AUXILIARY VARIABLE MARKOV CHAIN MONTE CARLO METHODS

MATTHEW MCKENZIE GRAHAM

Doctor of Philosophy
University of Edinburgh
2017

Matthew Mckenzie Graham: *Auxiliary variable Markov chain Monte Carlo methods,* 2017.
Submitted for the degree of Doctor of Philosophy, University of Edinburgh.

SUPERVISOR:
Dr. Amos J. Storkey

DECLARATION

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

November 2017

Matthew Mckenzie Graham

ABSTRACT

Markov chain Monte Carlo (MCMC) methods are a widely applicable class of algorithms for estimating integrals in statistical inference problems. A common approach in MCMC methods is to introduce additional auxiliary variables into the Markov chain state and perform transitions in the joint space of target and auxiliary variables. In this thesis we consider novel methods for using auxiliary variables within MCMC methods to allow approximate inference in otherwise intractable models and to improve sampling performance in models exhibiting challenging properties such as multimodality.

We first consider the pseudo-marginal framework. This extends the Metropolis–Hastings algorithm to cases where we only have access to an unbiased estimator of the density of target distribution. The resulting chains can sometimes show 'sticking' behaviour where long series of proposed updates are rejected. Further the algorithms can be difficult to tune and it is not immediately clear how to generalise the approach to alternative transition operators. We show that if the auxiliary variables used in the density estimator are included in the chain state it is possible to use new transition operators such as those based on slice-sampling algorithms within a pseudo-marginal setting. This auxiliary pseudo-marginal approach leads to easier to tune methods and is often able to improve sampling efficiency over existing approaches.

As a second contribution we consider inference in probabilistic models defined via a generative process with the probability density of the outputs of this process only implicitly defined. The *approximate Bayesian computation* (ABC) framework allows inference in such models when conditioning on the values of observed model variables by making the approximation that generated observed variables are 'close' rather than exactly equal to observed data. Although making the inference problem more tractable, the approximation error introduced in ABC methods can be difficult to quantify and standard algorithms tend to perform poorly when conditioning on high dimensional observations. This often requires further approximation by reducing the observations to lower dimensional summary statistics.

We show how including all of the random variables used in generating model outputs as auxiliary variables in a Markov chain state can

allow the use of more efficient and robust MCMC methods such as slice sampling and *Hamiltonian Monte Carlo* (HMC) within an ABC framework. In some cases this can allow inference when conditioning on the full set of observed values when standard ABC methods require reduction to lower dimensional summaries for tractability. Further we introduce a novel constrained HMC method for performing inference in a restricted class of differentiable generative models which allows conditioning the generated observed variables to be arbitrarily close to observed data while maintaining computational tractability.

As a final topic we consider the use of an auxiliary temperature variable in MCMC methods to improve exploration of multimodal target densities and allow estimation of normalising constants. Existing approaches such as simulated tempering and annealed importance sampling use temperature variables which take on only a discrete set of values. The performance of these methods can be sensitive to the number and spacing of the temperature values used, and the discrete nature of the temperature variable prevents the use of gradient-based methods such as HMC to update the temperature alongside the target variables. We introduce new MCMC methods which instead use a continuous temperature variable. This both removes the need to tune the choice of discrete temperature values and allows the temperature variable to be updated jointly with the target variables within a HMC method.

LAY SUMMARY

Much of the information we receive about the world is uncertain. By implication the conclusions we draw from this noisy and incomplete information are also subject to uncertainty. Probability theory offers a consistent framework for describing the uncertainty in our beliefs about the world in the form of a probabilistic model and in making inferences about the variables in that model. Although the basic rules of probability which underlie the inference process are easy to state, for problems of even moderate complexity the calculations involved in performing inference are typically intractable to compute exactly as they involve an exhaustive iteration over a combinatorially large or even infinite number of possible configurations of the model variables.

This thesis is concerned with the development of efficient methods for approximate inference in complex probabilistic models. Such methods trade-off a loss of exactness for an increase in computational tractability. In particular we focus here on the topic of Markov chain Monte Carlo methods, which are a class of approaches for approximating the computations involved in inference. A noisy dynamic is constructed which explores the probability distribution on configurations of the model variables which are plausible given our observed information. The model variable values sampled by this dynamic can then be used to represent our beliefs about the model variables given the observed information and to efficiently approximate the calculations involved in inference by computing averages over the sampled values.

This thesis specifically considers methods which augment the space of model variables being explored by the dynamic with additional auxiliary variables. In some cases this allows the robustness or efficiency of the resulting sampling methods to be improved, for example by making it easier for the sampler to move between separated regions of high probability in the model variable space. In other settings redefining the state space of the problem can allow us to perform inference in settings where we do not have an explicit form for the distribution on the variables of interest, for example in simulator models where we can generate plausible values for the model variables but not necessarily express the probability distribution on those variables.

ACKNOWLEDGEMENTS

It has been a privilege to have had Amos Storkey as my supervisor. He has unerringly been able to provide helpful insight in to any problem I came to him with and the enthusiasm and humour he has brought to all our discussions has made it a joy to work with him. I also feel very fortunate to have had the chance to collaborate with Iain Murray on the work described in Chapter 3 and I am extremely grateful to him for all the additional helpful and perceptive advice, ideas and comments he has given me during my research. My thanks also to Peggy Seriès, Matthias Hennig and Mark van Rossum for providing helpful guidance at review meetings over the course of my research and to my examiners, Dennis Prangle and Lukasz Szpruch, for their patience in reviewing this thesis and for their useful comments and feedback.

I am fortunate to have been part of the Doctoral Training Centre (DTC) in Neuroinformatics and Computational Neuroscience during my studies in Edinburgh and I would like to thank the DTC executive committee members for all of the support and opportunities the DTC has provided me with. Many thanks also to Teresa Ironside, Mila Vukomanovic and Alison May-Edie for the excellent support they provided in their tenures in charge of DTC administrative duties. I am very grateful to the funding bodies of the DTC, the Engineering and Physical Sciences Research Council, Biotechnology and Biological Sciences Research Council and Medical Research Council, for their financial support of my research and travel. My thanks also to the Scottish Informatics and Computing Science Alliance and Disney Research for providing additional travel funding.

It has been a pleasure to work with and around such an interesting, friendly and diverse group of people at the School of Informatics. My particular thanks to my fellow DTC students for many interesting and thought-provoking discussions and journal club presentations but also for the friendships that have made my time in Edinburgh so enjoyable. I am also very grateful to the other members of Amos' research group and the wider Institute for Adaptive and Neural Computation machine learning group for numerous helpful paper discussions and whiteboard brainstorm sessions. Last but definitely not least - I could not have got to this point without the continual support and encouragement of my family and friends. My heartfelt thanks to all of you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ABC	approximate Bayesian computation
AD	automatic differentiation
ADVI	automatic differentiation variational inference
AIS	annealed importance sampling
APM	auxiliary pseudo-marginal
BBVI	black box variational inference
BDMC	bidirectional Monte Carlo
CT-HMC	continuously tempered Hamiltonian Monte Carlo
CDF	cumulative distribution function
CPU	central processing unit
CT	continuous tempering
ELBO	evidence lower bound
EP	expectation propagation
ESS	effective sample size
GAN	generative-adversarial network
GPU	graphics processing unit
HMC	Hamiltonian Monte Carlo
IID	independently and identically distributed
iff	if and only if
IWAE	importance weighted autoencoder
KL	Kullback–Leibler
MCMC	Markov chain Monte Carlo
MH	Metropolis–Hastings
MI	Metropolis independence
NUTS	no U-turn sampler
ODE	ordinary differential equation
PM	pseudo-marginal
PRNG	pseudo-random number generator
PSRF	potential scale reduction factor
RMHMC	Riemannian-manifold Hamiltonian Monte Carlo
RMSE	root mean squared error
SDE	stochastic differential equation
SMC	sequential Monte Carlo
SS	slice sampling
ST	simulated tempering
VAE	variational autoencoder
WRT	with respect to

1 INTRODUCTION

The actual science of logic is conversant at present only with things either certain, impossible, or entirely doubtful, none of which (fortunately) we have to reason on. Therefore the true logic for this world is the calculus of probabilities
—James Clerk Maxwell

Inference is the process of drawing conclusions from evidence. While deductive logic offers a framework for inferring conclusions from absolute statements of truth, it does not apply to the more typical real-world setting where the information we receive is uncertain. To make inferences under conditions of uncertainty, we must instead turn to probability theory, which both offers a consistent framework for quantifying our beliefs and making inferences given these beliefs.

The key computational task in inference is computing integrals with respect to probability distributions on the variables in a proposed model. Typically these integrals will not have analytic solutions and the large number of variables being integrated over mean numerical quadrature methods are impractically costly. In these cases we must resort to approximate methods which trade-off an introduction of error for an increase in computational tractability. *Markov chain Monte Carlo* (MCMC) methods are a very generally applicable class of approximate inference techniques which estimate the integrals of interest by computing averages over the states of a Markov chain.

The topic of this thesis is the development of MCMC methods. In particular we introduce several novel methods which exploit reparameterisations and augmentations of the state of a Markov chain to improve upon the computational efficiency, ease of use or degree of approximation error of existing approaches.

In this chapter we discuss the basic concepts of probabilistic modelling which underpin the inference methods discussed in later chapters. In particular we review the terminology and basic concepts of the measure-theoretic description of probability as some of the later results in the thesis are most clearly described within this framework. We also introduce graphical models as a compact way of visualising structure in probabilistic models. Finally we conclude with a discussion of the specific inference problems that the methods presented in the rest of this thesis are intended to help tackle.

1.1 PROBABILITY THEORY

A σ-algebra, $\mathcal{E}$, on a set S is set of subsets of S with $S \in \mathcal{E}$, $\emptyset \in \mathcal{E}$ and which is closed under complement and countable unions and intersections.

A *probability space* is defined as a triplet $(S, \mathcal{E}, \mathsf{P})$ where

- S is the *sample space*, the set of all possible outcomes,
- $\mathcal{E}$ is the *event space*, a σ-algebra on S, defining all possible events (measurable subsets of S),
- P is the *probability measure*, a finite measure satisfying $\mathsf{P}(S) = 1$, which specifies the probabilities of events in $\mathcal{E}$.

Kolmogorov's axioms:

1. *Non-negativity:* $\mathsf{P}(E) \geq 0 \; \forall E \in \mathcal{E}$,
2. *Normalisation:* $\mathsf{P}(S) = 1$,
3. *Countable additivity: for any countable set of disjoint events* $\{E_i\}_i : E_i \in \mathcal{F} \; \forall i$, $E_i \cap E_j = \emptyset \; \forall i \neq j$, $\mathsf{P}(\cup_i E_i) = \sum_i \mathsf{P}(E_i)$.

Given this definition of a probability space, Kolmogorov's axioms [142] can be used to derive a measure-theoretic formulation of probability theory. The probability of an event $E \in \mathcal{E}$ is defined as its measure $\mathsf{P}(E)$. Two events $A, B \in \mathcal{E}$ are *independent* if $\mathsf{P}(A \cap B) = \mathsf{P}(A)\mathsf{P}(B)$.

The key advantage of the measure-theoretic approach to probability is that it provides a consistent definition of the probability of an event in any space we can define a measure on. This allows a unified treatment of the common cases of probability distributions of discrete and continuous random variables but also makes it possible to consider distributions on more general spaces. In Chapter 4 we will consider problems which involve distributions on implicitly-defined manifolds where this generality will be key to understanding the proposed methods.

1.1.1 Random variables

If $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ are two measurable spaces, a function $f : X \to Y$ is measurable if $f^{-1}(E) \in \mathcal{F} \; \forall E \in \mathcal{G}$.

When modelling real-world processes, rather than considering events as subsets of an abstract sample space, it is usually more helpful to consider *random variables* which represent quantities in the model of interest. A random variable $\mathsf{x} : S \to X$ is defined as a measurable function from the sample space to a measurable space $(X, \mathcal{F})$.

The Borel σ-algebra $\mathscr{B}(\mathbb{R})$ is the smallest σ-algebra on $\mathbb{R}$ which contains all open real intervals.

Often X is the reals, $\mathbb{R}$, and $\mathcal{F}$ is the Borel σ-algebra on the reals, $\mathscr{B}(\mathbb{R})$, in which case we refer to a *real random variable*. It is also common to consider cases where X is a real vector space, $\mathbb{R}^D$, and $\mathcal{F} = \mathscr{B}(\mathbb{R}^D)$ - in this case refer to a *real random vector* and use the notation $\mathbf{x} : S \to X$. A final specific case is when X is countable and $\mathcal{F}$ is the power set $\mathscr{P}(X)$ in which case we refer to x as a *discrete random variable*. As we are most often concerned with real-valued random variables and vectors in this thesis, when it is unambiguous to do so we drop the 'real' qualifier and simply refer to *random variables* and *random vectors*.

Due to the definition of a random variable as a measurable function, we can define a pushforward measure on a random variable x

$$\mathsf{P}_\mathsf{x}(A) = \mathsf{P} \circ \mathsf{x}^{-1}(A) = \mathsf{P}(\{s \in S : \mathsf{x}(s) \in A\}) \quad \forall A \in \mathcal{F}. \tag{1.1}$$

If $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ are two measurable spaces, μ a measure on these spaces and $f : X \to Y$ a measurable function, the pushforward measure μ_f satisfies $\mu_f(A) = \mu \circ f^{-1}(A)$ $\forall A \in \mathcal{G}$.

The measure P_x specifies that the probability of the event that the random variable x takes a value in a measurable set $A \in \mathcal{F}$ is $\mathsf{P}_\mathsf{x}(A)$. We typically describe P_x as the *distribution* of x.

1.1.2 Joint and conditional probability

Typically we will jointly define multiple random variables on the same probability space. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space and $\mathsf{x} : S \to X$, $\mathsf{y} : S \to Y$ be two random variables with corresponding σ-algebras $\mathcal{F}$ and $\mathcal{G}$. Then the *joint probability* of x and y is defined as

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \mathsf{P}\left(\mathsf{x}^{-1}(A) \cap \mathsf{y}^{-1}(B)\right) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.2}$$

The joint probability is related to P_x and P_y by

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, Y) = \mathsf{P}_\mathsf{x}(A), \; \mathsf{P}_{\mathsf{x},\mathsf{y}}(X, B) = \mathsf{P}_\mathsf{y}(B) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.3}$$

In this context P_x and P_y are referred to as *marginal distributions* of the joint distribution. Two random variables x and y are said to be independent if and only if

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \mathsf{P}_\mathsf{x}(A)\,\mathsf{P}_\mathsf{y}(B) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.4}$$

A particularly key concept for inference is the definition of *conditional probability*. The conditional probability of an event $A \in \mathcal{E}$ occurring given another event $B \in \mathcal{E}$ has occurred is denoted $\mathsf{P}(A \mid B)$ and we have the definition

$$\mathsf{P}(A \mid B) = \frac{\mathsf{P}(A \cap B)}{\mathsf{P}(B)} \quad \forall A \in \mathcal{E}, B \in \mathcal{E} : \mathsf{P}(B) \neq 0. \tag{1.5}$$

In Kolmogorov's probability theory, (1.5) is given as an additional definition distinct from the basic axioms. In alternatives such as the work of Cox [65, 66] and de Finetti [88], conditional probabilities are instead viewed as a primitive.

Correspondingly we denote the conditional probability of the event of the random variable x taking a value in $A \in \mathcal{F}$ given the event that the random variable y takes a value in $B \in \mathcal{G}$ as $\mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \mid B)$. Using (1.5) and (1.2), $\mathsf{P}_{\mathsf{x}|\mathsf{y}}$ and $\mathsf{P}_{\mathsf{y}|\mathsf{x}}$ can be shown to satisfy

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \mid B)\,\mathsf{P}_\mathsf{y}(B) = \mathsf{P}_{\mathsf{y}|\mathsf{x}}(B \mid A)\,\mathsf{P}_\mathsf{x}(A) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.6}$$

This decomposition of a joint probability into a product of a conditional and marginal is sometimes referred to as the product rule. An implication of (1.6) is what is often termed *Bayes' theorem*

$$\mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \,|\, B) = \frac{\mathsf{P}_{\mathsf{y}|\mathsf{x}}(B \,|\, A)\, \mathsf{P}_{\mathsf{x}}(A)}{\mathsf{P}_{\mathsf{y}}(B)} \quad \forall A \in \mathcal{F},\, B \in \mathcal{G} : \mathsf{P}_{\mathsf{y}}(B) \neq 0, \tag{1.7}$$

which will be of key importance in the later discussion of inference.

The definition in (1.2) of the joint probability of a pair of random variables can be extended to arbitrarily large collections of random variables. Similarly conditional probabilities can be defined for collections of multiple jointly dependent random variables, with the product rule given in (1.6) generalising to a combinatorial number of possible factorisations of the joint probability. Graphical models offer a convenient way of representing the dependencies between large collections of random variables and any resulting factorisation structure in their joint probability, and are discussed in Section 1.2.

1.1.3 Probability densities

So far we have not specified how the probability measure P is defined and by consequence the probability (distribution) of a random variable. The Radon–Nikodym theorem guarantees that for a pair of $\sigma-$finite measures μ and ν on a measurable space $(X, \mathcal{F})$ where ν is absolutely continuous with respect to μ, then there is a unique (up to μ-null sets) measurable function $f : X \to [0, \infty)$ termed a *density* such that

A measure on X is σ-finite if X is a countable union of finite measure sets.

$$\nu(A) = \int_A f \,\mathrm{d}\mu = \int_A f(x)\, \mu(\mathrm{d}x) \quad \forall A \in \mathcal{F}. \tag{1.8}$$

If μ and ν are measures on a measurable space $(X, \mathcal{F})$ then ν has absolute continuity WRT to μ if $\forall A \in \mathcal{F}$, $\mu(A) = 0 \Rightarrow \nu(A) = 0$.

The two Lebesgue integral notations above are equivalent and we will use them interchangeably. The density function f is also termed the *Radon-Nikodym derivative* of ν with respect to μ, denoted $\frac{\mathrm{d}\nu}{\mathrm{d}\mu}$. Density functions therefore represent a convenient way to define a probability distribution with respect to a reference measure we will term the *base measure*. The key requirement defining what is an appropriate base measure to use is that the probability measure of interest is absolutely continuous with respect to it.

It can also be shown that if $f = \frac{\mathrm{d}\nu}{\mathrm{d}\mu}$ and g is a measurable function

$$\int_X g(x)\, \nu(\mathrm{d}x) = \int_X g(x)\, f(x)\, \mu(\mathrm{d}x), \tag{1.9}$$

which we will use later when discussing calculation of expectations.

Real random variables will typically have a distribution P_x defined by a probability density $\mathsf{p}_\mathsf{x} : \mathbb{R} \to [0, \infty)$ with respect to the *Lebesgue measure*, λ, on $\mathbb{R}$,

$$\mathsf{P}_\mathsf{x}(A) = \int_A \mathsf{p}_\mathsf{x}(x)\, \lambda(\mathrm{d}x) = \int_A \mathsf{p}_\mathsf{x}(x)\, \mathrm{d}x \qquad \forall A \in \mathscr{B}(\mathbb{R}). \tag{1.10}$$

The Lebesgue measure assigns a measure to subsets of a Euclidean space, and for $\mathbb{R}$, $\mathbb{R}^2$ and $\mathbb{R}^3$ formalises the intuitive concepts of length, area and volume of subsets respectively.

Analogously for a random vector $\mathbf{x}$ with density $\mathsf{p}_\mathbf{x} : \mathbb{R}^D \to [0, \infty)$ with respect to the D-dimensional Lebesgue measure λ^D we have that

$$\mathsf{P}_\mathbf{x}(A) = \int_A \mathsf{p}_\mathbf{x}(\boldsymbol{x})\, \lambda^D(\mathrm{d}\boldsymbol{x}) = \int_A \mathsf{p}_\mathbf{x}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \qquad \forall A \in \mathscr{B}(\mathbb{R}^D). \tag{1.11}$$

The notation in the second equalities in (1.10) and (1.11) uses a convention that will be used throughout this thesis that integrals without an explicit measure are with respect to the Lebesgue measure.

The probability distribution of a discrete random variable can be defined via probability density $\mathsf{p}_\mathsf{x} : X \to [0, 1]$ with respect to the *counting measure* #,

$$\mathsf{P}_\mathsf{x}(A) = \int_A \mathsf{p}_\mathsf{x}(x)\, \#(\mathrm{d}x) = \sum_{x \in A} \mathsf{p}_\mathsf{x}(x) \qquad \forall A \in \mathscr{P}(X). \tag{1.12}$$

The counting measure # is defined as $\#(A) = |A|$ for all finite A and $\#(A) = +\infty$ otherwise.

