euler-Maruyama discretisation scheme

procedure UNADJUSTED LANGEVIN ALGORITHM
  $x_0 \sim \pi(\cdot)$ ▷ Draw initial value x_0
  Pick step size δ
  for $n = 1 \dots$ **do**
    $\xi_{n-1} \sim \mathcal{N}(0, I_{d\times d})$
    $x_n \leftarrow x_{n-1} - \delta \nabla U(x_{n-1}) + \sqrt{2\delta}\xi_{n-1}$
  end for
end procedure

SK-ROCK

To allow for larger step sizes while keeping the discretisation error under control, more advanced integrators to discretise (2.8) can be used. One such class are stabilised explicit integrators such as a stochastic second kind orthogonal Runge-Kutta-Chebyshev method (SK-ROCK) (Abdulle et al., 2018), which are particularly strong for stiff stochastic differential equations. We will see numeric evidence of their strong performance in Chapter 3, Algorithm 2 describes the implementation.

Algorithm 2 SK-ROCK algorithm, modified from Pereyra et al. (2020), T_j is the Chebyshev polynomial of the first kind of order j

procedure SK-ROCK
  $x_0 \sim \pi(\cdot)$ ▷ Draw initial value x_0
  Pick $s \in \{3, \dots, 15\}$ ▷ Number of extrapolation points
  $\eta = 0.05$
  $l_s \leftarrow ((s-0.5)^2 \times (2 - 4/3\eta) - 1.5$
  $\omega_0 \leftarrow 1 + \eta/s^2$
  $\omega_1 \leftarrow T_s(\omega_0)/T_s'(\omega_0)$
  $\mu_1 \leftarrow \omega_1/\omega_0$
  $\nu_1 \leftarrow s\omega_1/2$
  $k_1 \leftarrow s\omega_1/\omega_0$
  Pick step size δ
  for $n = 1 \dots$ **do**
    $\xi_{n-1} \sim \mathcal{N}(0, I_{d\times d})$
    $K_0 \leftarrow x_{n-1}$
    $K_1 \leftarrow x_{n-1} + \mu_1 \delta \nabla U(x_{n-1} + \nu_1 \sqrt{2\delta}\xi_{n-1}) + k_1 \sqrt{2\delta}\xi_{n-1}$
    for $j = \{2, \dots, s\}$ **do**
      $\mu_j \leftarrow 2\omega_1 T_{j-1}(\omega_0)/T_j(\omega_0); \quad \nu_j \leftarrow 2\omega_0 T_{j-1}(\omega_0)/T_j(\omega_0); \quad k_j \leftarrow 1 - \nu_j$
      $K_j \leftarrow \mu_j \delta \nabla U(K_{j-1}) + \nu_j K_{j-1} + k_j K_{j-2}$
    end for
    $x_n \leftarrow K_s$
  end for
end procedure

UULA

One possible discretisation of the underdamped Langevin dynamics (2.9) was studied in Ma et al. (2019), which we present here as it is the one we use in the numerical experiments in Chapter 3, another popular integrator is the BAOAB scheme which, amongst other integrators, is discussed in Leimkuhler and Matthews (2016). If the current position and velocity are (x_t, v_t), the next iteration is given by

$$\begin{cases} x_{t+1} = x_t + \frac{1-\beta}{\gamma} v_t - \frac{1}{\gamma}(\nu - \frac{1-\beta}{\gamma\xi})\nabla U(x_t) + W_x \\ v_{t+1} = \beta v_t - \frac{1-\beta}{\gamma\xi}\nabla U(x_t) + W_v, \end{cases} \tag{2.11}$$

where $\nu = t_{n+1} - t_n$ is the step size, $\beta = \exp(-\gamma\xi\nu)$, and $(W_x, W_v) \sim \mathcal{N}(0, \Sigma)$ is Gaussian noise with covariance

$$\Sigma = \begin{pmatrix} \frac{1}{\gamma}\left(2\nu - \frac{3}{\gamma\xi} + \frac{4\beta}{\gamma\xi} - \frac{\beta^2}{\gamma\xi}\right) I_{d\times d} & \frac{1+\beta^2-2\beta}{\gamma\xi} I_{d\times d} \\ \frac{1+\beta^2-2\beta}{\gamma\xi} I_{d\times d} & \frac{1-\beta^2}{\xi} I_{d\times d} \end{pmatrix}.$$

Recent theoretical advances have elucidated non-asymptotic properties and the speed of convergence in probability metrics (KL-divergence, Wasserstein distance, etc.) for various formulations of Langevin-based algorithms, see e.g. Cheng et al. (2017); Dalalyan (2017); Wibisono (2019). In the case where the Hamiltonian H is m-strongly convex and Lipschitz-differentiable with parameter L, the number of samples required to achieve ϵ precision in Wasserstein-2 distance scales with $O(\sqrt{d})$ for the underdamped Langevin equation, with d the dimension of the model, compared to $O(d)$ for the overdamped dynamics Cheng et al. (2017).

2.2.2 Hamiltonian Dynamics

Similar to the underdamped Langevin equations, Hamiltonian dynamics are defined on an extended state space. We define the variable of interest to be $z = (x, p)$ and the augmented Hamiltonian as $H(x, p) = U(x) + p^T M^{-1} p/2$. M is called the *mass matrix*, motivated from the origin of the dynamics in the physics literature, where the dynamics describe how an object with position x, momentum p, and mass M moves on a frictionless surface. The *Hamiltonian dynamics* are

$$\begin{cases} dX_t &= M^{-1} P_t \\ dP_t &= -\nabla U(X_t) \end{cases}. \tag{2.12}$$

Notably, the Hamiltonian dynamics have no stochastic component, they indeed have $D = 0$ in the generic SDE (2.6) and further

$$Q = \begin{pmatrix} 0 & I_{d\times d} \\ -I_{d\times d} & 0 \end{pmatrix}.$$

The use of these dynamics in sampling goes back to Duane et al. (1987), where the method was developed as a *hybrid Monte Carlo* method, while the method is today better

known as *Hamiltonian Monte Carlo* for obvious reasons. The desired target distribution π is the marginal of x of the joint distribution which is proportional to $\exp(-H(x,v)) = \exp(-U(x) - p^T M^{-1} p/2)$. While the stationarity of the desired target follows immediately from Theorem 1, ergodicity is not normally given: the deterministic dynamics preserve the energy of the system, and will thus not explore the full space. To alleviate this, the velocity component is resampled, and ergodicity can be shown under mild assumptions, see Durmus et al. (2017b); Livingstone et al. (2019).

While the continuous-time dynamics preserve the stationary distribution, discretising it will normally lead to non-neglible numerical errors. A first step to reduce these is to use a symplectic integrator (Leimkuhler and Matthews, 2016) such as the popular leapfrog integrator. The leapfrog integrator with step size δ starting from (X_t, P_t) is defined by

$$\begin{aligned} P_{t+1/2} &= P_t - \frac{\delta}{2}\nabla U(X_t) \\ X_{t+1} &= X_t + \delta M^{-1} P_{t+1/2} \\ P_{t+1} &= P_{t+1/2} - \frac{\delta}{2}\nabla U(X_{t+1}). \end{aligned}$$

Furthermore, a correction step will also be added to correct for the discretisation bias, we will discuss this in Section 2.3.2.

Hamiltonian dynamics can be seen as an example of a piece-wise deterministic Markov process, characterised by deterministic dynamics combined with occasional jumps. Such processes will be discussed in more detail in Section 2.4.

2.3 Reversible Sampling Algorithms

2.3.1 Gibbs Sampling

To sample from multivariate distributions π defined over $\mathcal{X} \subset \mathbb{R}^d$, a popular sampler is the Gibbs Sampler (Geman and Geman, 1984) which breaks the full sampling problem into one-dimensional ones. The Gibbs sampler in its simplest form requires conditional distributions to be analytically tractable, this requirement can be relaxed as we will discuss in Section 2.3.2. Algorithm 3 describes an implementation of the Random Scan Gibbs Sampler.

Algorithm 3

procedure RANDOM SCAN GIBBS SAMPLER
  $x_0 \sim \pi(\cdot)$ ▷ Draw initial value x_0
  for $n = 1 \dots$ **do**
    $i \sim \mathcal{U}(\{1, \dots, d\})$ ▷ Sample a coordinate uniformly
    $\omega \sim \pi(X^i = \cdot | X^{-i} = x_{n-1}^{-i})$ ▷ Sample from full conditionals
    $x_n^i \leftarrow \omega$, $x_n^{-i} \leftarrow x_{n-1}^{-i}$
  end for
end procedure

Instead of sampling a coordinate uniformly which characterises the *random scan* Gibbs sampler, one may also iterate through the coordinates in a systematic fashion which, not surprisingly, defines the *systematic scan* Gibbs sampler. The random scan version is π-reversible while the systematic scan sampler is not, yet both versions have the desired stationary distribution (Casella and George, 1992). However, we emphasise that this crucially depends on the initialisation and that the chain might not explore the full space, and therefore cannot be ergodic in general. As an illustration, take a simple distribution with point-masses at two locations: $\pi(\{(0,0)\}) = \pi(\{(1,1)\}) = 1/2$. If initialised at either point, the sampler will be stuck and never move to the other point, yet, as the chain is initialised by a draw from the target, any sample obtained using Algorithm 3 will trivially be a sample from desired distribution. On the other hand, taking the definition of ergodicity from Section 2.1, one can easily check that the set $A = \{(0,0)\}$ is K-invariant, yet $\mu(A) = 1/2$, so μ is not ergodic. Ergodicity of the chain arising from the Gibbs sampler can thus only be shown on a case-to-case basis. Other samplers do not suffer from such behaviour, as their global proposal moves normally ensure complete exploration of the state space.

2.3.2 The Metropolis-Hastings Algorithm

Over 60 years after its first explorations, the workhorse of MCMC arguably remains to be the Metropolis-Hastings algorithm, introduced to the statistical physics community in Metropolis et al. (1953), and generalised in Hastings (1970) and again in Tierney et al. (1998). We describe here the most basic ideas and algorithm, more elaborate schemes are discussed in the subsequent sections and chapters.

In what follows, we assume the existence of a Markov transistion kernel $Q(x, dy)$ which admits a density $q(y|x)$ with respect to some base measure (normally, but not always, the Lebesgue measure). We further assume the target distribution π admits a density f, for

simplicity with respect to the same base measure, and refer the reader to Tierney et al. (1998) for the general case. Given a current state x_n, the algorithm works as follows: a new state is proposed by sampling from the transition kernel, which is then accepted or rejected according to the Metropolis-Hastings criterion. Roughly speaking, a proposal y is accepted with a probability $a(x, y)$, proportional to how much more probable the proposal is in comparison to the current iterate. More precisely, $a(x, y) = 1$ if $f(x)q(y|x) = 0$, otherwise

$$a(x, y) = \min\left(1, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right). \tag{2.13}$$

Algorithm 4 describes the full Metropolis-Hastings algorithm.

Algorithm 4

procedure GENERIC METROPOLIS-HASTINGS SAMPLER
  $x_0 \sim \pi(\cdot)$ ▷ Draw initial value x_0
  for $n = 1 \dots$ **do**
    $y \sim Q(x_{n-1}, \cdot)$ ▷ Sample a proposal from the transition kernel
    $r \leftarrow f(y)q(x_{n-1}|y)/(f(x_{n-1})q(y|x_{n-1}))$ ▷ Calculate Metropolis-Hastings ratio
    $a \leftarrow \min(1, r)$ ▷ Calculate acceptance probability
    $u \sim \mathcal{U}([0, 1])$
    if $u < a$ **then**
      $x_n \leftarrow y$ ▷ Accept proposed move
    else
      $x_n \leftarrow x_{n-1}$ ▷ Reject proposal
    end if
  end for
end procedure

The stationarity of π follows from the Detailed Balance Condition 2.3 being satisfied, with the Metropolis-Hastings kernel being given by

$$K(x, dz) = \int q(y|x)\Big((1 - a(x, y))\delta_x(dz) + a(x, y)\delta_y(dz)\Big)dy,$$

with a as defined in (2.13).

In the following sections we will look at some explicit examples of different MH algorithms, ordered by increasing level of complexity. All of them require their step sizes to go to 0 as higher dimensional distributions are considered: the scaling limits hold for a product of d independent normal distributions, but this assumption can be slightly relaxed, see e.g. Beskos et al. (2013). If the step size is kept fixed while increasing the dimension

of the target distribution, the acceptance rates of these MH algorithms will go to 0. Chapter 5 discusses two algorithms which allow step sizes to be kept constant when increasing the dimensionality of the target, this is achieved through appropriate preconditioning and using a Crank-Nicolson discretisation scheme for an underlying SDE.

Random Walk Metropolis Hastings

One of the most popular variants of Algorithm 4 for targets defined on $\mathbb{R}^d$ uses a Gaussian transition kernel $Q(x,\cdot) = \mathcal{N}(\cdot|x,C)$ with a covariance matrix specified by the user. With this choice, $q(x|y) = q(y|x)$, and the Metropolis-Hastings ratio simplifies to $r = f(y)/f(x)$. Oftentimes, $C = \delta I$, that is, the transition kernel is a scaled isotropic Gaussian distribution centered at the current iterate, this is the *Random Walk Metropolis Hastings Algorithm* (RWMH). In the Bayesian setting, if the prior distribution π_0 is a Gaussian distribution, another popular choice it to set C to be the prior covariance, which is coined a *preconditioned* version of RWMH. Preconditioning will play an important role in later parts of this thesis, and can significantly improve the mixing behaviour of the chains.
While stationarity of the target measure follows from the Detailed Balance Condition 2.3, ergodicity is similarly easy to show for the RWMH as the only K-invariant sets are $A = \emptyset$ and $A = \mathbb{R}^d$.

RWMH is arguably the simplest Metropolis-Hastings algorithm, it only requires evaluations of the target density. Preconditioning already introduces some additional complexity, as sampling from multivariate normal distributions with complicated covariances gets computationally costly in high-dimensional spaces. Another problem with RWMH is that one needs to take smaller steps if targeting higher dimensional distributions: the step size δ for a product of d independent normal distributions should be scaled as $O(d^{-1})$ to achieve the optimal acceptance ratio of around 0.234 (Roberts et al., 1997). Under this acceptance ratio, the resulting Markov chain decorrelates the fastest.

Metropolis-Within-Gibbs Sampler

As mentioned in Section 2.3.1, the Gibbs sampler requires sampling from the conditional distributions. If this is not possible, one can use the Metropolis-Hastings algorithm to sample from the conditional distributions. This procedure is known as the Metropolis-within-Gibbs algorithm.

Metropolis-adjusted Langevin Algorithm

The Euler-Maruyama discretisation of the unadjusted Langevin algorithm (2.10) leads to a bias that is often corrected with a Metropolis-Hastings step, leading to the popular Metropolis-adjusted Langevin Algorithm (MALA) Roberts et al. (1996), where the transition kernel in Algorithm 4 is given by

$$Q(x,\cdot) = \mathcal{N}(\cdot|x - \delta\nabla U(x), \delta C),$$

where again $C = \delta I$, and δ denotes a step size. As before, a preconditioned version uses e.g. the prior covariance matrix as proposal covariance. The step size is tuned to achieve an acceptance rate of around 50% to 70%, as this is close to the optimal acceptance rate for a product of independent Gaussian distributions (Roberts and Rosenthal, 1998).

Various modifications and extensions of this method exist, such as the covariance being dependent on the current location (Roberts and Stramer, 2002) to improve mixing, the resulting chain arises from discretising SDEs with location-dependent diffusive behaviour (Kent, 1978) with an additional MH correction step.
Here again, stationarity follows from the Detailed Balance Condition 2.3 being satisfied, and ergodicity can be shown as for RWMH.

Whether this correction step is included or not, depends on the application: for large-scale, data-intensive models the unadjusted Langevin algorithms (ULA) is often preferred, and full gradient evaluations are replaced by stochastic gradients which use only a subset of the data to calculate gradients (Welling and Teh, 2011). While giving up asymptotic exactness, mixing is generally improved Durmus et al. (2017a) and costly gradient evaluations are not wasted on rejected proposals.

MALA is more complex than RWMH, as it additionally requires evaluations of gradients of the potential U. The benefit of this method is that the step size δ needs to be scaled only as $O(d^{-1/3})$ to achieve the algorithm's optimal acceptance ratio, which for MALA is around 0.574 (Roberts et al., 2001).

HMC

As for the Langevin equations, simulating the Hamiltonian dynamics numerically leads to errors that can badly impact the inference results. To alleviate this, a MH correction step is included.

Here again, several modifications and improvements exist. One such modification is a location-dependent covariance (Girolami and Calderhead, 2011), another one addresses the issue of choosing the number of leapfrog steps: the authors of Hoffman and Gelman (2014) propose the No-U-Turn Sampler which runs leapfrog steps until the trajectories backtrack, preventing this backtracking behaviour while at the same time maximising the number of steps taken. Slight further modifications of the No-U-Turn Sampler further improved the algorithm which is one of the most widely used MCMC algorithms today.

Stationarity of the chain follows, once more, from the Detailed Balance Condition 2.3, where the momentum variable is flipped after the integration of the dynamics, i.e. after l leapfrog steps one sets $P_{t+l} \leftarrow -P_{t+l}$. As the momentum variable is resampled, this is not impacting the performance of the sampler, and only aids the theoretical analysis. Ergodicity is harder to show, and we refer the reader to Livingstone et al. (2019) for various results on ergodicity of HMC.

In comparison to MALA, HMC is yet more complex, as an auxiliary variable is introduced and more complicated dynamics need to be simulated. The advantage of this is that HMC requires the leapfrog step size to be scaled only as $O(d^{-1/4})$ to achieve an optimal acceptance ratio of 0.651 (Beskos et al., 2013).

2.4 Piece-Wise Deterministic Markov Processes

While the reversibility of Markov chains gives rise to a simple-to-check criterion to ensure that the chain targets the distribution of interest, the back-tracking behaviour also slows down mixing. To alleviate this, non-reversible samplers have been of increased interest over the last decade, and we will here introduce a subclass of them, the Piece-wise Deterministic Markov Processes (PDMPs). These are almost everywhere continuous Markov processes, characterised by deterministic dynamics interspersed with jumps occurring at random times. They are described by a *jump kernel* that depends on the current state, which we denote by $Q(z, \cdot)$ indicating its Markovian nature, and a *rate function* $\rho(z)$ which triggers events τ of an inhomogeneous Poisson process (iPP):

$$p(\tau > t) = \exp\left(-\int_0^t \rho(z(s))ds\right). \tag{2.14}$$

A generic PDMP sampler is given by Algorithm 5, and one aims to find Q and ρ such that the resulting Markov process admits the target distribution μ as a stationary distribution. In

most cases, the deterministic dynamics are chosen to be easy to simulate, and the challenge when using PDMPs then lies in the simulation of the iPP, which we discuss at the end of in this section.

Algorithm 5

procedure GENERIC PDMP
  $z_0 \sim \mu(z)$ ▷ Draw initial value z_0 from μ
  $t \leftarrow 0$
  for $n = 1 \dots$ **do**
    Solve $\dot{\zeta} = g(\zeta)$, starting from $\zeta_0 = z_t$ ▷ Solve deterministic dynamics
    $\tau \sim \text{iPP}(\rho(\zeta_t))$ ▷ Calculate event time along trajectory
    $z_{[t:t+\tau]} \leftarrow \zeta_{[0:\tau]}$ ▷ Amend z trajectory
    $z_{t+\tau} \sim Q(z_{t+\tau}, \cdot)$ ▷ Jump
    $t \leftarrow t + \tau$ ▷ Update time
  end for
end procedure

In the processes discussed subsequently in this section, the target space $\mathcal{X}$ is once more augmented by a velocity variable, and we define the augmented variable $z = (x, v)$. Let ν be the marginal of v, and let the desired joint target distribution be $\mu(dz) = \mu(dx, dv) = \pi(dx)\nu(dv)$. We further restrict ourselves to processes where the trajectories in the variable of interest x are continuous, and the jumps occur exclusively in the velocity component. In this, these samplers are similar to the Hamiltonian Monte Carlo method (without the MH correction step), where the momentum variable is resampled after fixed time steps (rather than being sampled from an iPP), and the variables otherwise follow the deterministic dynamics (2.12).

The samplers discussed here further require μ to admit a differentiable target density, and we again write this density as $f(z) = f(x, v) \propto \exp(-H(x, v))$. U will again denote the potential of π.

The Zig-Zag Sampler

The Zig-Zag Sampler (ZZS), introduced by Bierkens et al. (2019a), derives its name from the trajectories it produces, see the left panels in Figure 2.1. The augmented velocities are of the form $v \in \{-1, 1\}^d$ and the distribution over these velocities is uniform, such that the joint distribution is given by $\mu(dx, dv) = \pi(dx)\mathcal{U}(dv)$. The process $(z_t)_{t\geq 0} = (x_t, v_t)_{t\geq 0}$ leaves the joint distribution of x and v invariant. The deterministic dynamics are given as the solution to the ordinary differential equation $(\dot{x}_t, \dot{v}_t) = (v_t, 0)$, that is, they follow straight lines

between jumps. The jumps are triggered through d inhomogeneous Poisson processes (iPPs), which are associated with the individual dimensions of the problem. Intuitively, when the trajectory is moving into less probable regions, a velocity component is flipped with a certain probability, such that the process targets the desired distribution. The ideal rate at which a velocity v_i is flipped is given by

$$\rho^i_{ZZ}(t; x, v) = \max\left\{0, \frac{\partial}{\partial x_i} U(x + tv) \cdot v_i\right\}. \tag{2.15}$$

Globally, the rate at which any change in velocity occurs is given by

$$\rho_{ZZ}(t; x, v) = \sum_{i=1}^{d} \rho^i_{ZZ}(t; x, v).$$

The inhomogeneous Poisson processes with rates (2.15) are simulated, returning proposed event times t_j and the first event is picked to initiate a jump:

$$j^* = \underset{j=1,\dots,d}{\arg\min}\, \tau_j, \qquad \tau = \tau_{j^*}.$$

The velocity variable is then updated by applying the Flip operator $\mathfrak{F}_{j^*}$ to v:

$$\mathfrak{F}_{j^*}(v_i) = \begin{cases} -v_i & \text{if } i = j^* \\ v_i & \text{else.} \end{cases}$$

This continuous-time Markov process, arising from the combination of following the trajectories and flipping velocity components at random times, has stationary distribution $\pi(dx)\mathcal{U}(dv)$, and we refer the reader to Bierkens et al. (2019b) for proofs of ergodicity.

In practice, the main difficulty of using the ZZS lies in the simulation of the iPPs, a problem not exclusive to the ZZS. We will postpone the discussion of simulating them until after the next two subsections.

The Bouncy Particle Sampler

The Bouncy Particle Sampler (BPS) is also named after the looks of its trajectories, which look like a particle moving through the x space and bouncing off an invisible wall at random intervals, see the right panels in Figure 2.1. It was developed by Peters and de With (2012) to solve problems in statistical physics, and introduced to the general statistics community by

Bouchard-Côté et al. (2018). For the BPS, the augmented velocity variable is continuous, following a standard normal distribution. The deterministic dynamics between jumps are the same as for the ZZS, given by the flow $(\dot{x}_t, \dot{v}_t) = (v_t, 0)$. The jumps are once more triggered by an inhomogeneous Poisson process, with the (single) rate function given by

$$\rho_{BPS}(t; x, v) = \max\{0, \langle v, \nabla U(x + tv)\rangle\}. \tag{2.16}$$

Whenever an event occurs, the velocity is updated by applying a reflection operator $\Re_x(v)$ (following the name of the method, alternatively called a bounce operator) which reflects the velocity at a hyperplane orthogonal to the gradient at x:

$$\Re_x(v) = v - 2\frac{\langle v, \nabla U(x)\rangle}{||\nabla U(x)||^2}\nabla U(x). \tag{2.17}$$

The practical difficulty when using the BPS is again the simulation of the iPP which we discuss in the next subsection. The continuous-time Markov process arising from the BPS has stationary distribution $\pi(dx)\mathcal{N}(dv; 0, I)$, but to ensure ergodicity of the sampler, the velocity needs to be resampled regularly. For a full theoretical treatment of the ergodicity properties of the BPS, we refer the reader to Deligiannidis et al. (2019); Durmus et al. (2020) and for more general theoretical treatment of PDMPs to Andrieu et al. (2018). The BPS can further be localised as described in Bouchard-Côté et al. (2018) if the potential function U factorises over groups of variables, as we will see in Chapter 3, this is crucial for competitive performance of the BPS when targeting higher-dimensional distributions.

The Hamiltonian Bouncy Particle Sampler

An alternative specification for the dynamics of the BPS was introduced in Vanetti et al. (2017), which we will now detail. Consider the Hamiltonian of both the target variable and the velocity $H(x, v)$ defined by

$$H(x, v) = U(x) + \frac{1}{2}v^t v = -\ell(x) - \log \pi_0(x) + \frac{1}{2}v^t v + c,$$

where c is the normalisation constant which we will drop from the calculations for notational convenience, this does not affect the sampling properties. For an auxiliary spherical potential

$V(x) = \frac{1}{2}(x-m)^T\Sigma^{-1}(x-m)$, the Hamiltonian

$$H(x,v) = \underbrace{-\ell(x) - \log\pi_0(x) - V(x)}_{\hat{U}(x)} + \underbrace{V(x) + \frac{1}{2}v^t v}_{\hat{H}(x,v)}, \tag{2.18}$$

naturally splits into two parts. The dynamics of the Hamiltonian $\hat{H}$ are available in closed form and can be solved explicitly for any m and Σ:

$$\begin{cases} dx_t &= \nabla_v \hat{H}(x_t, v_t) = v_t \\ dv_t &= -\nabla_x \hat{H}(x_t, v_t) = \Sigma^{-1}(x-m). \end{cases} \tag{2.19}$$

If the model under consideration has a Gaussian component in x, one is advised to choose the auxiliary potential to equal this energy function. For example, if π_0 is Gaussian, setting $V(x) = -\log\pi_0(x)$ reduces the potential $\hat{U}(x)$ in (2.18) to only depend on the likelihood. The rates and reflection operator of the resulting Hamiltonian BPS (HBPS) are defined by (Vanetti et al., 2017)

$$\rho_{HBPS}(t) = \max\{0, \langle v_t, \nabla\hat{U}(x_t)\rangle\}$$
$$\hat{\Re}_x(v) = v - 2\frac{\langle v, \nabla\hat{U}(x)\rangle}{\|\nabla\hat{U}(x)\|^2}\nabla\hat{U}(x),$$

and the deterministic flow is determined by the Hamiltonian dynamics (2.19). The resulting continuous-time Markov process has $\pi(dx)\mathcal{N}(dv; 0, I)$ as stationary distribution for any choice of m and Σ. Choosing these auxiliary variables carefully, the Hamiltonian BPS can be localised, in the sense that a factor decomposition is made explicitly available.

Simulating inhomogeneous Poisson processes

For both the ZZS and the BPS, the difficulty lies in the simulation the event times of the respective inhomogeneous Poisson processes (iPP), which initiate a change in velocity. For a single iPP with rate $\rho(t) = \rho(t; x, v)$,we define

$$\varrho(t) = \int_0^t \rho(s)ds.$$

For an iPP, the probability of the first event happening after time t is given by

$$p(\tau_1 > t) = \exp(-\varrho(t)) = \exp\left(-\int_0^t \rho(s)ds\right).$$

The first event time τ_1 can thus be sampled by sampling an event time of a homogeneous Poisson point process with constant rate 1, and using an inverse transform to obtain $\tau_1 = \varrho^{-1}(-\log(u))$, where $u \sim \mathcal{U}([0,1])$. This inverse transform will in general be not available.

An alternative approach is the thinning method, which requires a bound $\bar{\rho}(t)$ on the rate of interest $\rho(t)$. This can be either a global bound $\bar{\rho}(t) = c$ for some constant $c \in \mathbb{R}$, or a bound calculated using a fixed *look-ahead* $\theta > 0$: on the interval $[0,\theta]$, the rate function is continuous and attains a maximum such that we can choose

$$\bar{\rho} = \max_{t\in[0,\theta]} \rho(t) + \gamma, \qquad \gamma > 0,$$

where the extra rate γ ensures positivity of $\bar{\rho}$. An event time is simulated using the exponential distribution, $\tau \sim \mathrm{Exp}(\bar{\rho})$, and accepted with probability $\rho(\tau)/\bar{\rho}$; note that we can always calculate this as $\bar{\rho}$ is positive. If rejected, the procedure is repeated from $t = \tau$; if $\tau > \theta$, the procedure is repeated from $t = \theta$.

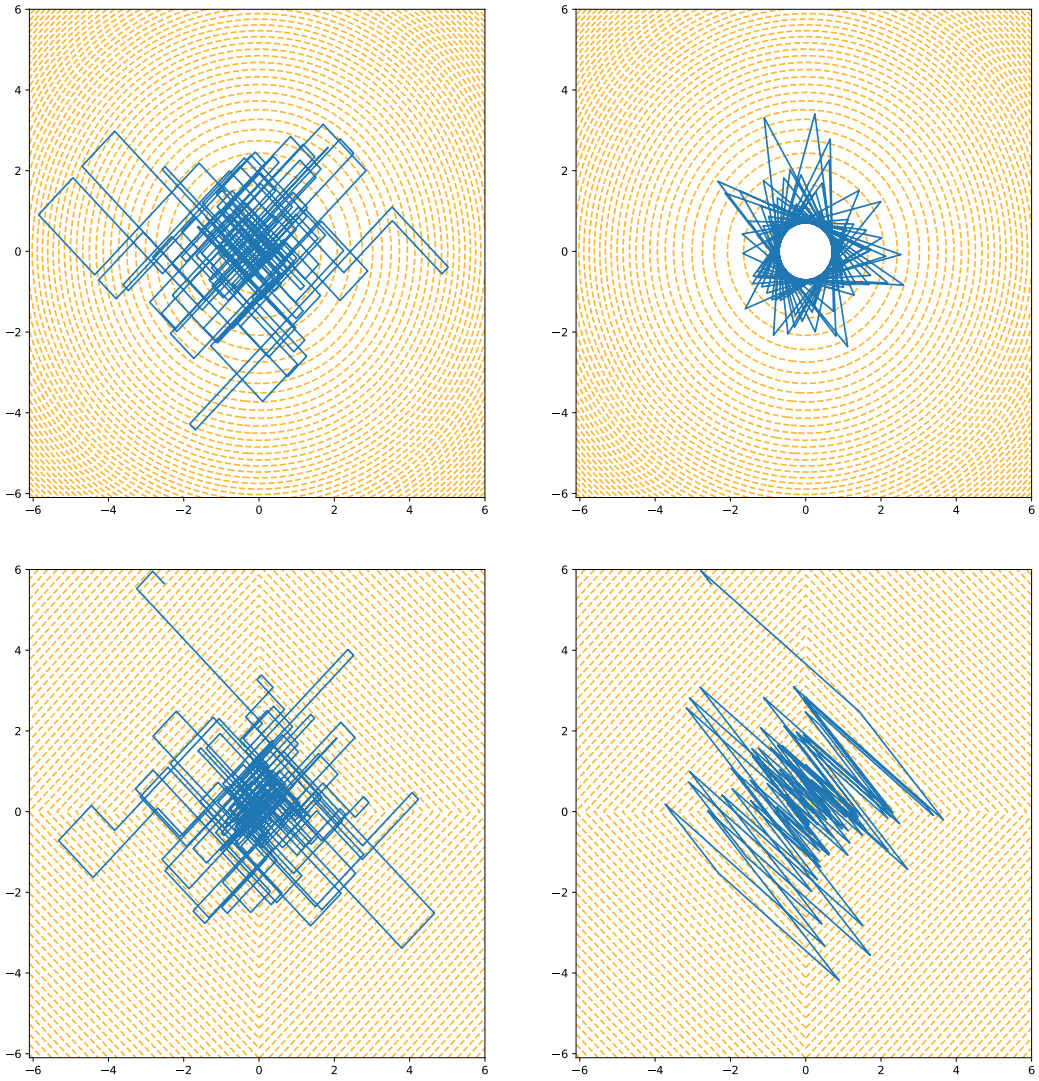


Fig. 2.1 Visualisation of trajectories from the BPS and ZZS.
Top row: Contours of a Gaussian potential in orange, ZZS trajectory (top left) and BPS trajectory (top right). The need for velocity resampling is obvious for the BPS, without velocity resampling, the trajectories will never enter the centre.
Bottom row: Contours of a Laplace potential in orange, ZZS trajectory (bottom left) and BPS trajectory (bottom right). Resampling velocities is crucial for fast exploration of the space.

Chapter 3

Non-Differentiable Posteriors

A modified version of the work presented in this chapter, together with additional work by Jacob V. Goldman which is excluded here, is published as Goldman et al. (2020). The article is joint work with Sumeetpal S. Singh; Jacob V. Goldman and I are joint first authors as our contributions to the publication are equal. Unless stated otherwise in the text, the work presented here is my own.

Throughout this chapter, $\pi(x)$ denotes the pdf of π and $\pi^\lambda(x)$ the pdf of π^λ to simplify notation.

3.1 Introduction

This chapter addresses the problem of efficient sampling from Bayesian posteriors that admit only almost everywhere differentiable density functions. This class of distributions includes all log-concave posteriors, as any convex function is almost everywhere differentiable; the class further contains all posteriors which are a product of a log-concave prior and a differentiable likelihood, or, more generally, posteriors which are the product of an almost everywhere differentiable prior and an almost everywhere differentiable likelihood. Such distributions can be found in numerous applications.

Their origins lie in optimisation, where the use of non-differentiable convex functions to regularise optimisation objectives often results in desired sparse solutions. The Least Absolute Shrinkage and Selection Operator (LASSO, Tibshirani (1996)), as well as its Bayesian interpretation thereof (Park and Casella, 2008), are widely used, and various other

penalties for sparse regularisation have been developed. These penalties can be interpreted as potentials of priors, and can be thus be translated to the Bayesian paradigm. Popular examples are the following:

- In image analysis, this approach has been used for image denoising (Rudin et al., 1992), image deconvolution (Babacan et al., 2008; Chambolle et al., 2010), and compressed sensing (Candès et al., 2006), amongst others.
- The non-convex but almost everywhere differentiable Bessel-K prior (Hosseini, 2019) has been used for sparse regularisation (Goldman et al., 2020), as an alternative to the Bernoulli-Laplace prior (Chaari et al., 2013) and the spike-and-slab prior (Ishwaran et al., 2005).
- In circular statistics, the wrapped Laplace distribution is used in sound source separation (Mitianoudis, 2012), and for modelling directional data in animal movements (Fernández-Durán, 2004; McVinish and Mengersen, 2008). As shown in Example 3.4.7, these circular posteriors can be multi-modal with many points of non-differentiability.
- The nuclear norm prior, corresponding to the 1-Schatten norm penalty in optimisation, and used for low-rank matrix completion (Babacan et al., 2008; Koltchinskii et al., 2011), is log-concave but not differentiable.

