Advances in Bayesian Computation: Bridging Modern Machine Learning and Traditional Monte Carlo Methods

Alberto Cabezas González, BSc (Hons), MRes

Department of Mathematics and Statistics

Lancaster University

A thesis submitted for the degree of

Doctor of Philosophy

March, 2025

Advances in Bayesian Computation: Bridging Modern Machine Learning and Traditional Monte Carlo Methods

Alberto Cabezas González, BSc (Hons), MRes.

Department of Mathematics and Statistics, Lancaster University

A thesis submitted for the degree of *Doctor of Philosophy*. March, 2025.

Abstract

Bayesian inference requires solving integrals over probability spaces, but except for certain scenarios, they can only be approximated using Monte Carlo integration. Bayesian computation emerged specifically to develop efficient approximation methods, with sampling algorithms at its forefront. Nowadays, automatic differentiation and fast array computation software make numerical optimization the cutting edge in machine learning. These methods were introduced in Bayesian computation as variational inference and have been essential in accelerating Bayesian inference. This thesis is a methodological contribution to advancements in Bayesian computation, combining fast density approximation techniques with traditional asymptotically exact Monte Carlo methods.

We start with a comprehensive review of Monte Carlo and variational inference techniques. Introduce an efficient, dimension-independent, and gradient-free sampling algorithm leveraging parallel computing architectures. Develop a novel Bayesian computation method that integrates flow matching with Markovian sampling, enhancing the exploration of complex target distributions through adaptive tempering mechanisms. Our work extends Bayesian nonparametric approaches to linear regression models, effectively handling outliers and heteroskedasticity via Dirichlet process mixtures. Finally, we present BlackJAX, a library for Bayesian inference, enabling researchers and practitioners to build and experiment with new algorithms seamlessly. These contributions collectively advance Bayesian computation, offering robust and efficient tools for empirical applications.

Acknowledgements

First, I want to thank Chris, who was closest to me throughout this process. His guidance went beyond helping me write this thesis—his advice helped me navigate the complexities of publishing and provided insight into the world of academia. I learned a lot from Chris and, though my immediate superior, after four years of collaboration, I indeed consider him my friend. I know that his influence will extend far beyond this degree, shaping my career for years to come. I also thank Marco, whose advice on research and independent study was invaluable. His pragmatic, sometimes bleak perspective on academia gave me a unique and necessary lens through which to view my own path.

I am grateful to my co-authors. Working with Louis was an invaluable experience—collaborating with him challenged me in new ways and helped fundamentally shape my research approach. Junpeng, Rémi, Adrien, and everyone involved in BlackJAX for sharing their expertise on inference algorithms and shaping the tools that made the experiments in this thesis possible.

To my friends and colleagues in the department, thank you for your support, the discussions, and the much-needed distractions. I also thank the department's staff, whose help made the logistics of this degree much smoother.

Carlotta, my partner and intellectual peer, has been integral to my life through the years. She knows me better than anyone and has been instrumental in shaping my thinking and growth. She is the person I see myself working alongside in the years to come. Her presence is invaluable, both personally and professionally.

Finally, I dedicate this thesis to my parents, Iovana and Jorge. They have believed in me from the start and have provided support in every possible way. They are my foundation and the reason I could accomplish this. Without them, none of this would have been possible.

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. This thesis does not exceed the maximum permitted word length of 80,000 words including appendices and footnotes, but excluding the bibliography. A rough estimate of the word count is: 30501

Alberto Cabezas González

Contents

1 Introduction **1**

1.1 Bayesian Statistics . . . 3

1.1.1 The Basics of Inference . . . 4

1.1.2 The Computational Challenge of Inference . . . 6

1.1.3 The Linear Model . . . 8

1.1.4 Bayesian Nonparametric Inference . . . 11

1.2 Contributions and Thesis Outline . . . 13

2 Monte Carlo Methods **16**

2.1 Monte Carlo Integration . . . 18

2.1.1 Importance Sampling . . . 19

2.2 Markov Chain Methods . . . 21

2.2.1 The Gibbs Sampler . . . 24

2.2.2 The Metropolis-Hastings Algorithm . . . 25

2.2.3 The Slice Sampler . . . 31

2.2.4 Diagnostic Tools . . . 36

3 Approximate Inference Methods **41**

3.1 Divergence Measure . . . 42

3.1.1 Kullback-Leibler Divergence . . . 43

3.1.2 Monte Carlo Variational Inference . . . 46

3.2 Variational Family . . . 48

3.2.1 Discrete Normalizing Flows . . . 50
3.2.2 Continuous Normalizing Flows . . . 52

4 Transport Elliptical Slice Sampling 55
4.1 Introduction . . . 56
4.2 Transport Elliptical Slice Sampler . . . 57
4.2.1 Elliptical slice sampling . . . 58
4.2.2 Normalizing flows . . . 58
4.2.3 Fixed transport maps with elliptical slice sampling . . . 60
4.2.4 Adaptive transport maps . . . 64
4.2.5 Choice of transport map . . . 66
4.3 Related work . . . 66
4.4 Experiments . . . 68
4.4.1 Biochemical oxygen demand model . . . 69
4.4.2 Sparse logistic regression . . . 71
4.4.3 Regime switching Hidden Markov model . . . 73
4.4.4 Predator-prey system . . . 74
4.5 Discussion . . . 78

5 Bayesian Inference with Markovian Flow Matching 79
5.1 Introduction . . . 80
5.2 Preliminaries . . . 81
5.3 Markovian Flow Matching . . . 84
5.3.1 Continuous Normalizing Flows . . . 84
5.3.2 Flow Matching . . . 85
5.3.3 Flow-informed Random Walk . . . 86
5.3.4 Adaptively-Tempered Markovian Flow Matching . . . 90
5.3.5 Convergence . . . 92
5.4 Related work . . . 94
5.5 Experiments . . . 95

5.5.1 4-mode Gaussian mixture . . . 96
5.5.2 16-mode Gaussian mixture . . . 97
5.5.3 Field system . . . 98
5.5.4 Log-Gaussian Cox point process . . . 101
5.6 Discussion . . . 103

6 Robust Bayesian Nonparametric Variable Selection for Linear Regression 104

6.1 Introduction . . . 105
6.2 Bayesian Variable Selection for Linear Regression . . . 106
6.2.1 Spike-and-slab priors . . . 107
6.2.2 Horseshoe priors . . . 107
6.3 Nonparametric Stochastic Variable Selection . . . 109
6.3.1 Background to Dirichlet processes . . . 109
6.3.2 Dirichlet process mixture model . . . 110
6.3.3 Dirichlet process variable selection . . . 111
6.3.3.1 Posterior inference . . . 112
6.3.3.2 Heavy Tails: Student-t extension . . . 117
6.4 Experiments . . . 120
6.4.1 Homoskedastic and Heteroskedastic Errors . . . 121
6.4.2 Support Recovery . . . 124
6.4.3 Reconstruction of transcription regulatory networks . . . 128
6.5 Discussion . . . 129

7 BlackJAX: Composable Bayesian inference in JAX 131

7.1 Introduction . . . 132
7.2 Design principles . . . 133
7.2.1 Lower-level API . . . 134
7.2.2 Basic components . . . 135
7.2.3 Existing sampling libraries . . . 136

7.3 Past impact of BlackJAX on the practice of Bayesian inference 137
7.4 Roadmap to the future of BlackJAX 138
7.4.1 Enhanced algorithm portfolio 138
7.4.2 Documentation and tutorials 139
7.5 Project openness and development 140
7.6 Discussion 140

8 Conclusions **142**
8.1 Future work 143

Appendix A BlackJAX: Composable Bayesian inference in JAX **145**
A.1 Skeletons 145

References **150**

List of Tables

4.1 Biochemical oxygen demand model. Algorithm diagnostics where $\tau_{\max}$ is the maximum integrated autocorrelation time over all dimensions; ESS is the corresponding minimum effective sample size. Results are averaged over multiple chains of each sampler, and σ_τ is the empirical standard deviation of $\tau_{\max}$ over these runs. 70

4.2 Sparse logistic regression. Algorithm diagnostics where $\tau_{\max}$ is the maximum integrated autocorrelation time over all dimensions; ESS is the corresponding minimum effective sample size. Results are averaged over multiple chains of each sampler, and σ_τ is the empirical standard deviation of $\tau_{\max}$ over these runs. 72

4.3 Regime switching Hidden Markov model. Algorithm diagnostics where $\tau_{\max}$ is the maximum integrated autocorrelation time over all dimensions; ESS is the corresponding minimum effective sample size. Results are averaged over multiple chains of each sampler, and σ_τ is the empirical standard deviation of $\tau_{\max}$ over these runs. 73

5.1 Diagnostics for the two synthetic examples. MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs. 97

5.2 Diagnostics for the two real data examples. KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs. . . 100

6.1 Tabulated results of true positive (TP) and false positive (FP) results with 95% credible intervals for the inclusion of regression coefficients for $p = 50, 100, 200$ with nonzero coefficients TP $= 14, 28, 56$. 127

6.2 Logarithmic-loss errors for the five DREAM Gene Network detection datasets (N1-N5). 129

List of Figures

3.1 Gaussian approximation that underestimates (left) and overestimates (right) the real variance of the skewed target. 45

4.1 Illustration of Algorithm 12 using an exact transport map, i.e. equality in (4.2) holds: sampling from the Banana density $\pi(x_1, x_2) \propto \exp\left(-[x_1^2/8 + (x_2 - x_1^2/4)^2]/2\right)$ using the transport map $T(u_1, u_2) = (\sqrt{8}u_1, u_2 + 2u_1^2)$ starts by transforming the target space to the reference space via a change of variables, drawing samples from an ellipsis on the extended reference space (not pictured) and pushing samples back to the target space. 56

4.2 Samples from the target density $\pi(\theta)$ of the Biochemical oxygen demand model acquired by the TESS algorithm, mapped to $\hat{\phi}(\theta)$ (4.1), with diffeomorphism T_ψ learned from the warm-up procedure of Algorithm 13. With an approximation that overestimates the real variance (right) of our target (left), we can capture its global, non-Gaussian structure and explore it using a dimension-independent and gradient-free method. 71

4.3 Posterior density pair plots for the regime switching hidden Markov model using samples drawn with transport elliptical slice sampling on the left and MEADS (M. D. Hoffman and Sountsov, 2022) on the right. Parameters ρ, σ_1 and σ_2 are log transformed and parameters $p_{1,1}$, $p_{2,2}$ and ξ_{10} are sigmoid function transformed. 75

4.4 Density plots of the approximate posterior distribution for the initial values and scale parameters from the predator-prey system model, drawn with transport elliptical slice sampling on the left and MEADS on the right. 77

5.1 Illustration of a transformation learned using Algorithm 17. **Top**: Standard Gaussian samples are transformed to a multimodal mixture by running the dynamics forward in time given the learned vector field, left to right. **Bottom**: Samples from the multimodal mixture are transformed to standard Gaussian by running the dynamics backwards in time, right to left. 81

5.2 Comparison between MFM, FAB, DDS, and NF-MCMC. Samples from the target density for the 4-mode Gaussian mixture example. . . 98

5.3 Comparison between MFM, FAB, DDS, and NF-MCMC. Samples from the target density for the 16-mode Gaussian mixture example. . 99

5.4 Comparison between MFM, FAB, DDS, and NF-MCMC. Representative samples from the target density for the Field system example. . . 101

5.5 Visualization of the standardized 10×10 square meter plot and the locations of 126 Scots pine saplings in Finland. 102

6.1 Boxplots of estimation error $\|\hat{\beta} - \beta_0\|_2 / \|\beta_0\|_2$ on *Scenario 1* (S1) and *Scenario 2* (S2) for algorithms Dirichlet process horseshoe, Dirichlet process spike-and-slab, Horseshoe, and Spike-and-slab. 122

6.2 Boxplots of estimation error $\|\hat{\beta} - \beta_0\|_2 / \|\beta_0\|_2$ on *Scenario 1* (S1) and *Scenario 2* (S2) for algorithms Dirichlet process horseshoe, Dirichlet process spike-and-slab, Horseshoe, and Spike-and-slab on their heavy tailed Student-t extension. 123

6.3 Boxplots of the derived number of clusters K from the posterior of $\sigma_1, \ldots, \sigma_n$ for $p = 200$ for algorithms Dirichlet process horseshoe, Dirichlet process spike-and-slab, with a Gaussian likelihood on the left and a Student-t likelihood on the right, where the true number of clusters is $K = 5, 5, 6, 7, 9$ for $n = 10, 20, 50, 100, 200$. 125

6.4 Histogram of the derived number of clusters K from the posterior of $\sigma_1, \ldots, \sigma_n$ for $n = 200$ and $p = 50$, 200, and 2000, where the two number of clusters is $K = 9$. 125

6.5 Histogram of the derived number of clusters K from the posterior of $\sigma_1, \ldots, \sigma_n$ for $n = 200$ and $p = 50$, 200, and 2000 on their heavy-tailed Student-t extension, where the two number of clusters is $K = 9$. . . . 126

Chapter 1

Introduction

A syllogism applies deductive reasoning to arrive at a specific conclusion based on general assumptions. It is a logical argument that uses general knowledge to grasp specific phenomena. For instance, it was once believed that atoms were the smallest units of matter. By adding a second premise, that the smallest units of matter cannot be divided, we conclude that atoms cannot be divided.

This syllogism was proven wrong by the discovery of subatomic particles, and it became evident that atoms could be divided into protons, neutrons, and electrons. As scientific research progressed, new information contradicted the initial premises, invalidating the conclusion. Deductive reasoning, while logically valid, relies heavily on the correctness of its premises, and since its conclusions are based on the best available knowledge, they can be misguided.

Bacon's (1620) *Novum Organum* first formalized an opposing logic, renovating Aristotle's original deductive logic on *Organum*. Bacon argued that true knowledge must be derived from the natural world, not from abstract reasoning alone. His method emphasized inductive reasoning, where general principles are derived from specific assumptions. Nowadays, it is widespread in any field to reason inductively when gaining general knowledge of the physical world, observing, experiencing, and gathering insights to discern broader patterns from specific instances.

It took over a century for the logic of inductive reasoning to be described by a

unique mathematical result known as Bayes' Theorem (Bayes, 1763),

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{1.1}$$

characterized by the probabilities P of events A and B. Everything is interpreted through the concept of conditional probability, and the notion of learning by experience described by stochastic dependence and information updating. The theorem describes how information on event B can be used to update expectations on event A given that there is some known dependence between both events. The theorem states the logic of inductive reasoning in terms of probabilities.

Although the theoretical groundwork was laid, implementing Bayes' theory in practice would require avoiding some ideological hurdles in the early 20th century. The debate became surprisingly personal between those who perceived probabilities as an objective law of nature and those who saw them as a subjective perception of the observer. The main point of conflict was the need for an initial or prior belief on the subjective probability of an event from which to update. Setting prior probabilities is a major methodological challenge for Bayesian theory but offers a complete probabilistic assessment of knowledge update from experimentation in exchange.

An integration problem also limited the practice of Bayesian statistics. It was not until the late 20th century that computers became widely available, and Bayesian computation arose to approximate those intractable integrals. This work is a methodological contribution to the approximations used in Bayesian inference. It tries to bridge the gap between fast density approximation methods using numerical optimization, a cornerstone of modern machine learning methods, and asymptotically exact Monte Carlo methods, the traditional solution in Bayesian computation. It also presents a modern software suite for the use of classic and the development of novel Bayesian inference algorithms and an algorithm for inference in linear variable selection models with robust variance.

1.1 Bayesian Statistics

This work is based on the subjective or Bayesian notion of probability. We are interested in two probability assessments: that of an event A, representing the current state of knowledge of the event, $P(A)$; and that of an event A conditional on the occurrence of event B, representing an update on our knowledge of event A based on our knowledge of event B, $P(A|B)$. Probabilities need to be coherent for them to be useful. This means assigning a unique value to any event A such that $0 \leq P(A) \leq 1$, assigning $P(A) = 1$ if A is sure to happen and $P(A) = 0$ if A is sure not to happen.

The theorem of compound probability tells us that another way to express coherence is to make sure that for any two events, A and B,

$$P(A \cap B) = P(B)P(A|B). \tag{1.2}$$

This equation gives us some insight into the properties of conditional probabilities. Consider the case where events A and B are independent, then $P(A \cap B) = P(A)P(B)$, making $P(A|B) = P(A)$. In other words, if A and B are independent, our current knowledge about event A remains unchanged if we know the occurrence of event B. From it, we can also derive Bayes' theorem (1.1) given the symmetry $P(A \cap B) = P(B \cap A)$, giving us a way to represent the update from initial to conditional beliefs mathematically.

Kolmogorov (1933) gave an axiomatic approach to probabilities, describing them regardless of their statistical interpretation, by drawing an analogy between probabilities and measures. Start by defining a set of elements or outcomes Ω whose probability we care to measure. Since we might want to measure various elements of Ω at a time, we must pair it with a set of subsets of Ω, constituting a σ-algebra $\mathcal{F}$. Three axioms define a σ-algebra: $\Omega \in \mathcal{F}$, if $E_1 \in \mathcal{F}$ then $\overline{E_1} \in \mathcal{F}$, and for any possibly infinite set of elements $\{E_i \ : \ i = 1, 2, \dots\}$ it must be that $\cup_i E_i \in \mathcal{F}$. Then, we can define probabilities on the σ-algebra using three axioms again:

(i) for any $E_1 \in \mathcal{F}$ it corresponds a probability $0 \leq P(E_1) \leq 1$;

(ii) since Ω represents all outcomes $P(\Omega) = 1$;

(iii) for every sequence $E_1, E_2, \ldots$ of pairwise disjoint elements of $\mathcal{F}$,

$$P\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} P(E_i). \tag{1.3}$$

These three properties allow us to have coherent probabilities that behave coherently when using limits. The triplet $(\Omega, \mathcal{F}, P)$ is called a probability space and describes any random experiment.

Kolmogorov's definition allows us to borrow useful tools from measure theory like the Lebesgue integral, which we use to define a Lebesgue-integrable function called the *density function* $p(\omega)$ of P (with respect to the Lebesgue measure) such that

$$P(E) = \int_E p(\omega) d\omega. \tag{1.4}$$

1.1.1 The Basics of Inference

In terms of an experiment, different runs are likely to generate different outcomes $\omega \in \Omega$, where groups of these different outcomes are contained in the σ-algebra $\mathcal{F}$. In practice, observations are a random variable $Y : \Omega \to \mathbb{R}$, transforming the probability space outcomes to real values. For any element $A \in \mathcal{R}$, where $\mathcal{R}$ is the σ-algebra of $\mathbb{R}$, the probability of event A is

$$P(Y \in A) = P(\{\omega \ : \ \omega \in \Omega, \ Y(\omega) \in A\}). \tag{1.5}$$

We can think of the sequence $Y_1, \ldots, Y_n$ as observations or data points of our experiment; these allow us to reason inductively, that is, make inferences on the probabilities of the experiment (general) from observations (specific).

If we knew the probability measure of the experiment, there would be no statistical problem. Instead, we assume a family of probability measures on the measurable space $(\Omega, \mathcal{F})$ indexed by a parameter space $\mathcal{X}$, $\{P_x \ : \ x \in \mathcal{X}\}$, with density functions—also referred to as the *likelihood* of data—,

$$P(A|x) = \int_A p(y|x) dy, \quad \forall x \in \mathcal{X}. \tag{1.6}$$

The parameter space $\mathcal{X}$ can be a set of everything from real numbers to distributions in the case of nonparametric Bayesian inference. In all cases, we are interested in updating our knowledge about $\mathcal{X}$, which elements are more or less likely to generate the experiment, given observations from $\mathbb{R}$, represented as the *probability model* in its simplified form

$$(\mathbb{R},\ p(y|x),\ x \in \mathcal{X})\,. \tag{1.7}$$

Replicating the experiment under the same conditions gives us a sequence of independent and identically distributed observations. The joint probability of this sequence will equal the product of each of their probabilities. Then a *statistical model* is that containing all the sampled results, representing acquired knowledge from n independent repetitions of the experiment,

$$\left(\mathbb{R}^n,\ \prod_{i=1}^{n} p(y_i|x),\ x \in \mathcal{X}\right). \tag{1.8}$$

A sequence of independent and identically distributed realizations represents all the acquired knowledge or *data*, $\mathcal{D} = \{y_i\ :\ i = 1, \ldots, n\}$.

Similarly, we describe the uncertainty around the parameter space by making it a random variable X with density function $p(x)$, called a *prior density*. It represents the unconditional or current knowledge about the parameter space $\mathcal{X}$ and the starting point for the Bayesian update. The choice of prior density is crucial and will dominate posterior knowledge of the experiment. Although practitioners must be careful not to add frivolous prior information to their knowledge, using it to convey background or expert knowledge is crucial for effective inference.

Using Bayes' theorem (1.1), the prior knowledge is updated using the likelihood of observing the independent and identically distributed sequence of experimental results by taking the product of its densities, creating a *posterior density* representing

an update on our knowledge,

$$p(x|\mathcal{D}) = Z^{-1} \prod_{i=1}^{n} p(y_i|x)p(x) \tag{1.9}$$

$$Z = \int_{\mathcal{X}} \prod_{i=1}^{n} p(y_i|x)p(x)dx. \tag{1.10}$$

where Z is the *model evidence* or marginal likelihood of observations, which ensures the posterior is the density of a probability space. In general, we cannot evaluate the model evidence as it requires computing an intractable integral, making the posterior (1.9) known up to a constant of proportionality.

1.1.2 The Computational Challenge of Inference

The computational challenge in Bayesian statistics is evaluating integrals over the posterior probability space. For a generic function on the parameter space $h(x)$, the problem is evaluating the integral

$$\mathbb{E}[h(X)|\mathcal{D}] = \int_{\mathcal{X}} h(x)p(x|\mathcal{D})dx. \tag{1.11}$$

For example, in tasks such as calculating posterior averages over the parameter space $\mathbb{E}[X|\mathcal{D}] = \int_{\mathcal{X}} xp(x|\mathcal{D})dx$ or finding the density of future observations given known data $p(y|\mathcal{D}) = \int_{\mathcal{X}} p(y|x)p(x|\mathcal{D})dx$, known as the *posterior predictive density*. These integral computations are pivotal for drawing inferences from the model, as they allow for summarising posterior distributions and predicting future observations based on current data. Two principal strategies emerge to address this integral challenge: sampling and density function approximation.

Sampling Approach: Generate a series of samples of the parameter space, used to approximate integrals with sums. Sampling aims to reproduce the target distribution's statistical properties by generating a sequence of parameter configurations $\{x_i \ : \ i = 1, \ldots, N\}$, which converge to the desired integral over the distribution as per the law of large numbers. This approach uses a robust numerical method that ensures convergence to expectations under the posterior distribution called Monte Carlo integration, which is the focus of Chapter 2 of this thesis.

Ideally, the sequence of samples is generated independently using a direct probabilistic method. While straightforward, direct probabilistic methods often struggle with the high-dimensional, multimodal distributions characteristic of Bayesian posteriors. This difficulty in maintaining a reasonable acceptance rate makes these methods less favoured for complex Bayesian models. Alternatively, Markov chain methods introduce a Markovian dependence on the generated samples, adding crucial information for navigating the probability space. In practice, the success of a sampling strategy hinges on its ability to efficiently explore the state space, particularly for distributions with complex geometries.

Density Approximation Approach: Optimizes the choice within a specified family of density functions, minimizing a notion of distance between the target and approximate density, an indirect approach to solving integrals. This method seeks a tractable density $q(x)$ approximating the posterior $p(x|\mathcal{D})$, using variational inference techniques to minimize their divergence. This optimisation avoids inference using the complex posterior altogether, instead using an approximate imitation. This approach is the focus of Chapter 3 of this thesis.

Although the approximation is never exact, the method accelerates the inference process, often yielding much quicker results than sampling methods. Recent advancements in this domain have been directed towards enhancing the flexibility of the family of density functions used for approximation and refining the divergence measures employed in the optimization process. The density function approximation approach strives to balance the trade-off between computational efficiency and approximation accuracy. Approximate inference remains a compelling and viable strategy for Bayesian inference, especially in scenarios where speed is of the essence.

The interplay between these methodologies underpins the computational framework of Bayesian inference. Each approach addresses the challenge of approximating integrals on the posterior probability space from different angles. By delving into the specifics of each approach in the following chapters, we pave the way for a comprehensive understanding of their use.

1.1.3 The Linear Model

A good starting inference example is the linear model, which provides a foundational approach for understanding relationships between variables. In this context, a dependent or endogenous variable Y is modelled through a linear combination of one or more independent or exogenous variables $z_1, \ldots, z_k$,

$$Y = \sum_{j=1}^{k} \beta_j z_j + \epsilon, \tag{1.12}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_k]^T$ are unknown parameters we need to make inferences on, and ϵ represents a random error term accounting for the influence of unobservable variables and measurement error.

The i-th observation in a dataset y_i has its own set of explanatory variables $z_{i,1}, \ldots, z_{i,k}$ and error term ϵ_i and is assumed to be generated by,

$$y_i = \sum_{j=1}^{k} \beta_j z_{i,j} + \epsilon_i. \tag{1.13}$$

The objective of the inference problem is to use the data $\mathcal{D} = \{(y_i, z_{i,1}, \ldots, z_{i,k}) \ : \ i = 1 \ldots, n\}$ to acquire knowledge about unknown parameters. Probabilities are added through the error term ϵ_i, which in its most basic form are assumed independent and identically distributed, but more robust solutions relax one or both of these assumptions. This section will focus on the simplest modelling choice while Chapter 6 presents an inference algorithm for more robust modelling.

Organizing the data into a matrix format for multiple observations, the matrix Z with shape $n \times k$ contains all observations on the k independent variables,

$$Z = \begin{bmatrix} z_{1,1} & \cdots & z_{1,k} \\ \vdots & \ddots & \vdots \\ z_{n,1} & \cdots & z_{n,k} \end{bmatrix}. \tag{1.14}$$

Putting all dependent observations and error terms in vectors $\mathbf{y} = [y_i, \ldots, y_n]^T$ and $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_n]^T$, the complete model is represented as a linear equation,

$$\mathbf{y} = Z\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{1.15}$$

The error terms $\boldsymbol{\epsilon}$ are assumed to be independent and Gaussian with zero mean and variance σ^2,

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_n). \tag{1.16}$$

Then the likelihood function for the linear model, given the linear parameters and the variance, is defined as

$$p(\mathbf{y}|Z, \boldsymbol{\beta}, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - Z\boldsymbol{\beta})^\top(\mathbf{y} - Z\boldsymbol{\beta})\right). \tag{1.17}$$

After assigning prior distributions to the parameters $X = (\boldsymbol{\beta}, \sigma^2)$, depicting our beliefs about them before observing the data, their posterior distribution given the data can be determined using Bayes' theorem,

$$p(\boldsymbol{\beta}, \sigma^2|\mathbf{y}, Z) \propto p(\mathbf{y}|Z, \boldsymbol{\beta}, \sigma^2)p(\boldsymbol{\beta}, \sigma^2), \tag{1.18}$$

where $p(\boldsymbol{\beta}, \sigma^2)$ is the joint prior distribution of the parameters.

Choosing an appropriate prior distribution is a critical aspect of Bayesian inference. The choice of a prior can range from expressing total ignorance to incorporating specific, informed beliefs based on previous studies or expert opinion. In ignorance, a non-informative prior, such as a flat or Jeffreys' prior, is used. Jeffreys's (1946) prior is particularly favoured in Bayesian statistics for its property of being invariant under reparameterization, providing a uniform prior over all scales of the parameter.

On the other hand, prior knowledge can be introduced into the model using conjugate priors, a practical option resulting in a tractable posterior: if the prior is chosen from a specific family of distributions, and the likelihood density has a specific form, the posterior distribution resulting from the Bayesian update remains within that same family. Conjugacy simplifies the computational issue since the posterior distribution is known exactly.

Consider the case of known variance parameter σ^2, then the conjugate prior for the linear parameters $\boldsymbol{\beta}$ is a multivariate normal distribution,

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}_0, C), \tag{1.19}$$

where $\boldsymbol{\beta}_0$ is the mean vector and C is the covariance matrix, indicating our prior expectation and uncertainty about the values of $\boldsymbol{\beta}$. Assuming density (1.17) for the observations, the posterior distribution for $\boldsymbol{\beta}$ is also normal, with covariance matrix and mean vector:

$$C^* = \left(\sigma^{-2} Z^T Z + C^{-1}\right)^{-1}, \tag{1.20}$$

$$\boldsymbol{\beta}^* = C^* \left(\sigma^{-2} Z^T y + C^{-1} \boldsymbol{\beta}_0\right). \tag{1.21}$$

Using conjugate priors the posterior mean $\boldsymbol{\beta}^*$ is a weighted average between the least squares estimator $\hat{\boldsymbol{\beta}} = (Z^T Z)^{-1} Z^T y$ and the prior mean $\boldsymbol{\beta}_0$, with weights derived from the precision of the data $\sigma^{-2} Z^T Z$ and the precision of the prior C^{-1}.

When both the regression coefficients $\boldsymbol{\beta}$ and the variance σ^2 are unknown, their joint conjugate prior is a multivariate normal for $\boldsymbol{\beta}$ with its covariance depending on σ^2 with inverse Gamma distribution,

$$\boldsymbol{\beta} | \sigma^2 \sim \mathcal{N}(\boldsymbol{\beta}_0, \sigma^2 C), \tag{1.22}$$

$$\sigma^2 \sim \mathrm{IG}(a_0, b_0). \tag{1.23}$$

Assuming again a Gaussian density for the observations, the posterior distribution is also a Gaussian-inverse Gamma distribution with parameters:

$$C^* = \left(Z^T Z + C^{-1}\right)^{-1}, \tag{1.24}$$

$$\boldsymbol{\beta}^* = C^* \left(Z^T y + C^{-1} \boldsymbol{\beta}_0\right), \tag{1.25}$$

$$a^* = a_0 + \frac{n}{2} \tag{1.26}$$

$$b^* = b_0 + \frac{1}{2}\left(\mathbf{y}^T \mathbf{y} + \boldsymbol{\beta}_0^T C^{-1} \boldsymbol{\beta}_0 + \boldsymbol{\beta}^{*T} C^{*-1} \boldsymbol{\beta}^*\right). \tag{1.27}$$

Since we know the exact form of the posterior distribution, we can evaluate some functions in the integral problem (1.11). We can also generate independent samples from the posterior distribution, assuming we can generate independent Gaussian and inverse Gamma samples. This makes inference straightforward to approximate using Monte Carlo integration, discussed in Chapter 2.

1.1.4 Bayesian Nonparametric Inference

When the parameter space is a set of distributions, inference is said to be done in a nonparametric space. In this case, the choice in the parameter space defines the likelihood of the data, thus making fewer, possibly unverifiable, assumptions about the data-generating mechanism. Bayesian nonparametric methods are particularly useful when the data suggest a model complexity that grows with the size of the dataset or when a few parameters cannot neatly summarize the data-generating mechanism.

Let $\mathcal{P}$ be the space of probability density functions such that any probability distribution P, on the measurable space $(\Omega, \mathcal{F})$, has its density function $p \in \mathcal{P}$. Then, the statistical nonparametric model, containing all the acquired knowledge from n independent repetitions of the experiment,

$$\left(\mathbb{R}^n,\ \prod_{i=1}^{n} p(y_i),\ p \in \mathcal{P}\right). \tag{1.28}$$

A density can no longer describe prior knowledge, and the posterior distribution cannot be represented using Bayes' formula (1.1); instead, stochastic processes define random probability measures and the posterior is derived analytically using conjugacy. Nonetheless, using priors over function spaces provides the flexibility to model an infinite array of possible distributions defining the likelihood of the data.

Dirichlet Process Prior: Introduced by Ferguson (1973), the Dirichlet process is the first important breakthrough in constructing nonparametric priors. It is practically attractive because of its conjugacy: under certain conditions on its hyperparameters, it yields a Dirichlet process posterior if used as a prior.

The Dirichlet process is characterized by a base measure P_0 and a concentration parameter α, denoting a random probability measure with Dirichlet process distribution as

$$P \sim DP(\alpha, P_0). \tag{1.29}$$

Then, for a measurable partition $(E_1, \ldots, E_k)$, the vector $(P(E_1), \ldots, P(E_k))$ follows a Dirichlet distribution with parameters $(\alpha P_0(E_1), \ldots, \alpha P_0(E_k))$. Here, P_0 represents

the prior mean measure, while α determines the variance around P_0. A higher α indicates greater confidence in P_0, leading to fewer deviations from the base measure.

One common representation of the Dirichlet process is the Chinese restaurant process (Picard and J. Pitman, 2006). It draws the analogy to a Chinese restaurant with an infinite number of tables that serve an infinite number of dishes and sequentially sit an infinite number of customers. Each table serves a unique dish and counts the customers at that table n_t. The first customer $n = 1$ sits at any table and is served a dish sampled from P_0. Every customer after that $n \geq 1$ sits at any of the occupied tables t with probability $n_t/n + \alpha$ and is served the dish from that table or at a new table with probability $\alpha/n + \alpha$ and is served a new dish sampled from P_0.

These describe the prediction rules of the process, specifying its underlying random predictive distribution: given an observed sample $(y_1, \ldots, y_n)$ from P, with K unique values $(\hat{y}_1, \ldots, \hat{y}_K)$ with counts $(n_1, \ldots, n_K)$, then

$$P(Y_{n+1}|y_1, \ldots, y_n, \alpha, P_0) = \sum_{t=1}^{K} \frac{n_t}{n+\alpha}\delta_{\hat{y}_i} + \frac{\alpha}{n+\alpha}P_0. \tag{1.30}$$

This results in a partitioning of the data where the number of clusters grows logarithmically with the number of observations, providing a framework for clustering that also naturally handles the emergence of new clusters as more data is observed.

If we assume the distribution of our data has a Dirichlet process prior,

$$Y \sim P \tag{1.31}$$

$$P \sim DP(\alpha, P_0), \tag{1.32}$$

and suppose we observe data $y_1, \ldots, y_n$ drawn from P. By conjugacy of the Dirichlet process, the posterior distribution of P given the data is also a Dirichlet process, with updated concentration parameter $\alpha^* = \alpha + n$ and base measure given by the predictive rule (1.30). This property allows for the development of efficient Gibbs sampling algorithms to sample from the posterior predictive distribution of the data.

1.2 Contributions and Thesis Outline

The main material in this thesis is presented in six chapters, which contain background on the area (Chapters 2 and 3) and new research that has been submitted for journal publication (Chapters 4, 5, 6, and 7). A brief outline for each chapter is as follows:

Chapter 2: Monte Carlo Methods

> Provides an introduction to Monte Carlo methods. It covers fundamental techniques such as rejection sampling, importance sampling, and various instances of Markov chain Monte Carlo methods. The methods are detailed, offering insights into their theoretical background and practical applications. The chapter serves as foundational material for the development of new methods.

Chapter 3: Approximate Inference Methods

> Provides an introduction to variational inference methods. It begins by discussing divergence measures, particularly the Kullback-Leibler divergence. Then, it shifts to the variational family, particularly normalizing flows and their use as flexible variational approximations. This treatment lays the foundation for applying variational inference techniques in subsequent chapters.

Chapter 4: Transport Elliptical Slice Sampling

This chapter is a journal contribution and has been published with co-author Professor Christopher Nemeth. The paper appeared in Proceedings of The 26th International Conference on Artificial Intelligence and Statistics. PMLR, 2023. p. 3664-3676.

> Introduces a new framework for efficient sampling from complex probability distributions by combining normalizing flows with elliptical slice sampling. The method learns a map from the non-Gaussian target distribution to an approximately Gaussian one, then uses the elliptical

slice sampler to sample from this transformed distribution. Our transport elliptical slice sampler leverages parallel computer architectures to run multiple Markov chains simultaneously, enhancing efficiency.

Chapter 5: Bayesian Inference with Markovian Flow Matching

This chapter is a journal contribution and has been published with co-authors Dr Louis Sharrock and Professor Christopher Nemeth. The paper appeared in Advances in Neural Information Processing Systems, vol. 37, 2024.

Adapts flow matching for probabilistic inference. Building on the flow matching method by Lipman et al. (2022), we integrate Markovian sampling to evaluate the flow matching objective. Our sequential method employs Markov chain samples to define the flow matching objective's probability path, enhanced by an adaptive tempering mechanism to detect multiple modes in the target distribution. We establish convergence to the local optima of the flow matching objective and demonstrate our methods using various examples.

Chapter 6: Robust Bayesian Nonparametric Variable Selection for Linear Regression

This chapter is a journal contribution and has been published with co-authors Dr Marco Battiston and Professor Christopher Nemeth. The paper appeared in Stat, Volume 13, Issue 2, 2024. p. e696.

Presents a Bayesian nonparametric approach to linear regression that addresses variable selection, outliers, and heteroskedasticity. Our model is a Dirichlet process scale mixture that offers robust performance in the presence of common data issues. We derive closed-form full conditional distributions for all parameters, enabling an efficient Gibbs sampler for posterior inference. Additionally, we extend the model to handle heavy-tailed response variables.

Chapter 7: BlackJAX: Composable Bayesian inference in JAX

This chapter is a journal contribution and has been submitted for publication with co-authors Dr Adrien Corenflos, Dr Junpeng Lao, and Dr Rémi Louf. The paper is available as an arXiv preprint, arXiv:2402.10797.

> Introduces BlackJAX, a library for Monte Carlo and approximate inference algorithms. BlackJAX allows users to build and experiment with new algorithms using a functional approach. Written in pure Python and using JAX for array computations (Bradbury et al., 2018), BlackJAX efficiently runs on CPUs, GPUs, and TPUs. It integrates seamlessly with probabilistic programming languages, working directly with unnormalized target log density functions.

Chapter 2

Monte Carlo Methods

The sampling of complex posterior distributions begins with generating random variables, that might be transformed using information from the target posterior, and are then chosen or weighted to ensure they target the correct distribution. In other words, Monte Carlo methods hinge on simulating a sequence of independent and identically distributed random variables from well-characterized distributions. This process is anchored in the generation of uniform random variables. How to generate uniform random variables falls beyond the scope of this work; still, we explore foundational methods that transition from uniform to non-uniform random variables.

The inverse transform method generates simple random variables $\mathcal{X} = \mathbb{R}$ by converting the variability of a uniform random variable U in $[0, 1]$ to that of a random variable X, using its *distribution function*,

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} p(x)dx. \tag{2.1}$$

Make a random variable X a function from $[0, 1]$ to $\mathbb{R}$, mapped with the *generalized inverse function* F^{-1} of a non-decreasing function F on $\mathbb{R}$,

$$F^{-}(u) = \inf\{x : F(x) \geq u\}. \tag{2.2}$$

A random variable $X \sim P$ is generated when transforming a uniform random variable $F^{-}(U) \sim P$ (Lemma 2.4 in Robert and Casella, 2004). Distributions with explicit

forms of F^-, such as the exponential, double-exponential, or Weibull distributions, allow for practical implementation. However, this method is confined to a select group and alternative techniques do not depend on the analytical properties of densities, which achieve wider applicability.

A different approach is needed for simulating general multivariate random variables $\mathcal{X} = \mathbb{R}^d$ with density p without relying on p's analytical form beyond its evaluation. The essence of this approach is in the representation

$$p(x) = \int_0^{p(x)} du, \tag{2.3}$$

making p the marginal density of the uniform joint distribution on (X, U) restricted to the space where $0 < u < p(x)$. By introducing an auxiliary variable U, the problem becomes generating X and U from their joint uniform distribution in the set $\{(x, u) : 0 < u < p(x)\}$ (Theorem 2.15 in Robert and Casella, 2004). This technique sidesteps the direct use of p, except for calculating $p(x)$, offering a universal solution for simulating any random variable with a density function.