The co-domain of a probability density p_x for a discrete random variable is restricted to $[0, 1]$ due to the non-negativity and normalisation requirements for the probability measure P_x, with $\sum_{x \in X} \mathsf{p}_\mathsf{x}(x) = 1$. Commonly for the case of a discrete random variable, the density p_x is instead referred to as a *probability mass function*, with density reserved for real random variables. We will however use *probability density* in both cases in keeping with the earlier definition of a density, this avoiding difficulties with terminology and notation when defining joint probabilities on a mixture of real and discrete random variables.

The joint probability $\mathsf{P}_{\mathsf{x},\mathsf{y}}$ of a pair of random variables x and y with co-domains the measurable spaces $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ respectively, can be defined via a joint probability density $\mathsf{p}_{\mathsf{x},\mathsf{y}} : X \times Y \to [0, \infty)$ with respect to a product measure $\mu_{\mathsf{x},\mathsf{y}} = \mu_\mathsf{x} \times \mu_\mathsf{y}$ by

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \int_{A \times B} \mathsf{p}_{\mathsf{x},\mathsf{y}}(x, y)\, \mu_{\mathsf{x},\mathsf{y}}(\mathrm{d}x, \mathrm{d}y) \tag{1.13}$$

If $(X_1, \mathcal{F}_1, \mu_1)$ and $(X_2, \mathcal{F}_2, \mu_2)$ are two measure spaces, the product measure $\mu_1 \times \mu_2$ on a measurable space $(X_1 \times X_2, \mathcal{F}_1 \otimes \mathcal{F}_2)$ is defined as satisfying $(\mu_1 \times \mu_2)(A_1 \times A_2) = \mu_1(A_1)\mu_2(A_2)$ $\forall A_1 \in \mathcal{F}_1,\ A_2 \in \mathcal{F}_2$.

As a consequence of Fubini's theorem, the integral with respect to $\mu_{x,y}$ can be expressed as iterated integrals with respect to μ_x and μ_y

$$\begin{aligned} \mathsf{P}_{x,y}(A, B) &= \int_A \int_B \mathsf{p}_{x,y}(x, y)\, \mu_x(\mathrm{d}x)\, \mu_y(\mathrm{d}y) \\ &= \int_B \int_A \mathsf{p}_{x,y}(x, y)\, \mu_y(\mathrm{d}y)\, \mu_x(\mathrm{d}x) \quad \forall A \in \mathcal{F},\, B \in \mathcal{G}. \end{aligned} \tag{1.14}$$

The two measures μ_x and μ_y can differ for example $\mu_x = \lambda$ and $\mu_y = \#$ if x is a real random variable and y is a discrete random variable.

When dealing with random variables, we will often only specify the co-domain of the random variable(s) and a (joint) probability density, with the base measure being implicitly defined as the Lebesgue measure for real random variables (or vectors), counting measure for discrete random variables and an appropriate product measure for a mix of random variables. Similarly we will usually neglect to explicitly define the probability space $(S, \mathcal{E}, \mathsf{P})$ which the random variable(s) map from. In this case we will typically use the loose notation $\mathsf{x} \in X$ to mean a random variable x with co-domain X.

Tables A.1, A.2 and A.3 in Appendix A give definitions of the densities and shorthand notation for some common parametric probability distributions that we use in this thesis.

1.1.4 Transforms of random variables

A key concept we make use of in this thesis is defining transformations of random variables. Let x be a random variable with co-domain the measurable space $(X, \mathcal{F})$. Further let $(Y, \mathcal{G})$ be a second measurable space and $\phi : X \to Y$ a measurable function between the two spaces. If we define $\mathsf{y} = \phi \circ \mathsf{x}$ then analogously to our original definition of P_x as the pushforward measure of P under the measurable function defining x, we can define P_y in terms of P_x as

$$\mathsf{P}_y(A) = \mathsf{P}_x \circ \phi^{-1}(A) = \mathsf{P}_x(\{x \in X : \phi(x) \in A\}) \quad \forall A \in \mathcal{G}, \tag{1.15}$$

i.e. the probability of the event $\mathsf{y} \in A$ is equal to the probability of x taking a value in the pre-image under ϕ of A. To calculate probabilities of transformed random variables therefore we will therefore need to be able to find the pre-images of sets under the transformation.

If the distribution P_x is defined by a density p_x with respect to a measure μ_x, we can also in some cases find a density p_y on the transformed variable $\mathsf{y} = \phi(\mathsf{x})$ with respect to a (potentially different) measure μ_y

$$\mathsf{P}_\mathsf{y}(A) = \int_{\phi^{-1}(A)} \mathsf{p}_\mathsf{x}(x)\, \mu_\mathsf{x}(\mathrm{d}x) = \int_A \mathsf{p}_\mathsf{y}(y)\, \mu_\mathsf{y}(\mathrm{d}y) \quad \forall A \in \mathcal{G}. \tag{1.16}$$

For random variables with countable co-domains where the integral in (1.16) corresponds to a sum, a p_y satisfying (1.16) is simple to identify. If x is a discrete random variable with probability density p_x with respect to the counting measure, then $\mathsf{y} = \phi(\mathsf{x})$ will necessarily also be a discrete random variable. Applying (1.16) for $\mathsf{p}_\mathsf{x} = \frac{\mathrm{d}\mathsf{P}_\mathsf{x}}{\mathrm{d}\#}$ we have that[1]

$$\begin{aligned}\int_{\phi^{-1}(A)} \mathsf{p}_\mathsf{x}(x)\, \#(\mathrm{d}x) &= \sum_{x \in \phi^{-1}(A)} \mathsf{p}_\mathsf{x}(x) = \sum_{y \in A} \sum_{x \in \phi^{-1}(y)} \mathsf{p}_\mathsf{x}(x) \\ &= \int_A \sum_{x \in \phi^{-1}(y)} \mathsf{p}_\mathsf{x}(x)\, \#(\mathrm{d}y) \quad \forall A \in \mathcal{G}. \end{aligned} \tag{1.17}$$

We can therefore define $\mathsf{p}_\mathsf{y} = \frac{\mathrm{d}\mathsf{P}_\mathsf{y}}{\mathrm{d}\#}$ in terms of p_x as

$$\mathsf{p}_\mathsf{y}(y) = \sum_{x \in \phi^{-1}(y)} \mathsf{p}_\mathsf{x}(x) \quad \forall y \in Y. \tag{1.18}$$

In the special case that ϕ is bijective with an inverse ϕ^{-1} we have that

$$\mathsf{p}_\mathsf{y}(y) = \mathsf{p}_\mathsf{x} \circ \phi^{-1}(y) \quad \forall y \in Y. \tag{1.19}$$

For transformations of real random variables and vectors, the situation is more complicated as we need to account for any local contraction or expansion of space by the transformation. We will consider here the special case where the transformation is a *diffeomorphism*: a differentiable bijection which has an inverse which is also differentiable. In Chapter 4 we will consider how this can be generalised to non-bijective differentiable functions, including the case where the dimensionalities of the domain and co-domain of the function differ.

1 We use $\phi^{-1}(y)$ as a shorthand here for $\phi^{-1}(\{y\})$ i.e. the pre-image of a singleton set $\{y\}$ under ϕ or equivalently the *fibre* of an element y under ϕ. In cases where an inverse function exists we will also use the same notation, which of the three meanings is intended should be clear from the context.

Let $X \subseteq \mathbb{R}^N$ and $Y \subseteq \mathbb{R}^N$ and $\boldsymbol{\phi} : X \to Y$ be a diffeomorphism. Then the *Jacobian* of $\boldsymbol{\phi}$ is defined as

$$\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \phi_1}{\partial x_1} & \cdots & \frac{\partial \phi_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial \phi_N}{\partial x_1} & \cdots & \frac{\partial \phi_N}{\partial x_N} \end{bmatrix}. \tag{1.20}$$

The Jacobian matrix describes the local transformation of an infinitesimal volume element $\mathrm{d}\boldsymbol{x}$ in X under the map $\boldsymbol{\phi}$. In particular the corresponding volume element in Y under the map will be an infinitesimal parallelotope spanned by the columns of the Jacobian $\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x})$. The Jacobian matrix determinant $\left|\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x})\right|$ which corresponds to the volume of this parallelotope therefore determines hows the volume elements scales under the map - a value more than one corresponds to a local expansion and less than one to a contraction. Informally we therefore have that $\mathrm{d}\boldsymbol{y} = \left|\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x})\right| \mathrm{d}\boldsymbol{x}$ and applying the same arguments to the inverse map $\boldsymbol{\phi}^{-1}$, $\mathrm{d}\boldsymbol{x} = \left|\mathbf{J}_{\boldsymbol{\phi}^{-1}}(\boldsymbol{y})\right| \mathrm{d}\boldsymbol{y}$.

Let $\mathbf{x}$ be a random vector taking values in the measurable space $(X, \mathscr{B}(X))$ and define $\mathbf{y} = \boldsymbol{\phi} \circ \mathbf{x}$ as a random vector taking values in $(Y, \mathscr{B}(Y))$. If $\mathsf{P}_{\mathbf{x}}$ has a density $\mathsf{p}_{\mathbf{x}}$ with respect to the Lebesgue measure

$$\begin{aligned} \mathsf{P}_{\mathbf{y}}(A) = \mathsf{P}_{\mathbf{x}} \circ \boldsymbol{\phi}^{-1}(A) &= \int_{\boldsymbol{\phi}^{-1}(A)} \mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\ &= \int_A \mathsf{p}_{\mathbf{x}} \circ \boldsymbol{\phi}^{-1}(\boldsymbol{y}) \left|\mathbf{J}_{\boldsymbol{\phi}^{-1}}(\boldsymbol{y})\right| \mathrm{d}\boldsymbol{y}. \end{aligned} \tag{1.21}$$

Therefore $\mathsf{P}_{\mathbf{y}}$ has a density $\mathsf{p}_{\mathbf{y}}$ with respect to the Lebesgue measure

$$\mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) = \mathsf{p}_{\mathbf{x}} \circ \boldsymbol{\phi}^{-1}(\boldsymbol{y}) \left|\mathbf{J}_{\boldsymbol{\phi}^{-1}}(\boldsymbol{y})\right| \quad \forall \boldsymbol{y} \in Y. \tag{1.22}$$

In both the cases considered, we have seen that if the function ϕ the random variable x is mapped through is bijective, it is tractable to compute a density on the mapped random variable y as the pre-image $\phi^{-1}(y)$ of a point $y \in Y$ is itself a point. Bijectivity is a very limiting condition however, with many models involving non-bijective transformations of random variables. In Chapter 4 we will discuss methods for performing inference in generative models defined by complex, non-dimension preserving and non-bijective transformations of random variables.

1.1.5 Expectations

A key operation when working with probabilistic models is computing expectations. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space, and $\mathsf{x} : S \to X$ a random variable on this space. The *expected value of* x is defined as

$$\mathbb{E}[\mathsf{x}] = \int_S \mathsf{x} \, \mathrm{dP}. \tag{1.23}$$

Usually it will be more convenient to express expectations in terms of the probability P_x. If $g : X \to \mathbb{R}$ is an integrable function then

$$\int_X g(x) \, \mathsf{P}_\mathsf{x}(\mathrm{d}x) = \int_S g \circ \mathsf{x}(s) \, \mathsf{P}(\mathrm{d}s). \tag{1.24}$$

If we take g as the identity map we therefore have that

$$\mathbb{E}[\mathsf{x}] = \int_X x \, \mathsf{P}_\mathsf{x}(\mathrm{d}x). \tag{1.25}$$

If P_x is given by a density $\mathsf{p}_\mathsf{x} = \frac{\mathrm{dP}_\mathsf{x}}{\mathrm{d}\mu}$ then using (1.9) we also have

$$\mathbb{E}[\mathsf{x}] = \int_X x \, \mathsf{p}_\mathsf{x}(x) \, \mu(\mathrm{d}x). \tag{1.26}$$

A further useful implication of (1.24) is what is sometimes termed the *Law of the unconscious statistician*. Let $\mathsf{x} : S \to X$ be a random variable, $\phi : X \to Y$ a measurable function and define $\mathsf{y} = \phi \circ \mathsf{x}$. Then

$$\mathbb{E}[\mathsf{y}] = \int_S \mathsf{y}(s) \, \mathsf{P}(\mathrm{d}s) = \int_S \phi \circ \mathsf{x}(s) \, \mathsf{P}(\mathrm{d}s) = \int_X \phi(x) \, \mathsf{P}_\mathsf{x}(\mathrm{d}x), \tag{1.27}$$

i.e. the expectation of y can be calculated by integrating ϕ with respect to P_x. This means we can calculate the expected value of a transformed random variable $\mathsf{y} = \phi(\mathsf{x})$ without using the change of variables formulae from Section 1.1.4 to explicitly calculate the density p_y and with a relatively weak condition of measurability on ϕ.

Closely related to the expected value are the *variance* and *covariance* of a random variable. The variance of a random variable x is defined

$$\mathbb{V}[\mathsf{x}] = \mathbb{E}\left[(\mathsf{x} - \mathbb{E}[\mathsf{x}])^2\right] = \mathbb{E}\left[\mathsf{x}^2\right] - \mathbb{E}[\mathsf{x}]^2. \tag{1.28}$$

For a pair of random variables x and y their covariance is defined

$$\mathbb{C}[\mathsf{x}, \mathsf{y}] = \mathbb{E}[(\mathsf{x} - \mathbb{E}[\mathsf{x}])(\mathsf{y} - \mathbb{E}[\mathsf{y}])]. \tag{1.29}$$

1.1.6 Conditional expectations and densities

A related concept, and one which will be key to the discussion of inference, is conditional expectation. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space, $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ two measurable spaces and $\mathsf{x} : S \to X$ and $\mathsf{y} : S \to Y$ two random variables. Then the *conditional expectation of* x *given* y, is defined as a measurable function $\mathbb{E}[\mathsf{x} \,|\, \mathsf{y}] : Y \to X$ satisfying

$$\int_{\mathsf{y}^{-1}(B)} \mathsf{x}(s)\, \mathsf{P}(\mathrm{d}s) = \int_B \mathbb{E}[\mathsf{x} \,|\, \mathsf{y}](y)\, \mathsf{P}_\mathsf{y}(\mathrm{d}y) \quad \forall B \in \mathcal{G}. \tag{1.30}$$

$\mathbb{E}[\mathsf{x} \,|\, \mathsf{y}]$ is guaranteed to be uniquely defined almost everywhere in Y by (1.30), i.e. up to P_y-null sets. As a particular case where $B = Y$ we recover what is sometimes termed the *Law of total expectation*

$$\int_S \mathsf{x}\, \mathrm{d}\mathsf{P} = \int_S \mathbb{E}[\mathsf{x} \,|\, \mathsf{y}] \circ \mathsf{y}\, \mathrm{d}\mathsf{P} \implies \mathbb{E}[\mathsf{x}] = \mathbb{E}[\mathbb{E}[\mathsf{x} \,|\, \mathsf{y}] \circ \mathsf{y}]. \tag{1.31}$$

We will also use a more standard notation for the conditional expectation evaluated at point $\mathbb{E}[\mathsf{x} \,|\, \mathsf{y} = y] \equiv \mathbb{E}[\mathsf{x} \,|\, \mathsf{y}](y)$ but use the latter in this section to stress its definition as a measurable function.

Conditional expectation can be used to define the *regular conditional distribution* of a random variable conditioned on another random variable taking a particular value

$$\mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \,|\, y) = \mathbb{E}[\mathbb{1}_A \circ \mathsf{x} \,|\, \mathsf{y}](y) \quad \forall y \in Y,\, A \in \mathcal{F}. \tag{1.32}$$

We have reused our notation for conditional probability of random variables from Section 1.1.2 here, however it should be clear from whether the value conditioned on is a point or a set which is being referred to. A regular conditional distribution $\mathsf{P}_{\mathsf{x}|\mathsf{y}}(\cdot \,|\, y)$ defines a valid probability measure on $(X, \mathcal{F})$ for P_y-almost all y and using (1.30) we have

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \int_B \mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \,|\, y)\, \mathsf{P}_\mathsf{y}(\mathrm{d}y) \qquad \forall A \in \mathcal{F},\, B \in \mathcal{G}. \tag{1.33}$$

We can use this relationship to also motivate a definition of conditional density. We require that a joint density $\mathsf{p}_{\mathsf{x},\mathsf{y}} = \frac{\mathrm{d}\mathsf{P}_{\mathsf{x},\mathsf{y}}}{\mathrm{d}(\mu_\mathsf{x} \times \mu_\mathsf{y})}$ exists and has marginal density $\mathsf{p}_\mathsf{y} = \frac{\mathrm{d}\mathsf{P}_\mathsf{y}}{\mathrm{d}\mu_\mathsf{y}}$. Then for all $A \in \mathcal{F}$, $B \in \mathcal{G}$

$$\int_B \mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \,|\, y)\, \mathsf{P}_\mathsf{y}(\mathrm{d}y) = \int_B \int_A \mathsf{p}_{\mathsf{x},\mathsf{y}}(x, y)\, \mu_\mathsf{x}(\mathrm{d}x)\, \mu_\mathsf{y}(\mathrm{d}y) \tag{1.34}$$

If we define the *conditional density* $p_{x|y}$ as

$$p_{x|y}(x \mid y) = \begin{cases} \frac{p_{x,y}(x,y)}{p_y(y)} & \forall x \in X,\, y \in Y : p_y(y) > 0 \\ 0 & \forall x \in X,\, y \in Y : p_y(y) = 0 \end{cases} \tag{1.35}$$

then substituting this definition in to (1.34) we have

$$\int_B P_{x|y}(A \mid y)\, P_y(\mathrm{d}y) = \int_B \int_A p_{x|y}(x \mid y)\, \mu_x(\mathrm{d}x)\, P_y(\mathrm{d}y). \tag{1.36}$$

Therefore $p_{x|y}$ is the density of the regular conditional distribution $P_{x|y}$. We also have that if $p_{x,y}$ and p_y can be defined that

$$\mathbb{E}[x \mid y](y) = \int_X x\, p_{x|y}(x \mid y)\, \mu_x(\mathrm{d}x) \quad \forall y \in Y : p_y(y) > 0. \tag{1.37}$$

Note that the initial definition of conditional expectation in (1.30) was not dependent on a joint density $p_{x,y}$ being defined.

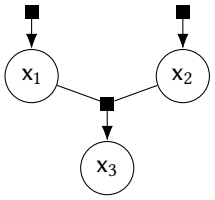
1.2 GRAPHICAL MODELS

Graphical models = statistics × graph theory × computer science
—Zoubin Ghahramani

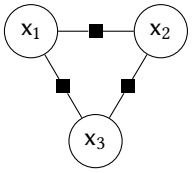
When working with probabilistic models involving large numbers of random variables, it will often be the case that not all the variables are jointly dependent on each other but that instead there are more local relationships between them. Graphical models, which use graphs to describe the dependencies between random variables, are a useful framework for visualising the structure of complex models.

Several graphical formalisms for representing dependency structure in probabilistic models have been proposed, with *directed graphical models* [206] (also known as *Bayesian networks*) and *undirected graphical models* [138] (also known as *Markov random fields*) both common in practice and each offering a more natural representation for certain model classes. In this thesis we will instead use *factor graphs* [90, 91] which combine the representational abilities of both directed and undirected graphical models while also offering a richer framework for representing fine-grained information about model structure.

Factor graphs are bipartite graphs consisting of two node types: *variable nodes*, displayed as labelled circles and representing individual (potentially non-scalar) random variables in a probabilistic model, and *factor nodes*, displayed as filled squares and representing individual



(a) Example directed factor graph.



(b) Example undirected factor graph.

Figure 1.1.: Examples of simple directed and undirected factor graphs. Square black nodes correspond to individual factors depending on the connected variables represented by circular nodes in the joint density.

factors in the joint density across the random variables in the model. Edges between nodes in a factor graph are always between nodes of disparate types i.e. between factor and variable nodes, but never between factor and factor or variable and variable nodes.

Factors may be either directed or undirected. Undirected factors, denoted by factor nodes in which all edges connecting to variable nodes are undirected, correspond to a factor in the joint density which depends on all of the variables with nodes connected to the factor, but without any requirement that the factor corresponds to a conditional or marginal probability density.