Sampling from all such distributions is challenging, as the (unadjusted) diffusion-based algorithms discussed in Section 2.2 require gradients everywhere in order to control the error from the discretisation bias. To evade this problem, one can target a smoothed density function instead of the log-concave prior directly, using, for example, the proximal operator. This operator is a minimisation programme which allows to smooth out non-differentiable functions. It is popular in the convex optimisation community (Nitanda, 2014; Parikh et al., 2014), the negative log-density function corresponding to the smoothed target is commonly known as the Moreau-Yosida envelope (MYE, Bauschke et al. (2011)). The obtained prior distributions will be called Moreau-Yosida priors in the subsequent sections, denoted by π^{λ}, and is continuously differentiable, allowing the employment of the diffusion algorithms discussed above.

Since the introduction of proximal operators to the sampling literature by Pereyra (2016), various extensions and modifications have been proposed. In Pereyra (2016), the author proposed to use a Langevin diffusion on π^{λ} to obtain proposals within a Metropolis-Hastings scheme. As not every step gets accepted, the authors of Durmus et al. (2018) have considered

the *unadjusted* Langevin diffusion on π^λ, with the idea that no computation time is wasted on rejected proposals. While a small bias is introduced when targeting π^λ instead of π, this can be chosen to be of a similar magnitude as the discretisation error of the diffusion, and the fast mixing of the chain makes it the preferable choice whenever practitioners do not need asymptotically exact solutions. This is, for example, the case in many imaging applications. Furthermore, statistical models are (usually) only an approximation to reality, such that the usefulness of asymptotic exactness can be questioned anyways. To allow larger step sizes when targeting the diffusion, the use of stabilised integrators of the diffusion have been considered in Pereyra et al. (2020). Lastly, a stochastic proximal algorithm and its convergence properties have been studied in Salim et al. (2019). As the underdamped Langevin equation is known to converge faster on strongly log-concave target distributions (Cheng et al., 2017), we propose to use the same diffusion to target the MYE-smoothed target π^λ in Section 3.3.2.

The accuracy of Monte Carlo estimates obtained using the approximate target crucially depends on a parameter λ, which controls the tightness of the approximation, and consequently controls the error. Our Theorem 4 highlights this dependence. While a large λ leads to better mixing of the associated Markov chains, a small λ results in better estimates. Good guidance on how to choose this tuning parameter is available in Durmus et al. (2018). If *exact* sampling is required, an additional Metropolis-Hastings correction step can correct for the discretisation bias and the approximation error, at the cost of slower mixing behaviour.

An alternative sampling approach is the use of piece-wise deterministic Markov processes (PDMPs), as they only require almost everywhere differentiability of the target density (Goldman et al., 2020). These samplers are asymptotically exact, and are applicable as long as global bounds on the gradients are available, defining the gradients to be 0 on the nullset $A = \{x : \pi(x) \text{ is not differentiable}\}$. In contrast to the methods using the MYE, PDMPs are not restricted to log-concave posteriors.

This chapter is now organised as follows: in Section 3.2 we recapitulate properties of proximal operators and the Moreau-Yosida envelope. Theorem 4 and Lemma 1 at the end of that section are new, their proofs can be found in Appendix 3.6. Section 3.3 presents existing sampling algorithms for the specified posteriors, and introduces a new second-order method. Extensive numerical examples are reported in Section 3.4, and a discussion on the benefits and disadvantages of the discussed methods in different settings can be found in Section 3.5.

3.2 Proximal Operators and the Moreau-Yosida Envelope

3.2.1 Proximal Operators

We will now describe, and collect a few useful results about, the proximal operator. These allow to simplify the gradient evaluation of the approximations to non-differentiable posteriors discussed in the next subsection. For any convex function $g : \mathbb{R}^d \to \mathbb{R}$, the proximal operator prox_g^λ with tightness parameter λ is defined through the minimisation problem

$$\text{prox}_g^\lambda(x) = \arg\min_u \left[g(u) + \frac{1}{2\lambda}\|x-u\|^2 \right]. \tag{3.1}$$

The proximal operator can also be interpreted as a generalisation of the Euclidean projection operator (Parikh et al., 2014). Importantly, for differentiable functions $g \in C^1(\mathbb{R}^d)$ the prox_g^λ operator satisfies

$$p = \text{prox}_g^\lambda(x) \iff x - p = \lambda \nabla g(p), \tag{3.2}$$

or, if g is convex but not differentiable, it still satisfies

$$p = \text{prox}_g^\lambda(x) \iff x - p \in \lambda \partial g(p),$$

where $\partial g(p)$ is the sub-differential $\partial g(p) = \{u \in \mathbb{R}^d : \forall y \in \mathbb{R}^d, (y-p)^T u + g(p) \leq g(y)\}$.

3.2.2 The Moreau-Yosida Envelope

The *Moreau-Yosida envelope* (MYE) which is defined by

$$g^\lambda(x) = \inf_z \left[g(z) + \frac{1}{2\lambda}\|x-z\|^2 \right] \tag{3.3}$$

can be used to approximate any non-differentiable and convex function g. Intuitively, the MYE enforces differentiability by adding a quadratic at any point of non-differentiability. It is indeed possible to calculate the derivative of g^λ using the proximal operator defined in the previous section (Bauschke et al., 2011, Theorem 12.30):

Theorem 2. *Let $g : \mathcal{X} \to]-\infty, \infty]$ be a proper lower semi-continuous convex function, and let $\lambda > 0$. Then g^λ is Fréchet differentiable, and its gradient is $1/\lambda$-Lipschitz continuous,*

given by

$$\nabla g^{\lambda}(x) = \frac{1}{\lambda}(x - \operatorname{prox}_{g}^{\lambda}(x)). \tag{3.4}$$

Furthermore, the MYE itself is also a convex function (as the infimum is a convex function and the quadratic term preserves minima). If g is L-Lipschitz continuous, g is close to its MYE g^{λ}, as shown by the following theorem from (Hosseini et al., 2019, Proposition 3.4 with $\lambda = 1/r$):

Theorem 3. *Let $g : \mathcal{X} \to]-\infty, \infty]$ be a proper lower semi-continuous convex function, and L-Lipschitz. Let $\lambda > 0$. Then for any $x \in \boldsymbol{dom}(g)$,*

$$0 \leq g(x) - g^{\lambda}(x) \leq \frac{L^2 \lambda}{2}.$$

Therefore, the tightness parameter λ should be chosen of the order $O(L^{-2})$, which ensures the MYE is reasonably close to g. The upper bound is often achieved: in Figure 3.1 the choice $\lambda = 0.25$ results in a Lipschitz constant $L = 1$, and for any x with $|x| > \lambda$ it is easy to see that $g(x) - g^{\lambda}(x) = \frac{\lambda}{2}$.

Rearranging Equation (3.4) reveals that iterative application of the proximal operator just corresponds to gradient descent of the MYE-smoothed version of g. In Figure 3.1, we provide a simple visual aid that illustrates the behaviour of both the MYE and proximal operator and the respective densities in a basic example, $g(x) = |x|$ such that $g \in C^0$. The resulting MYE in this case is the Huber loss function. In general, we do not have access to closed-form solutions to either the MYE or the proximal operator, and finding the solution to either using numerical methods can be quite computationally expensive.

Having recalled these basic facts about the proximal operator, we now provide an easily verifiable bound on the precision achievable when using the MYE in sampling algorithms: Theorem 4 quantifies the error obtained if we compute (exactly) an expectation with respect to the smoothed target distribution versus the true (non-differentiable) distribution in the sense that for suitably regular $f : \mathcal{X} \to \mathbb{R}$, $\mathbb{E}_{\pi^{\lambda}}(f) \approx \mathbb{E}_{\pi}(f)$. The benefit of the first inequality is that we can easily verify the right-hand side numerically and thus get an error bound estimate.

Theorem 4. *Let $g = -\log \pi$ be the negative logarithm of a probability density function, with g being a proper lower semi-continuous convex function, and L-Lipschitz. Let g^{λ} be the Moreau-Yosida envelope to g, and let $\pi^{\lambda}(x) = \exp(-g^{\lambda}(x))/(\int \exp(-g^{\lambda}(z))dz)$ be a probability*

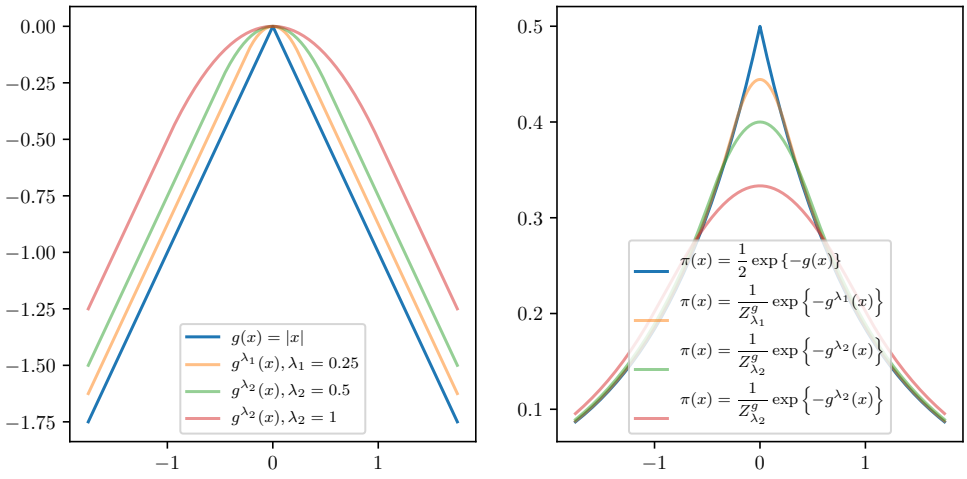


Fig. 3.1 Left: Plot of the negative log-density (i.e. potential) of the Laplace distribution (blue), and three MYE envelopes with different tightness parameters $\lambda \in \{0.25, 0.5, 1\}$. Right: The densities corresponding to the potentials in the left plot. The normalising constant of the MYE-adjusted density is $Z^g_\lambda = \int_{\mathbb{R}} \exp\{-g^\lambda(x)\}dx$.

density function. Then for any π- and π^λ-integrable $f : \mathcal{X} \to \mathbb{R}$:

$$|\mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f)| \leq (\exp(L^2\lambda) - 1)\mathbb{E}_{\pi^\lambda}(|f|) \tag{3.5}$$

$$|\mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f)| \leq (\exp(L^2\lambda) - 1)\mathbb{E}_\pi(|f|). \tag{3.6}$$

The same inequalities hold if $g = g_1 + g_2$ with a convex and Lipschitz-continuous g_1 and a differentiable (but not necessarily Lipschitz-continuous) g_2: in that case, one takes the MYE of g_1 only, resulting in the approximate pdf $\pi^\lambda(x) = \exp(-g_1^\lambda(x) - g_2(x))/(\int \exp(-g_1^\lambda(z) - g_2(z))dz)$.

Proof. See Supplementary Material 3.6.1. □

Choosing $f(x) = sgn(\pi^\lambda(x) - \pi(x))$ in this theorem allows to recover Proposition 3.1 in Durmus et al. (2018) by only considering a one-sided inequality in Equation 3.20 in the proof of the theorem, and can thus be viewed as a generalisation thereof. Theorem 4 shows that for exact integral estimates, one wants to pick λ as small as possible. This, however, does come at a cost, as the gradients of the MYE approximated target grow as $\lambda \to 0$ such that one needs to take a smaller step size in the diffusion algorithms discussed in the next section.

The relation between different approximation parameters and the size of the gradient is given by the next lemma.

Lemma 1. *Let $g : \mathcal{X} \to]-\infty, \infty]$ be a proper lower semi-continuous convex function, and let $0 < \lambda_1 \leq \lambda_2$; then for the corresponding Moreau-Yosida envelopes g^{λ_1} and g^{λ_2}, we have*

$$\|\nabla g^{\lambda_1}(x)\| \geq \|\nabla g^{\lambda_2}(x)\| \quad \forall x \in \mathcal{X}.$$

Proof. See Supplementary Material 3.6.2. □

3.3 Sampling Algorithms

Due to the non-differentiability of the target distribution, gradient-based methods cannot be applied in a straightforward fashion. One should, however, exploit the almost everywhere differentiability. This can be done in two ways.

If the target is log-concave (and thus almost everywhere differentiable), one can define the approximate target π^λ and sample from it using any algorithm discussed in Chapter 2. In particular, the next two subsections define diffusion-based samplers targeting the approximate π^λ. For the first one, a Metropolis-Hastings correction step can be added to correct for the discretisation and approximation error.

Another approach, that works for any almost everywhere differentiable target distribution, is to use PDMPs as introduced in Section 2.4. As shown in Goldman et al. (2020), they do not require global differentiability: one defines an extended gradient, which is the normal gradient where it is defined, and 0 on the nullset $A = \{x : \pi(x) \text{ is not differentiable}\}$. Crucially, the proof to show that the target is a stationary measure under the PDMP requires integrating the gradient with respect to the target, and changing it on a nullset does not affect the calculations.

3.3.1 Overdamped Langevin Samplers

For a general non-differentiable distribution π one may, as proposed in Pereyra (2016) and further studied in Durmus et al. (2018), target the MYE-smoothed version π^λ instead. In particular, the gradient of (2.8) is split into a likelihood derivative and an evaluation of the

proximal operator via Theorem 2:

$$\nabla \log \pi^{\lambda}(x) = \nabla \ell(x) + \frac{1}{\lambda}\Big(x - \operatorname{prox}^{\lambda}_{\log \pi_0}(x)\Big) =: -\nabla U^{\lambda}(x). \tag{3.7}$$

The resulting sampler is known as proximal MALA (pMALA) or MY-ULA, depending on whether or not a Metropolis-Hastings step is included.

If using the unadjusted version, the step-size is not forced to comply with theoretically-optimal acceptance rates (Roberts and Rosenthal, 1998), rather, the step size is chosen to be of the order of λ (Durmus et al., 2018).

As in Section 2.2.1, one can use a stablised explicit integrator instead of the classic Euler-Maruyama scheme, which has been studied by Pereyra et al. (2020) who use a stochastic second kind orthogonal Runge-Kutta-Chebyshev method (SK-ROCK) proposed in Abdulle et al. (2018). This boils down to using U^{λ} instead of U in Algorithm 2.

3.3.2 An Underdamped Langevin Sampler

A natural extension of the MY-ULA algorithm is to use the more elaborate dynamics of the underdamped Langevin SDE (2.9) to explore the smoothed target distribution π^{λ}, where the smoothed potential U^{λ} is used in the discretisation (2.11).

As with MY-ULA, the convergence of a discretisation of these dynamics depends on the Lipschitz constant of the gradient Cheng et al. (2017), which implies that the trade-off between posterior accuracy and mixing speed remains with MY-UULA. In spite of this, the improved dimensional scaling of MY-UULA can provide improvements over MY-ULA, see e.g. Examples 3.4.1 and 3.4.2.

All the experiments in this thesis were, unless stated otherwise, run with $\gamma = 2$, $L = 1/\lambda$, and $\nu = \lambda/2$, where λ is the tightness parameter of the respective MYE.

3.4 Numerics

In this section, we compare the various methods in multiple numerical examples. The goal is to give practitioners guidance as to when to use which algorithm, which we discuss in the subsequent Section 3.5. In most experiments, we compare the performance of the Moreau-Yosida Unadjusted Langevin Algorithm (MY-ULA, Durmus et al. (2018)), the new

Moreau-Yosida Unadjusted *Underdamped* Langevin Algorithm (MY-UULA), the stochastic second kind orthogonal Runge-Kutta-Chebyshev method (SK-ROCK, Abdulle et al. (2018); Pereyra et al. (2020)), the proximal Metropolis-Adjusted Langevin Algorithm (pMALA, Pereyra (2016)), the Bouncy-Particle Sampler (BPS, Bouchard-Côté et al. (2018)), and the Zig-Zag Sampler (ZZS, Bierkens et al. (2019a)). The rates for the BPS and ZZS are computed using the thinning method described in Section 2.4, even though they are exactly computable in the first examples. We believe this is a realistic and fair comparison, as the findings generalise to more complicated setups where exact rates are not available.

In Example 3.4.7 the MYE-based methods are not applicable, and we instead compare the BPS, the ZZS, and a Random Walk Metropolis-Hastings algorithm (RWMH).

The code for the first four experiments is available online at https://github.com/TorbenSell/anisotropic-isotropic-laplace-gaussian.

3.4.1 Isotropic Laplace Distribution

A first simple example is a three-dimensional, isotropic, centered Laplace distribution, which has the density

$$\pi(x) = \prod_{i=1}^{3} \frac{1}{2\beta_i} \exp\left(-\frac{|x|}{\beta_i}\right), \tag{3.8}$$

with $\beta_i = 1$ for $i \in \{1,2,3\}$. For the MYE-based algorithms, we set $\lambda = 1/4$, resulting in fairly good approximations, see the first three rows in Figure 3.2. The tuning parameters were chosen to be $\delta = \lambda/4$ for MY-ULA, as in Section 3.3.2 for MY-UULA, $\delta = 0.012$ and $s = 15$ for SK-ROCK, and $\delta = 2\lambda = 1.5$ for pMALA, giving an acceptance rate of 65%. The effective sample sizes per second (ESS/s) are summarised in Table 3.1, and the estimated marginals within 120 seconds are shown in Figure 3.2.

It is worth noting that, in this isotropic example, the BPS and ZZS cannot compete with pMALA. This is presumably due to the slightly more complicated implementation of the PDMPs which requires the simulation of event times from an inhomogeneous Poisson process, as discussed in Section 2.4. Furthermore, the additional complexity of SK-ROCK does not yield an advantage over the naïve discretisations of the Langevin equations here, indeed a large s on this target results in effectively wasted computation time. This remains true for the next three experiments: SK-ROCK is generally strong on high-dimensional

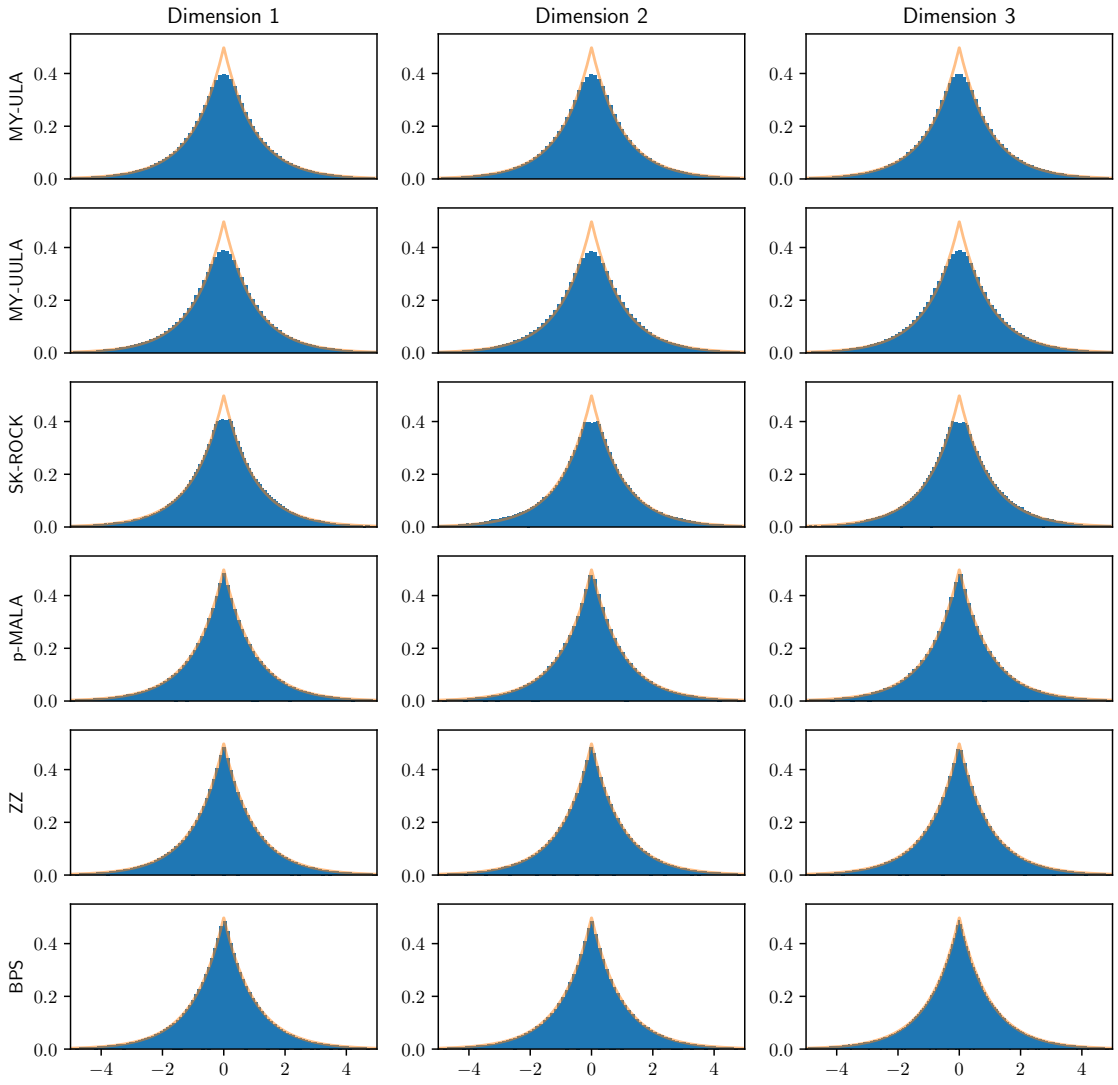


Fig. 3.2 All algorithms are targeting a three-dimensional isotropic Laplace distribution. All algorithms were given the same computational budget of 120 seconds for a fair comparison.

Algorithm	MY-ULA	MY-UULA	SK-ROCK	pMALA	BPS	ZZS
$\beta_1 = 1$	233.93	448.61	13.29	2148.79	293.35	447.52
$\beta_2 = 1$	223.90	387.47	19.14	2034.02	252.21	513.01
$\beta_3 = 1$	231.15	402.24	18.59	1896.67	239.32	468.11

Table 3.1 Effective sample size per second for the different algorithm when targeting an isotropic Laplace distribution. Recall that the first three algorithms are asymptotically biased, while the last three are asymptotically exact. Due to the isotropy, we expect the ESS/s to be similar across dimensions.

models, in which MY-ULA and MY-UULA require very small step sizes (to control the discretisation error from the respective Langevin equations), and SK-ROCK allows for bigger step sizes while controlling the error at a similar rate. pMALA will also perform badly in high-dimensional examples as the step size will decrease with dimension to ensure proposals are accepted with roughly the optimal rate. These experiments should thus not be interpreted as evidence that SK-ROCK is bad, but rather showing the relative benefit of PDMPs on anisotropic targets; as they highlight the difficulties that diffusion algorithms face when targeting such distributions: the step sizes will need to be scaled to the most narrow marginal, leading to bad mixing behaviour in the wide marginals.

3.4.2 Anisotropic Laplace Distribution

As a more complicated example, we now change the setup to an anisotropic Laplace distribution, where we set $\beta_i \in \{1, 0.1, 0.01\}$ in 3.8. For the MYE-based algorithms, we set $\lambda = 10^{-4}$, resulting in fairly good approximations even in the narrowest marginal, see the first rows in Figure 3.3. The tuning parameters were chosen to be $\delta = \lambda/4$ for MY-ULA, as in Section 3.3.2 for MY-UULA, $\delta = 0.0012$ and $s = 15$ for SK-ROCK, and $\delta = 2\lambda = 0.00034$ for pMALA, giving an acceptance rate of 65%. The ESS/s are summarised in Table 3.2, and the estimated marginals within 500 seconds are shown in Figure 3.3. Even after this relatively long runtime, the histogram estimates from the diffusion-based samplers are not perfectly aligning with the first marginal density.

Algorithm	MY-ULA	MY-UULA	SK-ROCK	pMALA	BPS	ZZS
$\beta_1 = 1$	7.81	6.64	2.47	4.81	2.73	22.06
$\beta_2 = 0.1$	14.19	18.45	142.52	48.70	73.77	193.39
$\beta_3 = 0.01$	943.88	1452.83	2578.15	4585.26	764.70	1480.38

Table 3.2 Effective sample size per second for the different algorithm when targeting an anisotropic Laplace distribution. Recall that the first three algorithms are asymptotically biased, while the last three are asymptotically exact.

In this setting, the marginal corresponding to $\beta_1 = 1$ is the slowest mixing one, and the ZZS outperforms all other samplers measured by the ESS/s. This a nicely illustrating the strong performance of the ZZS in examples where dimensions are independent. SK-ROCK can also be chosen to be run with a larger step size, in which case the ESS/s will increase at the cost of the histogram estimate to the third marginal deviating from the truth. For

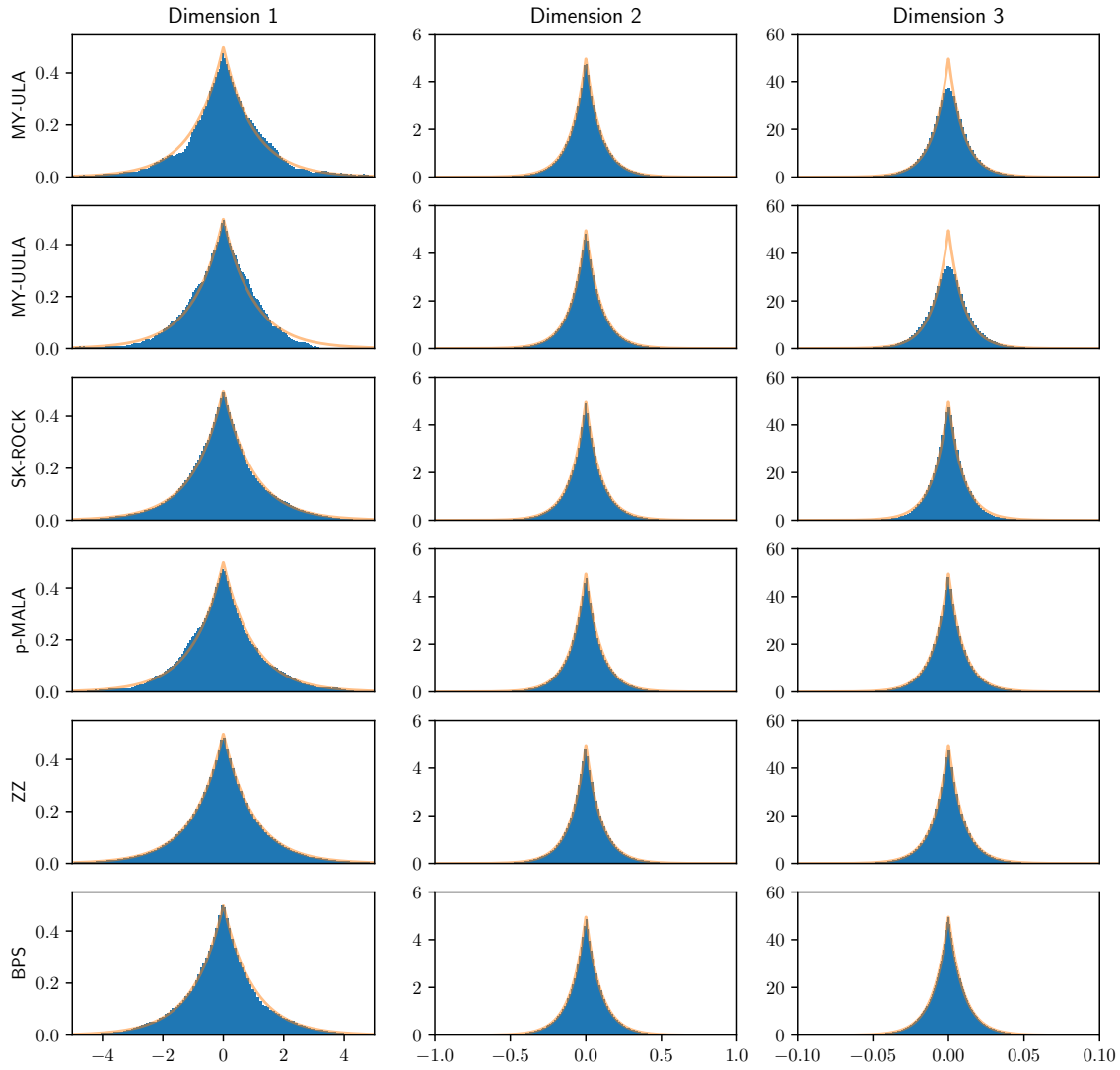


Fig. 3.3 All algorithms are targeting a three-dimensional anisotropic Laplace distribution. The first three rows correspond to the approximate algorithms, and none of them manage to fully capture the narrowest component. The ZZS perfectly captures the last component, and shows good results in the first component. The BPS (in its global form) mixes slowly in the first component, but well in the last. All algorithms were given the same computational budget of 500 seconds for a fair comparison.

MY-UULA, increasing the step size too much will result in a bimodal target, emphasising the importance of strong log-concavity for this method to work well.

3.4.3 Isotropic Gaussian Distribution

The Laplace distribution used in the previous experiments is *weakly* log-concave. To analyse the behaviour of the different algorithms in *strongly* log-concave examples, we repeat the previous two experiments with centered Gaussian distributions. While the Gaussian distribution is everywhere differentiable, we pretend it has a point of non-differentiability somewhere, and accordingly use the same algorithms as before.

In the first setup, we choose the identity matrix as the covariance matrix, and set $\lambda = 0.1$, which gives good approximations, see Figure 3.4. MY-ULA is run with $\delta = \lambda/4$, MY-UULA as described in Section 3.3 but with $\nu = 2\lambda$, SK-ROCK with $\delta = 0.048$ and $s = 15$, and pMALA works with $\delta = 2\lambda = 1$ giving 73% acceptances. Table 3.3 summarises the ESS/s for the different algorithms, and Figure 3.4 the visual results after 120 seconds runtime.

Algorithm	MY-ULA	MY-UULA	SK-ROCK	pMALA	BPS	ZZS
$\sigma_1 = 1$	331.06	1110.52	123.69	2856.59	894.64	1343.55
$\sigma_2 = 1$	339.86	1060.05	115.34	2879.52	969.02	1267.24
$\sigma_3 = 1$	333.64	1122.32	143.43	2802.90	1038.68	1383.02

Table 3.3 Effective sample size per second for the different algorithm when targeting a 3-dimensional isotropic Gaussian distribution. Recall that the first three algorithms are asymptotically biased, while the last three are asymptotically exact.

As for the isotropic Laplace example, the elaborate integration scheme used by SK-ROCK does not yield any benefit in the isotropic Gaussian example. Interestingly, all exact algorithms perform better than any approximate one. This is because the approximate algorithms need to be run with comparatively small step sizes to ensure they do not deviate too much from the desired target distribution. Running them with larger step sizes will increase the discretisation bias. We refer the reader to Durmus et al. (2018) for a discussion on how to choose both the envelope tightness and the step size of MY-ULA, and to Pereyra et al. (2020) for a discussion on how to choose the step size of SK-ROCK.

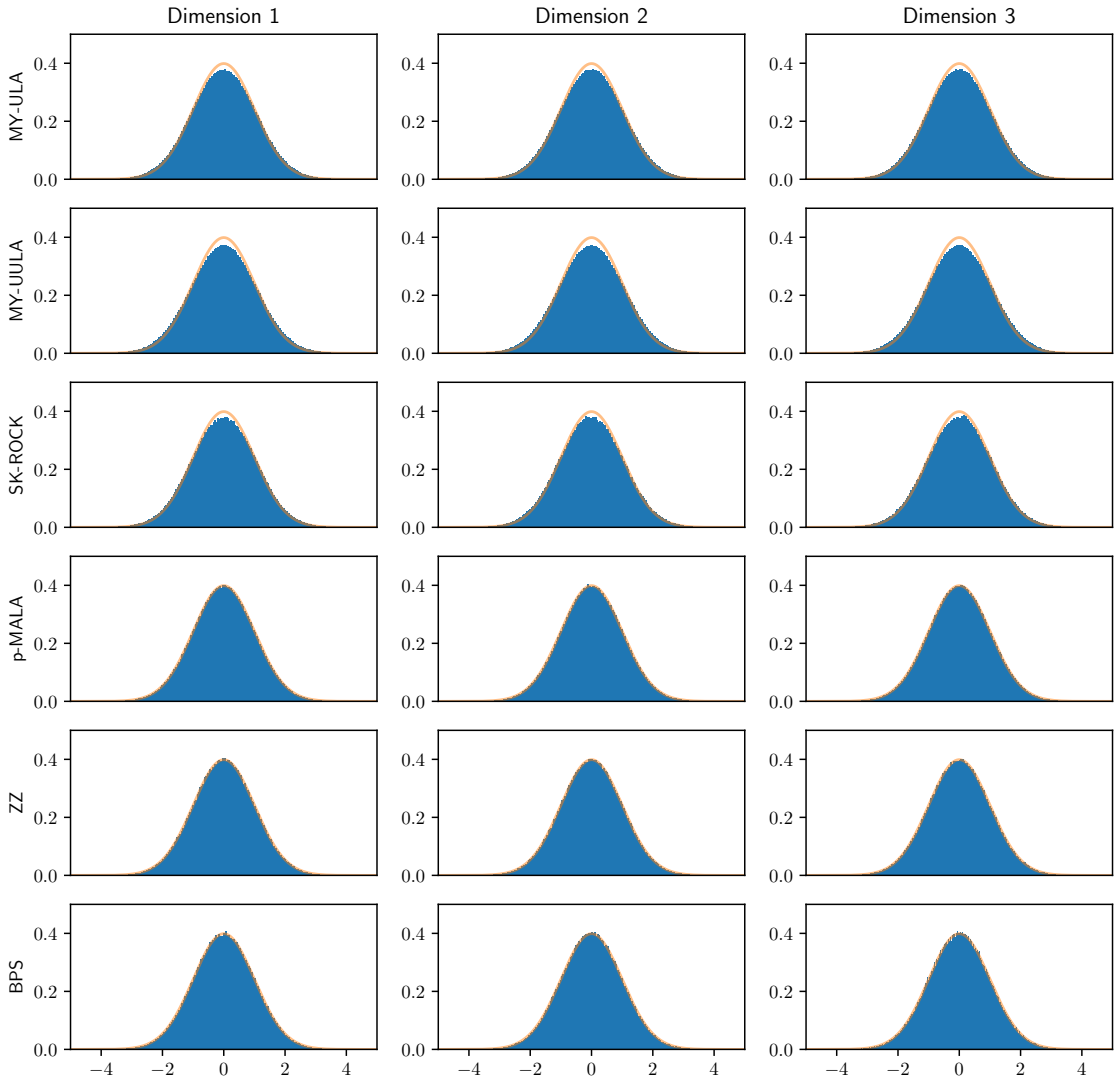


Fig. 3.4 All algorithms are targeting a three-dimensional isotropic Gaussian distribution. The first three rows correspond to the approximate algorithms. All algorithms were given the same computational budget of 100 seconds for a fair comparison.