The problem remains, however, of finding the subset of $\mathbb{R}^d$ such that $u < p(x)$ for any given u. One way around it is finding a sampleable density function q and a constant $M \geq 1$ such that $p(x) \leq Mq(x)$ for all $x \in \mathbb{R}^d$, then generating $Z \sim q$ and U uniformly in $\{u \ : \ 0 < u < Mq(z)\}$ until $0 < u < p(z)$, making $Z \sim p$ (Corollary 2.17 in Robert and Casella, 2004). The implementation of this result is called the Accept-Reject method, detailed in Algorithm 1.

This discussion lays the groundwork for understanding the core principles of simulation. While direct simulation methods such as the Accept-Reject provide a foundational framework, simulation's state of the art has evolved, embracing more sophisticated methods, which we will explore in subsequent sections. Nevertheless, we will return to these principles at the end of this chapter when discussing the slice sampler. For a thorough discussion of Monte Carlo methods commonly used in practice, see Robert and Casella (2004), and for a general discussion of the generation of non-uniform random variables, see Devroye (2006).

Algorithm 1 Accept-Reject algorithm

Require: q, p, M

1: Generate $x \sim q$
2: Generate $u \sim \text{Uniform}(0, 1)$
3: **if** $u \leq p(x)/Mq(x)$ **then**
4: Return x
5: **else**
6: Go to 1.
7: **end if**

2.1 Monte Carlo Integration

Consider evaluating an expectation

$$\mathbb{E}_p[h(X)] = \int_{\mathcal{X}} h(x)p(x)dx, \tag{2.4}$$

where $h(x)$ is a real-valued function on space $\mathcal{X}$, but doing it analytically is infeasible. With a series $X_1, X_2, \ldots, X_N$ of random variables, independent and identically distributed with density p, one can define an estimator

$$\overline{h} = \frac{1}{N} \sum_{i=1}^{N} h(x_i). \tag{2.5}$$

This estimator, essentially the sample mean of the transformed variables $h(x_i)$, leverages the Central Limit Theorem to show that $\overline{h}$ converges in distribution to a Gaussian distribution centred on $\mathbb{E}_p[h(X)]$. The variance of the estimate will depend on the variance of the function $h(x)$,

$$\sigma^2 = \text{Var}[h(X)] = \mathbb{E}_p\left[h(X)^2\right] - \mathbb{E}_p[h(X)]^2, \tag{2.6}$$

making the estimator asymptotically unbiased with a variance that scales inversely proportional to the square root of the sample size,

$$\sqrt{N}\left(\overline{h} - \mathbb{E}_p[h(X)]\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right). \tag{2.7}$$

The elegance of Monte Carlo integration lies in its simplicity and the foundational principles of elementary statistics it employs. For example, constructing an asymptotic 95% confidence interval for $\mathbb{E}_p[h(X)]$ can be achieved through

$$\overline{h} \pm 1.96 \cdot \frac{\hat{\sigma}}{\sqrt{N}}, \tag{2.8}$$

where $\hat{\sigma}^2$ is the empirical variance of $h(x_i)$,

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(h(x_i) - \overline{h})^2. \tag{2.9}$$

Monte Carlo integration provides a robust framework for approximating expectations unattainable analytically, with their convergence properties firmly rooted in the Central Limit Theorem. While the scale of the estimate's variance inherently limits the method's precision, its ability to provide estimations where analytical solutions are impossible makes it an indispensable tool.

2.1.1 Importance Sampling

Importance sampling emerges as a principal alternative to direct sampling. This method generates samples $x_1, x_2, \ldots, x_N$ from a carefully chosen distribution q, distinct from the original distribution p, similar to the Accept-Reject method. Then, since integrals such as (2.4) have an alternative representation incorporating both q and p,

$$\mathbb{E}_p[h(X)] = \mathbb{E}_q\left[h(X)\frac{p(X)}{q(X)}\right] = \int_{\mathcal{X}} h(x)\frac{p(x)}{q(x)}q(x)dx, \tag{2.10}$$

samples from q can be reweighed to get the importance Monte Carlo approximation,

$$\overline{h}_I = \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)}h(x_i), \tag{2.11}$$

which converges like regular Monte Carlo (2.7) with variance

$$\sigma^2 = \mathbb{E}_q\left[h^2(X)\frac{p^2(X)}{q^2(X)}\right] - \mathbb{E}_q\left[h(X)\frac{p(X)}{q(X)}\right]^2. \tag{2.12}$$

Importance sampling's strength lies in its flexibility in choosing the instrumental distribution q, which can significantly affect the estimator's efficiency and feasibility. This method offers robustness for sensitivity analyses within Bayesian frameworks, allowing the same sample generated from q to be applied across various functions h and different densities p. This characteristic is particularly valuable for applications requiring multiple evaluations under varying modelling assumptions.

Importance density q significantly influences the variance and stability of the estimate $\overline{h}_I$. Although the choice of q offers considerable flexibility, with the estimator converging almost surely for a wide range of densities, not all choices are equally beneficial. The variance of the importance sampling estimator is finite when its weighted second moment,

$$\mathbb{E}_q \left[h^2(X) \frac{p^2(X)}{q^2(X)} \right] = \int_{\mathcal{X}} h^2(x) \frac{p^2(x)}{q(x)} dx, \tag{2.13}$$

is finite. This highlights the unsuitability of instrumental distributions q with tails lighter than those of p, as these can lead to unbounded ratios of p/q, thus inflating the variance to infinity for all target functions h. When the ratio p/q is unbounded, the weights $p(x_i)/q(x_i)$ assigned to samples can vary significantly, attributing excessive influence to some samples rather than others and causing the estimator to fluctuate significantly between iterations.

The choice of the instrumental density q plays a similar role in the Accept-Reject algorithm. Hence, a q satisfying $p(z) < Mq(z)$ can also serve as an instrumental density for importance sampling. This ensures the finiteness of variance for importance sampling estimators and highlights an intersection between the two methods. However, the Accept-Reject method produces a sample as a subsample of another set simulated from q, introducing a layer of selection that importance sampling does not inherently possess, making a comparison between the variances of the two methods complex.

2.2 Markov Chain Methods

If generating independent and identically distributed samples is impractical, because there is no explicit form for the inverse function or there is no density that closely dominates the target, we can instead guide the generation of the new samples using information from the previous samples. Specifically, make the sequence of random variables $X_1, X_2, \ldots, X_N$ a time-homogeneous Markov chain, where each variable X_{i+1} is conditional on the previous X_i alone according to a probability transition kernel $K(dx'|x)$, where for any $x \in \mathcal{X}$, $K(dx'|x)$ is a probability measure on the parameter space $\mathcal{X}$.

Dependent samples can be used for the Monte Carlo estimator (2.5), but the convergence property of the estimator changes since the samples are no longer independent. For a stationary Markov chain, the Central Limit Theorem (2.7) holds, but the variance is not simply (2.6) and also includes the sum of the covariances between $h(X_i)$ and $h(X_{i+k})$ for all lags k,

$$\sigma^2 = \mathrm{Var}[h(X_i)] + 2\sum_{k=1}^{\infty} \mathrm{Cov}[h(X_i), h(X_{i+k})]. \tag{2.14}$$

This expression captures the impact of correlations within the Markov chain, adding to the estimate's uncertainty.

For a Markov chain to be stationary, the distribution of the initial X_0 needs to be the stationary distribution of the Markov chain. Truly stationary Markov chains are never used since the ability to simulate the invariant distribution from the outset would avoid the need for a Markov chain approach. Yet, the theory developed for stationary chains holds significant value, as it offers insights into the behaviour of nonstationary chains under stationary transition dynamics but with different starting distributions.

Given conditions on the Markov transition kernel of stationarity on the target density and aperiodic irreducibility, if the Central Limit Theorem applies to one initial distribution, it extends to all initial distributions, maintaining the same asymptotic variance (Proposition 17.1.6 in Meyn and Tweedie, 2012). This crucial concept

allows us to operate with chains that do not start in equilibrium but with asymptotic properties, including variance, informed by the stationary case.

Stationarity: A transition kernel K is stationary—or invariant—for target probability distribution P when it satisfies the condition:

$$\int_{\mathcal{X}} K(dx'|x)P(dx) = P(dx'). \tag{2.15}$$

This can also be expressed through expectations, where for any bounded and continuous test function h, the equality

$$\int_{\mathcal{X}}\int_{\mathcal{X}} h(x')K(dx'|x)p(x)dx = \int_{\mathcal{X}} h(x)p(x)dx \tag{2.16}$$

holds. Meaning that if a random variable X_0 is distributed according to p, subsequent variables $X_1, X_2, \ldots, X_N$ derived from the chain will follow the distribution p. Stationarity ensures that the distribution of states within the Markov chain converges over time, preserving the stationary state's distribution once reached.

Reversibility: The reversibility condition implies stationarity. A transition kernel K is reversible relative to P when it satisfies the *detailed balance condition*,

$$K(dx'|x)P(dx) = K(dx|x')P(dx'), \tag{2.17}$$

indicating that the transition from state x to state x' under p is equally likely to the transition from x' back to x. This condition not only ensures the stationarity of π but also that the chain can proceed forward and backward in time without altering the overall statistical properties of the sequence. This symmetry simplifies many computational and theoretical aspects of working with Markov chains, especially in Monte Carlo methods, and maintaining the detailed balance condition is a common method of ensuring the stationarity of the target density.

Reversibility also holds up to a one-to-one transformation $T : \mathcal{X} \to \mathcal{X}$ that leaves the target distribution invariant,

$$\int_{\mathcal{X}} p(x)dx = \int_{\mathcal{X}} p(T^{-1}(x))dx, \tag{2.18}$$

where $p(T^{-1}(x))$ is the density of $x' = T(x)$, then the modified detailed balance condition,

$$K(dx'|x)P(dx) = K(T^{-1}(dx)|T^{-1}(x'))P(dx'), \tag{2.19}$$

ensures reversibility. This form of reversibility becomes relevant when using Langevin dynamics in the Markov transition.

Aperiodic Irreducibility: For any natural number n, the nth step transition probability from a state x to a state x' is given by the iterative relation

$$K^n(dx'|x) = \int_{\mathcal{X}} K^{n-1}(dx'|y)K(dy|x), \tag{2.20}$$

where $K^1(dx'|x) := K(dx'|x)$. A transition kernel K is considered aperiodically irreducible with respect to a distribution P if, for any measurable set A with $P(A) > 0$ and all initial conditions x such that $P(X = x) > 0$, there exists some $n_0 > 0$ such that for any $n \geq n_0$,

$$K^n(A|x) > 0. \tag{2.21}$$

This means that reaching any set A within a finite number of steps with a positive probability is possible from nearly any starting point in a nonperiodic manner. This formalizes the Markov chain's ability to access all relevant parts of the state space over time, ensuring a complete exploration of the state space.

Stationarity and aperiodic irreducibility ensure that a Markov chain exhibits *pathwise ergodic* behaviour, a fundamental aspect that enables long-term predictions and analysis. Formally, for a Markov chain with transition kernel K in a state space $\mathcal{X}$ with a stationary probability measure P, and assuming the chain is aperiodically irreducible, it is true that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} h(x_i) = \int h(x)p(x)dx = \mathbb{E}_p[h(X)]. \tag{2.22}$$

This implies that for any bounded measurable function h and almost all initial conditions x_0, the time average of h over the chain converges to the expected

value of h under the distribution P (Theorem 17.1.7 in Meyn and Tweedie, 2012). Ergodicity is a crucial property as it guarantees that, irrespective of the chain's initial distribution, the time averages of functions computed along the chain will converge to their expected values under the stationary distribution P, allowing for statistical inference based on its long-run averages.

2.2.1 The Gibbs Sampler

Monte Carlo methods encounter practical challenges as the dimensionality of the model increases since they rely on identifying a suitable proposal distribution. The Gibbs sampler offers an alternative through sequential sampling of low-dimensional subsets of the model's parameters. This method constructs a Markov chain by updating components of x conditionally on the others, reducing the problem to a series of lower-dimensional sampling steps, which are often more manageable.

The Gibbs sampler was introduced by S. Geman and D. Geman (1984) for image restoration, and Gelfand and A. F. Smith (1990) broadened its application beyond image analysis, making it a fundamental tool in statistical modelling. Start by partitioning the parameter variable X into r distinct blocks, such that $X = (X_1, \ldots, X_r)$. Each block X_j is updated by drawing from its *full conditional distribution*,

$$P_j(X_j = x_j | X_{-j} = x_{-j}) = \frac{P(\{X_j = x_j\} \cap \{X_{-j} = x_{-j}\})}{P(X_j = x_j)}, \quad j = 1, \ldots, r, \tag{2.23}$$

while holding the other blocks constant, where X_{-j} are all the joined partitions except j. The iterative process of updating all parameter blocks, one at a time, forms one complete iteration of the Gibbs sampler, detailed in Algorithm 2.

Gibbs sampling maintains the desired stationary distribution through the detailed balance condition: consider the transition probabilities in Gibbs sampling when updating the j partition,

$$K_j(dx'|x) = P_j(dx'_j|x_{-j}), \tag{2.24}$$

Algorithm 2 Gibbs Sampler

Require: $(x_j, P_j), \quad j = 1, \ldots, r$

1: Generate $x'_1 \sim P_1(\cdot|x_2, \ldots, x_r)$

2: Generate $x'_2 \sim P_2(\cdot|x'_1, x_3, \ldots, x_r)$

3: $\quad \vdots$

4: Generate $x'_r \sim P_r(\cdot|x'_1, \ldots, x'_{r-1})$

5: Return $(x'_1, \ldots, x'_r)$

where x_{-j} represents the current state of all partitions except j (regardless if they have been already updated or not). Then we have that,

$$\begin{aligned} K_j(dx'|x)P(dx) &= P_j(dx'_j|x_{-j})P_j(dx_j|x_{-j})P(dx_{-j}) \\ &= P_j(dx_j|x'_{-j})P_j(dx'_j|x'_{-j})P(dx'_{-j}) = K_j(dx|x')P(dx'), \end{aligned} \tag{2.25}$$

given that $x'_{-j} = x_{-j}$ since all partitions other than j remain unchanged in the transition. Since the detailed balance condition holds for each individual update, it holds for the sequence of updates, preserving the stationary distribution $P(dx)$. Verifying a convergence result for the Gibbs sampler requires assumptions on the continuity, boundedness and support of the target density (Theorem 2 in G. Roberts and A. Smith, 1994).

2.2.2 The Metropolis-Hastings Algorithm

Introduced for simulating the thermodynamic properties of particles by generating random walk proposals on the state space (Metropolis et al., 1953), it was extended to accommodate asymmetric proposals by Hastings (1970a), becoming a pivotal tool from statistical physics to Bayesian statistics. The Metropolis-Hastings algorithm uses a generally applicable acceptance mechanism to ensure the detailed balance of the Markov transition kernel with respect to the target density, enabling sampling without knowledge of the target normalization constant.

The algorithm involves a two-step cycle: proposing a move to a new state x' from the current state x based on a proposal transition kernel $T(dx'|x)$, followed

by an acceptance-rejection criterion that ensures the invariance of $P(dx)$ within the resulting Markov chain. This only ensures that $P(dx)$ acts as a stationary distribution for the Markov chain and generally does not guarantee its irreducibility.

Since the detailed balance condition is not satisfied automatically by the proposal kernel T, the accept-reject step is a correction to this discrepancy. For it to be applicable, the proposal must have a degree of reversibility; that is, for any states x and x', the measures $T(dx'|x)P(dx)$ and $T(dx|x')P(dx')$ must be mutually absolutely continuous. This requirement guarantees the positivity and finiteness of the Metropolis-Hastings ratio,

$$r(x, x') = \frac{T(dx'|x)P(dx')}{T(dx|x')P(dx)} \tag{2.26}$$

which is the probability of the proposed move being accepted. Notice that the distribution P has to be known only up to a multiplicative constant since constants cancel out when evaluating the ratio. The Metropolis-Hastings algorithm is detailed in Algorithm 3.

Algorithm 3 Metropolis-Hastings algorithm

Require: x, $T(dx'|x)$

1: Generate $x' \sim T(\cdot|x)$
2: Generate $u \sim \text{Uniform}(0, 1)$
3: **if** $u \leq r(x, x')$ **then**
4: Return x'
5: **else**
6: Return x
7: **end if**

In direct probabilistic methods like the Accept-Reject algorithm, rejected moves result in discarding the proposed state, and the process is repeated until acceptance. Conversely, the Metropolis-Hastings algorithm retains the current state x as the new sample when the proposed move is rejected. Therefore, frequently rejected configurations are counted multiple times in calculating averages, increasing the

correlation among sequential samples and increasing the estimator's variance if a well-tuned proposal is not used.

The probability transition kernel of the Metropolis-Hastings chain is

$$K(dx'|x) = \min(1, r(x,x'))T(dx'|x) + (1-\alpha(x))\delta_x(dx') \tag{2.27}$$
$$\alpha(x) = \int \min(1, r(x,y))T(\mathrm{d}y|x),$$

where $\alpha(x)$ represents the probability of accepting any move with initial condition x, encoding all possible rejected steps.

Using the identity $r(x, x_0) = 1/r(x_0, x)$ alongside the algebraic equality $\min(1, r) = r\min(1, 1/r)$, since $r > 0$, we have that

$$\begin{aligned}\min(1, r(x,x'))T(dx'|x)P(dx) &= \min(1, r(x',x))r(x,x')T(dx'|x)P(dx) \\ &= \min(1, r(x',x))T(dx|x')P(dx'); \end{aligned} \tag{2.28}$$

furthermore, given test function φ, and integrating over the parameter space with respect to the rejection component,

$$\int_{\mathcal{X}}\int_{\mathcal{X}} \varphi(x,x')(1-\alpha(x))\delta_x(dx')P(dx) = \int_{\mathcal{X}}\int_{\mathcal{X}} \varphi(x,x')(1-\alpha(x'))\delta_{x'}(dx)P(dx'), \tag{2.29}$$

which implies that $(1-\alpha(x))\delta_x(dx')P(dx) = (1-\alpha(x'))\delta_{x'}(dx)P(dx')$, essentially verifying the detailed balance condition on the transition kernel (2.27) using as acceptance probability $R(r(x,x')) = \min(1, r(x,x'))$. Other functions satisfying the property $R(r) = rR(1/r)$ could induce a dynamics reversible with respect to P; nevertheless, the Metropolis-Hastings rate is optimal in terms of asymptotic variance (Theorem 2.2.1 in Peskun, 1973).

If the Metropolis-Hastings ratio $r(x,x') > 0$ for all $x, x' \in \mathcal{X}$, then the aperiodic irreducibility of the proposal transition T ensures the aperiodic irreducibility of the Markov chain generated by the algorithm. Hence, the pathwise ergodicity of the algorithm is influenced by the choice of the proposal kernel T. Beyond its theoretical implications for convergence, selecting a suitable proposal kernel is paramount for

an efficient algorithm. An optimal acceptance/rejection rate ensures a compromise between large moves that decorrelate the chain but are more likely to be rejected and small moves that are likely to be accepted. The forthcoming will illustrate some options for the transition kernel and its implications for algorithmic efficiency.

Independent Metropolis-Hastings (Alg. 4): The independent proposal is the Markov chain counterpart of the Accept-Reject method, where the proposal distribution q, independent of the initial state x, serves as the transition distribution. Despite the independence in generating proposals, the resultant sample does not exhibit independence since the acceptance probability for a proposed move depends on the current state, diverging from the independent and identically distributed scenario unless $p = q$.

Algorithm 4 Independent Metropolis-Hastings algorithm

Require: x

1: Generate $x' \sim q$.
2: Generate $u \sim \text{Uniform}(0, 1)$
3: **if** $u \leq \frac{p(x')q(x)}{p(x)q(x')}$ **then**
4: Return x'
5: **else**
6: Return x
7: **end if**

The convergence of the chain generated by this algorithm is contingent upon the properties of q. Specifically, the chain is irreducible and aperiodic-—thus ergodic-—if q is almost everywhere positive on the support of p. The stronger result of uniform ergodicity is ensured when a constant M exists such that $f(x) < Mg(x)$ for all $x \in \mathcal{X}$, further illustrating its relationship to the Accept-Reject algorithm (Theorem 7.8 in Robert and Casella, 2004). However, the independent Metropolis-Hastings accepts more proposed values than the Accept-Reject on average, improving its effectiveness in comparison (Lemma 7.9 in Robert and Casella, 2004).

Random Walk Metropolis-Hastings (Alg. 5): The random walk proposal

uses local exploration around the current state of the Markov chain by generating a random perturbation centred on it. The proposal kernel $T(dx'|x)$ is defined by the transformation

$$X' = X + \epsilon, \tag{2.30}$$

where ϵ is a random perturbation with distribution g, independent of X.

Common choices for g include uniform distributions over spheres centred at the origin, standard distributions like the Gaussian distribution, and heavy-tailed variants like the Student–t and χ^2 distributions. A symmetric g, satisfying $g(-x) = g(x)$, aligns with the original formulation by Metropolis et al. (1953), leading to the reversible implementation detailed in Algorithm 5.

Algorithm 5 Random Walk Metropolis-Hastings algorithm

Require: x

Generate $\epsilon \sim g$.
Let $x' \leftarrow x + \epsilon$
Generate $u \sim \text{Uniform}(0, 1)$
if $u \leq p(x')/p(x)$ **then**
    Return x'
else
    Return x
end if

The algorithm's convergence properties and optimal scaling are essential considerations for efficiently exploring the target distribution. A seminal result in this domain is the work by Gelman, Gilks, and G. O. Roberts (1997), who established that the optimal acceptance rate for high-dimensional target distributions for the Random Walk Metropolis-Hastings algorithm is approximately 23.4%. This result implies that the algorithm's proposal distribution should be tuned to accept about one in four proposed moves, for instance, by setting the variance when g is Gaussian. The underlying intuition is that larger moves lead to faster state space

exploration but with higher rejection rates, which can inefficiently recycle the current state and lead to high autocorrelation, while smaller moves have higher acceptance rates but can result in slow exploration due to the small distance covered in each step, also leading to high autocorrelation.

Metropolis Adjusted Langevin Dynamics (Alg. 6): The Metropolis-Adjusted Langevin Algorithm (MALA) is a non-symmetric transition kernel derived from the discretization of continuous stochastic dynamics, using gradient information from the initial position to transition to high-density areas in the distribution, and maintaining detailed balance with the Metropolis-Hastings correction. Integrating continuous dynamics allows MALA to offer a proposal mechanism that better explores the target space.

The proposal mechanism is the discretization of a Langevin diffusion, a continuous-time process given by the stochastic differential equation,

$$dX_t = dB_t + \frac{1}{2}\nabla \log p(X_t)dt, \tag{2.31}$$

where B_t is the standard Brownian motion. This process is discretized for its use in MALA, transforming the continuous dynamics into a random walk-like transition,

$$X' = X + \frac{\tau^2}{2}\nabla \log p(X) + \tau\epsilon, \tag{2.32}$$

where $\epsilon \sim N(0, \mathbb{I}_d)$ and $\tau > 0$ controls the scale of the discretization. The discretization introduces a bias, corrected using Metropolis-Hastings, ensuring convergence to the target distribution. This bias also makes the transition nonreversible, with density proportional to

$$\exp\left(-\frac{\left\|X' - X - \frac{\tau^2}{2}\nabla \log p(X)\right\|^2}{2\tau^2}\right), \tag{2.33}$$

making the Metropolis-Hastings ratio,

$$r_{MALA}(x, x') = \frac{p(x')\exp\left(-\left\|x' - x - \frac{\tau^2}{2}\nabla \log p(x)\right\|^2 / 2\tau^2\right)}{p(x)\exp\left(-\left\|x - x' - \frac{\tau^2}{2}\nabla \log p(x')\right\|^2 / 2\tau^2\right)}. \tag{2.34}$$

Algorithm 6 Metropolis-Adjusted Langevin Algorithm (MALA)

Require: x, τ, p

1: Generate $\epsilon \sim \mathcal{N}(0, \mathbb{I}_d)$.
2: Let $x' \leftarrow x + \frac{\tau^2}{2}\nabla \log p(x) + \tau\epsilon$
3: Generate $u \sim \text{Uniform}(0, 1)$
4: **if** $u \leq r_{MALA}(x, x')$ **then**
5: Return x'
6: **else**
7: Return x
8: **end if**

The study of optimal scaling for MALA by G. O. Roberts and Rosenthal (1998) analyses how the discretization step size τ influences the algorithm's convergence properties. They prove an optimal acceptance rate of 57.4% for MALA, targeting the most efficient trade-off between acceptance frequency and the thoroughness of state space exploration. Their findings highlight that, under optimal scaling, MALA can significantly outperform traditional random walk Metropolis-Hastings approaches by including local gradient information of the target in the transition, achieving a more effective exploration of complex probability landscapes.

2.2.3 The Slice Sampler

Based on the alternative representation of the target density (2.3), we can generate a Markov chain whose stationary distribution aligns with the uniform distribution over $S(p) = \{(x, u) : 0 < u < p(x)\}$. Slice sampling employs a random walk within $S(p)$, proposing a method leading to a uniform stationary distribution over the set. A straightforward strategy alternates movements along the u-axis and x-axis with uniform steps in both directions, avoiding a Metropolis-Hastings correction to achieve uniformity over $S(p)$.

The slice sampling method, proposed by Radford M Neal (2003), requires two

uniform random walk steps over the subgraph of p, applicable in any Euclidean parameter space $\mathcal{X} = \mathbb{R}^d$, detailed in Algorithm 7. Step 2 of the algorithm is nontrivial, but there are options to adaptively propose samples that are rejected until a sample from the set is found. For instance, Radford M Neal (2003) proposes the shrinkage procedure, which draws an initial sample and iteratively shrinks the range from which subsequent samples are drawn if they fall outside the slice, guaranteeing new samples are less likely to be rejected after each rejection while still being reversible with respect to the uniform density.

Algorithm 7 Radford M Neal (2003) Slice Sampler

Require: x, u

1: Generate $u' \sim \text{Uniform}(0, p(x))$

2: Generate $x' \sim \text{Uniform}(\{x \ : \ u' \leq p(x)\})$

3: Return x', u'

The stationary distribution of the Markov chain generated by slice sampling is uniform over the subgraph $S(p)$, the area under the curve of the target density function p. An initial horizontal slice is generated when sampling u' uniformly from the interval $(0, p(x))$, setting the minimum height to evaluate x on its graph. The second slice samples a new state x' uniformly from the set of all values within the set $\{x \ : \ u' \leq p(x)\}$.

Uniform sampling within each slice ensures that the transition from x to x' does not favour any particular region of $S(p)$ more than another, preserving uniformity over $S(p)$ across iterations. This conclusion holds even if the target density p is known only up to a normalization constant because the relative proportions of the density function are preserved under scaling, and thus, the algorithm's ability to generate a uniform distribution over $S(p)$ remains unaffected.

Sampling uniformly from the subgraph $S(p)$ might be intractable, especially as the dimension d increases. This can be avoided using a generalization beyond the straightforward approach of Radford M Neal (2003) slice sampler, which operates under single-slice uniform sampling, by leveraging the concept of multiple slices to

facilitate the sampling process. The core idea is to decompose the target density into the product of positive functions,

$$p(x) = \prod_{i=1}^{k} p_i(x) \tag{2.35}$$

which may not necessarily be densities themselves. Such a decomposition is particularly useful in Bayesian settings, where $p_i(x)$ could represent individual likelihood or the prior. By introducing multiple auxiliary variables u_i, each corresponding to a component p_i, the algorithm constructs a multi-dimensional sampling framework detailed in Algorithm 8.

Algorithm 8 Generalized Slice Sampler

Require: x, u

1: **for** $i = 1, \ldots, k$ **do**
2: Generate $u_i' \sim \text{Uniform}(0, p_i(x))$
3: **end for**
4: Generate $x' \sim \text{Uniform}(\{x \ : \ u_i' \leq p_i(x),\ i = 1, \ldots, k\})$
5: Return x', $u_1', \ldots, u_k'$

The decomposition in (2.35) allows for the representation

$$p(x) = \int_0^{p_1(x)} du \times \cdots \times \int_0^{p_k(x)} du \tag{2.36}$$

embedding p within a higher-dimensional uniform joint distribution involving x and k auxiliary variables. The generalized slice sampler increases the number of steps needed for one iteration but offers a more flexible framework for constructing uniform proposals, up to one dimension at a time. This approach preserves the advantages of slice sampling, such as its independence from the normalization constant of p, and introduces greater flexibility in navigating the distribution's landscape.

Elliptical Slice Sampler (Alg. 9): Murray, R. Adams, and MacKay (2010) present an efficient method suitable for models where the posterior target density (1.9) has a multivariate Gaussian prior density $\mathcal{N}(0, \Sigma)$ and any likelihood function

$L(x) = \prod_i p(y_i|x)$, making the target distribution

$$p(x|\mathcal{D}) \propto L(x)\mathcal{N}(0, \Sigma). \tag{2.37}$$

The approach builds on a Metropolis-Hastings proposal introduced by R. Neal (1998), proposing new states x' as a combination of the current state x and an auxiliary random draw ν from the prior, established by a step size parameter $\epsilon \in [-1, 1]$,

$$x' = \sqrt{1 - \epsilon^2}x + \epsilon\nu, \quad \nu \sim \mathcal{N}(0, \Sigma). \tag{2.38}$$

The proposed state ranges from a sample of the prior $\epsilon = \pm 1$ to the current state $\epsilon = 0$. Since the proposal is symmetric with respect to the prior, the proposed move is accepted based on the likelihood ratio between x' and x. This method is useful for probabilistic models where posterior variables exhibit strong prior dependencies, like a Gaussian process prior.

The elliptical slice sampler modifies the latter approach by proposing states on an ellipse

$$x' = \nu \sin\theta + x \cos\theta, \quad \nu \sim \mathcal{N}(0, \Sigma), \tag{2.39}$$

where parameter $\theta \in [0, 2\pi]$ is generated using slice sampling. In other words, the parameter θ is chosen uniformly on subgraphs of the likelihood function of the model $S(L)$, generating a new sample that always leaves the prior invariant. This reparameterization allows an adaptive choice of the step size without preliminary tuning using a shrinkage procedure similar to that proposed by Radford M Neal (2003). This method is implemented in Algorithm 9.

Nonreversible Metropolis Updates (Alg. 10): Radford M Neal (2020) builds on the foundation laid by slice sampling, where s is sampled uniformly between 0 and $p(x)$ to form a joint distribution that is uniform over the region $0 < s < p(x)$, to make a (symmetric) nonreversible Metropolis accept/reject step. The innovation is in the retention of s between updates, using its current value to inform accept/reject decisions. Metropolis updates can be seen within this framework, with new samples being accepted or rejected depending on whether $p(x') > s$. This strategy diverges

Algorithm 9 Murray, R. Adams, and MacKay (2010) Elliptical Slice Sampler

Require: x, L, Σ

1: $\nu \sim \mathcal{N}(0, \Sigma)$
2: $u \sim \text{Uniform}(0, L(x))$
3: $\theta \sim \text{Uniform}(0, 2\pi)$
4: $[\theta_{min}, \theta_{max}] \leftarrow [\theta - 2\pi, \theta]$
5: $x' \leftarrow x \cos\theta + \nu \sin\theta$
6: **if** $u \leq L(x')$ **then**
7: Return x'
8: **else**
9: **if** $\theta < 0$ **then**
10: $\theta_{min} \leftarrow \theta$
11: **else**
12: $\theta_{max} \leftarrow \theta$
13: **end if**
14: $\theta \sim \text{Uniform}(\theta_{min}, \theta_{max})$
15: Go to 5.
16: **end if**

from traditional slice sampling by allowing for nonreversible updates to s, translating it by a fixed amount, potentially adding noise, and reflecting off boundaries defined by the density of x.

The method involves a variable v such that $s = |v|p(x)$, uniformly distributed over $[-1, +1]$. Updates to v consist of adding a fixed value $\delta \in \mathbb{R}$ and some optional random noise and reflecting off the boundaries $[-1, +1]$, ensuring an invariant uniform distribution over $[-1, +1]$. Furthermore, acceptance of a proposed move from x to x' requires adjusting s (and hence v) to keep the auxiliary variable unchanged, preserving the reversibility of the move with respect to the augmented state space. The generic nonuniform Metropolis update is presented in Algorithm 10, where the proposal kernel T needs to be symmetric.

This non-reversible updating mechanism encourages $|v|$ to oscillate slowly between values near 0 (where acceptance rates are high) and near 1 (where acceptance is less likely), potentially clustering acceptances and rejections and enhancing the chain's exploration. While the average acceptance rate remains unchanged, as v maintains a uniform distribution regardless of x, clustering acceptances and rejections can lead to more efficient sampling over the state space. Particularly when some variables are updated by other methods, such as Gibbs sampling on discrete variables of the parameter space, where clustering acceptances or rejections can help the exploration in sync with other changing parameters of the model.

2.2.4 Diagnostic Tools

Diagnosing the quality of the samples in a Markov chain first requires ensuring the chain has converged to its stationary distribution and then estimating the variance of its estimator. Because the chain's initial condition is not the target density, it might take some time to converge to its required density, and once it does, the covariance between samples needs to be considered in the estimator's variance.

Assume that the chain has converged, then the covariance between samples of

Algorithm 10 Radford M Neal (2020) Nonreversible Metropolis Algorithm

Require: $x, v, \delta, \sigma, T(dx'|x)$

$x' \sim T(\cdot|x)$

$\epsilon \sim \mathcal{N}(0, \sigma)$

$v \leftarrow v + \delta + \epsilon$

while $v > +1$ **do**

    $v \leftarrow v - 2$

end while

while $v < -1$ **do**

    $v \leftarrow v + 2$

end while

if $|v| \leq p(x')/p(x)$ **then**

    $v' \leftarrow vp(x)/p(x')$

    Return x', v'

else

    Return x, v

end if

the chain is

$$C(k) = \text{Cov}[X_i, X_{i+k}], \tag{2.40}$$

and thus its overall variance is given by (2.14), and can be rewritten in terms of (2.40),

$$\sigma^2 = C(0) + 2\sum_{k=1}^{\infty} C(k). \tag{2.41}$$

In time series analysis, $C(k)$ is called the *autocovariance function* of the chain and $C(k)/C(0)$ its *autocorrelation function*, making σ^2 the integrated autocovariance function and

$$\tau = 1 + 2\sum_{k=1}^{\infty} \frac{C(k)}{C(0)}, \tag{2.42}$$

the integrated autocorrelation function. Let $\bar{x}$ be the Monte Carlo estimate (2.5) of the chain's average, we could do a sample estimate of the autocovariance function,

$$\hat{C}(k) = \sum_{0<i<N-k} (x_i - \bar{x})(x_{i+k} - \bar{x}), \tag{2.43}$$

and plug it into (2.41) for a sample estimate of the chain's variance. However, the variance of the estimate $\hat{C}(k)$ will cause the sum in (2.41) to explode as k increases, increasing the noise in the estimate the larger the chain. Instead, Geyer (1992) proposes the alternative function

$$c(k) = C(2k) + C(2k+1), \tag{2.44}$$

which is strictly positive, strictly decreasing, and strictly convex, and the alternative approximation for the integrated autocorrelation function,

$$\hat{\tau} = -1 + 2\sum_{k=0}^{N} \hat{c}(k) \tag{2.45}$$

$$\hat{c}(k) = \hat{C}(2K) + \hat{C}(2k+1). \tag{2.46}$$

When comparing experiments in the following chapters, we usually run multiple chains c on multi-dimensional j configuration spaces,

$$\tau_{c,j} = -1 + 2\sum_{k=0}^{N} \hat{c}_{c,j}(k) \tag{2.47}$$

$$\hat{c}_{c,j}(k) = \hat{C}(2K) + \hat{C}(2k+1) \tag{2.48}$$

$$\hat{C}_{c,j}(k) = \sum_{0<i<N-k} (x_{c,i,j} - \bar{x}_{c,\cdot,j})(x_{c,i+t,j} - \bar{x}_{c,\cdot,j}) \tag{2.49}$$

$$\bar{x}_{c,\cdot,j} = \frac{1}{N}\sum_{i=1}^{N} x_{c,i,j}, \tag{2.50}$$

where the autocovariance is computed using the Fourier transform method from Wolff, Collaboration, et al. (2004). We present the autocorrelations and effective sample sizes for the median of all chains and the worst-case dimension,

$$\tau_{\max} = \max_{j=1,\ldots,d} \left[\underset{c=1,\ldots,C}{\text{median}}\ \tau_{c,j} \right] \tag{2.51}$$

$$\text{ESS} = \min_{j=1,\ldots,d} \left[\underset{c=1,\ldots,C}{\text{median}}\ \frac{NC}{\tau_{c,j}} \right]. \tag{2.52}$$

Diagnosing the convergence of a Markov chain to its stationary distribution is crucial to ensure the accuracy of posterior estimates. Measuring the divergence of the samples from the target verifies that the chain has sufficiently explored the target space. The divergences used in this work are the Kernelized Stein Discrepancy (KSD; Liu, Lee, and M. Jordan, 2016), useful when comparing samples to an unnormalized target density, and the Maximum Mean Discrepancy (MMD; Gretton et al., 2012), useful when comparing two sets of samples.

The Kernelized Stein discrepancy is computable even if the target density p is unnormalized, providing a convenient way to assess samples' compatibility directly with the model. Its U- and V-statistics are calculated using the inverse multiquadric

kernel $k(x, x') = (1 + (x - x')^T(x - x'))^{-1/2}$ as

$$\text{U-stat} = \frac{1}{CN(CN-1)} \sum_{c,i} \sum_{c' \neq c, i' \neq i} \mathcal{A}_\pi \mathcal{A}'_\pi K(\mathbf{x}_{c,i}, \mathbf{x}_{c',i'}) \tag{2.53}$$

$$\text{V-stat} = \frac{1}{C^2N^2} \sum_{c,i} \sum_{c',i'} \mathcal{A}_\pi \mathcal{A}'_\pi K(\mathbf{x}_{c,i}, \mathbf{x}_{c',i'}) \tag{2.54}$$

$$\begin{aligned} \mathcal{A}_p \mathcal{A}'_p K(x, x') &= \nabla_x \cdot \nabla_{x'} k(x, x') + \nabla_x k(x, x') \cdot \nabla_{x'} \log p(x') \\ &\quad + \nabla_{x'} k(x, x') \cdot \nabla_x \log p(x) + k(x, x') \nabla_x \log p(x) \cdot \nabla_{x'} \log p(x). \end{aligned} \tag{2.55}$$

It can be shown that the U-statistic is an unbiased estimate of $\mathbb{E}_{x,x' \sim p'}[\mathcal{A}_p \mathcal{A}'_p K(x, x')]$ for process p' generating the samples, while the V-statistic is biased but always non-negative (Liu, Lee, and M. Jordan, 2016). If $p = p'$ then $\mathbb{E}_{x,x' \sim p'}[\mathcal{A}_p \mathcal{A}'_p K(x, x')] = 0$ by Stein's identity (Stein et al., 2004).