Directed factors, factor nodes in which at least one edge from the factor node to a variable node is directed, correspond to a conditional density on the variables pointed to by directed edges given the values of the variables connected to the the factor node by undirected edges. If there are no undirected edges then the factor instead corresponds to a marginal density. Graphs with directed factors must not contain any directed cycles, i.e. connected loops of edges in which one of every pair of edges connected to a factor on the loop is directed and all of the directed edges point in the same sense around the loop.

Figure 1.1a shows a simple example of fully directed factor graph for three random variables. The graph implies that the joint density for the model can be factorised as

$$\mathrm{p}_{\mathrm{x}_1,\mathrm{x}_2,\mathrm{x}_3}(x_1, x_2, x_3) = \mathrm{p}_{\mathrm{x}_3|\mathrm{x}_1,\mathrm{x}_2}(x_3 \mid x_1, x_2)\, \mathrm{p}_{\mathrm{x}_1}(x_1)\, \mathrm{p}_{\mathrm{x}_2}(x_2). \tag{1.38}$$

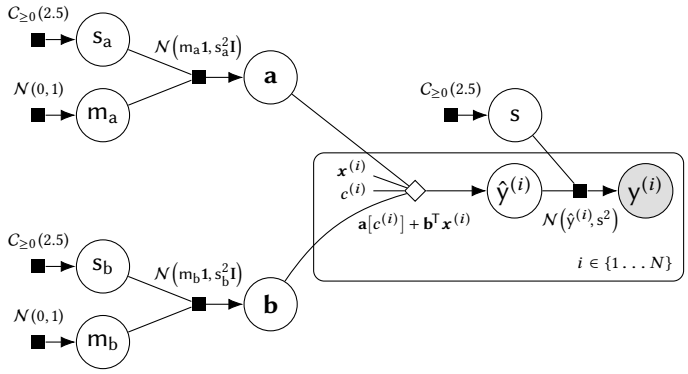


Figure 1.2.: Hierarchical linear regression model factor graph showing examples of extended factor graph notation.

Figure 1.1b shows a fully undirected factor graph on three random variables. If $\psi_{i,j}$ denotes the unnormalised density factor on the pair $(\mathsf{x}_i, \mathsf{x}_j)$ then the graph implies the joint density can be factorised as

$$\mathrm{p}_{\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_3}(x_1, x_2, x_3) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{1,3}(x_1, x_3)\psi_{2,3}(x_2, x_3) \tag{1.39}$$

with Z a normalising constant such that the density integrates to one.

Figure 1.2 shows examples of some additional useful factor graph notation we will use in this thesis. We use as an example a factor graph corresponding to a hierarchical linear regression model which will be discussed in Chapter 5.

It will often be useful to be able to explicitly represent deterministic functions applied to the random variables in a factor graph. For this purpose we introduce an additional node type denoted by an unfilled diamond (◇). The semantics of this node type are similar to standard directed factor nodes. Variables acting as inputs to the function are connected to the node by undirected edges and the variable corresponding to the function output indicated by a directed edge from the node to the relevant variable. Like standard factor nodes, the deterministic factor nodes only ever connect to variable nodes. The operations performed by the function on the inputs will usually be included as a label adjacent to the node as illustrated by the example in Figure 1.2. A deterministic factor node can informally[2] be considered equivalent to a directed

2 A Dirac delta is not strictly a density as it is not the Radon–Nikodym derivative of an absolutely continuous measure, however informally we treat is as the density of a singular Dirac measure with respect to the Lebesgue measure $\int f(x)\,\delta(\mathrm{d}x) \simeq \int f(x)\delta(x)\,\mathrm{d}x$.

factor node with a Dirac delta conditional density on the output variable which concentrates all the probability mass at the output of the function applied to the inputs variables.

The deterministic node notation allows generative models consisting of complex compositions of deterministic functions and probabilistic sampling operations to be represented in a unified framework. Subgraphs of a directed factor graph consisting entirely of deterministic nodes can be viewed as *computation graphs*, a graphical formalism typically used in numerical computing frameworks to support efficient *automatic differentiation* algorithms. We exploit this idea in later in the thesis to allow propagation of derivatives through complex probabilistic models and make extensive use of automatic differentiation implementations in frameworks such as *Theano* [248] in numerical experiments. In Appendix B we provide a short review of the basic concepts of computation graphs and automatic differentiation and a discussion of their links to directed factor graphs.

In some cases constant values used in a model will be included in a factor graph as plain nodes indicated only by a label. The $\boldsymbol{x}^{(i)}$ and $c^{(i)}$ nodes in Figure 1.2 are an example of this notation.

A commonly used convention in factor graphs is *plate notation* [50], with an example of a plate shown by the rounded rectangle bounding some of the nodes in Figure 1.2. Plates are used to indicate a subgraph in the model which is replicated multiple times (with the replications being indexed over a set which is typically indicated in the lower right corner of the plate as in Figure 1.2). The subgraph entirely contained on the plate is assumed to be replicated the relevant number of times, with any edges crossing into the plate from variable nodes outside of the plate being repeated once for each subgraph replication.

Each of the factors in Figure 1.2 is labelled with a shorthand for a probability density function corresponding to the conditional or marginal density factor associated with the node. Definitions for the shorthand notations that are used for densities in this thesis are given in Appendix A. The dependence of the factors on the value of the random variable the density is defined on is omitted in the labels for brevity.

A final additional notation used in Figure 1.2 is the use of a shaded variable node (corresponding to $\mathrm{y}^{(i)}$) to indicate a random variable corresponding to an observed quantity in the model.

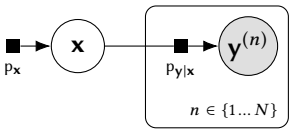


Figure 1.3.: Factor graph of N observations $\mathbf{y}^{(n)}$ independently and identically distributed according to a distribution with parameters $\mathbf{x}$.

1.3 INFERENCE

Having now introduced the tools and notation we use to define probabilistic models, we will now describe the inference problems we consider approximate approaches to solving in this thesis. We begin with a overview of *Bayesian inference*.

You cannot do inference without making assumptions
—David Mackay

The starting point for any inference problem is to define a model specifying proposed relationships between the observed data and unknown quantities to be inferred. The model codifies the assumptions we make about the problem and any prior beliefs we have. In virtually all real inference problems the model will be a simplified description of a much more complex underlying process, usually motivated by prior empirical observations that the behaviour proposed by the model is a reasonable description of reality. For now we will consider the model as a singular fixed object we will perform inference with. We consider probabilistic model comparison in a subsequent subsection.

Amongst the simplest, but also most common, modelling assumptions made is that the observed data values are *independently and identically distributed* (IID) according to a parametric probability distribution. If we denote the collection of N observed variables $\{\mathbf{y}^{(n)}\}_{n=1}^{N}$ then we assume that each is independently generated from a distribution $\mathrm{P}_{\mathbf{y}^{(n)}|\mathbf{x}} = \mathrm{P}_{\mathbf{y}|\mathbf{x}}\ \forall n \in \{1 \dots N\}$ with density $\mathrm{p}_{\mathbf{y}|\mathbf{x}} = \frac{\mathrm{dP}_{\mathbf{y}|\mathbf{x}}}{\mathrm{d}\mu_{\mathbf{y}}}$ and governed by a set of unknown parameters $\mathbf{x} \in X$.

Any beliefs we have about the plausible values for the parameters prior to observing data are integrated into the model by choosing an appropriate, typically parametric, marginal distribution $\mathrm{P}_{\mathbf{x}}$, with this distribution, and the corresponding density $\mathrm{p}_{\mathbf{x}} = \frac{\mathrm{dP}_{\mathbf{x}}}{\mathrm{d}\mu_{\mathbf{x}}}$, referred to as the *prior*. The joint density on the model variables then factorises as

$$\mathrm{p}_{\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)},\mathbf{x}}(\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)}, \boldsymbol{x}) = \prod_{n=1}^{N} \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^{(n)} \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) \tag{1.40}$$

with this structure illustrated as a directed factor graph in Figure 1.3. In analogy to the naming of the prior, the conditional distribution on the unknown model parameters after conditioning on observed data values is termed the *posterior* and from the definition of conditional density (1.35) we can express its density as

$$\mathrm{p}_{\mathbf{x}|\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(N)}}(\boldsymbol{x} \mid \boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}) = \frac{\prod_{n=1}^{N} \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^{(n)} \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})}{\mathrm{p}_{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(N)}}(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)})}. \tag{1.41}$$

Bayesian inference is named after Thomas Bayes, an 18th century Presbyterian minister, who proved a special case of what is now termed Bayes' theorem. Pierre-Simon Laplace later independently derived a more general statement of Bayes' theorem closer to the modern form.

This expression relating the posterior on the unknown parameters to the prior distribution and model of the observations is an example of *Bayes' theorem*. Typically the product of the conditional densities $\mathrm{p}_{\mathbf{y}|\mathbf{x}}$ is termed the *likelihood* and considered a function of the value $\boldsymbol{x}$ of the unknown parameters $\mathbf{x}$, with the observed data values $\{\boldsymbol{y}^{(n)}\}_{n=1}^{N}$ fixed. The denominator of the right-hand side (1.41), the marginal density on the observed variables, can be written as a integral marginalising out the parameters from the joint density

$$\mathrm{p}_{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(N)}}(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}) = \int_X \prod_{n=1}^{N} \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^{(n)} \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mu_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}). \tag{1.42}$$

A conditional density $\mathrm{p}_{\mathbf{u}|\mathbf{v}}$ is from the exponential family if it can be written as $\mathrm{p}_{\mathbf{u}|\mathbf{v}}(\boldsymbol{u} \mid \boldsymbol{v}) = \frac{h(\boldsymbol{u}) \exp(\boldsymbol{\eta}(\boldsymbol{v})^{\mathsf{T}} \boldsymbol{t}(\boldsymbol{u}))}{z(\boldsymbol{v})}$, with $\boldsymbol{\eta}(\boldsymbol{v})$ termed the natural parameters *and $\boldsymbol{t}(\boldsymbol{u})$ termed the* sufficient statistics.

This term is often described as the *marginal likelihood* or the *model evidence*. Generally this integral will not have an analytic solution though there are exceptions to this in a few special cases. For example if the densities $\mathrm{p}_{\mathbf{y}|\mathbf{x}}$ and $\mathrm{p}_{\mathbf{x}}$ are both of *exponential family distributions* and form a conjugate pair such that the posterior density is in the same family as the prior density then (1.42) will have a closed-form solution. For models in which the parameters are discrete the integral in (1.42) corresponds to a summation and so is in theory exactly solvable, though if the total number of possible configurations of the parameters is very large this summation can be infeasible to compute in practice. If the parameters are instead real-valued but of a low-dimensionality it may be possible to use numerical quadrature methods [70] to compute the integral in (1.42) to a reasonable accuracy. Quadrature methods involve evaluating the integrand across a grid of points and then computing a weighted sum of these values. For a fixed grid resolution however the cost of quadrature scales exponentially with the dimension of the space being integrated over - if G points are used per dimension, for a D dimensional parameter space evaluating (1.42) would require summing the joint density over G^D parameter values.

For real-valued parameter spaces of a more than ~ 10 dimensions[3] evaluating the model evidence term (1.42) is therefore typically computationally intractable. We can therefore often only evalulate the posterior density (1.41) up to an unknown constant. The posterior density itself is usually not of direct interest as it is only a proxy for describing the posterior distribution and is dependent on the particular model parameterisation chosen. However most quantities of interest from an inference perspective involve integrating functions against the posterior distribution and as with the model evidence these integrals will typically be intractable to compute exactly.

For example under an IID assumption the density of the *predictive distribution* of a new data point $\mathbf{y}^*$ given the previously observed data is formed by integrating $\mathrm{p}_{\mathbf{y}|\mathbf{x}}$ against the posterior distribution

$$\begin{aligned}
&\mathrm{p}_{\mathbf{y}^*|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{y}^* \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)}) \\
&\quad = \int_X \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^* \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{x} \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)})\, \mu_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \qquad (1.43) \\
&\quad = \mathbb{E}\left[\mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^* \mid \mathbf{x}) \mid \mathbf{y}^{(1)} = \boldsymbol{y}^{(1)}, \dots, \mathbf{y}^{(N)} = \boldsymbol{y}^{(N)}\right].
\end{aligned}$$

If we wish to for example minimise the expected prediction error under some loss function this will involve integrating against this predictive distribution and so as a sub-task integrating against the posterior distribution on the model parameters. Similarly evaluating statistics of the unknown parameters under the posterior such as their mean or covariance corresponds to computing conditional expectations. In general any inferential output which takes in to account all of the information available from the posterior distribution will involve integrating against the posterior and so the computation of integrals is the key computational task in inference.

As exact evaluation of the integrals of interest is usually intractable we must instead resort to *approximate inference* methods which trade-off an introduction of some level of approximation for an increase in computational tractability.

3 The C-based implementation by Steven G. Johnson of an adaptive multi-dimensional quadrature algorithm [25] available at `https://github.com/stevengj/cubature` recommends using the package for integrals of up to around $D = 7$. Running a provided test cases for the integral of a Gaussian density across a D-dimensional space with a target error tolerance of 10^{-5} took around 2.5 seconds for $D = 5$, 50 seconds for $D = 6$ and 17 minutes for $D = 7$ on one core of a desktop CPU. Extrapolating the ~ 20-fold increase in run time per dimension, for $D = 10$ the run-time would be around 100 days.

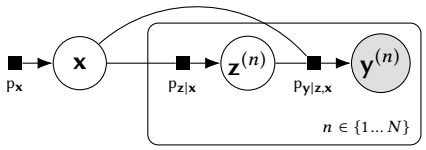


Figure 1.4.: Factor graph of a simple hierarchical latent variable model with N observed variables $\mathbf{y}^{(n)}$ each associated with a local latent variable $\mathbf{z}^{(n)}$, with both observed and latent variables dependent on a set of global latent variables (parameters) $\mathbf{x}$.

The IID assumption is widely made in inference problems and although it will not be always be entirely valid in practice, it will often be a reasonable approximation. For real-valued parameter spaces X and densities $p_{\mathbf{y}|\mathbf{x}}$ and $p_{\mathbf{x}}$ meeting certain regularity conditions, if an IID assumption is valid then the posterior distribution will asymptotically tend to a multivariate normal distribution as the number of data points N tends to infinity [123]. For inference in models of large IID datasets where the conditions for asymptotic normality are met, while the dimensionality of the parameter space will often still require the use of approximate inference methods, the close to normal geometry of the posterior distribution will typically mean even relatively simple approximate inference methods can achieve good results.

In this thesis we will primarily be concerned with methods for performing inference in models which do not fit into this mould. In the following subsections we discuss some specific issues that can prove challenging to standard approximate inference approaches and which the methods contributed in this thesis are intended to help address.

1.3.1 Hierarchical models

In the preceding discussion of inference in a model of a IID dataset, it was assumed that the only unknown variables in the model were a set of parameters $\mathbf{x}$, the quantity of which did not depend on the number of data points N. This structure can be overly restrictive with it common that the process being modelled includes unknown quantities associated with each observed variable. Models will therefore often include local (per data point) latent variables in addition to a set of global latent variables (or parameters). This grouping structure in the observed and unobserved variables in a model can extend to multiple levels and such models often are termed *hierarchical* or *multilevel* models.

A simple example of a hierarchical model is shown as a factor graph in Figure 1.4. As in the factor graph in Figure 1.3 we assume there are N observed variables $\{\mathbf{y}^{(n)}\}_{n=1}^N$ and a vector of global latent variables $\mathbf{x}$. We further define N local latent variables $\{\mathbf{z}^{(n)}\}_{n=1}^N$ paired with each observed variable. In Figure 1.4 we assume each pair of local latent and observed variables $(\mathbf{z}^{(n)}, \mathbf{y}^{(n)})$ are conditionally independent of the other pairs $\{\mathbf{z}^{(m)}, \mathbf{y}^{(m)}\}_{m\neq n}$ given the global latent variables $\mathbf{z}$. More complex structures are also common - for example dynamical state space models for time series data assume dependencies between the latent variables corresponding to adjacent time points.

Although powerful, the introduction of local latent variables in to models can significantly increase the complexity of inference. At a basic level, as the number of unobserved variables is now dependent on the data set size, the total dimensionality of the space which needs to be integrated over when performing inference will typically be much higher than for models with a fixed number of parameters. This means the need for inference methods which scale well with dimensionality is even more essential. The growth of the the number of unobserved variables with the data set size N will typically also mean that we can no longer expect asymptotic normality of the full posterior. Often the posterior distribution on the local and global latent variables will have a complex geometry, with strong dependencies between the global and local latent variables that can limit the performance of many standard approximate inference approaches [36].

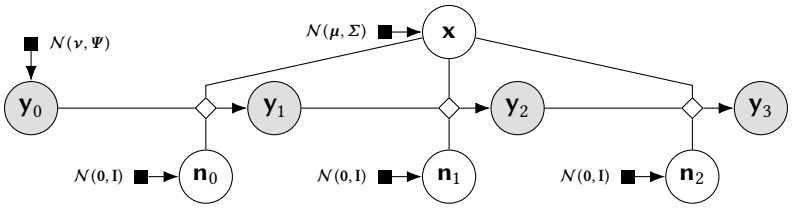
In some cases the posterior distributions of the local latent variables associated with the observed data will not be of direct interest to the downstream task. For example the conditional independence structure in Figure 1.4 means that the predictive distribution on a new unseen datapoint $\mathbf{y}^*$ given the observed data has density

$$\begin{aligned} &\mathrm{p}_{\mathbf{y}^*|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{y}^* \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)}) \\ &\quad = \int_Z \int_X \mathrm{p}_{\mathbf{y}|\mathbf{x},\mathbf{z}}(\boldsymbol{y}^* \mid \boldsymbol{x}, \boldsymbol{z})\, \mathrm{p}_{\mathbf{z}|\mathbf{x}}(\boldsymbol{z} \mid \boldsymbol{x}) \\ &\qquad\qquad \mathrm{p}_{\mathbf{x}|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{x} \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)})\, \mu_{\mathbf{x}}(\mathrm{d}\boldsymbol{x})\, \mu_{\mathbf{z}}(\mathrm{d}\boldsymbol{z}). \end{aligned} \tag{1.44}$$

Predictions under the model will therefore not depend on the values of the local latent variables $\{\mathbf{z}^{(n)}\}_{n=1}^N$, and so ideally we would marginalise out these variables from the full posterior distribution on all

$\boldsymbol{y}_0 \sim \mathcal{N}(\cdot \mid \boldsymbol{\nu}, \boldsymbol{\Psi})$
$\boldsymbol{x} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$
for $t \in \{1 \dots T\}$ **do**
  $\boldsymbol{n}_{t-1} \sim \mathcal{N}(\cdot \mid \mathbf{0}, \mathbf{I})$
  $\boldsymbol{y}_t \leftarrow \boldsymbol{y}_{t-1} + h\boldsymbol{m}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) - \frac{h}{2}\boldsymbol{s}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) \odot \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{y}}(\boldsymbol{y}_{t-1}, \boldsymbol{x})$
  $\boldsymbol{y}_t \leftarrow \boldsymbol{y}_t + \sqrt{h}\,\boldsymbol{s}(\boldsymbol{y}_{t-1}, z) \odot \boldsymbol{n}_{t-1} + \frac{h}{2}\boldsymbol{s}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) \odot \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{y}}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) \odot \boldsymbol{n}_{t-1}^2$

(a) Pseudo-code for Milstein method integration of SDE model.



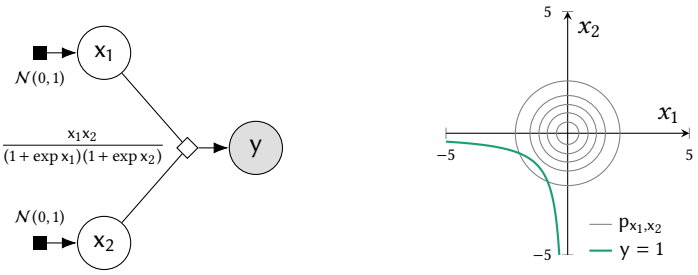
(b) Directed factor graph of 3 time steps of SDE simulation.