3.4.4 Anisotropic Gaussian Distribution

To complete the first examples, we look at an anisotropic Gaussian distribution, choosing the three-dimensional diagonal covariance matrix to be such that $\Sigma_{i,i} = \sigma_i^2$, with $\sigma_i \in \{1, 0.1, 0.01\}$. The variances are chosen such that the level of anisotropy (measurable by looking at the ratio Var_{max}/Var_{min} which is 10^4 here) is the same as in Example 3.4.2. We chose $\lambda = 10^{-5}$, which gives good approximate histogram estimates, see Figure 3.5. We again set $\delta = \lambda/4$ for MY-ULA, the parameters for MY-UULA as in Section 3.3, and $\delta = 0.000048$ and $s = 15$ for SK-ROCK. For pMALA, we set $\delta = 2\lambda = 0.0003$, giving an acceptance probability of 65%. Estimates of the ESS/s are summarised in Table 3.4, the histogram estimates of the marginals as obtained by the different samplers are given in Figure 3.5.

Algorithm	MY-ULA	MY-UULA	SK-ROCK	pMALA	BPS	ZZS
$\sigma_1 = 1$	11.31	5.83	1.31	4.27	3.15	56.30
$\sigma_2 = 0.1$	13.15	15.58	14.64	93.49	161.44	570.76
$\sigma_3 = 0.01$	322.43	1242.65	1056.08	4395.51	1247.77	2657.70

Table 3.4 Effective sample size per second for the different algorithm when targeting an anisotropic Gaussian distribution. Recall that the first three algorithms are asymptotically biased, while the last three are asymptotically exact.

As in Example 3.4.2 on the anisotropic Laplace distribution, all algorithms mix slowest on the wide marginal, here corresponding to $\sigma_1 = 1$. The ZZS again performs best on this marginal in terms of the ESS/s.

3.4.5 Nuclear-Norm Models for Low-Rank Matrix Estimation

Another illustration of the methods' performance in exact sampling, we consider a nuclear-norm model example taken from Pereyra (2016). Let $x \in \mathbb{R}^d = \mathbb{R}^{n\times n}$ be an unknown low-rank matrix, and let observations be noisy measurements thereof: $y = x + \xi$, where the entries of ξ are i.i.d. $N(0, \sigma^2)$. We assume that x is a low-rank matrix, and our aim is to sample from the posterior distribution of x given by

$$\pi(x) \propto \exp\left(-\frac{1}{2\sigma^2}\|x-y\|_F - \alpha\|x\|_*\right), \tag{3.9}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\|\cdot\|_*$ denotes the nuclear norm which favours low-rank matrices and penalizes high-rank ones. Conveniently, the proximal operator of the

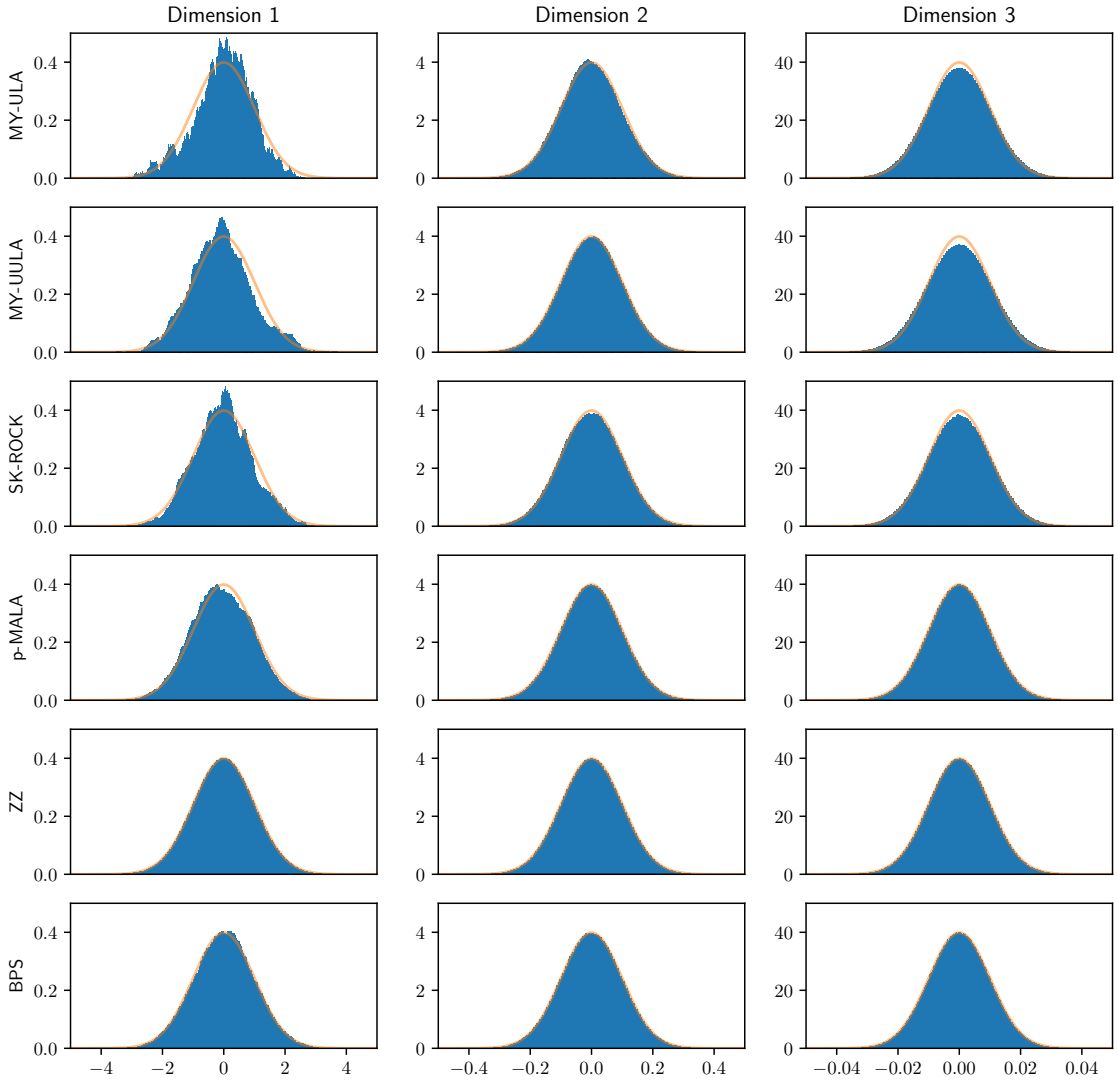


Fig. 3.5 All algorithms are targeting a three-dimensional anisotropic Gaussian distribution. The first three rows correspond to the approximate algorithms, and none of them manage to fully capture the narrowest component. The ZZS perfectly captures the last component, and shows good results in the first component. The BPS (in its global form) mixes slowly in the first component, but well in the last. The last four algorithm were given the same computational budget of 200 seconds for a fair comparison, MY-ULA and MY-UULA were given 600 seconds.

nuclear norm is available in closed form: Let $x = Q\Sigma V^T$ be the singular value decomposition of x, with $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_n)$. Then the proximal operator is given by

$$\mathrm{prox}^{\lambda}_{\alpha\|\cdot\|_*}(x) = Q\mathrm{diag}\left(\mathrm{sgn}(\sigma_1)\max(|\sigma_1| - \alpha\lambda, 0), \dots, \mathrm{sgn}(\sigma_n)\max(|\sigma_n| - \alpha\lambda, 0)\right) V^T,$$

i.e., one applies the soft thresholding operator to the singular values of x. We can thus efficiently compute the gradient to use in the Langevin-based samplers,

$$\nabla U^{\lambda}(x) = \frac{1}{\sigma^2}(x - y) + \frac{1}{\lambda}\Big(x - \mathrm{prox}^{\lambda}_{\alpha\|\cdot\|_*}(x)\Big). \tag{3.10}$$

We generated y by adding Gaussian noise to a matrix $x^{\mathrm{true}} \in \mathbb{R}^{64\times 64}$ with entries $x^{\mathrm{true}}_{i,j} \in \{0, 0.7, 1\}$. The matrix x^{true} is visually a checkerboard with white, grey, or black checks.

For SK-ROCK, MY-ULA, and MY-UULA, we set $\lambda = \sigma^2$. The step size for MY-ULA is set to $\delta = 2\lambda$, for SK-ROCK we set $s = 10$ and $\delta \approx 0.242$, and MY-UULA is tuned as described in Section 3.3.2. For pMALA setting $\delta/2 = \lambda = 0.000035$ gave around 65% acceptances. A particular issue for the BPS in this model is the lack of factor decomposition due both to non-linearity of the nuclear norm and the proximal operator, which prevents us from using a localised, and therefore faster, version of the BPS. In an attempt to mitigate the resulting debilitated dynamics, we note that the likelihood in this case is equivalent to a isotropic Gaussian distribution in x as well. Defining an auxiliary potential by $V(x|y) = \|x - y\|^2/2$, we propose to generate dynamics according to the Hamiltonian flow (see Section 2.4) corresponding to $(\dot{x}, \dot{v}) = (v_t, -(x_t - y)/\sigma^2)$, which has the explicit solution

$$\begin{pmatrix} x_t \\ v_t \end{pmatrix} = \begin{pmatrix} v_0 \sin\left(\frac{t}{\sigma}\right)\sigma + (x_0 - y)\cos\left(\frac{t}{\sigma}\right) + y \\ -(x_0 - y)\sin\left(\frac{t}{\sigma}\right) + v_0 \cos\left(\frac{t}{\sigma}\right) \end{pmatrix}.$$

By this choice of V it follows that the gradient employed in the rate and reflection operator subsequently is

$$\nabla \hat{U}^{\lambda}(x) = \frac{1}{\lambda}\Big(x - \mathrm{prox}^{\lambda}_{\alpha\|\cdot\|_*}(x)\Big). \tag{3.11}$$

Figure 3.6 shows the mean squared error between the posterior mean estimate of the respective algorithms, as calculated every second, and the 'true' posterior mean, as estimated by a very long run using an asymptotically unbiased algorithm. All algorithms are started at the same point, not too far away from the region of high probability. One can see that while MY-ULA quickly gives good estimates, the second-order scheme MY-UULA quickly yields

better estimates. Interestingly, SK-ROCK performs worse here. The BPS does not yield any useful estimates in reasonable time, but after a while the HBPS gives the second best results. For completeness, we note that the Zig-Zag Sampler is not able to computationally compete with any of the other methods, as a single reflection requires the evaluation of the full gradient, which is prohibitively expensive. We also estimated the slowest and fastest mixing components of the checkerboard by estimating the sample covariance matrix during a long run of an exact sampler, and taking the first and last eigenvector thereof as the direction where the chain mixes slowest, and fastest, respectively. The autocorrelation plots for these components are shown in the second and third panel of Figure 3.6.

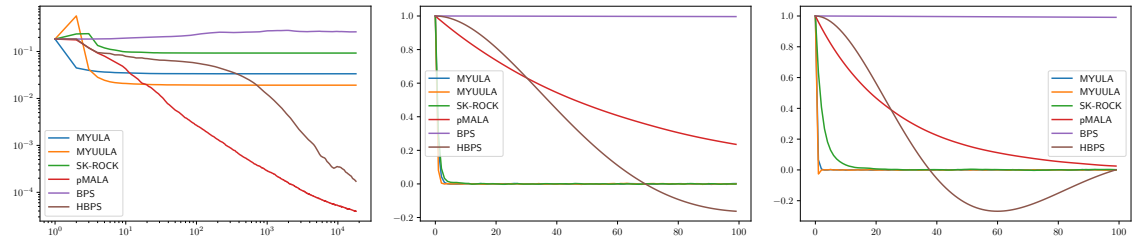


Fig. 3.6 Results from the nuclear norm example. Left: MSE over time, for the different algorithms, run for half an hour each, on a log-log-scale. Middle: Autocorrelation for the slowest component, number of samples adjusted for a fair comparison. Right: Autocorrelation for the fastest component, number of samples adjusted for a fair comparison.

3.4.6 Image Deblurring

Uncertainty quantification in images is generally a challenging computational problem, with samples from the posterior used to estimate credible intervals or provide model comparisons. We focus on a purely illustrative example involving the total variation prior similar to Example 4.1.2 in Durmus et al. (2018). Let $x \in \mathbb{R}^d = \mathbb{R}^{d_1 \times d_2}$ be an image which we observe through $y = Hx + \xi$, where H is a blurring operator that blurs a pixel $x_{i,j}$ uniformly with its closest neighbours (5×5 patch), and $\xi \sim N(0, \sigma^2 I_{d_1 \times d_2})$. The log-prior is proportional to $-TV(x) = -\alpha \|\nabla_D x\|_1$, where ∇_D is the two-dimensional discrete gradient operator as defined in Chambolle (2004), and α is a fixed parameter. The application of the TV prior is common in a wide array of imaging applications, as it emphasises smooth surfaces bounded by distinct edges. As the authors of Durmus et al. (2018) we chose the 256×256 'boat' test image, and

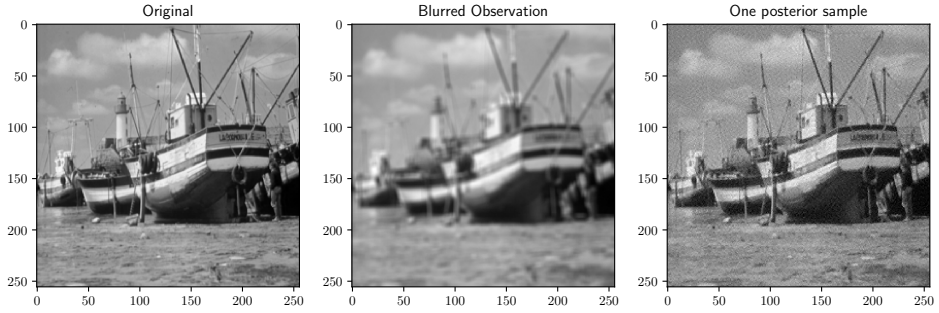


Fig. 3.7 Left: The original 256×256 image. Center: The image after the application of the uniform blur operator. Right: A representative sample from the posterior distribution given in equation (3.12), obtained using the LBPS.

set $\alpha = 0.03$, $\sigma = 0.47$. The posterior is given by

$$\pi(x) \propto \exp\left(-\frac{1}{2\sigma^2}\|Hx - y\|_2^2 - \alpha TV(x)\right). \tag{3.12}$$

The TV-prior decomposes into a sum where each entry only depends on neighbouring points; the uniform blur operator is similarly local. This implies in combination that the posterior can be factorised at granularities defined by the user, and we can therefore apply the local BPS. We stress that the global BPS struggles in high dimensions (Deligiannidis et al., 2018), and thus localisation is necessary for it to be a competitive algorithm in these settings. The proximal operator is not available in closed form for the TV-prior, and hence requires evaluation via numerical schemes such as the Douglas-Rachford algorithm (Lions and Mercier, 1979) or the Chambolle-Pock algorithm (Chambolle and Pock, 2011). While these algorithms in general are efficient, they slow down significantly as the precision of the envelope is increased.

We compare the performances of the LBPS, the ZZS, pMALA, MY-ULA, MY-UULA, and SK-ROCK. For both the LBPS and the ZZS we estimated bounds on the prior- and likelihood-gradients to get constant upper bounds on the respective rates, we then used these constant bounds to generate computationally cheap events, avoiding any global evaluations of the gradient: these bounds were set to 4.06 for the ZZS and to 9.44 for the LBPS (with 4×4 blocks). For the ZZS, the evaluated rates never exceeded this bound, for the LBPS they did in a negligible 0.217% of events. For pMALA, we set $\lambda = 2\delta = 0.006$, giving us an acceptance ratio of 67%. For the last three samplers, we chose $\lambda = 0.45$ following the

guidance in Durmus et al. (2018), and set $\delta = 0.9$ and $s = 10$ for SK-ROCK. The goal is to sample from the posterior distribution when observing a blurred image, see Figure 3.7. Figure 3.8 shows the mean squared error (MSE) and the structural similarity index (SSIM) between the mean estimates of the various algorithms and the 'true' mean, as estimated by a long run of an asymptotically exact algorithm. Notably, unlike MY-(U)ULA, pMALA, and SK-ROCK, which require the evaluation of the proximal operator (which is not localisable), the LBPS and ZZS can be sped up using parallelisation techniques: the implementation we used applied global rates to avoid recalculating the full posterior gradient after every event, but one may calculate the factor gradients at hardly any extra computational cost if one calculates them in parallel.

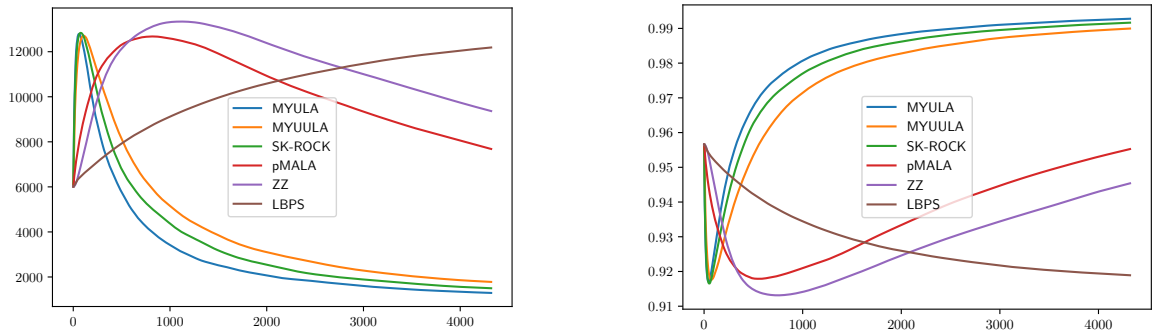


Fig. 3.8 Results from the Image Deblurring example. Left: The MSE of the mean estimates, estimated every 10 seconds. Right: The SSIM of the mean estimates, estimated every 10 seconds.

3.4.7 Circular Bayesian Statistics

Circular statistics addresses inference from periodic data such as angles and rotations, for example when analysing the direction ants move in when responding to an evenly illuminated black target Jander (1957), this can be modelled by an asymmetric wrapped Laplace distribution Fernández-Durán (2004). This example illustrates the applicability of PDMPs when dealing with circular, non-differentiable, and multi-modal posteriors. Here, the MYE is not well defined, precluding the use of the samplers based on the MYE-approximation.

In Jander (1957), the author studies which direction ants walk in when being placed in the middle of an evenly illuminated arena with two black discs on the side. These directions were observed for 253 ants, the observations are summarised in Figure 3.9. For the one-disc example, Fernández-Durán (2004) shows that the wrapped Laplace distribution is a good

fit to the observations. For the two-disc model, we thus assume a mixture of two wrapped Laplace distributions with means $\mu_i \in [0, 2\pi)$, scale parameters $\lambda_i > 0$, and skew parameters $\kappa_i > 0$, $i \in \{1, 2\}$. The likelihood of a data point $y \in [0, 2\pi)$ given the mixture component is calculated by

$$\theta = \begin{cases} y - \mu_i, & \text{for } y - \mu_i > 0 \\ y - \mu_i + 2\pi, & \text{for } y - \mu_i \leq 0 \end{cases}$$
$$L(\theta|\mu_i, \lambda_i, \kappa_i) = \frac{\lambda_i \kappa_i}{1 + \kappa_i^2} \left(\frac{e^{-\lambda_i \kappa_i \theta}}{1 - e^{-2\pi\lambda_i\kappa_i}} + \frac{e^{(\lambda_i/\kappa_i)\theta}}{e^{2\pi(\lambda_i/\kappa_i)} - 1} \right),$$

where the definition of the auxiliary variable θ handles the periodic extension due to the shift by the mean. The prior on the μ_i are uniform distributions on $[0, 2\pi]$, the ones on the scale parameters λ_i are Exponential(1) distributions, the one on the κ_i are Gamma$(2, 1/2)$ distributions, and the one on the mixture parameter ρ is a Beta$(100, 100)$ distribution.

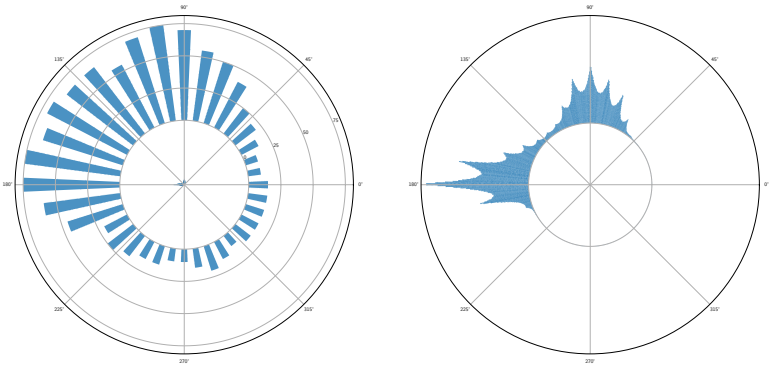


Fig. 3.9 Left: Observations, taken from (Jander, 1957, Fig. 18B). Right: Marginal posterior density for the mean parameter μ_1.

Figure 3.9 shows the posterior distribution for μ_1 as estimated by the BPS. The reader should in particular note the multi modality of the distribution, which especially the BPS samples from effectively: Table 3.5 summarises the effective sample size per second for the BPS, the ZZS, and a Random Walk Metropolis Hastings Sampler.

3.5 Discussion

Both PDMPs and diffusion-based samplers have been shown to work in various examples, albeit differently well. This discussion aims to provide the reader with an understanding of

Algorithm	μ_1	λ_1	κ_1	ρ
BPS	3.36	164.80	6.29	2095.44
ZZS	0.64	20.90	1.12	655.00
RWMH	2.88	37.01	4.52	1153.13

Table 3.5 Effective sample size per second for different variables. The ESS/s for the variables from the second mixture are similar, as is expected due to the mixture components being indistinguishable from one another.

the strengths and weaknesses of the respective methods, and to guide the practitioner as to which algorithm to use. In the following, we discuss the key aspects one needs to consider.

Dimensionality: PDMPs perform worse as the dimension increases: the ZZS requires at least d events to completely change the direction of the velocity vector, and for each of these event the gradient in the respective direction needs to be re-evaluated, which (depending on the problem) can become prohibitively expensive. The BPS is known to suffer in high-dimensions too, with problems arising even when targeting isotropic Gaussians (Bouchard-Côté et al., 2018) requiring many refreshments. If the problem is localisable (such as in Example 3.4.6), the local version of the BPS can alleviate these issues by reducing the problem size to multiple smaller problems. An interesting direction for future research would be to parallelise the dynamics of these smaller problems, which would be a strong argument for PDMPs in high-dimensional settings, as discussed in Example 3.4.6. A first step in this direction is to compute the gradients of the different factors in parallel which does not impact the correctness of the algorithm: multiple workers calculate the local gradients in parallel and pass them on to a master process, this master process then draws the exponential variables using the calculated gradient and executes the respective PDMP jump. A second step is in the spirit of Chapter 4, which would allow workers to work on different factors in parallel; this requires further assumptions and a more careful analysis, as the factors interact with each other, prohibiting naive parallelisation. The smoothing by the MYE always results in a non-localisable target, such that the computation of the proximal operator cannot be broken down into smaller problems. Furthermore, in the large data regime, data subsampling is straightforward for the PDMP samplers.

Anisotropic targets: As illustrated in Example 3.4.2, and similarly in (the everywhere differentiable) Example 3.4.4, especially the ZZS is able to adapt to highly anisotropic targets. The diffusion-based samplers would improve if one has a preconditioner available, but if this is not the case, they struggle: a tight envelope results in small step sizes such that mixing

in the 'slowly mixing components' takes very long. However, if one chooses a rather crude approximation, the approximation error in the 'fast mixing components' grows. This is graphically visible in Figures 3.3 and 3.5.

Log-concavity: As illustrated in Example 3.4.7, PDMPs allow targeting any almost everywhere differentiable posterior, while the diffusion-based algorithms rely on the MYE to be well defined which it only is for log-concave targets. In strongly log-concave settings such as Examples 3.4.3 and 3.4.4 when the almost everywhere gradient of the target is not globally Lipschitz continuous[1], a very tight envelope is needed to ensure a good approximation: this however, results in small step sizes and thus slow mixing. To alleviate this problem, one should try to split the posterior potential g into a non-differentiable part g_1, which has almost everywhere Lipschitz-continuous gradients, and a differentiable part g_2, which may not have a globally Lipschitz-continuous gradient. One then forms the MYE only over the first part, and this is indeed the strategy implemented in Examples 3.4.6 and 3.4.5, for the latter see Equation (3.10). An error bound for this split is stated in Theorem 4.
In weakly log-concave settings, the samplers based on the overdamped Langevin diffusion struggle as the gradients do not grow when $|x|$ diverges. In high-dimensional examples this seemed to be less important, as the dimensionality becomes the decisive factor of efficiency. In the absence of any log-concavity, such that the MYE of the target is not well defined, PDMPs are a viable option.

Exact Sampling: The PDMPs are inherently asymptotically exact samplers, while the unadjusted diffusion-based samplers are not. However, it is possible to incorporate a Metropolis-Hastings correction step in MY-ULA to account for this, giving the proximal Metropolis-Adjusted Langevin Algorithm (pMALA, Pereyra (2016)). pMALA is recommended to be tuned such that one achieves around 50% - 70% acceptances, which may result in slower mixing in comparison to the unadjusted algorithms, and generally suffers in the same settings as these algorithms, e.g. on very anisotropic target distributions, see Examples 3.4.2 and 3.4.4.

Proximal Operators and Event Rates: The proximal operators for some functions are available in closed form (see e.g. Polson et al. (2015)), however often the minimisation problems needs to be solved numerically, in which case the evaluation becomes expensive. This is the case in Example 3.4.6, but the high dimension of the example was a more important factor in terms of efficiency of the different samplers.

[1]Note that strong log-concavity implies the existence of a dominating quadratic, such that no Lipschitz constant can globally exist.

On the other end, the PDMPs will perform significantly worse if one cannot find good bound on the event rates, in which case one needs to evaluate the gradients more often.

This chapter showed that sampling algorithms based on PDMPs allow exact sampling from non-differentiable target distributions. In particular, gradients were exploited if they exist almost everywhere, which is a common scenario in many real-world applications using non-differentiable priors. This has particular relevance in cases like cancer tumor classification (Golub et al., 1999) where accuracy is at a premium (Goldman et al., 2020), and in situations where the proximal operator is prohibitively expensive to calculate. In comparison to gradient-based methods, PDMPs can naturally handle anisotropy of the posterior without preconditioning, and furthermore can be localised whenever the distribution is of product-form. In contrast, the unadjusted diffusion algorithms perform well in cases where low accuracy of the envelope is acceptable and when the posterior cannot be localised. Utilising second-order information in the MY-UULA algorithm can prove beneficial to mixing, but finding clear guidance as to when they are preferable remains an open problem. Furthermore, we have illustrated efficient sampling of non-convex posteriors where Moreau-Yosida methods are not applicable. In conclusion, this chapter has shown that PDMPs are a very able tool whenever accurate posterior inference is required in complex non-differentiable scenarios, and discussed which distribution characteristics suggest one or the other class of algorithms to be preferable.

3.6 Appendix

3.6.1 Proof of Theorem 4

We now prove Theorem 4 which stated the following:

Theorem. *Let $g = -\log\pi$ be the negative logarithm of a probability density function, with g being a proper lower semi-continuous convex function, and L-Lipschitz. Let g^λ be the Moreau-Yosida envelope to g, and let $\pi^\lambda(x) = \exp(-g^\lambda(x))/(\int \exp(-g^\lambda(z))dz)$ be a probability density function. Then for any π- and π^λ-integrable $f : \mathcal{X} \to \mathbb{R}$:*

$$|\mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f)| \leq (\exp(L^2\lambda) - 1)\mathbb{E}_{\pi^\lambda}(|f|) \tag{3.13}$$

$$|\mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f)| \leq (\exp(L^2\lambda) - 1)\mathbb{E}_\pi(|f|). \tag{3.14}$$

The same inequalities hold if $g = g_1 + g_2$ with a convex and Lipschitz-continuous g_1 and a differentiable (but not necessarily Lipschitz-continuous) g_2: in that case, one takes the MYE of g_1 only, resulting in the approximate pdf $\pi^\lambda(x) = \exp(-g_1^\lambda(x) - g_2(x))/(\int \exp(-g_1^\lambda(z) - g_2(z))dz)$.

Proof of Theorem 4. From Theorem 3 we immediately get the inequalities

$$-g(x) \leq -g^\lambda(x) \tag{3.15}$$

$$-g^\lambda(x) \leq -g(x) + \frac{L^2\lambda}{2}, \tag{3.16}$$

and thus also

$$\int \exp(-g^\lambda(z))dz \geq \int \exp(-g(z))dz = 1 \tag{3.17}$$

and

$$\int \exp(-g^\lambda(z))dz \leq \int \exp(-g(z))\exp(L^2\lambda/2)dz = \exp(L^2\lambda/2). \tag{3.18}$$

Let $f \geq 0$, then

$$\begin{aligned}
\mathbb{E}_{\pi^\lambda}(f) &= \int f(x)\frac{\exp(-g^\lambda(x))}{\int \exp(-g^\lambda(z))dz}dx \\
&\overset{(3.17)}{\leq} \int f(x)\exp(-g^\lambda(x))dx \\
&\overset{(3.16)}{\leq} \exp(L^2\lambda/2)\int f(x)\exp(-g(x))dx = \exp(L^2\lambda/2)\mathbb{E}_\pi(f).
\end{aligned}$$

Similarly, again for $f \geq 0$,

$$\begin{aligned}
\mathbb{E}_{\pi^\lambda}(f) &= \int f(x)\frac{\exp(-g^\lambda(x))}{\int \exp(-g^\lambda(z))dz}dx \\
&\overset{(3.18)}{\geq} \exp(-L^2\lambda/2)\int f(x)\exp(-g^\lambda(x))dx \\
&\overset{(3.15)}{\geq} \exp(-L^2\lambda/2)\int f(x)\exp(-g(x))dx \\
&= \exp(-L^2\lambda/2)\mathbb{E}_\pi(f).
\end{aligned}$$

In summary, for any non-negative f, we have

$$\exp(-L^2\lambda/2)\mathbb{E}_\pi(f) \le \mathbb{E}_{\pi^\lambda}(f) \le \exp(L^2\lambda/2)\mathbb{E}_\pi(f). \tag{3.19}$$

Subtracting $\mathbb{E}_\pi(f) \ge 0$ from these inequalities lets us derive

$$\begin{aligned}
-(\exp(L^2\lambda/2)-1)\mathbb{E}_\pi(f) &= -\max\{\exp(L^2\lambda/2)-1, 1-\exp(-L^2\lambda/2)\}\mathbb{E}_\pi(f) \\
&= \min\{1-\exp(L^2\lambda/2), \exp(-L^2\lambda/2)-1\}\mathbb{E}_\pi(f) \\
&\le (\exp(-L^2\lambda/2)-1)\mathbb{E}_\pi(f) \\
&\overset{(3.19)}{\le} \mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f) \\
&\overset{(3.19)}{\le} (\exp(L^2\lambda/2)-1)\mathbb{E}_\pi(f) \\
&\le \max\{\exp(L^2\lambda/2)-1, 1-\exp(-L^2\lambda/2)\}\mathbb{E}_\pi(f) \\
&= (\exp(L^2\lambda/2)-1)\mathbb{E}_\pi(f),
\end{aligned} \tag{3.20}$$

and therefore

$$|\mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f)| \le (\exp(L^2\lambda)-1)\mathbb{E}_\pi(f) \tag{3.21}$$

holds for any non-negative f.

For general f, we consider the standard decomposition $f = f^+ - f^-$ with $f^+ \ge 0$ and $f^- \ge 0$. Then $|f| = f^+ + f^-$, and as

$$\begin{aligned}
|\mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f)| &= |\mathbb{E}_{\pi^\lambda}(f^+) - \mathbb{E}_\pi(f^+) - [\mathbb{E}_{\pi^\lambda}(f^-) - \mathbb{E}_\pi(f^-)]| \\
&\le |\mathbb{E}_{\pi^\lambda}(f^+) - \mathbb{E}_\pi(f^+)| + |\mathbb{E}_{\pi^\lambda}(f^-) - \mathbb{E}_\pi(f^-)| \\
&\overset{(3.21)}{\le} (\exp(L^2\lambda)-1)\mathbb{E}_\pi(f^+) + (\exp(L^2\lambda)-1)\mathbb{E}_\pi(f^-) \\
&= (\exp(L^2\lambda)-1)\mathbb{E}_\pi(|f|),
\end{aligned}$$

we have proved the first bound in the Theorem.