The Maximum Mean Discrepancy represents distances between distributions as distances between mean embeddings of features. It is defined by a feature map $\varphi : \mathbb{R}^d \to \mathcal{F}$ and measures the discrepancy between two distributions P and Q,

$$\text{MMD}(P, Q) = \|\mathbb{E}_P[\varphi(X)] - \mathbb{E}_Q[\varphi(X)]\|_{\mathcal{F}}, \tag{2.56}$$

where the strength of the metric depends on the expressiveness of the feature map, which corresponds to a kernel by $\langle \varphi(x), \varphi(y) \rangle_{\mathcal{F}} = k(x, y)$. Using the Gaussian kernel $k(x, y) = \exp(-\|x - y\|^2/2)$ we empirically compute the MMD between two chains $X_1, \ldots, X_N$ and $Y_1, \ldots, Y_N$,

$$\text{MMD}^2(X, Y) = \frac{1}{N(N-1)} \sum_{i \neq j} k(x_i, x_j) - \frac{2}{N^2} \sum_{i,j} k(x_i, y_j) + \frac{1}{N(N-1)} \sum_{i \neq j} k(y_i, y_j). \tag{2.57}$$

Chapter 3

Approximate Inference Methods

In Bayesian computation, sampling methods have long stood as the cornerstone, offering a robust framework with a precise understanding of our estimate's variance backed by elementary statistical results. However, as the complexity of models and the size of data sets grow, the practical limitations of traditional sampling techniques, especially in terms of computational efficiency, become increasingly apparent. Despite their theoretical appeal, these methods often fall short in scenarios demanding rapid decision-making or when handling models of considerable complexity.

Advances in the literature ushered in the alternative strategy of target density approximation (Saul, Jaakkola, and M. I. Jordan, 1996; M. I. Jordan et al., 1999; Beal, 2003). These methods do not approximate the integral directly; instead, they focus on approximating the target posterior density $p(x|\mathcal{D})$ with a tractable density $q(x)$, minimizing a notion of divergence between the two. Although density approximation may lack the asymptotic guarantees of sampling methods, its ability to provide rapid inference solutions and the pragmatic acknowledgement of the approximate nature of real-world inference tasks explain its rapid increase in popularity.

In essence, the method presented in this chapter transforms the problem of Bayesian computation from integration to an optimization task. Specifically, the task of searching for an optimal density q within a variational family of densities $\mathcal{Q}$ that yields the most accurate approximation of the posterior. This requires minimizing

a divergence measure D between the approximate distribution $q(x)$ and the exact posterior $p(x|\mathcal{D})$, formally expressed as

$$q^*(x) = \arg\min_{q \in \mathcal{Q}} D[q(x)||p(x|\mathcal{D})], \tag{3.1}$$

where $q^*(x)$ denotes the distribution that best approximates the exact posterior within the constraints of the chosen divergence measure and variational family. Resulting in the approximation of (1.11),

$$\mathbb{E}[h(X)|\mathcal{D}] \approx \int_{\mathcal{X}} h(x) q^*(x) dx, \tag{3.2}$$

where the right-hand side can be solved analytically if simple enough or approximated using Monte Carlo integration, assuming that q^* is easy to sample from. This methodical shift streamlines the inference process and opens up new possibilities for efficiently handling complex models, such as using neural networks to construct $\mathcal{Q}$. The first section of this chapter focuses on the divergence measure, while the second section focuses on the variational family.

3.1 Divergence Measure

Consider the posterior density (1.9) and its intractable model evidence

$$Z = \int_{\mathcal{X}} \prod_{i=1}^{n} p(y_i|x) p(x) dx = \int_{\mathcal{X}} p(\mathcal{D}, x) dx. \tag{3.3}$$

An approach that avoids the direct computation of the posterior density can maximize the likelihood of observations (or model evidence) by maximizing its lower bound dependent on a parametrised surrogate density q. Using Jensen's inequality, we can get a lower bound on the logarithm of the model evidence, varying depending on the approximate density q, called the Evidence Lower Bound (ELBO),

$$\begin{aligned} \log Z &= \log \int_{\mathcal{X}} p(\mathcal{D}, x) \frac{q(x)}{q(x)} dx \\ &\geq \int_{\mathcal{X}} \log \frac{p(\mathcal{D}, x)}{q(x)} q(x) dx = \mathbb{E}_q \left[\log \frac{p(\mathcal{D}, x)}{q(x)} \right]. \end{aligned} \tag{3.4}$$

The ELBO serves as a surrogate objective for $\log Z$, with the optimization of $q(x)$ effectively providing a tighter approximation to the model evidence. Equivalently, finding the density $q \in \mathcal{Q}$ minimizing the negative ELBO in (3.1) effectively finds the density that closest approximates the target posterior.

The ELBO is instrumental within the Bayesian framework as it provides a divergence from the true posterior; however, the concept of divergence transcends this specific application and requires a more general definition. Defining a general divergence thus connects the specific goals of Bayesian approximation with the broader landscape of optimization problems.

3.1.1 Kullback-Leibler Divergence

Let $\mathcal{P}$ be the space of probability density functions such that any probability distribution P which makes a probability space with the σ-algebra of the configuration space $\mathcal{X}$ has its density function $p \in \mathcal{P}$. Then, $D : \mathcal{P} \times \mathcal{P} \to [0, \infty)$ is a *divergence* on $\mathcal{P}$ when it is true that $D(p, q) \geq 0$ for any $p, q \in \mathcal{P}$ and $D(p, q) = 0$ if and only if $p = q$. A divergence establishes a statistical distance between probability distributions defined on the space $\mathcal{X}$, serving as a metric for quantifying the "closeness" between a target distribution and its approximation.

The Kullback-Leibler Divergence (KLD; Kullback and Leibler, 1951) is a divergence between two probability distributions, quantifying the amount of natural information units lost when $q(x)$ is used to approximate $p(x|\mathcal{D})$. Formally, for distributions p and q in the space $\mathcal{P}$,

$$\mathrm{KLD}[q(x)||p(x|\mathcal{D})] = \int_{\mathcal{X}} \log \frac{q(x)}{p(x|\mathcal{D})} q(x) dx. \tag{3.5}$$

The nonnegativity of the KLD is given by the concavity of the logarithm and Jensen's inequality,

$$\int_{\mathcal{X}} \log \frac{p(x|\mathcal{D})}{q(x)} q(x) dx \leq \log \int_{\mathcal{X}} \frac{q(x)}{q(x)} p(x|\mathcal{D}) dx = \log 1 = 0, \tag{3.6}$$

and negating both sides,

$$0 \leq \int_{\mathcal{X}} \log \frac{q(x)}{p(x|\mathcal{D})} q(x)dx = \mathrm{KLD}[q(x)||p(x|\mathcal{D})], \tag{3.7}$$

with equality if and only if $p(x|\mathcal{D}) = q(x)$ for all $x \in \mathcal{X}$ with positive probability.

Direct minimization of the KLD requires an evaluation of the target distribution itself. In the case of Bayesian inference, it requires the computation of the posterior distribution (1.9),

$$p(x|\mathcal{D}) = Z^{-1} \prod_{i=1}^{n} p(y_i|x)p(x) = Z^{-1} p(\mathcal{D}, x) \tag{3.8}$$

which depends on the model evidence Z, an intractable integral. However, notice that using the nonnegativity property of KLD (3.7) we can derive the ELBO (3.4) using the definition of the Bayesian posterior density,

$$\begin{aligned} 0 \leq \mathrm{KLD}[q(x)||p(x|\mathcal{D})] &= -\int_{\mathcal{X}} \log \frac{p(\mathcal{D}, x)}{q(x)} q(x)dx + \log Z \\ &= \mathrm{KLD}[q(x)||p(\mathcal{D}, x)] + \log Z. \end{aligned} \tag{3.9}$$

Variational inference (VI) reformulates the divergence to avoid the direct computation of intractable quantities. Considering that the model evidence Z is independent of the configuration x, then reusing the previous logic of the ELBO, we instead minimize the variational free energy when doing VI,

$$\mathrm{KLD}[q(x)||p(\mathcal{D}, x)] = \mathrm{KLD}[q(x)||p(x|\mathcal{D})] - \log Z \tag{3.10}$$

which is the negative of the ELBO.

An important property of the KLD is its asymmetry $\mathrm{KLD}[q(x)||p(x|\mathcal{D})] \neq \mathrm{KLD}[p(x|\mathcal{D})||q(x)]$, which plays a crucial role in the outcome of the optimisation (3.1). $\mathrm{KLD}[q(x)||p(x|\mathcal{D})]$, the *exclusive* or *reverse* KLD, is used as a black-box variational inference objective since it can be easily approximated using Monte Carlo integration. The *inclusive* or *forward* KLD requires solving an integral of the form (1.11) and is used in specific problems where good approximations can be found for the integral,

$$\mathrm{KLD}[p(x|\mathcal{D})||q(x)] = \int_{\mathcal{X}} \log \frac{p(x|\mathcal{D})}{q(x)} p(x|\mathcal{D})dx. \tag{3.11}$$

Minimizing the reverse KLD$[q(x)||p(x|\mathcal{D})]$ avoids assigning probability mass in q to regions where p is negligible or zero, "mode-seeking" while avoiding the tails of p when q is constrained to be unimodal. The penalty for q assigning mass to regions where p is zero is infinite but finite when assigning zero mass to q where p is positive. The penalty is apparent by the fraction inside the logarithm in (3.5), which explodes to infinity when $p(x|\mathcal{D})$ is null and $q(x)$ is not. The one-dimensional, unimodal example, with a skewed target to approximate within a Gaussian family, is illustrated on the left side of Figure 3.1. Notice how the approximation misses the skewed left tail of the target and centres around the mean of the target.

Minimizing the forward KLD$[p(x|\mathcal{D})||q(x)]$ acts oppositely, forcing the support of q to cover the support of p completely, "mass-covering" the tails of p. The contrary penalty is clear from the fraction inside the logarithm in (3.11), which explodes to infinity when $q(x)$ is null and $p(x|\mathcal{D})$ is not. The one-dimensional, unimodal example is illustrated on the right side of Figure 3.1, where the approximation covers the whole mass of the target.

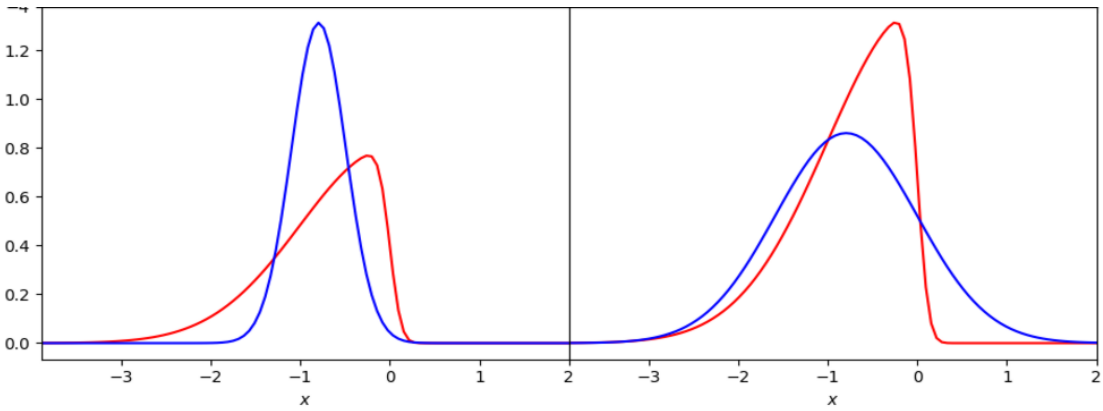


Figure 3.1: Gaussian approximation that underestimates (left) and overestimates (right) the real variance of the skewed target.

Minimizing the forward KLD results in an approximation where $p(x|\mathcal{D}) \leq Mq(x)$ for all x with positive target density, mirroring the requirements for proposal densities in the Accept-Reject, importance sampling, and independent Metropolis-Hastings methods of Chapter 2. This alignment emphasizes the forward KLD's utility in

creating well-suited approximations for a wide range of sampling-based inference tasks, where ensuring coverage over the entire support of p is crucial for effective approximation.

However, while the approximation resulting from minimizing the forward KLD is appealing, practical challenges limit its direct application. The computation of the forward KLD is nontrivial since it requires solving analytically an integral over $p(x|\mathcal{D})$ or sampling it for a Monte Carlo approximation, which is the essence of the Bayesian computational problem. The next two chapters propose black-box algorithms for Bayesian inference using the forward KLD in a sampling scheme.

3.1.2 Monte Carlo Variational Inference

Focus on the reverse KLD as an optimization objective, specifically on the variational free energy objective (3.10), which avoids computing the intractable model evidence,

$$\mathrm{KLD}[q(x)||p(\mathcal{D}, x)] = \mathbb{E}_q[\log q(X)] - \mathbb{E}_q[\log p(\mathcal{D}, X)]. \tag{3.12}$$

Limiting its use to situations where we can solve each expectation analytically is too restrictive for the expressiveness of the approximation q. Instead, we use the same ideas of Monte Carlo integration discussed in Chapter 2 to approximate the optimization objective. However, to use a first-order optimization method in (3.1), we need access to the gradient of (3.12) with respect to the hyperparameters that specify q in $\mathcal{Q}$, a nontrivial problem when these parameters define the probability space we are evaluating expectations over.

Knowing the importance of independent samples on the estimate's variance, the crucial constraint on the approximation distribution is that it is easy to sample from. This might still seem restrictive to well-known families of univariate distributions where we can apply the inverse transform to its generalized inverse function. However, a generally applicable trick will open the door to any bijective transformation of a vector of independent random variables and to neural networks building flexible variational families.

Assuming that X is a continuous random variable with density q_ψ, where the range of possible hyperparameters $\psi \in \Psi$ define the variational family $\mathcal{Q}$, the optimization problem becomes,

$$\psi^* = \arg\min_{\psi \in \Psi} \ \mathrm{KLD}[q_\psi(x)||p(x|\mathcal{D})], \tag{3.13}$$

where $q^* = q_{\psi^*}$. Make sampling $X \sim q_\psi$ equivalent to first sampling $Z \sim q_0$, a vector of independent configurations with no hyperparameters, and then computing $X = T_\psi(Z)$ for some parametrised diffeomorphic map T_ψ. Diffeomorphism ensures the map is differentiable and bijective; then, using the change of variables formula, we have the variational objective (3.12),

$$\mathrm{KLD}[q_\psi(x)||p(\mathcal{D}, x)] = \mathbb{E}_{q_0}[\log q_\psi(T_\psi(Z))] - \mathbb{E}_{q_0}[\log p(\mathcal{D}, T_\psi(Z))]. \tag{3.14}$$

Since q_0 is independent of the hyperparameters ψ, the gradient is evaluated only in the transformation inside the expectation and not in the probability space defining it,

$$\nabla_\psi \,\mathrm{KLD}[q_\psi(x)||p(\mathcal{D}, x)] = \mathbb{E}_{q_0}[\nabla_\psi \log q_\psi(T_\psi(Z))] - \mathbb{E}_{q_0}[\nabla_\psi \log p(\mathcal{D}, T_\psi(Z))]. \tag{3.15}$$

Now, we can use Monte Carlo integration to approximate the gradients of the log densities as we would with any other function of the random variable,

$$\begin{aligned} \nabla_\psi \,\mathrm{KLD}[q_\psi(x)||p(\mathcal{D}, x)] &\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\psi \log q_\psi(T_\psi(z_i)) - \frac{1}{N}\sum_{i=1}^{N} \nabla_\psi \log p(\mathcal{D}, T_\psi(z_i)) \\ &= \nabla_\psi \frac{1}{N}\sum_{i=1}^{N} \log q_\psi(T_\psi(z_i)) - \log p(\mathcal{D}, T_\psi(z_i)), \quad z_i \sim q_0. \end{aligned} \tag{3.16}$$

The forward KLD instead relies on expectations over the target probability space,

$$\mathrm{KLD}[p(x|\mathcal{D})||q_\psi(x)] = \log Z + \mathbb{E}_p[\log p(\mathcal{D}, X)] - \mathbb{E}_p[\log q_\psi(X)], \tag{3.17}$$

and the first-order derivative of this objective with respect to the hyperparameters

ψ requires evaluating only one expectation,

$$\begin{aligned}\nabla_\psi \operatorname{KLD}[p(x|\mathcal{D})||q_\psi(x)] &= -\mathbb{E}_p[\nabla_\psi \log q_\psi(X)] \\ &\approx -\nabla_\psi \frac{1}{N}\sum_{i=1}^{N} \log q_\psi(x_i), \quad x_i \sim p. \end{aligned} \tag{3.18}$$

The real challenge is sampling from the target, a problem that can generally only be asymptotically solved using methods described in Chapter 2. Chapter 4 introduces a novel scheme to train (3.18) using samples generated with Markov chain methods.

3.2 Variational Family

The derivation of the Monte Carlo approximation to the gradient of the reverse KLD (3.16) suggests a generic way to construct expressive variational families: find a base distribution q_0 that is easy to sample from, parameterise a diffeomorphism T_ψ, use it to define the parametrised approximation q_ψ. Notice that now, using the change of variables formula, the approximate density is defined in terms of the base distribution and the diffeomorphism,

$$q_\psi(x) = q_0(T_\psi^{-1}(x))|\det \nabla T_\psi^{-1}(x)|. \tag{3.19}$$

Since (3.19) needs to be computed at each gradient evaluation of the KLD and the determinant of a d-dimensional matrix requires $d!$ operations to evaluate in general, we want to constrain T_ψ so that its Jacobian matrix has some structure that facilitates the computation of its determinant.

Start with a scenario where we want to approximate our target density using the multivariate Gaussian family, parametrised by their mean vector μ and covariance matrix Σ. Making L the square root of the covariance matrix, for instance, using a Cholesky decomposition $\Sigma = LL^T$, we can make $Z \sim q_0$ a random vector of d independent standard Gaussian variables and use $T_\psi(Z) = \mu + LZ$ to generate the multivariate Gaussian random vector X, where $\psi = (\mu, L)$ and L is lower triangular with positive diagonal elements.

Sampling d independent standard Gaussian random variables is straightforward and cheap using the Box-Muller transform: start with two uniform random variables (U_1, U_2), use the inverse transform method described at the start of Chapter 2 to convert them into polar coordinates in the Gaussian sphere,

$$(R, \theta) = (-2 \log U_1, 2\pi U_2), \tag{3.20}$$

and convert them back to Cartesian points,

$$(Z_1, Z_2) = (\sqrt{R} \cos \theta, \sqrt{R} \sin \theta). \tag{3.21}$$

Since Σ is a positive definite matrix, we can be sure the transformation T_ψ is bijective for any hyperparameter ψ. Also, the Jacobian matrices of T_ψ and T_ψ^{-1} are lower triangular, requiring only d operations to calculate its determinant. The diagonal elements of L need to be positive, so instead, we optimize their logarithm over all real values. We can now use first-order optimization methods to optimize over reverse KLD using (3.16) and get an approximation like on the left of Figure 3.1, or over the forward KLD using (3.18) and get an approximation like on the right of Figure 3.1.

This is an example of a linear transformation with a triangular shift matrix; other constraints can be put on the shift matrix to make the computation of its Jacobian simple, such as orthogonal matrices (Tomczak and Welling, 2016), or parametrise its LU factorization (Durk P Kingma and Dhariwal, 2018) or QR decomposition (Hoogeboom, Van Den Berg, and Welling, 2019) instead of the shift matrix directly. Non-linear transformations are more restrictive since they require a specific form for a constrained Jacobian, some examples are the planar and radial flows of Rezende and Mohamed (2015). The following sections will discuss a general strategy to build expressive transformations using many hyperparameters, leveraging neural networks on their parameterisation, while keeping its Jacobian matrix constrained.

3.2.1 Discrete Normalizing Flows

The linear transformation described above is an example of a diffeomorphism with a triangular structure. That is, a transformation where each dimension is a function of only the current dimension and all dimensions that come before it, assuming some specific order for the dimensions of $X = [X_1, \ldots, X_d]^T$ and $Z = [Z_1, \ldots, Z_d]^T$,

$$X_i = T_i(Z_1, \ldots, Z_i), \quad i = 1, \ldots, d. \tag{3.22}$$

In terms of optimal transport, this is called Rosenblatt's (1952) transformation, and it serves as a regularity constraint that would normally be enforced using a cost function on the distance between the original Z and the transformed X (see discussion in El Moselhy and Y. M. Marzouk, 2012). This structural constraint ensures both the existence and uniqueness of a transport map that transforms samples from the reference distribution to samples from the target distribution under some constraints on both measures, such as having density functions (Bogachev, Kolesnikov, and Medvedev, 2005). Thus, it provides a theoretical guarantee of universality or the guarantee that there is some triangular transformation that can learn our target.

Considering the theoretical guarantees of triangular transformations, a general class of transformations that maintain a triangular or autoregressive structure, referred to as autoregressive normalizing flows in the machine learning community, are defined as,

$$X_i = T_i(Z_i \ ; \ \Psi(Z_1, \ldots, Z_{i-1})), \quad i = 1, \ldots, d, \tag{3.23}$$

where Ψ can be any *conditioning* function generating the parameters of the *coupling* function T_i. Since the hyperparameters depend only on $Z_1, \ldots, Z_{i-1}$, the transformation still has a triangular structure, and the determinant of its Jacobian matrix is just the product of the matrix's diagonal elements,

$$\det \nabla T(z) = \prod_{i=1}^{d} \frac{\partial T_i}{\partial z_i}. \tag{3.24}$$

Neural networks are used for the conditioning function, making it arbitrarily complex, while simpler invertible transformations are used as coupling functions. A favoured example of the latter is a linear transformation,

$$T_i(Z_i \ : \ \psi_1, \psi_2) = \exp \psi_1 Z_i + \psi_2, \tag{3.25}$$

$$\Psi(Z_1, \ldots, Z_{i-1}) = (\psi_1, \psi_2), \tag{3.26}$$

introduced in the inverse autoregressive (Durk P Kingma, Salimans, et al., 2016) and masked autoregressive (Papamakarios, Pavlakou, and Murray, 2017) flows. While more complex coupling functions have been proposed, see Kobyzev, Prince, and Brubaker (2020) for an extensive treatment, the linear coupling's simplicity is usually preferred when creating arbitrary complexity by using large neural networks in the conditioning function and composing a sequence of simple linear transformations.

An important property of diffeomorphic transformations is that they are composable: for any finite amount of invertible and differentiable transformations $T_1, ..., T_K$, their composition $T_K \circ \cdots \circ T_1$ is also invertible and differentiable, and the determinant of the Jacobian matrix for the composition can be easily derived using the identity,

$$\det \nabla(T_K \circ \cdots \circ T_1)(Z) = \det \nabla T_K((T_{K-1} \circ \cdots \circ T_1)(Z)) \times \cdots \times \det \nabla T_1(Z). \tag{3.27}$$

Thus, composing a sequence of diffeomorphic transformations is simple: evaluate each transformation and the determinant of its Jacobian in sequence and use the last transformed value and the product of the determinants in (3.19).

Notice that if we have samples from Z and want to transform them to $X = T(Z)$ using an autoregressive flow, each dimension can be evaluated in parallel since they all depend on values of Z. However, if we have samples from X and want to evaluate their density (3.19), we would need to invert the transformation $Z = T^{-1}(X)$, and each dimension needs to be evaluated sequentially since they all depend on previous dimensions of Z. This makes the forward transformation much more efficient to evaluate as the dimension of the target grows. If we focus only on sampling or density

evaluation, we can make the parallel computation T or T^{-1}. But, if we need both sampling and density evaluation in our analysis, an autoregressive flow might be computationally restrictive.

The coupling flow allows parallel computation of T and its inverse T^{-1} at the cost of losing the theoretical guarantees of the autoregressive flow. We now need to split the input of the flow into two disjoint partitions $(Z^A, Z^B), (X^A, X^B) \in \mathbb{R}^m \times \mathbb{R}^{d-m}$ then the coupling flow is defined as,

$$X^A = T_A(Z^A \ : \ \Psi(Z^B)) \tag{3.28}$$

$$X^B = Z^B, \tag{3.29}$$

where again Ψ is a conditioning function generating the parameters of the coupling function T_A. Usually, this transformation is composed with the opposite transforming $X^B = T_B(Z^B \ : \ \Psi(Z^A))$ while leaving $X^A = Z^A$ for a complete transformation of all parameters.

Since the input generating the parameters of the transformation is fixed, the forward and inverse of this flow can be evaluated in parallel. Also, its Jacobian matrix is a block triangular matrix, and its determinant is simply,

$$\det \nabla T(z) = \prod_{i \in A} \frac{\partial T_A}{\partial z_i}. \tag{3.30}$$

Some examples of coupling flows are NICE (Dinh, Krueger, and Y. Bengio, 2014), RealNVP (Dinh, Sohl-Dickstein, and S. Bengio, 2016), and Glow (Durk P Kingma and Dhariwal, 2018), where conditioning functions are again neural networks and coupling functions are linear, component-wise transformations.

3.2.2 Continuous Normalizing Flows

Composing a series of one-step transformations $T_1, ..., T_K$ creates a flow of K discrete time steps. Increasing the number of time steps in the flow allows for arbitrarily complex transformations. An alternative strategy is to consider a single time step instead and parameterise the flow's infinitesimal dynamics, integrating them to get

the bijective transformation. In other words, we construct the flow by defining an ordinary differential equation (ODE) that describes the flow's evolution over time, where time represents a real-valued scalar variable analogous to the number of steps.

Continuous normalizing flows (CNFs) provide a more flexible way to model complex distributions, allowing for smooth transformations that can capture intricate structures in the data more naturally. A neural network specifies the time dependent vector field of the ODE, modelling the probability density path between reference and target density. Then, we use numerical tools to solve the ODE and construct a time dependent diffeomorphic map used as a normalizing flow.

R. T. Chen et al. (2018) introduced the use of an ODE defined in times $t \in [0, 1]$ and letting a diffeomorphism be its solution at time $t = 1$. This can be seen as an infinitely deep normalizing flow by taking the limit on the number and size of discrete steps in the transformation. Let v_t be a *time-dependent vector field* that runs continuously in the unit interval. This vector field can be used to construct a time-dependent diffeomorphic map called a *flow* $\phi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, defined via the ODE,

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(x) = v_t(\phi_t(x)) \tag{3.31}$$

$$\phi_0(x) = x. \tag{3.32}$$

Given a reference density q_0, and the flow ϕ, we can generate a *probability density path* $q : [0, 1] \times \mathbb{R}^d \to \mathbb{R}_+$ as the pushforward of q_0 under ϕ, $q_t := [\phi_t]_\sharp q_0$ for $t \in [0, 1]$, yielding, via the instantaneous change of variables formula (e.g., R. Chen and Lipman, 2024),

$$\log q_t(x_t) = \log q_0(x) - \int_0^t \nabla \cdot v_s(x_s)\mathrm{d}s, \tag{3.33}$$

where $x_s := \phi_s(x)$, and ∇ is the divergence operator, i.e. the trace of the Jacobian matrix.

The trace operator might appear more computationally feasible than the determinant, but it requires d backpropagation gradient calculations to compute.

We can instead approximate it using one backpropagation gradient calculation using Hutchinson's trace estimator (Hutchinson, 1989; Grathwohl et al., 2018),

$$\nabla \cdot v_t(\phi_t(x)) \approx z^T \nabla v_t(\phi_t(x)) z, \tag{3.34}$$

where z can be any d-dimensional random vector with zero mean and unit covariance, making it feasible to handle high-dimensional problems.

In modern applications, the vector field v_t is parameterized using a neural network v_t^ψ, in which case the ODE is referred to as a *neural ODE* (R. T. Chen et al., 2018), yielding a deep parametric model ϕ_t^ψ for the flow. In terms of normalizing flows, the diffeomorphic transformation $T_\psi = \phi_1^\psi$ and the approximate density (3.19) is given by $q_1^\psi := [\phi_1^\psi]_\sharp q_0$,

$$\log q_1^\psi(x_t) = \log q_0(x) - \int_0^1 \nabla \cdot v_s^\psi(x_s) \mathrm{d}s. \tag{3.35}$$

We can train CNFs by incorporating the pushforward distribution into the KLD as in the discrete case. This approach minimises the KLD between the target and its approximation (3.35), ensuring that the flow accurately models the probability path transforming reference to target. However, alternative methods for training CNFs avoid computing costly integrals at each iteration. These methods will be explored in detail in Chapter 5, introducing a scheme that uses Markov chain samples to learn a CNF approximating the target.

Chapter 4

Transport Elliptical Slice Sampling

We propose a new framework for efficient sampling from complex probability distributions using a combination of normalizing flows and elliptical slice sampling (Murray, R. Adams, and MacKay, 2010). The central idea is to learn a diffeomorphism, through normalizing flows, that maps the non-Gaussian structure of the target distribution to an approximately Gaussian distribution. We then use the elliptical slice sampler, an efficient and tuning-free Markov chain Monte Carlo (MCMC) algorithm, to sample from the transformed distribution. The samples are then *pulled back* using the inverse normalizing flow, yielding samples that approximate the stationary target distribution of interest. Our transport elliptical slice sampler (TESS) is optimized for modern computer architectures, where its adaptation mechanism utilizes parallel cores to run multiple Markov chains for a few iterations rapidly. Numerical demonstrations show that TESS produces Monte Carlo samples from the target distribution with lower autocorrelation compared to non-transformed samplers and demonstrates significant efficiency improvements when compared to gradient-based proposals designed for parallel computer architectures, given a flexible enough diffeomorphism.

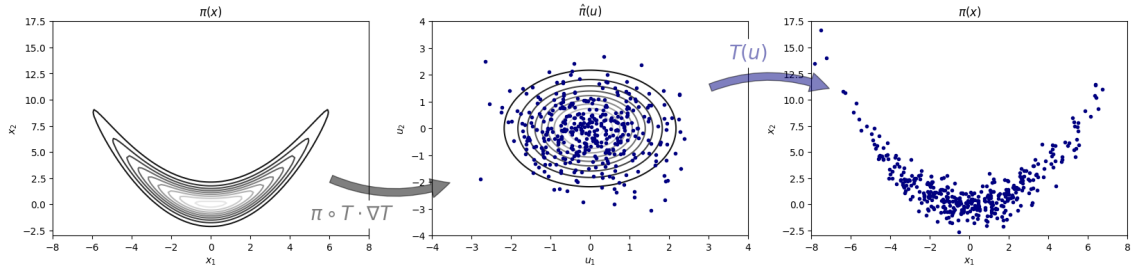


Figure 4.1: Illustration of Algorithm 12 using an exact transport map, i.e. equality in (4.2) holds: sampling from the Banana density $\pi(x_1, x_2) \propto \exp\left(-[x_1^2/8 + (x_2 - x_1^2/4)^2]/2\right)$ using the transport map $T(u_1, u_2) = (\sqrt{8}u_1, u_2 + 2u_1^2)$ starts by transforming the target space to the reference space via a change of variables, drawing samples from an ellipsis on the extended reference space (not pictured) and pushing samples back to the target space.

4.1 Introduction

Markov Chain Monte Carlo (MCMC) algorithms enable scientists to draw samples from complex distributions, typically produced by models that represent the intricate details in real-world datasets. The exploration of these complex and high-dimensional distributions is challenging, and to be efficient, practitioners use the local pointwise information of the target distribution to create a Markov chain of dependent samples. The ideal outcome would be to have independent samples, but the Markov chain approach generates sequentially correlated samples. Therefore, a major focus in MCMC research is to develop algorithms that reduce these correlations and generate samples that approximate independence.

Designing efficient MCMC algorithms usually relies on using local gradient information from the target distribution; by discretizing, for example, Hamiltonian (Duane et al., 1987; Radford M Neal et al., 2011) or Langevin (Rossky, Doll, and Friedman, 1978; Grenander and Miller, 1994) dynamics of a process stationary on our target distribution. Calculating gradients has been automated (Linnainmaa, 1976), but optimizing these algorithms to minimize both computations and

correlations between sequential samples efficiently requires algorithmic parameters to be manually tuned. Much work has been dedicated to developing efficient, black-box methods to tune these parameters, with notable examples including the NUTS (M. D. Hoffman, Gelman, et al., 2014) algorithm, which is widely available in probabilistic programming languages (Salvatier, Wiecki, and C. Fonnesbeck, 2016; Carpenter et al., 2017; Bingham et al., 2019; Phan, Pradhan, and Jankowiak, 2019).

Within the machine learning community, variational inference (VI; M. I. Jordan et al., 1999) has grown in popularity as an inexact but comparatively faster approach to solving the same inferential problem. As such, MCMC has lost its preferential status as the default approach for Bayesian inference for prediction and uncertainty quantification in this thriving community. Recent efforts (M. Hoffman, Radul, and Sountsov, 2021; M. D. Hoffman and Sountsov, 2022) have focused on speeding up MCMC by focusing on widening instead of lengthening computations on modern computer architectures, e.g. utilizing GPUs or TPUs, which allow for vast parallel computations. Tuning parallel MCMC chains has proven to be a somewhat different challenge from its sequential counterpart (Radul et al., 2020), and parallel efforts need to consider the lockstep necessity of gradient evaluations of parallel chains on modern vector oriented libraries (Abadi et al., 2016; Bradbury et al., 2018; Paszke et al., 2019).

4.2 Transport Elliptical Slice Sampler

In this research, we assume that $x \in \mathcal{X} \subset \mathbb{R}^d$ are model parameters and $\mathcal{D}$ represents our data. Our goal is to then approximate the posterior distribution, whereby Bayes rule the posterior density is given by $\pi(x) \propto L(\mathcal{D}|x)\pi_0(x)$, for $L(\mathcal{D}|x)$ the likelihood function and $\pi_0(x)$ the prior density. Our goal is to introduce a new MCMC algorithm which leverages the tuning-free nature of elliptical slice sampling with the efficient density transformation tools of normalising flows, thus creating the *transport elliptical slice sampler* (TESS); an adaptive mechanism that allows scientists to perform fast

parallel sampling from unnormalized densities. An intuitive pictorial representation of our TESS algorithm is given in Figure 4.1.

4.2.1 Elliptical slice sampling

Introduced by Murray, R. Adams, and MacKay (2010) as a simple MCMC algorithm with no tuning parameters, the elliptical slice sampler builds on a Metropolis-Hasting sampler introduced by R. Neal (1998), which is designed for situations where the prior $\pi_0(x)$ is Gaussian. Without loss of generality, we can assume that the prior is a standard Gaussian density $\pi_0(x) = \phi(x)$.The algorithm of R. Neal (1998) proceeds by first proposing a new state of the Markov chain, $x' = \sqrt{1-\beta^2}x + \beta v$, where $v \in \mathbb{R}^d$ is an independent *momentum* variable following a standard Gaussian distribution. The proposal moves x along the half ellipse, which connects the points v and $-v$, which pass through x, for values $\beta \in [-1, 1]$.

Elliptical slice sampling, instead, uses the proposal $x' = x\cos\theta + v\sin\theta$ which moves on the full ellipse connecting x, $-x$, v and $-v$ for $\theta \in [0, 2\pi]$. This ellipse moves in the contours of a standard Gaussian distribution, making it ideal for sampling in a well-tuned pullback space. Both proposals leave the prior density invariant, and elliptical slice sampling uses the slice sampling algorithm (Radford M Neal, 2003) to choose a value θ, which ensures that the likelihood $L(\mathcal{D}|x)$ is invariant. Overall, this proposal scheme keeps the target posterior invariant (Murray, R. Adams, and MacKay, 2010), details of which are presented in Algorithm 11.

4.2.2 Normalizing flows

Normalizing flows (NF; Rezende and Mohamed, 2015) are a flexible class of transformations produced by the sequential composition of invertible and differentiable mappings. Using NF involves choosing a simple *reference density*, for example, a standard Gaussian distribution $\phi(\cdot)$, and a parameterized diffeomorphism T_ψ, with optimized parameters ψ, to transform the reference density to our target $\pi(x)$ via a change of variables. In other words, we want to find a map T_ψ such that for $u \sim \phi$

Algorithm 11 Elliptical slice sampler (Murray, R. Adams, and MacKay, 2010)

Require: $x, L(\mathcal{D}|\cdot)$

1: $v \sim \mathcal{N}(0, \mathbb{I}_d)$
2: $w \sim \text{Uniform}(0, 1)$
3: $\log s \leftarrow \log L(\mathcal{D}|x) + \log w$
4: $\theta \sim \text{Uniform}(0, 2\pi)$
5: $[\theta_{min}, \theta_{max}] \leftarrow [\theta - 2\pi, \theta]$
6: $x' \leftarrow x \cos\theta + v \sin\theta$
7: **if** $\log L(\mathcal{D}|x') > \log s$ **then**
8: Return x'
9: **else**
10: **if** $\theta < 0$ **then**
11: $\theta_{min} \leftarrow \theta$
12: **else**
13: $\theta_{max} \leftarrow \theta$
14: **end if**
15: $\theta \sim \text{Uniform}(\theta_{min}, \theta_{max})$
16: Go to 6.
17: **end if**

and $x = T_\psi(u)$, we have $x \sim \pi$. Assuming this function exists, applying a change of variable yields the following identities

$$\pi(x) = \phi(T_\psi^{-1}(x))|\det \nabla T_\psi^{-1}(x)| =: \hat{\phi}(x) \tag{4.1}$$

$$\phi(u) = \pi(T_\psi(u))|\det \nabla T_\psi(u)| =: \hat{\pi}(u), \tag{4.2}$$

where ∇T_ψ and ∇T_ψ^{-1} are the Jacobian matrices of T_ψ and its inverse, respectively. In the context of VI, an approximation of $\hat{\phi}(x)$ would approximate our target density when carrying out inference since this approximation is both normalized and trivial to sample from.

4.2.3 Fixed transport maps with elliptical slice sampling

To fulfil our requirement for a simple and cost-effective MCMC proposal, we begin by generalizing the dimension-independent, gradient-free, and tuning-free elliptical slice sampler. We will add tuning parameters to our generalized elliptical slice sampler using NF. The diffeomorphism T_ψ will be responsible for efficiently exploring the posterior target density by transforming the proposal's dynamics and tracing the contours of a standard Gaussian density to follow the contours of an approximation of the target. Following previous works in the transport Monte Carlo field (as described in Section 4.3), we present TESS as a two-step procedure. Firstly, we learn the transport map between the target and reference densities. Secondly, we utilize the transport map within the elliptical slice sampler to generate samples from the target density.

1. Map optimization To estimate the parameters ψ of our NF map, we minimize divergence between our target density $\pi(x)$ and the *push-forward* reference density $\hat{\phi}(x)$ (4.1). For our intended purpose, by the law of the unconscious statistician, this is equivalent to minimizing the divergence between the *pull-back* target density $\hat{\pi}(u)$ (4.2) and the reference density $\phi(u)$. The Kullback-Leibler divergence (KL; Kullback and Leibler, 1951) is arguably the most widely used and studied divergence, here presented in the context of approximate Bayesian inference but also studied

in other branches of statistics and information theory (Joyce, 2011). It not only has a tractable Monte Carlo estimate, but it is directly related to the foundation of VI and provides intuition into the connection between maximizing the likelihood of observational data and minimizing the distance between target and reference densities (D. M. Blei, Kucukelbir, and McAuliffe, 2017),

$$\mathrm{KL}(\pi||\hat{\phi}) = \int \log \frac{\pi(x)}{\hat{\phi}(x)} \pi(x) dx. \tag{4.3}$$

The optimal transport map is found by optimizing the parameters ψ of the diffeomorphism T_ψ such that the Kullback-Leibler divergence between the target and reference densities is minimised, i.e.

$$\psi^* = \arg\min_{\psi \in \Psi} \mathrm{KL}(\pi||\hat{\phi}). \tag{4.4}$$

2. Sampling from the target Our proposed sampling method generalizes the elliptical slice sampler by targeting the extended state space $\pi(x)\phi(v)$ for any posterior density $\pi(x)$, regardless of the choice of the prior distribution. The target density is preconditioned using a transform via a normalizing flow to map to a standard Gaussian distribution. That is, given a map T_{ψ^*}, with fixed parameters ψ^*, such that $\hat{\pi}(u) \approx \phi(u)$ we proceed as follows: i) from an initial state $(x, v) = (T^*_\psi(u), v)$, ii) move around an ellipse connecting u and v and iii) accept the new state according to a slice variable chosen uniformly on the interval $[0, \hat{\pi}(u)\phi(v)]$. One iteration of this method is detailed in Algorithm 12.