Figure 1.5.: Example of a simulator model corresponding to Milstein method integration of a set of SDEs, $\mathrm{d}\mathbf{y}(t) = \boldsymbol{m}(\mathbf{y}(t), \mathbf{x})\,\mathrm{d}t + \boldsymbol{s}(\mathbf{y}(t), \mathbf{x})\,\mathrm{d}\mathbf{n}(t)$, specified as pseudo-code in (a) and a directed factor graph in (b). The dynamics of model are governed by parameters $\mathbf{x}$. In the pseudo-code the notation $\sim$ followed by a distribution shorthand represents generating a value from the associated distribution.

unobserved variables $\mathrm{P}_{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)}, \mathbf{x} \mid \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)}}$ to obtain the posterior distribution on just the global latent variables $\mathrm{P}_{\mathbf{x} \mid \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)}}$. The distribution $\mathrm{P}_{\mathbf{x} \mid \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)}}$ is defined on a much lower dimensional space and will often have a simpler geometry which makes it more amenable to approximate inference methods, however generally the marginalisation over the local latent variables will not be analytically tractable. We can in some cases however approximately marginalise out the local latent variables - we discuss methods based on this idea in Chapter 3.

1.3.2 Simulator models

The probabilistic models considered so far have been defined by explicitly specifying a density over the all the variables in the model, for example via a factor graph. Rather than defining the density on the variables in a model an alternative approach is for a process for generating values for the variables in a model to be specified procedurally in code, with the resulting joint density on the model variables then only implicitly defined. Such models are sometimes termed *simulator* or *implicit* models [77].



(a) Directed factor graph for model with two latent variables (x_1, x_2), and an observed variable y.

(b) Plot of marginal density on latent variables (contours) and set of values for which $y = 1$ (green curve).

Figure 1.6.: Simple example of an implicit probabilistic model where the observed variable is a non-bijective function of two latent variables.

A common setting in which such models occur is the simulation of a mechanistic model of a physical process for example described by a set of *stochastic differential equations* (SDEs). In implementations of such simulator models, the stochasticity in the model will be introduced via draws from a pseudo-random number generator. Given these random inputs, the output of the simulator is then calculated as a series of deterministic operations and so can be described by a computation graph. The overall composition of directed factor nodes specifying the generation of random inputs from known densities by the random number generator and computation graph describing the operations performed by the simulator code together therefore define a directed factor graph. An example of a simulator model corresponding to approximate integration of a set of SDEs using the Milstein method [175] is shown as both pseudo-code and a directed factor graph in Figure 1.5.

The main complicating factor in performing inference in simulator models is the unavailability of an explicit density function on the model variables which is a prerequisite for most approximate inference methods. Computing a density function on the unobserved variables to be inferred (for example parameters of the dynamics of a SDE model) and simulated observed variables that are conditioned on requires that all other random variables used in the model are marginalised over. In some cases this marginalisation may technically be possible to exactly solve and so a density function possible to compute in theory but the complexity of the model structure means that the density is unavailable in practice.

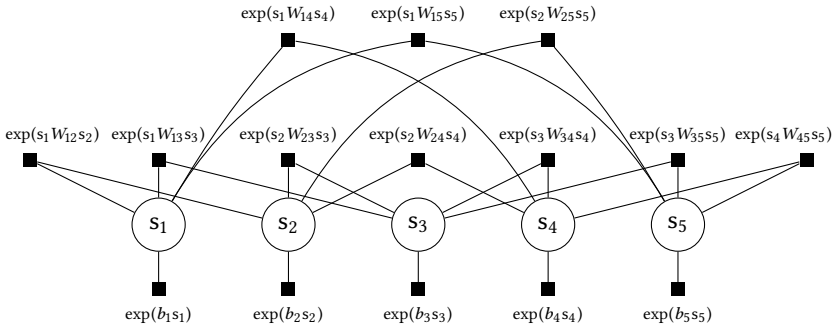


Figure 1.7.: Five unit Boltzmann machine factor graph.

In many cases however the density function may not be exactly evaluable even in theory. A key difference of simulator models from the probabilistic models considered previously is that the observed variables in the model are defined via deterministic transformations of other random variables. Using our above intuition that any simulator model can be expressed as a directed factor graph with deterministic factor nodes, this means that the observed variables in the graph correspond to the outputs of deterministic factors rather than the more usual case of the observed variables being connected to probabilistic factors.

An illustration of such a case for a simple three variable model is shown in Figure 1.6. Here the observed variable y is a deterministic function of two latent (unobserved) variables x_1 and x_2. There is no analytic solution in terms of elementary functions for x_1 as a function of y and x_2 or for x_2 as a function of x_1 and y. This means the Dirac delta term corresponding to the deterministic factor cannot be integrated out. Due to the presence of the Dirac delta the joint density $\mathsf{p}_{\mathsf{y},\mathsf{x}_1,\mathsf{x}_2}$ is not well defined (the joint distribution $\mathsf{P}_{\mathsf{y},\mathsf{x}_1,\mathsf{x}_2}$ is not absolutely continuous with respect to the Lebesgue measure) which complicates evaluations of conditional expectations such as $\mathbb{E}[f(\mathsf{x}_1, \mathsf{x}_2) \mid \mathsf{y} = 1]$. In particular the set of x_1 and x_2 values corresponding to solutions to $\mathsf{y} = y$ for an particular y (illustrated for $y = 1$ as the green curve in Figure 1.6b) is an implicitly defined manifold (here a one-dimensional curve) in the x_1–x_2 space with zero Lebesgue measure, and the conditional distribution $\mathsf{P}_{\mathsf{x}_1,\mathsf{x}_2|\mathsf{y}}$ has support only on this manifold. We explore methods for performing inference in implicit models in Chapter 4.

1.3.3 Undirected models

When introducing factor graphs we stated that factors can be both directed and undirected. In the preceding discussion we concentrated on directed models, both in the form of models explicitly specified via directed factor graphs as in the examples in Figure 1.3 and 1.4, and simulator models which as we argued in the previous subsection can be considered as implicitly defining a directed factor graph. A key defining feature of models corresponding to directed factor graphs is that they are natural descriptions of generative processes, with independent sampling from the joint distribution across model variables typically simple to perform via ancestral sampling (in the case of simulator models this being their defining feature).

Undirected models (which we will use here to mean models specified by factor graphs consisting solely of undirected factors) offer a complementary approach for defining a probabilistic model. Each undirected factor node is associated with a non-negative function defining a factor in the joint density across all model variables. Unlike a directed factor, this function does not correspond to a conditional or marginal density. Instead it describes a more general notion of 'compatibility' between the values of sets of variables in the model, defining a series of soft constraints as to which joint configurations are plausible (corresponding to a high value for the factor) or implausible (corresponding to a low value). This makes undirected models a natural representation for models of systems of mutually interacting components without a specific directivity in those interactions. For example they are commonly used in models of images to represent dependencies between pixel values, to model networks of stochastically spiking neurons in the brain and models of magnetic interactions in particle lattices. Unlike directed models, generating samples from the joint distribution on variables in an undirected model is typically a non-trivial task, with no general equivalent to ancestral sampling.

A particularly common form of undirected model is the *Boltzmann machine* [1] also known as a *pairwise binary Markov random field* [138] or in statistical physics settings an *Ising spin model* [133]. A Boltzmann machine consists of a set of binary random variables $\mathbf{s} = [\mathrm{s}_1 \cdots \mathrm{s}_D]^\mathsf{T}$; these are typically chosen to take values in $U = \{0, 1\}^D$ or $S = \{-1, +1\}^D$ - we will favour $S = \{-1, +1\}^D$. The joint distribution across the variables

is parameterised by a symmetric weight matrix $\boldsymbol{W} \in \mathbb{R}^{D\times D}$ and a bias vector $\boldsymbol{b} \in \mathbb{R}^D$ and defined as

$$\mathrm{p}_{\mathbf{s}}(\boldsymbol{s}) = \frac{1}{Z}\exp\left(\frac{1}{2}\boldsymbol{s}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^{\mathsf{T}}\boldsymbol{b}\right), \quad Z = \sum_{\boldsymbol{s}\in S}\exp\left(\frac{1}{2}\boldsymbol{s}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^{\mathsf{T}}\boldsymbol{b}\right). \tag{1.45}$$

Evaluation of the normalising constant Z involves a summation over 2^D states and so for large D quickly become intractable to compute exactly. Evaluation of expectations with respect to the Boltzmann machine distribution also involves an exhaustive summation across S and so will also be intractable for high D.

If $\mathbf{s}_1$ and $\mathbf{s}_2$ are an arbitrary partition of the variables in $\mathbf{s}$ then importantly the conditional distribution $\mathrm{P}_{\mathbf{s}_1|\mathbf{s}_2}$ will also be a Boltzmann machine distribution. However unless the dimensionality of $\mathbf{s}_1$ is small enough that exhaustive summation over its possible states is feasible, then evaluating normalising constants of this conditional distribution and expectations with respect to it will also be intractable. Therefore inference in Boltzmann machines conditioned on observations of part of the state can be considered as a special case of computing expectations and the normalising constants of (non-conditioned) Boltzmann machine distributions, with the same challenges applying to both.

Figure 1.7 shows the factor graph for a Boltzmann machine distribution on five binary random variables $\{\mathrm{s}_i\}_{i=1}^5$. Each of the weights W_{ij} defines an undirected factor between a pair of variables $\mathrm{s}_i W_{ij} \mathrm{s}_j$. As the variables take on signed binary values, this factor is equal to W_{ij} when the variables are equal and so take the same sign and equal to $-W_{ij}$ when the variables take differing values. If W_{ij} is positive this factor therefore favours states where s_i and s_j are in the same configuration, while if W_{ij} is negative states with s_i and s_j in opposing configurations are preferred.

Boltzmann machine systems with a mixture of positive and negative weights will often be *frustrated* with no one global configuration which satisfies the preferences specified by each weight, and instead there being multiple states which each locally satisfy a subset of the soft constraints specified by the weights. This typically leads to a highly multi-modal distribution on the states of the system, with collections

of nearby[4] states of high-probability separated sets of states with very low probability.

This multi-modality typically makes frustrated Boltzmann machines very challenging distributions to perform approximate inference with. In particular methods based on constructing Markov chains which explore the state of the system tend to converge very slowly as they will typically remain confined to a particular high-probability region of the state space for many iterations. In Chapter 5 we will consider methods for constructing Markov chains with improved exploration of challenging multi-modal target distributions, including methods for estimating expectations and normalising constants of frustrated Boltzmann machine distributions.

1.3.4 Model comparison

So far we have discussed inferring the unobserved variables in a single fixed model. An important second level of inference is comparing competing models for the same observed data. This can be treated consistently within the probabilistic framework we have discussed.

Ockham's Razor is a philosophical principle, commonly attributed to the 14th century Franciscan friar William of Ockham, that states if there exist multiple explanations for observations, all else being equal we should prefer the simplest.

Given observed data, we would like to be able to make a judgement as to which of two (or more) proposed models better describes the data. To be useful this comparison must take into account the relative complexity of the models; a model with more free variables will generally be able to fit observed data more closely, however *Ockham's Razor* (and corresponding empirical evidence of the loss of predictive power of overly complex models) suggests we should prefer simpler models where possible. By marginalising over the free, unobserved variables in a model, probabilistic model comparison automatically embodies Ockham's Razor [161].

A concrete structure for model comparison is to assume that there are a finite set of M models, indexed by an *indicator* variable $\mathsf{m} \in \{1 \ldots M\}$. All models share the same observed variables[5] $\mathbf{y}$, and there are a set of per model vectors of unobserved variables $\{\mathbf{x}_m\}_{m=1}^M$ which are assumed to be independent (before conditioning on observations). More complex structures could be assumed such as the models sharing a set

4 Nearby here being in terms of the Hamming distance between the binary states.

5 For notational simplicity here we assume all observed variables have been concatenated in to one vector and similarly for the unobserved variables, with any internal model factorisation structure such as discussed in the preceding sections omitted.

of common unobserved variables, however we only consider the case of independent models here. The joint density on the observations, model indicator and latent variables is then assumed to factorise as

$$\begin{aligned}\mathrm{p}_{\mathbf{y},\mathrm{m},\mathbf{x}_1,\ldots,\mathbf{x}_M}(\boldsymbol{y}, m, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_M) = \\ \mathrm{p}_{\mathbf{y}|\mathrm{m},\mathbf{x}_m}(\boldsymbol{y} \mid m, \boldsymbol{x}_m)\, \mathrm{p}_{\mathrm{m}}(m) \prod_{n=1}^{M} \mathrm{p}_{\mathbf{x}_n}(\boldsymbol{x}_n).\end{aligned} \tag{1.46}$$

The marginal density on the model indicator p_{m} represents our prior beliefs about the relative probabilities of the models before observing data. Importantly the value of the model indicator variable m selects the relevant per model conditional density on the observed variables given latent variables $\mathrm{p}_{\mathbf{y}|\mathrm{m},\mathbf{x}_m}$; this represents the assumption that conditioned on the model indicator assuming a particular model index m the observed variables are conditionally independent of the latent variables of all other models $\mathbf{y} \perp \{\mathbf{x}_n\}_{n \neq m} \mid \mathrm{m} = m$.

Given this computational set up, the task in model comparison is then to compute the relative probabilities of each of the models given observed data. These probabilities are given by

$$\mathrm{p}_{\mathrm{m}|\mathbf{y}}(m \mid \boldsymbol{y}) = \frac{\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid m)\, \mathrm{p}_{\mathrm{m}}(m)}{\sum_{n=1}^{M} \Big(\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid n)\, \mathrm{p}_{\mathrm{m}}(n) \Big)}, \tag{1.47}$$

which can be seen as a direct analogue to Bayes' theorem for the posterior density on unobserved random variables for a single model. The key quantities needed to evaluate the model posterior probabilities are the marginal densities $\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid m)$ evaluated at the observed data. Computing these values requires marginalising out the unobserved variables from the per model joint densities $\mathrm{p}_{\mathbf{y},\mathbf{x}_m|\mathrm{m}}$

$$\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid m) = \int_{X_m} \mathrm{p}_{\mathbf{y}|\mathrm{m},\mathbf{x}_m}(\boldsymbol{y} \mid m, \boldsymbol{x}) \mathrm{p}_{\mathbf{x}_m}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}. \tag{1.48}$$

This value is equivalent to the denominator in Bayes' theorem (1.42), this explaining the naming of this term as the *model evidence*.

As described previously, evaluating the model evidence requires integrating across the space of all unobserved variables. The key computational challenge in being able to perform probabilistic model comparison with complex high dimensional models is therefore again being able to efficiently to compute integrals in high dimensional spaces. Unlike the integrals required for making predictions using a single model

however, the model evidence integral cannot be naturally expressed as an expectation of a function with respect to the posterior distribution. This can complicate approximate computation of model evidence terms compared to other quantities involved in inference. In Chapter 5 we will consider extensions to a class of methods proposed for estimating model evidence terms.

A common criticism of the model comparison framework we have described is that the model posterior probabilities $p_{m|\mathbf{y}}$ can be highly sensitive to the choice of the prior distribution placed on the unobserved per-model variables $P_{\mathbf{x}_m}$ [2, 136]. Within the context of Bayesian inference the prior distribution is often viewed as a distinct entity from the observation model $P_{\mathbf{y}|m,\mathbf{x}_m}$, with the prior understood as encoding our beliefs about the unobserved variables $\mathbf{x}_m$ before observing data. That the model evidence terms and so model posterior can be sensitive to the specific choices of prior distributions is therefore viewed as a disadvantage as the priors are viewed as being somewhat arbitrary or subjective. This in turn means the model posterior probabilities are similarly subjective and this subjectivity is viewed as inherently undesirable.

In our opinion this criticism is ill-founded. All inferences are inherently subjective. Both the observation model and prior are based on assumptions about a problem [95] and it is their combination which defines an overall generative model which we use to perform inference with. A prior distribution can only be interpreted in the context of the observation model it is combined with [99] and is no more or less subjective than that observation model. If it is reasonable for the model posterior to be sensitive to the choice of the observation model, it therefore seems equally reasonable (and can be argued to be desirable [258]) for it to be sensitive to the choice of prior. If a practitioner is worried about a deleterious effect of arbitrarily chosen priors on the quality of model comparison results, this could be argued to be reflective of a need to improve the choice of prior rather than indicating an issue in the inference framework.

A further criticism levelled at probabilistic model comparison (and probabilistic modelling more generally) is the potential difficulties in interpreting inferences under a setting of model misspecification, i.e. when the generative model (or models) used to perform inference does not match the true data generating process. Bernardo and Smith proposed the nomenclature of *M-closed* to refer to model comparison under a

setting in which the list of models being compared includes the 'true' model describing the data generating process as a member, and *M-open* for the case in which the 'true' model is not included [24, §6.1.2][6]. The model posterior probabilities $p_{m|y}$ can be interpreted in an M-closed setting as degrees beliefs of which of the set of models is the true model, however their intepretation in the M-open case is less clear.

As commented in the introduction to this section, virtually all models are simplifications of much more complex processes, so in this sense inference with misspecified models and model comparison in an M-open setting is the norm. In [24], Bernardo and Smith suggest reposing model comparison as a decision problem in maximising the expectation of a utility function, for example the accuracy of predictions about future observations. Various alternatives have been proposed to the model comparison framework described above based on this idea, for example using cross-validation to estimate predictive accuracy of models on held-out data [63]. The use of nonparameteric models which are sufficiently flexible able to arbitarily closely approximate the underlying data generating process (given sufficient data) as a proxy reference model has also been proposed [120, 155].

Although the issue of interpretation of inferences under model misspecification is important from both a philosophical and practical perspective, in this thesis we concentrate solely on the computational aspects of inference, and we will assume being able to (approximately) compute model evidence integrals is at least in some cases desirable without making any claims to the validity of the probabilistic model comparison framework in a particular setting. Further though we have motivated the computation of model evidence integrals within a setting of comparing multiple models, we will only directly consider computation of model evidence terms for a single model, with the implicit assumption that this could be repeated for all models of interest to allow (in conjuction with specification of prior probabilities on the models) estimation of the model posterior probabilities.

6 A third alternative *M-completed* is also defined for the case where the true model is known but not included in set of models being compared due to being computational intractable or non-interpretable.

1.4 SUMMARY

Probabilistic modelling offers a natural way to formalise our beliefs and assumptions about a problem and make inferences given those beliefs. Once a model has been defined the theoretical basis of the inference process is elegantly simple. Underlying this simplicity however are some very significant implementation challenges. The key computational task is the evaluation of integrals across high-dimensional spaces, which typically do not have closed form solutions and are intractable to compute using standard numerical integration approaches.

This intractability necessitates the use of approximate inference methods, the focus of this thesis. In particular we propose several novel extensions to MCMC methods, a class of approaches for drawing dependent samples from high-dimensional target distributions. In the next chapter we review the basic Monte Carlo method for integration and associated methods for generating and using independent pseudo-random variates to estimate integrals. We then introduce the key Markov chain theory underlying MCMC methods and review some key existing MCMC algorithms. We will then conclude with an outline of the remainder of the thesis, in particular giving a a summary of the novel contributions made and how these relate to the specific inference problems discussed in the last section of this chapter.

2 APPROXIMATE INFERENCE

In the previous chapter we argued that the key computational challenge in performing inference in probabilistic models is being able to evaluate integrals with respect to probability distributions defined on high-dimensional spaces. Generally these integrals will not have analytic solutions and for models with even moderate numbers of unobserved variables, numerical quadrature approaches to evaluating integrals are computationally infeasible.

Truth is much too complicated to allow anything but approximations.
—John von Neumann

In this chapter we will review some of the key algorithms proposed for computing *approximate* solutions to inference problems. A unifying aspect to all of these methods is trading off some loss of the accuracy of the answers provided to inferential queries, for a potentially significant increase in computational tractability. The literature on *approximate inference* methods is vast and so necessarily this chapter will only be a partial review of the methods directly relevant to this thesis.

Approximate inference methods can be roughly partitioned into two groups: methods in which integrals with respect to the target distribution are estimated by averaging over samples from a distribution over the target space and those in which a more tractable approximation to the target distribution is found by optimising the approximation to be 'close' to the target distribution. In this chapter we will concentrate on the sampling-based approaches to approximate inference.

In particular we will focus on *Markov chain Monte Carlo* (MCMC) methods, as these form the key basis for the contributions discussed in later chapters. We will review the key theory underlying Monte Carlo integration and MCMC methods and some of the standard algorithms for implementing these approaches. We will conclude with a discussion of auxiliary variable MCMC methods which are central to the methods discussed in the remainder of this thesis.

Although they are not the main focus of this thesis we will make use of several optimisation-based approximate inference methods within the algorithms discussed in the following chapters. We review the ideas underlying these methods in Appendix C.