Since we can exchange the roles of π and π_λ in (3.19), we can follow the same chain of arguments to also get

$$|\mathbb{E}_{\pi^\lambda}(f) - \mathbb{E}_\pi(f)| \le (\exp(L^2\lambda/2)-1)\mathbb{E}_{\pi^\lambda}(|f|).$$

If $g = g_1 + g_2$ with Lipschitz-continuous g_1 and differentiable, but not necessarily Lipschitz-continuous, g_2, one takes the MYE of g_1 and notes that 3.15 and 3.16 hold for g_1. Adding g_2 on both sides of the inequality shows that these inequalities remain true for g such that the proof still holds. □

3.6.2 Proof of Lemma 1

We now prove Lemma 1 which stated the following:

Lemma. *Let $g : \mathcal{X} \to]-\infty, \infty]$ be a proper lower semi-continuous convex function, and let $0 < \lambda_1 \leq \lambda_2$; then for the corresponding Moreau-Yosida envelopes g^{λ_1} and g^{λ_2}, we have*

$$\|\nabla g^{\lambda_1}(x)\| \geq \|\nabla g^{\lambda_2}(x)\| \quad \forall x \in \mathcal{X}.$$

Proof. The case $\lambda_1 = \lambda_2$ is trivial so assume $\lambda_1 < \lambda_2$.
Firstly recall that for convex g any MYE is also convex. Further note that g^{λ_2} is a Moreau-Yosida envelope for g^{λ_1}, with $g^{\lambda_2} = (g^{\lambda_1})^{\lambda_2 - \lambda_1}$ (Bauschke et al., 2011, Proposition 12.22 (ii)). We may thus define $h = g^{\lambda_1}$, $\lambda = \lambda_2 - \lambda_1$, such that the statement of the lemma is equivalent to

Lemma (Equivalent Formulation of Lemma 1)**.** *For any convex and differentiable function $h : \mathcal{X} \to]-\infty, \infty]$, and for any $\lambda > 0$, the Moreau-Yosida envelope h^λ satisfies*

$$\|\nabla h(x)\| \geq \|\nabla h^\lambda(x)\| \quad \forall x \in \mathcal{X}.$$

We define $p = \operatorname{prox}_h^\lambda(x)$. By theorem 2, $\nabla h^\lambda(x) = (x - p)/\lambda$; and by convexity (and differentiability) of h, we have for any $x \in \mathcal{X}$:

$$\begin{aligned}
0 &\leq \langle \nabla h(p) - \nabla h(x), p - x \rangle \\
&= \langle \nabla h(p) - \nabla h(x), -\lambda \nabla h(p) \rangle \\
&= -\lambda \|\nabla h(p)\|^2 + \lambda \langle \nabla h(x), \nabla h(p) \rangle \\
&\leq -\lambda \|\nabla h(p)\|^2 + \frac{\lambda}{2}\|\nabla h(x)\|^2 + \frac{\lambda}{2}\|\nabla h(p)\|^2 \\
&= \frac{\lambda}{2}\|\nabla h(x)\|^2 - \frac{\lambda}{2}\|\nabla h(p)\|^2,
\end{aligned}$$

where the first inequality is a necessary and sufficient condition for convexity of a differentiable function, and the last inequality follows from Young's inequality as $\langle x, y \rangle \leq$

$\|x\|^2/2 + \|y\|^2/2$.

$$\|\nabla h(x)\|^2 \geq \|\nabla h(p)\|^2 \overset{(3.2)}{=} \|\frac{1}{\lambda}(x-p)\| = \|\nabla h^{\lambda}(x)\|$$

as required. The last equality is given by Theorem 2. □

Chapter 4

Parallelisation for Localisable Inverse Problems

4.1 Introduction

We already saw in Chapter 3 how the performance of samplers can be significantly improved if the problem factorises into smaller problems. This chapter will investigate this further by exploring ways to parallelise computation when the target of interest is a localisable Bayesian inverse problem.

Throughout this chapter, the variable of interest is $x \in \mathbb{R}^d$. It will be illustrative to think of $\mathbb{R}^d$ as $\mathbb{R}^{d_1 \times d_2}$, where the variables $x_{i,j}$ are defined on a grid. This is common when solving two-dimensional partial differential equations (PDEs) or when the object of interest is an image. Given x, the observations are obtained through

$$y = h(x) + \xi, \tag{4.1}$$

where h is the observation function, and ξ is noise from some known probability distribution. Examples of observation functions are forward operators of PDEs, where the entries of x are a discretisation of the domain (Stuart, 2010; Wróblewska et al., 2016), blurring kernels in image analysis (Idier, 2013), and Fourier transforms, to name a few.

One talks about a *Bayesian* inverse problem if one defines a prior distribution π_0 on x, considers the likelihood $\mathcal{L}(y|x)$, and is interested in finding the posterior distribution

$\pi(dx) \propto \pi_0(dx)L(y|x)$. As discussed in the introductory chapter of this thesis, this normally boils down to the problem of sampling from the posterior distribution.

Especially in high-dimensional problems, using global sampling strategies becomes prohibitively costly, and exploiting the localised structure of the problem is key to successful sampling strategies.

Intuitively, localised inverse problems are characterised by a spatial structure of the variables, with variables being uncorrelated under the posterior if they are far from one another. This is the case for many examples in imaging problems. There, the priors are chosen to enforce some smoothness of the image, but pixels get more and more uncorrelated as one increases the distance between them. Similarly, the likelihood often corresponds to observing a blurred pixel with some noise, crucially, an observation will not normally depend on what the image looks like at far away pixels. Outside of image analysis, localisation can be found e.g. in numerical weather prediction. There, localisation is often enforced when using ensemble Kalman filters (Hamill et al., 2001), which is achieved by taking a banded forecast covariance matrix and setting small off-diagonal elements to zero (Gaspari and Cohn, 1999; Morzfeld et al., 2019) in a way that leaves the matrix positive definite.

While localised problems exist, one will find many problems to be only approximately localised, in the sense that while variables are increasingly decorrelated in space, they are never completely uncorrelated. In Morzfeld et al. (2019), the localised posterior is defined and targeted, by setting correlations to zero if they are small enough. This introduces an error, which is difficult to analyse. One way to circumvent this problem is to incorporate the localised posterior within a delayed-acceptance method. These methods, going back to Christen and Fox (2005), use an approximate target distribution to decide whether proposals are promising or not, and only evaluate the costly posterior if they are. For fully localised posteriors, this additional correction step is not necessary.

All methods proposed in this chapter are exploiting the localised structure of the problem to define subsets of the variables which can be sampled from in parallel, which significantly improves the performance of the sampling algorithms.

The new contributions of this chapter are as follows:

- We distinguish the localisation assumptions from Morzfeld et al. (2019) by defining *full localisation* and *approximate localisation*, the latter assumption satisfied by a class of inverse problems which contains all fully localised problems.

- We propose two different samplers that both have processes working on different blocks of variables in parallel. Both of them are proved to be asymptotically exact on their respective class of problems, one for fully localised problems, the other for approximately localised ones.
- The performance of the proposed samplers is empirically investigated in numerical examples.

The rest of this chapter is organised as follows: Section 4.2 rigorously defines *fully localised* and *approximately localised* problems. Section 4.3 introduces two new algorithms to be used for (approximately) localised problems, which are used for sampling in Section 4.4. The chapter ends with a discussion in Section 4.5.

4.2 Localisation

Localised problems are characterised by priors and likelihoods that show correlations or conditional dependencies of variables only in a neighbourhood.

In Morzfeld et al. (2019), only Gaussian priors and Gaussian noise in the observation model (4.1) are considered, i.e. $\pi_0 = \mathcal{N}(m, C)$ and $\xi \sim \mathcal{N}(0, R)$ for a covariance matrix R. The assumption of Gaussian priors is overly restrictive, as other priors can also have meaningful local properties. One such example is a prior where each x_i follows an independent Laplace distribution, similar to Example 3.4.2 in the previous chapter. It is thus useful to relax the assumption of Gaussian priors, and define a *fully localised inverse problem* as one satisfying the following assumptions, which are inspired by and similar to the ones found in Morzfeld et al. (2019):

1. (a) the state dimension d is large and the number of observations k is of the same magnitude, i.e., $k = O(d)$;
   (b) in the case of a Gaussian prior, the prior precision matrix is banded; in the case of non-Gaussian priors, variables are conditionally independent given a small number of other variables;
   (c) each predicted observation $[h(x)]_j$ has dependence on only $l \ll d$ components of x, and R is diagonal. Normally the variables which $[h(x)]_j$ depends on are in a neighbourhood of x_j, but this is not strictly necessary.

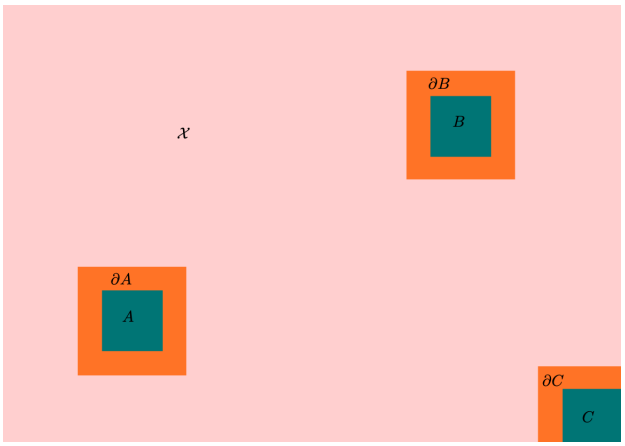


Fig. 4.1 Illustration of a fully localised Bayesian inverse problem. The variables in the set A and the ones outside $A \cup \partial A$ are conditionally independent given the ones in ∂A, similarly for B and C. This allows working on the regions A, B, and C in parallel, see Algorithm 6. If negligible dependence remains after conditioning on ∂A, Algorithm 6 is not targeting the correct target, but a delayed-acceptance step can be incorporated to correct for the error, see Algorithm 7.

Note that in contrast to the assumptions in Morzfeld et al. (2019), the stated assumptions are stricter, and they are visualised in Figure 4.1. We will see how this is beneficial to one of the algorithms in the next section, and emphasise that there are examples satisfying the strict localisation assumptions: one example is an imaging problem, where the Gaussian prior has a banded precision, and the observation function is a uniform blur kernel acting only locally. This is the setup of the numerical Example 4.4.1.

As a second set of assumptions, we relax the strict localisation assumptions in the spirit of those posed by Morzfeld et al. (2019). We refer to a problem as *approximately localised* if the following conditions are satisfied:

2. (a) the state dimension d is large and the number of observations k is of the same magnitude, i.e., $k = O(d)$;

   (b) in the case of a Gaussian prior, the prior precision matrix is *nearly* banded; in the case of non-Gaussian priors, variables are *almost* conditionally independent given a small number of other variables;

   (c) each predicted observation $[h(x)]_j$ has *significant* dependence on only $l \ll d$ components of x (i.e., the contribution from the remaining components of x is *small*), and R is diagonal.

This approximate localisation condition is fairly vague, and this is not without consequence. The larger the dependencies of different variables get, the less localised the problem is, and the worse the localisation error gets: this error is defined as the error $\|\pi - \pi^{loc}\|$, where π^{loc} is the localised posterior obtained by enforcing Assumptions 1. In the next subsection, we will discuss how to obtain a localised posterior distribution.

4.2.1 Localised Posteriors

Under a compatibility assumption (Arnold and Press, 1989), a joint density function $f(x_1, \ldots, x_d)$ is fully characterised by the full conditional distributions $f(x_i|x_{-i})$. This justifies the use of the Gibbs sampler in the first place, and we will use this property here to define the localised posterior distribution π_{loc} through its density f_{loc}. We assume there exists, for each $j \in \{1, \ldots, d\}$ and suitable sets of neighbours denoted by ∂j, a good approximation f_{loc} such that

$$f_{loc}(x_j|X^{\partial j} = x^{\partial j}) = f_{loc}(x_j|X^{-j} = x^{-j}) \approx f(x_j|X^{-j} = x^{-j}), \tag{4.2}$$

and define the localised posterior as the joint density arising from this approximation, assuming the compatibility assumptions are satisfied. Good localised posteriors have conditionals that are close to the full conditionals of π in some metric such has the total variation distance. We emphasise that the localised posterior is not unique, and introduce a few examples in the following.

Example 1: Gaussian density with tridiagonal precision matrix. Let $\pi = \mathcal{N}(0, \Sigma^{-1})$ for a tridiagonal precision matrix Σ. Then Equation (4.2) holds for the Gaussian density with the neighbourhood of variable j given by $\partial j = \{j-1, j+1\}$. A similar result holds for sparse precision matrices, where $\partial j = \{i : \Sigma_{i,j} \neq 0\}$, this is the case for Example 4.4.1.

Example 2: Gaussian density with banded covariance matrix. If the prior covariance matrix is banded, the precision matrix Σ will generally not be, but a banded approximation $\hat{\Sigma}$ thereof exists (Bickel and Lindner, 2012) by setting all values below a certain threshold to 0. Using this banded approximation of the precision matrix yields a localised posterior with the same neighbourhood as in the previous example. The choice of threshold impacts how localised the posterior is, as more entries in $\hat{\Sigma}$ will be non-zero for larger thresholds. We will see a localised Gaussian posterior in Example 4.4.2.

Example 3: Localisation of a PDE model. Consider a PDE that maps a function u to another function p_u (for a specific example we point to the groundwater flow example in Section 5.2). We aim to infer the true function u^* from noisy observations y_i of $p_{u^*}(x_i)$, $i = 1..n$, that is

$$y_i - p_{u^*}(x_i) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2),$$

giving rise to the log-likelihood

$$\ell(y|u) = -\frac{1}{\sigma^2} \sum_i (y_i - p_u(x_i))^2.$$

When solving the PDE numerically, we will discretise the domain, and we may assume that changing u on only a few grid points will not affect p_{u^*} on grid points far away. This can be modelled by an approximate solution operator $\hat{p}_u$ that solves the forward model on a smaller domain, giving the approximate log-likelihood

$$\hat{\ell}(y|u) = -\frac{1}{\sigma^2} \sum_i (y_i - \hat{p}_u(x_i))^2.$$

The neighbourhood of a variable j is then the set of points included in the calculation of the local solution, i.e.

$$\partial j = \{i : \forall u, v \text{ s.t. } \forall k \neq j,\ u_k = v_k \text{ and } \hat{p}_u(x_i) \neq \hat{p}_v(x_i)\}.$$

As in the previous example, the size of the local domain chosen influences the accuracy of the approximation 4.2. Note that this neighbourhood structure considers only the likelihood, the full neighbourhood is the union of the one arising from the prior and the one from the likelihood.

4.3 Sampling Algorithms

4.3.1 Parallelisation of a Metropolis-within-Gibbs Algorithm

For completely localised problems, that is, problems satisfying Assumptions 1, it is possible to modify the blocked Metropolis-within-Gibbs method to update conditionally independent blocks in parallel. This method is a modification of the Gibbs sampler introduced in Chapter

2, where instead of a single variable, a block of variables gets updated jointly. Given a sample x_{n-1}, one updates the variables in block A according to $\pi(X^A = dx^A | X^{-A} = x_{n-1}^{-A})$. For fully localised problems, we have that the variables in a set A depend only on a subset of all variables, which we denote by ∂A to indicate the localisation, i.e.

$$\pi\left(X^A = dx^A | X^{-A} = x_{n-1}^{-A}\right) = \pi\left(X^A = dx^A | X^{\partial A} = x_{n-1}^{\partial A}\right).$$

Furthermore, if $A \cap \partial B = \emptyset$ and if $\partial A \cap B = \emptyset$

$$\begin{aligned}
&\pi\left(X^A = dx^A, X^B = dx^B | X^{-(A\cup B)} = x_{n-1}^{-(A\cup B)}\right) \\
&= \pi\left(X^A = dx^A | X^B = x^B, X^{-(A\cup B)} = x_{n-1}^{-(A\cup B)}\right)\pi\left(X^B = dx^B | X^{-(A\cup B)} = x_{n-1}^{-(A\cup B)}\right) \\
&= \pi\left(X^A = dx^A | X^{\partial A} = x_{n-1}^{\partial A}\right)\pi\left(X^B = dx^B | X^{\partial B} = x_{n-1}^{\partial B}\right),
\end{aligned} \tag{4.3}$$

where the second equality is due to conditional independence. This allows to update the blocks A and B simultaneously, a generalisation to more blocks is straightforward.

One of the main benefits of this method is that no worker process ever needs to be idle: once a local update has been completed, the worker process can release the region it just worked on, which can then be used by other processes. The process will then pick another available region, block it, propose a move, accept or reject it, and then repeat the process of releasing the old region and finding a new region to work on. This requires some care when implementing the method, but allows all worker processes to be used efficiently, rather than forcing a worker process to wait until the other processes have finished their updates. The proposals for a block of variables x^{S_j} are generated from a kernel conditioning on the neighbourhood variables $x^{\partial S_j}$, the transition kernel is accordingly defined by

$$Q(x^{S_j}, dy^{S_j} | X^{\partial S_j} = x^{\partial S_j})$$

and its density is, for ease of notation, denoted

$$q(y^{S_j} | x^{S_j}, x^{\partial S_j}).$$

Algorithm 6 specifies the complete method.

To show that this method works, we view it as follows. Let K_j be the transition kernels arising from updating the j-th variable block (i.e. all variables in S_j). A systematic scan blocked Gibbs sampler would apply K_1, then K_2, etc. up to K_M; a random scan blocked Gibbs sampler would sample j uniformly from $\{1, \ldots, M\}$, and we here propose to pick K_j

Algorithm 6 Parallel implementation of the blocked Metropolis-within-Gibbs algorithm. In the parallelised iterations, parts need to be executed using *locks* to ensure workers do not accidentally work on the same region. Note that a new sample is created (lines 6 and 25) which is then locally changed by the workers (lines 10 to 23, particularly line 19).

1: **procedure** PARALLELISED, BLOCKED METROPOLIS-WITHIN-GIBBS
2: $x_0 \sim \pi(\cdot)$ ▷ Draw initial value x_0
3: $\mathcal{S} = \{S_1, \dots, S_M\}$ ▷ List of blocks
4: Set τ ▷ Sample time
5: Set w ▷ Set number of workers
6: $x_1 \leftarrow x_0$ ▷ Copy first sample
7: **for** $n = 1 \dots$ **do**
8: **for** $j \in \{1, \dots, w\}$ **do** ▷ Iterations run in parallel
9: $t_j \leftarrow$current time
10: **while** current time$-t_j < \tau$ **do** ▷ While loop executed for fixed time
11: WITH LOCK: Sample S_j from free blocks in $\mathcal{S}$ and block $\partial S_j \in \mathcal{S}$
12: propose $y^{S_j} \sim Q(x^{S_j}, \cdot | X^{\partial S_j} = x^{\partial S_j})$ ▷ Propose local update
13: $r_j \leftarrow q(x_n^{S_j} | y^{S_j}, x^{\partial S_j}) / q(y^{S_j} | x_n^{S_j}, x^{\partial S_j}))$
14: $s_j \leftarrow f(y^{S_j} | X^{\partial S_j} = x^{\partial S_j}) / f(x_n^{S_j} | X^{\partial S_j} = x^{\partial S_j})$
15: $a_j \leftarrow \min(1, r_j s_j)$ ▷ Calculate acceptance probability
16: $u_j \sim U([0,1])$
17: **if** $u_j < a_j$ **then**
18: Update $x_n^{S_j} = y^{S_j}$ ▷ Accept proposed move
19: **end if**
20: WITH LOCK: Release S_j and ∂S_j into free blocks in $\mathcal{S}$
21: **end while**
22: **end for**
23: $x_{n+1} \leftarrow x_n$ ▷ Current global sample is stored, new sample prepared for local changes
24: **end for**
25: **end procedure**

in a slightly more advanced way. Algorithm 6 corresponds to first apply one of the K_j with equal probability, and then allow only certain updates after that. The blocking in Algorithm 6 ensures that while some other kernels are applied, the required calculations for another kernel K_l can be executed. This is possible as no variables involved in these calculations get updated during the calculation process.

To be more precise, we assume for simplicity that only two worker processes are involved, and define the set

$$\kappa = \{(K_i, K_j) : i \neq j, S_i \cap \partial S_j = S_j \cap \partial S_i = \emptyset\}.$$

Then Algorithm 6 can be interpreted as the following blocked Gibbs sampler:

- Initially, (K_i, K_j) is drawn uniformly from κ;
- Initialise $x_0 \sim \pi$
- Iterate for $n = 1..N$:

  Pick K_i or K_j with a certain probability $p_{w,n}$, wlog we pick K_i (if we pick K_j, change i and j in the next steps)

  Apply the kernel K_i to x^n, update $x^{n+1} \sim K_i(x^n, \cdot)$

  Sample (K_l, K_j) from κ, keeping K_j fixed

  Relabel l to i

Note that the probabilities $p_{w,n}$ depend on the runtime of the processors, but we assume they are bounded away from 0. As each kernel is a Metropolis-Hastings update of some variables, they are all π-reversible and thus leave π invariant. If the resulting Markov chain admits a unique stationary distribution, this will therefore be π. As before in Section 2.3.1, ergodicity can not be shown in general (it is straightforward to construct a target distribution similar to the one in Section 2.3.1 that gets stuck). If for any initial choice of (K_i, K_j), all kernels in κ will be picked and all K_i are in a tuple in κ, Algorithm 6 results in a irreducible, positive recurrent, and aperiodic Markov chain for the examples in Section 4.4. Aperiodicity follows as the same kernel might be applied twice consecutively, which breaks any periodicity. Irreducibility and positive recurrence follows from the choice of the kernels K_i and the fact that for any sets A and B with $\pi(A) > 0$, $\pi(B) > 0$, there exists some $t \in \mathbb{N}$ and a sequence of kernels $K_{j_1}, K_{j_2}, \ldots, K_{j_t}$ such that this sequence of kernels has positive probability of being picked and further

$$\int_B \int_X \cdots \int_X \int_A K_{j_t}(x^{t-1}, y) K_{j_{t-1}}(x^{t-2}, x^{t-1}) \cdots K_{j_1}(x^0, x^1) \pi(x^0) dx^0 dx^1 \cdots dx^{t-1} dy > 0,$$

as for a Gaussian density with Gaussian proposals the kernels are irreducible and positive recurrent on their respective variables. In summary, we start in A and end in B with positive probability.

The general case for w worker processes follows similarly by defining κ to be the set of all admissible w-tuples of kernels, such that the kernels in each tuple work on separated variable blocks. If for any initial tuple choice all kernels will be picked, the arguments generalise.

4.3.2 Parallelisation in a Delayed-Acceptance Metropolis-Hastings Algorithm

In only approximately localisable problems satisfying Assumptions 2, one can still apply Algorithm 6. However, as the conditional independence we used in (4.3) does not hold, the algorithm will not target the correct posterior π, but the localised version thereof, π^{loc}, which is defined through the joint density $\hat{f}$ when using the localised conditionals from Equation (4.2). For Gaussian targets, this is discussed extensively in Morzfeld et al. (2019). Recall the the error $\|\pi - \pi^{loc}\|$ is difficult to estimate, and a thorough analysis of the problem at hand is required to ensure the localised posterior is close to the desired target distribution.

To circumvent this problem, we propose to use a delayed-acceptance scheme (Christen and Fox, 2005) to target the full conditionals in a blocked Gibbs sampler. The delayed-acceptance method is generally particularly strong when the posterior π is computationally expensive to evaluate, but a cheap approximation $\hat{\pi}_x$ is available. Both the posterior and its approximation admit densities, denoted f and $\hat{f}_x$, respectively. As the subscript indicates, the approximation may depend on the current state. Given a current state x_{n-1}, the method proposes a new state y and uses the approximation to decide if the proposal y should be outright rejected, or if it is a promising candidate. The acceptance probability at this first stage is given by

$$a_1(x,y) = \min\left(1, \frac{q(y|x)\hat{f}_x(y)}{q(x|y)\hat{f}_x(x)}\right).$$

If the proposal is rejected, one sets $x_n = x_{n-1}$. Otherwise, if the proposal is accepted in this first stage, the true posterior is evaluated, and the proposal is finally accepted with probability

$$a_2(x,y) = \min\left(1, \frac{a_1(y,x)q(x|y)f(y)}{a_1(x,y)q(y|x)f(x)}\right), \tag{4.4}$$

setting $x_n = y$, or $x_n = x_{n-1}$ if the proposal is rejected at the second stage. The motivation behind the delayed-acceptance is to reject bad proposals early, and save computational resources by only investing into the evaluation of the posterior if the proposal looks promising. As nicely pointed out by Quiroz et al. (2018), the second stage can be interpreted as the proposal density arising from a mixture of two proposals: one either uses the proposal density used in the first stage, or (if the proposal was already rejected in the first stage) proposes to stay at x_n. This allows to verify the detailed balance condition (2.3), ensuring the delayed-acceptance kernel targets the desired posterior. It is useful to note that, if $\hat{\pi}_x = \hat{\pi}_y$ or

if $\hat{\pi}_x$ does not depend on the current iterate x, the acceptance probability at the second stage simplifies as stated in the following theorem:

Theorem 5. *If the posterior approximation for a single iteration of the delayed acceptance algorithm satisfies $\hat{\pi}_x = \hat{\pi}_y$, or if $\hat{\pi} = \hat{\pi}_x$ does not depend on a sample x at all, the acceptance probability at the second stage of the delayed acceptance algorithm simplifies to*

$$a_2(x, y) = \min\left(1, \frac{\hat{f}(x)f(y)}{\hat{f}(y)f(x)}\right). \tag{4.5}$$

Proof. See Appendix 4.6.1. □

This theorem is useful in practice, as in Algorithm 7 one targets the full conditional $\pi_{x,\cup S_j}$ using the approximate conditional $\hat{\pi}_{x,\cup S_j}$ with densities given by

$$f_{x,\cup S_j}\left(y^{\cup S_j}\right) := \frac{1}{Z_{x,\cup S_j}} \prod_j f\left(y^{S_j} | X^{S_j^C} = x^{S_j^C}\right), \tag{4.6}$$

and

$$\hat{f}_{x,\cup S_j}\left(y^{\cup S_j}\right) := \frac{1}{\hat{Z}_{x,\cup S_j}} \prod_j \hat{f}\left(y^{S_j} | X^{\partial S_j} = x^{\partial S_j}\right), \tag{4.7}$$

respectively.

The normalisation constant $\hat{Z}_{x,\cup S_j}$ depends only on the variables in $\cup \partial S_j$ which is fixed throughout one iteration and therefore does not change, and we can thus apply Theorem 5 even when the densities are only available up to a normalisation constant, and the simplified acceptance rate (4.5) can be used.

If the problem satisfies Conditions 2 and one has access to a localised posterior with the approximate conditionals $\hat{\pi}_{x,\cup S_j}$, we propose to use Algorithm 7 which is a blocked Gibbs sampler using the delayed-acceptance algorithm to sample from the full conditionals. The localised structure allows to calculate the approximate conditionals in parallel: given a set of blocks $S_{j_1}, \ldots, S_{j_w}$, the approximate target used in the first stage is the localised posterior $\hat{\pi}_{x,\cup S_j}$ with density $\hat{f}_{x,\cup S_j}$ described in Equation (4.7). The costly full conditional $\pi_{x,\cup S_j}$ is only evaluated if the proposals from the first stage are promising, and the acceptance probability at the second stage will be increasing as the correlations and conditional dependencies between variables are decreasing. If the problem is fully localised, satisfying Assumptions 1, all proposals passing the first stage will be accepted at the second stage as $\hat{f} = f$, and thus

$a_2(x,y) = 1$ in (4.5) always. In that case, the delayed acceptance step is superfluous, and Algorithm 7 reduces to Algorithm 6.

From a computational point of view, the delayed acceptance algorithm has some disadvantages. While the local proposal processes can work in parallel, the global master process has to wait until all worker processes have completed their calculations. This will cause some workers to be idle, and, yet worse, the global second stage cannot be parallelised. This is in stark contrast to Algorithm 6, as discussed in the previous subsection. To alleviate this problem a little bit, each local proposal on the first stage can be rejected separately, such that the proposal carried forward to the second acceptance stage is the concatenation of all accepted local proposals, the rejected local proposals, and the variables not touched in the first stage. While this is promising especially for large numbers of workers, we do not investigate this further here.

4.4 Numerics

In this section we compare the algorithms developed in the previous section to a Metropolis-Adjusted Langevin Algorithm (MALA) on two imaging examples. Throughout this section, the prior is the normal distribution given by $\mathcal{N}(0, \delta L^{-1})$, where L is the two-dimensional Laplacian, and $\delta = 0.1$, which is the same prior as used in Example 5.4 in Morzfeld et al. (2019). The two-dimensional Laplacian is the matrix which has $L_{i,i} = 4$ on all diagonal entries, and a $L_{i,j} = -1$ on the entries corresponding to the four direct neighbour variables j, assuming periodic boundary conditions. The precision matrix is $\Omega = 10L$ and as such trivially localised, as each variable depends only on the four direct neighbours, and is conditionally independent of all others given those.

The true image x is the 128×128 sector of the 'airplane' image pictured in the left panel of Figure 4.2. It is observed through $y = Hx + \xi$, where $\xi \sim \mathcal{N}(0, \sigma^2 I_{128,128})$ with $\sigma^2 = 0.1^2$, and H is a blur operator to be specified below in the examples. The noise level is higher than in the setup of Example 5.4 in Morzfeld et al. (2019), as the goal in that paper was to show the better performance of the Gibbs sampler itself, while here the focus lies on Metropolis-within-Gibbs schemes. The blur operators are again assumed to satisfy periodic boundary conditions. We note the similarity of this example to the imaging example in Section 3.4.6, where a total variation prior was used instead of a Gaussian prior. For both

Algorithm 7 The delayed acceptance algorithm exploiting local proposals which can be run in parallel. The second stage acceptance ratio is simplified as suggested in Theorem 5.

procedure PARALLELISED DELAYED-ACCPETANCE
  $x_0 \sim \pi(\cdot)$ ▷ Draw initial value x_0
  Set w ▷ Set number of workers
  for $n = 1 \dots$ **do**
    for $j \in \{1, \dots, w\}$ **do**
      Sample S_j ▷ Pick w blocks of variables such that they are approximately conditionally independent given all other variables
    end for
    for $j \in \{1, \dots, w\}$ **do** ▷ Iterations run in parallel
      propose $y^{S_j} \sim Q(x^{S_j}, \cdot | X^{\partial S_j} = x^{\partial S_j})$ ▷ Propose local update
      $r_j \leftarrow q(x^{S_j}_{n-1} | y^{S_j}, x^{\partial S_j}) / q(y^{S_j} | x^{S_j}_{n-1}, x^{\partial S_j}))$
      $s_j \leftarrow \hat{f}(y^{S_j} | X^{\partial S_j} = x^{\partial S_j}) / (\hat{f}(x_{n-1} | X^{\partial S_j} = x^{\partial S_j})$
    end for
    $r_1 \leftarrow \prod_j r_j s_j$ ▷ Calculate first stage MH ratio
    $a_1 \leftarrow \min(1, r_1)$ ▷ Calculate first stage acceptance probability
    $u_1 \sim \mathcal{U}([0,1])$
    if $u_1 < a_1$ **then**
      $y \leftarrow y^{S_j}, x^{(\cup S_j)^C}$ ▷ Globalise proposals to a global one
      $r_2 \leftarrow (f(y)/f(x)) \times \prod_j s_j^{-1}$ ▷ Calculate second stage MH ratio
      $a_2 \leftarrow \min(1, r_2)$ ▷ Calculate second stage acceptance probability
      $u_2 \sim \mathcal{U}([0,1])$
      if $u_2 < a_2$ **then**
        $x_n \leftarrow y$ ▷ Accept proposed move
      else
        $x_n \leftarrow x_{n-1}$ ▷ Reject proposal
      end if
    else
      $x_n \leftarrow x_{n-1}$ ▷ Reject proposal
    end if
  end for
end procedure

examples, the resulting posterior is thus given by

$$\pi(x) \propto \exp\left(\frac{1}{2\sigma^2}\|y - Hx\|_2^2 - \frac{1}{2\delta}x^T L x\right),$$

where H is the used blur kernel, and the posterior mean is available analytically as

$$\mathbb{E}_\pi(x) = \left[\frac{1}{\sigma^2}H^2 + \frac{1}{\delta}L\right]^{-1}\frac{1}{\sigma^2}Hy.$$

The code for the numerical examples in this section is available online at https://github.com/TorbenSell/localised-parallelised.

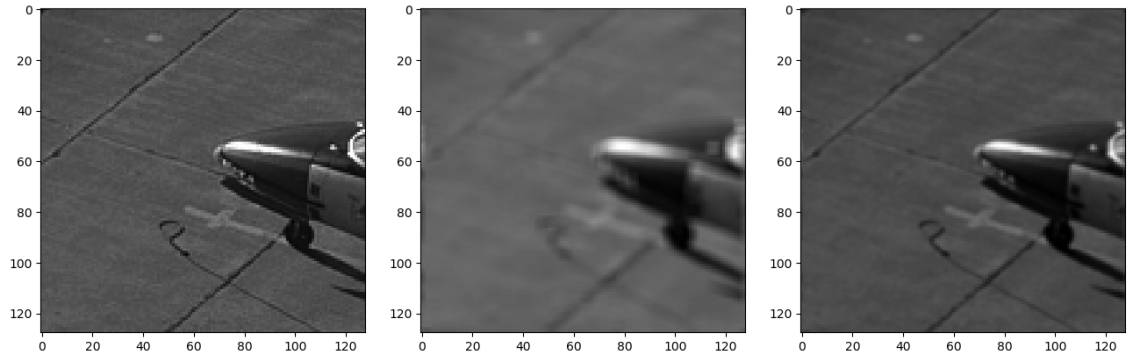


Fig. 4.2 Left: True image. Centre: Blurred observation using the uniform blur operator of Example 4.4.1. Right: Blurred observation using the exponential blur operator of Example 4.4.2.

4.4.1 A Fully Localised Imaging Problem

In this example, the blur operator H is chosen to be such that it blurs a pixel $x_{i,j}$ with its closest neighbours in a 5×5 patch. This results in a fully localised example, and the parallel blocked Metropolis-within-Gibbs (para-MwG, Algorithm 6, with Q chosen to be a local MALA propoal) is compared to MALA. For the latter, we set the step size $\delta = 1.1\times10^{-3}$, resulting in 65% acceptances. For para-MwG, 4 workers worked in parallel on 8×8 blocks, conditioned on all other variables in a larger 16×16 block. The step size was set to $\delta = \times10^{-2}$, resulting in 72% acceptances. The computer used for the simulations only has two processor cores, and thus only two workers effectively work in parallel, such that the results are not fully representing the full potential of the method. Theoretically, if the image is split into 256 fixed blocks, up to 36 workers can be employed in parallel if the computer architecture allows this. Figure 4.3 shows the mean squared error (MSE) and the structural similarity index (SSIM) between the 'true' posterior mean and the means as estimated by the two algorithms. The 'true' posterior mean has been estimated by a long run of MALA.