Proposition 1. *The transition kernel of the Markov chain derived from Algorithm 12 leaves the target density $\pi(x)\phi(v)$ invariant.*

Proof. As established in Murray, R. Adams, and MacKay (2010) and Nishihara, Murray, and R. P. Adams (2014), the elliptical slice sampler and generalized elliptical sampler target the correct stationary distribution as the algorithm is reversible and produces an irreducible, aperiodic Markov chain.

The same result holds for the TESS algorithm from initial state $u = T^{-1}_\psi(x)$ and where (u, v) and (u', v') represent the initial and accepted transformed parameters

Algorithm 12 Transport Elliptical Slice Sampler

Require: $u, T_\psi(\cdot), \hat{\pi}(\cdot)$

$v \sim \mathcal{N}(0, \mathbb{I}_d)$

$w \sim \text{Uniform}(0, 1)$

$\log s \leftarrow \log \hat{\pi}(u) + \log \phi(v) + \log w$

$\theta \sim \text{Uniform}(0, 2\pi)$

$[\theta_{min}, \theta_{max}] \leftarrow [\theta - 2\pi, \theta]$

$u' \leftarrow u \cos\theta + v \sin\theta$

$v' \leftarrow v \cos\theta - u \sin\theta$

if $\log \hat{\pi}(u') + \log \phi(v') > \log s$ **then**

    $x' \leftarrow T_\psi(u')$

    Return x', u'

else

    if $\theta < 0$ **then**

        $\theta_{min} \leftarrow \theta$

    else

        $\theta_{max} \leftarrow \theta$

    end if

    $\theta \sim \text{Uniform}(\theta_{min}, \theta_{max})$

    Go to 6.

end if

of the sampler (steps 1 and 6-7), with s the slice variable (step 3) and $\{\theta_k\}_{k=1}^K$ the parameters representing points in the slice expressed in radians until acceptance at K (steps 4 and 17). Let

$$\theta'_k = \begin{cases} \theta_k - \theta_K, & \text{if } k < K, \\ -\theta_K & \text{if } k = K, \end{cases} \tag{4.5}$$

then by the properties of the elliptical slice sampler, the transformation

$$(u, v, s, \{\theta_k\}_{k=1}^K) \mapsto (u', v', s, \{\theta'_k\}_{k=1}^K)$$

is bijective, preserves volume and $p(\{\theta_k\}_{k=1}^K|u, v, s) = p(\{\theta'_k\}_{k=1}^K|u', v', s)$. Using the uniform density of the slice variable s it is easy to see that

$$p(u', v', \{\theta_k\}_{k=1}^K, s|u, v)\hat{\pi}(u)\phi(v) = p(u, v, \{\theta'_k\}_{k=1}^K, s|u', v')\hat{\pi}(u')\phi(v'),$$

and so if $(u, v) \sim \hat{\pi}(u)\phi(v)$ then $(u', v') \sim \hat{\pi}(u')\phi(v')$. Finally, as $x = T_\psi(u)$ we have $(x', v') \sim \pi(T_\psi(u'))|\det \nabla T_\psi(u')|\phi(v') = \pi(x')\phi(v')$.

□

The TESS algorithm will likely be geometrically ergodic under certain transformations if those transformations lead to nice tail properties on the *pulled back* target. A sketch of this argument follows from three key components: (i) Natarovskii, Rudolf, and Sprungk (2021) show that the standard elliptical slice sampler is geometrically ergodic if the target density has tails which are rotationally invariant and monotonically decreasing, e.g. $\exp(-c||x||)$ for $c > 0$. (ii) This implies geometric ergodicity for TESS if for a target density π and Markov transition kernel $P(x, \cdot)$, with $C > 0$ and $\gamma \in (0, 1)$, geometric ergodicity of the elliptical slice sampler holds when

$$||P^n(x, \cdot) - \pi||_{TV} \leq C(1 + ||x||)\gamma^n, \quad \forall n \in \mathbb{N}, \forall x \in \mathbb{R}^d.$$

Then for $\tilde{P}(x, \cdot)$ the transition kernel of TESS we have,

$$\begin{aligned} ||\tilde{P}^n(u, \cdot) - \hat{\pi}||_{TV} &= ||P^n(T_\psi(u), \cdot) - \hat{\pi}||_{TV} \\ &\leq C(1 + ||T_\psi(u)||)\gamma^n, \end{aligned}$$

which holds only if the transformation T_ψ leads to nice tail properties for $\hat{\pi}$. (iii) Following from Theorems 2 and 3 of Johnson and Geyer (2012), if there exists a diffeomorphism which ensures that T pulls in the tails of the distribution enough, then geometric ergodicity holds on the transformed distribution. An open question is determining the necessary conditions on T_ψ for this result to hold beyond simple transformations.

4.2.4 Adaptive transport maps

There are two key components to TESS, the MCMC sampling phase and the transformation function T_ψ, which so far we have treated as two independent procedures. However, the function T_ψ is parameterised by ψ, and these parameters must be learnt using samples from the target $\pi(x)$. Therefore, we propose an adaptive version of TESS that alternates between optimizing ψ and sampling x to produce an accurate map between the reference measure and the target distribution.

The parameters ψ are optimized by first running the TESS sampling procedure (Alg. 12) using k parallel Monte Carlo chains with an initial value of ψ, initialized randomly, resulting in k approximate samples from our target $\pi(x)$. We then run m iterations of a stochastic gradient descent algorithm on the loss function

$$\mathrm{KL}(\pi(x)||\hat{\phi}(x)) \approx \frac{1}{k}\sum_{i=1}^{k}\log\frac{\pi(x_i)}{\hat{\phi}(x_i)}. \tag{4.6}$$

The warm-up stage of the sampler repeats this process for h epochs with batches of size k, adjusting the inherited parameters from the previous epoch and finally fixing the parameters to then iterate N times Algorithm 12, generating samples from our extended target space $\pi(x)\phi(v)$. This adaptive sampling algorithm is detailed in Algorithm 13.

An important property of the Kullback-Leibler divergence is that it is an asymmetric divergence, i.e. $\mathrm{KL}(\pi||\hat{\phi}) \neq \mathrm{KL}(\hat{\phi}||\pi)$. Minimizing $\mathrm{KL}(\hat{\phi}||\pi)$ forces $\pi(x)$ to cover the mass of $\hat{\phi}(x)$, thus producing a poor approximation of the tails of the posterior target density. Alternatively, minimizing $\mathrm{KL}(\pi||\hat{\phi})$ forces $\hat{\phi}(x)$ to cover

the mass of $\pi(x)$, providing an overconfident approximation to the target density that can be corrected using a sampling method that leaves the target distribution invariant.

Algorithm 13 Adaptive TESS

Require: $u_{1:k}^{(0)}, h, m, N,$ `TESS` ▷ `TESS` applies Algorithm 12

Set initial parameters of T_ψ and $\hat{\pi}$.
for $t \leftarrow 1, \ldots, h$ **do** ▷ Warm-up
    for $i \leftarrow 1, \ldots, k$ **do**
        $x_i^{(t)}, u_i^{(t)} \leftarrow \texttt{TESS}(u_i^{(t-1)}, T_\psi, \hat{\pi})$
    end for
    Update ψ in T_ψ by running m iterations of gradient descent on (4.6) using samples $x_{1:k}^{(t)}$.
end for
$u_{1:k}^{(0)} \leftarrow u_{1:k}^{(h)}$
for $t \leftarrow 1, \ldots, N$ **do** ▷ Sampling
    for $i \leftarrow 1, \ldots, k$ **do**
        $x_i^{(t)}, u_i^{(t)} \leftarrow \texttt{TESS}(u_i^{(t-1)}, T_\psi, \hat{\pi})$
    end for
end for
Return $x_{1:k}^{(1)}, \ldots, x_{1:k}^{(N)}$

We follow the approach of M. Hoffman, Sountsov, et al. (2019) and initialize the parameters of the NF using an approximation of the parameters that minimize $\mathrm{KL}(\hat{\phi}||\pi)$ via a stochastic gradient descent scheme. In other words, minimizing the Monte Carlo approximation

$$\mathrm{KL}(\phi(u)||\hat{\pi}(u)) \approx \frac{1}{M}\sum_{i=1}^{M} \log \frac{\phi(u_i)}{\hat{\pi}(u_i)}, \quad u_i \overset{iid}{\sim} \phi. \tag{4.7}$$

4.2.5 Choice of transport map

A wide class of linear and nonlinear functions can be used within our normalizing flow map. This work focuses on the coupling architecture for T_ψ introduced by Dinh, Krueger, and Y. Bengio (2014). Consider the disjoint partition $x = (x^A, x^B) \in \mathbb{R}^p \times \mathbb{R}^{d-p}$ and a coupling function $t(\cdot; \psi) : \mathbb{R}^p \to \mathbb{R}^p$ parameterized by some set of parameters ψ. Then, one can define a transformation $G : \mathbb{R}^d \to \mathbb{R}^d$ by the formula

$$x^A = t(u^A; \Psi(u^B)) := e^{\psi_1} \odot u^A + \psi_2 \tag{4.8}$$

$$x^B = u^B, \tag{4.9}$$

given parameters $\Psi : \mathbb{R}^{d-p} \to \mathbb{R}^p \times \mathbb{R}^p$ learned only from the extended input. Here we assume an affine bijection, defined in (4.8), and make Ψ a dense feedforward neural network, for further generalizations and variations see Kobyzev, Prince, and Brubaker (2020). The main practical advantages of the coupling architecture with affine transformations are that it is easily inverted through a shift and scale of the transformed x^A with parameters given by the unchanged $x^B = u^B$, and that the modulus determinant of its Jacobian matrix can be easily computed as $|\det \nabla G(x)| = \prod_{i=1}^d (e^{\psi_1})_i$. Furthermore, since the inverse of the transformation is of similar structure, also its constant of volume change can be easily derived as $|\det \nabla G^{-1}(x)| = \prod_{i=1}^d (e^{-\psi_1})_i$. Both of these use parameters given by $\Psi(u^B) = \Psi(x^B)$, and we drop the absolute value from our computations since the multiplied values are non-negative. We allow for arbitrary complexity of our NF by introducing a transformation $D : \mathbb{R}^d \to \mathbb{R}^d$ with the same structure as G but with the roles of the random variables reversed, i.e. $x^A = u^A$ and $x^B = t(u^B; \Psi(u^A))$. Hence making our final NF a sequential composition of $n \geq 1$ transformations $T_\psi = D_n \circ G_n \circ \cdots \circ D_1 \circ G_1$.

4.3 Related work

Elliptical slice sampling The original elliptical slice sampler paper (Murray, R. Adams, and MacKay, 2010) presented a simple algorithm that worked well on

scenarios of strong prior (Gaussian) information. Nishihara, Murray, and R. P. Adams (2014) were the first to explore the idea of generalizing this algorithm to any target distribution while trying to maintain a simple kernel. Their proposal used a Student-t distribution to approximate the target under the premise that this proposal would adequately cover the tails of the target density. Their work also considered an adaptive mechanism using parallel computing architectures, accelerating the MCMC sampler by utilizing multiple chains with fewer iterations per chain. Fagan, Bhandari, and Cunningham (2016) also used a generalized elliptical slice sampling proposal paired with a preconditioning step to alleviate complex geometry on their target, using expectation propagation to learn correlation structures for subsets of the parameter space. The main difference between previous elliptical slice sampling work and our methodology is normalizing flows to create a transport map between a Gaussian density (for which the sampler works well) and the target density of interest. As shown in Section 4.4, utilizing the richness of nonlinear transport maps produces a fast and highly efficient MCMC algorithm.

Monte Carlo transport maps Our work draws inspiration and is closely related to several threads of work that approach the problem of simulation by simplifying the structure of the target density through a preconditioning step. For general MCMC proposals, Parno and Y. M. Marzouk (2018) introduced the idea of learning a diffeomorphism using samples from an MCMC algorithm to approximate (4.3). Their work built on El Moselhy and Y. M. Marzouk (2012)'s proposal for approximate inference, adding an MCMC kernel that corrects the approximation and provides asymptotically exact samples. Their work showed that a relatively simple transformation can provide valuable information about the global structure of the target density, thus improving the efficiency of MCMC algorithms that use local gradient information on certain, especially degenerate, test cases.

MCMC with normalizing flows The NeuTra Hamiltonian Monte Carlo (HMC) algorithm introduced in M. Hoffman, Sountsov, et al. (2019) combines neural transport maps with the HMC algorithm. This builds on the earlier work

of Y. Marzouk et al. (2016), who frame the approximate inference problem as solving a two-step process, where firstly an optimization problem is solved to find a preconditioned diffeomorphism which minimizes (4.7). Then, the preconditioned target is sampled using an HMC algorithm. The NeuTra algorithm relies on gradient-based proposals to explore the target density. A key difference from the transport elliptical slice sampler is that gradients of the target density are not required, this makes the algorithm faster than gradient-based MCMC algorithms, and as illustrated in Section 4.4, this is achieved without sacrificing sampling accuracy due to the transport mapping. Additionally, TESS can be applied in settings where it is either infeasible to calculate target gradients or they may be unstable (e.g Neal's funnel density (Gorinova, Moore, and M. Hoffman, 2020)).

4.4 Experiments

In this section, we compare the performance of the adaptive form of TESS (Alg. 13) with the performance of several state-of-the-art MCMC algorithms designed for parallel computer architectures. Specifically, MEADS (M. D. Hoffman and Sountsov, 2022), ChEES-HMC (M. Hoffman, Radul, and Sountsov, 2021), and the popular NUTS algorithm (M. D. Hoffman, Gelman, et al., 2014) where an adaptive step size is tuned such that the average cross-chain harmonic-mean acceptance rate is approximately 0.8. We also precondition the latter NUTS method, using the same NF as in TESS, which leads to the NeuTra algorithm (M. Hoffman, Sountsov, et al., 2019). We compare the effect of TESS's overfitted adapted transformation against an underfitted transformation (i.e. reversing the KL), which, unlike TESS, is done independently and a priori to the sampling process. Each experiment runs all algorithms on 128 parallel chains for 400 warm-up iterations per chain, during which hyperparameters are tuned. Then, 100 iterations are used to produce posterior samples with fixed hyperparameters. MEADS separates the 128 chains into 4 batches of 32 chains each and tunes parameters during all 500 iterations, but only the last

100 are used as posterior samples. The code to reproduce the experiments is provided at albcab/TESS.

The transform map used in all experiments uses $n = 2$ transformations of a ψ-parameterized dense feedforward neural network with two hidden layers of the same dimension as the input (see Sec. 4.2.5 for details). The Adam (Diederik P Kingma and Ba, 2014) method estimates ψ, with a learning rate that decays exponentially over 400 iterations at a rate of 0.1 using a different initial learning rate for each experiment. For the adaptive TESS algorithm, we set $m = 1$ on all experiments.

We compare the experimental results of each algorithm based on their Monte Carlo sample efficiency, as indicated by the maximum integrated autocorrelation time (τ_{max}) with standard deviation (σ_τ). Additionally, we present the effective sample size (ESS) in terms of the median worst-case integrated autocorrelation time for individual chains and when all chains are grouped together. A more efficient algorithm is indicated by lower autocorrelations and higher ESS, indicating that samples are closer to independent. To demonstrate the impact of computational cost on each algorithm, we normalize the ESS by the run time in seconds. ESS/sec considers the time spent adapting and sampling and provides a fair comparison between algorithms. To assess the accuracy of the posterior approximation for each algorithm, we use the kernelized Stein discrepancy with U- and V-statistics, as described in (Gorham and Mackey, 2017). Lower U- and V- statistic values indicate a better approximation of the target posterior.

4.4.1 Biochemical oxygen demand model

We start with an experiment from (Parno and Y. M. Marzouk, 2018) designed to undermine gradient methods because of its rapidly changing posterior correlation structure, which is challenging for standard samplers to explore. Gradient methods capture local geometry, but the local geometry in this example is not representative of the global geometry of the target and thus provides insufficient information for efficient sampling. On the other hand, the non-linear transformation of the target

Algorithm	τ_{max}	σ_τ	ESS	ESS/chain	ESS/sec	Stein U-stat.	Stein V-stat.
TESS	**0.555**	1.485	**11523**	**90**	**1129.199**	**4.269e+02**	**4.570e+02**
MEADS	9.959	1.468	643	5	208.613	1.476e+15	1.486e+15

Table 4.1: Biochemical oxygen demand model. Algorithm diagnostics where $\tau_{\max}$ is the maximum integrated autocorrelation time over all dimensions; ESS is the corresponding minimum effective sample size. Results are averaged over multiple chains of each sampler, and σ_τ is the empirical standard deviation of $\tau_{\max}$ over these runs.

space with a NF-based approach captures the global, non-Gaussian structure of the target density.

The simple biochemical oxygen demand model is given by $B(t) = \theta_0(1-\exp(-\theta_1 t))$ for times $t < 5$. In this synthetic data experiment, we set the parameters $\theta_0 = 1$ and $\theta_1 = 0.1$ and simulate $y(t_i)$ observations at times t_i evenly spaced in $[0, 5)$ for $i = 1, \ldots, 20$ such that $y(t_i) = \theta_0(1 - \exp(-\theta_1 t_i)) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_y^2)$ and fixed $\sigma_y^2 = 2 \times 10^{-4}$. The target posterior density is given by the likelihood $L(\mathbf{y}|\theta_0, \theta_1) = \prod_i \mathcal{N}(y(t_i); B(t_i; \theta_0, \theta_1), \sigma_y^2)$ and flat prior $\pi_0(\theta_0, \theta_1) \propto 1$. The numerical results are shown in Table 4.1 and Figure 4.2 plots the Monte Carlo approximation of the posterior for the original and transformed densities.

It is clear from the results that local gradient information is insufficient to efficiently sample from the rapidly changing local correlation structure of the target density. On the other hand, the learned transport map from the warm-up procedure of TESS provides a mass-covering approximation of the global structure of the target, demonstrated in Figure 4.2 by $\hat{\phi}(u)$, which allows the algorithm to move farther away from its initial position, exploring the entire target space efficiently, and yielding not only shorter autocorrelation times but also the correct posterior estimates of the parameter space. In this case, gradient-based algorithms are forced to take very small steps while encountering large rejection probabilities, thus inefficiently producing samples from the posterior.

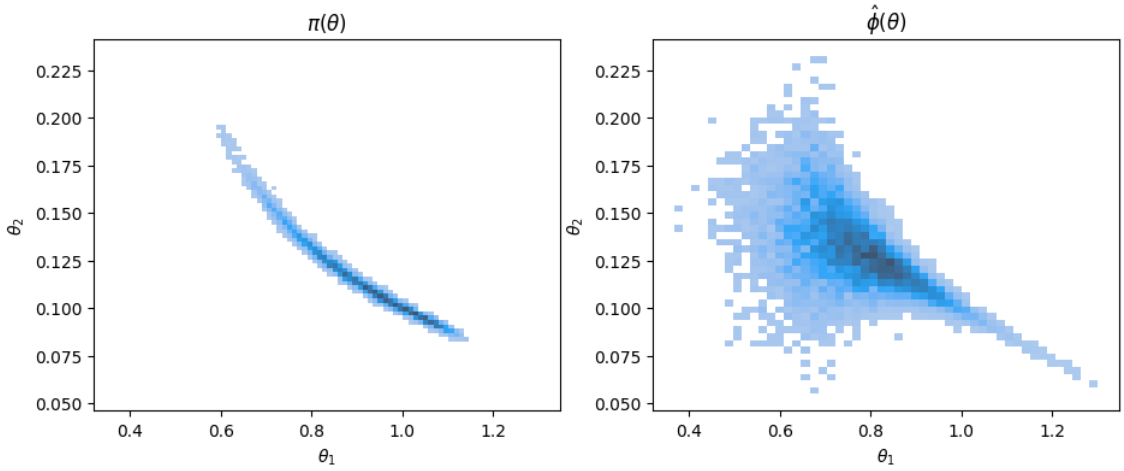


Figure 4.2: Samples from the target density $\pi(\theta)$ of the Biochemical oxygen demand model acquired by the TESS algorithm, mapped to $\hat{\phi}(\theta)$ (4.1), with diffeomorphism T_ψ learned from the warm-up procedure of Algorithm 13. With an approximation that overestimates the real variance (right) of our target (left), we can capture its global, non-Gaussian structure and explore it using a dimension-independent and gradient-free method.

4.4.2 Sparse logistic regression

Next, we consider a sparse logistic regression model with hierarchies. Regression parameters of the logistic likelihood are given a horseshoe prior (Carvalho, Polson, and Scott, 2009), which induces sparsity on the regressors, i.e. variable selection. These types of hierarchies on the prior scale of a parameter create funnel geometries that are hard to explore efficiently without a local or global structure of the target.

Algorithms are run on the non-centred parameterization (Papaspiliopoulos, G. O. Roberts, and Sköld, 2007) of our model using the numerical version of the German credit dataset. The target posterior is defined by the likelihood $L(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\lambda}, \tau) = \prod_i \text{Bernoulli}(y_i; \sigma((\tau\boldsymbol{\lambda} \odot \boldsymbol{\beta})^T X_i))$, with sigmoid function $\sigma(\cdot)$, and prior $\pi_0(\boldsymbol{\beta}, \boldsymbol{\lambda}, \tau) = \text{Gamma}(\tau; 1/2, 1/2) \prod_j \mathcal{N}(\beta_j; 0, 1) \,\text{Gamma}(\lambda_j; 1/2, 1/2)$. Numerical results for each MCMC algorithm are shown in Table 4.2. Notice how NUTS and NeuTra provide the best results, but long sampling times reflect their inefficiency when running in

Algorithm	τ_{max}	σ_τ	ESS	ESS/chain	ESS/sec	Stein U-stat.	Stein V-stat.
TESS	5.182	0.352	1235	10	34.744	1.591e+00	1.693e+00
MEADS	7.105	0.413	901	7	49.453	9.408e-01	1.079e+00
ChEES-HMC	5.666	0.380	1130	9	**81.588**	1.193e+00	1.312e+00
NUTS	4.734	0.833	1352	11	0.379	1.004e+00	1.138e+00
NeuTra	**2.482**	1.949	**2579**	**20**	0.401	**3.618e-01**	**4.971e-01**

Table 4.2: Sparse logistic regression. Algorithm diagnostics where $\tau_{\max}$ is the maximum integrated autocorrelation time over all dimensions; ESS is the corresponding minimum effective sample size. Results are averaged over multiple chains of each sampler, and σ_τ is the empirical standard deviation of $\tau_{\max}$ over these runs.

parallel. Every iteration takes as long as the longest chain takes to iterate. Waiting for all chains to catch up severely slows down sampling time, the same effect can be observed in all experiments.

As the parameter space dimension grows ($d = 51$ in this example), TESS will require more samples, i.e., more chains, for a low variance estimate of (4.7). In addition, a more complicated NF is required to capture the non-Gaussian structure of the high-dimensional target space. When either fails and the diffeomorphism T cannot capture the structure of the target space, the simple sampling procedure inherited from the elliptical slice sampler will struggle to sample from the target space, even if producing uncorrelated samples. We purposely illustrate the effect of a deficient transformation on a high-dimensional problem so that the practitioner can understand the caveats of our method. Studying ways to lower the variance of (4.7), using control variates (Lemieux, 2014) and similar methods (Botev and Ridder, 2017), as well as alternative NF schemes is left to future work.

Algorithm	τ_{max}	σ_τ	ESS	ESS/chain	ESS/sec	Stein U-stat.	Stein V-stat.
TESS	**0.267**	0.893	**23985**	**187**	**985.969**	5.120e-02	1.301e-01
MEADS	1.382	1.197	4631	36	319.949	3.066e-01	3.867e-01
ChEES-HMC	3.451	1.825	1855	14	121.756	**-8.203e-03**	**7.073e-02**
NUTS	0.282	0.403	22672	177	182.255	2.222e-02	1.009e-01
NeuTra	0.441	1.020	14530	114	209.069	1.092e-01	1.880e-01

Table 4.3: Regime switching Hidden Markov model. Algorithm diagnostics where $\tau_{\max}$ is the maximum integrated autocorrelation time over all dimensions; ESS is the corresponding minimum effective sample size. Results are averaged over multiple chains of each sampler, and σ_τ is the empirical standard deviation of $\tau_{\max}$ over these runs.

4.4.3 Regime switching Hidden Markov model

An important use of inference and uncertainty quantification is on time series data. In this example, we analyze financial time series, specifically the daily difference in log price data of Google's stock, referred to as returns r_t, for $t = 1, \ldots, 431$. We shall assume that at any given time t the stock's returns will follow one of two regimes: an independent random walk regime $r_t \sim \mathcal{N}(\alpha_1, \sigma_1^2)$, or an autoregressive regime $r_t \sim \mathcal{N}(\alpha_2 + \rho r_{t-1}, \sigma_2^2)$. We define the two regimes as $s_t \in \{0, 1\}$ and the probability of switching between or remaining within a regime at time t will depend on the regime at $t-1$, i.e. p_{s_{t-1}, s_t} for $s_{t-1}, s_t \in \{0, 1\}$. The transition probabilities $p_{1,1}$ and $p_{2,2}$, and their complementary probabilities $p_{1,2} = 1 - p_{1,1}$ and $p_{2,1} = 1 - p_{2,2}$ are treated as model parameters. Since the regime is unobserved at any time, we carry the probability of belonging to either regime over time as $\xi_{1t} + \xi_{2t} = 1$. Finally, we define the initial values for returns r_0 and the probability of belonging to one of the two regimes ξ_{10}.

The regime switching model is defined by the likelihood

$$L(\mathbf{r}|\boldsymbol{\alpha}, \rho, \boldsymbol{\sigma}^2, \mathbf{p}, r_0, \xi_{10}) = \prod_t \xi_{1t}\eta_{1t} + (1-\xi_{1t})\eta_{2t}, \tag{4.10}$$

$$\text{where } \xi_{1t} = \frac{\xi_{1t-1}\eta_{1t}}{\xi_{1t-1}\eta_{1t} + (1-\xi_{1t-1})\eta_{2t}},$$

and $\eta_{jt} = p_{j,1}\mathcal{N}(r_t; \alpha_1, \sigma_1^2) + p_{j,2}\mathcal{N}(r_t; \alpha_2 + \rho r_{t-1}, \sigma_2^2)$ for $j \in \{0, 1\}$. The prior distributions for the parameters are

$$\alpha_1, \alpha_2, r_0 \sim \mathcal{N}(0, 1), \ \rho \sim \mathcal{N}^0(1, 0.1), \tag{4.11}$$

$$\sigma_1, \sigma_2 \sim \mathcal{C}^+(1), \tag{4.12}$$

$$p_{1,1}, p_{2,2} \sim \text{Beta}(10, 2), \ \xi_{10} \sim \text{Beta}(2, 2), \tag{4.13}$$

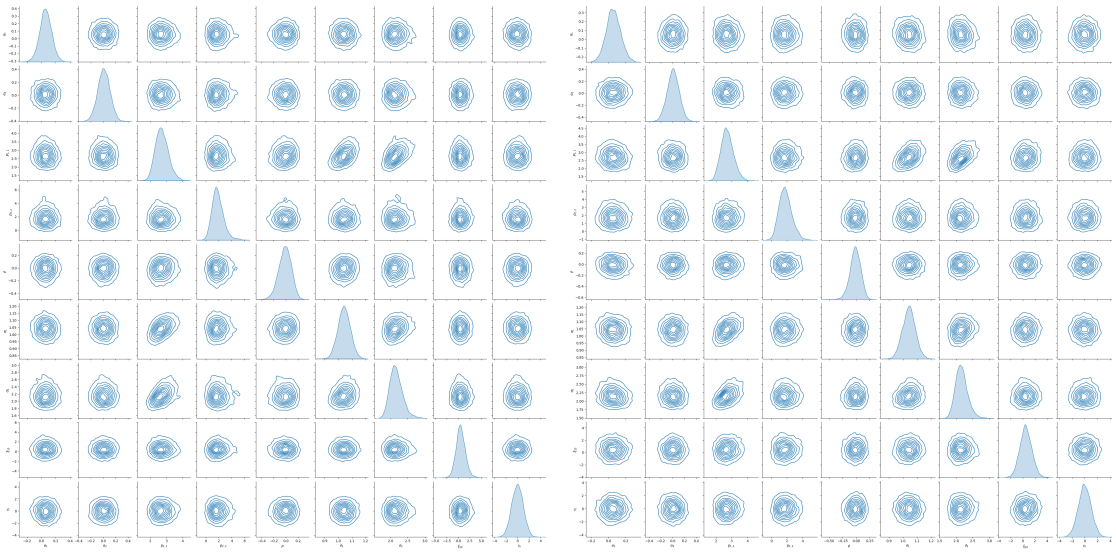
where $\mathcal{N}^0$ indicates a Gaussian distribution which is truncated at zero and $\mathcal{C}^+$ is the half-Cauchy distribution. Numerical results are shown in Table 4.3.

The marginal unimodality and somewhat independent correlation structure of the parameters make this posterior distribution easy to sample from; diagnostic results show the best performance for all algorithms with respect to other models. TESS's learned flexible transformation of the target density, allowing it to propose uncorrelated sequential samples, is fundamental for its superior diagnostics. ChEES-HMC outputs the samples with the lowest Stein discrepancy, but since it uses the same step size for all target dimensions, it struggles to mix well with the worst-case dimension. On the other hand, a flexible transport map can capture the covariance structure of the target, allowing fast mixing even on the worst-case dimension. Pair density plots are presented in Figure 4.3.

4.4.4 Predator-prey system

We consider a likelihood defined as a solution of an ODE system, specifically, the predator-prey system defined by the Lotka-Volterra equations (Goel, Maitra, and Montroll, 1971),

$$\frac{dp}{dt} = \alpha p - \beta pq, \quad \text{and} \quad \frac{dq}{dt} = -\gamma q + \delta pq, \tag{4.14}$$



Samples from TESS Samples from MEADS

Figure 4.3: Posterior density pair plots for the regime switching hidden Markov model using samples drawn with transport elliptical slice sampling on the left and MEADS (M. D. Hoffman and Sountsov, 2022) on the right. Parameters ρ, σ_1 and σ_2 are log transformed and parameters $p_{1,1}$, $p_{2,2}$ and ξ_{10} are sigmoid function transformed.

where p and q are the prey and predator populations, respectively. We can solve the ODE system of equations numerically and account for measurement error by modelling the observations as $\log p_t \sim \mathcal{N}(\log p(t), \sigma_p^2)$ and $\log q_t \sim \mathcal{N}(\log q(t), \sigma_q^2)$ for all $t > 0$. Furthermore, $p(0)$ and $q(0)$ are the initial values. Since we cannot analytically solve the system of equations, we approximate its solution using the Runge–Kutta method, adding an approximation error to our likelihood function. Data for the Hudson's Bay historical lynx-hare population are used as observations in the model. The likelihood is defined as

$$L(\mathbf{p}, \mathbf{q}|\theta) = \prod_t \mathcal{N}\left(\begin{pmatrix}\log p_t \\ \log q_t\end{pmatrix}; \begin{pmatrix}\log p(t) \\ \log q(t)\end{pmatrix}, \begin{pmatrix}\sigma_p^2 & 0 \\ 0 & \sigma_q^2\end{pmatrix}\right)$$

where $\theta = (\alpha, \beta, \gamma, \delta, \sigma_p^2, \sigma_q^2, p(0), q(0))$ and $\{p(t), q(t)\}_{t>0}$ are approximate solutions to the Lotka-Volterra system of equations initialized at $(p(0), q(0))$. Prior distributions for parameters are

$$\alpha, \gamma \sim \mathcal{N}^0(1, 1/2), \quad \beta, \delta \sim \mathcal{N}^0(1/20, 1/20), \tag{4.15}$$

$$\log \sigma_p, \log \sigma_q \sim \mathcal{N}(-1, 1), \tag{4.16}$$

$$\log p(0), \log q(0) \sim \mathcal{N}(\log 10, 1), \tag{4.17}$$

where $\mathcal{N}^0$ is a Gaussian distribution truncated at zero.

This experiment exhibits a situation similar to Section 4.4.1: gradient methods, without global information on the structure of our target, lack enough information to move efficiently around its rapidly changing correlation structure. On the other hand, TESS captures the global structure of the target using a NF and can move purposely around it when sampling. Figure 4.4 illustrates the contrast: MEADS, lacking global information about the geometry of the target, is unable to converge towards a sensible solution, exploring a region of the target space with large error variance and insignificant initial positions, both for the predator and the prey populations; on the other hand, TESS can converge towards reasonable initial populations and concentrate sampling around small error variance. Samples from the other gradient methods give results similar to those of MEADS. Gradient methods need a learned

correlation matrix that captures the global correlation structure of the target and uses gradient information to propose large steps locally. At the same time, TESS can capture both the global correlation and local structure by learning an overconfident transport map and then using this information on a cheap and gradient-free method for sampling.

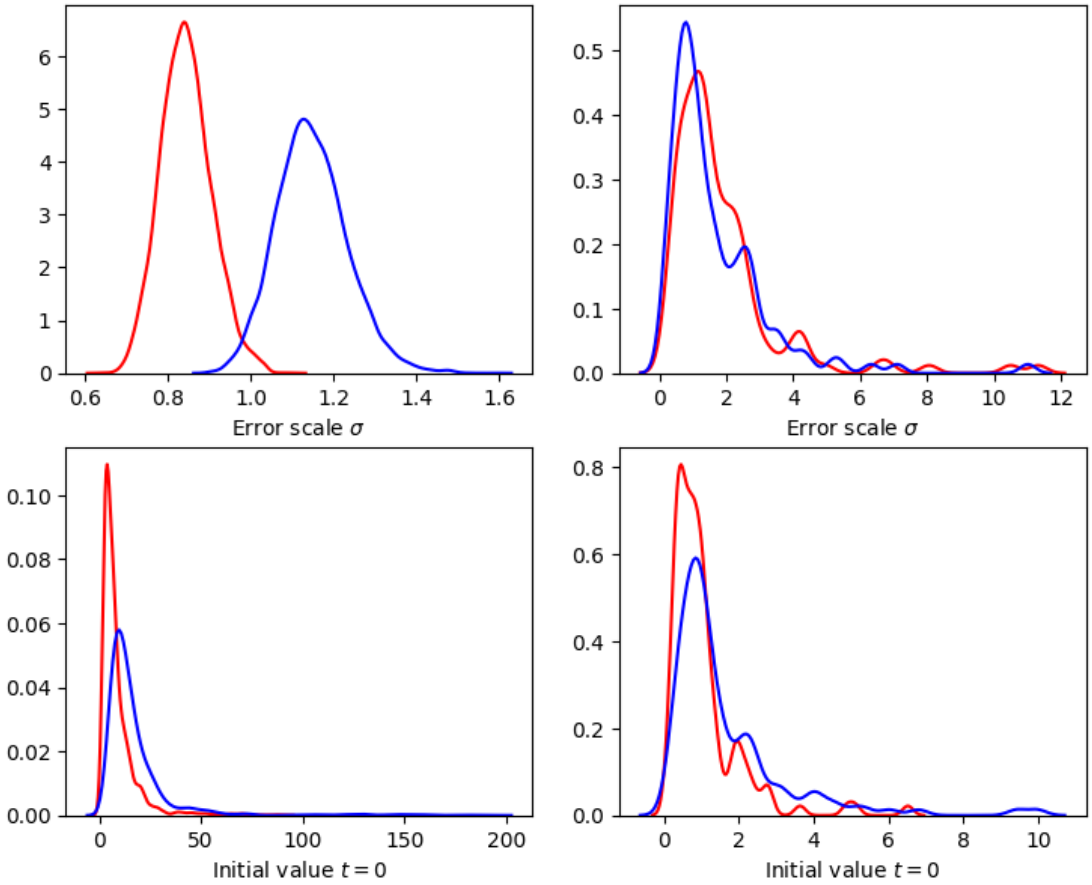


Figure 4.4: Density plots of the approximate posterior distribution for the initial values and scale parameters from the predator-prey system model, drawn with transport elliptical slice sampling on the left and MEADS on the right.

4.5 Discussion

This research proposes TESS, an MCMC algorithm that performs dimension-independent and gradient-free sampling from any unnormalized target density. We also propose an adaptive version of our algorithm that learns a non-Gaussian approximation to the target, helping the algorithm explore complex geometries efficiently. TESS can also utilize parallel computer architectures to accelerate sampling from posterior distributions. We believe this will allow practitioners to perform uncertainty quantification of their models with parallel computational resources and little time.

We found that our algorithm can outperform gradient-based competitors in various models. However, it is important to develop flexible transport maps and low-variance Monte Carlo approximations of the KL divergence, especially for high-dimensional models. Future work will explore the role of the transport map on the algorithm's efficiency and its efficacy in capturing issues in Bayesian posterior geometries and develop flexible transport maps for high-dimensional models.

Chapter 5

Bayesian Inference with Markovian Flow Matching

Continuous normalizing flows (CNFs) learn the probability path between a reference and a target density by modelling the vector field generating said path using neural networks. Recently, Lipman et al. (2022) introduced a simple and inexpensive method for training CNFs in the context of generative modelling, termed flow matching (FM). In this work, we re-purpose this method for probabilistic inference by incorporating Markovian sampling methods in evaluating the FM objective and using the learned probability path to assist with Bayesian sampling. We propose a sequential method, which uses samples from a Markov chain to fix the probability path defining the FM objective. We augment this scheme with an adaptive tempering mechanism that allows the discovery of multiple modes in the target. Under mild assumptions, we establish convergence to a local optimum of the FM objective, discuss improvements in the convergence rate, and illustrate our methods on synthetic and real-world examples.

5.1 Introduction

Bayesian inference has become an indispensable tool in various scientific disciplines, offering a robust framework for integrating prior knowledge with observed data to produce posterior distributions which reflect parameter uncertainty. In many applications, however, it is impossible to derive a closed-form representation for the posterior, so approximation techniques are needed.

Markov chain Monte Carlo (MCMC) and Variational Inference (VI) are two popular methods for approximating posterior distributions. While MCMC relies on the construction of a Markov process which admits the target as its invariant distribution, VI learns an approximation by identifying the closest member from a predefined family of distributions. State-of-the-art VI methods use normalizing flows (NFs), consisting of a sequence of invertible transformations between a reference and a target distribution, to define a flexible variational family (Rezende and Mohamed, 2015). More recently, there has been growing interest in continuous normalizing flows (CNFs), which define a path between distributions using ordinary differential equations (R. T. Chen et al., 2018). CNFs avoid the need for strong constraints on the flow (Grathwohl et al., 2018), with the downside of expensive maximum likelihood training.