2.1 MONTE CARLO METHODS

Inference at both the level of computing conditional expectations of unobserved variables in a model and in evaluating evidence terms to allow model comparison involves integrating functions against a probability distribution. Typically the distribution of interest will be defined by a probability density with respect to a base measure. Therefore we wish to be able to compute integrals of the form

$$\int_X f(\boldsymbol{x})\, P(\mathrm{d}\boldsymbol{x}) = \int_X f(\boldsymbol{x})\, p(\boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x}) \tag{2.1}$$

where p is the density of a target distribution P on a space X with respect to a base measure μ and f is a measurable function. For instance in the case of computing the *posterior mean* in a Bayesian inference problem with observed variables $\mathbf{y}$ and latent variables $\mathbf{x}$ where the posterior density $\mathrm{p}_{\mathbf{x}|\mathbf{y}}$ is defined with respect to the D-dimensional Lebesgue measure, we would have $p(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y})$ for an observed $\boldsymbol{y}$, $\mu(\boldsymbol{x}) = \lambda^D(\boldsymbol{x})$ and $f(\boldsymbol{x}) = \boldsymbol{x}$. Often we will only be able to evaluate p up to an unknown normalising constant i.e. $p(\boldsymbol{x}) = \frac{1}{Z}\tilde{p}(\boldsymbol{x})$ with we able to evaluate $\tilde{p}$ pointwise but Z intractable to compute. For example in a Bayesian inference setting $\tilde{p}(\boldsymbol{x})$ would be the joint density $\mathrm{p}_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y})$ and Z the model evidence $\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})$. When performing inference in undirected models, we would instead have that $\tilde{p}$ is the product of unnormalised factors and Z the corresponding normaliser.

2.1.1 Monte Carlo integration

The eponym of the Monte Carlo method is a Monocan casino, favoured haunt of the uncle of Stanisław Ulam, one of the method's inventors.

The framework that unifies all of the methods we will discuss in this section is the *Monte Carlo* method for integration [256]. Let $\mathbf{x}$ be a random vector distributed according to the target distribution i.e. $\mathrm{P}_{\mathbf{x}} = P$. Given an arbitrary measurable function $f : X \to \mathbb{R}$ we define a random variable $\mathrm{f} = f(\mathbf{x})$. Our task is to compute expectations $\mathbb{E}[\mathrm{f}] = \bar{f}$ corresponding to the integral (2.1). We assume that $\mathbb{E}[\mathrm{f}]$ exists and both $\mathbb{E}[\mathrm{f}]$ and $\mathbb{V}[\mathrm{f}]$ are finite. For now we assume we have a way of generating values of N random variables $\{\mathbf{x}_n\}_{n=1}^N$, each marginally distributed according to the target distribution i.e. $\mathrm{P}_{\mathbf{x}_n} = P \; \forall n \in 1 \ldots N$ but potentially not independent of each other. We define random variables

$$\mathrm{f}_n = f(\mathbf{x}_n) \quad \forall n \in \{1 \ldots N\} \quad \text{and} \quad \hat{\mathrm{f}}_N = \frac{1}{N}\sum_{n=1}^{N} \mathrm{f}_n. \tag{2.2}$$

Due to linearity of the expectation operator, we have that

$$\mathbb{E}\left[\hat{\mathsf{f}}_N\right] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[\mathsf{f}_n] = \frac{1}{N}\sum_{n=1}^{N}\bar{f} = \bar{f} \tag{2.3}$$

and so that in expectation $\hat{\mathsf{f}}_N$ is equal to $\bar{f}$, i.e. realisations of $\hat{\mathsf{f}}_N$ are *unbiased estimators* of $\bar{f}$. Note that this result does not require any independence assumptions about the generated random variables. Now considering the variance of $\hat{\mathsf{f}}_N$ we can show that

$$\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \frac{\mathbb{V}[\mathsf{f}]}{N}\left(1 + \frac{2}{N}\sum_{n=1}^{N-1}\sum_{m=1}^{n-1}\frac{\mathbb{C}[\mathsf{f}_n, \mathsf{f}_m]}{\mathbb{V}[\mathsf{f}]}\right). \tag{2.4}$$

If the generated random variables $\{\mathbf{x}_n\}_{n=1}^N$ and so $\{\mathsf{f}_n\}_{n=1}^N$ are independent, then $\mathbb{C}[\mathsf{f}_n, \mathsf{f}_m] = 0\ \forall m \neq n$. In this case (2.4) reduces to $\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \mathbb{V}[\mathsf{f}]/N$, i.e. the variance of the *Monte Carlo estimate* $\hat{\mathsf{f}}_N$ for $\bar{f}$ is inversely proportional to the number of samples N. Importantly this scaling does not depend on the dimension of $\mathbf{x}$.

Therefore if we can generate a set of independent random variables from the target distribution, we can estimate expectations that asymptotically tend to the true value as N increases, with a typical deviation from the true value (as measured by the standard deviation, i.e. the square root of variance) that is $O\left(N^{-\frac{1}{2}}\right)$. In comparison a fourth-order quadrature method such as *Simpson's rule* has an error that is $O\left(N^{-\frac{4}{D}}\right)$ for a grid of N points uniformly spaced across a D dimensional space. Asymptotically for $D > 8$, Monte Carlo integration will therefore give better convergence than Simpson's rule, and even for smaller dimensions large constant factors in the Simpson's rule dependence can mean Monte Carlo performs better for practical N.

Note that computing Monte Carlo estimates from independent random variables is not optimal in terms of minimising $\mathbb{V}\left[\hat{\mathsf{f}}_N\right]$ for a given f; the covariance terms in (2.4) can be negative which can reduce the overall variance. A wide range of *variance reduction* methods have been proposed to exploit this and produce lower variance of Monte Carlo estimates for a given f [143]. Although these methods can be important in practice for achieving an estimator with a practical variance for a specific f of interest, we will generally concentrate on the case where we do not necessarily know f in advance.

Figure 2.1.: Binary representation of linear congruential generator sequence $s_{n+1} = 37s_n + 61 \bmod 128$. Columns left to right represents successive integer states in sequence. From least significant (bottom) to most significant (top), the bits in each column have patterns repeating with periods 2, 4, 8, 16, 32, 64, 128.

2.1.2 Pseudo-random number generation

The generation of random numbers is too important to be left to chance.
—Robert R. Coveyou

Virtually all statistical computations involving random numbers in practice make use of *pseudo-random number generators* (PRNGs). Rather than generating samples via a truly random process[1], PRNGs produce deterministic sequences of integers in a fixed range that nonetheless maintain many of the properties of a random sequence. In particular through careful choice of the updates, sequences with a very long period (number of iterations before the sequence begins repeating), a uniform distribution across the numbers in the sequence range and low correlation between successive states can be constructed.

A very simple example of a class of PRNGs is the *linear congruential generator* [149] which obeys the recurrent update

$$s_{n+1} = (as_n + c) \mod m \quad \text{with} \quad 0 < a < m,\ 0 \leq c < m, \tag{2.5}$$

with a, c and m integer parameters. If a, c and m are chosen appropriately, iterating the update (2.5) from an initial seed $0 \leq s_0 < m$, will produce a sequence of states which visits all the integers in $[0, m)$ before repeating. An example state sequence with $m = 128$ is shown in Figure 2.1. In practice, linear congruential generators produce sequences with poor statistical properties, particularly when used to generate random points in high dimensional spaces [166], hence most modern numerical computing libraries use more robust PRNGs such as the *Mersenne-Twister* [167], which is used in all experiments in this thesis.

The raw output of a PRNG is an integer sequence, with typically the sequence elements uniformly distributed over all integers in a range $[0, 2^n)$ for some $n \in \mathbb{N}$. All real values are represented at a finite precision on computers, typically using a floating point representation [11] of *single* (24-bit mantissa) or *double* (53-bit mantissa) precision. Through

1 We consider a true random process as one in which it is impossible to precisely predict the next value in the sequence given the previous values.

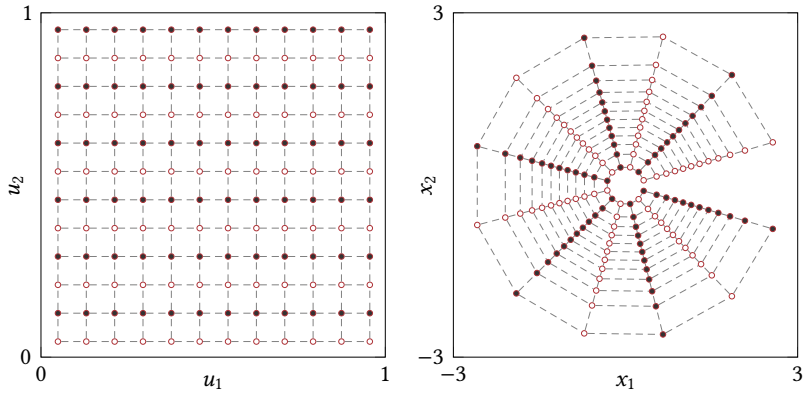


Figure 2.2.: Visualisation of Box–Muller transform. Left axis shows uniform grid on $U = [0,1]^2$ and right-axis shows grid points after mapping through $\boldsymbol{g}$ in transformed space $X = \mathbb{R}^2$.

an appropriate linear transformation, the integer outputs of a PRNG can be converted to floating-point values uniformly distributed across a finite interval. PRNG implementations typically provide a primitive to generate floating-point values uniformly distributed on $[0,1)$. Given the ability to generate sequences of (effectively) independent samples from a uniform distribution $\mathcal{U}(0,1)$, the question is then how to use these to produce random samples from arbitrary densities.

2.1.3 Transform sampling

Samples from many standard distributions can be generated by exploiting the transformation of random variables relationships discussed in 1.1.4. Let $\mathbf{u}$ be a D-dimensional vector of independent random variables marginally distributed according to $\mathcal{U}(0,1)$ and $\boldsymbol{g} : [0,1)^D \to X$ be a diffeomorphism with $X \subseteq \mathbb{R}^D$. If we define $\mathbf{x} = \boldsymbol{g}(\mathbf{u})$, then by the change of variables formula (1.22) we have that

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \left| \frac{\partial \boldsymbol{g}^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|. \tag{2.6}$$

For example for $D = 2$, $X = \mathbb{R}^2$ and a bijective map $\boldsymbol{g}$ defined by

$$\boldsymbol{g}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{bmatrix} \sqrt{-2\log u_1}\cos(2\pi u_1) \\ \sqrt{-2\log u_1}\sin(2\pi u_2) \end{bmatrix}, \quad \boldsymbol{g}^{-1}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{bmatrix} \exp\left(-\frac{1}{2}(x_1^2 + x_2^2)\right) \\ \frac{1}{2\pi}\arctan\left(\frac{x_1}{x_2}\right) \end{bmatrix},$$

then we have that the density of the transformed $\mathbf{x} = \boldsymbol{g}(\mathbf{u})$ is

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x_1^2}{2}\right)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x_2^2}{2}\right), \tag{2.7}$$

i.e. x_1 and x_2 are independent random variables with standard normal distributions $\mathcal{N}(0,1)$. This is the *Box–Muller transform* [48], and allows generation of independent standard normal variables given a PRNG primitive for sampling from $\mathcal{U}(0,1)$. A visualisation of the transformation of space applied by the method is shown in Figure 2.2.

A general method for sampling from univariate distributions is to use an inverse *cumulative distribution function* (CDF) transform. For a probability density p on a scalar random variable, the corresponding CDF $r : \mathbb{R} \to [0,1]$ is defined as

$$r(x) = \int_{-\infty}^{x} p(v)\,\mathrm{d}v \implies \frac{\partial r(x)}{\partial x} = p(x). \tag{2.8}$$

If u is a standard uniform random variable and $\mathrm{x} = r^{-1}(\mathrm{u})$ then

$$\mathrm{p}_{\mathrm{x}}(x) = \left|\frac{\partial r(x)}{\partial x}\right| = p(x). \tag{2.9}$$

To be able to use the inverse CDF transform method we need to be able to evaluate r^{-1}, sometimes termed the *quantile function*. Often neither the CDF or quantile function of a univariate distribution will have closed form solutions however we can use polynomial approximation methods and iterative solvers to evaluate both to arbitrary precision [197]. For some distributions such as the standard normal $\mathcal{N}(0,1)$ even though the CDF and quantile function do not have analytic forms in terms of elementary functions it is common for numerical computing libraries to provide approximations to both which are accurate to within small multiples of machine precision. Although the inverse CDF transform method gives a general recipe for sampling from univariate densities, it is not easy to generalise to multivariate densities and alternatives can be simpler to implement and more numerically stable.

2.1.4 Rejection sampling

An important class of generic sampling methods, particularly due their use as a building block in other algorithms, is rejection sampling [193]. Rejection sampling uses the observation that to sample from a distribution with density $p : X \to [0,\infty)$ it is sufficient to uniformly sample from the volume under the graph of $(\boldsymbol{x}, p(\boldsymbol{x}))$.

The key requirement in rejection sampling is to identify a *proposal distribution* Q which we can generate independent samples from and has

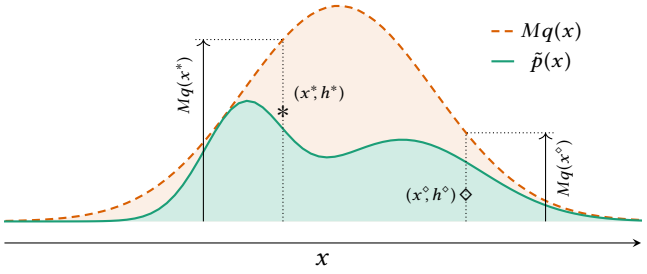


Figure 2.3.: Visualisation of rejection sampling. The green curve shows the (unnormalised) target density $\tilde{p}$, the green region underneath representing the area we wish to sample points uniformly from. The dashed orange curve shows the scaled proposal density Mq, with the orange (plus green) region representing the area we uniformly propose values from. Two example proposals are shown: $\diamond$ is under the target density and so accepted; $*$ is outside of the green region and so would be rejected.

Algorithm 1 Rejection sampling.

Input: $\tilde{p}$: unnormalised target density, q : normalised density of proposal distribution Q, M : constant such that $\tilde{p}(\boldsymbol{x}) \leq Mq(\boldsymbol{x})\ \forall \boldsymbol{x} \in X$.
Output: Independent sample from distribution with density $p(\boldsymbol{x}) = \frac{1}{Z}\tilde{p}(\boldsymbol{x})$.

1: **repeat**
2: $\quad \boldsymbol{x} \sim q(\cdot)$
3: $\quad h \sim \mathcal{U}(\cdot \mid 0, Mq(\boldsymbol{x}))$
4: **until** $h \leq \tilde{p}(\boldsymbol{x})$
5: **return** $\boldsymbol{x}$

a density $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$ that upper bounds the potentially unnormalised target density $\tilde{p}$ across its full support X when multiplied by a known constant M, i.e. $\tilde{p}(x) \leq Mq(x)\ \forall x \in X$. The requirement to be able to generate independent samples from Q can be met for example by distributions amenable to transform sampling, e.g. the standard normal. The second requirement is generally more challenging and as we will see the efficiency of rejection sampling methods is very dependent on how tight the bound can be made.

Algorithm 1 describes the rejection sampling method to produce a single independent sample from a target distribution. A visualisation of how the algorithm works for a univariate target distribution is shown in Figure 2.3. The overall aim is to generate points uniformly distributed across the green area under the (unnormalised) target density curve. This is achieved by generating points uniformly under the dashed orange curve corresponding to the scaled proposal density and then accepting only those which are below the green curve. To generate a point

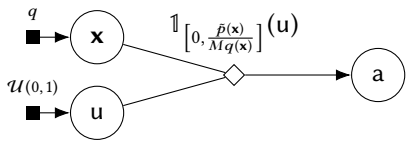


Figure 2.4.: Factor graph of rejection sampling process.

under the dashed orange curve we first generate an x from the proposal distribution and then generate an auxiliary 'height' variable by sampling uniformly from $[0, Mq(x)]$. If the sampled height is below the green curve we accept (as in the $\diamond$ example in Figure 2.3) else we reject the sample (as in the $*$ example in Figure 2.3).

Figure 2.4 shows the rejection sampling generative process as a directed factor graph, with $\mathbf{x}$ be a random variable representing the proposal, u the uniform auxiliary variable drawn to sample the 'height' and a a binary variable that indicates whether the proposal is accepted ($a = 1$) or not ($a = 0$). By marginalising out u we have that that

$$\mathrm{p}_{\mathbf{x},\mathrm{a}}(\boldsymbol{x}, a) = q(\boldsymbol{x})\left(\frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}\right)^{a}\left(1 - \frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}\right)^{1-a}, \tag{2.10}$$

and further marginalising over the proposal $\mathbf{x}$

$$\mathrm{p}_{\mathrm{a}}(a) = \left(\frac{Z}{M}\right)^{a}\left(1 - \frac{Z}{M}\right)^{1-a}. \tag{2.11}$$

Conditioning on the proposal being accepted we therefore have that

$$\mathrm{p}_{\mathbf{x}|\mathrm{a}}(\boldsymbol{x} \mid 1) = \frac{q(\boldsymbol{x})\frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}}{\frac{Z}{M}} = \frac{\tilde{p}(\boldsymbol{x})}{Z} = p(\boldsymbol{x}). \tag{2.12}$$

Therefore the accepted proposals are distributed according to the target density as required. Further from (2.11) we have that the $\mathrm{p}_{\mathrm{a}}(1) = \frac{Z}{M}$. This suggests we can use the accept rate to estimate Z but also hints at the difficulty in finding a M which guarantees the upper bound requirement as for $\frac{Z}{M}$ to be a valid probability $M \geq Z$ i.e. M needs to be an upper bound on the unknown normalising constant Z. This relationship also suggests it is desirable to set M as small as possible to maximise the acceptance rate.

Although rejection sampling can be an efficient method of sampling from univariate target distributions (particularly for distributions with log-concave densities where adaptive variants are available [106]), it

generally scales very poorly with the dimensionality of the target distribution. This is as the ratio of the volume under the target density to the volume under the scaled proposal density (in terms of Figure 2.3 the ratio of the green area to the green plus orange regions), and so the probability of accepting a proposal, will tend become exponentially smaller as the dimensionality increases. This is an example of the so-called *curse of dimensionality*. Therefore although rejection sampling can be a useful subroutine for generating random variables from low-dimensional distributions, in general it is not a viable option for generating samples directly for high-dimensional Monte Carlo integration.

2.1.5 Importance sampling

So far we have considered methods for generating samples directly from a target distribution. Although samples can be of value in themselves for giving a representative set of plausible values from the target distribution (e.g. for visualisation purposes), usually the end goal is to estimate integrals of the form in (2.1).

Importance sampling [135] is a Monte Carlo method which allows arbitrary integrals to be estimated. If Q is a distribution, with density $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$, which is absolutely continuous with respect to the target distribution (which requires that $p(\boldsymbol{x}) > 0 \Rightarrow q(\boldsymbol{x}) > 0$), then importance sampling is based on the identity

$$\bar{f} = \frac{\int_X f(\boldsymbol{x})\,\tilde{p}(\boldsymbol{x})\,\mu(\mathrm{d}\boldsymbol{x})}{\int_X \tilde{p}(\boldsymbol{x})\,\mu(\mathrm{d}\boldsymbol{x})} = \frac{\int_X f(\boldsymbol{x})\,\frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\,q(\boldsymbol{x})\,\mu(\mathrm{d}\boldsymbol{x})}{\int_X \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\,q(\boldsymbol{x})\,\mu(\mathrm{d}\boldsymbol{x})}. \tag{2.13}$$

Each of the numerator and denominator in (2.13) take the form of an expectation of a measurable function of a random variable $\mathbf{x}$ with distribution Q. Further the denominator is exactly equal to $Z = \int_X \tilde{p}(\boldsymbol{x})\,\mu(\mathrm{d}\boldsymbol{x})$. We therefore have that

$$Z\bar{f} = \mathbb{E}[w(\mathbf{x})f(\mathbf{x})] \;\text{ and }\; Z = \mathbb{E}[w(\mathbf{x})] \;\text{ with }\; w(\boldsymbol{x}) = \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}. \tag{2.14}$$

If we can generate random variables $\{\mathbf{x}_n\}_{n=1}^N$ each marginally distributed according to Q we can therefore form Monte Carlo estimates of both the numerator and denominator. We define $\hat{\mathrm{Z}}_N$ and $\hat{\mathrm{g}}_N$ as

$$\hat{\mathrm{Z}}_N = \frac{1}{N}\sum_{n=1}^N w(\mathbf{x}_n) \;\text{ and }\; \hat{\mathrm{g}}_N = \frac{1}{\hat{\mathrm{Z}}}\sum_{n=1}^N w(\mathbf{x}_n)f(\mathbf{x}_n). \tag{2.15}$$

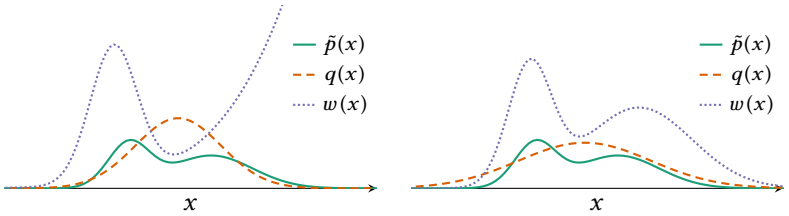


Figure 2.5.: Visualisation of importance sampling. On both axes the green curve shows the unnormalised target density $\tilde{p}$, the dashed orange curve the density q values are sampled from and the dotted violet curve the importance weighting function $w(x) = \frac{\tilde{p}(x)}{q(x)}$ to estimate expectations with respect to the target density using samples from q. In the left axis the q chosen is under-dispersed compared to $\tilde{p}$ leading to very large w values in the right tail. In contrast in the right axis, the broader q leads to less extreme variation in w.