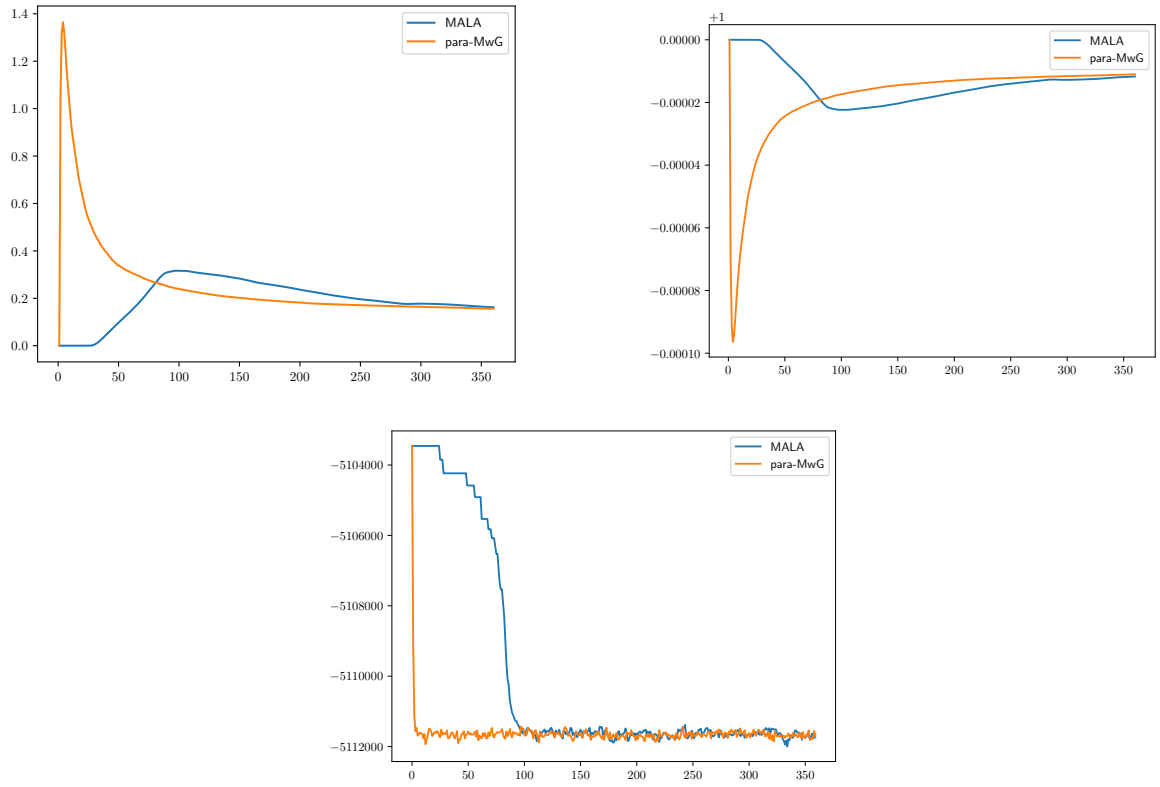


Fig. 4.3 Results from the fully localised Image Deblurring example. Left: The MSE of the mean estimates, estimated every 15 seconds. Right: The SSIM of the mean estimates, estimated every 10 seconds over an hour. Para-MwG can be sped up by a factor of around 18 by using a better computer architecture. Bottom: Log-posterior traceplots to assess mixing behaviour of the chains.

4.4.2 An Approximately Localised Imaging Problem

In this example, an exponential blur operator with standard deviation 0.7 is used. This kernel results in the posterior not being fully localised, but as the correlations and conditional dependencies decay quickly, the approximate localisation assumptions are satisfied.

The following three algorithms are compared: MALA, the parallel delayed acceptance algorithm (para-DA, Algorithm 7), and para-MwG (Algorithm 6). The latter one targets the localised posterior, but the mean estimate is still close to the truth, and the mean squared error (MSE) decays much more quickly than the MSE obtained from para-DA. Again, the block sizes are chosen to be of size 8×8, and the local proposals are conditioned on the other variables in a 16×16 block. The step sizes were set to $\delta = 8 \times 10^{-4}$ for MALA (giving 75% acceptances), $\delta = \times 10^{-2}$ for para-MwG (giving 65% acceptances), and $\delta = 5 \times 10^{-3}$ for para-DA. For para-DA, 78% of proposals were rejected at the first stage, less than 1%

was rejected at the second stage, and 21% were accepted. The results are summarised in Figure 4.4. Para-DA performs significantly worse here. This is not surprising: the small local moves are often rejected, as the approximation is not good enough to yield a high acceptance rate at the second stage. The size of the conditioning regions ∂S_j could be increased, which would result in higher acceptance rates at the second stage. This, however, would come at the price of computationally more expensive proposals. Therefore, para-DA can only be efficiently employed if accurate estimations are crucial and the problem is almost fully localised. Otherwise, MALA or para-MwG are the better options for sampling from approximately localised problems.

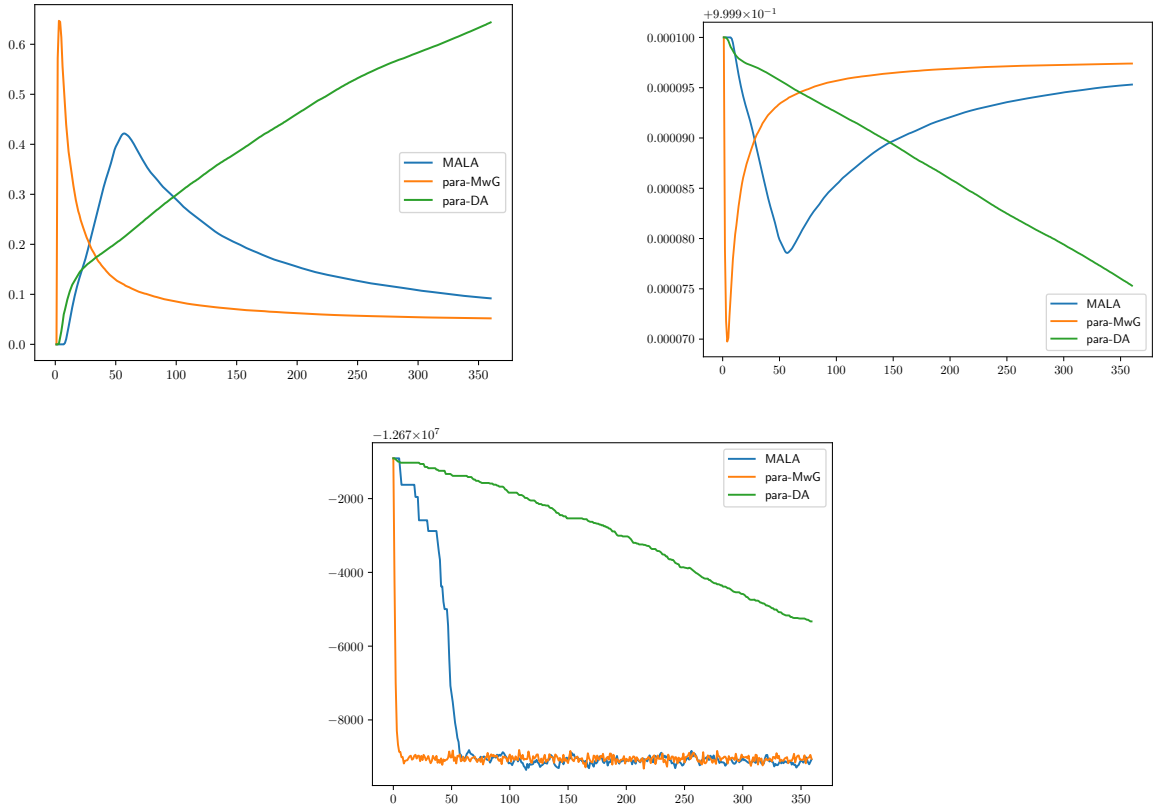


Fig. 4.4 Results from the approximately localised Image Deblurring example. Left: The MSE of the mean estimates, estimated every 15 seconds. Right: The SSIM of the mean estimates, estimated every 10 seconds over an hour. While para-MwG only targets an approximate posterior distribution π_{loc}, the mean estimate of π_{loc} seems to be close to the true mean of π. Bottom: Log-posterior traceplots to assess mixing behaviour of the chains.

4.5 Discussion

Especially inference on the fully localised inverse problem massively profits from exploiting the localised structure. The advantage of using a blocked and parallelised Metropolis-within-Gibbs grows with the size of the problem at hand. In approximately localised inverse problems, it may be worth accepting the error from completely localising the problem, as global algorithms suffer: MALA is slow as the global proposals and calculating the global acceptance ratio is computationally expensive, similarly, the global correction step in the delayed acceptance method is often costly and results in very slow mixing of the resulting Markov chain.

Multiple questions remain for future research:

- Can one find bounds on the localisation error, depending on the entries of the precision matrix?
- The TV prior used in Example 3.4.6 is localisable, but the MYE smoothed version is not. Is it possible to define a smooth envelope of the (non-differentiable) target to be used in diffusion-based algorithms? An interesting approach is developed in Vono et al. (2020), where an auxiliary variable allows to use a Gibbs sampler for sampling in a TV example.
- If the target is non-differentiable, what is the best strategy for parallelisation? As seen in Chapter 3, PDMPs are a viable option as they are strong on localisable targets, see also the Gibbs Zig-Zag samplers (Sachs et al., 2020). Is a PDMP-within-Gibbs algorithm the best way of global communication, or do more efficient strategies exist?
- How can one localise other priors to exploit the benefits of localisation in a larger class of problems?
- What is the optimal balance between the numbers of workers and the size of the blocks they work on?

To conclude, we note that parallelisation is key to successful and efficient sampling in high-dimensional models, and para-MwG is a strong algorithm for fully localised problems and many approximately localised ones.

4.6 Appendix

4.6.1 Proof of Theorem 5

We now prove Theorem 5 which stated the following:

Theorem. *If the posterior approximation for a single iteration of the delayed acceptance algorithm satisfies $\hat{\pi}_x = \hat{\pi}_y$, or if $\hat{\pi} = \hat{\pi}_x$ does not depend on a sample x at all, the acceptance probability at the second stage of the delayed acceptance algorithm simplifies to*

$$a_2(x,y) = \min\left(1, \frac{\hat{f}(x)f(y)}{\hat{f}(y)f(x)}\right).$$

Proof of Theorem 5.

$$\begin{aligned}
a_2(x,y) &= \min\left(1, \frac{a_1(y,x)q(x|y)f(y)}{a_1(x,y)q(y|x)f(x)}\right) \\
&= \begin{cases} \min\left(1, \frac{1\times q(x|y)f(y)}{\frac{q(x|y)\hat{f}(y)}{q(y|x)\hat{f}(x)}q(y|x)f(x)}\right) & \text{if } q(x|y)\hat{f}(y) \le q(y|x)\hat{f}(x) \\ \min\left(1, \frac{\frac{q(y|x)\hat{f}(x)}{q(x|y)\hat{f}(y)}q(x|y)f(y)}{1\times q(y|x)f(x)}\right) & \text{if } q(x|y)\hat{f}(y) \ge q(y|x)\hat{f}(x) \end{cases} \\
&= \begin{cases} \min\left(1, \frac{\hat{f}(x)f(y)}{\hat{f}(y)f(x)}\right) & \text{if } q(x|y)\hat{f}(y) \le q(y|x)\hat{f}(x) \\ \min\left(1, \frac{\hat{f}(x)f(y)}{\hat{f}(y)f(x)}\right) & \text{if } q(x|y)\hat{f}(y) \ge q(y|x)\hat{f}(x) \end{cases} \\
&= \min\left(1, \frac{\hat{f}(x)f(y)}{\hat{f}(y)f(x)}\right).
\end{aligned}$$

Note that if $q(x|y)\hat{f}(y) \ge q(y|x)\hat{f}(x)$, the acceptance probabilities coincide, so a_2 is still well defined in that case. □

Chapter 5

Function Space Priors

A modified version of the work presented in this chapter and the next is under review for publication as Sell and Singh (2020).

To emphasise that the object of interest in this chapter and the next is a function, the distributions are from now on defined over variables u rather than x; x will, throughout this chapter and the next, denote a location in $\mathcal{X} \subset \mathbb{R}^d$ over which these functions $u = u(x)$ are defined. To avoid confusion between functions f and the probability density functions of π and π_0, the latter ones are denoted π and π_0 as well, overloading the notation to achieve better readability.

5.1 Introduction

Generating samples from probability measures on function spaces is both a challenging computational problem and a very useful tool for many applications, including mathematical modelling in bioinformatics (Quarteroni et al., 2017), data assimilation in reservoir models (Iglesias et al., 2013), and velocity field estimation in glaciology (Minchew et al., 2015), amongst many others. This chapter addresses the problem of defining a computationally and statistically favourable function space prior.

In Bayesian inference on separable Hilbert spaces (Stuart, 2010), many posterior measures π are absolutely continuous with respect to their prior π_0 (often a Gaussian measure, see Knapik et al. (2011) and Dashti et al. (2013), but not always, see Dashti et al. (2011), Hosseini (2017), and Hosseini and Nigam (2017)), with the likelihood acting as the Radon-Nikodym

derivative $d\pi/d\pi_0 \propto \mathcal{L}$. When the priors are Gaussian measures, truncating their Karhunen-Loève expansions reduces the problem of sampling from infinite-dimensional measures to sampling from a finite-dimensional parameter space. This approximation to the true posterior gets better by including more parameters in the truncated expansion. The practical applicability of these priors are, however, restricted to inferring unknown functions with low-dimensional domain, as the orthogonal basis required results in the complexity scaling exponential with the dimension of the unknown function's domain.

Another approach to define function space priors are Bayesian Neural Networks (BNNs) (Neal, 1995, 2012) which drew a lot of attention in the machine learning community over the past years. These priors are obtained by placing a prior distribution over the weights and biases of a neural network, with the default choice being a centered Gaussian prior on the weights with variances that scale as $O(1/N^{(l)})$, where $N^{(l)}$ is the number of nodes in layer l. Some authors argue for heavy-tailed priors on the parameters which has been initially investigated in Neal (1995).

Although some theoretical results exist at least for the choice of Gaussian priors on the parameters (Matthews et al., 2018), popular criticisms include the lack of interpretability of the resulting BNNs, and recent work (Wenzel et al., 2020) has highlighted inter alia that novel priors are needed. Sampling approaches include Hamiltonian Monte Carlo (Neal, 1995), and more advanced integrators (Leimkuhler et al., 2019); However, inference is often limited to finding the maximum-a-posteriori (MAP) estimate of the posterior (Welling and Teh, 2011), and one cannot easily expand the parameter space to obtain more accurate estimates as is the case for the Karhunen-Loève expansion: one would either have to adjust the prior variances for all nodes within the amended layer, or have to deal with exploding functions.

Other popular function space priors include Deep Gaussian Processes (Damianou and Lawrence, 2013), for which few theoretical results (Dunlop et al., 2018) are known. Another prior is constructed as a mixture of Gaussian measures and results in a non-Gaussian prior; even infinitely many mixtures can be considered through the Dirichlet process for which inference algorithms exist (Neal, 2000). Taking a basis $\{\phi_i\}_{i=1}^{\infty}$ of a Hilbert space and placing any prior on the coefficients β_i in the expansion $\sum \beta_i \phi_i$ yields another prior, known as a *random basis expansion*. The latter two priors are examples of priors used in nonparametric Bayesian statistics, and are discussed extensively in Ghosal and Van der Vaart (2017). None of these prior can be straightforwardly used in conjunction with Hilbert space MCMC techniques which is the focus of this work, and we thus omit a discussion thereof.

To calculate expectations with respect to the respective posteriors, computational methods are required as the integrals are usually not analytically tractable. Two popular sampling algorithms for posteriors defined on Hilbert spaces are the preconditioned Crank-Nicolson (pCN) algorithm and its likelihood-informed counterpart the preconditioned Crank-Nicolson Langevin (pCNL), which arise from clever (and in a way optimal) discretisations of certain stochastic differential equations (Cotter et al., 2013), are asymptotically exact, and are dimension-independent in the sense that their step-size does not depend on the discretisation (Eberle et al., 2014; Hairer et al., 2014). This stands in stark contrast to the well-known dimensional-dependent scaling of popular MCMC algorithms such as Random Walk Metropolis and Metropolis Adjusted Langevin Algorithm (Roberts and Rosenthal, 1998; Roberts et al., 2001). Practical implementations of pCN and pCNL exploit the equivalence of Gaussian measures and the Karhunen-Loève (KL) expansion which holds under fairly general assumptions. Owing to the orthogonality of the basis in the KL expansion, these methods come with a variety of theoretical results, such as concentration inequalities and contraction rates, see e.g. Agapiou et al. (2013); Knapik et al. (2011); Nickl and Giordano (2020); van der Vaart et al. (2008). Geometric (Beskos et al., 2017) and likelihood-informed (Cui et al., 2016) modifications of pCNL can reduce the computational cost provided one knows which basis functions are informed by the data, but they cannot circumvent the costly scaling in the domain dimension. This is presumably the main reason why these methods have rarely been used for inferring unknown functions with domains larger than dimension two (i.e. $\mathbb{R}^2$) in reported examples in the literature.

This chapter introduces a new neural network based prior, coined *trace-class* neural network priors, which allows for scalable (in the domain dimension) Bayesian function space inference. Hilbert space MCMC algorithms are then used to sample from the resulting posteriors, and, owing to their stability under mesh-refinement, enhances the practical utility of our framework.

The new contributions of this chapter are as follows:

- We introduce a *trace-class* prior for neural networks, and demonstrate its practical utility. The prior is independent, centred, and Gaussian across weights but is non-exchangeable over the weights within each layer and has a summable variance sequence. The latter ensures it is a valid prior for an infinite width network, while the former results in parameters being better identified from an inference perspective.

- We prove that this prior is appropriate for use with Hilbert space MCMC methods even in the infinite-width limit (Theorem 6), and is thus suitable for application to problems with high-dimensional state spaces owing to the inherent scalability of neural networks to its number of inputs.

The rest of this chapter is organised as follows: we firstly motivate the work in this Chapter in Section 5.2. In Section 5.3 we introduce the general inference problem, describe the canonical approximation for functions on $\mathbb{R}^d$, describe MCMC methods on an infinite-dimensional Hilbert space including their construction and the assumptions under which these methods are well-defined. Section 5.4 introduces the trace-class neural network prior and states one of our main theoretical results, showing that the proposed prior satisfies the necessary assumptions to be used with a Hilbert space MCMC algorithm. We conclude with a comparison of the Karhunen-Loève and the trace-class neural network priors in Section 5.5.

5.2 Groundwater Flow - A Bayesian Inverse Problem

To motivate the work in this Chapter, consider the following example taken from Beskos et al. (2017). The aim is to recover the permeability of an aquifer. The following PDE connects the log-permeability u of a porous medium to the hydraulic head function p:

$$\begin{aligned} -\nabla \cdot (\exp(u(x))\nabla p(x)) &= 0 \\ p(x) &= x_1 \qquad \text{if } x_2 = 0 \\ p(x) &= 1 - x_1 \qquad \text{if } x_2 = 1 \\ \frac{\partial p(x)}{\partial x_1} &= 0 \qquad \text{if } x_1 \in \{0, 1\}. \end{aligned} \tag{5.1}$$

To enforce the permeability to be positive, it is defined as the exponential of the log-permeability $u(x)$, and we define the forward operator A which maps a log-permeability u to the solution p_u of the PDE (5.1), i.e. $p_u = A(u)$, which can be evaluated at a point $x \in [0, 1]^2$ as $p_u(x) = A(u)(x)$.

We will later assume the existence of a true u^*. Consider the following (orthonormal) functions on $[0,1]^2$, defined using double indices $i = (i_1, i_2)$:

$$\varphi_i(x) = 2\cos\left(\pi\left(i_1 + \frac{1}{2}\right)x_1\right)\cos\left(\pi\left(i_2 + \frac{1}{2}\right)x_2\right). \tag{5.2}$$

The true u^* is defined as $u^*(x) = \sum_i u_i^* \varphi_i(x)$ with

$$u_i^* = \lambda_i \sin\left((i_1 - 1/2)^2 + (i_2 - 1/2)^2\right) \cdot \delta[1 \le i_1, i_2 \le 10],$$

where

$$\lambda_i^2 = \frac{1}{(\pi^2((i_1 + 1/2)^2 + (i_2 + 1/2)^2)^{1.1}}.$$

The data is simulated as follows. Taking u^*, one solves the PDE 5.1 on a 40×40 grid to get the true hydraulic head function p^*, which is observed in 33 locations $\{x_i\}_{i=1}^{33}$. The data is given by $y_i = p^*(x_i) + \varepsilon_i$, where the noise is i.i.d. $\varepsilon_i \sim \mathcal{N}(0, 0.01^2)$. Figure 5.1 shows the true permeability u^*, and the resulting hydraulic head function $p^* = A(u^*)$ with the location of the 33 observations.

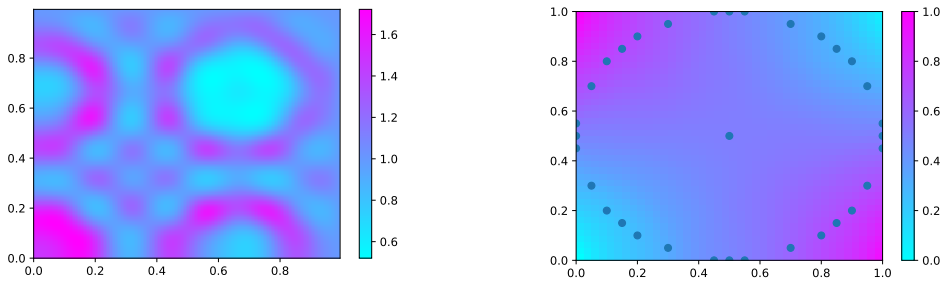


Fig. 5.1 The true permeability $\exp(u^*)$ (left), and its associated hydraulic head function with the location of the 33 observations (right).

In practice, the true log-permeability is unknown, and a statistician's task is to infer it from the measurements $\{y_i\}_{i=1}^{33}$. Within the Bayesian framework, this can be achieved by defining a prior distribution π_0 on u, and using the likelihood associated with the observations to obtain the posterior π. The reader should note that both these distributions are defined over the infinite-dimensional Hilbert space $\mathcal{H} = L^2([0,1]^2, \mathbb{R})$, the space of square-integrable functions from $[0,1]^2$ to $\mathbb{R}$. In Stuart (2010), it is shown that this leads to a well-posed inference problem, and we will discuss the general setup of Bayesian inference on infinite-dimensional Hilbert spaces in the next section, before discussing two distinct ways of

defining prior distributions on u. The first one is the Karhunen-Loève expansion arising from a Gaussian measure on $\mathcal{H}$, the second one is a trace-class neural network prior.

5.3 Problem Formulation

The objective is to sample from a target distribution π defined over an infinite dimensional separable Hilbert space.
The targets of interest in this work are Bayesian posterior distributions arising from a Gaussian prior measure π_0 and a likelihood which can be evaluated point wise. One such likelihood is a Gaussian likelihood given by observations of a PDE solution such as in Section 5.5, one for continuous control problems will be introduced in Chapter 6. In what follows, we will assume that the posterior has a density with respect to the prior, in which case the Radon-Nikodym derivative is well defined and the posterior density with respect to the prior is given by

$$\frac{d\pi}{d\pi_0}(u) = \frac{1}{Z}\exp(\ell(y|u)),$$

where ℓ is the log-likelihood, and $Z = \int \exp(\ell(y|u))\pi_0(du)$ is the normalisation constant.
Remark: For the posterior to be well-defined, the normalisation constant has to be positive and finite, i.e. $0 < Z < \infty$. For infinite-dimensional spaces, this is not generally the case even for Gaussian prior measures. For Gaussian priors, Stuart (2010) shows that the likelihood arising from Gaussian observations of forward solutions of certain PDEs (such as in the motivational example in the previous section) gives rise to a well-defined posterior, our Theorem 7 shows that the likelihood arising in stochastic control problems also results in a well-defined posterior under an appropriate Gaussian prior.

For any such infinite-dimensional separable Hilbert space $\mathcal{H}$, say $\mathcal{H} = L^2(\mathcal{X}, \mathbb{R})$ to frame the discussion in this section (or later on in Section 5.4 the sequence space $\mathcal{H} = \ell^2$), there exists an orthonormal basis $\{\varphi_i\}_{i=1}^{\infty}$ such that any element $u \in \mathcal{H}$ can be obtained as the limit $u(x) = \lim_{N\to\infty} \sum_{i=1}^{N} a_i\varphi_i(x)$, where $a_i = \langle u, \varphi_i\rangle_{\mathcal{H}}$. Let the prior $\pi_0 = \mathcal{N}(0, C)$ be a Gaussian measure on $\mathcal{H}$. If the operator C is trace-class with orthonormal eigenvalue-eigenfunction pairs $(\lambda_i^2, \varphi_i(x))$, $i = 1, 2, \ldots$, one can sample from π_0 by sampling a sequence of $\xi_i \sim \mathcal{N}(0, \lambda_i^2)$ and by then defining

$$u(x) = \sum_{i=1}^{\infty} \xi_i\varphi_i(x). \tag{5.3}$$

This is the Karhunen-Loève (KL) expansion (Giné and Nickl, 2016). One may thus think of a sample from the Gaussian measure as the sum of a sequence of 1-dimensional Gaussians with summable variances[1]. This allows us to truncate the series expansion such that we have N active terms, with the remainder giving an estimate for the approximation error:

$$u(x) = \sum_{i=1}^{N} \xi_i \varphi_i(x) + \sum_{i=N+1}^{\infty} \xi_i \varphi_i(x), \qquad \|u(x) - \sum_{i=1}^{N} \xi_i \varphi_i(x)\| \leq \sum_{i=N+1}^{\infty} \|\xi_i \varphi_i(x)\|.$$

Other more elaborate truncation schemes are discussed in Cotter et al. (2013), but we will focus on a fixed number of terms for computational and notational convenience. For some applications, φ_i for large i can be interpreted as high-oscillating functions which may not be discernible by the observation operator, see the example in Section 5.5.

We emphasise that the above discussion holds not only for the space $\mathcal{H} = L^2(\mathcal{X}, \mathbb{R})$, which is predominantly how it is applied in Beskos et al. (2017, 2008); Cotter et al. (2013), but also for $\mathcal{H} = \ell^2$, which will be of particular importance in this chapter. In infinite-dimensional spaces, one has to be careful to ensure the posterior is well defined, see Stuart (2010) for a discussion on Gaussian priors and likelihoods given through possibly non-linear mappings, observed with Gaussian noise. We will work with the following assumptions, which we prove are satisfied for the likelihood defined in Chapter 6.

1. π_0 is a Gaussian prior defined on a separable Hilbert space $\mathcal{H}$, with a trace-class covariance operator C, that is, the eigenvalues λ_i^2 corresponding to the eigenfunctions φ_i satisfy $\sum_i \lambda_i^2 < \infty$;
2. The posterior is well defined, i.e. the integral of the likelihood with respect to the prior is positive and finite.

5.3.1 A Canonical Approximation for Functions on $\mathbb{R}^d$

Consider a d-dimensional hypercube $\mathcal{X} = [0,1]^d$, the Hilbert space $\mathcal{H} = L^2(\mathcal{X}, \mathbb{R})$, and a Gaussian prior measure π on $\mathcal{H}$. A Bayesian approach entails choosing the covariance matrix C for the prior π, and we discuss a canonical choice below. If the problem requires it, as in Section 5.5 where a PDE is solved, it is possible to choose C such that the samples are almost surely differentiable.

[1] If the variances are not summable, the sum in (5.3) diverges with positive probability.

One problem with this approach is that it scales badly with dimension: say one has eigenvalues λ_i and basis functions φ_i for a 1-dimensional function, and truncates the KL expansion (5.3) after N terms. The easiest way to scale this basis up to a d-dimensional domain is by taking a tensor product of the basis, see e.g. Iserles and Nørsett (2009) for the multivariate Fourier basis, or Wojtaszczyk (1997) for Wavelets and other basis expansions. For the KL expansion, we thus get, for a multi-index $k = (k_1, \dots, k_d)$ with $k_i = 1, \dots, N$,

$$u(x) = \sum_k \xi_k \varphi_k(x) = \sum_{k_1=1}^{N} \cdots \sum_{k_d=1}^{N} \xi_{k_1,\dots,k_d} \Big(\varphi_{k_1}(x_1) \cdots \varphi_{k_d}(x_d) \Big), \tag{5.4}$$

where $\xi_{k_1,\dots,k_d} \sim \mathcal{N}(0, \lambda^2_{k_1,\dots,k_d})$ with $\lambda_{k_1,\dots,k_d}$ being a function of the respective eigenvalues λ_{k_i} capturing the correlation between dimensions. In total, there are N^d active terms, that is, the complexity is exponential in the dimension d. This will be computationally prohibitively expensive, even for moderately small d.

To circumvent the exponential cost in the domain dimension, one may want to exploit any knowledge of independence that one knows of. Assume, for example, that the function of interest can be approximated as $u(x) \approx v_0 + \sum_{i=1}^{d} u_i(x_i)$ with $u_i : [0,1] \to \mathbb{R}$, $\int u_i(x_i) dx_i = 0$. With this approximation, the number of terms to be inferred is linear in d. In practice, this is often oversimplifying. More generally, one can use approximations including higher order functions following Sobol (1993), e.g.

$$u(x) \approx \sum_{i=1}^{d} u_i(x_i) + \sum_{i=1}^{d} \sum_{j=i+1}^{d} u_{i,j}(x_i, x_j), \tag{5.5}$$

with $dN + \frac{d(d-1)}{2} N^2$ coefficients to be estimated. Using such an approximation can achieve a massive dimension reduction, avoiding the inference of all N^d coefficients, but requires good knowledge of the properties of the quantities of interest. With the approximation (5.5) in mind, one restricts oneself to the prior on finitely many random functions u_i and $u_{i,j}$, each of which themselves is sampled from a Gaussian measure $\mathcal{N}(0, C_1)$, or $\mathcal{N}(0, C_2)$, respectively. One identifies each of these functions with their Karhunen-Loève expansion

$$u_i(x_i) = \sum_{k=1}^{\infty} \xi_{i,k} \varphi_k(x_i), \quad u_{i,j}(x_{i,j}) = \sum_{k=1}^{\infty} \xi_{i,j,k} \psi_k(x_i, x_j) \tag{5.6}$$

where the φ_k and ψ_k are the eigenfunctions corresponding to the eigenvalues $\lambda^2_{\varphi,k}$ and $\lambda^2_{\psi,k}$, respectively. The $\xi_{i,k}$ and $\xi_{i,j,k}$ are independent normal random variables $\xi_{i,k} \sim \mathcal{N}(0, \lambda^2_{\varphi,k})$ and

$\xi_{i,j,k} \sim \mathcal{N}(0, \lambda^2_{\psi,k})$. As before one requires the covariance operators to be *trace-class*, and truncates the expansion (5.6) after a finite number of term.

The numerical experiments using the KL function space prior in this chapter and the next are based on the following Fourier basis functions, φ_k defined on $[0,1]$, $\psi_k = \psi_{k_1,k_2}$ defined on $[0,1]^2$ and indexed by a double index $k = (k_1, k_2)$:

$$\begin{aligned}
&\varphi_{2k}(x_i) = \sin(\pi k x_i) \\
&\varphi_{2k+1}(x_i) = \cos(\pi k x_i) \\
&\psi_{2k_1,2k_2}(x_i, x_j) = \varphi_{2k_1}(x_i)\varphi_{2k_2}(x_j) = \sin(\pi k_1 x_i)\sin(\pi k_2 x_j) \\
&\psi_{2k_1+1,2k_2}(x_i, x_j) = \varphi_{2k_1+1}(x_i)\varphi_{2k_2}(x_j) = \cos(\pi k_1 x_i)\sin(\pi k_2 x_j) \\
&\psi_{2k_1,2k_2+1}(x_i, x_j) = \varphi_{2k_1}(x_i)\varphi_{2k_2+1}(x_j) = \sin(\pi k_1 x_i)\cos(\pi k_2 x_j) \\
&\psi_{2k_1+1,2k_2+1}(x_i, x_j) = \varphi_{2k_1+1}(x_i)\varphi_{2k_2+1}(x_j) = \cos(\pi k_1 x_i)\cos(\pi k_2 x_j),
\end{aligned} \tag{5.7}$$

for $i \neq j$, with corresponding eigenvalues

$$\begin{aligned}
\lambda^2_{\varphi,2k} &= \lambda^2_{\varphi,2k+1} = \frac{1}{k^\alpha}, \\
\lambda^2_{\psi,2k_1,2k_2} = \lambda^2_{\psi,2k_1+1,2k_2} &= \lambda^2_{\psi,2k_1,2k_2+1} = \lambda^2_{\psi,2k_1+1,2k_2+1} = \frac{1}{\left(\sqrt{k_1^2 + k_2^2}\right)^\alpha}.
\end{aligned} \tag{5.8}$$

See Figure 5.2 for some representative draws from this prior, which is a modification from the prior used in Section 4.2 of Beskos et al. (2017). The covariance operator is of the form $-\Delta^{-\alpha}$ where Δ denotes the Laplacian, and we allow both Dirichlet and Neumann boundary conditions, unlike the authors in Beskos et al. (2017) who consider Dirichlet boundary conditions only.

Approximations such as (5.5) require a good understanding of the functions of interest, which is generally overly restrictive: the statistician or researcher needs to have a good understanding about which coefficients of the eigen expansion are informed by the likelihood, and should therefore be included in the analysis. Section 5.4 will introduce a prior which scales favourably with the domain-dimension as it does not require pre-defining an orthogonal basis.

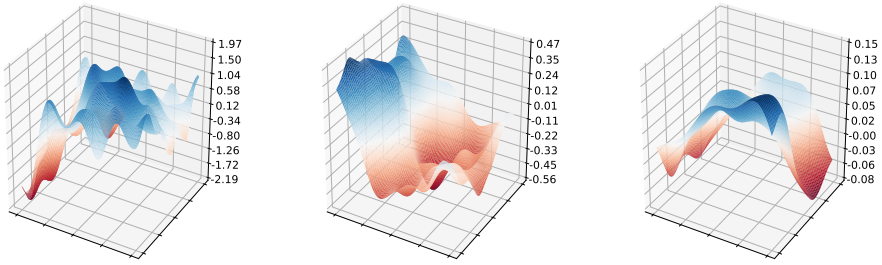


Fig. 5.2 Three samples from the Karhunen-Loève prior; the basis functions are the two-dimensional Fourier functions. In ascending order from left to right we set $\alpha \in \{1.001, 1.5, 3\}$ with the eigenvalues scaling as $\lambda_k^2 \propto 1/(k_1^2 + k_2^2)^\alpha$, for the double index $k = (k_1, k_2)$. The tuning parameter α controls the smoothness of the samples.