The use of NFs, particularly as a tool for preconditioning complex Bayesian posteriors, has proven instrumental in accelerating sampling methods (Parno and Y. M. Marzouk, 2018; M. Hoffman, Sountsov, et al., 2019; Karamanis, Beutler, et al., 2022). The synergy between local MCMC algorithms and normalizing flows has also been explored, leading to enhanced mixing rates and effective estimation of complex posteriors (Gabrié, Rotskoff, and Vanden-Eijnden, 2022).

Our contributions. This work introduces a new probabilistic inference scheme that integrates flow matching with Markovian sampling techniques. Our *flow-informed MCMC algorithm* is a sequential method that utilizes Markov chain samples to define the probability path for the FM objective and can effectively handle complex posteriors (see Figure 5.1 for an illustration). Our scheme includes an adaptive

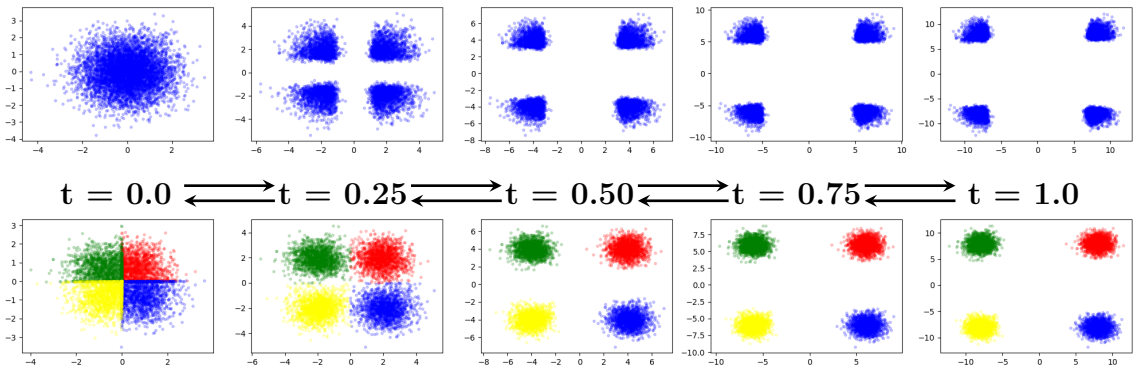


Figure 5.1: Illustration of a transformation learned using Algorithm 17. **Top**: Standard Gaussian samples are transformed to a multimodal mixture by running the dynamics forward in time given the learned vector field, left to right. **Bottom**: Samples from the multimodal mixture are transformed to standard Gaussian by running the dynamics backwards in time, right to left.

tempering mechanism, essential for discovering multiple modes in complex target distributions. Under mild assumptions, we establish that our method converges to a local optimum of the FM objective. We then empirically demonstrate that our approach accelerates convergence and significantly improves the fitting of continuous flows, thereby improving accuracy in modelling multimodal posterior distributions.

5.2 Preliminaries

Bayesian inference aims to evaluate integrals with respect to a posterior probability measure $\mu(\mathrm{d}x)$ on $\mathbb{R}^d$, with density $\pi(x)$ with respect to the Lebesgue measure. The posterior density is given by Bayes' Theorem as

$$\pi(x) = Z^{-1}\mathcal{L}(\mathcal{D}|x)\pi_0(x), \tag{5.1}$$

where $\mathcal{L}$ is the likelihood of the data $\mathcal{D}$, π_0 is the prior density, and $Z = \int \mathcal{L}(\mathcal{D}|x)\pi_0(x)\mathrm{d}x$ is the model evidence or marginal likelihood. In general, we cannot evaluate the evidence as it requires computing a potentially high-dimensional integral. Therefore, (5.1) is usually only known up to a constant of proportionality.

Thus, integrals with respect to the posterior or target measure must be estimated using Monte Carlo methods. For this, samples are generated using a sampling algorithm (e.g., MCMC), or else by approximating the target with an easy-to-sample measure.

Markov Chain Monte Carlo (MCMC) methods are a cornerstone of computational Bayesian statistics, yet they present several design challenges crucial for their practical application. MCMC algorithms require a user-defined transition kernel K, which can be numerically iterated to form a time-homogeneous Markov chain that is stationary, reversible, and irreducible with respect to the target distribution (Stoltz, Rousset, et al., 2010). An MCMC algorithm is reversible if it satisfies the detailed balance condition,

$$K(\mathrm{d}y|x)\pi(\mathrm{d}x) = K(\mathrm{d}x|y)\pi(\mathrm{d}y), \tag{5.2}$$

which is satisfied by the Metropolis-Hastings algorithm (Hastings, 1970b), with transition kernel:

$$K(\mathrm{d}y|x) = \alpha(x,y)q(\mathrm{d}y|x) + (1 - b(x))\delta_x(\mathrm{d}y). \tag{5.3}$$

Assuming that x is a sample from π, then a new sample y is generated from the proposal distribution $q(\cdot|x)$ and accepted with probability:

$$\alpha(x,y) = \min\left\{1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right\}, \tag{5.4}$$

where $1 - b(x) \in [0,1]$ is the probability that the Markov chain remains at x,

$$b(x) = \int_{\mathbb{R}^d} \alpha(x,y)q(\mathrm{d}y|x). \tag{5.5}$$

The choice of transition kernel is critical to ensuring that the MCMC algorithm can explore the target distribution within a reasonable number of Monte Carlo iterations, which can be formally defined as the mixing time. A key goal is to derive transition kernels with fast mixing times, which can be applied generically to any target distribution. Moreover, issues like mode collapse, where the chain gets trapped

in local modes of a multimodal distribution and fails to explore other significant regions, pose a significant challenge. This is particularly problematic in complex target distribution landscapes, where neglecting substantial probability mass can lead to biased inferences.

Sequential Monte Carlo (SMC) methods provide a means of approximating distributions with a set of N particles $\{x^{(i)}\}_{i=1}^{N}$. The particles are updated using importance sampling techniques, which can be applied recursively over a number of iterations to approximate the target distribution of interest (Del Moral, Doucet, and Jasra, 2006b). A particle approximation for the density of the target π is given by,

$$\tilde{\pi}(x) = \sum_{i=1}^{N} w_i \delta_{x^{(i)}}(x), \tag{5.6}$$

where w_i are the weights of the particles, and δ is the Dirac delta function. For a number of iterations $t = 1, \ldots, T$, SMC provides a scheme for evolving a set of particles from a simple distribution $p_0(x)$ at time $t = 0$, such as a Gaussian distribution, to the target distribution $\pi(x) = p_T(x)$ at time $t = T$ (Chopin, 2002; Jasra, Stephens, and Holmes, 2007).

A common way to construct the sequence of densities used in SMC is to create a geometric interpolation from the prior to the target posterior density (Radford M Neal, 2001):

$$\pi_t(x) \propto \mathcal{L}(\mathcal{D}|x)^{\beta_t} \pi_0(x), \quad t = 1, \ldots, T, \tag{5.7}$$

where $\beta_{1:T}$ is a sequence of temperatures which satisfies $0 < \beta_1 < \cdots < \beta_T = 1$. In practice, it can be difficult to choose a good sequence of temperatures that provides a smooth transition between densities. One heuristic for adaptively setting the temperature schedule relies on the effective sample size (ESS) of the particle approximation. By setting the ESS to a user-specified percentage α of the number of particles N, the next temperature β' in the schedule can be found using the bisection method (Beskos et al., 2016):

$$\frac{\left(\sum_{i=1}^{N} w_i^{\beta}(\beta')\right)^2}{\sum_{i=1}^{N} w_i^{\beta}(\beta')^2} = \alpha N, \tag{5.8}$$

where $w_i^{\beta}(\beta') = \mathcal{L}(\mathcal{D}|x)^{\beta'}\pi_0(x)/\mathcal{L}(\mathcal{D}|x)^{\beta}\pi_0(x) = \mathcal{L}(\mathcal{D}|x)^{\beta'-\beta}$ are the new importance weights given the current temperature β.

5.3 Markovian Flow Matching

In this section, we present a new approach for adaptively learning MCMC kernels using CNFs. A key challenge for MCMC practitioners is to find good proposal distributions that allow for fast and efficient sampling from complex posterior geometries. Our Markovian flow matching algorithm incorporates the global and local geometry of the target distribution to enhance sampling.

5.3.1 Continuous Normalizing Flows

Let v_t be a *time-dependent vector field* that runs continuously in the unit interval. This vector field can be used to construct a time-dependent diffeomorphic map called a *flow* $\phi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, defined via the ordinary differential equation (ODE):

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(x) = v_t(\phi_t(x)). \tag{5.9}$$

Given an initial condition $\phi_0(x) = x_0 \sim p_0$, we can transform it by running the dynamics forward in time

$$\phi_1(x) = x_0 + \int_0^1 v_t(\phi_t(x))\mathrm{d}t. \tag{5.10}$$

Alternatively, given an endpoint $\phi_1(x) = x_1 \sim p_1$, we can transform it by reversing the dynamics,

$$\phi_0^{-1}(x) = x_1 + \int_1^0 v_t(\phi_t(x))\mathrm{d}t. \tag{5.11}$$

Commonly, the vector field v_t is parameterized using a neural network v_t^{θ}, in which case the ODE in (5.9) is referred to as a *neural ODE* (R. T. Chen et al., 2018). This yields a deep parametric model for the flow ϕ_t, known as a *continuous normalizing flow* (CNF).

The change in the log density of the initial distribution at time t, can be characterized directly as

$$\frac{\mathrm{d}}{\mathrm{d}t} \log p_t(\phi_t(x)) = -\nabla \cdot v_t(\phi_t(x)), \tag{5.12}$$

where ∇ is the divergence operator, i.e. the trace of the Jacobian matrix (R. T. Chen et al., 2018). In high-dimensional problems, this term can be expensive to compute, in which case an approximation making the computation d times more efficient can be achieved by using Hutchinson's trace estimator (Hutchinson, 1989; Grathwohl et al., 2018):

$$\nabla \cdot v_t(\phi_t(x)) \approx z^T \nabla v_t(\phi_t(x)) z, \tag{5.13}$$

where z can be any d-dimensional random vector with zero mean and unit covariance. The estimator's variance can grow with the size and spectrum of the matrix. If the matrix has eigenvalues with large variance or is poorly conditioned, more random vectors are needed to achieve accurate results. Given two distributions with densities p_0 and p_1, we can construct a *probability density path* $p_t := [\phi_t]_\sharp p_0$, for $0 \leq t \leq 1$, via

$$\log p_t(\phi_t(x)) = \log p_0(\phi_0(x)) - \int_0^t \nabla \cdot v_s(\phi_s(x)) \mathrm{d}s. \tag{5.14}$$

In this case, we say that the vector field v_t *generates* the probability path p_t.

5.3.2 Flow Matching

To train the flow to map between a *reference density* p_0 and a *target density* π, in principle we can maximize the log-likelihood, since the density is tractable via (5.14). In practice, however, this is often not feasible, since both sampling and likelihood evaluation require multiple network passes to solve the ODE in (5.9).

An alternative training objective for CNFs is provided by *flow matching* (FM), an approach recently introduced by Lipman et al. (2022). Given a target probability density path p_t, with corresponding vector field u_t, we can learn a CNF defined by v_t^θ by minimizing the expected squared error,

$$\mathbb{E}_{t, p_t(x)} \| v_t^\theta(x) - u_t(x) \|_2^2, \tag{5.15}$$

where θ represents the parameters of the neural network model. The key insight in Lipman et al. (2022) is that, while we do not have direct access to the marginal target vector field, u_t, and so we cannot minimize (5.15) directly, it is equivalent to minimizing the conditional flow-matching loss

$$\mathrm{FM}(\theta) = \mathbb{E}_{t,p(x_1),p_t(x|x_1)}||v_t^\theta(x) - u_t(x|x_1)||_2^2, \tag{5.16}$$

where $p_t(x|x_1)$ is the conditional probability density path satisfying $p_0(x|x_1) = p_0$ and collapsing to a distribution centered around the sample from the target π, e.g. $p_1(x|x_1) = \mathcal{N}(x|x_1, \sigma^2\mathbb{I}_d)$, with sufficiently small $\sigma > 0$.

For simplicity, we assume that the conditional probability path is Gaussian, i.e. $p_t(x|x_1) = \mathcal{N}(x|m_t(x_1), s_t(x_1)^2\mathbb{I}_d)$. If we assume that the reference density p_0 is a standard Gaussian, then we can force the probability paths to converge to this standard Gaussian by fixing $m_0(x_1) = 0$ and $s_0(x_1) = 1$ at $t = 0$ and fixing the ending point at $t = 1$ with $m_1(x_1) = x_1$ and $s_1(x_1) = \sigma_{min}$, for some sufficiently small σ_{min}. For $0 < t < 1$, we use the optimal transport conditional probability path (Lipman et al., 2022) by setting $m_t(x_1) = tx_1$ and $s_t(x_1) = 1 - (1 - \sigma_{min})t$.

5.3.3 Flow-informed Random Walk

CNFs are designed to learn a mapping between simple and complex distributions, capturing the global geometry of the target distribution. This understanding of the global geometry of the target can be leveraged to create informed Markov transition proposals, thereby enhancing the efficiency of MCMC methods. Essentially, these flows simplify the probability space, allowing MCMC methods to explore and sample from the target space more effectively.

Using CNFs for Bayesian inference, we can use the probability path (5.14), between the target posterior density π and a reference density p_0. The target π can be evaluated on the reference space using the forward dynamics

$$\log[\phi_1]_\sharp\pi(x_0) = \log\pi(\phi_1(x_0)) - \int_0^1 \nabla \cdot v_t(\phi_t(x_0))\mathrm{d}t. \tag{5.17}$$

This is called the *pullback* target density, where the samples are pushed to target space using (5.10) and $X_1 \sim \pi$. The key challenge using this scheme for Bayesian inference is that the MCMC kernel must preserve detailed balance (5.3) using the pullback target density.

This is equivalent to using a transformation-informed proposal in a Markov transition step (Parno and Y. M. Marzouk, 2018). Where, following the reverse dynamics (5.11), MCMC samples are pushed to the target space with *pushforward* reference density,

$$\log[\phi_0^{-1}]_\sharp p_0(x_1) = \log p_0(\phi_0^{-1}(x_1)) - \int_1^0 \nabla \cdot v_t(\phi_t(x_1))\mathrm{d}t. \tag{5.18}$$

Initial positions are transformed to reference space by running the dynamics backward in time, MCMC proposals are generated in the reference space using any standard MCMC scheme, and accepted proposals are transformed back to target space.

We use the flow-informed random-walk algorithm since it outperforms all other algorithms, especially on high-dimensional problems where overfitting of the CNF can be corrected with stochastic steps, while exacerbated by independent proposals (Karamanis, Beutler, et al., 2022). The flow-informed random-walk transition P, detailed in Algorithm 14, can be written down as

$$P(\mathrm{d}y|x, \pi, \theta) = \alpha(x, y)\rho^\theta(\mathrm{d}y|x) + (1 - b(x))\delta_x(\mathrm{d}y),$$

where $\rho^\theta(\mathrm{d}y|x)$ is defined by the transition:

$$x_1 = \phi_1(x), \tag{5.19}$$

$$x_0 = \phi_0^{-1}(x) = x_1 + \int_1^0 v_t^\theta(\phi_t(x))\mathrm{d}t, \tag{5.20}$$

$$\phi_0(y) = y_0 \sim \mathcal{N}(\cdot|x_0, \sigma_{opt}^2), \tag{5.21}$$

$$y_1 = \phi_1(y) = y_0 + \int_0^1 v_t^\theta(\phi_t(y))\mathrm{d}t, \tag{5.22}$$

and $\alpha(x, y)$ and $b(x)$ as in (5.4) and (5.5) respectively.

Alternative schemes for a flow-informed Markov transition kernel are independent MH, detailed in Algorithm 15, and conditional importance sampling, detailed in Algorithm 16.

Algorithm 14 Flow-informed Random Walk

Input: x, π, θ

Output: x'

$\sigma_{opt} \leftarrow 2.38/\sqrt{d}$ ▷ Optimal scaling (G. O. Roberts and Rosenthal, 2001)

Pullback initial position (5.20)*:*

$\phi_1(x) = x_{t=1} \leftarrow x$

$$\begin{bmatrix} x_0 \\ \log p_1(x_1) - \log p_0(x_0) \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ 0 \end{bmatrix} + \int_1^0 \begin{bmatrix} v_t^\theta(\phi_t(x)) \\ -\nabla \cdot v_t^\theta(\phi_t(x)) \end{bmatrix} \mathrm{d}t$$

Random-walk proposal (5.21)*:*

$\phi_0(y) = y_{t=0} \sim \mathcal{N}(\cdot|x_0, \sigma_{opt}^2)$

Pushforward proposal (5.22)*:*

$$\begin{bmatrix} y_1 \\ \log p_1(y_1) - \log p_0(y_0) \end{bmatrix} \leftarrow \begin{bmatrix} y_0 \\ 0 \end{bmatrix} + \int_0^1 \begin{bmatrix} v_t^\theta(\phi_t(y)) \\ -\nabla \cdot v_t^\theta(\phi_t(y)) \end{bmatrix} \mathrm{d}t$$

Metropolis-Hastings correction:

$\alpha \leftarrow \min\left\{1, \frac{\pi(y_1)\exp(-\nabla_t \log p(y_1))}{\pi(x_1)\exp(\nabla_t \log p(x_0))}\right\}$

With probability α make $x' \leftarrow y_1$ else $x' \leftarrow x$

Algorithm 15 Flow-informed Independent Metropolis Hastings

Input: initial x, target density π, vector field v_t^θ, reference density q_0, flow parameters θ.

Output: x'

$\phi_1(u) = u_{t=1} \leftarrow x$

$$\begin{bmatrix} u_0 \\ \nabla_t \log p(u_0) \end{bmatrix} \leftarrow \begin{bmatrix} u_1 \\ 0 \end{bmatrix} + \int_1^0 \begin{bmatrix} v_t^\theta(\phi_t(u)) \\ -\nabla \cdot v_t^\theta(\phi_t(u)) \end{bmatrix} dt$$

$\phi_0(x) = x_{t=0} \sim q_0$

$$\begin{bmatrix} x_1 \\ \log p(x_1) \end{bmatrix} \leftarrow \begin{bmatrix} x_0 \\ \log p_0(x_0) \end{bmatrix} + \int_0^1 \begin{bmatrix} v_t^\theta(\phi_t(x)) \\ -\nabla \cdot v_t^\theta(\phi_t(x)) \end{bmatrix} dt$$

$\alpha \leftarrow \min\left\{1, \frac{\pi(x_1)p_0(u_0)\exp(-\nabla_t \log p(u_0))}{\exp(\log p(x_1))\pi(u_1)}\right\}$

With probability α make $x' \leftarrow x_1$ else $x' \leftarrow x$

Algorithm 16 Flow-informed Conditional Importance Sampling

Input: initial x, target density π, vector field v_t^θ, reference density q_0, flow parameters θ, number of importance samples K.

Output: x'

$\phi_1(u) = u_{t=1} \leftarrow x$

$$\begin{bmatrix} u_0 \\ \nabla_t \log p(u_0) \end{bmatrix} \leftarrow \begin{bmatrix} u_1 \\ 0 \end{bmatrix} + \int_1^0 \begin{bmatrix} v_t^\theta(\phi_t(u)) \\ -\nabla \cdot v_t^\theta(\phi_t(u)) \end{bmatrix} dt$$

$w_0 \leftarrow \frac{\pi(u_1)}{p_0(u_0)\exp(-\nabla_t \log p(u_0))}$

$x^{(0)} \leftarrow x$

for $k = 1 : K$ **do**

  $\phi_0(x) = x_{t=0} \sim q_0$

  $$\begin{bmatrix} x_1 \\ \log p(x_1) \end{bmatrix} \leftarrow \begin{bmatrix} x_0 \\ \log p_0(x_0) \end{bmatrix} + \int_0^1 \begin{bmatrix} v_t^\theta(\phi_t(x)) \\ -\nabla \cdot v_t^\theta(\phi_t(x)) \end{bmatrix} dt$$

  $w_k \leftarrow \frac{\pi(x_1)}{\exp(\log p(x_1))}$

  $x^{(k)} \leftarrow x_1$

end for

Choose k' with probability $P(k' = k) \propto w_k$, then make $x' \leftarrow x^{(k')}$

5.3.4 Adaptively-Tempered Markovian Flow Matching

If the geometry of the target posterior distribution is complicated, e.g. multimodal, then it can be challenging to learn a CNF that easily maps between a simple reference distribution p_0 and the target posterior distribution π. In this section, we consider a sequence of smooth transitions between the reference and target distributions using the annealed target distributions (5.7). The annealed targets act as intermediary steps within the flow-informed MCMC scheme.

Using adaptively tempered SMC, N particles are initialized from the prior distribution π_0, and the likelihood is gradually added by increasing the temperature β_t at every iteration, $t = 1, \ldots, T$. The temperature β_t is optimized using the effective sample size (5.8).

Our proposed MCMC scheme combines Markov transition kernels, which act globally P and locally Q on the target distribution. The Markov kernel P, outlined in Algorithm 14, generates samples from the reference space using the random-walk sampler and pushes these through to the target space via the flow (5.10), where the vector field v_t^θ is a neural network parameterized by θ. The Markov transition kernel Q, uses the local gradient of the target distribution within a Metropolis-adjusted Langevin algorithm (MALA):

$$\begin{aligned} Q(\mathrm{d}y|x,\pi) &= \alpha(x,y)q(\mathrm{d}y|x) + (1-b(x))\delta_x(\mathrm{d}y), \\ q(y|x) &\propto \exp\left(-\frac{1}{4\tau}\|y - x - \tau\nabla\log\pi(x)\|^2\right), \end{aligned} \tag{5.23}$$

where $\alpha(x,y)$ and $b(x)$ as in (5.4) and (5.5) respectively.

The overall MCMC scheme, which combines Markov kernels P and Q is outlined in Algorithm 17. The balance between local (Q) and global (P) MCMC steps is controlled by the hyperparameter t_Q, which determines the number of local steps taken per global step.

The quality of the flow-informed random walk kernel (P) is determined by the neural network vector field v_t^θ. The parameters of the neural network θ are updated by minimizing the flow matching loss (5.16) using stochastic gradient descent, with

learning rate ϵ. The expectation in the flow matching loss is approximated using the N particles to produce a Monte Carlo approximation,

$$\mathrm{FM}(\theta; x^{(1:N)}, \sigma_{min}) = \frac{1}{N}\sum_{i=1}^{N} \|v_t^\theta(\psi_t(x_0^{(i)}|x^{(i)})) - u_t(\psi_t(x_0^{(i)}|x^{(i)})|x_i)\|_2^2, \quad x_0^{(i)} \overset{iid}{\sim} p_0,$$

$$\text{where} \quad \psi_t(x_0|x) = (1-(1-\sigma_{min})t)x_0 + tx,$$

$$\text{and} \quad u_t(y|x) = \frac{x-(1-\sigma_{min})y}{1-(1-\sigma_{min})t}.$$

The full MCMC algorithm is run for T iterations where the vector field learns the target distribution by gradually adding the likelihood until $\beta_T = 1$. The combination of Markov kernels P and Q trades off faster computations with faster convergence of samples. The adaptive scheme is summarized in Algorithm 17.

Algorithm 17 Markovian Flow Matching

Input: σ_{min}, θ_0, α, T, N, t_Q, $\varepsilon_{1:T}$.
Output: optimal flow network parameters $\theta_T \approx \theta^*$
$x_i \sim \pi_0$ for $i = 1, \ldots, N$
$\beta \leftarrow$ solve (5.8) for $0 < \beta' \leq 1$
for $t = 1 : T$ **do**
    if $\beta \neq 1$ **then**
        $\beta \leftarrow$ solve (5.8) for $\beta < \beta' \leq 1$
    end if
    if $t \mod t_Q + 1 = 0$ **then**
        Flow-informed Markov transition: (Alg. 14)
        $x_i \sim P(\cdot|x_i, \pi_0\mathcal{L}^\beta, \theta_{t-1})$ for $i = 1, \ldots, N$
    else
        Markov transition:
        $x_i \sim Q(\cdot|x_i, \pi_0\mathcal{L}^\beta)$ for $i = 1, \ldots, N$
    end if
    $\theta_t \leftarrow \theta_{t-1} + \varepsilon_t \nabla_\theta FM(\theta_{t-1}; x_{1:N}, \sigma_{min})$
end for

Particle systems allow the state's mutation to N computing cores running in parallel at each iteration. The particle generation can be run in parallel using modern vector-oriented libraries, and then each particle is used to approximate the loss and update the parameters. Thus, the speedup gained by using more than one core scales linearly with the number of cores as long as there are as many cores as there are particles.

5.3.5 Convergence

Under certain regularity conditions, Proposition 2 ensures that the parameter estimate θ_T as provided by Algorithm 17 converges to a local optimum of the FM objective θ^*. The assumptions guarantee convergence by balancing the step size (large enough to explore but small enough to stabilize); ensure stability, continuity, and integrability of the Markov transition kernel with respect to θ and the Markov chain dynamics; and control the smoothness and Lipschitz continuity of the gradient, crucial for ensuring convergence. The proof relies on the connection between the algorithm and the stability of an underlying ODE; establishing that the stochastic algorithm's iterates asymptotically track the ODE's stable dynamics and converges almost surely to the local minimizer.

Proposition 2. *Assume that C1–C6 hold. If θ_t, for $t > 0$, defined by Algorithm 17 is a bounded sequence and almost surely visits a compact subset of the domain of attraction of θ^* infinitely often, then $\theta_t \to \theta^*$, almost surely.*

Proof. Let θ^* be a minimiser of the FM objective (5.16). Consider the ODE

$$\frac{d}{dt}\vartheta_t(\theta) = \nabla FM(\vartheta_t(\theta); x) \tag{5.24}$$

and its solution $\vartheta_t(\theta)$ for $t \geq 0$. The minimiser θ^* is a stability point of this ODE, i.e. $\vartheta_t(\theta^*) = \theta^*$ for all $t \geq 0$. The domain of attraction of θ^* is a set Θ containing θ^* where for any $\theta \in \Theta$, $\vartheta_t(\theta) \in \Theta$ for all $t \geq 0$ and $\vartheta_t(\theta) \to \theta^*$. Let O be a compact subset of Θ and $\mathcal{X}$ be an open set in $\mathbb{R}^d$.

Consider the Markov transition kernel given by a cycle of t_Q repeated transitions of a MALA kernel and a flow-informed RWMH transition:

$$M(x'|x,\theta) = \int P(x'|x_{t_Q},\theta)Q(x_{t_Q}|x_{t_Q-1})\dots Q(x_1|x)dx_{t_Q}\dots dx_1, \tag{5.25}$$

which is $\pi-$invariant since both P and Q are $\pi-$invariant. Let $M^t(x'|x,\theta)$ be the repeated application of the Markov transition kernel:

$$M^t(x'|x,\theta) = \int\cdots\int M(x_1|x,\theta)M(x_2|x_1,\theta)\dots M(x'|x_{t-1},\theta)dx_{t-1}\dots dx_1. \tag{5.26}$$

Assume for a $q > 1$ sufficiently large:

C1 Assume that the step size sequence satisfies $\sum_{k=1}^{\infty}\epsilon_k = \infty$ and $\sum_{k=1}^{\infty}\epsilon_k^2 < \infty$.

C2 (Integrability) There exists a constant C_1 such that for any $\theta \in \Theta$, $x \in \mathcal{X}$ and $k \geq 1$,

$$\int(1+|x'|^q)M^k(x'|x,\theta)dx' \leq C_1(1+|x|^q). \tag{5.27}$$

C3 (Convergence of the Markov Chain) Let π be the unique invariant measure for M. For each $\theta \in \Theta$,

$$\limsup_{k\to\infty}\sup_{x\in\mathcal{X}}\frac{1}{1+|x|^q}\int(1+|x'|^q)|M^k(x'|x,\theta)-\pi(x')|dx' = 0. \tag{5.28}$$

C4 (Continuity in θ) There exists a constant C_2 such that for any $\theta,\theta' \in O$,

$$\left|\int(1+|x'|^q)(M^k(x'|x,\theta)-M^k(x'|x,\theta'))dx'\right| \leq C_2|\theta-\theta'|(1+|x|^q). \tag{5.29}$$

C5 (Continuity in x) There exists a constant C_3 such that for any $x_1,x_2 \in \mathcal{X}$,

$$\sup_{\theta\in\Theta}\left|\int(1+|x'|^{q+1})(M^k(x'|x_1,\theta)-M^k(x'|x_2,\theta))dx'\right| \leq C_3|x_1-x_2|(1+|x_1|^q+|x_2|^q). \tag{5.30}$$

C6 (Conditions on the Score Function) There exist positive constants p, K_1, K_2, K_3 and $v > 1/2$ such that for all $\theta,\theta' \in O$ and $x, x_1, x_2 \in \mathcal{X}$,

$$|\nabla_\theta \log[X_0^\theta]_\# p_0(x)| \leq K_1(1+|x|^{p+1}), \tag{5.31}$$

$$|\nabla_\theta \log[X_0^\theta]_\# p_0(x_1) - \nabla_\theta \log[X_0^\theta]_\# p_0(x_2)| \leq K_2|x_1-x_2|(1+|x_1|^p+|x_2|^p), \tag{5.32}$$

$$|\nabla_\theta \log[X_0^\theta]_\# p_0(x) - \nabla_\theta \log[X_0^{\theta'}]_\# p_0(x)| \leq K_3|\theta-\theta'|^v(1+|z|^{p+1}). \tag{5.33}$$

The constants C_1, C_2, C_3 and v may depend on the compact set O and the real number q. With the above assumptions, the result follows from Theorem 1 of Gu and Kong (1998) by making $\Pi_\theta = M(\cdot|\cdot, \theta)$, $H(\theta, x_1) = \nabla_\theta \mathbb{E}_{t,p(x|x_1)}||v_t^\theta(x) - u_t(x|x_1)||^2$. □

5.4 Related work

In the domain of training CNFs, Zhang and Y. Chen (2021) introduces the Path Integral Sampler, a methodology leveraging concepts from control theory to define a loss function. Vargas, Grathwohl, and Doucet (2023) employ diffusion processes to generate samples from complex distributions, offering an innovative perspective on CNF training. Tong et al. (2023) use MCMC samples in a more straightforward manner within FM for Bayesian posteriors. While they leverage MCMC samples naively, our method employs a more intricate integration of these samples, offering a nuanced distinction in handling Bayesian posterior estimations. Their approach, although effective in specific scenarios, might not capture the complexities our method addresses, particularly in the context of computational efficiency and accuracy in diverse data distributions. The experiments in Section 5.5 illustrate the improvement brought by our more intricate design, Tong et al. (2023)'s method is by construction equivalent to our method when $t_Q = T$, since it uses only local gradient updates to generate samples that are used to learn the flow.

Within the extensive literature on discrete normalizing flows for preconditioning the target posterior, our method draws inspiration from key developments. Parno and Y. M. Marzouk (2018) pioneered preconditioning the target posterior. Building on this, M. Hoffman, Sountsov, et al. (2019) incorporated discrete normalizing flows into the preconditioning process, and Cabezas and Nemeth (2023) used a similar approach with elliptical slice sampling. Naesseth, Lindsten, and D. Blei (2020) used an independent MH proposal from the learned flow, and Gabrié, Rotskoff, and Vanden-Eijnden (2022) introduced a hybrid approach, alternating between local

and global updates to enhance efficiency. Karamanis, Beutler, et al. (2022) further refined this concept by integrating SMC strategies to temper the target in their scheme. We've developed a unique approach by integrating MCMC samples within the FM framework, leveraging the flexibility of CNFs. This shift to continuous flows allows for greater flexibility in modelling, freeing us from the constraints of specific architectures typically associated with discrete flows.

5.5 Experiments

In this section, we evaluate the performance of MFM (Algorithm 17) on two synthetic and two real data examples. Our method is benchmarked against three relevant methods. The Denoising Diffusion Sampler (DDS; Vargas, Grathwohl, and Doucet, 2023) is a VI method which approximates the reversed diffusion process from a reference distribution to an extended target distribution by minimizing the KL divergence. Adaptive Monte Carlo with Normalizing Flows (NF-MCMC; Gabrié, Rotskoff, and Vanden-Eijnden, 2022) is an augmented MCMC scheme which uses a mixture of MALA and adaptive transition kernels learned using discrete NFs. Finally, Flow Annealed Importance Sampling Bootstrap (FAB; Midgley et al., 2022) is an augmented AIS scheme minimizing the mass-covering α-divergence with $\alpha = 2$.

For each experiment, all MALA kernels use the same step size, targeting an acceptance rate of close to 1 since we estimate expectations using the current ensemble of particles, rather than a single long chain. Following Zhang and Y. Chen (2021), we parameterize the vector field as

$$v_t^\theta(x) = \mathrm{NN}(x, t; \theta_1) + \mathrm{NN}(t; \theta_2) \times \nabla \log \pi(x), \tag{5.34}$$

where the neural networks are standard MLPs with 2 hidden layers, using a Fourier feature augmentation for t (Tancik et al., 2020). This architecture is also used by DDS (Vargas, Grathwohl, and Doucet, 2023, Section 4). Meanwhile, FAB and NF-MCMC use rational quadratic splines (Durkan et al., 2019). Flows are trained using Adam (Diederik P Kingma and Ba, 2014) with a linear decay schedule terminating

at $\varepsilon_K = 0$. We report results for all methods averaged over 10 independent runs with varying random seeds. The code to reproduce the experiments is provided at albcab/mfm.

Code for the numerical experiments is written in Python with array computations handled by JAX (Bradbury et al., 2018). The implementation of relevant methods for comparison is sourced from open source repositories: DDS using franciscovargas/denoising_diffusion_samplers, NF-MCMC using kazewong/flowMC, and FAB using lollcat/fab-jax. All experiments are run on an NVidia V100 GPU with 32GB of memory. In the following subsections, we will give more details on the modelling and hyperparameter choices for each experiment, along with additional results.

5.5.1 4-mode Gaussian mixture

Our first example is a mixture of four Gaussians, evenly spaced and equally weighted, in two-dimensional space. The four mixture components have means $(8, 8)$, $(-8, 8)$, $(8, -8)$, $(-8, -8)$, and all have identity covariance. This ensures that the modes are sufficiently separated to mean that jumping between modes requires trajectories over sets with close to null probability. Given the synthetic nature of the problem, we can measure approximation quality using the Maximum Mean Discrepancy (MMD) (e.g., Gretton et al., 2012). We can also include, as a benchmark, the results for an approximation learned using FM with true target samples. Diagnostics for all models are presented in Table 5.1, and learned flow samples in Fig. 5.2.

For this experiment, all methods use $N = 128$ parallel chains for training and 128 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.2, and methods with splines use 4 coupling layers with 8 bins and range limited to $[-16, 16]$. We present results for $K = 5 \cdot 10^3$ learning iterations for MFM, and $K = 10^3$ iterations for the other algorithms, since this renders the total computational cost of all algorithms somewhat comparable.

In this experiment, only our method (Fig. 5.2a) and DDS (Fig. 5.2c) learn the

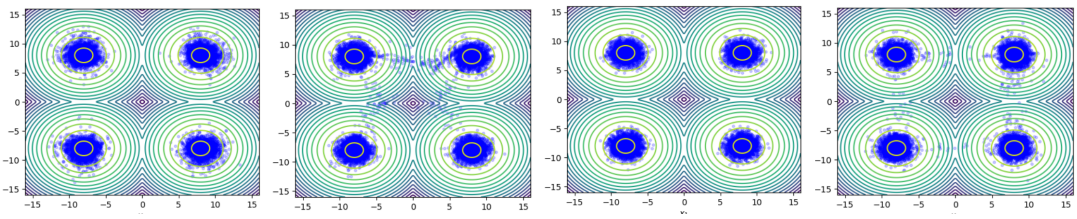
	4-mode		16-mode	
	MMD	seconds	MMD	seconds
FM w/ π samples	5.45e-4±3.51e-4	20.5 ± 0.41	2.16e-3±3.95e-4	22.8 ± 0.22
MFM $k_Q = 10^2$	1.39e-3±8.22e-4	38.1 ± 1.29	3.57e-3±7.68e-4	38.5 ± 1.75
DDS	1.76e-4±2.32e-4	$114. \pm 0.68$	1.02e-1±4.10e-2	$115. \pm 0.64$
NF-MCMC	5.85e-3±3.91e-3	72.0 ± 11.7	8.05e-3±1.42e-2	67.0 ± 12.3
FAB	2.69e-4±2.06e-4	$101. \pm 3.24$	1.51e-3±1.06e-3	$102. \pm 4.32$

Table 5.1: Diagnostics for the two synthetic examples. MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

fully separated modes, reflecting the greater expressivity of CNFs in comparison to the discrete NFs used in, e.g., NF-MCMC (Fig. 5.2d). It is worth noting that DDS provides a closer approximation to the real target than MFM and, notably, even FM trained using true target samples (top row). Given that both methods use the same network architecture but a different learning objective, this suggests a potential limitation with the FM objective, at least when using this network architecture. This being said, MFM is notably more efficient than DDS (as well as the other methods) in terms of total computation time. While this is not a critical consideration in this synthetic, low-dimensional setting, it is a significant advantage of MFM in higher-dimensional settings involving real data (e.g., Section 5.5.3 and Section 5.5.4).

5.5.2 16-mode Gaussian mixture

The second experiment is a mixture of bivariate Gaussians with 16 mixture components. This is a modification of the 4-mode example, with contrasting qualities that illustrate other characteristics of each of the presented methods. In this case, the modes are evenly distributed on $[-16, 16]^2$, with random log-normal variances.



(a) MFM $k_Q = 10^2$ (b) FAB (c) DDS (d) NF-MCMC

Figure 5.2: Comparison between MFM, FAB, DDS, and NF-MCMC. Samples from the target density for the 4-mode Gaussian mixture example.

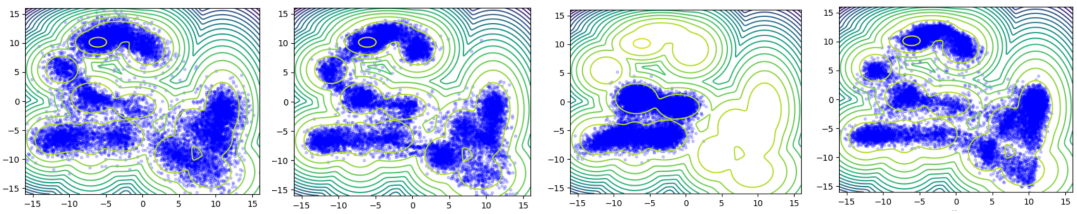
The number of modes reduces the size of sets of (near) null probability between the modes, making jumping between them easier. To increase the difficulty of this model, all methods are initialized on a concentrated region of the sampling space. Diagnostics are presented in Table 5.1 and learned flow samples in Fig. 5.3.

Like the 4-mode example, all methods use $N = 128$ parallel chains for training and 128 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.2, and methods with splines use 4 coupling layers with 8 bins and range limited to $[-16, 16]$. We present results for $K = 5 \cdot 10^3$ learning iterations for MFM and $K = 10^3$ iterations for all other algorithms, which yields a more comparable total computation time.

In this example, DDS collapses to the modes closest to the initial positions while our method captures the whole target. Since the modes are no longer separated by areas of near-zero probability, the discrete NF methods are now able to accurately capture the target density. In this case, DDS marginally outperforms MFM as measured by the MMD, but this slight improvement in performance comes at the cost of a much higher run-time.