By the same argument as Section 2.1.1, $\mathbb{E}\left[\hat{Z}_N\right] = Z$ and $\mathbb{E}[\hat{g}_N] = Z\bar{f}$. We can therefore use importance sampling to form an unbiased estimate of the unknown normalising constant Z.

If we define $\hat{f}_N = \hat{g}_N / \hat{Z}_N$, then this is a biased[2] estimator for $\bar{f}$ as in general the expectation of the ratio of two random variables is not equal to the ratios of their expectations. However if both the numerator and denominator have finite variance, i.e. $\mathbb{V}\left[\hat{Z}_N\right] < \infty$ and $\mathbb{V}[\hat{g}_N] < \infty$, then $\hat{f}_N$ is a *consistent* estimator for $\bar{f}$ i.e. $\lim_{N\to\infty} \mathbb{E}\left[\hat{f}_N\right] = \bar{f}$.

The $w(x_n)$ values are typically termed the *importance weights*. If a few of the weights are very large, the weighted sums in (2.15) will be dominated by those few values, reducing the effective number of samples in the Monte Carlo estimates. This can particularly be a problem if the are regions of X with low probability under q where $p(\boldsymbol{x}) \gg q(\boldsymbol{x})$. As sampling points in these regions will be a rare event, a large number of samples may be needed to diagnose the issue adding further difficulty. A general recommendation is to use densities q with tails as least as heavy of those of p, and in general the closer the match between q and p the better [161, 198]. Figure 2.5 shows a visualisation of the effect of the choice of q on the importance weights.

When previously discussing rejection sampling, we introduced an auxiliary binary accept indicator variable, a, associated with each proposed

2 In cases where the normalising constant Z is known, we can instead use $w(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$ in which case the ratio estimator is not required and an unbiased estimates can be calculated. As the problems we are interested in will generally have unknown Z we do not consider this case further

sample $\mathbf{x}$ (see Figure 2.4). If we generate N independent proposal – indicator pairs $\{\mathbf{x}_n, \mathsf{a}_n\}_{n=1}^N$ then the number of accepted proposals is $\mathsf{N}_{\text{acc}} = \sum_{n=1}^N \mathsf{a}_n$. Conditioned on N_{acc} being a value more than one, the generated rejection sampling variables $\{\mathbf{x}_n, \mathsf{a}_n\}_{n=1}^N$ can be used to form an *unbiased* Monte Carlo estimate of $\bar{f}$ using the estimator

$$\hat{\mathsf{f}}_N^{\text{RS}} = \frac{\sum_{n=1}^N \mathsf{a}_n f(\mathbf{x}_n)}{\sum_{m=1}^N \mathsf{a}_m}, \tag{2.16}$$

which just corresponds to computing the empirical mean of the accepted proposals i.e. the standard Monte Carlo estimator. In comparison importance sampling forms a biased but consistent estimator for $\bar{f}$ from N samples $\{\mathbf{x}_n\}_{n=1}^N$ from a distribution Q using the estimator

$$\hat{\mathsf{f}}_N^{\text{IS}} = \frac{\sum_{n=1}^N w(\mathbf{x}_n) f(\mathbf{x}_n)}{\sum_{m=1}^N w(\mathbf{x}_m)}. \tag{2.17}$$

From this perspective the accept indicators a_n in rejection sampling can be seen to act like binary importance weights, in contrast importance sampling using 'soft' weights which mean all sampled $\mathbf{x}_n$ make a contribution to the estimator (assuming $w(\boldsymbol{x}) \neq 0 \; \forall \boldsymbol{x} \in X$). However this correspondence is only loose. The rejection sampling estimator $\hat{\mathsf{f}}_N^{\text{RS}}$ is unbiased unlike $\hat{\mathsf{f}}_N^{\text{IS}}$, but this unbiasedness relies on conditioning on a non-zero value for N_{acc} (i.e. the number of accepted samples to generate) and continuing to propose points until this condition is met. In contrast importance sampling generates a fixed number of samples from Q and does not use any auxiliary random variables.

Unlike rejection sampling, there is no need in importance sampling for q to upper-bound the target density. This allows more freedom in the choice of q however it is still important to choose q to be as close as possible to the target while remaining tractable to generate samples from. In general for target densities defined on high-dimensional spaces, it can be difficult to find an appropriate q such that the variation in importance weights is not too extreme [161].

2.2 MARKOV CHAIN MONTE CARLO

When introducing the Monte Carlo method we saw that is was not necessary for the random variables used in a Monte Carlo estimator to be independent. While it can be impractically computationally expens-

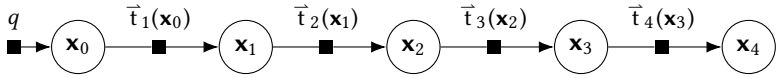


Figure 2.6.: Markov chain factor graph. The initial state $\mathbf{x}_0$ is sampled according to a density q and each subsequent state $\mathbf{x}_n$ is then generated from a transition density $\vec{\mathsf{t}}_n$ conditioned on the previous state $\mathbf{x}_{n-1}$.

ive to generate independent samples from complex high-dimensional target distributions, simulating a stochastic process which converges in distribution to the target and produces a sequence of *dependent* random variables is often a more tractable task. This is the idea exploited by *Markov chain Monte Carlo* (MCMC) methods.

A *Markov chain* is an ordered sequence of random variables $\{\mathbf{x}_n\}_{n=0}^{N}$ which have the *Markov property* — for all $n \in \{1 \dots N\}$, $\mathbf{x}_n$ is conditionally independent of $\{\mathbf{x}_n\}_{m<n-1}$ given $\mathbf{x}_{n-1}$. This conditional independence structure is visualised as a factor graph in Figure 2.6.

For a Markov chain defined on a general measurable state space $(X, \mathcal{F})$, the probability distribution of a state $\mathbf{x}_n$ given the state $\mathbf{x}_{n-1}$ is specified for each $n \in \{1 \dots N\}$ by a *transition operator*, $\vec{\mathsf{T}}_n : \mathcal{F} \times X \to [0, 1]$. In particular the transition operators define a series of regular conditional distributions for each $n \in \{1 \dots N\}$

$$\mathsf{P}_{\mathbf{x}_n | \mathbf{x}_{n-1}}(A \mid \boldsymbol{x}) = \vec{\mathsf{T}}_n(A \mid \boldsymbol{x}) \quad \forall A \in \mathcal{F},\ \boldsymbol{x} \in X. \tag{2.18}$$

We will often assume that the chain is *homogeneous*, i.e. that the same transition operator is used for all steps $\vec{\mathsf{T}}_n = \vec{\mathsf{T}}\ \forall n \in \{1 \dots N\}$.

The key property required of a transition operator for use in MCMC methods is that the target distribution P is *invariant* under the transition, that is it satisfies

$$P(A) = \int_X \vec{\mathsf{T}}(A \mid \boldsymbol{x})\, P(\mathrm{d}\boldsymbol{x}) \quad \forall A \in \mathcal{F}, \tag{2.19}$$

The invariance property means that if a chain state $\mathbf{x}_n$ is distributed according to the target P, all subsequent chain states $\mathbf{x}_{n+1}$, $\mathbf{x}_{n+2} \dots$ will also be marginally distributed according to the target. Therefore given a single random sample $\mathbf{x}_0$ from the target distribution, a series of dependent states marginally distributed according to the target could be generated and used to form Monte Carlo estimates of expectations.

Being able to generate even one exact sample from a complex high-dimensional target distribution is generally infeasible. Importantly however the marginal distribution on the chain state $\mathrm{P}_{\mathbf{x}_n}$ of a Markov chain with a transition operator which leaves the target distribution invariant will converge to the target distribution irrespective of the distribution of the initial chain state if the target distribution is the *unique* invariant distribution of the chain.

To have a unique invariant distribution, a chain must be *irreducible* and *aperiodic* [250]. For a chain on a measurable space $(X, \mathcal{F})$, irreducibility is defined with respect to a measure ν, which could but does not necessarily need to be the target distribution P. A chain is ν-irreducible if starting at any point in X there is a non-zero probability of moving to any set with positive ν-measure in a finite number of steps, i.e.

$$\forall \boldsymbol{x} \in X,\, A \in \mathcal{F} : \nu(A) > 0 \;\; \exists\, m \in \mathbb{Z}^+ : \mathrm{P}_{\mathbf{x}_m|\mathbf{x}_0}(A \,|\, \boldsymbol{x}) > 0. \tag{2.20}$$

A chain is periodic (and aperiodic otherwise) if disjoint regions of X are visited cyclically, i.e. there exists an integer $r > 1$ and an ordered set of r disjoint P-positive subsets of X, $\{A_i\}_{i=1}^{r}$ such that $\vec{\mathrm{T}}(A_j \,|\, \boldsymbol{x}) = 1\; \forall \boldsymbol{x} \in A_i,\; i \in \{1 \ldots r\},\; j = (i+1) \mod r$.

If we can construct a ν-irreducible and aperiodic Markov chain $\{\mathbf{x}_n\}_{n=0}^{N}$ which has the target distribution P as its invariant distribution, then a MCMC estimator $\hat{\mathrm{f}}_N = \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x}_n)$ converges almost surely as $N \to \infty$ to $\bar{f} = \int_X f \mathrm{d}P$ for all starting states except for a ν-null set[3] [174]. This convergence of *time-averages* (i.e. over states at different steps of the Markov chain) to *space-averages* (i.e. with respect to the stationary distribution across the state space), is termed *ergodicity* and is a consequence of the *Birkhoff–Khinchin ergodic theorem* [40].

Although irreducibility and aperiodicity of a Markov chain which leaves the target distribution invariant are sufficient for convergence of MCMC estimators, this does not tell us anything about the rate of that convergence and so how to quantify the error introduced by computing estimates with a Markov chain simulated for only a finite number of steps. Stronger notions of ergodicity can be used to help quantify convergence; we will concentrate on *geometric ergodicity* here. We first

3 The 'except for a ν-null set' caveat can be removed by requiring the stronger property of *Harris recurrence* [122].

define a notion of distance between two measures μ and ν on a measurable space $(X, \mathcal{F})$, the *total variation distance*, as

$$\|\mu - \nu\|_{\text{TV}} = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)|. \tag{2.21}$$

For a ν-irreducible and aperiodic chain with invariant distribution P our earlier statement that the distribution on the chain state converges to P can now be restated more precisely as that for ν-almost all initial states $\mathbf{x}_0 = \boldsymbol{x}$, $\lim_{n\to\infty} \|\mathsf{P}_{\mathbf{x}_n|\mathbf{x}_0}(\cdot \mid \boldsymbol{x}) - P\|_{\text{TV}} = 0$. Geometric ergodicity makes a stronger statement that the convergence in total variation distance is geometric in n, i.e. that

$$\|\mathsf{P}_{\mathbf{x}_n|\mathbf{x}_0}(\cdot \mid \boldsymbol{x}) - P\|_{\text{TV}} \leq m(\boldsymbol{x}) r^n \tag{2.22}$$

for a positive measurable function m which depends on the initial chain state $\boldsymbol{x}$ and rate constant $r \in [0, 1)$. For chains which are geometrically ergodic, we can derive an expression for the *asymptotic variance* of an MCMC estimator $\hat{\mathsf{f}}_N$ related to the variance of a simple Monte Carlo estimator previously considered in Section 2.1.1.

A stochastic process is stationary if the joint distribution of the states at any set of time points does not change if all those times are shifted by a constant.

As in Section 2.1.1 we define $\mathsf{f}_n = f(\mathbf{x}_n)$ and $\hat{\mathsf{f}}_N = \frac{1}{N}\sum_{n=1}^N \mathsf{f}_n$, with here the $\{\mathbf{x}_n\}_{n=1}^N$ the states of a Markov chain. For a homogeneous Markov chain with a unique invariant distribution P which is *stationary*, the marginal distribution on the states $\mathsf{P}_{\mathbf{x}_n}$ is equal to P for all n and we can use the expression for the variance of a general Monte Carlo estimator (which did not assume independence of the random variables) stated earlier in (2.4). Further the stationarity of the chain means that the covariance $\mathbb{C}[\mathsf{f}_n, \mathsf{f}_m]$ depends only on the difference $n - m$, and so the variance of the estimator simplifies to

$$\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \frac{\mathbb{V}[\mathsf{f}]}{N}\left(1 + 2\sum_{n=1}^{N-1}\left(\frac{N-n}{N}\frac{\mathbb{C}[\mathsf{f}_0, \mathsf{f}_n]}{\mathbb{V}[\mathsf{f}]}\right)\right). \tag{2.23}$$

If we multiply both sides of (2.23) by N and define $\rho_n = \frac{\mathbb{C}[\mathsf{f}_0, \mathsf{f}_n]}{\mathbb{V}[\mathsf{f}]}$ (the lag n autocorrelations of $\{\mathsf{f}_n\}$), under the assumption that $\sum_{n=1}^\infty |\rho_n| < \infty$ in the limit of $N \to \infty$ we have that

$$\lim_{N\to\infty}\left(N\,\mathbb{V}\left[\hat{\mathsf{f}}_N\right]\right) = \mathbb{V}[\mathsf{f}]\left(1 + 2\sum_{n=1}^{\infty}\rho_n\right). \tag{2.24}$$

Now considering a chain which is geometrically ergodic from its initial state, if $\mathbb{E}\left[|\mathsf{f}|^{2+\delta}\right]$ is finite for some $\delta > 0$ then it can be shown [58, 101, 228] that (2.24) is also the asymptotic variance for a MCMC estimator calculated using the chain states.

This motivates a definition of the *effective sample size* (ESS) for an MCMC estimator $\hat{\mathsf{f}}_N$ computed using a geometrically ergodic chain as

$$N_{\text{eff}} = \frac{N}{1 + 2\sum_{n=1}^{\infty} \rho_n}. \tag{2.25}$$

The ESS quantifies the number of independent samples that would be required in a Monte Carlo estimator to give an equivalent variance to the MCMC estimator $\hat{\mathsf{f}}_N$ in the asymptotic limit $N \to \infty$. In practice we cannot evaluate the exact autocorrelations and so we can only compute an estimated ESS, $\hat{N}_{\text{eff}}$, from one or more simulated chains with the estimation method needing to be carefully chosen to ensure reasonable values [249]. Although the assumption of geometric ergodicity can often be hard to verify in practice and ESS estimates can give misleading results in chains far from convergence, when used appropriately estimated ESSs can still be a useful heuristic for evaluating and comparing the efficiency of Markov chain estimators and are often available as a standard diagnostic in MCMC software packages [55, 211, 236].

So far we have not discussed how to construct a transition operator giving a chain with the required invariant distribution. As a notational convenience we will consider the transition operator as being specified by a conditional density we term the *transition density* $\vec{\mathsf{t}} : X \times X \to [0, \infty)$ which is defined with respect to a base measure μ (which we assume to be the same as that which the target density we wish to integrate against is defined with respect to, hence the reuse of notation). The transition operator is then

$$\vec{\mathsf{T}}(A \mid \boldsymbol{x}) = \int_A \vec{\mathsf{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x}') \quad \forall A \in \mathcal{F},\ \boldsymbol{x} \in X. \tag{2.26}$$

In practice the probability measure defined by a transition operator will often have a singular component, for example corresponding to a non-zero probability of the chain remaining in the current state. In this case $\vec{\mathsf{T}}$ is not absolutely continuous with respect to μ and a transition density is not strictly well defined. As we did in the previous chapter however we will informally use Dirac deltas to represent a 'density' of

singular measures, and so still consider a transition density as existing. The requirement that the transition operator leaves the target distribution invariant, can then be expressed in terms of the target density p and transition density $\vec{\mathrm{t}}$ as

$$p(\boldsymbol{x}') = \int_X \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x}) \quad \forall \boldsymbol{x}' \in X. \tag{2.27}$$

Finding a transition density which leaves the target density invariant by satisfying (2.27) seems difficult in general as it involves evaluating an integral against the target density - precisely the computational task which we have been forced to seek approximate solutions to. We can make progress by considering the joint density of a pair of successive states for a chain with invariant distribution P that has converged to stationarity. Then we have that

$$\mathrm{p}_{\mathbf{x}_n, \mathbf{x}_{n-1}}(\boldsymbol{x}', \boldsymbol{x}) = \mathrm{p}_{\mathbf{x}_n \mid \mathbf{x}_{n-1}}(\boldsymbol{x}' \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}_{n-1}}(\boldsymbol{x}) = \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}). \tag{2.28}$$

We can also consider factorising this joint density into the product of the marginal density of the current state $\mathrm{p}_{\mathbf{x}_n}$ and the conditional density of the previous state given the current state $\mathrm{p}_{\mathbf{x}_{n-1} \mid \boldsymbol{x}_n}$. Due to stationarity $\mathrm{p}_{\mathbf{x}_n}$ is also equal to p and so we have that $\mathrm{p}_{\mathbf{x}_{n-1} \mid \boldsymbol{x}_n}$ must be the density of a transition operator which also leaves P invariant, corresponding to a time reversed version of the original (stationary) Markov chain[4]. If we therefore denote $\overleftarrow{\mathrm{t}} = \mathrm{p}_{\mathbf{x}_{n-1} \mid \boldsymbol{x}_n}$ (and which we will term the *backward transition density* in contrast to $\vec{\mathrm{t}}$ which in this context we will qualify as the *forward transition density*), we have that

$$\vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) = \overleftarrow{\mathrm{t}}(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}') \quad \forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X. \tag{2.29}$$

Integrating both sides with respect to $\boldsymbol{x}$, we have that $\forall \boldsymbol{x}' \in X$

$$\int_X \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x}) = \int_X \overleftarrow{\mathrm{t}}(\boldsymbol{x} \mid \boldsymbol{x}')\, \mu(\mathrm{d}\boldsymbol{x})\, p(\boldsymbol{x}') = p(\boldsymbol{x}'), \tag{2.30}$$

and so that (2.27) is satisfied, with the last inequality arising due to $\overleftarrow{\mathrm{t}}$ being a normalised density on its first argument. Therefore if we can find a pair of transition densities, $\vec{\mathrm{t}}$ and $\overleftarrow{\mathrm{t}}$, satisfying (2.29), then the transition operator specified by $\vec{\mathrm{t}}$ will leave the target distribution P

4 The time reversal of a Markov chain is always itself a Markov chain irrespective of stationarity (as the defining conditional independence structure is symmetric with respect to the direction of time), however the reverse of a homogeneous Markov chain which is not stationary will not in general itself be homogeneous.

invariant (and by an equivalent argument so will the transition operator specified by $\overleftarrow{\mathrm{t}}$). We can further simplify (2.29) by requiring that $\overrightarrow{\mathrm{t}} = \overleftarrow{\mathrm{t}} = \mathrm{t}$, i.e. that both forward and backward transition densities (and corresponding operators) take the same form and so that the chain at stationarity is *reversible*, in which case have that

$$\mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) = \mathrm{t}(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}') \quad \forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X. \tag{2.31}$$

This is often termed the *detailed balance* condition. Importantly both the detailed balance (2.31) and *generalised balance* (2.29) conditions can also be written in terms of the unnormalised density $\tilde{p}$ by multiplying both sides by Z, and so can be checked even when Z is unknown.

The restriction to reversible transition operators in detailed balance, while sufficient for (2.27) to hold is not necessary. Markov chains which satisfy the generalised balance condition but not detailed balance are termed *non-reversible*, and there are theoretical results suggesting that non-reversible Markov chains can sometimes achieve significantly improved convergence compared to related reversible chains [74, 132, 191]. While there are several general purpose frameworks for specifying reversible transition operators which leave a target distribution invariant, developing methods for constructing irreversible transition operators with a desired invariant distribution has proven more challenging. The approaches proposed to date are generally limited in practice to special cases such as finite state spaces [243, 244, 255] or chains with tractable invariant distributions such as the multivariate normal [38].