5.3.2 Algorithms on Hilbert Spaces

This section recapitulates how to define 'sensible' Metropolis-Hastings Markov chain Monte Carlo algorithms for inference over the ξ_i in (5.3). The use of Markov chains is a popular approach to sample from distributions on finite-dimensional state spaces (see Brooks et al. (2011) for an overview of MCMC methods). Here, we review algorithms which can theoretically deal with arbitrarily many basis coefficients, without the usual problem of the acceptance probability degenerating as one includes more coefficients. This property, known as *stability under mesh refinement*, is not satisfied by the popular Random Walk Metropolis algorithm (RWMH, Hastings (1970)), or by the Metropolis Adjusted Langevin Algorithm (MALA, Roberts et al. (1996)).

Two algorithms which are both dimension independent are the preconditioned Crank-Nicolson (pCN) and the preconditioned Crank-Nicolson Langevin (pCNL) algorithms, the former introduced as early as Neal (1998) and both derived and discussed in Cotter et al. (2013), see also Beskos et al. (2017, 2008); Rudolf and Sprungk (2018) for further reading and generalisations. Motivated by the idea of increasing dimensions translating to evaluating a function on a finer mesh, we refer to the dimension independence of these algorithms as stability under mesh refinement. Both algorithms can be seen as a discretisation of the following stochastic partial differential equation:

$$\frac{du}{ds} = -\mathcal{K}(C^{-1}u - \gamma D\ell(u)) + \sqrt{2\mathcal{K}}\frac{dB}{ds}, \tag{5.9}$$

where $D\ell$ is the Fréchet derivative of the log-likelihood[2], $\mathcal{K}$ is a preconditioner, C is the covariance operator of the Gaussian prior measure, B is a Brownian motion, and γ a tuning parameter: if $\gamma = 0$, the invariant distribution of (5.9) is the prior π_0, and for $\gamma = 1$ the invariant distribution is the posterior π. With the choice $\mathcal{K} = C$ (the preconditioned case, such that the dynamics are scaled to the prior variances), discretising (5.9) using a Crank-Nicolson scheme results in pCN (for $\gamma = 0$) and pCNL (for $\gamma = 1$). The resulting discretisations can be simplified to

$$v = \sqrt{1-\beta^2}u + \beta w, \quad w \sim \mathcal{N}(0, C), \qquad \text{(pCN)} \tag{5.10}$$

$$v = \frac{1}{2+\delta}\left[(2-\delta)u + 2\delta C\mathcal{D}\ell(u) + \sqrt{8\delta}w\right], \quad w \sim \mathcal{N}(0, C), \qquad \text{(pCNL)} \tag{5.11}$$

for step sizes $\beta \in (0, 1]$ and $\delta \in (0, 2)$, respectively. Note that due to the discretisation scheme used, pCN is prior-reversible, and using it as a proposal in a Metropolis-Hastings sampler to target the posterior, the proposal is accepted with probability $\min\{1, \exp(-\ell(u) + \ell(v))\}$. If the pCNL dynamics are used as a proposal for a MH scheme, the acceptance probability is given by $\min\{1, \exp(\rho(u, v) - \rho(v, u))\}$ where

$$\rho(u, v) = -\ell(u) - \frac{1}{2}\langle v - u, \mathcal{D}\ell(u)\rangle - \frac{\delta}{4}\langle u + v, \mathcal{D}\ell(u)\rangle + \frac{\delta}{4}\| \sqrt{C}\mathcal{D}\ell(u)\|^2.$$

Note that pCN and pCNL are special cases of Algorithm 4 with

$$Q(u, \cdot) = \mathcal{N}\left(\sqrt{1-\beta^2}u, \beta^2 C\right) \qquad \text{(pCN)}$$

and

$$Q(u, \cdot) = \mathcal{N}\left(\frac{2-\delta}{2+\delta}u + \frac{2\delta}{2+\delta}C\mathcal{D}\ell(u), \frac{8\delta}{(2+\delta)^2}C\right). \qquad \text{(pCNL)}$$

Both pCN and pCNL are such that, for an uninformative likelihood, all moves are accepted. In practice, the likelihood Assumptions 3 and 4 ensure that, unlike RWMH or MALA, neither pCN nor pCNL require their step size β or δ to go to 0 as one includes more coefficients in the KL expansions (Cotter et al., 2013).

For mathematical completeness, we emphasise that in order to ensure that the processes arising from the above defined transition kernels have the desired stationary distribution, one

[2]Note that we use the log-likelihood ℓ rather than the potential $-\ell$ as the authors of Cotter et al. (2013).

needs to check that they yield a strong aperiodic, recurrent Harris chain (giving the existence of a unique stationary distribution (Athreya and Ney, 1978)), and satisfy the detailed balance condition (showing that the stationary distribution is the one of interest (Tierney et al., 1998)).

To conclude this section, we state the assumptions (Cotter et al., 2013, Assumptions 6.1) under which both pCN (Cotter et al., 2013, Thm 6.2) and pCNL (see Theorem 9 in the appendix) are well defined, where Assumption 5 is only required for pCNL (Beskos et al., 2017):

3. There exist constants $K > 0$, $p > 0$ such that $0 \leq -\ell(y|u) < K(1 + \|u\|^p)$ holds for all $u \in \mathcal{H}$;

4. $\forall r > 0\ \exists K(r) > 0$ such that for all u, v with $\max(\|u\|, \|v\|) < r$:

$$|\ell(y|u) - \ell(y|v)| \leq K(r)\|u - v\|;$$

5. $\forall u \in \mathcal{H}$: $C\mathcal{D}\ell(u) \in \mathrm{Im}(C^{1/2})$, that is, the *preconditioned* differential operator is in the support of the prior with probability 1.

5.4 Trace-Class Neural Network Priors

The Gaussian prior on $\mathcal{H} = L^2(\mathcal{X}, \mathbb{R})$ exploits the isometry between the function space $L^2(\mathcal{X}, \mathbb{R})$ and the sequence space ℓ^2 using the Karhunen-Loève expansion (Giné and Nickl, 2016), but the computational complexity of using a basis-expansion on a high-dimensional domain is unfeasible even when using approximate function representations such as in Sobol (1993).

Neural networks have shown excellent empirical performance in high-dimensional function regression tasks, and Bayesian neural networks (BNNs) use their architecture to define priors over such functions. BNNs are popular as they empirically show good results, and are computationally fast. While some theoretical results are known (Hornik, 1991; Matthews et al., 2018), the interpretability of the posteriors arising from Bayesian neural networks are limited, and priors are often understood as a regularisation method in optimisation (Welling and Teh, 2011), rather than the actual prior belief one has on the weights; see also Lipton (2018) for a broader discussion of interpretability.

We now propose a prior that is also defined over the parameters θ of the neural network, but is, in contrast to existing priors for neural networks, well-defined in the infinite-width limit. As inference is done over the parameters which form a countable sequence, the Hilbert space of interest is $\mathcal{H} = \ell^2$, the space of square-summable sequences, for which the dimension-independent MCMC methods discussed in Section 5.3.2 are applicable. Through the architecture of the neural network, the prior π_0 over the parameters implicitly defines a prior on the output function of the neural network. Under mild assumptions on the network architecture, and if $\mathcal{X}$ is compact, the output functions v_θ are π_0-almost surely square-integrable over $\mathcal{X}$, and the prior thus naturally defines a prior over $L^2(\mathcal{X}, \mathbb{R})$ as well. The proposed prior is also more flexible than the Karhunen-Loève expansion of a Gaussian measure: one needs not to specify a covariance operator and find its eigenfunctions. By giving up the orthogonality of these eigenfunctions which allow for a rich theoretical analysis, one gains on the performance side. We coin the term *trace-class neural network prior* to emphasise that the prior leads to a well-defined function space prior if the variances of all parameters are appropriately summable. The term is well-established for Gaussian measures, where these are called trace-class if the eigenvalues of the covariance operator are summable.

To set the scene, consider a n-layer feed-forward fully-connected neural network. Let $\alpha > 1$ be a fixed constant, and let $\sigma_{w_l}^2, \sigma_{b_l}^2 \in \mathbb{R}^+$ for $l = 1 \dots n+1$. The layer width of layer l is N^l, the input is $x \in [0,1]^d$, and the output is $u(x) = f_1^{(n+1)}(x) \in \mathbb{R}$; for notational convenience we write $N^0 = d$ and $N^{n+1} = 1$. The network is described fully by the set of weights and biases,

$$w = \left\{ w_{i,j}^{(l)} \right\}_{i=1,j=1,l=1}^{N^l, N^{l-1}, n+1}, \qquad b = \left\{ b_i^{(l)} \right\}_{i=1,l=1}^{N^l, n+1}, \qquad \theta = (w, b), \tag{5.12}$$

where we have summarised w and b as θ. The prior π_0 is now defined as follows: the individual weights and biases in each layer l are independent and normally distributed, and we emphasise here that the novelty is to choose the variances not uniformly, but to decrease them as one moves into the tail nodes:

$$\begin{aligned} W_{i,j}^{(1)} &\sim \mathcal{N}\left(0, \frac{\sigma_{w^{(1)}}^2}{i^\alpha}\right), \\ W_{i,j}^{(l)} &\sim \mathcal{N}\left(0, \frac{\sigma_{w^{(l)}}^2}{(ij)^\alpha}\right) \quad \text{for } l = 2 \dots n+1, \\ B_i^{(l)} &\sim \mathcal{N}\left(0, \frac{\sigma_{b^{(l)}}^2}{i^\alpha}\right), \end{aligned} \tag{5.13}$$

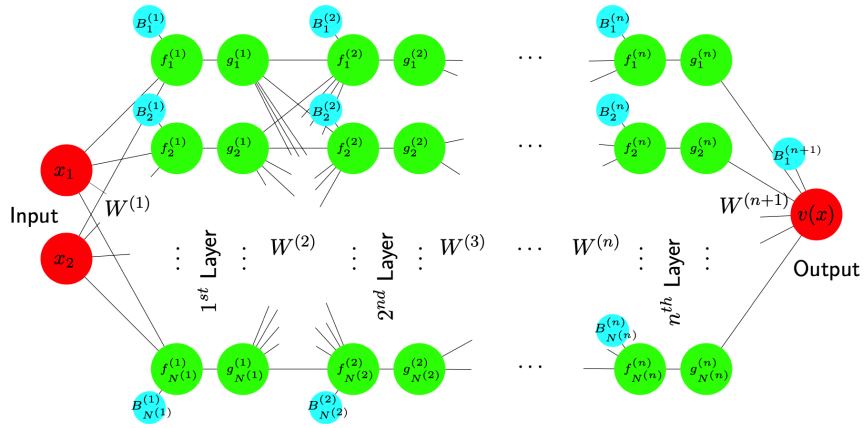


Fig. 5.3 A n-layer feed-forward neural network, used to define a function $u : \mathbb{R}^2 \to \mathbb{R}$. Note that $g_i^{(l)} = \varphi(f_i^{(l)})$.

where i, j, and l range over the obvious indices, cf. (5.12). The reader should note that the prior is invariant with respect to permutation of the input variables, thus avoiding preferential treatment of any of the inputs.

Given an *activation function* $\varphi : \mathbb{R} \to \mathbb{R}$, we define the random functions in the respective layers by

$$\begin{aligned} f_i^{(1)}(x) &= b_i^{(1)} + \sum_{j=1}^{d} w_{i,j}^{(1)} x_j, \qquad i = 1 \dots N^1 \\ f_i^{(l)}(x) &= b_i^{(l)} + \sum_{j=1}^{N^{l-1}} w_{i,j}^{(l)} \varphi(f_j^{(l-1)}(x)), \qquad i = 1 \dots N^l,\ l = 2 \dots n \\ u(x) = f_1^{(n+1)}(x) &= b_1^{(n+1)} + \sum_{j=1}^{N^n} w_{1,j}^{(n+1)} \varphi(f_j^{(n)}(x)), \end{aligned} \tag{5.14}$$

see Figure 5.3 for an illustration.

The tuning parameter α controls how much information one believes concentrates on the first nodes. If $\alpha > 1$ we refer to the prior as *trace-class*, coining the term *trace-class neural network priors*. If one believes that many nodes are important, one should choose α close to 1, larger values of α result in strong concentration of information on the first nodes. See Figure 5.4 for three representative draws from the neural network prior. As the next theorem will show, this allows indeed to define an infinitely wide network by taking $N^l = \infty$, and the variances can be summarised in a diagonal covariance operator C; this prior

is well-defined on an infinite-dimensional Hilbert space (isometric to ℓ^2), and can thus be used in the algorithms from Section 5.3. In practice, one truncates the number of nodes within each layer as for the priors described before, or one may randomly switch nodes on and off similarly to the random truncation prior used in Cotter et al. (2013).

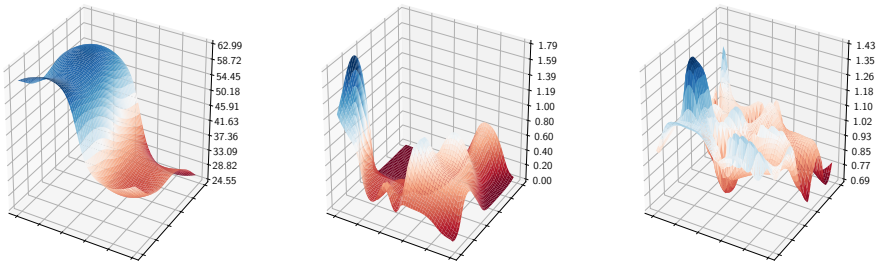


Fig. 5.4 Three samples from the trace-class neural network prior, for a network with 3 fully-connected layers using Tanh activation functions with 100 nodes per layer; inputs are x_1, x_2, $\sin(\pi x_1)$, and $\sin(\pi x_2)$ (as in Example 5.5). Tuning parameters are set to $\alpha = 2$, $\sigma^2_{w^{(l)}} = \sigma^2_{b^{(l)}} = 1$ for all $l = 1 \dots n+1$ (left), $\alpha = 1.0001$, $\sigma^2_{w^{(l)}} = \sigma^2_{b^{(l)}} = 5$ for all $l = 1 \dots n+1$ (centre), $\alpha = 1.001$, $\sigma^2_{w^{(l)}} = \sigma^2_{b^{(l)}} = 10$ for all $l = 1 \dots n$, $\sigma^2_{w^{(n+1)}} = \sigma^2_{b^{(n+1)}} = 1/30$ (right). Note the difference in the magnitudes on the z-axis.

In what follows, we will often write $u(x) = u_\theta(x)$ to emphasise the dependence of the function samples on the weights and biases. In order to be able to interpret the samples, we generally want the prior to satisfy the following desirable properties:

6. $\forall x \in [0,1]^d$ one has $|f_i^{(l)}(x)| < \infty$ π_0-almost surely, $\mathbb{E} f_i^{(l)}(x) = 0$, and $\exists\ \sigma_l^2$ such that $\mathbb{E}\left[(f_i^{(l)}(x))^2\right] < \sigma_l^2 / i^\alpha$, where the expectation is taken with respect to the prior on the parameters of the neural network; in particular this holds for $u(x) = f_1^{(n+1)}(x)$; this assumption ensures that the prior is well-defined;

7. $\forall x, y \in [0,1]^d \exists\ c_l \geq 0$ such that $\mathbb{E}\left[\left(f_i^{(l)}(x) - f_i^{(l)}(y)\right)^2\right] \leq c_l \|x-y\|^2 / i^\alpha$, with the expectation again taken with respect to the prior; in particular this gives $\mathbb{E}\left[u(x) - u(y)\right]^2 \leq c_{n+1}\|x-y\|^2$; this assumption ensures that the functions one samples behave nicely, and that the output function u is sufficiently smooth;

8. $\partial u(x)/\partial x \neq 0$ π_0-almost surely, i.e. information from the input layer informs the output layer.

We now state a theorem which shows that the proposed prior satisfies the desired properties of a prior. To this end, we use an activation function[3] $\varphi : \mathbb{R} \to \mathbb{R}$ which satisfies the following condition:

9. φ is Lipschitz continuous with Lipschitz constant 1 and $\varphi(0) = 0$. In particular this implies $\forall x \in \mathbb{R}$: $|\varphi(x)| < |x|$.[4] Furthermore, this implies that φ is differentiable almost everywhere, with the derivative being essentially bounded by 1.

Theorem 6. *Under Assumption 9, if $\sigma^2_{w^{(l)}} > 0$ for all l, and if the neural network is* trace-class *(i.e. $\alpha > 1$), the prior is well-defined in the limit $N^{(l)} \to \infty$ for all l, and satisfies Properties 1, 6, and 7. Property 8 holds if additionally, $\varphi(x) = 0$ only on a Lebesgue-nullset.*

The proof can be found in Appendix 5.6.1.

Deep limit and the choice of tuning parameters

We briefly comment on the behaviour of the trace-class neural network prior in the deep limit, and on the choice of the tuning parameters α, $\sigma^2_{w^{(l)}}$, and $\sigma^2_{b^{(l)}}$. From the proof of Theorem 6, in particular equations (5.25) and (5.30), it is obvious that these tuning parameters have to depend on the numbers of layers used as one considers the deep limit $n \to \infty$. The same observation will be made when considering Theorem 7, where equation (6.35) requires a similar scaling of these tuning parameters in the deep limit. For the remainder of this thesis, we will focus on finite layer depths.

5.4.1 Identifiability Issues and Remedies

It is well-known that the output function of a standard neural network does not depend on the labeling of functions within each layer. However, unlike a prior that has uniform variances within each layer, swapping $f^{(l)}_i$ and $f^{(l)}_{i+1}$ (effectively by swapping their corresponding weights and biases) will lead from θ to a new θ' such that $\pi_0(\theta) \neq \pi_0(\theta')$, and thus avoid the label-switching problem. To facilitate faster mixing by allowing jumps between these different configurations, we propose Algorithm 8. The algorithm is in particular useful when

[3] As will be clear from the proof of Theorem 6, one may use different activation functions at different layers, which will then all have to satisfy this assumption.

[4] The generalisation to arbitrary Lipschitz constants and the implication $\exists\, c > 0$ such that $\forall x \in \mathbb{R}$: $|\varphi(x)| < c|x|$ is straightforward.

using likelihood-informed MCMC algorithms: the likelihood gradients help 'selecting' the functions that are important, and then following with a swapping step ensures that they have high prior mass as well. The posterior is invariant with respect to the transition kernel, as the likelihood does not depend on the labelling of the nodes, and as the proposal is symmetric:

$$a(\theta, \theta') = \frac{\pi_0(\theta')}{\pi_0(\theta)} \frac{\mathcal{L}(\theta')}{\mathcal{L}(\theta)} \frac{q(\theta|\theta')}{q(\theta'|\theta)} = \frac{\pi_0(\theta')}{\pi_0(\theta)}. \tag{5.15}$$

Algorithm 8

procedure NODE SWAP(θ) ▷ Input current iterate θ
  $\theta' \leftarrow \theta$
  $l \sim \mathcal{U}(\{1, \dots, n\})$ ▷ Sample random layer
  $i \sim Geom(\alpha^{-1})$ ▷ Sample random node
  while $i \geq N_l$ **do** ▷ Repeat process until we have a valid node index
    $i \sim Geom(\alpha^{-1})$
  end while
  $w^{(l)\prime}_{i+1,j} \leftarrow w^{(l)}_{i,j}$
  $w^{(l)\prime}_{i,j} \leftarrow w^{(l)}_{i+1,j}$
  $w^{(l+1)\prime}_{i,j+1} \leftarrow w^{(l+1)}_{i,j}$
  $w^{(l+1)\prime}_{i,j} \leftarrow w^{(l+1)}_{i,j+1}$
  $b^{(l)\prime}_{i+1} \leftarrow b^{(l)}_{i}$
  $b^{(l)\prime}_{i} \leftarrow b^{(l)}_{i+1}$
  $u \sim Unif([0,1])$
  $a = \min(1, \pi_0(\theta')/\pi_0(\theta))$ ▷ Metropolis-Hastings acceptance probability cf. (5.15)
  if u<a **then**
    return θ' ▷ Accept node swap
  else
    return θ ▷ Reject node swap
  end if
end procedure

5.5 Illustrative Groundwater Flow Example

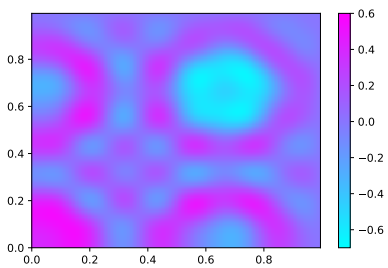
We now return to the Bayesian inverse problem introduced in Section 5.2.

The code for the examples in this section is available online at https://github.com/TorbenSell/trace-class-neural-networks and is similar to the setup in Beskos et al. (2017)[5].

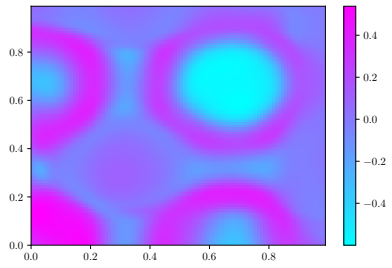
[5]While we could not perfectly replicate their results, we aimed to stick as close to their results as possible.

5.5.1 Ability to Approximate Complicated Functions

The first aim is to show that the trace-class neural network prior is able to visually recover relatively complicated functions. To this end we define the function $u^* : [0,1]^2 \to \mathbb{R}_+$ as in Section 5.2, and observe this function on a 20×20 grid with independent Gaussian noise $\mathcal{N}(0, 0.01^2)$. For the neural network prior we use three hidden layers with 10 nodes per layer, Tanh activation function, and used a four dimensional input space with the inputs $(x_1, x_2, \sin(\pi x_1), \sin(\pi x_2))$. We set the tuning parameters to $\alpha = 1.001$, $\sigma_{w_1}^2 = \sigma_{b_1}^2 = 100$, $\sigma_{w_2}^2 = 1/30$, and $\sigma_{b_1}^2 = 1/10$, and ran the preconditioned Crank-Nicolson algorithm on the parameters of the neural network. As Figure 5.5 shows, the neural network prior is able to approximate the true u^* when given many, in this example 400, observations.



(a) True log-permeability u^*.



(b) Mean estimate obtained using the NN-based posterior.

Fig. 5.5 The neural network estimates the true $u^*(x)$ which is noisily observed on every grid point x of a 20×20 grid. In real applications, only few observations will be available, this example simply illustrates that many observations lead to close approximations for the trace-class neural network prior.

5.5.2 Groundwater Flow - A Bayesian Inverse Problem

We now compare two priors when aiming to infer the unknown function u^* in the setup described in Section 5.2. The first one is the trace-class neural network prior described, with the same choice of tuning parameters as before in Section 5.5.1. The second prior is a Gaussian measure on $[0,1]^2$ with the following orthonormal basis and corresponding

eigenvalues defined using double indices $i = (i_1, i_2)$:

$$\begin{aligned} \varphi_i(x) &= 2\cos\left(\pi\left(i_1 + \frac{1}{2}\right)x_1\right)\cos\left(\pi\left(i_2 + \frac{1}{2}\right)x_2\right), \\ \lambda_i^2 &= \frac{1}{(\pi^2\left((i_1 + 1/2)^2 + (i_2 + 1/2)^2\right)^{1.1}}. \end{aligned} \tag{5.16}$$

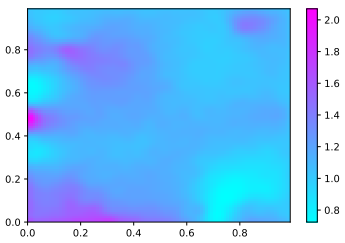
Note that the basis functions $\varphi_i(x)$ and the eigenvalues λ_i^2 are the same ones used to define the true log-permeability u^*, and the experiment is thus tailored to good performance when using this prior. In the experiments, we truncated the basis expansion using $1 \leq i_1, i_2 \leq 25$, which gives a similar number of parameters as we used in the neural network example. The true u^* used to simulate data is the same as in Section 5.2, and the log-likelihood is given by

$$\ell(y|u) = -\frac{1}{2\sigma^2}\left(\sum_{i=1}^{33}(y_i - A(u)(x_i))^2\right)$$

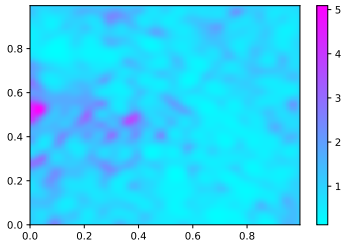
with $\sigma = 0.01$, and $A(u)(x)$ being the forward solution of the PDE (5.1) evaluated at location x.

Both experiments used a similar number of iterations and stored 1000 MCMC samples to obtain the mean estimates in Figure 5.6, the figure also includes representative prior and posterior samples. Convergence diagnostics in the form of traceplots of the respective Markov chains are displayed in Figure 5.7, and visual posterior predictive checks are shown in Figure 5.8.

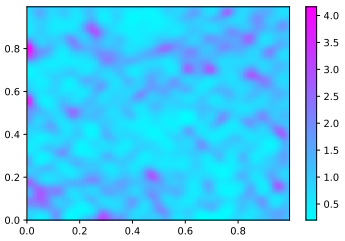
The results of this Example are worth a short discussion. As evident from the choice of the eigenfunction and eigenvalues of the Gaussian prior, and the specific form of the true log-permeability, the Gaussian posterior can be expected to perform reasonably well even in the presented setting of a fairly uninformative likelihood that uses only 33 data points and a highly non-linear forward operator. The trace-class neural network posterior can thus not be expected to compete on the performance level, yet the obtained mean estimate is visually not 'worse' than the one from the posterior which uses the Gaussian prior.
The posterior samples displayed in Figure 5.6 are very different, and preference of one or the other prior depends on what log-permeabilities may be expected in the problem at hand: largely connected regions of different materials suggest the use of the trace-class neural network prior; a surface of many different materials that are roughly mixed suggest to use the Gaussian prior.
Lastly, this example was mainly to motivate the discussion, and show that the trace-class



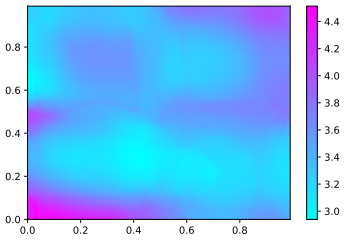
(a) Posterior mean estimate, using the Gaussian prior.



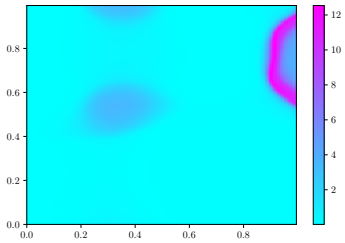
(b) Sample from Gaussian prior.



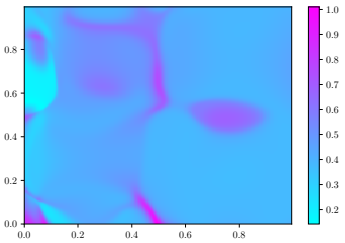
(c) Sample from posterior using the Gaussian prior.



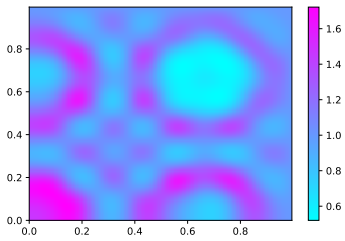
(d) Posterior mean estimate, using the NN prior.



(e) Sample from trace-class NN prior.



(f) Sample from posterior using the trace-class NN prior.



(g) True permeability $\exp(u^*)$.

Fig. 5.6 *Top row:* The mean estimate for $\exp(u)$ obtained using pCN for the Gaussian prior (a), a sample from the KL prior (b), and a sample from the KL posterior (c). *Middle row:* The mean estimate for $\exp(u)$ obtained using pCN for the neural network prior (d), a sample from the trace-class neural network prior (e), and a sample from the trace-class neural network posterior (f). *Bottom row:* The true permeability $\exp(u^*)$ (g), which was estimated by the means in tiles (a) and (d).

neural network prior does not yield completely unfeasible estimates in this example tailored to the Gaussian prior. The strength of the trace-class neural network prior only shows when targeting functions on higher-dimensional domains, which was discussed in Section 5.4 and will be validated empirically in Section 6.4 of the next chapter.

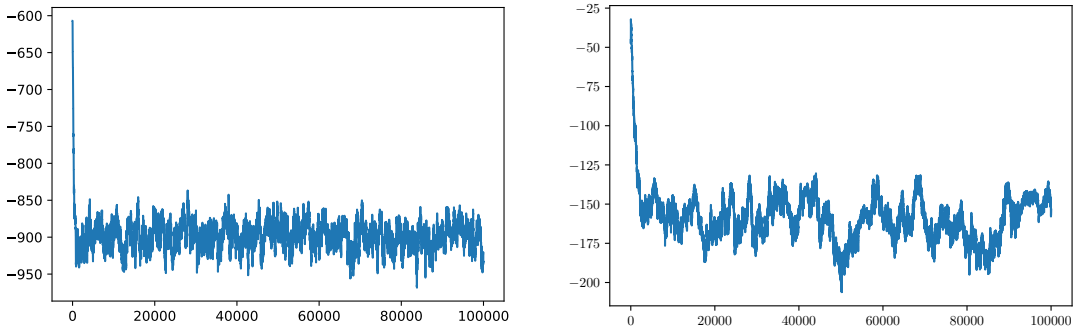
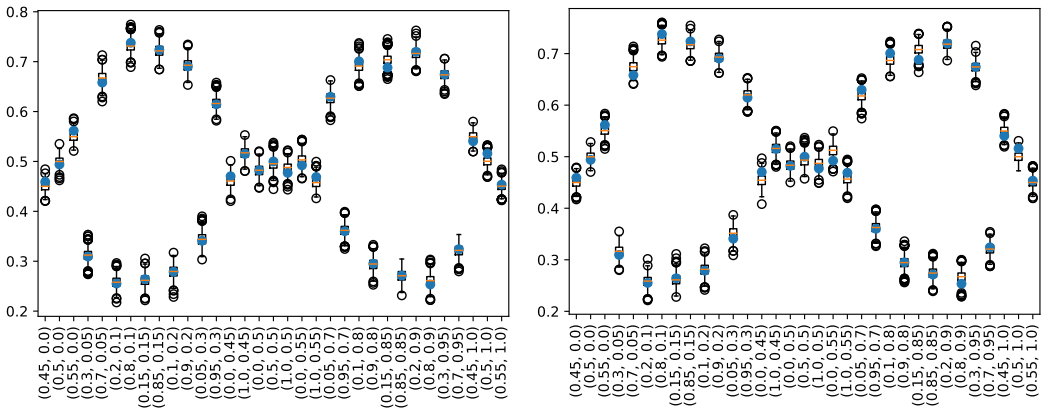


Fig. 5.7 On the left the log-posterior values for the first 100,000 samples from the KL posterior when running the pCN algorithm. On the right the first 100,000 log-posterior values when running the pCN algorithm on the posterior distribution over the parameters of the trace-class neural network.



(a) KL posterior predictive.

(b) NN posterior predictive.

Fig. 5.8 Visual posterior predictive check for both the KL- and NN-based posteriors. The observed values at each of the 33 observation locations (see figure 5.6) are shown as a blue dot, the box plots are 100 samples from the posterior predictive distribution (Gelman et al., 2013, Section 6.3). Both posteriors show similar predictive performance indicating that they arise from similarly well-suited priors.

5.6 Appendix

5.6.1 Proof of Theorem 6

Proof of Theorem 6. We firstly show assumption 1: Let the Hilbert space $\mathcal{H}$ be the set

$$\mathcal{H} = \left\{ \theta = (w, b) : \sum_{l=1}^{n+1} \sum_{i=1}^{\infty} \left((b_i^{(l)})^2 + \sum_{j=1}^{\infty} (w_{i,j}^{(l)})^2 \right) < \infty \right\} \tag{5.17}$$

equipped with the inner product

$$\langle \theta, \tau \rangle = \langle (w, b), (\varpi, \beta) \rangle = \sum_{l=1}^{n+1} \sum_{i=1}^{\infty} \left(b_i^{(l)} \beta_i^{(l)} + \sum_{j=1}^{\infty} w_{i,j}^{(l)} \varpi_{i,j}^{(l)} \right).$$

This Hilbert space can be interpreted as the l^2 sequence space, and equivalently as the direct sum of countably many l^2 sequence spaces $\mathcal{H}_i^{(l)} = \{(w_{i,j}^{(l)}, b_i^{(l)}) : (b_i^{(l)})^2 + \sum_{j=1}^{\infty} (w_{i,j}^{(l)})^2 < \infty\}$. We also define the layer-wise Hilbert spaces $\mathcal{H}^{(l)}$ as $\mathcal{H}^{(l)} = \{(w_{i,j}^{(l)}, b_i^{(l)}) : \sum_{i=1}^{\infty} \left[(b_i^{(l)})^2 + \sum_{j=1}^{\infty} (w_{i,j}^{(l)})^2 \right] < \infty\}$.

We now show that any sample from the prior (5.13) is in the Hilbert space $\mathcal{H}$ almost surely. To see this, let (W, B) be a draw from the prior. We define the random variables

$$S_k^{(l)} = \sum_{i=1}^{k} \left((B_i^{(l)}) + \sum_{j=1}^{k} (W_{i,j}^{(l)}) \right). \tag{5.18}$$

It's easy to check that $S_k^{(l)}$ is a L^2-bounded martingale if

$$\sum_{i=1}^{\infty} \mathbb{E}\left[(B_i^{(l)})^2 \right] < \infty \quad \text{and} \quad \sum_{i,j=1}^{\infty} \mathbb{E}\left[(W_{i,j}^{(l)})^2 \right] < \infty,$$

which is the case for our choice of the prior, as long as $\alpha > 1$. Therefore $S_k^{(l)}$ converges almost surely to some limit $S^{(l)} \in L^2$ by the L^2-martingale convergence theorem, so the associated sequence of weights and biases $(W_{i,j}^{(l)}, B_i^{(l)})_{i,j=1..\infty} \in \mathcal{H}^{(l)}$. As we have only finitely many layers, this implies $(W, B) \in \mathcal{H}$ as required.
To see the second part of assumption 1, note that the sequences e_1, e_2, ... form a basis of l^2, and therefore of $\mathcal{H}$, and the summability of the variances (given if $\alpha > 1$) ensures that the prior covariance operator is indeed trace-class.