5.5.3 Field system

Our first real-world example considers the Allen–Cahn equation, used as a benchmark in Gabrié, Rotskoff, and Vanden-Eijnden (2022). The Allen–Cahn equation is defined in terms of a random field $\phi : [0, 1] \to \mathbb{R}$ satisfying the following stochastic partial



(a) MFM $k_Q = 10^2$ (b) FAB (c) DDS (d) NF-MCMC

Figure 5.3: Comparison between MFM, FAB, DDS, and NF-MCMC. Samples from the target density for the 16-mode Gaussian mixture example.

differential equation:

$$\frac{\partial \phi}{\partial t} = a\frac{\partial^2 \phi}{\partial s^2} + a^{-1}(\phi - \phi^3) + \sqrt{2\beta^{-1}}\eta(t, s), \tag{5.35}$$

where $a > 0$ is a parameter, β is the inverse temperature, $s \in [0, 1]$ denotes the spatial variable, and η is spatiotemporal white noise. The imposition of Dirichlet boundary conditions ensures $\phi(s = 0) = \phi(s = 1) = 0$.

The associated Hamiltonian, reflecting a spatial coupling term penalizing changes in ϕ, takes the form:

$$U_*[\phi] = \beta \int_0^1 \left[\frac{a}{2} \left(\frac{\partial \phi}{\partial s} \right)^2 + \frac{1}{4a} \left(1 - \phi^2(s) \right)^2 \right] ds. \tag{5.36}$$

At low temperatures, this coupling induces alignment of the field in either the positive or negative direction, leading to two global minima, ϕ_+ and ϕ_-, with typical values of ± 1.

This fundamental reaction-diffusion equation is central to the study of phase transitions in condensed matter systems. Incorporating random forcing terms or thermal fluctuations allows for a stochastic treatment of the dynamics, capturing the inherent randomness and uncertainties in physical systems. This system leads to a discretized target density of the form

$$\log \pi(x) = -\beta \left(\frac{a}{2\Delta s} \sum_{i=1}^{d+1} (x_i - x_{i-1})^2 + \frac{\Delta s}{4a} \sum_{i=1}^{d} (1 - x_i^2)^2 \right), \tag{5.37}$$

where $\Delta s = \frac{1}{d}$, and with boundary conditions $x_0 = x_{d+1} = 0$. In our experiments, we set the dimensionality $d = 64$. Meanwhile, parameter values are chosen to ensure bimodality at $x = \pm 1$: $a = 0.1$ and inverse temperature $\beta = 20$.

The bimodality induced by the two global minima complicates mixing when using traditional MCMC updates. Learning the global geometry of the target and using that information to propose transitions facilitates movement between modes. Unlike previous work e.g., Gabrié, Rotskoff, and Vanden-Eijnden, 2022, we deliberately choose not to employ an informed base measure. Instead, we opt for a standard Gaussian with no additional information, making the problem significantly more challenging. This choice illustrates the robustness of our approach.

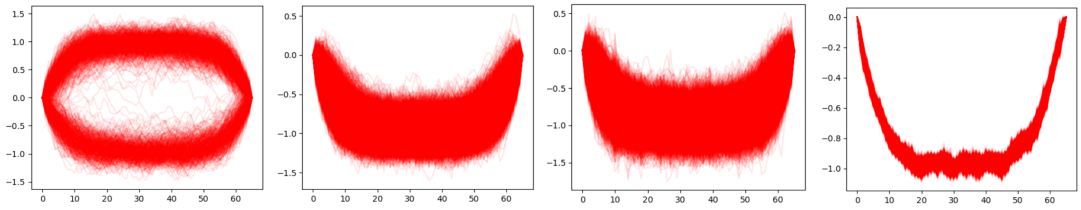
Numerical diagnostics for each method are presented in Table 5.2. In this case, we use the Kernelized Stein Discrepancy (KSD) as a measure of sample quality e.g., Liu, Lee, and M. Jordan, 2016; Gorham and Mackey, 2017. While this is not a perfect metric, it does allow us to qualitatively compare the different methods considered.

For this example, all methods use $N = 1024$ parallel chains for training and 256 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.0001, and methods with splines use 8 coupling layers with 8 bins and range limited to $[-5, 5]$. We present results for $k_Q = 10^2$ for MFM.

	Field system			Log-Gaussian Cox		
$K = 10^4$	KSD U-stat.	KSD V-stat.	seconds	KSD U-stat.	KSD V-stat.	seconds
MFM	2.17 ± 2.19	20.4 ± 2.30	146 ± 1.93	1.05e-1±0.05	31.2 ± 0.16	910 ± 47.1
DDS	15.2 ± 35.9	18.0 ± 36.9	2400 ± 8.65	7.59e-2±0.02	24.7 ± 0.08	3260 ± 8.41
NF-MCMC	548 ± 325	549 ± 325	2000 ± 15.6	11.8 ± 7.55	89.0 ± 238	215 ± 46.4
FAB	0.14 ± 0.42	1.78 ± 0.42	3880 ± 7.19	1.55e-1±0.06	52.3 ± 2.02	1040 ± 2.78

Table 5.2: Diagnostics for the two real data examples. KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

In this case, the tempering mechanism of our method is crucial for ensuring that the learned flow does not collapse on one of the modes and instead explores both global minima. This is confirmed when plotting the samples generated in the grid in Fig. 5.4. This experiment demonstrates the capability of our method to capture complex multi-modal densities, even without an informed base measure, at a *significantly* lower computational cost (e.g., 10-25x faster) than competing methods. It is worth noting that MFM (and the other two benchmarks) significantly outperformed NF-MCMC in this example, despite the similarities between MFM and NF-MCMC. While we tested various hyperparameter configurations for NF-MCMC, we were not able to find a setting that achieved comparable results.



(a) MFM $k_Q = 10^2$ (b) FAB (c) DDS (d) NF-MCMC

Figure 5.4: Comparison between MFM, FAB, DDS, and NF-MCMC. Representative samples from the target density for the Field system example.

5.5.4 Log-Gaussian Cox point process

Bayesian inference for high-dimensional spatial models is known to be challenging. One such model is the log-Gaussian Cox point process, which here we use to model the locations of 126 Scots pine saplings in a natural forest in Finland.

The original 10×10 square meter plot is standardized to the unit square, as depicted in Fig. 5.5. We discretize the unit square $[0, 1]^2$ into a $M = 40 \times 40$ regular grid. The latent intensity process $\Lambda = \{\Lambda_m\}_{m \in M}$ is specified as $\Lambda_m = \exp(X_m)$, where $X = \{X_m\}_{m \in M}$ is a Gaussian process with a constant mean $\mu_0 \in \mathbb{R}$ and exponential covariance function $\Sigma_0(m, n) = \sigma^2 \exp\left(-|m - n|/(30\beta)\right)$ for $m, n \in M$, i.e. $X \sim \mathcal{N}(\mu_0 1_d, \Sigma_0)$ for $1_d = [1, \ldots, 1]^T \in \mathbb{R}^d$ with dimension $d = 900$. The chosen

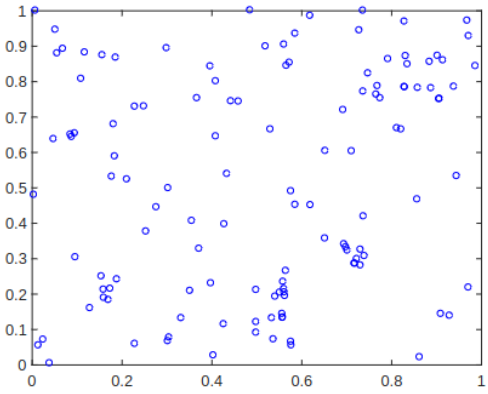


Figure 5.5: Visualization of the standardized 10×10 square meter plot and the locations of 126 Scots pine saplings in Finland.

parameter values are $\sigma^2 = 1.91$, $\beta = 1/33$, and $\mu_0 = \log(126) - \sigma^2/2$, estimated by Møller, Syversveen, and Waagepetersen (1998). The number of points in each grid cell $Y = \{Y_m\}_{m \in M} \in \mathbb{N}^{40 \times 40}$ are modelled as conditionally independent and Poisson distributed with means $a\Lambda_m$,

$$\mathcal{L}(Y|X) = \prod_{m \in [1:40]^2} \exp(x_m y_m - a \exp(x_m)), \tag{5.38}$$

where $a = 1/40^2$ represents the area of each grid cell.

For this example, all methods use $N = 128$ parallel chains for training and 1024 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.01, and methods with splines use 8 coupling layers with 8 bins and range limited to $[-10, 10]$. We present results for $k_Q = 10^3$ for MFM. We were unable to run NF-MCMC and FAB for $K = 10^4$ iterations because of memory issues; instead, we present results for $K = 10^3$ iterations only for the models using discrete normalizing flows.

In Table 5.2, we report diagnostics for each algorithm. In this case, the lack of multimodality in the target makes it a good fit for non-tempered schemes. Similar to the previous example, NF-MCMC is unable to obtain an accurate approximation to the target distribution. We suspect that this may be a result of non-convergence: due to memory issues, it was not possible to run NF-MCMC (or FAB) for more than

$K = 10^3$ iterations. This also explains the (relatively) smaller run times of these algorithms in this example. By a small margin, DDS provides the best approximation of the target, slightly outperforming MFM and FAB. Meanwhile, MFM provides a good approximation to the target at a lower computational cost with respect to its counterparts.

5.6 Discussion

We introduce a new computational approach to Bayesian inference that uses sampling methods to define the probability path for the FM objective. We also develop an adaptive tempering mechanism that discovers multiple modes in the target distribution. This innovation addresses a critical challenge in Bayesian inference involving multimodal distributions. Furthermore, we have established our method's convergence to a local optimum of the FM objective.

The experiments in Section 5.5 showcase our method's ability to accelerate convergence and improve the fitting of CNFs in various synthetic and real-world examples. These examples illustrate the method's superiority in terms of computational efficiency and accuracy in complex data distributions. The comparative analysis with relevant methods underscores the enhanced performance of our approach.

A promising avenue for future research lies in developing tailored CNFs specifically designed for certain types of posterior distributions. This approach would go beyond the current practice of including the log-posterior gradient and exploit unique characteristics intrinsic to each model when constructing the flow. Such customization would be particularly valuable for models with complex or highly structured data distributions. Investigating the application of our approach to a broader range of fields, such as bioinformatics, finance, and environmental science, could also yield significant insights, especially in scenarios with large-scale data or intricate model structures.

Chapter 6

Robust Bayesian Nonparametric Variable Selection for Linear Regression

Spike-and-slab and horseshoe regression are arguably the most popular Bayesian variable selection approaches for linear regression models. However, their performance can deteriorate if outliers and heteroskedasticity are present in the data, which are common features in many real-world statistics and machine learning applications. This work proposes a Bayesian nonparametric approach to linear regression that performs variable selection while accounting for outliers and heteroskedasticity. Our proposed model is an instance of a Dirichlet process scale mixture model with the advantage that we can derive the full conditional distributions of all parameters in closed form, hence producing an efficient Gibbs sampler for posterior inference. Moreover, we present how to extend the model to account for heavy-tailed response variables. The model's performance is tested against competing algorithms on synthetic and real-world datasets.

6.1 Introduction

Bayesian variable selection is a popular tool in statistics and machine learning that can be used for feature selection in linear regression models (Schoot et al., 2021). The two most popular models are arguably the spike-and-slab and horseshoe regression models. Both models rely on choosing a sparsity-inducing prior over the regression coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$. In the spike-and-slab model, the prior is a mixture between a point mass at zero and a diffuse prior, while in the horseshoe model, a continuous prior balances the local and global shrinkage (see Section 6.2). Compared to the estimator for the regression coefficients in standard linear regression, the posterior distribution under these priors produces a shrinkage effect toward zero in the posterior point estimates. This is especially useful in the $p >> n$ regime, where the number of attributes p can be much larger than the number of observations n.

This work proposes extensions of the spike-and-slab and horseshoe regression models to deal with heteroskedasticity and outliers in the data. Specifically, we propose a Bayesian nonparametric model in which each observation has its specific variance σ_i^2, sampled from an unknown distribution P. The distribution P is a Dirichlet process prior and the resulting model is a Dirichlet process scale mixture model. Due to the discreteness of the Dirichlet process, the resulting vector of variances $(\sigma_1^2, \ldots, \sigma_n^2)$ will be partitioned into groups, hence also producing a corresponding clustering of the observations, in which observations belonging to the same group will have the same conditional variance. Moreover, some observations may be allocated to a group with a much larger variance than the others, allowing for outliers in the data.

The key features of our proposed model are:

- **Parsimonious regression construction:** Our nonparametric method performs feature selection, with the posterior concentrating on the most relevant prediction coefficients. However, unlike variable coefficient models, there is no increase in the degrees of freedom associated with the number of regression

parameters (or smoothness of parameters to control, c.f. functional regression).

- **Interpretable model structure:** The proposed linear regression model changes the structure of the marginal variance components while retaining the highly interpretable properties of a standard sparse regression formulation.
- **Posterior credible intervals:** A fully Bayesian approach gives credible intervals for the regression coefficients. We propose several ways to perform posterior inference selection using these intervals, which provide uncertainty quantification for decision-makers working with high-dimensional data.
- **Efficient inference:** Under our model, full conditional distributions of all parameters can be derived in *closed form*, hence producing an efficient Gibbs algorithm that gives a tuning-free and rejection-free Markov chain Monte Carlo (MCMC) sampler.

The rest of the chapter is organized as follows. Section 6.2 briefly reviews the spike-and-slab and horseshoe regression models. Section 6.3 introduces the Bayesian nonparametric framework, our Dirichlet process mixture model for linear regression, and details of our MCMC sampler. In Subsection 6.3.3.2, we provide an extension to our model to account for heavy-tailed response variables. Finally, Section 6.4 compares our proposed model against popular alternative models on synthetic data and real-world data examples.

6.2 Bayesian Variable Selection for Linear Regression

The Bayesian approach to variable selection for linear regression models is to introduce sparsity-inducing priors on the regression coefficients. The two most popular approaches in the literature, which we shall review here, are the discrete mixture priors known as the spike-and-slab (Mitchell and Beauchamp, 1988; George and McCulloch, 1993), and the continuous shrinkage priors, most notably the horseshoe prior (Carvalho, Polson, and Scott, 2010).

6.2.1 Spike-and-slab priors

The original spike-and-slab model was initially proposed by Mitchell & Beauchamp (1988) and significantly developed by Madigan & Raftery (1994) and George & McCulloch (1997). The final adjustments to the model were completed by Ishwaran & Rao (2005) in what they refer to as the *stochastic variable selection* model. The spike-and-slab prior is intuitively simple and consists of two components. The *spike* is a delta function centered at zero indicating $\beta_j \approx 0$, and the *slab* gives probability mass to non-zero coefficients.

In our regression framework, we have data $\mathbf{y} \in \mathbb{R}^n$ that is explained by a matrix of attributes $\mathbf{X} \in \mathbb{R}^{n \times p}$ and coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$. Assuming a spike-and-slab prior for $\boldsymbol{\beta}$, we have the following model,

$$\begin{aligned} y_i|\boldsymbol{x}_i, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2), \quad i = 1, \dots, n \quad \text{with} \quad \sigma^2|b_1, b_2 \sim \text{IG}(b_1, b_2) \quad \text{and} \\ \beta_j|\eta_j, \tau_j^2 \sim \mathcal{N}(0, \eta_j \tau_j^2), \quad j = 1, \dots, p \quad \text{with} \quad \tau_j^2|a_1, a_2 \sim \text{IG}(a_1, a_2) \quad \text{and} \\ \eta_j|\nu_0, \omega \sim (1-\omega)\delta_{\nu_0}(\cdot) + \omega\delta_1(\cdot), \quad j = 1, \dots, p \quad \text{with} \quad \omega \sim \text{Unif}[0, 1] \end{aligned} \tag{6.1}$$

The likelihood for the data assumes a standard Gaussian regression model, and the prior for $\boldsymbol{\beta}$ is a scale mixture of Gaussians. The variable η_j is a latent indicator and $\delta_{v_0}(\cdot)$ denotes a point mass at v_0, where v_0 is a value chosen close to 0, and thus the variance of the prior on $\boldsymbol{\beta}$ is either almost zero or a broad Gaussian distribution.

6.2.2 Horseshoe priors

Spike-and-slab priors are intuitively appealing, but in practice, their discrete nature makes posterior inference computationally difficult. A popular alternative perspective on sparse Bayesian regression is given by the horseshoe prior construction (Carvalho, Polson, and Scott, 2009; Carvalho, Polson, and Scott, 2010, see). The horseshoe prior is a continuous shrinkage prior, which makes posterior computation more efficient when using gradient-based MCMC sampling tools such as STAN (Carpenter et al., 2017).

$$
\begin{aligned}
y_i|\boldsymbol{x}_i, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2), \quad i = 1, \ldots, n \quad \text{with} \quad \sigma^2|b_1, b_2 \sim \text{IG}(b_1, b_2) \quad \text{and} \\
\beta_j|\lambda_j^2, \tau^2 \sim \mathcal{N}(0, \lambda_j^2\tau^2), \quad j = 1, \ldots, p \quad \text{with} \quad \tau \sim \mathcal{C}^+(0, 1) \quad \text{and} \\
\lambda_j \sim \mathcal{C}^+(0, 1), \quad j = 1, \ldots, p
\end{aligned} \tag{6.2}
$$

Under the horseshoe regression model, the parameters λ_j and τ are the *local* and *global* shrinkage parameters, respectively. Following Carvalho, Polson, and Scott (2010), we choose half-Cauchy priors for λ_j and τ. The intuition behind the horseshoe prior is that the global parameter τ will force the regression coefficients towards zero. In contrast, the heavy tails of the half-Cauchy prior for the local shrinkage parameters λ_j will allow for non-zero $\boldsymbol{\beta}$ coefficients.

The horseshoe model used in our experiments follows the form originally proposed by Carvalho, Polson, and Scott (2010). A standard Gibbs sampling approach on this model parametrization is difficult to implement due to the non-conjugate posterior form of the shrinkage parameters in the model. In our implementation, we introduce auxiliary variables that lead to conjugate full conditionals for all parameters as suggested by Makalic and Schmidt (2015a) and allow for a straightforward implementation of Gibbs sampling.

The parameterization given by Makalic and Schmidt (2015a) uses an inverse gamma scale mixture representation of a random variable with a half-Cauchy distribution. It can be shown that $X \sim \mathcal{C}^+(0, A)$ if $X^2|a \sim \text{IG}(1/2, 1/a)$ and $a \sim \text{IG}(1/2, 1/A^2)$. Using this decomposition leads to the reparametrized horseshoe model

$$
\begin{aligned}
y_i|\boldsymbol{x}_i, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2), \quad i = 1, \ldots, n \quad \text{with} \quad \sigma^2|b_1, b_2 \sim \text{IG}(b_1, b_2) \quad \text{and} \\
\beta_j|\lambda_j^2, \tau^2 \sim \mathcal{N}(0, \lambda_j^2\tau^2), \quad j = 1, \ldots, p \quad \text{with} \quad \tau^2|\xi \sim \text{IG}(1/2, 1/\xi) \quad \text{and} \\
\lambda_j^2|\nu_j \sim \text{IG}(1/2, 1/\nu_j), \quad j = 1, \ldots, p \quad \text{with} \quad \nu_1, \ldots, \nu_p, \xi \sim \text{IG}(1/2, 1)
\end{aligned} \tag{6.3}
$$

for which the sampling schemes presented in the next section easily follow.

6.3 Nonparametric Stochastic Variable Selection

6.3.1 Background to Dirichlet processes

In Bayesian nonparametric statistics, unknown parameters are infinite-dimensional and cannot be parametrized by a subset of Euclidean space. The most common examples of nonparametric problems are the estimation of density functions, distribution functions, and nonparametric regression (Hjort et al., 2010).

The most popular nonparametric distribution function is the Dirichlet process, introduced in Ferguson (1973). In this setting, P has a *Dirichlet process* distribution with base measure P_0 and concentration parameter α, denoted $P \sim DP(\alpha, P_0)$. For every measurable partition $(A_1, \ldots, A_k)$, the random vector $(P(A_1), \ldots, P(A_k))$ has a Dirichlet distribution on the k-dimensional simplex with parameters $(\alpha P_0(A_1), \ldots, \alpha P_0(A_k))$. The base measure P_0 is the mean of the prior (i.e. $\mathbb{E}(P) = P_0$), while the concentration parameter α regulates prior uncertainty around P_0. A large α value implies a strong belief in the prior.

There are many different representations of the Dirichlet process (see Ghosal and Van der Vaart (2017), Chapter 4). For this research, we shall utilize the *Chinese restaurant process representation*, which gives the marginal distribution of the data when the unknown distribution P has been marginalized out. Specifically, if $\theta_{1:n}$ is an i.i.d. sample from P, $\theta_i | P \overset{iid}{\sim} P$, and $P \sim DP(\alpha, P_0)$, then the marginal distribution of $\theta_{1:n}$, when P is integrated out, is

$$\pi(\theta_{1:n}|\alpha, P_0) = \int \prod_{i=1}^{n} \mathcal{P}(\theta_i) \mathrm{DP}(d\mathcal{P}; \alpha, P_0),$$

which can be described by the marginal of θ_1 and the conditional distributions of $\theta_i|\theta_{1:i-1}$ for $i = 2, \ldots, n$, $\pi(\theta_{1:n}|\alpha, P_0) = \pi(\theta_1|\alpha, P_0) \prod_{i=2}^{n} \pi(\theta_i|\theta_{1:i-1}, \alpha, P_0)$, where $\pi(\theta_1|\alpha, P_0) = P_0$ and

$$\theta_i|\theta_{1:i-1}, \alpha, P_0 \sim \frac{1}{i-1+\alpha} \sum_{i=1}^{n} \delta_{\theta_i} + \frac{\alpha}{i-1+\alpha} P_0.$$

Thus, a sample $\theta_{1:n}$ from $\pi(\theta_{1:n}|\alpha, P_0)$ will display ties with positive probability. The

parameter α regulates the number of clusters in $\theta_{1:n}$. All else being equal, larger α will lead to more distinct values within $\theta_{1:n}$.

6.3.2 Dirichlet process mixture model

The Dirichlet process mixture model (DPM) (Lo, 1984) assumes each observation y_i is sampled from a countable mixture model. The likelihood of each mixture component, $K(y_i; \theta_k^*)$, is parametrized by an unknown parameter vector θ_k^*. In the following, K will be a Gaussian kernel, and θ_k^* is the group-specific variance. Both the mixture parameters θ^* and the mixture weights w_k can be encoded into a random measure $P = \sum_{k \geq 1} w_k \delta_{\theta_k^*}$ which is endowed with a Dirichlet process prior, hence producing the following mixture model for $i = 1, \ldots, n$

$$
\begin{aligned}
y_i | P &\sim \sum_{k=1}^{\infty} w_k K(y_i; \theta_k^*) = \int K(y_i; \theta) P(d\theta) \\
P &\sim DP(\alpha, P_0).
\end{aligned}
$$

By introducing latent class variables, $\{\theta_i\}_{i=1}^n$ s.t. $\mathbb{P}(\theta_i = \theta_k^*) = w_k$, the DPM model is equivalent to the latent variable mixture model

$$
\begin{aligned}
y_i | \theta_i &\sim K(y_i; \theta_i), \quad i = 1, \ldots, n \\
\theta_i | P &\overset{iid}{\sim} P, \quad i = 1, \ldots, n \\
P &\sim DP(\alpha, P_0)
\end{aligned}
\tag{6.4}
$$

The DPM model has been used in various statistics and machine learning applications for density estimation and clustering. In density estimation, the goal is to estimate the unknown density of the observations through a mixture model, while in clustering, the goal is to cluster observations into groups with a similar distribution. For the latter task, the discreteness property of the Dirichlet process is very convenient since, with positive probability, we will observe ties among the latent variables $\theta_{1:n}$. Two observations y_i and y_j having the same value of the latent variables θ_i and θ_j will be assigned to the same cluster and have the same distribution.

In Section 6.3.3 we utilize the properties of the Dirichlet process mixture model to propose a Bayesian nonparametric extension to the spike-and-slab and horseshoe regression models. We show that our new model can capture heteroskedasticity and outliers in the observations and capture clustering behavior in the variance structure. Under both DP variable selection models, the mixture component has a likelihood with a cluster-specific variance term, which is not fixed *apriori* but is learned from the data.

6.3.3 Dirichlet process variable selection

In both the spike-and-slab (6.1) and horseshoe (6.3) models, one assumes that each y_i has the same conditional variance σ^2. Under our nonparametric Dirichlet process model, we introduce an observation-dependent variance σ_i^2 for each data point. The vector $\boldsymbol{\sigma^2} := \sigma_{1:n}^2$ is assumed to be sampled from an unknown discrete distribution P sampled from a Dirichlet process. Specifically, we consider the following general hierarchical model,

$$\begin{aligned} y_i|\boldsymbol{x}_i, \boldsymbol{\beta}, \boldsymbol{\sigma}^2 \sim \mathcal{N}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma_i^2), \quad i = 1, \ldots, n \quad \text{with} \quad \boldsymbol{\beta} \sim \pi \quad \text{and} \\ \sigma_i^2|P \sim P(d\sigma_i^2), \quad i = 1, \ldots, n \quad \text{with} \quad P|\alpha \sim \text{DP}(\alpha, \text{IG}(b_1, b_2)) \\ \text{and} \alpha \sim \text{Gamma}(d_1, d_2) \end{aligned} \tag{6.5}$$

where the prior π for $\boldsymbol{\beta}$ either follows the spike-and-slab construction in lines 2-5 of eq. (6.1), or the horseshoe model in lines 2-4 of eq. (6.3). We refer to the general model in eq. (6.5) as the *Dirichlet process variable selection model* and recognize that the spike-and-slab and horseshoe models are special cases, which we denote as *Dirichlet process spike-and-slab* (DPSS) and *Dirichlet process horseshoe* (DPHS), respectively.

In this model, we are assuming a variance σ_i for each observation y_i, where the vector of variances $\sigma_1, \ldots, \sigma_n$ is assumed to be conditionally i.i.d. from an unknown distribution P. Specifically, we use a Dirichlet process as a nonparametric prior for P, centered at $\text{IG}(b_1, b_2)$ with concentration parameter α. From the properties

of the DP, P will almost surely be a discrete distribution. This implies that, with positive probability, we will observe *ties* among $\sigma_1, \ldots, \sigma_n$. In other words, some σ_i will take the same value. We denote by $\sigma_1^*, \ldots, \sigma_{K_n}^*$ the K_n distinct values assumed by $\sigma_1, \ldots, \sigma_n$. We will then have K_n clusters among our observations $\{y_i\}_{i=1}^n$, where observations within a cluster have the same variance, but different clusters have different variances. Specifically, if $\sigma_i = \sigma_j$, then y_i and y_j will be in the same cluster and have the same conditional variance but with possibly different means, depending on their attributes.

6.3.3.1 Posterior inference

Under the DPSS and DPHS models, posterior inference can be carried out efficiently using a Markov chain Monte Carlo sampler. We propose a general Gibbs sampler to sample $\sigma_{1:n}^2$ (see Algorithm 18) based on the algorithm by Escobar and West (1995), where at each iteration, we first resample a classification vector $c_{1:n}$ assigning each σ_i^2 to a block in the partition and then resample the distinct values $\sigma_{1:k}^{2*}$.

The partition generated by the variance value assigned to each observation is distributed, *a priori*, as a *Chinese Restaurant Process* (Aldous, 1985). This distribution can be understood intuitively as a Chinese restaurant serving an infinite amount of customers arriving in succession. Each customer must be seated on an unbounded number of tables, where the probability of seating a customer at an occupied table is proportional to the number of people already seated. Alternatively, the customer could be seated at a new table with probability proportional to the concentration parameter α. Assuming i.i.d. variances, we can assume, *a posteriori*, that each observation is the last customer to arrive and each table represents a cluster of variances (R. M. Neal, 2000). The customer/observation will be seated at an already occupied table or a new table with probability proportional to

$$\mathbb{P}[c_i = c \,|\, c_{-i}, y_{1:n}, \sigma_{1:K_n}^{2*}] \propto \begin{cases} n_{-i,c} N(y_i; \boldsymbol{x}_i^T \boldsymbol{\beta}, \sigma_c^{2*}) \text{ for } 1 \leq c \leq K^- \\ \\ \alpha g(y_i; \boldsymbol{x}_i, \boldsymbol{\beta}, b_1, b_2) \text{ for } c = K^- + 1, \end{cases} \tag{6.6}$$

Algorithm 18 DP variable selection Gibbs sampler

for t in 1: number of iterations **do**

for i in 1:n **do**

Sample classification value c_i with probability (6.6).

end for

for k in 1:K_n **do**

Sample σ_k^{2*}

$$\sigma_k^{2*} \sim \text{IG}\left(b_1 + \frac{n_k}{2},\ b_2 + \frac{1}{2}\sum_{i:C_k}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2\right),$$

where $C_k = \{\sigma_i^2 \mid c_i = k\}$

end for

if Spike-and-Slab **then**

Sample $\boldsymbol{\beta}$, τ_j, η_j and ω using Algorithm 19

else if Horseshoe **then**

Sample $\boldsymbol{\beta}$, λ_j and τ using Algorithm 20

end if

Sample α, by sampling the latent variables

$$\psi|\alpha, K_n \sim \text{Beta}(\alpha+1, n) \quad \text{then} \quad a|\psi, K_n \sim \text{Bernoulli}\left(\frac{w2}{w1+w2}\right)$$
$$\text{and} \quad \alpha|a, \psi, K_n \sim \text{Gamma}(d_1 + K_n + a, d_2 - \log\psi)$$

where $w_1 = d_1 + K_n + 1$ and $w_2 = n(d_2 - \log\psi)$.

end for

where K^- are the number of distinct c_j for $j \neq i$, labeled $\{1, \ldots, K^-\}$, and $n_{-i,c}$ is the number of $c_j = c$ for $j \neq i$. The likelihood of observation i being assigned a new variance, i.e. seated at a new table, is defined as

$$\begin{aligned} g(y_i; \boldsymbol{x}_i, \boldsymbol{\beta}, b_1, b_2) : &= \int \mathcal{N}(y_i; \boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \mathrm{IG}(\sigma^2; b_1, b_2) d\sigma^2 \\ &= \frac{b_2^{b_1}}{\sqrt{2\pi}} \frac{\Gamma(b_1 + 2^{-1})}{\Gamma(b_1)} \left(\frac{(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2}{2} + b_2 \right)^{-(b_1 + \frac{1}{2})} \end{aligned}$$

Finally, the concentration parameter α is sampled using a data augmentation trick (Ghosal and Van der Vaart, 2017, see pg.89).

We sample the regression coefficients $\boldsymbol{\beta}$ for the spike-and-slab and horseshoe models using Algorithms 19 and 20, respectively. The Gibbs sampler for the spike-and-slab model follows from Ishwaran and Rao (2005) while the Gibbs sampler for the horseshoe model is based on the data augmentation construction of Makalic and Schmidt (2015b). The full conditional distributions of all parameters in both models can be easily derived and have closed-form expressions (R. M. Neal, 2000; Ishwaran and Rao, 2005; Carvalho, Polson, and Scott, 2009). Alternatively, particularly for the case of $p \gg n$, we can get a random sample from the Multivariate Normal distribution of $\boldsymbol{\beta}$ using the linear solver of Bhattacharya, Chakraborty, and Mallick (2016) adapted to our Dirichlet process mixture model. This extension is detailed in Algorithm 21; this algorithm would replace the first step of Algorithms 19 and 20 once the parameters of matrices Σ and Λ are fixed. Since both Σ and Λ are diagonal matrices, the extension changes the cost of sampling $\boldsymbol{\beta}$ from $\mathcal{O}(p^3)$ when sampling directly from the multivariate normal to $\mathcal{O}(n^2 p)$ when using the linear solver.

An advantage of our Dirichlet process model construction is that the number of clusters K_n that partition the vector of variance parameters $\sigma^2_{1:n}$ are learned by the DP model and do not need to be fixed *apriori*.

Clustering the variance parameters means that we can account for outlier observations if one of the variances $\sigma^*_1, \ldots, \sigma^*_{K_n}$ is very large. Outlier observations belonging to that specific cluster will have much higher variance than the others and can take more extreme values. In this way, the model can simultaneously account

Algorithm 19 Sample β with the spike-and-slab model

Sample $\boldsymbol{\beta}$

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\mu}_{\beta|\cdot}, \Sigma_{\beta|\cdot})$$

where $\boldsymbol{\mu}_{\beta|\cdot} = (X^T\Sigma^{-1}X + \Lambda^{-1})^{-1}X^T\Sigma^{-1}\boldsymbol{y}$ and $\Sigma_{\beta|\cdot}^{-1} = X^T\Sigma^{-1}X + \Lambda^{-1}$, with $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2)$ and $\Lambda = \text{diag}(\tau_1^2\eta_1, \ldots, \tau_p^2\eta_p)$,

for j in 1:p **do**

Sample τ_j

$$\tau_j^{-2}|\boldsymbol{\beta}, \eta \sim \text{Gamma}\left(a_1 + 1/2, a_2 + \beta_j^2/2\eta_j\right),$$

end for

for j in 1:p **do**

Sample η_j

$$\eta_j|\beta, \tau, \omega \sim \frac{w_{1,j}}{w_{1,j} + w_{2,j}}\delta_{v_0}(\cdot) + \frac{w_{2,j}}{w_{1,j} + w_{2,j}}\delta_1(\cdot)\ ,$$

where $w_{1,j} = (1-\omega)v_0^{-1/2}\exp(-\beta_j^2/2v_0\tau_k^2)$ and $w_{2,j} = \omega\exp(-\beta_j^2/2\tau_j^2)$.

end for

Sample ω

$$\omega|\eta, \tau \sim \text{Beta}(1 + |\{j \mid \eta_j = 1\}|,\ 1 + |\{j \mid \eta_j = v_0\}|)$$

Algorithm 20 Sample β with the horseshoe model

Sample $\boldsymbol{\beta}$

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\mu}_{\beta|\cdot}, \Sigma_{\beta|\cdot})$$

where $\boldsymbol{\mu}_{\beta|\cdot} = (X^T\Sigma^{-1}X + \Lambda^{-1})^{-1}X^T\Sigma^{-1}\boldsymbol{y}$ and $\Sigma^{-1}_{\beta|\cdot} = X^T\Sigma^{-1}X + \Lambda^{-1}$, with $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2)$ and $\Lambda = \text{diag}(\tau^2\lambda_1^2, \ldots, \tau^2\lambda_p^2)$,

for j in 1:p **do**

Sample the local shrinkage parameter λ_j^2 and auxiliary variable ν_j,

$$\lambda_j^2|\cdot \sim \text{IG}(1, 1/\nu_j + \beta_j^2/2\tau^2), \quad \nu_j|\cdot \sim \text{IG}(1, 1 + 1/\lambda_j^2);$$

end for

Sample the global shrinkage parameter τ^2 the auxiliary variable ξ,

$$\tau^2 \sim \text{IG}\left((p+1)/2, 1/\xi + \sum_{j=1}^{p} \beta_j^2/2\lambda_j^2\right), \quad \xi \sim \text{IG}(1, 1 + 1/\tau^2);$$

Algorithm 21 Linear solver sampling of $\boldsymbol{\beta}$ given Σ and Λ.

Sample $u \sim \mathcal{N}(0, \Lambda)$ and $\delta \sim \mathcal{N}(0, \mathbb{I}_n)$.

Let $v = \Sigma^{-1/2}X + \delta$.

Solve $(\Sigma^{-1/2}X\Lambda X^T\Sigma^{-1/2} + \mathbb{I}_n)w = \Sigma^{-1/2}\boldsymbol{y} - v$.

Let $\boldsymbol{\beta} = u + \Lambda X^T\Sigma^{-1/2}w$.

for heteroskedasticity and outliers characterized by clusters with extreme variances.

6.3.3.2 Heavy Tails: Student-t extension

In many real-world datasets, the response variable $\mathbf{y}$ may contain larger-than-expected values, commonly referred to as *outliers*. A common approach to model such data is to replace the normal distribution assumption for the conditional distribution of y_i given $\boldsymbol{x}_i$ with a Student-t distribution. Compared to a normal distribution, the Student-t distribution has heavier tails and permits outlier observations with a higher probability than under the normal model. We can account for outliers in our Dirichlet process variable selection model (6.5) by replacing the conditional distribution of y_i with

$$y_i|\boldsymbol{x}_i, \boldsymbol{\beta}, \boldsymbol{\sigma^2} \sim \text{Student-t}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma_i^2, \nu) \quad i = 1, \ldots, n,$$

for some degrees of freedom ν. The parameter ν regulates the thickness of the tails of the distribution. A small value for ν leads to thicker tails with the expectation that large or extreme values for y_i will be observed. As $\nu \to \infty$, the Student-t distribution recovers the normal distribution.

A Gibbs sampler to perform posterior inference for the Student-t model can be derived from Algorithm 18 by including an additional data augmentation step. The required conditional distributions are derived by representing the Student-t distribution as a normal distribution with an inverse gamma variance. The remainder of this section describes these procedures.

We assume our response variables are distributed as $Y \sim \text{Student-t}(\mu, \sigma^2, \nu)$ with density,

$$f(y|\mu, \sigma^2, \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu\sigma^2}} \left(1 + \frac{(y-\mu)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}} \tag{6.7}$$

with mean $\mathbb{E}(Y) = \mu$ and variance $\text{Var}(Y) = \sigma^2 \frac{\nu}{\nu-2}$. A well-known reparameterization of the model follows that if,

$$Y|G \sim \mathcal{N}(\mu, G^{-1}) \quad \text{and} \quad G \sim \text{Gamma}(\nu/2, \nu\sigma^2/2),$$

Algorithm 22 DP variable selection Gibbs sampler, Student-t extension

for t in 1: number of iterations **do**

for i in 1:n **do**

Sample classification vector c_i with probability (6.9).

end for

for k in 1:K_n **do**

Sample σ_k^{2*}

$$\sigma_k^{2*} \sim \text{Gamma}\left(\frac{\nu}{2}n_k + b_1, \frac{\nu}{2}\sum_{i:C_k} G_i + b_2\right),$$

where $C_k = \{G_i \mid c_i = k\}$.

end for

if Spike-and-Slab **then**

Sample $\boldsymbol{\beta}$, τ_j, η_j and ω using Algorithm 19, replacing $\Sigma = \text{diag}(\sigma^2_{1:n})$ with $\Sigma = \text{diag}(G_{1:n})^{-1}$.

else if Horseshoe **then**

Sample $\boldsymbol{\beta}$, λ_j and τ using Algorithm 20, replacing $\Sigma = \text{diag}(\sigma^2_{1:n})$ with $\Sigma = \text{diag}(G_{1:n})^{-1}$.

end if

for i in 1:n **do**

Sample G_i

$$G_i \sim \text{Gamma}\left(\frac{\nu+1}{2}, \frac{1}{2}[(Y_i - X_i^T\beta)^2 + \nu\sigma_i^2]\right).$$

end for

Sample α, by sampling the latent variables

$$\psi|\alpha, K_n \sim \text{Beta}(\alpha+1, n) \quad \text{then} \quad a|\psi, K_n \sim \text{Bernoulli}\left(\frac{w2}{w1+w2}\right)$$
$$\text{and} \quad \alpha|a, \psi, K_n \sim \text{Gamma}(d_1 + K_n + a, d_2 - \log\psi)$$

where $w_1 = d_1 + K_n + 1$ and $w_2 = n(d_2 - \log\psi)$.

end for

then the marginal of Y (when integrating out G) is $Y \sim \text{Student-t}(\mu, \sigma^2, \nu)$.