Nonetheless non-reversible Markov chains are still commonly used in MCMC applications. Given a set of transition operators which each individually leave a target distribution invariant, the sequential composition of the transition operators will by induction necessarily also leave the target distribution invariant. Even if the individual transition operators are all reversible, the overall sequential composition will generally not be (instead having an adjoint 'backward' operator corresponding to applying the individual transitions in the reversed order). Sequentially combining several reversible transition operators is common in MCMC implementations, though this is more often the result of each individual operator not meaning the requirements for ergodicity in isolation and so needing to be combined with other operators, rather than due to a specific aim of introducing irreversibility.

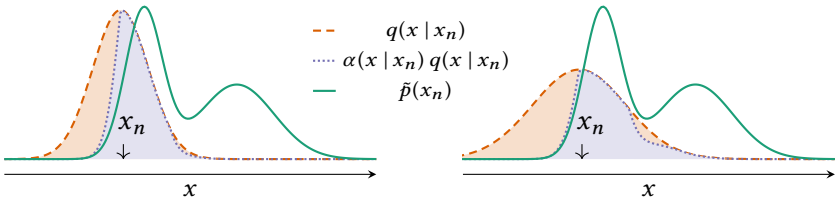


Figure 2.7.: Visualisation of Metropolis–Hastings algorithm in a univariate target distribution. The green curves shows the unnormalised target density. The arrows indicate the current chain state. The orange curves show the density of proposed moves from this state, with the left axis using a narrower proposal than the right. The violet curves show the proposal density scaled by the acceptance probability of the proposed move, this reducing the probability of transitions to states with lower density than the current state. The orange region between the violet and orange curves represents the probability mass reallocated to rejections by the downscaling by the acceptance function. The broader proposal in the right axis has an increased probability of making a move to the other mode in the target density but at a cost of an increased rejection probability.

Having now introduced the key theory underlying MCMC methods, we will now discuss practical implementations of the approach. In the following sub-sections we review two of the most popular frameworks for constructing reversible transition operators which leave a target distribution invariant: the *Metropolis–Hastings* algorithm and *Gibbs sampling*.

2.2.1 Metropolis–Hastings

Although the algorithm has come to be commonly known by Edward Metropolis' name as first author on the 1953 paper [173], it is believed that Arianna and Marshall Rosenbluth, two of the other co-authors, were the main contributors to the development of the algorithm [119].

The seminal *Metropolis–Hastings* algorithm provides a general framework for constructing Markov chains with a desired invariant distribution and is ubiquitous in MCMC methodology. The original Rosenbluth–Teller–Metropolis variant of the algorithm [173] dates to the very beginnings of the Monte Carlo method, having being first implemented on Los Alamos' MANIAC[5] one of the earliest programmable computers. The method was generalised in a key paper by Hastings [127], and the optimality among several competing alternatives of the form now used demonstrated by Peskun [207]. An extension to Markov chains on trans-dimensional spaces was proposed by Green [117].

An outline of the method is given in Algorithm 2 and a visualisation of its application to a univariate target distribution shown in Figure 2.7. The key idea is to propose updates to the state using an arbitrary trans-

5 *Mathematical Analyzer, Numerical Integrator and Computer.*

Algorithm 2 Metropolis–Hastings.

Input: $\boldsymbol{x}_n$: current chain state, $\tilde{p}$: unnormalised target density,
q : normalised proposal density which we can sample according to.
Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p(\cdot) \implies \boldsymbol{x}_{n+1} \sim p(\cdot)$.

$\boldsymbol{x}^* \sim q(\cdot \mid \boldsymbol{x}_n)$ ▷ Generate proposed new state
$u \sim \mathcal{U}(\cdot \mid 0, 1)$
if $u < \frac{\tilde{p}(\boldsymbol{x}^*)\, q(\boldsymbol{x}_n \mid \boldsymbol{x}^*)}{\tilde{p}(\boldsymbol{x})\, q(\boldsymbol{x}^* \mid \boldsymbol{x}_n)}$ **then**
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}^*$ ▷ Proposed move accepted
else
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_n$ ▷ Proposed move rejected

ition operator and then correct for this transition operator not necessarily leaving the target distribution invariant by stochastically accepting or rejecting the proposal. If a proposal is rejected the chain remains at the current state, otherwise the chain state takes on the proposed value. The transition density corresponding to Algorithm 2 is

$$\begin{aligned} \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x}) = \alpha(\boldsymbol{x}' \mid \boldsymbol{x})\, q(\boldsymbol{x}' \mid \boldsymbol{x}) + \\ \left(1 - \int_X \alpha(\boldsymbol{x}^* \mid \boldsymbol{x})\, q(\boldsymbol{x}^* \mid \boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x}^*)\right) \delta(\boldsymbol{x}' - \boldsymbol{x}), \end{aligned} \tag{2.32}$$

with the *acceptance probability* $\alpha : X \times X \to [0, 1]$ defined as

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}\right\} = \min\left\{1, \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, \tilde{p}(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, \tilde{p}(\boldsymbol{x})}\right\}, \tag{2.33}$$

and $q : X \times X \to [0, \infty)$ the *proposal density*.

The original Rosenbluth–Teller–Metropolis algorithm used a symmetric proposal density $q(\boldsymbol{x}' \mid \boldsymbol{x}) = q(\boldsymbol{x} \mid \boldsymbol{x}')\ \forall \boldsymbol{x} \in X, \boldsymbol{x}' \in X$ (with the extension to the non-symmetric case being due to Hastings [127]), in which case the acceptance probability definition simplifies to

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})}\right\} = \min\left\{1, \frac{\tilde{p}(\boldsymbol{x}')}{\tilde{p}(\boldsymbol{x})}\right\}. \tag{2.34}$$

Note that in both (2.33) and (2.34) the target density only appears as a ratio and so only need be known up to a constant.

For the purposes of verifying the detailed balance condition (2.31), the density of *self-transitions*, i.e. a transition to the same state, can be ignored as (2.31) is trivially satisfied for $\boldsymbol{x}' = \boldsymbol{x}$. Considering therefore the cases $\boldsymbol{x} \neq \boldsymbol{x}'$ where the Dirac delta term representing the singular

component corresponding to rejected proposals can be neglected, we have $\forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X : \boldsymbol{x} \neq \boldsymbol{x}$

$$\begin{aligned}
\mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) &= \min\left\{1,\, \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}\right\} q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) && (2.35)\\
&= \min\{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}),\, q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')\} && (2.36)\\
&= \min\left\{\frac{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')},\, 1\right\} q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}') && (2.37)\\
&= \mathrm{t}(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}'). && (2.38)
\end{aligned}$$

Therefore the detailed balance condition is satisfied, and the Metropolis–Hastings transition operator leaves the target distribution P invariant.

A special case for chains on a Euclidean state space $X = \mathbb{R}^D$, is when the proposal transition operator is deterministic and corresponds to a differentiable involution of the current state. Let $\boldsymbol{\phi} : X \to X$ be an involution, i.e. $\boldsymbol{\phi} \circ \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}\ \forall X$ with Jacobian determinant $D_{\boldsymbol{\phi}}(\boldsymbol{x}) = \left|\frac{\partial \boldsymbol{\phi}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|$ which is defined and non-zero P-almost everywhere. Then if we define a transition operator via the transition density

$$\begin{aligned}
\mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x}) &= \delta(\boldsymbol{x}' - \boldsymbol{\phi}(\boldsymbol{x}))\alpha(\boldsymbol{x}) + \delta(\boldsymbol{x}' - \boldsymbol{x})(1 - \alpha(\boldsymbol{x})),\\
\alpha(\boldsymbol{x}) &= \min\left\{1,\, \frac{p \circ \boldsymbol{\phi}(\boldsymbol{x})}{p(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x})\right\},
\end{aligned} \tag{2.39}$$

then this transition operator will leave the target distribution P invariant. This deterministic transition operator variant is as a special case of the trans-dimensional Metropolis–Hastings extension introduced by Green [102, 117]. To generate from this transition operator from a current state $\boldsymbol{x}$ we compute the proposed move $\boldsymbol{\phi}(\boldsymbol{x})$ and accept the move with probability $\alpha(\boldsymbol{x})$. We can demonstrate that this transition operator leaves P invariant by directly verifying (2.27)

$$\begin{aligned}
&\int_X \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} && (2.40)\\
&= \int_X \delta(\boldsymbol{x}' - \boldsymbol{\phi}(\boldsymbol{x}))\, \alpha(\boldsymbol{x})\, p(\boldsymbol{x}) + \delta(\boldsymbol{x}' - \boldsymbol{x})(1 - \alpha(\boldsymbol{x}))\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} && (2.41)\\
&= \int_X \delta(\boldsymbol{x}' - \boldsymbol{y})\, \alpha \circ \boldsymbol{\phi}(\boldsymbol{y})\, p \circ \boldsymbol{\phi}(\boldsymbol{y})\, D_{\boldsymbol{\phi}}(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y} + (1 - \alpha(\boldsymbol{x}'))\, p(\boldsymbol{x}') && (2.42)\\
&= p(\boldsymbol{x}') + \alpha \circ \boldsymbol{\phi}(\boldsymbol{x}')\, p \circ \boldsymbol{\phi}(\boldsymbol{x}')\, D_{\boldsymbol{\phi}}(\boldsymbol{x}') - \alpha(\boldsymbol{x}')\, p(\boldsymbol{x}'). && (2.43)
\end{aligned}$$

In going from (2.42) to (2.43) we use a change of variables $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$ in the integral. As $\boldsymbol{\phi}$ is an involution we have that $\boldsymbol{\phi} \circ \boldsymbol{\phi}(\boldsymbol{x}') = \boldsymbol{x}'$ and $D_{\boldsymbol{\phi}} \circ \boldsymbol{\phi}(\boldsymbol{x}') = D_{\boldsymbol{\phi}}(\boldsymbol{x}')^{-1}$ and so

$$\alpha \circ \boldsymbol{\phi}(\boldsymbol{x}')p \circ \boldsymbol{\phi}(\boldsymbol{x}')D_{\boldsymbol{\phi}}(\boldsymbol{x}') = \min\left\{p \circ \boldsymbol{\phi}(\boldsymbol{x}')D_{\boldsymbol{\phi}}(\boldsymbol{x}'), p(\boldsymbol{x}')\right\} = \alpha(\boldsymbol{x}')p(\boldsymbol{x}').$$

The last two terms in (2.43) therefore cancel and so (2.27) is satisfied by the transition operator defined by (2.39).

Although this transition operator leaves the target distribution P invariant, it is clear that it will not generate an ergodic Markov chain. Starting from a point $\boldsymbol{x}$ the next chain state will be either $\boldsymbol{\phi}(\boldsymbol{x})$ if the proposed move is accepted or $\boldsymbol{x}$ if rejected. In the former case the next proposed move will be to $\boldsymbol{\phi} \circ \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$ i.e. back to the original state. Therefore the chain will visit a maximum of two states. However as noted previously we can sequentially compose individual transition operators which all leave a target distribution invariant. Therefore a deterministic proposal Metropolis–Hastings transition can be combined with other transition operators to ensure the chain is irreducible and aperiodic.

In general for a Metropolis–Hastings transition operator to be irreducible, it is necessary that the proposal operator is irreducible [250], however this is not sufficient. For a target density which is positive everywhere on $X = \mathbb{R}^D$, then a sufficient but not necessary condition for irreducibility is that the proposal density is positive everywhere [228]. If the set of points with a non-zero probability of rejection has non-zero P-measure, then the transition operator is aperiodic [250].

A common choice of proposal density when the target distribution is defined on $\mathbb{R}^D$ is a multivariate normal density centred at the current state i.e. $q(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}' \mid \boldsymbol{x}, \Sigma)$ which satisfies the positivity condition for irreducibility. In general we would achieve optimal performance with a proposal density covariance Σ which is proportional to the covariance of the target distribution [231]. In practice we do not have access to the true covariance and so typically an isotropic proposal density is used with covariance $\Sigma = \sigma^2\mathbf{I}$ controlled by a single scale parameter σ, often termed the *step size* or *proposal width*. This proposal density is symmetric so the simplified acceptance rule (2.34) can be used, further the proposal density depends only on the difference $\boldsymbol{x}' - \boldsymbol{x}$ with Metropolis–Hastings methods having these properties often termed *random-walk Metropolis*.

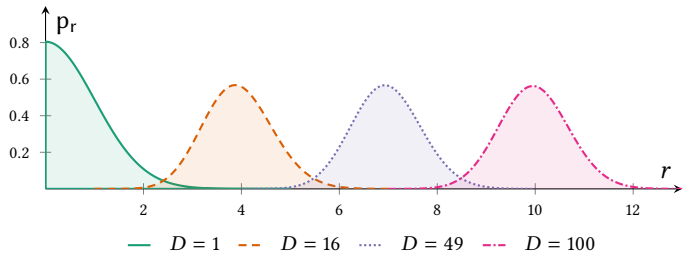


Figure 2.8.: Illustration of concentration of measure in a multivariate normal distribution. The plots shows the probability density of the distance from the origin $\mathrm{r} = \|\mathbf{x}\|_2$ of a D-dimensional multivariate normal random vector $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for different dimensionalities D. As the dimension increases most of the mass concentrates away from the origin around a spherical shell of radius $\sqrt{D}$. For a multivariate normal random vector with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ this generalises to the mass being mainly in an ellipsoidal shell aligned with the eigenvectors of $\boldsymbol{\Sigma}$ and centred at $\boldsymbol{\mu}$.

Random walk Metropolis methods have been extensively theoretically studied, with sufficient conditions known in some cases to ensure geometric ergodicity of a chain [172, 230] though these can be hard to verify in practical problems. There has also been much work on practical guidelines and methods for tuning the free parameters in the algorithm, including approaches for tuning the step-size using acceptance rates [94, 227] and adaptive variants which automatically estimate a non-isotropic proposal covariance [121, 231].

In general the choice of proposal density will be key in determining the efficiency of Metropolis–Hastings MCMC methods. Ideally we want to be able to propose large moves in the state space to reduce the dependencies between successive chain states and so increase the number of effective samples, however this needs to be balanced with maintaining a reasonable acceptance probability with large proposed moves often having a low acceptance probability. Figure 2.7 gives an illustration of this trade-off in a one-dimensional example.

In high-dimensional spaces this issue is much more severe due to the phenomenon of *concentration of measure*: in probability distributions defined on high-dimensional spaces most of the probability mass will tend to be concentrated into a 'small' subset of the space [13, 161]. An illustration of this phenomenon for the multivariate normal distribution is shown in Figure 2.8, where the mass in high dimensions is mostly located in a thin ellipsoidal shell. The region where most of the mass concentrates, termed the *typical set* of the distribution, will for the tar-

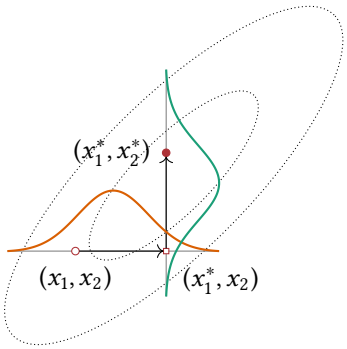


Figure 2.9.: Schematic of Gibbs sampling transition in a bivariate normal target distribution (ellipses indicate constant density contours). Given an initial state $\mathbf{x} = (x_1, x_2)$, the x_1 (horizontal) co-ordinate is first updated by independently sampling from the normal conditional $p_{x_1|x_2}(\cdot \mid x_2)$, represented by the orange curve. The new partially updated state is then $\mathbf{x} = (x_1^*, x_2)$. The second x_2 (vertical) co-ordinate is then independently resampled from the normal conditional $p_{x_2|x_1}(\cdot \mid x_1^*)$, shown by the green curve. The final updated state is then $\mathbf{x} = (x_1^*, x_2^*)$.

Algorithm 3 Sequential-scan Gibbs.

Input: $\boldsymbol{x}_n$: current chain state, I : ordered set of indices of all individual variables in chain state, $\{p_i\}_{i \in I}$: set of complete conditionals of target density p which can all be sampled from.

Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p(\cdot) \implies \boldsymbol{x}_{n+1} \sim p(\cdot)$.

1: $\boldsymbol{x} \leftarrow \boldsymbol{x}_n$
2: **for** $i \in I$ **do**
3: $\quad x_i \sim p_i(\cdot \mid \boldsymbol{x}_{\setminus i})$ ▷ Resample x_i from p_i given *current* $\boldsymbol{x}_{\setminus i}$.
4: $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}$

get distributions of interest generally have a significantly more complex geometry. Finding proposals which can make large moves in such settings is challenging: moves in most directions will have a probability of acceptance which exponentially drops to zero as the distance away from the current state is increased and so simple proposal densities which ignore the geometry the typical set such as those used in random-walk Metropolis will need to make very small moves to have a reasonable probability of acceptance [33].

2.2.2 Gibbs sampling

Gibbs sampling [93, 100], originally proposed by Geman and Geman for image restoration using a Markov random field image model, is based on the observation that a valid transition operator for a joint target distribution across many variables, is one which updates only a subset of the variables and leaves the conditional distribution on that subset

given the rest invariant. Although if used in isolation a transition operator which only updates some components of the state will not give an ergodic chain, as discussed previously multiple transition operators can be combined together to achieve ergodicity.

More specifically the original formulation of Gibbs sampling defines a Markov chain by sequentially independently resampling each individual variable in the model from its conditional distribution given the current values of the remaining variables. If I is an index set over the individual variables in the vector target state $\mathbf{x}$, then for each $i \in I$ we partition the state $\mathbf{x}$ into the i^{th} variable x_i and a vector containing all the remaining variables $\mathbf{x}_{\setminus i}$. For each $i \in I$ the target density can be factorised in to the marginal density $p_{\setminus i}$ on $\mathbf{x}_{\setminus i}$ and conditional density p_i on x_i given $\mathbf{x}_{\setminus i}$, i.e.

$$p(\boldsymbol{x}) = p_i(x_i \mid \boldsymbol{x}_{\setminus i})\, p_{\setminus i}(\boldsymbol{x}_{\setminus i}), \tag{2.44}$$

with the conditional densities $\{p_i\}_{i \in I}$ termed the *complete conditionals* of the target density. If each of these complete conditionals corresponds to a distribution we can generate samples from (for example using a transform method or rejection sampling) then we can apply the sequential Gibbs sampling transition operator defined in Algorithm 3 and visualised for a bivariate example in Figure 2.9.

The sequential Gibbs transition is irreducible and aperiodic under mild conditions [57, 229]. Rather than using a deterministic sequential scan through the variables, an alternative is to randomly sample without replacement the variable to update on each iteration; unlike the sequential scan version this defines a reversible transition operator. The random update variant is more amenable to theoretical analysis, however in practice the ease of implementation of the sequential scan variant and computational benefits in terms of memory access locality mean it seems to be more often used in practice [128]. A compromise between the completely random updates and a sequential scan is to randomly permute the update order after each complete scan.

A apparent advantage of Gibbs sampling over Metropolis–Hastings is the lack of a proposal density which needs to be tuned. This has helped popularise 'black-box' implementations of Gibbs sampling such as the probabilistic modelling packages BUGS [105] and JAGS [210]. A well-known issue with Gibbs sampling however is that its performance is

highly dependent on the parameterisation used for the target density [217], with strong correlations between variables leading to large dependencies between successive states and slow convergence to stationarity. This can be alleviated in some cases by using a suitable reparameterisation to reduce dependencies between variables, however this restores the difficulty of tuning free parameters.

Gibbs sampling updates do not necessarily need to be performed by sampling from complete conditionals of single variables - in some cases the complete conditional of a vector of variables has a tractable form which can be sampled from as a 'block'; this motivates the name *block Gibbs sampling* for such variants. By accounting for the dependencies between the variables in a block this can help alleviate some of the issues with highly correlated targets where applicable.

Compound terms such as *Metropolis-within-Gibbs* are sometimes used to refer to methods which sequentially apply Metropolis transition operators which each update only a subset of variables in the target distribution. We will however consider the defining feature of Gibbs sampling as being exact sampling from one or more conditionals rather than sequentially applying transition operators which update only subsets of variables and so will only refer to 'Gibbs sampling' in that context.