To see Assumption 6, by looking at the first layer we can easily check that for fixed $x \in [0,1]^d$, $f_i^{(1)}(x)$ is a mixture of centered Gaussian distributions, and the claim follows by noting that $\mathbb{E}B_i^{(1)} = \mathbb{E}W_{i,j}^{(1)} = 0$,

$$\begin{aligned}\mathbb{E}\Big[(f_i^{(1)}(x))^2\Big] &= \mathbb{E}\Big[(B_i^{(1)})^2\Big] + \sum_{j=1}^{d} \mathbb{E}\Big[(W_{i,j}^{(1)})^2\Big](x_j)^2 \\ &\leq \frac{\sigma_{b_1}^2}{i^\alpha} + \frac{\sigma_{w_1}^2}{i^\alpha} d \qquad (5.19) \\ &= \frac{1}{i^\alpha}\Big[\sigma_{b_1}^2 + \sigma_{w_1}^2 d\Big]. \qquad (5.20)\end{aligned}$$

We use induction over l, and define the following random variables, for which we truncate the i-th function of layer l after k terms:

$$f_{i,k}^{(l)}(x) = B_i^{(l)} + \sum_{j=1}^{k} W_{i,j}^{(l)} \varphi(f_j^{(l-1)}(x)) \qquad l = 2 \dots n+1.$$

It will be useful to note that by Assumption 9 and the induction hypothesis, we get

$$|\varphi(f_j^{(l-1)}(x))| \leq |(f_j^{(l-1)}(x))| < \infty. \qquad (5.21)$$

Furthermore, from the above and using Jensen's inequality, we also obtain

$$|\mathbb{E}\varphi(f_j^{(l-1)}(x))| \leq \mathbb{E}[|\varphi(f_j^{(l-1)}(x))|] \leq \mathbb{E}[|f_j^{(l-1)}(x)|] < \infty, \qquad (5.22)$$

where the last inequality holds as $f_j^{(l-1)}(x)$ is L^2 bounded by the induction hypothesis.

We now show that $f_{i,k}^{(l)}(x) \to f_i^{(l)}(x)$ almost surely, and in L^2, by applying the L^2 martingale convergence theorem. We thus need to show that $S_k(x) := f_{i,k}^{(l)}(x)$ is a L^2 bounded martingale, where we dropped the indices i and l for notational convenience. Indeed, with the natural filtration $(\mathcal{F}_k)_{k\in\mathbb{N}}$

$$\mathbb{E}[S_{k+1}(x)|\mathcal{F}_k] = 0,$$

as $W_{i,j}^{(l)}$ and $\varphi(f_j^{(l-1)}(x))$ are independent, the expectation of the former is centered, and the latter is finite. Additionally, by exploiting the independence, Assumption 9 and 5.22, we get

$$\mathbb{E}\left[(S_k(x))^2\right] = \mathbb{E}[B_i^{(l)}]^2 + \sum_{j=1}^{k} \mathbb{E}\left[(W_{i,j}^{(l)})^2\right] \mathbb{E}\left[(\varphi(f_j^{(l-1)}(x)))^2\right] \tag{5.23}$$

$$\leq \frac{\sigma_{b^{(l)}}^2}{i^\alpha} + \sigma_{w^{(l)}}^2 \sum_{j=1}^{k} \frac{1}{(ij)^\alpha} \mathbb{E}\left[(f_j^{(l-1)}(x))^2\right]$$

$$\leq \frac{\sigma_{b^{(l)}}^2}{i^\alpha} + \frac{\sigma_{w^{(l)}}^2 \sigma_{l-1}^2}{i^\alpha} \sum_{j=1}^{k} \frac{1}{j^{2\alpha}} \tag{5.24}$$

$$= \frac{1}{i^\alpha} \left[\sigma_{b^{(l)}}^2 + \sigma_{w^{(l)}}^2 \sigma_{l-1}^2 \sum_{j=1}^{k} \frac{1}{j^{2\alpha}} \right]. \tag{5.25}$$

This series converges for $\alpha > 1$, and we define the limit for $i = 1$ as σ_l^2. Thus, S_k is indeed a L^2 bounded martingale and trivially $\mathbb{E} f_i^{(l)} = 0$, proving Assumption 6.

It remains to show Assumption 7. For the first layer, we use independence to get

$$\mathbb{E}\left[(f_i^{(1)}(x) - f_i^{(1)}(y))^2\right] = \sum_{j=1}^{d} \mathbb{E}\left[(W_{i,j}^{(1)})^2\right] (x_j - y_j)^2$$

$$= \frac{\sigma_{w_1}^2}{i^\alpha} \|x - y\|^2. \tag{5.26}$$

For the subsequent layers, we again use induction over l. We define $S_k(x)$ as before and check that

$$\mathbb{E}\left[(S_k(x) S_k(y))^2\right] = \mathbb{E}\left[(B_i^{(l)})^2\right] + \sum_{j=1}^{k} \mathbb{E}\left[(W_{i,j}^{(l)})^2\right] \mathbb{E}[\varphi(F_j^{(l-1)}(x)) \varphi(F_j^{(l-1)}(y))]. \tag{5.27}$$

Using the induction hypothesis, Assumption 9, (5.23) and (5.27) we get

$$\begin{aligned}
\mathbb{E}\Big[(S_k(x) - S_k(y))^2\Big] &= \mathbb{E}\Big[(S_k(x))^2\Big] + \mathbb{E}\Big[(S_k(y))^2\Big] - 2\mathbb{E}[S_k(x)S_k(y)] \\
&= 2\mathbb{E}\Big[(B_i^{(l)})^2\Big] + \sum_{j=1}^{k} \mathbb{E}\Big[(W_{i,j}^{(1)})^2\Big]\Big(\mathbb{E}[\varphi(f_j^{(l-1)}(x))^2] + \mathbb{E}[\varphi(f_j^{(l-1)}(y))^2]\Big) \\
&\qquad - \sum_{j=1}^{k} 2E[S_k(x)S_k(y)] \qquad (5.28) \\
&= \sigma_{w^{(l)}}^2 \sum_{j=1}^{k} \frac{1}{(ij)^\alpha} \mathbb{E}\Big[(\varphi(f_j^{(l-1)}(x)) - \varphi(f_j^{(l-1)}(y)))^2\Big] \\
&\leq \sigma_{w^{(l)}}^2 \sum_{j=1}^{k} \frac{1}{(ij)^\alpha} \mathbb{E}\Big[(f_j^{(l-1)}(x) - f_j^{(l-1)}(y))^2\Big] \\
&\leq \frac{\sigma_{w^{(l)}}^2 c_{l-1}}{i^\alpha} \|x - y\|^2 \sum_{j=1}^{k} \frac{1}{j^{2\alpha}} \qquad (5.29) \\
&= \frac{1}{i^\alpha} \left[\sigma_{w^{(l)}}^2 c_{l-1} \sum_{j=1}^{k} \frac{1}{j^{2\alpha}}\right] \|x - y\|^2, \qquad (5.30)
\end{aligned}$$

such that the claim follows upon defining $c_l = \sigma_{w^{(l)}}^2 c_{l-1} \sum_{j=1}^{\infty} 1/j^{2\alpha}$.

Lastly, Property 8 follows immediately, as the property is only violated if all weights within a layer are 0, which is a zero probability event as long as $\sigma_{w^{(l)}}^2 > 0$. □

Chapter 6

Bayesian Inverse Reinforcement Learning

A modified version of the work presented in this chapter and the last is under review for publication as Sell and Singh (2020).

6.1 Introduction

The applications of reinforcement learning are various, including the control of autonomous cars (Pusse and Klusch, 2019), robotics (Kober et al., 2013), and chess (Silver et al., 2018) where the resulting algorithms beat world-class chess players.

In inverse reinforcement learning one aims to learn an agent's value function from observing their series of states and actions; having the agent's value function at hand allows one to mimic the behaviour of the agent. In a *Bayesian* approach to this problem, one defines a prior on a function space that includes all admissible value functions. The data observed from an agent's behaviour can then be used through a suitably defined likelihood (Ramachandran and Amir, 2007) to infer the value function.

For discrete state spaces, Singh et al. (2013) provide a method to quantify the uncertainty of the estimated value function. In this chapter, we will generalise those ideas to continuous state spaces by using priors introduced in the previous section.

We further demonstrate the usefulness of the methodology developed in Chapter 5 on a challenging 17-dimensional reinforcement learning example where the aim is to learn control strategies via a Bayesian formulation.

The new contributions of this chapter are as follows:

- We propose a suitable likelihood for Bayesian Reinforcement Learning (BRL) for inferring the unknown continuous state value function that best describes an observed state-action data sequence. Theorem 7 and Lemma 2 justify the use of this likelihood with Gaussian prior measures on function spaces, and with our proposed neural network prior. This likelihood is also potentially of interest to the machine learning community in its own right.
- We apply Hilbert space MCMC methods to infer the unknown optimal value function in two continuous state control problems, using both our new prior and likelihood function. We provide numerical evidence that, in contrast to standard function space priors, the trace-class neural network prior is scalable to higher dimensional problems.

The rest of this chapter is organised as follows: in Section 6.2 we recapitulate the necessary definitions of Markov decision processes. In Section 6.3 we state the Bayesian Reinforcement Learning (BRL) problem and introduce the likelihood to be used for inferring continuous state value functions from state-action data. We then show that the likelihood satisfies the assumptions needed to be admissible in a Hilbert space MCMC setting. Finally, Section 6.4 provides numerical results for the proposed prior from the previous chapter and the likelihood for different control problems. Proofs can be found in the appendix.

6.2 Markov Decision Processes

A *Markov Decision Process* is defined by a controlled Markov chain $\{X_n\}_{n\in\mathbb{N}}$ called the *state process*, the *control process* $\{A_n\}_{n\in\mathbb{N}}$, and an optimality criterion.
The state process takes values in a bounded set $\mathcal{X} \subset \mathbb{R}^d$, for simplicity we will consider the domain to be the d-dimensional hypercube $\mathcal{X} = [0,1]^d$. The control process is $\mathcal{A}$-valued, where $\mathcal{A} = \{1,\dots,M\}$ is a finite set. Given realisations of the state and actions process until time $n \geq 0$, the state process propagates according to

$$p(X_{n+1} = dx_{n+1} | X_{1:n} = x_{1:n}, A_{1:n} = a_{1:n}) = p(X_{n+1} = dx_{n+1} | X_n = x_n, A_n = a_n),$$

where for any state-action pair (x_n, a_n), we simply write $p(x_{n+1}|x_n, a_n)$ to denote this is a probability density function. In some applications, the state dynamics are deterministic, and thus there exists a map T such that

$$X_{n+1} = T(x_n, a_n). \tag{6.1}$$

The action process depends on a *policy* $\mu : \mathcal{X} \to \mathcal{A}$ which is a mapping from the state space into the action space. For a fixed state x_n, the policy defines a probability distribution over the actions; if the policy is deterministic, this can be written as

$$p(A_n = da_n | X_{1:n} = x_{1:n}, A_{1:n-1} = a_{1:n-1}) = \delta_{\mu(x_n)}(da_n)$$

As there are many possible mappings $\mu : \mathcal{X} \mapsto \mathcal{A}$, we assume the agent executes a policy that is in some way optimal. To be more precise, let $r : \mathcal{X} \to \mathbb{R}$ be the *reward function*, then the *accumulated reward* given a policy and an initial state $X_1 = x_1$ is

$$C_\mu(x_1) = \mathbb{E}_\mu \left[\sum_{n=1}^{\infty} \beta^n r(X_t) | X_1 = x_1 \right],$$

where $\beta \in (0, 1)$ is a discount factor. The discount factor serves two purposes: it ensures that the expectation is well defined, and also lets early actions be more important than later actions, see Kaelbling et al. (1996) for a more detailed discussion.
A policy μ^* is *optimal* if $C_{\mu^*}(x_1) \geq C_\mu(x_1)$ for all (μ, x_1). The solution to the Bellman equation (Bellman, 1952)

$$v(x) = \max_{a \in \mathcal{A}} \left[r(x) + \beta \sum_{x' \in \mathcal{X}} p(x'|x, a) v(x') \right]$$

is called the *optimal value function* (Bertsekas, 1995). Given this fixed point solution v, we can derive the optimal policy by

$$\mu^*(x) = \arg\max_{a \in \mathcal{A}} \left[\sum_{x' \in \mathcal{X}} p(x'|x, a) v(x') \right], \tag{6.2}$$

that is, the optimal action at any state is the one that maximises the expected value function at the next state.
For the observed actions, in what follows we assume the agent is not perfect, e.g. a human

expert, and chooses an action with a certain error. At each time step the chosen action is a random variable given by

$$A_n = \arg\max_{a \in \mathcal{A}} \left[\sum_{x' \in \mathcal{X}} p(x'|x,a) v(x') + \epsilon_n(a) \right], \tag{6.3}$$

where we assume $\epsilon_n \sim \mathcal{N}(0, \sigma^2 I_{M \times M})$. The Gaussian choice simplifies numerical calculations, and it is reasonable to assume that the variances for different actions are independent and identically distributed, but this assumption can be relaxed. When the state dynamics are deterministic, see (6.1), action selections occur according to

$$A_n = \arg\max_{a \in \mathcal{A}} \left[v(T(x,a)) + \epsilon_n(a) \right], \tag{6.4}$$

Our goal from now on will be to recover the agent's value function, and quantify the uncertainty thereof, by using the Hilbert space MCMC methods and the priors discussed in Sections 5.3 and 5.4.

6.3 Likelihood Definition

The data consists of a collection of state-action pairs $y = \{y_t\}_{t=1}^T = \{(x_t, a_t)\}_{t=1}^T$ and the aim is to infer the value function (and thus the policy through (6.2)) that leads to the actions a_t for the current state x_t. Using the noisy action selection procedure (6.3), the likelihood is

$$\mathcal{L}(y|v,\sigma) = \prod_{t=1}^{T} p(a_t|x_t,v,\sigma) = \prod_{t=1}^{T} p(a_t|v_t,\sigma), \tag{6.5}$$

by defining the vector v_t, containing the relevant evaluations of the value function to calculate the likelihood at y_t, i.e. using equation (6.4), the k-th entry of v_t is the evaluation of the value function $v(\cdot)$ at the location $T(x_t, k)$, corresponding to starting at x_t and taking action $a = k \in \mathcal{A}$.

For a single observation $y_t = (x_t, a_t)$, we now drop the subscript t to simplify notation, and assume wlog that the optimal action is action $k = 1$, permuting the labels if necessary. The probability $p(a = 1|v,\sigma)$ (where v is now a vector and $p(a|v,\sigma)$ is a probability mass function)

can be computed using (6.3) by

$$p(a=1|v,\sigma) = \int \mathbb{1}_{\{u\in\mathbb{R}^d : u_1 \geq u_j \forall j\neq 1\}} \mathcal{N}(u; v, \sigma^2 I_{M\times M}) du. \tag{6.6}$$

To compute this term, we make use of the fact that the value of the integral is the same as the probability $\mathbb{P}(X_1 > X_j,\ \forall j \neq 1)$, where $X_k \sim \mathcal{N}(v(T(x,1)), \sigma^2))$. This can be computed numerically using the pdf $\phi_1(\cdot)$ of X_1 and cdfs $\Phi_j(\cdot)$ of the respective X_j:

$$p(a=1|v,\sigma) = \int_{-\infty}^{\infty} \phi_1(t)\Phi_2(t)\dots\Phi_M(t)dt \tag{6.7}$$

$$= \frac{1}{\sigma}\int_{-\infty}^{\infty} \phi\Big(\frac{t-v_1}{\sigma}\Big)\prod_{j=2}^{M}\Phi\Big(\frac{t-v_j}{\sigma}\Big)dt. \tag{6.8}$$

If the noise in (6.3) is not diagonal, this simplification cannot be made, and the integral (6.6) is harder to compute. More advanced numerical methods exist to efficiently calculate such integrals using Monte-Carlo simulations (Genz, 1992).

6.3.1 Likelihood Gradient

Following from (6.8) we can compute the gradient of the likelihood in a data point (x_t, a_t) with respect to v_t. We again assume wlog that $a_t = 1 \in \mathcal{A}$ (by permuting the actions if necessary), and drop the subscript t, emphasising that v_k is the k-th entry of the vector v. The partial derivatives with respect to the v_k are given by

$$\frac{\partial}{\partial v_1} p(a=1|v,\sigma) = \frac{1}{\sigma}\int_{-\infty}^{\infty} \frac{t-v_1}{\sigma^2}\phi\Big(\frac{t-v_1}{\sigma}\Big)\prod_{j=2}^{M}\Phi\Big(\frac{t-v_j}{\sigma}\Big)dt \tag{6.9}$$

$$\frac{\partial}{\partial v_k} p(a=1|v,\sigma) = -\frac{1}{\sigma^2}\int_{-\infty}^{\infty} \phi\Big(\frac{t-v_1}{\sigma}\Big)\phi\Big(\frac{t-v_k}{\sigma}\Big)\prod_{j=2, j\neq k}^{M}\Phi\Big(\frac{t-v_j}{\sigma}\Big)dt \qquad k = 2\dots M \tag{6.10}$$

$$= -\frac{1}{\sigma^2}\phi\Big(\frac{v_1-v_k}{\sqrt{2}\sigma}\Big)\int_{-\infty}^{\infty} \phi\left(\frac{t-\frac{v_k+v_1}{2}}{\frac{\sigma}{\sqrt{2}}}\right)\prod_{j=2, j\neq k}^{M}\Phi\Big(\frac{t-v_j}{\sigma}\Big)dt, \tag{6.11}$$

where the last identity follows from the product of two Gaussian pdfs. This allows us, when using the neural network prior, to compute the gradient of the log-likelihood with respect to the parameters of the neural network, θ. We emphasise that the vector $v = v(\theta)$ depends on

these parameters, justifying the calculation of the Jacobian $\mathcal{D}_\theta v$. Using the chain rule, we get

$$\nabla_\theta \log p(a=1|v,\sigma) = \frac{\nabla_\theta p(a=1|v,\sigma)}{p(a=1|v,\sigma)} = \frac{(\mathcal{D}_\theta v)^T \nabla_v p(a=1|v,\sigma)}{p(a=1|v,\sigma)}. \tag{6.12}$$

To get the entire gradient of the log-likelihood, we simply need to sum over all data points:

$$\nabla_\theta \ell(y|v,\sigma) = \nabla_\theta \log\left(\prod_{t=1}^{T} p(a_t|v_t,\sigma)\right) = \nabla_\theta \sum_{t=1}^{T} \log p(a_t|v_t,\sigma) = \sum_{t=1}^{T} \nabla_\theta \log p(a_t|v_t,\sigma), \tag{6.13}$$

where we only need to keep in mind the permutation in the actions when using (6.12).

When calculating (6.12), we note that $1 \cdot \nabla_v p(a|v,\sigma) = 0$ by translation invariance of v: $\mathcal{L}(y|v,\sigma) = \mathcal{L}(y|v+c,\sigma)$ for any constant function c, i.e. $c(x) = c(x')$ for all $x, x' \in \mathcal{X}$. To avoid instabilities when calculating the gradient numerically, we can ensure that the mean of these gradients is 0 by using the following modification, which enhances the performance in practice:

$$(6.12) = \frac{\sum_{k=1}^{M} ((\mathcal{D}_\theta v)^T)_k (\frac{\partial}{\partial v_k} p(a=1|v,\sigma) - \sum_{k=1}^{M} \frac{\partial}{\partial v_k} p(a=1|v,\sigma))}{p(a=1|v,\sigma)}. \tag{6.14}$$

The following theorem justifies the use of this likelihood in the function space MCMC setting:

Theorem 7. *The log-likelihood $\ell(y|v,\sigma) = \log \mathcal{L}(y|v,\sigma)$ defined in* (6.5) *satisfies Assumptions 3 and 4, where $v \in \mathcal{H} = L^2$.*

The proof can be found in Appendix 6.6.1. We also note that when using the trace-class neural network prior from Section 5.4, the statements remain true if the likelihood is seen as a function of the parameters θ of the neural network:

Lemma 2. *The log-likelihood $\ell(y|v_\theta,\sigma)$ defined in* (6.5) *satisfies Assumptions 3 and 4, where now inference is over the weights and biases, $\theta \in \mathcal{H} = \ell^2$.*

Proof. As v is a Lipschitz continuous function of θ, $\ell(y|v(\theta),\sigma)$, as a composition of Lipschitz continuous functions, is also Lipschitz continuous in θ; and, with an activation function satisfying Condition 9, v grows at most polynomial in θ. □

The next theorem ensures that the pCNL algorithm is well-defined for both priors by making sure that proposals are π_0-almost surely in the image of the prior:

Theorem 8. *For any $u \sim \mathcal{N}(0, C)$, we have $\mathcal{N}(C\mathcal{D}\ell(u), C) \simeq \mathcal{N}(0, C)$, that is, the two Gaussian measures are absolutely continuous with respect to one another. In other words, Condition 5 holds, i.e. the preconditioned gradient of the log-likelihood is in the image of the prior. This holds for both the KL-prior where $D\ell(u)$ should be understood as the collection of derivatives with respect to the ξ_i and the prior is over $\mathcal{H} = L^2$, and for the trace-class neural network prior, where $D\ell(u)$ is the collection of derivatives with respect to each weight and bias and inference is over the parameters of the neural network such that $\mathcal{H} = \ell^2$.*

The proof can be found in appendix 6.6.2.

6.4 Numerical Illustrations

This section aims to validate the theory, and highlight the applicability of the proposed priors and methodology. In particular, Section 6.4.2 confirms that, empirically, as the layer width for the trace-class neural network prior grows, the acceptance probability does not go to 0, a property known as 'stable under mesh-refinement' or 'dimension-independence'; Section 6.4.3 compares the proposed trace-class neural network prior to the Karhunen-Loève prior, and highlights that, unlike the latter, the former is scalable to higher-dimensional domains; and Section 6.4.4 shows that the posteriors can learn and mimic policies, thus justifying the use of these priors in the reinforcement learning setup.

Throughout we use the Fourier basis (5.7) as the series expansion of choice when using the Karhunen-Loève based prior, as this proved to be a good choice for reinforcement learning problems (Konidaris et al., 2011). As a tuning parameter for the corresponding eigenvalues we set $\alpha = 2$ in (5.8), forcing the samples to be very smooth which we expect to be a sensible choice in the discussed control problems. For the trace-class neural network prior we used fully connected layers with tanh activation functions throughout. We set the variance parameters uniformly as $\sigma^2_{b^{(l)}} = \sigma^2_{w^{(l)}} = 2$, and set $\alpha = 1.5$, this again results in smooth sample functions.

The code for all examples in this chapter is available online at https://github.com/TorbenSell/trace-class-neural-networks.

6.4.1 Control Problems: Setup

We set the scene by describing the setup of the control problems which we use in the experiments.

Mountaincar

The first example is the popular mountaincar problem. The state space is the 2-dimensional domain $\mathcal{X} = [-1.2, 0.6] \times [-0.07, 0.07]$, where the first variable is the position x_1 of the car on a mountain slope, and the second variable represents its velocity x_2. The set of possible actions is $\mathcal{A} = \{-1, 0, 1\}$, representing exerting force to the left, not adding force, and exerting force to the right, respectively. The state transitions are deterministic, being given by Newtonian physics, and we refer the reader to the OpenAI documentation or to our code for the details. See Figure 6.1 for an illustration of the scenario.

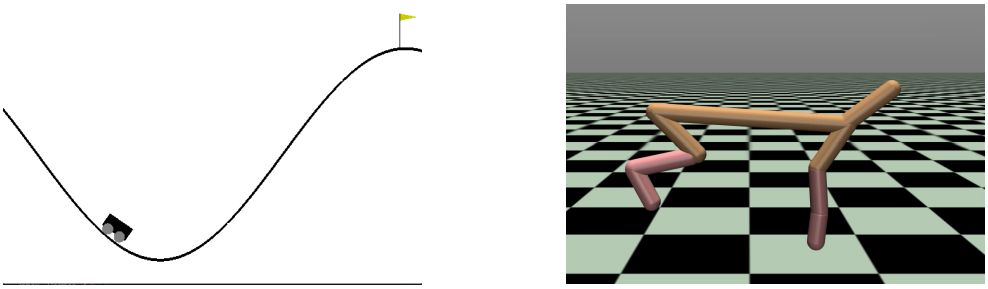


Fig. 6.1 *Left*: The setup for the mountaincar example. The car's goal is to reach the goal in as few steps as possible. The slope on the right is too steep to simply drive up the mountain, the car therefore has to gain momentum by going up the hill on the left first. *Right*: The HalfCheetah has states x_t in $\mathbb{R}^{17}$. Its goal is to run to the right as quickly as possible, while not moving its body parts more than necessary.

In the mountaincar problem, the reward is constant $r(x_1, x_2) = -1$ per step, until the car reaches the top of the mountain ($x_1 \geq 0.5$). The optimal policy is therefore to reach the mountaintop as quickly as possible, and we terminate an episode if the car didn't make it up the hill after 200 time steps. All data was generated from an optimal deterministic policy (Xiao, 2019) given by

$$\mu(x_1, x_2) = -1 + 2\mathbb{I}\{\min(-0.09(x_1 + 0.25)^2 + 0.03, 0.3(x_1 + 0.9)^4 - 0.008) \leq x_2\} \times \\ \times \mathbb{I}\{x_2 \leq -0.07(x_1 + 0.38)^2 + 0.07\},$$

and we set the noise level to $\sigma = 0.1$.

HalfCheetah

To show that our algorithm works in a more complicated setting, we looked at the HalfCheetah example from the MuJoCo library (Todorov et al., 2012) where the state x_t is a 17-dimensional vector. The original continuous actions space of the problem is 6-dimensional. We discretised the action space to $M = 8$ actions resulting in a computable likelihood (6.8), where we set the noise level to $\sigma = 0.1$. Positive rewards are given for moving forward, and negative rewards are given for moving backwards. A further penalty is deducted for 'large' effort actions, causing the goal to be to move forward as quickly as possible with as little effort as possible.

To generate data, we used as the expert the best performing computed policies from Berkeley's Deep Reinforcement Learning Course[1], and projected those actions onto 'our' discrete action space by choosing the discrete action minimising the Euclidean distance to the expert's action.

6.4.2 Dimension Independence of Trace-Class Neural Network Priors Under Mesh Refinement

We ran pCN for different network widths on the mountaincar example. The Network used has $l = 3$ hidden layers. As stated before, the tuning parameters in the prior are set to $\sigma^2_{b^{(l)}} = \sigma^2_{w^{(l)}} = 2$, and $\alpha = 1.5$. Table 6.1 displays the acceptance probability of pCN for fixed step size ($\beta_{pCN} = 1/10$) when targeting the posterior arising from the trace-class neural network prior and the mountaincar likelihood.

$N^{(l)}$, for all l	20	30	40	50	60	70	80	90	100
Acc. ratio (in %)	24.0	23.5	22.1	22.2	23.1	23.9	23.4	23.0	23.9
Total # of param.	921	1981	3441	5301	7561	10221	13281	16741	20601

Table 6.1 Acceptance ratios and total number of parameters (weights and biases) for different layer widths. 3 fully connected layers were used, and pCN was run over 3 hours for each of the layer widths. Notably the acceptance probability does not degenerate as more nodes are included per layer.

[1] CS294-112 HW 1: Imitation Learning, https://github.com/berkeleydeeprlcourse/homework/tree/master/hw1, accessed on December 4th, 2020.

6.4.3 Comparison of Priors

To compare the trace-class neural network prior to the Karhunen-Loève prior, we used a large number of parameters for each, such that the error from truncating after finitely many nodes, or finitely many terms, is negligible. For both the mountaincar and the HalfCheetah example, we used the same trace-class neural network prior, with 3 hidden layers, and 100 nodes per layer, resulting in $20,601$ parameters to be estimated for the mountaincar example, and $22,101$ for the HalfCheetah example. For the Karhunen-Loève prior in the mountaincar example we set the truncation parameter to $k_{max} = 70$ for (5.7) with eigenvalues (5.8) (recall that here $\alpha = 2$), resulting in a total of $19,880$ coefficients to be estimated. For the KL prior in the HalfCheetah example we used approximation (5.5), and otherwise the same eigenfunctions and eigenvalues; due to the higher domain dimension $d = 17$, one would have to estimate $2,667,980$ parameters. As this is too memory-expensive for the computer used for the experiments, we used $k_{max} = 10$ in the HalfCheetah example, resulting in $54,740$ parameters to be estimated. Note that this increase in parameters to be estimated is already due to the approximation (5.5) being used, and additionally truncating the expansions after fewer terms, highlighting the benefits of the dimension-robustness of the trace-class neural network prior.

To assess the quality of the priors, we ran pCN using 50 (for the mountaincar) and 100 (for the HalfCheetah) data points. For the mountaincar example, we fixed five test points z_j, $j = 1,\dots,5$, and compared the posteriors by evaluating $v(z_j)$ at these new locations as estimated through MCMC runs. The top row in Figure 6.2 shows the resulting uncertainty estimates. As the value function is invariant under translations, we adjusted all samples such that they take the value 0 at the state which the optimal action takes one to.

For the HalfCheetah example, we looked at one test point for illustration, see the bottom row in Figure 6.2, and summarised the performance on another 100 data points in the Table 6.2, where we compared how the respective samples from the posterior do, as well as how the mean of all samples from the posterior in Section 6.4.4 (with a smaller number of nodes for the NN prior, and fewer active terms in the KL prior[2]) does on predicting the correct action (last two columns). Not surprisingly, the mean function is better at picking the correct action.

[2]To calculate the mean function it is necessary to store the samples which (due to the used computer's limited memory capacity) would not be feasible for the very wide layer prior, nor all the terms in the KL prior.

Decision made by	KL samples	NN samples	KL mean	NN mean
Optimal Action	20.1%	32.1%	25%	42%
Non-optimal Action	79.9%	67.9%	75%	58%

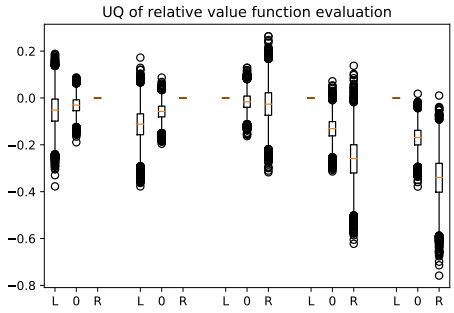
Table 6.2 Optimal action picked by samples, and the sample mean at a data point, for two different posteriors. The trace-class neural network prior outperforms the approximate KL prior.

6.4.4 Ability to Learn Policy

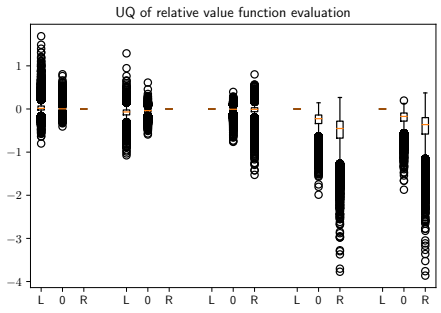
To asses if the posteriors can truly learn an agent's behaviour, we used the priors with a smaller number of parameters, and stored 1000 samples for each posterior. We then used these samples to obtain a mean value function which was used for decision making. For the trace-class neural network prior we used 3 layers with 10 nodes per layer for both examples (resulting in 261 parameters for the mountaincar example and 411 for the HalfCheetah); for the KL prior we used $k_{max} = 7$ for the mountaincar example (giving a total of 224 parameters), and $k_{max} = 5$ in the HalfCheetah example (a total of $8,730$ parameters). The results are summarised in Figure 6.3.

6.5 Discussion

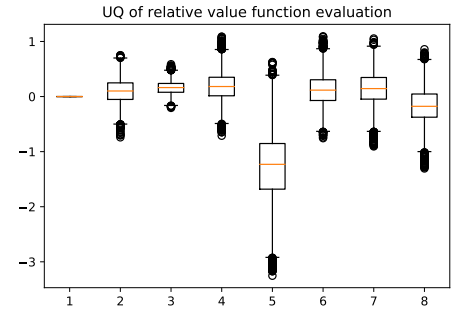
This chapter addresses the problem of effective Bayesian inference for unknown functions with higher dimensional domains. Unlike priors which require an orthogonal basis for the function space, and scale exponentially in the domain dimension, our proposed trace-class neural network prior easily scales to higher-dimensional domains as the dependence on the domain dimension is linear. When using the pCN sampling method, this prior also satisfies the desired property of being stable under mesh-refinement, in the sense that the acceptance probability of pCN does not degenerate to 0 when using more parameters for the neural network. Various questions remain unanswered though, and interesting directions of research open up: are there suitable further generalisations of the proposed prior, e.g. heavy-tailed or hierarchical ones, which still satisfy the desired properties, enabling the use of these priors in the function space setting? Furthermore, hierarchical priors could allow handling situations in which some of the model parameters are unknown. This will normally be the case, and rather than treating e.g. σ, the noise contaminating the actions, as fixed, a hyperprior on σ can account for this uncertainty. This requires careful consideration though: the action



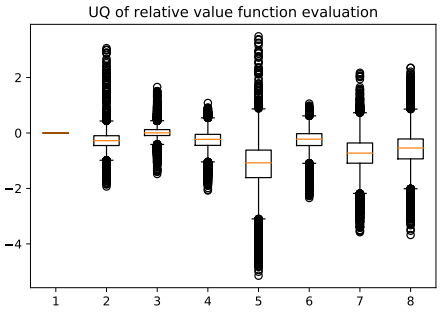
(a) Mountaincar: KL based posterior



(b) Mountaincar: NN based posterior



(c) HalfCheetah: KL based posterior

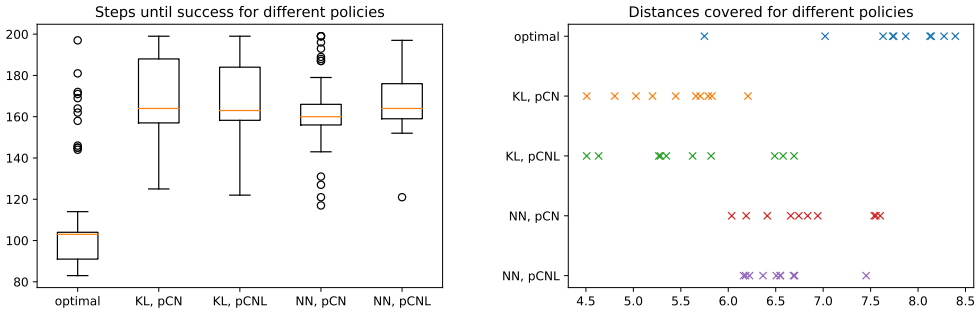


(d) HalfCheetah: NN based posterior

Fig. 6.2 Uncertainty estimates arising from the two different priors for the mountaincar and the HalfCheetah examples.
Top row: Mountaincar example. In each plot, five different states are looked at, the estimates of the value functions are shown, standardised such that the optimal action has value 0 always. Both the Karhunen-Loève based posterior and the neural network based posterior can make no clear judgement as to what the optimal actions for the first three shown states are. For the fourth and fifth states, both posteriors suggest a clear decision for action 'Left'. The reader should note that both the KL and the NN posteriors behave similarly in that they are uncertain in the first three states, and very decisive in the last two states.
Bottom row: HalfCheetah example. The optimal action is the first one in both plots, and samples are normalised such that they take the value 0 at the state the optimal value takes one to. The NN posterior correctly estimates the optimal action, the KL posterior doesn't.

variance σ and the choice of *prior* variances are closely related. To see this, consider finding the argmax in Equation (6.4), and let for simplicity $v(T(x,a_1)) = v_1$ and $v(T(x,a_2)) = v_2$. Then

$$\arg\max_a\{v_1+\epsilon_1, v_2+\epsilon_2\} = \arg\max_a\{10v_1+10\epsilon_1, 10v_2+10\epsilon_2\},$$



(a) Mountaincar: Results

(b) HalfCheetah: Results

Fig. 6.3 Results for the policy learning experiment.
Left: Mountaincar example. The number of steps until success is shown for different posteriors. If the goal was not reached after 200 steps, the run was counted as failure. Out of 100 runs, the policy following the KL posterior when using pCN gave 34 failures (24 when using pCNL), the NN posteriors gave 23 (for the posterior estimates obtained using pCN) and 25 (pCNL).
Right: HalfCheetah example. The different policies arising from the KL posterior (obtained once using pCN, once using pCNL) and the NN posterior were controlling the agent over 10 runs with 100 time steps. The distances covered per run are shown in the plot.

and we note that we get the same results to our original setup if we multiply the value function by 10 (or any other constant c, and the variance in the noise by 100 (or generally any c^2). This requires us to carefully investigate the interplay of the action variance and the prior parameters, which we will leave for future investigations.