The extension of the DPSS and DPHS models to a Student-t model is as follows

$$\begin{aligned} y_i|\boldsymbol{x}_i, \boldsymbol{\beta}, \sigma_i^2 \sim \text{Student-t}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma_i^2, \nu) \quad \text{with} \quad \boldsymbol{\beta} \sim \pi_{\text{SS/HS}} \quad \text{and} \\ \sigma_i^2|P \sim P \quad \text{with} \quad P \sim \text{DP}(\alpha, \text{Gamma}(b_1, b_2)) \end{aligned} \tag{6.8}$$

Using the reparameterized Student-t model, and integrating out P from (6.8), the joint posterior distribution $\pi(\beta, G_{1:n}, \sigma_{1:n}^2, \alpha|Y_{1:n}, X_{1:n})$ is proportional to,

$$\prod_{i=1}^{n} \mathcal{N}(y_i; \boldsymbol{x}_i^T \boldsymbol{\beta}, G_i^{-1}) \text{Gamma}(G_i; \nu/2, \nu\sigma_i^2/2) \pi(\sigma_{1:n}|\alpha) \pi(\alpha) \pi_{SS/HS}(\beta),$$

where, as before, $\pi(\sigma_{1:n}|\alpha)$ denotes the marginal likelihood of $\sigma_{1:n}$ when P is integrated out, which can be represented using the Chinese restaurant process. The proposed Gibbs sampler is specified in Algorithm 22, where at each iteration, we first resample a class vector $c_{1:n}$ assigning each σ_i^2 and G_i to a block in the partition and then resample the distinct values $\sigma_{1:k}^{2*}$ followed by each latent variable G_i. In the heavy-tailed case, the observation is related to a variance through its latent variable and hence, the posterior probability of the classification vector is

$$\mathbb{P}[c_i = c \,|\, c_{-i}, G_{1:n}, \sigma_{1:K_n}^{2*}] \propto \begin{cases} n_{-i,c} \text{Gamma}(G_i; \nu/2, \nu\sigma_c^{2*}/2) \text{ for } 1 \leq c \leq K^- \\ \\ \alpha g(G_i; \nu, b_1, b_2) \text{ for } c = K^- + 1, \end{cases} \tag{6.9}$$

where K^- be the number of distinct c_j for $j \neq i$, labeled $\{1, \ldots, K^-\}$, and $n_{-i,c}$ is the number of $c_j = c$ for $j \neq i$. Also, the likelihood of observation i being assigned a new variance is modified to

$$\begin{aligned} g(G_i; \nu, b_1, b_2) :&= \int \text{Gamma}(G_i; \nu/2, \nu\sigma^2/2) \text{Gamma}(\sigma^2; b_1, b_2) d\sigma^2 \\ &= \frac{(\nu/2)^{\nu/2}}{\Gamma(\nu/2)} G_i^{\frac{\nu}{2}-1} \frac{b_2^{b_1}}{\Gamma(b_1)} \frac{\Gamma(b_1 + \frac{\nu}{2})}{(b_2 + \nu G_i/2)^{b_1 + \frac{\nu}{2}}}. \end{aligned}$$

Finally, regression coefficients $\boldsymbol{\beta}$ are sampled using Algorithms 19 and 20 with small modifications to include the augmented variable G_i.

6.4 Experiments

This section compares the DPSS and DPHS models against the standard spike-and-slab and horseshoe models of Ishwaran and Rao (2005) and Carvalho, Polson, and Scott (2010). We begin by studying their predictive performance, ability to model heterogeneous data, and variable selection capabilities over a range of scenarios utilizing synthetic data, and then, we test the models' performance on a real-world dataset. Specifically, on reconstructing a network of genomic data through variable selection on a linear model. Code for synthetic and real experiments can be found in `https://github.com/albcab/RobustVariableSelection`.

Each algorithm is run for $J = 10,000$ iterations for all experiments with a burn-in period of $J/2$. Convergence and mixing of the Markov chain is confirmed through visual diagnostics. Hyper-parameters for the models are set to $a_1 = b_1 = 2.01$, $a_2 = b_2 = d_1 = 1$, and $d_2 = 1/2$, which leads to weakly informative priors. This claim is verified with simple cross validation for all the datasets. It is worth noting that one could place additional hyper-priors on these parameters and add extra steps to the MCMC algorithm. However, in doing so, we would risk rejecting steps, potentially leading to longer mixing times. It is worth noting that in comparable studies (e.g. Ishwaran and Rao, 2005), it is common to fix these parameters when assessing performance simply. Finally, both the dependent and independent variables are normalized to have zero mean for the testing of each model.

In synthetic data, Performance is evaluated over a range of regression scenarios varying the number of attributes $p = 50, 100, 200, 500, 1000, 2000$ and data lengths $n = 10, 20, 50, 100, 200$. The sparsity pattern will be similar across experiments according to the Briemann structure (Breiman et al., 2001). Specifically, we consider two scenarios for Gaussian linear regression: Scenario (1) - homoskedastic errors with $\epsilon \sim \mathcal{N}(0, 1)$ as the single component and $\epsilon \sim$ Student-t$(0, 1, 2)$ when testing the heavy tails extension; Scenario (2) - heteroskedastic errors with five components $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ or $\epsilon \sim$ Student-t$(0, \sigma_i^2, 2)$ for $\sigma_i^2 \in \{0.5, 1, 1.5, 2, 2.5\}$ distributed evenly among the observations as well as $1, 2$ and 4 outliers $\epsilon \sim \mathcal{N}(0, 10)$ or $\epsilon \sim$

Student-t$(0, 10, 2)$ in the case of $n = 50, 100, 200$, respectively.

In each scenario, and for each combination of attributes and data lengths, 100 synthetic datasets are created and parameters are estimated for each of these datasets. The results presented in the remainder of this section consider all 100 estimates for robustness.

6.4.1 Homoskedastic and Heteroskedastic Errors

We compare the Dirichlet process variable selection models against the standard spike-and-slab and horseshoe models in the presence of homoskedastic (*Scenario 1*) and heteroskedastic (*Scenario 2*) noise. Figure 6.1 presents boxplots of the posterior error $\|\hat{\beta} - \beta_0\|_2 / \|\beta_0\|_2$ for every combination of the number of observations n and parameters p, with *Scenario 1* plotted on the left and *Scenario 2* given on the right, while Figure 6.2 presents the same error boxplots when testing the heavy-tailed extension of the models.

Robustness - it is clear from the upper right plots on Figure 6.1 that for *Scenario 2*, the robustness provided by the DPSS and DPHS models improves the predictive estimates of parameters $\boldsymbol{\beta}$ when data is sufficient to make the model overdetermined, i.e. when n is at least as large as p. As expected, under *Scenario 1*, the results are similar for the models, allowing heterogeneous and homogeneous variances. Interesting behavior is observed when the model is underdetermined at varying degrees. When the model is highly underdetermined $n \ll p$, models with the horseshoe priors provide more robust results than spike-and-slab priors. However, as n approaches p the spike-and-slab prior provides lower errors than the horseshoe (albeit at a smaller scale), until the model becomes overdetermined and the horseshoe prior models are preferred. Similar results are observed for the heavy-tailed extension of the models. In contrast to the Gaussian models, when the models are at least determined $n \geq p$, the positive effect of robust variance modeling is less noticeable.

Coefficients - looking at Figure 6.1 and 6.2 from top to bottom, the effect of adding coefficients to the model is illustrated on the predictive capabilities of the

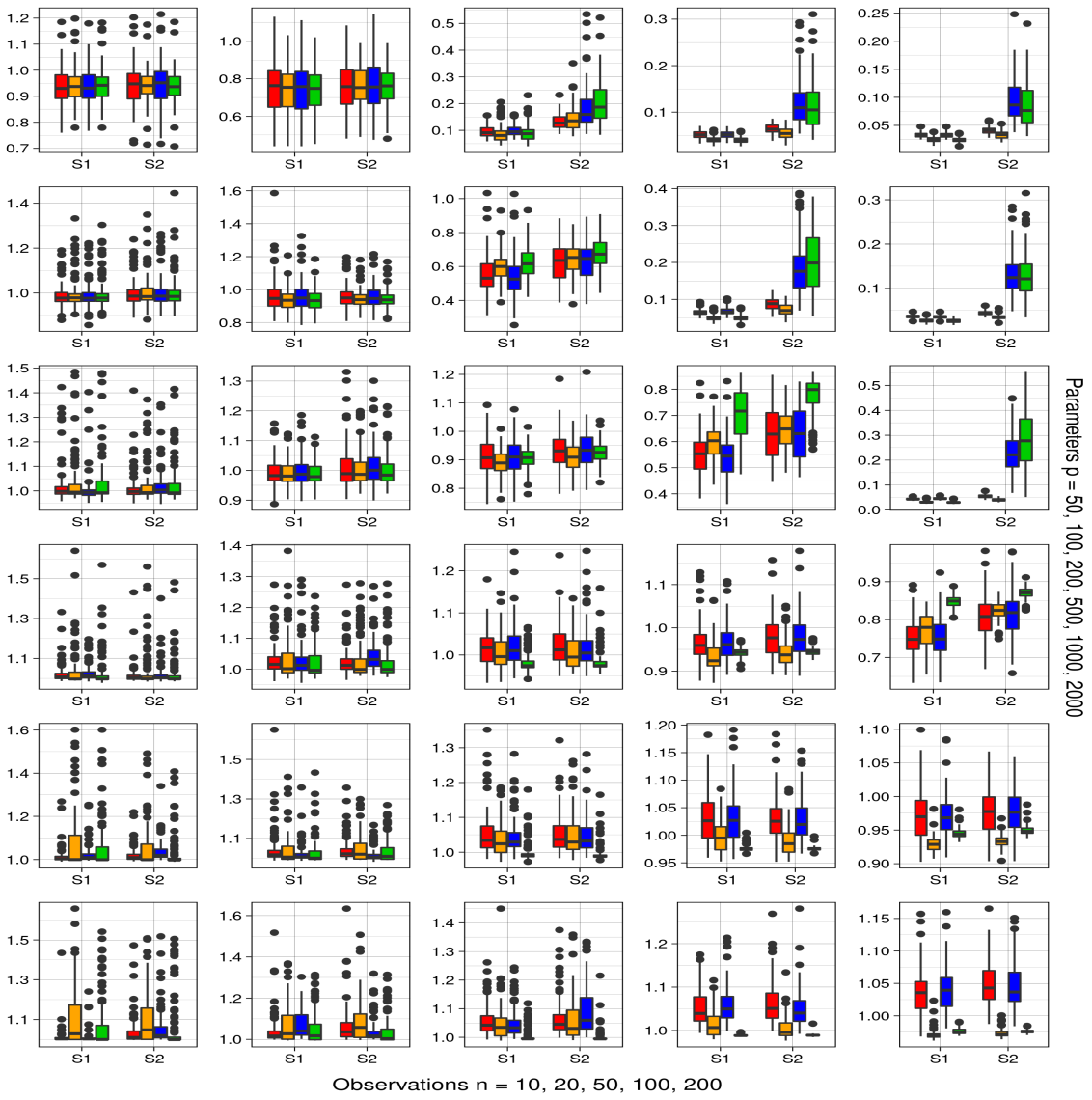


Figure 6.1: Boxplots of estimation error $\|\hat{\beta} - \beta_0\|_2 / \|\beta_0\|_2$ on *Scenario 1* (S1) and *Scenario 2* (S2) for algorithms Dirichlet process horseshoe, Dirichlet process spike-and-slab, Horseshoe, and Spike-and-slab.

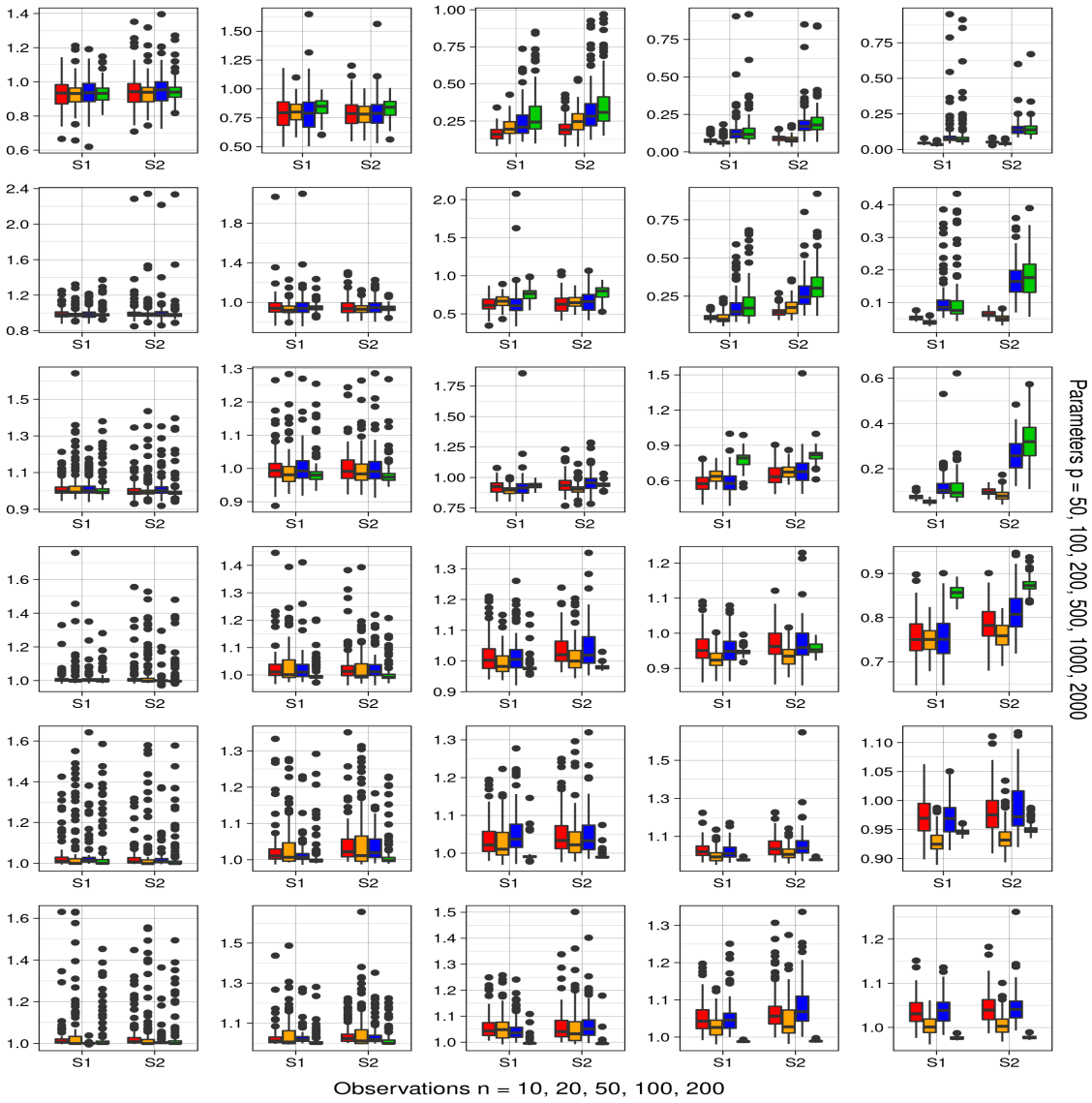


Figure 6.2: Boxplots of estimation error $\|\hat{\beta} - \beta_0\|_2 / \|\beta_0\|_2$ on *Scenario 1* (S1) and *Scenario 2* (S2) for algorithms Dirichlet process horseshoe, Dirichlet process spike-and-slab, Horseshoe, and Spike-and-slab on their heavy tailed Student-t extension.

models. The Dirichlet process model performs well in heterogeneity, particularly as the number of model coefficients exceeds the number of observations. The robustness added by the heterogeneous variances benefits the horseshoe prior in the overdetermined case and the spike-and-slab prior in the underdetermined case.

Clusters - one of the key advantages of the nonparametric approach we adopt is the availability of posterior densities for the number of variance components K. The Dirichlet process' assumption of a number of clusters that scale logarithmically towards infinity with n can be observed by looking at the boxplots for samples of K for an increasing n in Figure 6.3 for the heavy-tailed extension. A potentially useful caveat of the heavy-tailed extension of the robust models is that the number of clusters K increases at a slower rate than in the case of Gaussian errors. In the case where a finite number of clusters in the data is known, or the number of clusters K should scale faster or slower than logarithmically, priors that generalize the Dirichlet process can be used, e.g. the Pitman-Yor process (Jim Pitman and Yor, 1997). It is interesting to notice how, as the number of coefficients increases but the number of observations remains constant, the number of clusters K seem unbiased when the number of coefficients becomes much larger than the number of observations, as shown in Figures 6.4 and 6.5. A comparison of these two Figures also shows how the heavy-tailed extension of the robust model provides a more gradual increase in the number of clusters K.

6.4.2 Support Recovery

In addition to estimating the regression coefficients, a further task of interest, especially in high-dimensional regression, is to select a subset of attributes deemed significant for the predictive model. In a frequentist context, this is usually achieved via forward selection with sequential F-tests or with ℓ_1 or ℓ_0 shrinkage and/or selection, for instance, through the use of the AIC or BIC criteria.

One advantage of the Bayesian approach is that we can sample from the joint posterior of the coefficients, thus constructing credible intervals with relative ease.

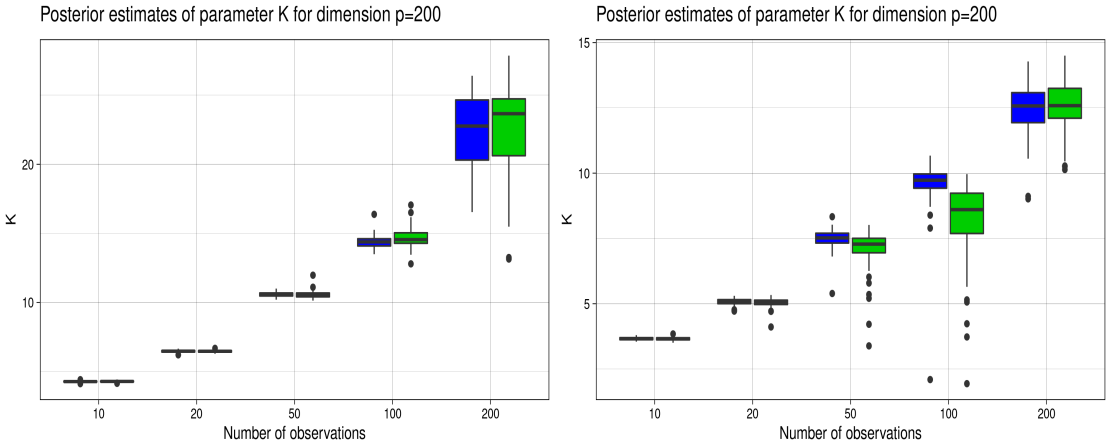


Figure 6.3: Boxplots of the derived number of clusters K from the posterior of $\sigma_1, \ldots, \sigma_n$ for $p = 200$ for algorithms Dirichlet process horseshoe, Dirichlet process spike-and-slab, with a Gaussian likelihood on the left and a Student-t likelihood on the right, where the true number of clusters is $K = 5, 5, 6, 7, 9$ for $n = 10, 20, 50, 100, 200$.

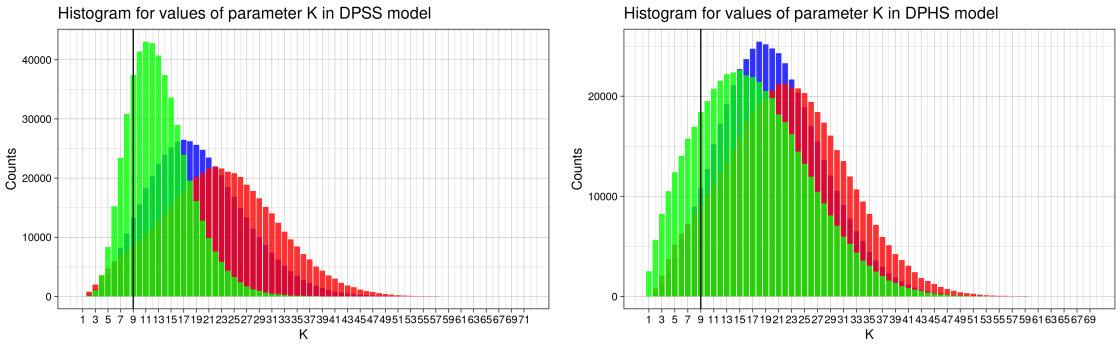


Figure 6.4: Histogram of the derived number of clusters K from the posterior of $\sigma_1, \ldots, \sigma_n$ for $n = 200$ and $p = 50$, 200, and 2000, where the two number of clusters is $K = 9$.

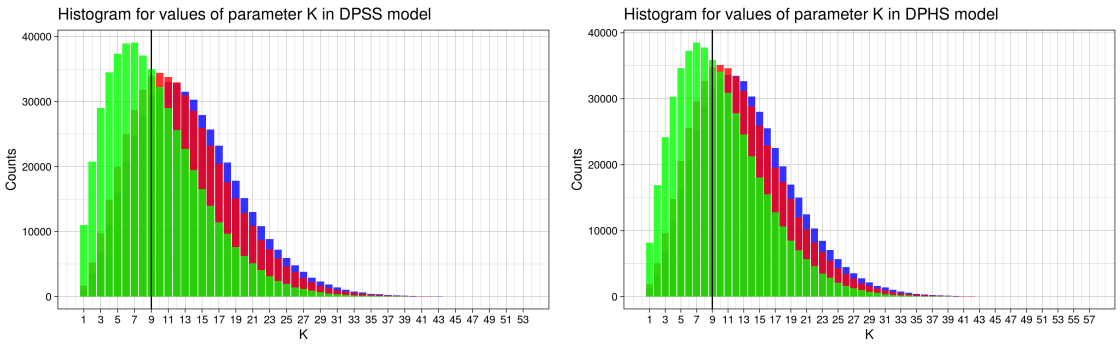


Figure 6.5: Histogram of the derived number of clusters K from the posterior of $\sigma_1, \ldots, \sigma_n$ for $n = 200$ and $p = 50$, 200, and 2000 on their heavy-tailed Student-t extension, where the two number of clusters is $K = 9$.

In this section, we compare two methods for estimating the support of the regression model:

- The first approach is to look at the posterior of the inclusion parameter η_j for selecting either the spike or slab. Specifically, if the mode of the posterior $\hat{\eta}_j$ is above level $1 - \zeta$ we add the index j to the estimated support set $\hat{\mathcal{M}}$. This method works only in the DPSS and SS models.
- The second approach is to look at the empirical posterior credible interval of $\hat{\beta}_i$ at the percentiles $\zeta/2 \times 100$ and $(1 - \zeta/2) \times 100$. If the constructed interval excludes zero, we add the index to the estimated support set $\hat{\mathcal{M}}$. This method is used for the DPHS and HS models.

We note that the second approach is somewhat similar to the *z-cut* method discussed in Ishwaran and Rao (2005); however, they construct a set $\hat{\mathcal{M}} := \{i \mid \forall i \; s.t. \; |\bar{\beta}_i| \geq z_{(1-\zeta/2)}\}$ where $\bar{\beta}_i$ represents the posterior mean of the regression coefficients.

To summarise the performance of the two approaches, we take the number of true and false positive (TP, FP) coefficients included in the model and consider how this changes as a function of n. These values will change as a function of ζ, and in practice, this may be tuned to favor either TP or FP results. For simplicity, we

report results with $\zeta = 0.05$. Table 6.1 summarises the performance of the posterior inclusion thresholding method for different values of n in the heteroskedastic scenario (*Scenario 2*) and the standard Gaussian formulation of the models; similar results were observed on their heavy-tailed extension. In the case of an overdetermined model, both the standard and robust implementations learn the underlying data-generating process correctly. However, when the model is underdetermined, the standard spike-and-slab tends to be more conservative in including variables. At the same time, the robust method favors a better estimate of TP with the drawback of more FP. Similar results can be observed in the case of the horseshoe prior. The support recovery, or model selection, of the robust models, will be detailed further when applied to real-world data, where the benefits of each prior are illustrated.

Model	n	20		50		100		200	
SS	TP	5	[0.95,8]	14	[12,14]	14	[14,14]	14	[14,14]
	FP	3	[0,7]	0	[0,1]	0	[0,0]	0	[0,0]
DPSS	TP	5	[2,8]	14	[12,14]	14	[14,14]	14	[14,14]
	FP	3	[0.95,8]	0	[0,1]	0	[0,1]	0	[0,0]
SS	TP	4	[1,10]	17	[1,25]	28	[28,28]	28	[28,28]
	FP	5	[0,13.15]	**18**	[0,43.3]	0	[0,1]	0	[0,1]
DPSS	TP	4	[1,9.05]	**20**	[14,26]	28	[28,28]	28	[28,28]
	FP	4	[0,14]	24	[6.95,46.05]	0	[0,1]	0	[0,1]
SS	TP	**4**	[0,18.15]	25	[0,51.05]	31	[1,56]	56	[56,56]
	FP	6	[0,36.55]	**53**	[0,124.2]	**62**	[0,144]	0	[0,1]
DPSS	TP	3	[0,12.1]	**42**	[16.95,56]	**54**	[47.9,56]	56	[56,56]
	FP	6	[0,22]	99	[25.9,142.05]	133	[95.8,144]	0	[0,1]

Table 6.1: Tabulated results of true positive (TP) and false positive (FP) results with 95% credible intervals for the inclusion of regression coefficients for $p = 50, 100, 200$ with nonzero coefficients TP $= 14, 28, 56$.

6.4.3 Reconstruction of transcription regulatory networks

We consider the problem of reconstructing genetic regulatory networks from gene expression data (Marbach et al., 2010). This problem can be modeled as a directed network, where each node corresponds to a different gene, and each connection represents a directed interaction between two genes at the transcription level. The sparse spike-and-slab and horseshoe priors provide an attractive approach to reconstructing these networks using support recovery methods.

The models are tested on data from the challenge posed at the *The Dialogue for Reverse Engineering Assessments and Methods* (DREAM) 2009 conference, specifically the multifactorial subchallenge. The challenge is reverse engineering five independent networks using 100 steady-state measurements for each network with 100 genes. All the genes' expression levels are measured under different perturbed conditions. Each gene X_i for $i = 1, ..., 100$ is treated independently and modeled through a linear relationship with the rest of the genes in the network $\boldsymbol{X}$, i.e. $X_i = \boldsymbol{X}^T\boldsymbol{\beta}_i + \epsilon$. Every $\boldsymbol{\beta}_i$ has an independent spike-and-slab or horseshoe prior and ϵ is a normal random variable with mean zero and homogeneous variances in the case of SS and HS, or heterogeneous variances, in the case of DPSS and DPHS. For each gene, the support is recovered using the method described in Section 6.4.2 for the case of a spike-and-slab prior with $\boldsymbol{\beta}_i$. In the case of a horseshoe prior, we follow Steinke, Seeger, and Tsuda (2007) and approximate the posterior probability of a connection from gene j to gene i by the probability of the event $|(\boldsymbol{\beta}_i)_j| > 0.1$ under the posterior for $\boldsymbol{\beta}_i$.

The performance of the different approaches is evaluated using the mean of the logarithmic loss of the probabilities of connections p_{ij} from gene j to each gene i, defined as

$$\frac{1}{100}\sum_{i=1}^{100}\left\{-\frac{1}{99}\sum_{j\neq i}[y_{ij}\log\, p_{ij} + (1-y_{ij})\log\,(1-p_{ij})]\right\},$$

where $y_{ij} = 1$ if there is a directed connection from j to i and $y_{ij} = 0$ otherwise. Table 6.2 presents these results for each tested model. The improvement in prediction given

by heterogeneous variances in the model is substantial. This supports the results from Section 6.4.2 on synthetic data, illustrating the improvement offered by robust models to correctly classify variables included in the model, specifically in their capability of reducing type II error in the predictions while correctly identifying the significant coefficients in the linear relationship. Interestingly, while the DPSS model provided better support recovery results in the synthetic data example, the DPHS provides better results in the real data example of gene network reconstruction.

	SS	DPSS	HS	DPHS
N1	0.1143	**0.0948**	0.0920	**0.0854**
N2	0.1498	**0.1380**	0.1248	**0.1173**
N3	0.1610	**0.1158**	0.1196	**0.0900**
N4	0.1588	**0.1118**	0.1124	**0.0970**
N5	0.1314	**0.1095**	0.1044	**0.0933**

Table 6.2: Logarithmic-loss errors for the five DREAM Gene Network detection datasets (N1-N5).

6.5 Discussion

This research outlines a nonparametric extension of popular Bayesian variable selection models to account for heterogeneity, outliers, and clustering effects. We also extend the model to allow for heavy-tailed data distributions. Fitting our Dirichlet process variable selection models is computationally efficient using a Gibbs sampling construction. The results presented for both synthetic and real data examples show a robust improvement in predictive accuracy on test data and improved efficiency in identifying key model attributes.

This work could be further extended to encompass other variable selection models, including extensions of existing models, such as the regularized horseshoe model (Piironen, Vehtari, et al., 2017). It would also be interesting to explore nonparametric

priors that allow for more flexible assumptions on the expected clustering effect, such as the Pitman-Yor process (Jim Pitman and Yor, 1997).

Chapter 7

BlackJAX: Composable Bayesian inference in JAX

BlackJAX is a library that implements sampling and variational inference algorithms commonly used in Bayesian computation. It is designed for ease of use, speed, and modularity by taking a functional approach to the algorithm's implementation. Designed from basic components to specific iterative procedures, BlackJAX allows the end user to build and experiment with new algorithms by composition. BlackJAX is written in pure Python using JAX (Bradbury et al., 2018) to compile and run NumpPy-like programs on CPUs, GPUs and TPUs. The library integrates well with probabilistic programming languages by working directly with the (unnormalized) target log density function, given that the function is pure. The library is intended for users who need to create complex sampling mechanisms beyond the black-box solution, researchers who want to experiment when developing new algorithms and students who want to learn how inference algorithms work.

7.1 Introduction

Sampling from a probability distribution, either manually defined or constructed using probabilistic programming languages (PPLs), is a recurring topic in statistics and machine learning. Automatic sampling software has historically been limited to Gibbs-type methods (Meyer and Yu, 2000; Lunn et al., 2000; Depaoli, Clifton, and Cobb, 2016), requiring knowledge of the model structure. Black-box samplers, typically relying on Hamiltonian Monte Carlo (HMC, Duane et al., 1987), allowed general, model-agnostic improvement in the applicability of the method. This was spearheaded by Stan (Carpenter et al., 2017), which leveraged the development of automatic differentiation. The same developments have allowed for automatically learning rich approximations via variational inference (VI, M. I. Jordan et al., 1999) to the models of interest. Together, these led to the creation of an array of modern PPLs that have pushed the boundaries on the feasibility of Bayesian computation (Abril-Pla et al., 2023; Bingham et al., 2019; Phan, Pradhan, and Jankowiak, 2019). While black-box samplers have paved the way, we believe that inference in today's models increasingly requires reintroducing structure-aware algorithms.

To achieve this, BlackJAX provides users with composable inferential building blocks are written using JAX (Bradbury et al., 2018), such as Metropolis–Hastings (Metropolis et al., 1953; Hastings, 1970a; Robert, 2016) accept/reject step, Hamiltonian or Langevin (Besag, 1994) dynamics, stochastic gradient utilities, resampling and tempering mechanisms for use within sequential Monte Carlo (SMC, Del Moral, Doucet, and Jasra, 2006a), or mean field approximations (M. I. Jordan et al., 1999), as well as other mechanisms. These components are unified under a convenient, functionally oriented API that can be combined to form new or existing algorithms immediately applicable to sequential and parallel modern computer architectures.

7.2 Design principles

BlackJAX supports both sampling algorithms, such as Markov Chain Monte Carlo (MCMC), Sequential Monte Carlo (SMC), and Stochastic Gradient MCMC (SGMCMC), and approximate inference algorithms, such as Variational Inference (VI). In both cases, the iterative procedure of the algorithm must be kept functionally pure, i.e. return values that are identical for identical arguments and have no side effects, allowing efficient parallelization. Thus, a fundamental component of the algorithm is a state object that contains all the necessary information for the next iteration. All sampling algorithms carry the current value in its state while some might also carry the density and gradient evaluations of the current value to minimize computations or other necessary objects for computing the transition. Similarly, all approximate inference algorithms carry the current parameters of the approximate distribution and might carry other necessary values to compute the iteration.

The user starts by initializing a state given an initial value. Then, recycling the current state, stepping or iterating to a new state with either a new sample or an improved approximation. In the latter case of VI, the user requires a function to generate samples given the current approximation. The stepping function also returns an information state concerning the current iteration, it includes essential information that might be useful for the user. The user-facing interface of each variant is illustrated in the following code:

```
# Generic sampling algorithm:
sampling_algorithm = blackjax.sampling_algorithm(logdensity_fn, ...)
state = sampling_algorithm.init(initial_position)
new_state, info = sampling_algorithm.step(rng_key, state)

# Generic approximate inference algorithm:
approx_algorithm = blackjax.approx_algorithm(logdensity_fn, optimizer, ...)
state = approx_algorithm.init(initial_position)
new_state, info = approx_algorithm.step(rng_key, state)
position_samples = approx_algorithm.sample(rng_key, state, num_samples)
```

7.2.1 Lower-level API

Users might need a tailored algorithm for the model they are trying to sample or approximate from, or they might try out different Markov transition kernels running in parallel with various particles in an SMC algorithm, or they might use optimization to approximate the hyperparameters of the models while sampling from the rest. For any of these cases, BlackJAX provides access to a lower-level API, giving access to functions implementing methods with more parameters. Then the user-facing interface becomes (using MCMC and VI for illustration, but it can be any algorithm type):

```
# Lower-level sampling algorithm:
init = blackjax.mcmc.sampling_algorithm.init
state = init(initial_position, logdensity_fn)
kernel = blackjax.mcmc.sampling_algorithm.build_kernel(...)
new_state, info = kernel(rng_key, state, logdensity_fn, ...)

# Lower-level approximate inference algorithm:
init = blackjax.vi.approx_algorithm.init
state = init(initial_position, optimizer, ...)
step = blackjax.vi.approx_algorithm.step
new_state, info = step(rng_key, state, logdensity_fn, optimizer, ...)
sample = blackjax.vi.approx_algorithm.sample
position_samples = sample(rng_key, state, num_samples)
```

Using these lower-level implementations, a Metropolis-within-Gibbs algorithm is easily implementable by using higher-order functions and composing various Markov transition kernels:

```
def cond_logdensity_fn(condition):
    return lambda eval: logdensity_fn(**eval, **condition)

state1 = init1(initial_position1, cond_logdensity_fn(initial_position2))
state2 = init2(initial_position2, cond_logdensity_fn(initial_position1))

def _kernel1(key, state1, state2):
    target_cond = cond_logdensity_fn(state2.position)
    state1 = init1(state1.position, target_cond)
    return kernel1(key, state1, target_cond, ...)

def _kernel2(key, state2, state1):
    target_cond = cond_logdensity_fn(state1.position)
    state2 = init2(state2.position, target_cond)
```

```
    return kernel2(key, state2, target_cond, ...)

def one_step(states, key):
    key1, key2 = jax.random.split(key)
    state1, state2 = states
    state1, info1 = _kernel1(key1, state1, state2)
    state2, info2 = _kernel2(key2, state2, state1)
    return (state1, state2), (info1, info2)
```

The functional design of the library, where programs are constructed by applying and composing functions, allows the end user to build and experiment with new algorithms by applying the same mathematical logic used to design them. The lower-level interface can apply different base kernels in an SMC setting or combine optimization within sampling algorithms. For a detailed and exhaustive example, see the implementation of the window_adaptation scheme for adaptation of step size and mass matrix in HMC.

7.2.2 Basic components

Basic components are functions that perform specific tasks but are generally applicable. All inference algorithms are composed of basic components; in a sense, these components provide the lowest level of algorithm abstraction. We assume that when implementing a new inference algorithm, scientists first break it into its basic components. With BlacKJAX, they can find all basic components already implemented before implementing their own.

For instance, BlackJAX contains two variants of the MH accept/reject step, starting from the computation of the accept/reject probability: the simpler `safe_energy_diff` if the proposal transition kernel is symmetric and the more general `compute_asymmetric_acceptance_ratio` if the proposal transition kernel is asymmetric. Hence, the HMC algorithm uses the former while the Metropolis adjusted Langevin algorithm (MALA Besag, 1994) uses the latter. After the acceptance probability of the proposal is computed, the proposal is accepted or rejected using `static_binomial_sampling`. In BlackJAX, this staple of MCMC

can be immediately swapped for the non-reversible slice sampling algorithm of Radford M Neal (2020) simply by replacing `static_binomial_sampling` with `nonreversible_slice_sampling`.

Consider the case of HMC as a further illustration. An iteration of HMC requires a proposal and an acceptance mechanism, traditionally using the MH with a symmetric proposal. The proposal mechanism of HMC requires the generation of a Gaussian momentum and a velocity Verlet integrator. Each of these basic components could be replaced with a different component, the user might choose to use a two or three-stage palindromic symplectic integrator (McLachlan, 1995) or to use slice sampling instead of the MH acceptance (Radford M Neal, 2003).

The utility of basic components comes when a team of researchers such as M. D. Hoffman and Sountsov (2022) could implement their generalized HMC algorithm simply by replacing the symmetric MH step of HMC with a non-reversible slice sampling step, as described above, and using a persistent generation of the momentum variable. For its implementation in BlackJAX see ghmc. Researchers can readily test and compare new ideas by changing or modifying two basic components while keeping the rest of the algorithm the same. Appendix A.1 has a detailed overview of how each type of algorithm is structured.

7.2.3 Existing sampling libraries

In the Python ecosystem, Bayesian inference libraries usually either implement domain-specific algorithms, such as EMCEE (Foreman-Mackey et al., 2019), Dynesty (Speagle, 2020), pocoMC (Karamanis, Nabergoj, et al., 2022), and Mici (Graham, n.d.) (to name a few) or are directly tied to a PPL (Bingham et al., 2019; Tran et al., 2019; Oriol et al., 2023; Carpenter et al., 2017; Lao et al., 2020). BlackJAX is one of the few Python libraries specifically aimed at users who want to use but also develop inference methods. Its design philosophy allows for more control over how the algorithm is built, enabling users to make modifications and tailor the implementation to specific needs, fostering innovation

and experimentation in probabilistic modelling. Efforts in the same direction can be found in FunMC (Sountsov, Radul, and Vasudevan, 2021), part of the TensorFlow Probability (Dillon et al., 2017) ecosystem. In Python, AeMCMC automatically constructs MCMC samplers for probabilistic models by exploiting the symbolic graphs structure of programs written in Aesara. In Julia, Mamba.jl provides a platform for implementing and applying MCMC methods to perform Bayesian analysis.

7.3 Past impact of BlackJAX on the practice of Bayesian inference

Since its inception, BlackJAX has already left a tangible mark on the landscape of Bayesian inference. The library's impact extends across both the applicative and methodological aspects of Bayesian analysis. BlackJAX joins the already existing rich ecosystem of JAX-powered scientific libraries (Schoenholz and Cubuk, 2020; Wilkinson, Särkkä, and Solin, 2023; Bonnet et al., 2023) and is directly compatible with several of them either as a client: consuming outputs from these (Pinder and Dodd, 2022; DeepMind et al., 2020), or as a component, used within the libraries (Phan, Pradhan, and Jankowiak, 2019; Kumar et al., 2019).

BlackJAX contains a comprehensive implementation of state-of-the-art HMC algorithms, including vanilla HMC with various integrators, the no-U-turn sampling (NUTS) to choose the number of integration steps at each iteration dynamically (M. D. Hoffman, Gelman, et al., 2014), and the generalized HMC algorithm (Horowitz, 1991). It also contains adaptation schemes for the algorithms' hyper-parameters: window adaptation and sophisticated calibration methods such as M. Hoffman, Radul, and Sountsov (2021) and M. D. Hoffman and Sountsov (2022).

Consequently, several papers have leveraged BlackJAX to conduct research in various fields (see, e.g., Galan et al., 2022; Price-Whelan et al., 2024; Balkenhol et al., 2024). Moreover, BlackJAX has made contributions to the methodological development of Bayesian inference: it has been adopted in a range of papers to

develop new Bayesian sampling methods (Staber and Da Veiga, 2022; Cabezas and Nemeth, 2023; Cooper et al., 2023).

Beyond research publications, BlackJAX has found a place in courses and tutorials, for example, functional programming by Darren Wilkinson and its use in Kevin Murphy's authoritative manuscript (Murphy, 2022). This usage attests to the library's recognition as a practical resource for teaching Bayesian concepts, making it accessible to a broader audience of learners and practitioners

7.4 Roadmap to the future of BlackJAX

The development of BlackJAX represents a significant step forward in Bayesian computation libraries. Building on the principles of ease of use, modularity, and efficiency, we have ambitious plans for its future to enhance its capabilities and usability further. Our roadmap outlines the fundamental directions we intend to pursue in its ongoing development.

7.4.1 Enhanced algorithm portfolio

We recognize that the field of Bayesian computation is dynamic and diverse, with emerging algorithms and techniques. Soon, we aim to expand the library's algorithm portfolio to include a broader range of state-of-the-art sampling and approximate inference methods. This expansion will empower users with more options for their specific computational needs. Current plans include:

Parallel and Sequential Tempering SMC Building on the existing SMC algorithms with annealing in BlackJAX, our immediate goal is to introduce parallel and sequential tempering SMC algorithms (Corenflos, Chopin, and Särkkä, 2022). By integrating these algorithms, BlackJAX aims to empower users with the capability to tackle a broader range of Bayesian inference problems.

Unbiased MCMC with Couplings This development is particularly exciting, as it aligns with recent advancements in the field. Unbiased MCMC techniques

promise to reduce bias in posterior inference, improving estimation accuracy (Jacob, O'Leary, and Atchadé, 2020). Including such methods in BlackJAX will provide users with additional tools for more reliable Bayesian analysis.

Structured VI Our roadmap also includes methods such as the Integrated Nested Laplace Approximation (INLA). INLA is known for its efficiency in approximating posteriors for models with structured additive effects (Rue, Martino, and Chopin, 2009). By integrating INLA and similar methods into BlackJAX, we aim to facilitate efficient and accurate inference for complex structured models.

Diagnostics BlackJAX will provide diagnostic tools to assess the performance of various sampling and approximate inference methods. These diagnostics will aid users in understanding the quality of their Bayesian inference results and identifying potential issues in their models.

7.4.2 Documentation and tutorials

Clear and comprehensive documentation is essential for any library's success. We plan to expand documentation and tutorials, especially those catering to users with varying levels of expertise. This will empower students and researchers to learn and work with BlackJAX effectively. We plan to strengthen BlackJAX's integration with popular probabilistic programming languages. This includes further tutorials and examples for defining probabilistic models in these languages and seamless interaction with BlackJAX for posterior inference. By bridging the gap between BlackJAX and probabilistic programming, we aim to simplify the development of complex probabilistic models.

We plan to introduce an inference database feature similar to Stan's posteriordb, allowing users to access a collection of posteriors for testing and benchmarking. This feature aligns with the principles of transparency and reproducibility in Bayesian inference. It will include reference implementations in probabilistic programming languages and reference posterior inferences in the form of posterior samples. The integration of inference databases will support users in assessing and validating their

Bayesian models.

7.5 Project openness and development

BlackJAX's source code is available under the permissive Apache License license, which allows users to use the software for any purpose, distribute it, modify it, and distribute modified versions under the license terms without concern for royalties. BlackJAX is developed by a community of open-source contributors at blackjax. A comprehensive test suite is run automatically by a continuous integration service before code is merged into the main codebase to maintain high project quality and usability.

Anyone is welcome to contribute to the BlackJAX project. Contributions can be made in code, documentation, expert reviews of open pull requests, or other forms of support, such as case studies using BlackJAX. Breaking decisions about the BlackJAX project are made by consensus among the BlackJAX core contributors, and all BlackJAX core contributors must agree on such a decision before it can be implemented.

The author has contributed to approximately 5% of the algorithms currently available in the library, focusing primarily on the library's design philosophy, documentation, and ongoing maintenance efforts. These contributions help ensure that BlackJAX remains accessible, well-structured, and sustainable for long-term community use.

7.6 Discussion

The landscape of Bayesian computation has evolved significantly in recent years, driven by the intersection of probabilistic programming languages (PPLs) and innovative sampling libraries. With its efficiency and versatility, Hamiltonian Monte Carlo (HMC) has played a pivotal role in developing black-box Bayesian inference.

Probabilistic programming languages like Stan have revolutionized Bayesian analysis by making advanced HMC accessible to a broader audience.

In this context, BlackJAX is an essential addition to the Bayesian computation toolkit. By taking a functional approach to algorithm design and providing a library of basic components, BlackJAX empowers users to create and experiment with new inference algorithms. Its support for a wide range of sampling and approximate inference methods, combined with a convenient, functionally oriented API, positions it as a powerful tool for seasoned researchers and students seeking to learn how inference algorithms work.

BlackJAX stands as a valuable tool in the landscape of Bayesian computation. The impact of BlackJAX on the practice of Bayesian inference is already evident in its role as an enabler of innovative research, a tool for methodological advancements, and an educational aid. Looking ahead, our roadmap includes practical goals such as expanding the algorithm portfolio to include techniques like parallel and sequential tempering SMC, unbiased MCMC with couplings, and structured variational inference methods like INLA. We also recognize the importance of documentation, integration with probabilistic programming languages, and the introduction of inference databases for improved accessibility and transparency. BlackJAX remains an open and collaborative project, welcoming contributions from the community to ensure its continued growth and utility in Bayesian analysis.

Chapter 8

Conclusions

This thesis addresses several challenges in Bayesian computation. First, methodological contributions bridge the gap between fast-density approximation techniques using numerical optimization and the traditional asymptotically exact Monte Carlo methods. Furthermore, we presented a contemporary software suite to facilitate the use and development of classic and new algorithms. We also introduced a novel algorithm tailored for inference in linear variable selection models with robust variance, enhancing the toolkit available for statisticians in this domain.

The first three chapters laid a comprehensive foundation by reviewing essential background literature. Chapter 2, in Monte Carlo methods, provides an introduction to fundamental techniques and various forms of Markov chain Monte Carlo methods. Chapter 3, in approximate or variational inference methods, discusses the Kullback-Leibler divergence and normalizing flows in their role of building flexible variational families.

Chapter 4 introduced Transport Elliptical Slice sampling, a novel algorithm for dimension-independent and gradient-free sampling from unnormalized target densities. It includes an adaptive scheme that learns a non-Gaussian approximation of the target, which aids sampling. Our method leverages parallel computer architectures to expedite sampling from posterior distributions. Our findings indicate that TESS can outperform gradient-based algorithms across various models.

Chapter 5 introduced a novel computational approach that integrates flow matching with sampling methods using a sequential scheme that learns the target probability path with samples from a Markov chain. We developed an adaptive tempering mechanism that effectively discovers multiple modes in the target distribution. Our method converges to a local optimum of the flow matching objective, as demonstrated through experiments in both synthetic and real-world scenarios.

Chapter 6 presented a nonparametric extension of popular Bayesian variable selection models that accounts for heterogeneity, outliers, and clustering effects. Our model is also extended to handle heavy-tailed data distributions. Using Gibbs sampling, our Dirichlet process variable selection model proved computationally efficient, and experimental results demonstrate improvements in predictive accuracy and efficiency in identifying key model attributes.

Chapter 7 introduced BlackJAX, an established Bayesian computation library that adopts a functional approach to algorithm design and offers a repository of basic components. BlackJAX enables users to develop and experiment with new inference algorithms, offers broad support for various sampling and approximate inference methods, and is coupled with a user-friendly API, making it a powerful resource for experienced researchers and students.

8.1 Future work

There remains considerable scope for further research. This section outlines several promising directions for future work to enhance the flexibility, efficiency, and applicability of the methodologies and tools presented.

Developing flexible transport maps and low-variance Monte Carlo approximations of the Kullback-Leibler divergence remains a critical challenge for high-dimensional models. Future research will focus on refining transport maps to address the complex geometries of Bayesian posterior distributions. By creating more adaptable transport

maps, we can better capture the intricacies of high-dimensional spaces, leading to more accurate and reliable inference outcomes.

This is particularly relevant for tailored continuous normalizing flows, which allow more modelling flexibility than their discrete counterparts. By exploiting the unique characteristics of different models, we can enhance the construction of these flows. Extending the application of our methods to diverse fields such as bioinformatics, finance, and environmental science could yield substantial insights, particularly when dealing with large-scale data or intricate model structures.

This research also opens up opportunities to extend variable selection by incorporating additional models, such as the regularized horseshoe model, providing more robust techniques. Furthermore, exploring other nonparametric priors, like the Pitman-Yor process, offers more flexible assumptions regarding clustering effects. These extensions would enable more robust modelling of variable selection in complex data environments.

We also have ambitious goals for the continued development of BlackJAX. We aim to expand its algorithm portfolio to include advanced techniques such as parallel and sequential tempering SMC, unbiased MCMC with couplings, and structured variational inference methods like INLA. Improving documentation, ensuring seamless integration with probabilistic programming languages, and introducing inference databases for better accessibility and transparency are also key priorities. BlackJAX will remain an open and collaborative project, encouraging community contributions to foster its growth and maintain its relevance in Bayesian computation.

These future research directions promise to build on the foundations in this thesis, ensuring that the tools and methods we develop continue to meet the evolving needs of researchers and practitioners in this dynamic field.

Appendix A

BlackJAX: Composable Bayesian inference in JAX

A.1 Skeletons

In this appendix, we present the basic skeleton of sampling and approximate inference algorithms:

```
from typing import Callable, NamedTuple, Tuple

import jax

# import basic compoments that are already implemented
# or that you have implemented with a general structure
# for example, if you do a Metropolis-Hastings accept/reject step:
import blackjax.mcmc.proposal as proposal
from blackjax.base import MCMCSamplingAlgorithm
from blackjax.types import PRNGKey, PyTree


class SamplingAlgoState(NamedTuple):
    """State of your sampling algorithm.
    """
    position: PyTree
    ...


class SamplingAlgoInfo(NamedTuple):
    """Additional information on your algorithm transition.
```

```
    """
    ...


def init(position: PyTree, logdensity_fn: Callable, *args, **kwargs):
    # build an inital state
    state = SamplingAlgoState(...)
    return state


def build_kernel(*args, **kwargs):

    def kernel(
        rng_key: PRNGKey,
        state: SamplingAlgoState,
        logdensity_fn: Callable,
        *args,
        **kwargs,
    ) -> Tuple[SamplingAlgoState, SamplingAlgoInfo]:
        """Generate a new sample with the sampling kernel."""

        # build everything you'll need
        proposal_generator = sampling_algorithm_proposal(...)

        # generate pseudorandom keys
        key_other, key_proposal = jax.random.split(rng_key, 2)

        # generate the proposal with all its parts
        proposal, info = proposal_generator(key_proposal, ...)
        proposal = SamplingAlgoState(...)

        return proposal, info

    return kernel


class sampling_algorithm:
    """Implements the (basic) user interface for your sampling kernel.
    """

    init = staticmethod(init)
    build_kernel = staticmethod(build_kernel)

    def __new__(  # type: ignore[misc]
        cls,
        logdensity_fn: Callable,
        *args,
        **kwargs,
```

```
    ) -> MCMCSamplingAlgorithm:
        kernel = cls.build_kernel(...)

        def init_fn(position: PyTree):
            return cls.init(position, logdensity_fn, ...)

        def step_fn(rng_key: PRNGKey, state):
            return kernel(
                rng_key,
                state,
                logdensity_fn,
                ...,
            )

        return MCMCSamplingAlgorithm(init_fn, step_fn)


# and other functions that help make ‘init‘ and/or ‘build_kernel‘ easier to read and understand
def sampling_algorithm_proposal(*args, **kwags) -> Callable:

    # as an example, a symmetric Metropolis-Hastings step would look like this:
    init_proposal, generate_proposal = proposal.proposal_generator(...)
    sample_proposal = proposal.static_binomial_sampling(...)

    def generate(rng_key, state):
        # propose a new sample
        proposal_state = ...

        # accept or reject the proposed sample
        proposal = init_proposal(state)
        new_proposal, is_diverging = generate_proposal(proposal.energy, proposal_state)
        sampled_proposal, *info = sample_proposal(rng_key, proposal, new_proposal)

        # build a new state and collect useful information
        sampled_state, info = ...

        return sampled_state, info

    return generate
```

Listing A.1: Basic skeleton of a sampling algorithm

```
from typing import Callable, NamedTuple, Tuple

import jax
from optax import GradientTransformation
```

```

# import basic compoments that are already implemented
# or that you have implemented with a general structure
from blackjax.base import VIAlgorithm
from blackjax.types import PRNGKey, PyTree


class ApproxInfState(NamedTuple):
    """State of your approximate inference algorithm.
    """
    optim_state: Callable
    ...


class ApproxInfInfo(NamedTuple):
    """Additional information on your algorithm transition.
    """
    ...


def init(position: PyTree, logdensity_fn: Callable, *args, **kwargs):
    # build an inital state
    state = ApproxInfState(...)
    return state


def step(
    rng_key: PRNGKey,
    state: ApproxInfInfo,
    logdensity_fn: Callable,
    optimizer: GradientTransformation,
    *args,
    **kwargs,
) -> Tuple[ApproxInfState, ApproxInfInfo]:
    """Approximate the target density using your approximation.
    """
    # extract the previous parameters from the state
    params = ...
    # generate pseudorandom keys
    key_other, key_update = jax.random.split(rng_key, 2)
    # update the parameters and build a new state
    new_state = ApproxInfState(...)
    info = ApproxInfInfo(...)

    return new_state, info


def sample(rng_key: PRNGKey, state: ApproxInfState, num_samples: int = 1):
```

```
    """Sample from your approximation."""
    # the sample should be a PyTree of the same structure as the ‘position‘ in
    the init function
    samples = ...
    return samples


class approx_algorithm:
    """Implements the (basic) user interface for your approximate inference
    method.
    """

    init = staticmethod(init)
    step = staticmethod(step)
    sample = staticmethod(sample)

    def __new__(  # type: ignore[misc]
        cls,
        logdensity_fn: Callable,
        optimizer: GradientTransformation,
        *args,
        **kwargs,
    ) -> VIAlgorithm:
        def init_fn(position: PyTree):
            return cls.init(position, optimizer, ...)

        def step_fn(rng_key: PRNGKey, state):
            return cls.step(
                rng_key,
                state,
                logdensity_fn,
                optimizer,
                ...,
            )

        def sample_fn(rng_key: PRNGKey, state, num_samples):
            return cls.sample(rng_key, state, num_samples)

        return VIAlgorithm(init_fn, step_fn, sample_fn)


# other functions that help make ‘init‘,‘ ‘step‘ and/or ‘sample‘ easier to read
    and understand
```

Listing A.2: Basic skeleton of an approximate inference algorithm

References

Abadi, Martín et al. (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467*.

Abril-Pla, Oriol et al. (2023). "PyMC: a modern, and comprehensive probabilistic programming framework in Python". In: *PeerJ Computer Science* 9, e1516.

Aldous, David J (1985). "Exchangeability and related topics". In: *École d'Été de Probabilités de Saint-Flour XIII—1983*. Springer, pp. 1–198.

Bacon, Francis (1620). *Instauratio Magna (Novum Organum)*. Publisher Unknown.

Balkenhol, L et al. (2024). "CANDL: Cosmic Microwave Background Analysis with a Differentiable Likelihood". In: *arXiv preprint arXiv:2401.13433*.

Bayes, Thomas (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". In: *Philosophical transactions of the Royal Society of London* 53, pp. 370–418.

Beal, Matthew James (2003). *Variational algorithms for approximate Bayesian inference*. University of London, University College London (United Kingdom).

Besag, Julian (1994). "Comments on "Representations of knowledge in complex systems" by U. Grenander and M. I. Miller". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 56.591-592, p. 4.

Beskos, A et al. (2016). "On the convergence of adaptive sequential Monte Carlo methods". In: *Annals of Applied Probability* 26.2, pp. 1111–1146.

Bhattacharya, Anirban, Antik Chakraborty, and Bani K Mallick (2016). "Fast sampling with Gaussian scale mixture priors in high-dimensional regression". In: *Biometrika*, asw042.

Bingham, Eli et al. (2019). "Pyro: Deep universal probabilistic programming". In: *The Journal of Machine Learning Research* 20.1, pp. 973–978.

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518, pp. 859–877.

Bogachev, Vladimir Igorevich, Aleksandr Viktorovich Kolesnikov, and Kirill Vladimirovich Medvedev (2005). "Triangular transformations of measures". In: *Sbornik: Mathematics* 196.3, p. 309.

Bonnet, Clément et al. (2023). "Jumanji: a Diverse Suite of Scalable Reinforcement Learning Environments in JAX". In: *arXiv preprint arXiv:2306.09884*. URL: https://arxiv.org/abs/2306.09884.

Botev, Zdravko and Ad Ridder (2017). "Variance reduction". In: *Wiley statsRef: Statistics reference online*, pp. 1–6.

Bradbury, James et al. (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. URL: http://github.com/google/jax.

Breiman, Leo et al. (2001). "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3, pp. 199–231.

Cabezas, Alberto and Christopher Nemeth (2023). "Transport elliptical slice sampling". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 3664–3676.

Carpenter, Bob et al. (2017). "Stan: A probabilistic programming language". In: *Journal of statistical software* 76.

Carvalho, Carlos M, Nicholas G Polson, and James G Scott (2009). "Handling sparsity via the horseshoe". In: *Artificial Intelligence and Statistics*, pp. 73–80.

— (2010). "The horseshoe estimator for sparse signals". In: *Biometrika* 97.2, pp. 465–480.

Chen, Ricky and Yaron Lipman (2024). "Flow Matching on General Geometries". In: *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Chen, Ricky TQ et al. (2018). "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31.

Chopin, Nicolas (2002). "A sequential particle filter method for static models". In: *Biometrika* 89.3, pp. 539–552.

Cooper, Alex et al. (2023). "Bayesian cross-validation by parallel Markov Chain Monte Carlo". In: *arXiv preprint arXiv:2310.07002*.

Corenflos, Adrien, Nicolas Chopin, and Simo Särkkä (2022). "De-Sequentialized Monte Carlo: a parallel-in-time particle smoother". In: *The Journal of Machine Learning Research* 23.1, pp. 12923–12961.

DeepMind et al. (2020). *The DeepMind JAX Ecosystem*. URL: http://github.com/google-deepmind.

Del Moral, Pierre, Arnaud Doucet, and Ajay Jasra (2006a). "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 68.3, pp. 411–436. DOI: https://doi.org/10.1111/j.1467-9868.2006.00553.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2006.00553.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2006.00553.x.

— (2006b). "Sequential monte carlo samplers". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 68.3, pp. 411–436.

Depaoli, Sarah, James P. Clifton, and Patrice R. Cobb (2016). "Just Another Gibbs Sampler (JAGS): Flexible Software for MCMC Implementation". In: *Journal of Educational and Behavioral Statistics* 41.6, pp. 628–649. DOI: 10.3102/1076998616664876. eprint: https://doi.org/10.3102/1076998616664876. URL: https://doi.org/10.3102/1076998616664876.

Devroye, Luc (2006). "Nonuniform random variate generation". In: *Handbooks in operations research and management science* 13, pp. 83–121.

Dillon, Joshua V. et al. (2017). "TensorFlow Distributions". In: *arXiv preprint arXiv:1711.10604*.

Dinh, Laurent, David Krueger, and Yoshua Bengio (2014). "Nice: Non-linear independent components estimation". In: *arXiv preprint arXiv:1410.8516*.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2016). "Density estimation using real nvp". In: *arXiv preprint arXiv:1605.08803*.

Duane, Simon et al. (1987). "Hybrid Monte Carlo". In: *Physics letters B* 195.2, pp. 216–222.

Durkan, Conor et al. (2019). "Neural spline flows". In: *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*. Vancouver, Canada.

El Moselhy, Tarek A and Youssef M Marzouk (2012). "Bayesian inference with optimal maps". In: *Journal of Computational Physics* 231.23, pp. 7815–7850.

Escobar, Michael D and Mike West (1995). "Bayesian density estimation and inference using mixtures". In: *Journal of the american statistical association* 90.430, pp. 577–588.

Fagan, Francois, Jalaj Bhandari, and John P Cunningham (2016). "Elliptical Slice Sampling with Expectation Propagation." In: *UAI*.

Ferguson, Thomas S (1973). "A Bayesian analysis of some nonparametric problems". In: *The annals of statistics*, pp. 209–230.

Foreman-Mackey, Daniel et al. (2019). "EMCEE v3: A Python ensemble sampling toolkit for affine-invariant MCMC". In: *arXiv preprint arXiv:1911.07688*.

Gabrié, Marylou, Grant M Rotskoff, and Eric Vanden-Eijnden (2022). "Adaptive Monte Carlo augmented with normalizing flows". In: *Proceedings of the National Academy of Sciences* 119.10, e2109420119.

Galan, Aymeric et al. (2022). "Using wavelets to capture deviations from smoothness in galaxy-scale strong lenses". In: *Astronomy & Astrophysics* 668, A155.

Gelfand, Alan E and Adrian FM Smith (1990). "Sampling-based approaches to calculating marginal densities". In: *Journal of the American statistical association* 85.410, pp. 398–409.

Gelman, Andrew, Walter R Gilks, and Gareth O Roberts (1997). "Weak convergence and optimal scaling of random walk Metropolis algorithms". In: *The annals of applied probability* 7.1, pp. 110–120.

Geman, Stuart and Donald Geman (1984). "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". In: *IEEE Transactions on pattern analysis and machine intelligence* 6, pp. 721–741.

George, Edward I and Robert E McCulloch (1993). "Variable selection via Gibbs sampling". In: *Journal of the American Statistical Association* 88.423, pp. 881–889.

Geyer, Charles J (1992). "Practical markov chain monte carlo". In: *Statistical science*, pp. 473–483.

Ghosal, Subhashis and Aad Van der Vaart (2017). *Fundamentals of nonparametric Bayesian inference*. Vol. 44. Cambridge University Press.

Goel, Narendra S, Samaresh C Maitra, and Elliott W Montroll (1971). "On the Volterra and other nonlinear models of interacting populations". In: *Reviews of modern physics* 43.2, p. 231.

Gorham, Jackson and Lester Mackey (2017). "Measuring sample quality with kernels". In: *International Conference on Machine Learning*. PMLR, pp. 1292–1301.

Gorinova, Maria, Dave Moore, and Matthew Hoffman (2020). "Automatic reparameterisation of probabilistic programs". In: *International Conference on Machine Learning*. PMLR, pp. 3648–3657.

Graham, Matthew M. (n.d.). *Mici: Manifold Markov chain Monte Carlo methods in Python*. DOI: 10.5281/zenodo.3517301. URL: https://github.com/matt-graham/mici.

Grathwohl, Will et al. (2018). "Ffjord: Free-form continuous dynamics for scalable reversible generative models". In: *arXiv preprint arXiv:1810.01367*.

Grenander, Ulf and Michael I Miller (1994). "Representations of knowledge in complex systems". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 56.4, pp. 549–581.

Gretton, Arthur et al. (2012). "A Kernel Two-Sample Test". In: *Journal of Machine Learning Research (JMLR)* 13, pp. 723–773.

Gu, Ming Gao and Fan Hui Kong (1998). "A stochastic approximation algorithm with Markov chain Monte-Carlo method for incomplete data estimation problems". In: *Proceedings of the National Academy of Sciences* 95.13, pp. 7270–7274.

Hastings, W. K. (Apr. 1970a). "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1, pp. 97–109. ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97. eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf. URL: https://doi.org/10.1093/biomet/57.1.97.

— (Apr. 1970b). "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1, pp. 97–109. ISSN: 0006-3444. eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/8545904/57-1-97.pdf.

Hjort, N. J. et al. (2010). *Bayesian Nonparametrics*. Cambridge University Press.

Hoffman, Matthew, Alexey Radul, and Pavel Sountsov (2021). "An adaptive-mcmc scheme for setting trajectory lengths in Hamiltonian Monte Carlo". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 3907–3915.

Hoffman, Matthew, Pavel Sountsov, et al. (2019). "Neutra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport". In: *arXiv preprint arXiv:1903.03704*.

Hoffman, Matthew D, Andrew Gelman, et al. (2014). "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *J. Mach. Learn. Res.* 15.1, pp. 1593–1623.

Hoffman, Matthew D and Pavel Sountsov (2022). "Tuning-free generalized Hamiltonian Monte Carlo". In: *International conference on artificial intelligence and statistics*. PMLR, pp. 7799–7813.

Hoogeboom, Emiel, Rianne Van Den Berg, and Max Welling (2019). "Emerging convolutions for generative normalizing flows". In: *International conference on machine learning*. PMLR, pp. 2771–2780.

Horowitz, Alan M (1991). "A generalized guided Monte Carlo algorithm". In: *Physics Letters B* 268.2, pp. 247–252.

Hutchinson, Michael F (1989). "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines". In: *Communications in Statistics-Simulation and Computation* 18.3, pp. 1059–1076.

Ishwaran, H. and J. S. Rao (2005). "Spike and Slab Variable Selection: Frequentist and Bayesian Strategies". In: *The Annals of Statistics* 33.2, pp. 730–773. DOI: 10.1214/009053604000001147.

Jacob, Pierre E, John O'Leary, and Yves F Atchadé (2020). "Unbiased Markov chain Monte Carlo methods with couplings". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82.3, pp. 543–600.

Jasra, Ajay, David A Stephens, and Christopher C Holmes (2007). "On population-based simulation for static inference". In: *Statistics and Computing* 17, pp. 263–279.

Jeffreys, Harold (1946). "An invariant form for the prior probability in estimation problems". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 186.1007, pp. 453–461.

Johnson, Leif T and Charles J Geyer (2012). "Variable transformation to obtain geometric ergodicity in the random-walk Metropolis algorithm". In: *The Annals of Statistics*, pp. 3050–3076.

Jordan, Michael I et al. (1999). "An introduction to variational methods for graphical models". In: *Machine learning* 37, pp. 183–233.

Joyce, James M. (2011). "Kullback-Leibler Divergence". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327.

Karamanis, Minas, Florian Beutler, et al. (2022). "Accelerating astronomical and cosmological inference with preconditioned Monte Carlo". In: *Monthly Notices of the Royal Astronomical Society* 516.2, pp. 1644–1653.

Karamanis, Minas, David Nabergoj, et al. (2022). "pocoMC: A Python package for accelerated Bayesian inference in astronomy and cosmology". In: *arXiv preprint arXiv:2207.05660*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kingma, Durk P and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in neural information processing systems* 31.

Kingma, Durk P, Tim Salimans, et al. (2016). "Improved variational inference with inverse autoregressive flow". In: *Advances in neural information processing systems* 29.

Kobyzev, Ivan, Simon JD Prince, and Marcus A Brubaker (2020). "Normalizing flows: An introduction and review of current methods". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11, pp. 3964–3979.

Kolmogorov, Andrey (1933). "Grundbegriffe der wahrscheinlichkeitsrechnung". In.

Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency". In: *The annals of mathematical statistics* 22.1, pp. 79–86.

Kumar, Ravin et al. (2019). "ArviZ a unified library for exploratory analysis of Bayesian models in Python". In: *Journal of Open Source Software* 4.33, p. 1143. DOI: 10.21105/joss.01143. URL: https://doi.org/10.21105/joss.01143.

Lao, Junpeng et al. (2020). "tfp.mcmc: Modern Markov chain Monte Carlo Tools Built for Modern Hardware". In: *arXiv preprint arXiv:2002.01184*.

Lemieux, Christiane (2014). "Control variates". In: *Wiley StatsRef: Statistics Reference Online*, pp. 1–8.

Linnainmaa, Seppo (1976). "Taylor expansion of the accumulated rounding error". In: *BIT Numerical Mathematics* 16.2, pp. 146–160.

Lipman, Yaron et al. (2022). "Flow matching for generative modeling". In: *arXiv preprint arXiv:2210.02747*.

Liu, Qiang, Jason Lee, and Michael Jordan (2016). "A kernelized Stein discrepancy for goodness-of-fit tests". In: *International conference on machine learning*. PMLR, pp. 276–284.

Lo, Albert Y (1984). "On a class of Bayesian nonparametric estimates: I. Density estimates". In: *The annals of statistics*, pp. 351–357.

Lunn, David J et al. (2000). "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility". In: *Statistics and computing* 10, pp. 325–337.

Makalic, Enes and Daniel F Schmidt (2015a). "A simple sampler for the horseshoe estimator". In: *IEEE Signal Processing Letters* 23.1, pp. 179–182.

— (2015b). "A simple sampler for the horseshoe estimator". In: *IEEE Signal Processing Letters* 23.1, pp. 179–182.

Marbach, Daniel et al. (2010). "Revealing strengths and weaknesses of methods for gene network inference". In: *Proceedings of the national academy of sciences* 107.14, pp. 6286–6291.

Marzouk, Youssef et al. (2016). "An introduction to sampling via measure transport". In: *arXiv preprint arXiv:1602.05023*.

McLachlan, Robert I (1995). "On the numerical integration of ordinary differential equations by symmetric composition methods". In: *SIAM Journal on Scientific Computing* 16.1, pp. 151–168.

Metropolis, Nicholas et al. (1953). "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6, pp. 1087–1092.

Meyer, Renate and Jun Yu (2000). "BUGS for a Bayesian analysis of stochastic volatility models". In: *The Econometrics Journal* 3.2, pp. 198–215.

Meyn, Sean P and Richard L Tweedie (2012). *Markov chains and stochastic stability*. Springer Science & Business Media.

Midgley, Laurence Illing et al. (2022). "Flow annealed importance sampling bootstrap". In: *arXiv preprint arXiv:2208.01893*.

Mitchell, Toby J and John J Beauchamp (1988). "Bayesian variable selection in linear regression". In: *Journal of the american statistical association* 83.404, pp. 1023–1032.

Møller, Jesper, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen (1998). "Log gaussian cox processes". In: *Scandinavian journal of statistics* 25.3, pp. 451–482.

Murphy, Kevin P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press. URL: probml.ai.

Murray, Iain, Ryan Adams, and David MacKay (2010). "Elliptical slice sampling". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 541–548.

Naesseth, Christian, Fredrik Lindsten, and David Blei (2020). "Markovian score climbing: Variational inference with KL (p—— q)". In: *Advances in Neural Information Processing Systems* 33, pp. 15499–15510.

Natarovskii, Viacheslav, Daniel Rudolf, and Björn Sprungk (2021). "Geometric convergence of elliptical slice sampling". In: *International Conference on Machine Learning*. PMLR, pp. 7969–7978.

Neal, R. (1998). "Regression and classification using Gaussian process priors". In: *Bayesian statistics*. Vol. 6, p. 475.

Neal, R. M. (2000). "Markov Chain Sampling Methods for Dirichlet Process Mixture Models". In: *Journal of Computational and Graphical Statistics* 9.2, pp. 249–265.

Neal, Radford M (2001). "Annealed importance sampling". In: *Statistics and computing* 11, pp. 125–139.

— (2003). "Slice sampling". In: *The annals of statistics* 31.3, pp. 705–767.

Neal, Radford M et al. (2011). "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo* 2.11, p. 2.

Neal, Radford M (2020). "Non-reversibly updating a uniform [0, 1] value for Metropolis accept/reject decisions". In: *arXiv preprint arXiv:2001.11950*.

Nishihara, Robert, Iain Murray, and Ryan P Adams (2014). "Parallel MCMC with generalized elliptical slice sampling". In: *The Journal of Machine Learning Research* 15.1, pp. 2087–2112.

Oriol, Abril-Pla et al. (2023). "PyMC: A Modern and Comprehensive Probabilistic Programming Framework in Python". In: *PeerJ Computer Science* 9, e1516. DOI: 10.7717/peerj-cs.1516.

Papamakarios, George, Theo Pavlakou, and Iain Murray (2017). "Masked autoregressive flow for density estimation". In: *Advances in neural information processing systems* 30.

Papaspiliopoulos, Omiros, Gareth O Roberts, and Martin Sköld (2007). "A general framework for the parametrization of hierarchical models". In: *Statistical Science*, pp. 59–73.

Parno, Matthew D and Youssef M Marzouk (2018). "Transport map accelerated Markov Chain Monte Carlo". In: *SIAM/ASA Journal on Uncertainty Quantification* 6.2, pp. 645–682.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 8024–8035.

Peskun, Peter H (1973). "Optimum monte-carlo sampling using markov chains". In: *Biometrika* 60.3, pp. 607–612.

Phan, Du, Neeraj Pradhan, and Martin Jankowiak (2019). "Composable effects for flexible and accelerated probabilistic programming in NumPyro". In: *arXiv preprint arXiv:1912.11554*.

Picard, J. and J. Pitman (2006). *Combinatorial Stochastic Processes: Ecole d'Eté de Probabilités de Saint-Flour XXXII - 2002*. Lecture Notes in Mathematics. Springer Berlin Heidelberg. ISBN: 9783540342663.

Piironen, Juho, Aki Vehtari, et al. (2017). "Sparsity information and regularization in the horseshoe and other shrinkage priors". In: *Electronic Journal of Statistics* 11.2, pp. 5018–5051.

Pinder, Thomas and Daniel Dodd (2022). "GPJax: A Gaussian process framework in JAX". In: *Journal of Open Source Software* 7.75, p. 4455. DOI: 10.21105/joss.04455. URL: https://doi.org/10.21105/joss.04455.

Pitman, Jim and Marc Yor (1997). "The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator". In: *The Annals of Probability*, pp. 855–900.

Price-Whelan, Adrian M et al. (2024). "Data-driven Dynamics with Orbital Torus Imaging: A Flexible Model of the Vertical Phase Space of the Galaxy". In: *arXiv preprint arXiv:2401.07903*.

Radul, Alexey et al. (2020). "Automatically batching control-intensive programs for modern accelerators". In: *Proceedings of Machine Learning and Systems* 2, pp. 390–399.

Rezende, Danilo and Shakir Mohamed (2015). "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR, pp. 1530–1538.

Robert, Christian P. (2016). "The Metropolis–Hastings algorithm". In: *arXiv preprint arXiv:1504.01896*.

Robert, Christian P. and George Casella (2004). *Monte Carlo statistical methods*. Springer Verlag.

Roberts, G.O. and A.F.M. Smith (1994). "Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms". In: *Stochastic Processes and their Applications* 49.2, pp. 207–216. ISSN: 0304-4149. DOI: https://doi.org/10.1016/0304-4149(94)90134-1. URL: https://www.sciencedirect.com/science/article/pii/0304414994901341.

Roberts, Gareth O and Jeffrey S Rosenthal (1998). "Optimal scaling of discrete approximations to Langevin diffusions". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60.1, pp. 255–268.

— (2001). "Optimal scaling for various Metropolis–Hastings algorithms". In: *Statistical science* 16.4, pp. 351–367.

Rosenblatt, Murray (1952). "Remarks on a multivariate transformation". In: *The annals of mathematical statistics* 23.3, pp. 470–472.

Rossky, Peter J, Jimmie D Doll, and Harold L Friedman (1978). "Brownian dynamics as smart Monte Carlo simulation". In: *The Journal of Chemical Physics* 69.10, pp. 4628–4633.

Rue, Håvard, Sara Martino, and Nicolas Chopin (2009). "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 71.2, pp. 319–392.

Salvatier, John, Thomas V Wiecki, and Christopher Fonnesbeck (2016). "Probabilistic programming in Python using PyMC3". In: *PeerJ Computer Science* 2, e55.

Saul, Lawrence K, Tommi Jaakkola, and Michael I Jordan (1996). "Mean field theory for sigmoid belief networks". In: *Journal of artificial intelligence research* 4, pp. 61–76.

Schoenholz, Samuel S. and Ekin D. Cubuk (2020). "JAX M.D. A Framework for Differentiable Physics". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc. URL: https://papers.nips.cc/paper/2020/file/83d3d4b6c9579515e1679aca8cbc8033-Paper.pdf.

Schoot, Rens van de et al. (2021). "Bayesian statistics and modelling". In: *Nature Reviews Methods Primers* 1.1, p. 1.

Sountsov, Pavel, Alexey Radul, and Srinivas Vasudevan (2021). "FunMC: A functional API for building Markov Chains". In: *arXiv preprint arXiv:2001.05035*.

Speagle, Joshua S (2020). "DYNESTY: a dynamic nested sampling package for estimating Bayesian posteriors and evidences". In: *Monthly Notices of the Royal Astronomical Society* 493.3, pp. 3132–3158.

Staber, Brian and Sébastien Da Veiga (2022). "Benchmarking Bayesian neural networks and evaluation metrics for regression tasks". In: *arXiv preprint arXiv:2206.06779*.

Stein, Charles et al. (2004). "Use of exchangeable pairs in the analysis of simulations". In: *Lecture Notes-Monograph Series*, pp. 1–26.

Steinke, Florian, Matthias Seeger, and Koji Tsuda (2007). "Experimental design for efficient identification of gene regulatory networks using sparse Bayesian models". In: *BMC systems biology* 1.1, pp. 1–15.

Stoltz, Gabriel, Mathias Rousset, et al. (2010). *Free energy computations: A mathematical perspective*. World Scientific.

Tancik, Matthew et al. (2020). "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Advances in Neural Information Processing Systems* 33, pp. 7537–7547.

Tomczak, Jakub M and Max Welling (2016). "Improving variational auto-encoders using householder flow". In: *arXiv preprint arXiv:1611.09630*.

Tong, Alexander et al. (2023). "Conditional flow matching: Simulation-free dynamic optimal transport". In: *arXiv preprint arXiv:2302.00482*.

Tran, Dustin et al. (2019). "Bayesian Layers: A module for neural network uncertainty". In: *Neural Information Processing Systems*.

Vargas, Francisco, Will Grathwohl, and Arnaud Doucet (2023). "Denoising diffusion samplers". In: *arXiv preprint arXiv:2302.13834*.

Wilkinson, William J, Simo Särkkä, and Arno Solin (2023). "Bayes–Newton Methods for Approximate Bayesian Inference with PSD Guarantees". In: *Journal of Machine Learning Research* 24.83, pp. 1–50.

Wolff, Ulli, Alpha Collaboration, et al. (2004). "Monte Carlo errors with less errors". In: *Computer Physics Communications* 156.2, pp. 143–153.

Zhang, Qinsheng and Yongxin Chen (2021). "Path integral sampler: a stochastic control approach for sampling". In: *arXiv preprint arXiv:2111.15141*.