Another question relates to the prior parameters, assuming now a fixed action noise: what are the optimal settings for the tuning parameters $\sigma^2_{w^{(l)}}$, $\sigma^2_{b^{(l)}}$ and α? How does the prior behave in the depth limit? Can one obtain contraction rates to ensure the concentration of the posterior samples around the true functions? A first idea here is to exploit the various generalisations of the universal approximation theorem, and combine them with the proof methodology used in this chapter and the previous.

We further introduced a likelihood suitable for Bayesian reinforcement learning where the underlying Markov decision process has a continuous state-space, and thus the unknown value function to be estimated has domain $\mathbb{R}^d$ as opposed to a discrete set. An interesting research direction is to generalise this to continuous action spaces as well.

Finally, we underscored the theory with numerical illustrations, illustrating the applicability of the prior in various control problems.

6.6 Appendix

6.6.1 Proof of Theorem 7

Proof of Theorem 7. In the following, we identify v with its finitely many observations v_t. One notes that the integral (6.6) is upper bounded by 1. For the lower bound we substitute $w = u - v$ in (6.6), allowing us to bound

$$\begin{aligned}
(6.6) &= \int \mathbb{1}_{\{w\in\mathbb{R}^d:(w+v)_1\geq(w+v)_j\forall\, j\neq 1\}}\mathcal{N}(w;0,\sigma^2 I_{M\times M})dw \\
&\geq \int \mathbb{1}_{\{w\in\mathbb{R}^d:(w+v)_1\geq 0\geq(w+v)_j\forall\, j\neq 1\}}\mathcal{N}(w;0,\sigma^2 I_{M\times M})dw \\
&= \int \mathbb{1}_{\{w\in\mathbb{R}^d:w_1\geq -v_1, w_j\leq -v_j\forall\, j\neq 1\}}\mathcal{N}(w;0,\sigma^2 I_{M\times M})dw \\
&= \int \mathbb{1}_{\{w\in\mathbb{R}^d:w_1\geq -v_1, w_j\geq v_j\forall\, j\neq 1\}}\mathcal{N}(w;0,\sigma^2 I_{M\times M})dw \qquad (6.15) \\
&\geq \int \mathbb{1}_{\{w\in\mathbb{R}^d:w_1\geq |-v_1|, w_j\geq |v_j|\forall\, j\neq 1\}}\mathcal{N}(w;0,\sigma^2 I_{M\times M})dw \\
&\geq \int \mathbb{1}_{\{w\in\mathbb{R}^d:w_1\geq s, w_j\geq s\forall\, j\neq 1\}}\mathcal{N}(w;0,\sigma^2 I_{M\times M})dw, \qquad (6.16)
\end{aligned}$$

where for (6.15) we exploited the fact that the normal is centered and isotropic, and to get (6.16) we pick
$s = \|v\|_\infty > 0$ if $v \neq 0$. The case $v = 0$ is trivial as the integral is then non-zero. (6.16) can now be treated as the tail probability for a centered multivariate normal distribution (see Hashorva (2005)) which satisfies the Savage condition (Savage, 1962). We thus get the bound

$$(6.16) \geq c\exp(-s^2) = c\exp(-\|v\|_\infty^2)$$

as long as $\|v\|_\infty > 2d/\sigma^4$. For all other v, the integral (6.6) is bounded away from zero, and the Assumption 3 holds trivially. The assumption thus holds for all v.
To see that Assumption 4 holds, note that the log-likelihood is continuously differentiable in v, and thus locally Lipschitz. □

6.6.2 Proof of Theorem 8

Proof of Theorem 8. Firstly, note that the Assumptions in Theorem 6 hold by Theorem 7, and additionally the log-likelihood ℓ satisfies $\sum_{k=1}^{M} \frac{\partial \ell}{\partial v_k} = 0$ (where v_k is the evaluation $v(x_k)$

of the value function at location x_k) by the translation invariance, and $\sum_{k=1}^{M}\left(\frac{\partial \ell}{\partial v_k}\right)^2 < \infty$ for any v μ_0-almost surely.

For the KL prior, the claim follows immediately from the Feldman-Hajek theorem (see e.g. Da Prato and Zabczyk (2014)).

For simplicity, we from now on assume that the likelihood consists of only one data point and omit the respective index, but the generalisation to multiple data points is trivial.

Following a similar argument as in the proof of Theorem 6, we will show that for fixed x

$$S_s = S_s(x) = \sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\frac{\sigma_{b_l}^2}{i^\alpha}\left(\frac{\partial \ell}{\partial B_i^{(l)}}(x)\right)^2 + \sum_{j=1}^{s}\frac{\sigma_{w_l}^2}{(ij)^\alpha}\left(\frac{\partial \ell}{\partial W_{i,j}^{(l)}}(x)\right)^2\right] \tag{6.17}$$

defines a L^1-bounded submartingale, converging almost surely to a random variable S_∞ by Doob's martingale convergence theorem. Once we have established the convergence, we note that this implies the equivalence

$$\mathcal{N}(C\mathcal{D}\ell(u), C) \simeq \mathcal{N}(0, C) \tag{6.18}$$

for all u π_0-almost surely by applying (a simplified version of) the Feldman-Hajek theorem (Da Prato and Zabczyk, 2014, Theorem 2.23) which states that two Gaussian measures $\mathcal{N}(m_1, Q)$ and $\mathcal{N}(m_2, Q)$ are absolutely continuous with respect to one another if and only if $m_1 - m_2 \in Q^{1/2}(\mathcal{H})$, i.e. $\sum_i (m_{1i} - m_{2i})^2/\lambda_i^2 < \infty$ (Da Prato and Zabczyk, 2014)[3]. Here, the latter corresponds to

$$\sum_{i=1}^{\infty}\frac{(C\mathcal{D}\ell(u))_i^2}{\lambda_i^2} = \sum_{i=1}^{\infty}\frac{\lambda_i^4(\mathcal{D}\ell(u))_i^2}{\lambda_i^2} = \sum_{i=1}^{\infty}\lambda_i^2(\mathcal{D}\ell(u))_i^2 \tag{6.19}$$

being finite (Da Prato and Zabczyk, 2014). Note that $\mathcal{D}\ell(u)$ is a random variable defined over the parameters of the neural network; Equation 6.19 is, after substituting both the eigenvalues of C and the derivatives with respect to the parameters into this expression, equivalent to Equation 6.17. We will show that S_s is finite π_0-almost surely such that the equivalence (6.18) holds.

[3] In our notation, the eigenvalues of C are λ_i^2, in comparison to the ones used in Da Prato and Zabczyk (2014) who define the eigenvalues of Q as λ.

Remark: We here also note that, if we want to apply the same argument for the case without preconditioning, we would require (by the same argument as above) that

$$\sum_{i=1}^{\infty} \frac{(\mathcal{D}\ell(u))_i^2}{\lambda_i^2} = \sum_{i=1}^{\infty} \frac{(\mathcal{D}\ell(u))_i^2}{\lambda_i^2} < \infty,$$

which will generally not be the case. One can check this by repeating the proof for the pCNL case (note that the likelihood contribution would have to dominate the diverging $1/\lambda_i^2$ terms).

It remains to show that S_s is a L^1-bounded submartingale. It is clear that S_s is a (non-negative) submartingale, and we only need to show that the expectation of S_s is bounded. We define $g_i^{(l)}(x) = \varphi(f_i^{(l)}(x))$ to simplify notation. From (5.25) we get $\mathbb{E}[(f_i^{(l)})^2] \leq \sigma_l^2/i^\alpha$, and combining this bound with Assumption 9, we get $\mathbb{E}[(g_i^{(l)})^2] \leq \sigma_l^2/i^\alpha$.

We ask the reader to recall Definition 5.14. The partial derivatives with respect to the biases and weights can be calculated by applying the chain rule multiple times, and they are given by

$$\frac{\partial \ell}{\partial W_{1,j}^{(n+1)}} = \sum_{k=1}^{M} \frac{\partial v_k}{\partial W_{1,j}^{(n+1)}} \frac{\partial \ell}{\partial v_k} = \sum_{k=1}^{M} g_{1,j}^{(n)}(x^k) \frac{\partial \ell}{\partial v_k} \tag{6.20}$$

$$\frac{\partial \ell}{\partial B_1^{(n+1)}} = \sum_{k=1}^{M} \frac{\partial v_k}{\partial B_{1,j}^{(n+1)}} \frac{\partial \ell}{\partial v_k} = \sum_{k=1}^{M} \frac{\partial \ell}{\partial v_k} \tag{6.21}$$

$$\frac{\partial \ell}{\partial W_{i,j}^{(l)}} = \frac{\partial f_i^{(l)}}{\partial W_{i,j}^{(l)}} \frac{\partial \ell}{\partial f_i^{(l)}} = g_j^{(l-1)} \frac{\partial \ell}{\partial f_i^{(l)}} \qquad l = 2 \dots n \tag{6.22}$$

$$\frac{\partial \ell}{\partial B_i^{(l)}} = \frac{\partial f_i^{(l)}}{\partial B_i^{(l)}} \frac{\partial \ell}{\partial f_i^{(l)}} = \frac{\partial \ell}{\partial f_i^{(l)}} \qquad l = 2 \dots n \tag{6.23}$$

$$\frac{\partial \ell}{\partial W_{i,j}^{(1)}} = x_j \frac{\partial \ell}{\partial f_i^{(1)}} \tag{6.24}$$

$$\frac{\partial \ell}{\partial B_i^{(1)}} = \frac{\partial \ell}{\partial f_i^{(1)}}. \tag{6.25}$$

We will have to consider the following partial derivatives that show up in the ones above:

$$\frac{\partial \ell}{\partial f_i^{(l)}(x)} = \frac{\partial g_i^{(l)}(x)}{\partial f_i^{(l)}(x)} \frac{\partial \ell}{\partial g_i^{(l)}(x)} \qquad l = 1 \dots n \tag{6.26}$$

$$\frac{\partial \ell}{\partial g_i^{(l)}(x)} = \sum_{j=1}^{s} \frac{\partial \ell}{\partial f_j^{(l+1)}(x)} W_{j,i}^{(l+1)} \qquad l = 1 \dots n, \tag{6.27}$$

where in the last line we have already established the dependence of $\frac{\partial \ell}{\partial g_i^{(l)}(x)}$ on s. Using Assumption 9, we get that $|\frac{\partial g_i^{(l)}}{\partial f_i^{(l)}}| \leq 1$.

Now, for $l = n$, by independence:

$$\mathbb{E}\left[\left(\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2\right] = \mathbb{E}\left[\left(\frac{\partial \ell}{\partial f_1^{(n+1)}} W_{1,i}^{(n+1)}\right)^2\right] \tag{6.28}$$

$$= \frac{\sigma_{w_{n+1}}^2}{i^\alpha} \sum_{k=1}^{M} \left(\frac{\partial \ell}{\partial v_k}\right)^2, \tag{6.29}$$

and further (again by independence), for $l < n$,

$$\mathbb{E}\left[\left(\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2\right] = \sum_{j=1}^{s} \mathbb{E}\left[\left(\frac{\partial \ell}{\partial f_j^{(l+1)}} W_{j,i}^{(l+1)}\right)^2\right] \tag{6.30}$$

$$= \frac{\sigma_{w_{l+1}}^2}{i^\alpha} \sum_{j=1}^{s} \frac{1}{j^\alpha} \mathbb{E}\left[\left(\frac{\partial \ell}{\partial f_j^{(l+1)}}\right)^2\right] \tag{6.31}$$

$$= \frac{\sigma_{w_{l+1}}^2}{i^\alpha} \sum_{j=1}^{s} \frac{1}{j^\alpha} \mathbb{E}\left[\left(\frac{\partial g_j^{(l+1)}}{\partial f_j^{(l+1)}} \frac{\partial \ell}{\partial g_j^{(l+1)}}\right)^2\right] \tag{6.32}$$

$$\leq \frac{\sigma_{w_{l+1}}^2}{i^\alpha} \sum_{j=1}^{s} \frac{1}{j^\alpha} \mathbb{E}\left[\left(\frac{\partial \ell}{\partial g_j^{(l+1)}}\right)^2\right]. \tag{6.33}$$

We can use induction to show that for $l = 1 \dots n$,

$$\mathbb{E}\left[\left(\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2\right] \leq \frac{d_l}{i^\alpha} \sum_{k=1}^{M} \left(\frac{\partial \ell}{\partial v_k}\right)^2, \tag{6.34}$$

where

$$d_l = \left(\prod_{r=l+1}^{n+1} \sigma_{w_r}^2\right)\left(\sum_{j=1}^{\infty} 1/j^{2\alpha}\right)^{n+1-l-1}. \tag{6.35}$$

Putting things together, we get that (6.17) defines a L^1 bounded submartingale if $\alpha > 1$ and $\sum_{k=1}^{M}\left(\frac{\partial \ell}{\partial v_k}\right)^2 < \infty$:

$$\begin{aligned}
\mathbb{E}[S_s] &= \mathbb{E}\sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\frac{\sigma_{b_l}^2}{i^\alpha}\left(\frac{\partial \ell}{\partial B_i^{(l)}}(x)\right)^2 + \sum_{j=1}^{s}\frac{\sigma_{w_l}^2}{(ij)^\alpha}\left(\frac{\partial \ell}{\partial W_{i,j}^{(l)}}(x)\right)^2\right] \\
&\leq \max_l\left[\max\left(\sigma_{b_l}^2, \sigma_{w_l}^2\right)\right]\mathbb{E}\sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\left(\frac{\partial \ell}{\partial B_i^{(l)}}(x)\right)^2 + \sum_{j=1}^{s}\left(\frac{\partial \ell}{\partial W_{i,j}^{(l)}}(x)\right)^2\right],
\end{aligned}$$

and it remains to show that the expectation part of this expression is finite:

$$\begin{aligned}
&E\sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\left(\frac{\partial \ell}{\partial B_i^{(l)}}(x)\right)^2 + \sum_{j=1}^{s}\left(\frac{\partial \ell}{\partial W_{i,j}^{(l)}}(x)\right)^2\right] \\
&= \mathbb{E}\sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\left(\frac{\partial g_i^{(l)}}{\partial f_i^{(l)}}\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2 + \sum_{j=1}^{s}\left(g_j^{(l-1)}\frac{\partial g_i^{(l)}}{\partial f_i^{(l)}}\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2\right] \\
&\leq \mathbb{E}\sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\left(\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2 + \sum_{j=1}^{s}\left(g_j^{(l-1)}\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2\right] \\
&= \sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\mathbb{E}\left[\left(\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2\right] + \sum_{j=1}^{s}\mathbb{E}\left[\left(g_j^{(l-1)}\frac{\partial \ell}{\partial g_i^{(l)}}\right)^2\right]\right] \\
&\leq \sum_{l=1}^{n+1}\sum_{i=1}^{s}\left[\frac{d_l}{i^\alpha}\sum_{k=1}^{M}\left(\frac{\partial \ell}{\partial v_k}\right)^2 + \frac{d_l}{i^\alpha}\sum_{k=1}^{M}\left(\frac{\partial \ell}{\partial v_k}\right)^2\sum_{j=1}^{s}\frac{\sigma_l^2}{j^\alpha}\right] \\
&\leq \sum_{k=1}^{M}\left(\frac{\partial \ell}{\partial v_k}\right)^2\sum_{l=1}^{n+1}d_l\sum_{i=1}^{s}\left[\frac{1}{i^\alpha}\left(1 + \sum_{j=1}^{s}\frac{\sigma_l^2}{j^\alpha}\right)\right] < \infty.
\end{aligned}$$

□

6.6.3 Proof of pCNL Being Well-Defined

Theorem 9. *The preconditioned Crank-Nicolson Langevin algorithm is a well-defined MCMC algorithm in the function space setting if* $C\mathcal{D}\Phi(u) \in Im(C^{1/2})$.

Part of this proof can be found in Beskos et al. (2017).

Proof. We assume that Theorem 6.2 in Cotter et al. (2013) holds. We can then check that the change of measure arising from the gradient term is absolutely continuous with respect to the measures used in Theorem 6.2 if, and only if, $C\mathcal{D}\Phi(u) \in \mathrm{Im}(C^{1/2})$ by the Cameron-Martin Theorem (Cameron and Martin, 1944). □

References

Abdulle, A., Almuslimani, I., and Vilmart, G. (2018). Optimal explicit stabilized integrator of weak order 1 for stiff and ergodic stochastic differential equations. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):937–964.

Agapiou, S., Larsson, S., and Stuart, A. M. (2013). Posterior contraction rates for the bayesian approach to linear ill-posed inverse problems. *Stochastic Processes and their Applications*, 123(10):3828–3860.

Andrieu, C., Durmus, A., Nüsken, N., and Roussel, J. (2018). Hypocoercivity of piecewise deterministic markov process-monte carlo. *arXiv preprint arXiv:1808.08592*.

Arnold, B. C. and Press, S. J. (1989). Compatible conditional distributions. *Journal of the American Statistical Association*, 84(405):152–156.

Athreya, K. B. and Ney, P. (1978). A new approach to the limit theory of recurrent markov chains. *Transactions of the American Mathematical Society*, 245:493–501.

Au, S.-K. and Beck, J. L. (1999). A new adaptive importance sampling scheme for reliability calculations. *Structural safety*, 21(2):135–158.

Babacan, S. D., Molina, R., and Katsaggelos, A. K. (2008). Variational bayesian blind deconvolution using a total variation prior. *IEEE Transactions on Image Processing*, 18(1):12–26.

Bauschke, H. H., Combettes, P. L., et al. (2011). *Convex analysis and monotone operator theory in Hilbert spaces, second edition*, volume 408. Springer.

Bayes, T. (1763). Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418.

Bélisle, C. J., Romeijn, H. E., and Smith, R. L. (1993). Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716.

Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

Beskos, A., Girolami, M., Lan, S., Farrell, P. E., and Stuart, A. M. (2017). Geometric mcmc for infinite-dimensional inverse problems. *Journal of Computational Physics*, 335:327–351.

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., Stuart, A., et al. (2013). Optimal tuning of the hybrid monte carlo algorithm. *Bernoulli*, 19(5A):1501–1534.

Beskos, A., Roberts, G., Stuart, A., and Voss, J. (2008). Mcmc methods for diffusion bridges. *Stochastics and Dynamics*, 8(03):319–350.

Bickel, P. and Lindner, M. (2012). Approximating the inverse of banded matrices by banded matrices with applications to probability and statistics. *Theory of Probability & Its Applications*, 56(1):1–20.

Bierkens, J., Fearnhead, P., Roberts, G., et al. (2019a). The zig-zag process and super-efficient sampling for bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320.

Bierkens, J., Roberts, G. O., Zitt, P.-A., et al. (2019b). Ergodicity of the zigzag process. *The Annals of Applied Probability*, 29(4):2266–2301.

Bouchard-Côté, A., Vollmer, S. J., and Doucet, A. (2018). The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method. *Journal of the American Statistical Association*, 113(522):855–867.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of markov chain monte carlo*. CRC press.

Cameron, R. H. and Martin, W. T. (1944). Transformations of weiner integrals under translations. *Annals of Mathematics*, pages 386–396.

Candès, E. J., Romberg, J., and Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509.

Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

Chaari, L., Tourneret, J.-Y., and Batatia, H. (2013). Sparse bayesian regularization using bernoulli-laplacian priors. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5. IEEE.

Chambolle, A. (2004). An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97.

Chambolle, A., Caselles, V., Cremers, D., Novaga, M., and Pock, T. (2010). An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227.

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145.

Cheng, X., Chatterji, N. S., Bartlett, P. L., and Jordan, M. I. (2017). Underdamped langevin mcmc: A non-asymptotic analysis. *arXiv preprint arXiv:1707.03663*.

Christen, J. A. and Fox, C. (2005). Markov chain monte carlo using an approximation. *Journal of Computational and Graphical statistics*, 14(4):795–810.

Cotter, S. L., Roberts, G. O., Stuart, A. M., and White, D. (2013). Mcmc methods for functions: modifying old algorithms to make them faster. *Statistical Science*, pages 424–446.

Cui, T., Law, K. J., and Marzouk, Y. M. (2016). Dimension-independent likelihood-informed mcmc. *Journal of Computational Physics*, 304:109–137.

Da Prato, G. and Zabczyk, J. (2014). *Stochastic equations in infinite dimensions*. Cambridge university press.

Dalalyan, A. S. (2017). Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):651–676.

Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215.

Dashti, M., Harris, S., and Stuart, A. (2011). Besov priors for bayesian inverse problems. *arXiv preprint arXiv:1105.0889*.

Dashti, M., Law, K. J., Stuart, A. M., and Voss, J. (2013). Map estimators and their consistency in bayesian nonparametric inverse problems. *Inverse Problems*, 29(9):095017.

Deligiannidis, G., Bouchard-Côté, A., Doucet, A., et al. (2019). Exponential ergodicity of the bouncy particle sampler. *The Annals of Statistics*, 47(3):1268–1287.

Deligiannidis, G., Paulin, D., Bouchard-Côté, A., and Doucet, A. (2018). Randomized hamiltonian monte carlo as scaling limit of the bouncy particle sampler and dimension-free convergence rates. *arXiv preprint arXiv:1808.04299*.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, 195(2):216–222.

Dunlop, M. M., Girolami, M. A., Stuart, A. M., and Teckentrup, A. L. (2018). How deep are deep gaussian processes? *The Journal of Machine Learning Research*, 19(1):2100–2145.

Durmus, A., Guillin, A., Monmarché, P., et al. (2020). Geometric ergodicity of the bouncy particle sampler. *Annals of Applied Probability*, 30(5):2069–2098.

Durmus, A., Moulines, E., et al. (2017a). Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587.

Durmus, A., Moulines, E., and Pereyra, M. (2018). Efficient bayesian computation by proximal markov chain monte carlo: when langevin meets moreau. *SIAM Journal on Imaging Sciences*, 11(1):473–506.

Durmus, A., Moulines, E., and Saksman, E. (2017b). On the convergence of hamiltonian monte carlo. *arXiv preprint arXiv:1705.00166*.

Eberle, A. et al. (2014). Error bounds for metropolis–hastings algorithms applied to perturbations of gaussian measures in high dimensions. *The Annals of Applied Probability*, 24(1):337–377.

Fernández-Durán, J. J. (2004). Circular distributions based on nonnegative trigonometric sums. *Biometrics*, 60(2):499–503.

Gaspari, G. and Cohn, S. E. (1999). Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554):723–757.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.

Genz, A. (1992). Numerical computation of multivariate normal probabilities. *Journal of computational and graphical statistics*, 1(2):141–149.

Ghosal, S. and Van der Vaart, A. (2017). *Fundamentals of nonparametric Bayesian inference*, volume 44. Cambridge University Press.

Giné, E. and Nickl, R. (2016). *Mathematical foundations of infinite-dimensional statistical models*, volume 40. Cambridge University Press.

Girolami, M. and Calderhead, B. (2011). Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

Glynn, P. W. and Iglehart, D. L. (1989). Importance sampling for stochastic simulations. *Management science*, 35(11):1367–1392.

Goldman, J. V., Sell, T., and Singh, S. S. (2020). Gradient-based markov chain monte carlo for bayesian inference with non-differentiable priors. *under double blind review*.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537.

Green, P. J., Łatuszyński, K., Pereyra, M., and Robert, C. P. (2015). Bayesian computation: a summary of the current state, and samples backwards and forwards. *Statistics and Computing*, 25(4):835–862.

Hairer, M., Stuart, A. M., Vollmer, S. J., et al. (2014). Spectral gaps for a metropolis–hastings algorithm in infinite dimensions. *The Annals of Applied Probability*, 24(6):2455–2490.

Hald, A. (1998). *A History of Mathematical Statistics from 1750 to 1930*, volume 2. Wiley New York.

Hamill, T. M., Whitaker, J. S., and Snyder, C. (2001). Distance-dependent filtering of background error covariance estimates in an ensemble kalman filter. *Monthly Weather Review*, 129(11):2776–2790.

Hashorva, E. (2005). Asymptotics and bounds for multivariate gaussian tails. *Journal of theoretical probability*, 18(1):79–97.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

Hosseini, B. (2017). Well-posed bayesian inverse problems with infinitely divisible and heavy-tailed prior measures. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):1024–1060.

Hosseini, B. (2019). Two metropolis–hastings algorithms for posterior measures with non-gaussian priors in infinite dimensions. *SIAM/ASA Journal on Uncertainty Quantification*, 7(4):1185–1223.

Hosseini, B. and Nigam, N. (2017). Well-posed bayesian inverse problems: Priors with exponential tails. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):436–465.

Hosseini, S., Mordukhovich, B. S., and Uschmajew, A. (2019). *Nonsmooth Optimization and Its Applications*. Springer.

Idier, J. (2013). *Bayesian approach to inverse problems*. John Wiley & Sons.

Iglesias, M. A., Law, K. J., and Stuart, A. M. (2013). Evaluation of gaussian approximations for data assimilation in reservoir models. *Computational Geosciences*, 17(5):851–885.

Iserles, A. and Nørsett, S. P. (2009). From high oscillation to rapid approximation iii: Multivariate expansions. *IMA journal of numerical analysis*, 29(4):882–916.

Ishwaran, H., Rao, J. S., et al. (2005). Spike and slab variable selection: frequentist and bayesian strategies. *The Annals of Statistics*, 33(2):730–773.

Jander, R. (1957). Die optische richtungsorientierung der roten waldameise (formica ruea l.). *Zeitschrift für vergleichende Physiologie*, 40(2):162–238.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kent, J. (1978). Time-reversible diffusions. *Advances in Applied Probability*, pages 819–835.

Kloeden, P. E. and Platen, E. (2013). *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media.

Knapik, B. T., Van Der Vaart, A. W., van Zanten, J. H., et al. (2011). Bayesian inverse problems with gaussian priors. *The Annals of Statistics*, 39(5):2626–2657.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Koltchinskii, V., Lounici, K., Tsybakov, A. B., et al. (2011). Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics*, 39(5):2302–2329.

Konidaris, G., Osentoski, S., and Thomas, P. (2011). Value function approximation in reinforcement learning using the fourier basis. In *Twenty-fifth AAAI conference on artificial intelligence*.

Leimkuhler, B. and Matthews, C. (2016). *Molecular Dynamics.* Springer.

Leimkuhler, B., Matthews, C., and Vlaar, T. (2019). Partitioned integrators for thermodynamic parameterization of neural networks. *arXiv preprint arXiv:1908.11843*.

Lions, P.-L. and Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, 16(3):31–57.

Livingstone, S., Betancourt, M., Byrne, S., Girolami, M., et al. (2019). On the geometric ergodicity of hamiltonian monte carlo. *Bernoulli*, 25(4A):3109–3138.

Ma, Y.-A., Chatterji, N., Cheng, X., Flammarion, N., Bartlett, P., and Jordan, M. I. (2019). Is there an analog of nesterov acceleration for mcmc? *arXiv preprint arXiv:1902.00996*.

Ma, Y.-A., Chen, T., and Fox, E. (2015). A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28:2917–2925.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*.

McVinish, R. and Mengersen, K. (2008). Semiparametric bayesian circular statistics. *Computational statistics & data analysis*, 52(10):4722–4730.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Minchew, B., Simons, M., Hensley, S., Björnsson, H., and Pálsson, F. (2015). Early melt season velocity fields of langjökull and hofsjökull, central iceland. *Journal of Glaciology*, 61(226):253–266.

Mitianoudis, N. (2012). A generalized directional laplacian distribution: Estimation, mixture models and audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2397–2408.

Morzfeld, M., Tong, X. T., and Marzouk, Y. M. (2019). Localization for mcmc: sampling high-dimensional posterior distributions with local structure. *Journal of Computational Physics*, 380:1–28.

Murray, I., Adams, R., and MacKay, D. (2010). Elliptical slice sampling. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 541–548. JMLR Workshop and Conference Proceedings.

Neal, R. (1995). Bayesian learning for neural networks [phd thesis]. *Toronto, Ontario, Canada: Department of Computer Science, University of Toronto*.

Neal, R. (1998). Regression and classification using gaussian process priors. *Bayesian statistics*, 6:475.

Neal, R. M. (2000). Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and computing*, 11(2):125–139.

Neal, R. M. (2003). Slice sampling. *Annals of statistics*, pages 705–741.

Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Nickl, R. and Giordano, M. (2020). Consistency of bayesian inference with gaussian process priors in an elliptic inverse problem. *Inverse Problems*.

Nitanda, A. (2014). Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582.

Parikh, N., Boyd, S., et al. (2014). Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239.

Park, T. and Casella, G. (2008). The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686.

Pereyra, M. (2016). Proximal markov chain monte carlo algorithms. *Statistics and Computing*, 26(4):745–760.

Pereyra, M., Mieles, L. V., and Zygalakis, K. C. (2020). Accelerating proximal markov chain monte carlo by using an explicit stabilized method. *SIAM Journal on Imaging Sciences*, 13(2):905–935.

Peters, E. and de With, G. (2012). Rejection-free monte carlo sampling for general potentials. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 85(2 Pt 2):026703.

Petersen, K. E. (1989). *Ergodic theory*, volume 2. Cambridge University Press.

Polson, N. G., Scott, J. G., Willard, B. T., et al. (2015). Proximal algorithms in statistics and machine learning. *Statistical Science*, 30(4):559–581.

Pusse, F. and Klusch, M. (2019). Hybrid online pomdp planning and deep reinforcement learning for safer self-driving cars. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020. IEEE.

Quarteroni, A., Manzoni, A., and Vergara, C. (2017). The cardiovascular system: mathematical modelling, numerical algorithms and clinical applications. *Acta Numerica*, 26:365–590.

Quiroz, M., Tran, M.-N., Villani, M., and Kohn, R. (2018). Speeding up mcmc by delayed acceptance and data subsampling. *Journal of Computational and Graphical Statistics*, 27(1):12–22.

Ramachandran, D. and Amir, E. (2007). Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591.

Roberts, G. O., Gelman, A., Gilks, W. R., et al. (1997). Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268.

Roberts, G. O., Rosenthal, J. S., et al. (2001). Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367.

Roberts, G. O. and Stramer, O. (2002). Langevin diffusions and metropolis-hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357.

Roberts, G. O., Tweedie, R. L., et al. (1996). Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268.

Rudolf, D. and Sprungk, B. (2018). On a generalization of the preconditioned crank–nicolson metropolis algorithm. *Foundations of Computational Mathematics*, 18(2):309–343.

Sachs, M., Sen, D., Lu, J., and Dunson, D. (2020). Posterior computation with the gibbs zig-zag sampler. *arXiv preprint arXiv:2004.04254*.

Salim, A., Kovalev, D., and Richtárik, P. (2019). Stochastic proximal langevin algorithm: Potential splitting and nonasymptotic rates. *arXiv preprint arXiv:1905.11768*.

Savage, I. R. (1962). Mills' ratio for multivariate normal distributions. *J. Res. Nat. Bur. Standards Sect. B*, 66:93–96.

Sell, T. and Singh, S. S. (2020). Dimension-robust function space mcmc with neural network priors. *arXiv preprint arXiv:2012.10943*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Singh, S. S., Chopin, N., and Whiteley, N. (2013). Bayesian learning of noisy markov decision processes. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(1):4.

Sobol, I. M. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical modelling and computational experiments*, 1(4):407–414.

Stuart, A. M. (2010). Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tierney, L. et al. (1998). A note on metropolis-hastings kernels for general state spaces. *The Annals of Applied Probability*, 8(1):1–9.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

van der Vaart, A. W., van Zanten, J. H., et al. (2008). Rates of contraction of posterior distributions based on gaussian process priors. *The Annals of Statistics*, 36(3):1435–1463.

Vanetti, P., Bouchard-Côté, A., Deligiannidis, G., and Doucet, A. (2017). Piecewise-deterministic markov chain monte carlo. *arXiv preprint arXiv:1707.05296*.

Vono, M., Dobigeon, N., and Chainais, P. (2020). Asymptotically exact data augmentation: Models, properties, and algorithms. *Journal of Computational and Graphical Statistics*, pages 1–14.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. (2020). How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*.

Wibisono, A. (2019). Proximal langevin algorithm: Rapid convergence under isoperimetry. *arXiv preprint arXiv:1911.01469*.

Wojtaszczyk, P. (1997). *A mathematical introduction to wavelets*, volume 37. Cambridge University Press.

Wróblewska, A., Frąckowiak, A., and Ciałkowski, M. (2016). Regularization of the inverse heat conduction problem by the discrete fourier transform. *Inverse Problems in Science and Engineering*, 24(2):195–212.

Xiao, Z. (2019). *Reinforcement Learning: Theory and Python Implementation*. China Machine Press.