2.3 AUXILIARY VARIABLE METHODS

Although Gibbs sampling and random-walk Metropolis are commonly used in practice, as discussed above both have drawbacks when applied to complex high-dimensional target distributions. One approach which has proven particularly successful for constructing alternative Markov transition operators which can overcome some of these shortcomings is the introduction of *auxiliary variables* in to the chain state. For concreteness of notation in the following discussion we let the variables of interest, which we term the target variables, be represented by the random vector $\mathbf{x} \in X$ and the introduced auxiliary variables by the random vector $\mathbf{a} \in A$. We assume for generality here multiple auxiliary variables are introduced, however methods using a single scalar auxiliary variable are a common special case.

One way of defining a joint distribution across the target and auxiliary variables is to specify an arbitrary conditional distribution $\mathrm{P}_{\mathbf{a}|\mathbf{x}}$

and choose the marginal distribution $\mathsf{P}_{\mathbf{x}}$ to be equal to the target distribution P. Given samples from a joint distribution we can estimate expectations with respect to the marginal distribution on a subset of the variables by simply ignoring the dimensions of the sampled state we wish to marginalise over. Therefore if we can construct a Markov chain with the resulting joint distribution $\mathsf{P}_{\mathbf{x},\mathbf{a}}$ as its unique invariant distribution, then we can use the target variable components of the sampled states to estimate expectations with respect to the target distribution. We will consider two MCMC methods using this approach, *slice sampling* and *Hamiltonian Monte Carlo* in Sections 2.3.1 and 2.3.2.

An alternative approach is to instead construct a joint distribution on the target and auxiliary variables such that the regular conditional distribution $\mathsf{P}_{\mathbf{x}|\mathbf{a}}$ is equal to the target distribution P across some set of values $A^* \subset A$ of the auxiliary variables. In terms of the density $\mathsf{p}_{\mathbf{x}|\mathbf{a}}$ this requires that

$$\mathsf{p}_{\mathbf{x}|\mathbf{a}}(\boldsymbol{x} \mid \boldsymbol{a}) = p(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in X, \boldsymbol{a} \in A^*. \tag{2.45}$$

If the marginal probability $\mathsf{P}_{\mathbf{a}}(A^*)$ is non-zero, then the sampled states $\{\boldsymbol{x}^{(n)}, \boldsymbol{a}^{(n)}\}_{n=1}^N$ of a Markov chain which has $\mathsf{P}_{\mathbf{x},\mathbf{a}}$ as its unique invariant distribution can be used to estimate expectations with respect to the target distribution by computing averages over only the sampled target variable values $\boldsymbol{x}^{(n)}$ for which the corresponding auxiliary variables $\boldsymbol{a}^{(n)}$ take values in A^* (these being at convergence samples from $\mathsf{P}_{\boldsymbol{x}\mid\boldsymbol{a}}(\cdot \mid \boldsymbol{a})$ and so the target distribution), i.e.

$$\int_X f(\boldsymbol{x})\, P(\mathrm{d}\boldsymbol{x}) = \lim_{N\to\infty} \frac{\sum_{n=1}^N \mathbb{1}_{A^*}(\boldsymbol{a}^{(n)}) f(\boldsymbol{x}^{(n)})}{\sum_{n=1}^N \mathbb{1}_{A^*}(\boldsymbol{a}^{(n)})}. \tag{2.46}$$

We will discuss *simulated tempering*, an MCMC method which introduces an auxiliary variable in this manner in Section 2.3.3.

An issue with this approach is that if $\mathsf{P}_{\mathbf{a}}(A^*)$ is small, the number of sampled states with $\boldsymbol{a} \in A^*$ may be very small or even zero. This can require a large number of samples N for the sufficient samples with auxiliary variables in the required set A^* to be generated to allow the MCMC estimates computed using (2.46) to be reliable.

The estimator in (2.46) has a close resemblance to the formulation of the Monte Carlo estimator corresponding to rejection sampling given in (2.16), with averages computed over the subset of samples meeting

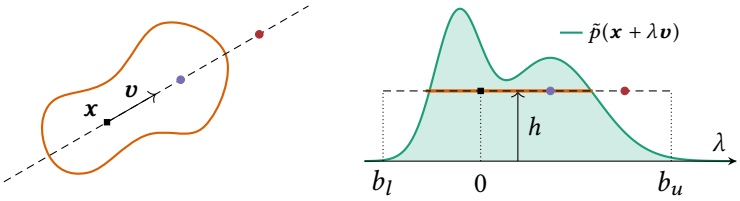


Figure 2.10.: Schematic of linear slice sampling, showing 'plan' (left) and 'cross-sectional' (right) views of a bivariate target density. Orange curve (left) and line (right) indicates a constant density slice S_h. The black square indicates current target state value $\boldsymbol{x}$ and the dashed line is *slice line*, the one-dimensional linear sub-space aligned with the vector $\boldsymbol{v}$ which a new value from the state will be sampled on. The extents of the dashed line segment represent the initial bracket new proposed states will be drawn from. Points are proposed on the slice line by drawing a value uniformly from the current bracket. The red circle represents an initial proposed point which is not in the slice and so the right bracket edge is shrunk to this point. The violet circle shows a second sampled point from the new reduced bracket, this point within the slice and so returned as the updated target state.

an 'acceptance' criteria. As noted previously rejection sampling can in fact be considered as an auxiliary variable method, with binary accept indicator variables $a \in \{0, 1\}$ introduced such that the conditional density $\mathrm{p}_{\mathbf{x}|a}(\boldsymbol{x} \mid 1)$ is equal to the target density $p(\boldsymbol{x})$ as shown in (2.12), i.e. exactly corresponding to the property in (2.45). Rejection sampling can therefore be seen to be an example of this construct, though in this case each pair of target – auxiliary variable samples are generated independently rather than by constructing a Markov chain.

When discussing the rejection sampling estimator (2.16) we saw there was a close link to importance sampling estimator (2.17), with the importance sampling estimator having the potential advantage however of using all of the generated samples in computing estimates. In Chapter 5 we will discuss a related alternative approach to constructing estimators for auxiliary variable methods based on conditioning like simulated tempering, which unlike the estimator in (2.46) allows using all of the samples in a Markov chain to compute estimates.

2.3.1 Slice sampling

Slice sampling is a family of auxiliary variable MCMC methods which exploit the same observation as used to motivate rejection sampling - to sample from a target distribution it is sufficient to uniformly sample from the volume beneath a graph of the target density function. Rather

Algorithm 4 Linear slice sampling.

Input: $\boldsymbol{x}_n$: current chain state, $\tilde{p}$: unnormalised target density,
q : slice vector density, M : maximum number of step out iterations.
Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p(\cdot) \implies \boldsymbol{x}_{n+1} \sim p(\cdot)$.

$h \sim \mathcal{U}(\cdot \mid 0, \tilde{p}(\boldsymbol{x}_n))$ ▷ Sample slice height
$\boldsymbol{v} \sim q(\cdot)$ ▷ Sample vector setting slice line and initial bracket width
$b_u \sim \mathcal{U}(\cdot \mid 0, 1)$ ▷ Uniformly sample bracket around current state
$b_l \leftarrow b_u - 1$
if $M > 0$ **then** $(b_l,\, b_u) \leftarrow$ LinearStepOut$(\boldsymbol{x}_n,\, b_l,\, b_u,\, M)$
$\lambda \sim \mathcal{U}(\cdot \mid b_l, b_u)$
while True **do**
    $\boldsymbol{x}^* \leftarrow \boldsymbol{x}_n + \lambda \boldsymbol{v}$ ▷ Update proposed state
    if $\tilde{p}(\boldsymbol{x}^*) \leq h$ **then** ▷ Proposed point not on slice
        if $\lambda < 0$ **then** $b_l \leftarrow \lambda$ **else** $b_u \leftarrow \lambda$ ▷ Shrink slice bracket
        $\lambda \sim \mathcal{U}(\cdot \mid b_l, b_u)$ ▷ Sample uniformly from new bracket
    else ▷ Proposed state on slice
        return $\boldsymbol{x}^*$
function LinearStepOut$(\boldsymbol{x}_n,\, b_l,\, b_u,\, M)$
    $L \sim \text{UniformInt}(\cdot \mid 0, M)$ ▷ Sample integer uniformly from $[0, M]$
    $U \leftarrow M - L$
    while $L > 0$ **and** $\tilde{p}(\boldsymbol{x}_n + b_l \boldsymbol{v}) > h$ **do** ▷ Step out lower bracket edge
        $b_l \leftarrow b_l - 1$
        $L \leftarrow L - 1$
    while $U > 0$ **and** $\tilde{p}(\boldsymbol{x}_n + b_u \boldsymbol{v}) > h$ **do** ▷ Step out upper bracket edge
        $b_u \leftarrow b_u + 1$
        $U \leftarrow U - 1$
    return b_l, b_u

than generate independent points from this volume as in rejection sampling, slice sampling instead constructs a transition operator which leaves the uniform distribution on this volume invariant.

The method we will concentrate on here was proposed by Neal [188, 190]. A related algorithm which uses per data-point auxiliary variables in Bayesian inference problems was developed by Damien, Wakefield and Walker [69]. Murray, Adams and Mackay later proposed *elliptical slice sampling* [183], an extension of Neal's slice sampling method which is particularly effective for target distributions which are well approximated by a multivariate normal distribution.

Slice sampling defines a Markov chain on an augmented state space by introducing an auxiliary *height* variable $\mathsf{h} \in [0, \infty)$ in addition to the target variables $\mathbf{x} \in X$. The conditional density on h is

$$\mathrm{p}_{\mathsf{h}|\mathbf{x}}(h \mid \mathbf{x}) = \mathcal{U}(h \mid 0, \tilde{p}(\boldsymbol{x})) = \frac{1}{\tilde{p}(\boldsymbol{x})} \mathbb{1}_{[0, \tilde{p}(\boldsymbol{x}))}(h), \tag{2.47}$$

i.e. uniform over the interval between zero and the unnormalised target density value. The joint density on the augmented space is then

$$\mathsf{p}_{\mathbf{x},\mathsf{h}}(\boldsymbol{x}, h) = \frac{1}{\tilde{p}(\boldsymbol{x})} \mathbb{1}_{[0,\tilde{p}(\boldsymbol{x}))}(h) \frac{\tilde{p}(\boldsymbol{x})}{Z} = \frac{1}{Z} \mathbb{1}_{[0,\tilde{p}(\boldsymbol{x}))}(h). \tag{2.48}$$

Marginalising (2.48) over h recovers the target density i.e. $\mathsf{p}_{\mathbf{x}} = p$.

The overall slice sampling transition is formed of the sequential composition of a transition operator which updates h given $\mathbf{x}$ and a second operator which updates $\mathbf{x}$ given h, each leaving the distributions corresponding to the conditional densities $\mathsf{p}_{\mathsf{h}|\mathbf{x}}$ and $\mathsf{p}_{\mathbf{x}|\mathsf{h}}$ respectively invariant, and so by the same argument as for Gibbs sampling the overall transition leaving the target distribution invariant. By construction the conditional density $\mathsf{p}_{\mathsf{h}|\mathbf{x}}$ is a simple uniform density and so the first transition operator is a Gibbs sampling update in which the height variable is independently resampled from $\mathcal{U}(0, \tilde{p}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the current value of the target state $\mathbf{x}$.

The conditional density $\mathsf{p}_{\mathbf{x}|\mathsf{h}}(\boldsymbol{x} \mid h)$ is also locally uniform, equal to a positive constant whenever $\tilde{p}(\boldsymbol{x}) > h$ and zero elsewhere. However we can usually only evaluate the density up to an unknown constant as we cannot compute the measure of the set $S_h = \{\boldsymbol{x} \in X : \tilde{p}(\boldsymbol{x}) > h\}$ that the density is non-zero over. In general S_h, which is the eponymous *slice* of slice sampling (so called as it represents a slice through the volume under the density curve at a fixed height h), will have a complex geometry including potentially consisting of several disconnected components in the case of multimodal densities. The complexity of the slices generally prevents us therefore from being able to independently sample a new value for $\mathbf{x}$ uniformly from S_h and so we cannot use a full Gibbs sampling scheme corresponding to sequentially independently sampling from $\mathsf{P}_{\mathsf{h}|\mathbf{x}}$ and $\mathsf{P}_{\mathbf{x}|\mathsf{h}}$.

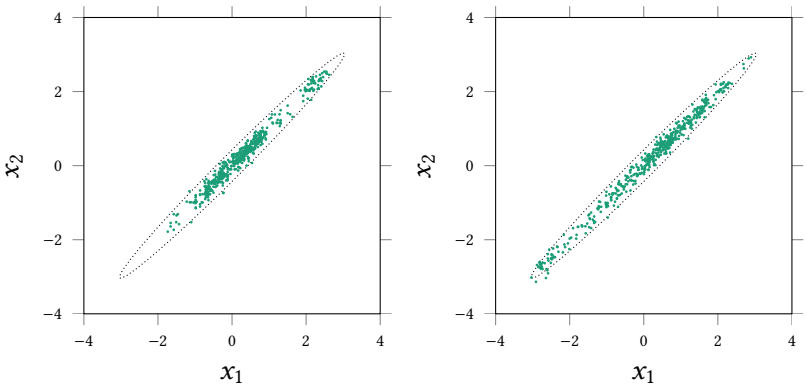
A key contribution of [190] was to introduce an elegant method for constructing a transition operator which leaves $\mathsf{P}_{\mathbf{x}|\mathsf{h}}$ invariant. In particular the algorithm has few free parameters to tune, has an efficiency which is relatively robust to the choices of the free choices that are introduced, and will for smooth target densities always move the target state by some amount (in contrast to the potential for rejections in Metropolis–Hastings methods). This method is summarised in Algorithm 4 and a visualisation of the process shown in Figure 2.10.

An important first step in the algorithm is reducing the problem of generating a point uniformly on the multidimensional slice S_h to making a move on a one-dimensional linear subspace of this slice (motivating our naming of this algorithm *linear slice sampling*) which includes the current $\mathbf{x}$ state. In the original description of the algorithm in [190] the one-dimensional subspace is chosen to be axis-aligned, corresponding to updating a single component of the target state.

In the case of an axis-aligned subspace the restriction of the slice to the one-dimensional subspace is entirely specified by the conditional density on the chosen variable component given the current values of the remaining components in the state. Slice sampling transitions for each variable in the target state can then be applied sequentially akin to Gibbs sampling, but with the advantage over Gibbs of not requiring the complete conditionals to be of a tractable form which we can generate exact samples from. If conditional independency structure in the target density means the complete conditionals depend only on local subsets of variables in the target state using updates of this form has the advantage of exploiting this locality. As with Gibbs sampling however applying slice sampling in this manner makes performance strongly dependent on the parameterisation of the target density, with large magnitude correlations likely to lead to slow exploration of the space.

In [190] various multivariate extensions of the algorithm are suggested which could help counter this issue, however they add significant implementation complexity compared to the basic algorithm. A simpler alternative is to define the one-dimensional subspace as being the line defined by a randomly chosen vector and passing through the current value of $\mathbf{x}$. If this vector is generated independently of the current state this is sufficient to ensure the overall transition retains the correct invariant distribution.

If little is known about the target distribution a reasonable default is to sample a unit vector of the required dimensionality by generating a random zero-mean isotropic covariance multivariate normal vector and then scaling it to unit norm; if an approximate covariance matrix $\hat{\Sigma}$ is known for the target density then instead generating the vector from $\mathcal{N}(\mathbf{0}, \Sigma)$ prior to normalising might be a better choice (as it favours moves aligned with the principle eigenvectors of Σ) however in this case elliptical slice sampling, which we will discuss shortly, will often be a better choice.



(a) Axis-aligned updates (b) Random direction updates

Figure 2.11.: Samples generated using (a) axis-aligned versus (b) random-direction linear slice sampling in a correlated bivariate normal distribution. In both cases 1000 transitions where performed (with random selection of axis to update on each iteration in (a)) with every second sampled state shown. the maximum number of step out iterations is $M = 4$ and the initial bracket width is fixed at $w = 1$. The dotted ellipse shows the contour of the target density which contains 0.99 of the mass. The random direction chain is able to explore the typical set of the target distribution more effectively in this case with the axis-aligned updates leading to slower diffusion along the major axis of the elliptical contour.

This random-direction slice sampling variant is discussed in comparison to elliptical slice sampling in [183]. It is also bears resemblance to the scheme proposed in [59] which uses the same auxiliary variable formulation as slice sampling, but there the random direction is chosen in $X \times [0, \infty)$ i.e. to update both $\mathbf{x}$ and h and not used with the remainder of Neal's slice sampling algorithm. An example comparison of applying axis-aligned and random-direction linear slice sampling updates to a strongly positively correlated bivariate normal target distribution is shown in Figure 2.11. In this toy example the isotropic random-direction updates are able to more effectively explore the target density.

The generation of the vector $\boldsymbol{v}$ determining the one-dimensional subspace of the slice the update is performed on is represented in Algorithm 4 by Line 2 by $\boldsymbol{v}$ being generated from a distribution with density q. As well as specifying the *direction* of the slice line, the vector $\boldsymbol{v}$ also specifies a scale along this line. In Neal's description of the algorithm this is represented by the explicit *bracket width* parameter w. Here instead we assume this parameter is implicitly defined by the Euclidean norm of the vector $\boldsymbol{v}$, through suitable choice of q this allowing for direction dependent scales and also the possibility of randomisation of the

scale; as we will see shortly however compared to for example random-walk Metropolis updates with a normal proposal, linear slice sampling is much less sensitive to the choice of scale parameters, therefore a single fixed scale will often be sufficient.

Once the slice line direction and scale has been chosen, the remainder of the algorithm can be split into two stages: selection of an initial bracket on the slice line and including the point corresponding to the current state; iteratively uniformly sampling points within the current bracket, accepting the point if it is within the slice S_h otherwise shrinking the bracket and repeating. The algorithm proposed by Neal ensures both these stages are performed reversibly such that the detailed balance condition (2.31) is maintained.

The *slice bracket* defines a contiguous interval $\lambda \in [b_l, b_u]$ on the slice line $\boldsymbol{x}^*(\lambda) = \boldsymbol{x}_n + \lambda \boldsymbol{v}$ and always includes the point $\lambda = 0$ corresponding to the current state. The initial bracket is chosen by sampling a upper bound b_u uniformly from $[0, 1]$ and then setting $b_l \leftarrow b_u - 1$; in the λ slice line coordinate system this corresponds to a bracket width of one, however in general the slice line vector $\boldsymbol{v}$ can have non-unit length and so defines the initial bracket width in the target variable space. Randomising the positioning of the current state within the bracket ensures reversibility as the resulting bracket would have an equal probability (density) of being selected from any other point in the bracket (which the final accepted point will be within).

In general only a subset of the points in the current slice bracket will be within the slice S_h. As new states are proposed by sampling a point uniformly from the current bracket, the probability of such a proposal being in the slice will be equal to the proportion of the bracket that intersects with the slice S_h. In general therefore it is desirable for the bracket to include as much of the slice as possible while not making the proportion of the bracket intersecting with the slice too small such that many points need to be proposed before a point on the slice is found. The magnitude of $\boldsymbol{v}$ determines the initial bracket extents and so should ideally chosen based on any knowledge of the 'typical scale' of the target density. Often we will have little prior knowledge about such scaling however and the scale will often vary significantly across the target space, and so we may choose an initial bracket which includes only a small proportion of the slice.

The stepping out routine proposed by [190] and detailed in Lines 14 to 23 in Algorithm 4 is designed to counter this issue. The initial slice bracket $[b_l, b_u]$ is iteratively 'stepped-out' by incrementing / decrementing the upper / lower bracket bounds until the corresponding endpoint of the bracket lies outside the slice or a pre-determined maximum number of steps out have been performed. Ideally the step out routine will return a bracket which contains all of the intersection of the slice with the slice line while not also including too great a proportion of off slice points; in general the slice may be non-convex or consist of multiple disconnected components and so the intersection of the slice line with the slice may consist of multiple disconnected intervals in which case the stepping out routine will likely only expand the slice to include a subset of these intervals. The adaptivity provided by the stepping out routine will still however generally help to make the performance of the sampler much less sensitive to the choice of the bracket scale in contrast to for example random-walk Metropolis algorithms.

Analogously to the randomisation of the initial bracket positioning, in the stepping out routine if a maximum number of step out iterations M is set, the resulting step 'budget' is randomly allocated between increments of the upper bound b_u and decrements of the lower bound b_l such that final extended bracket generated by the step out routine would have an equal probability of being generated from any point within the generated bracket interval. If M is set to zero this corresponds to not performing any stepping out and simply using the initial sampled bracket; although reducing the robustness of the algorithm to the choice of the initial bracket width this option has the advantage of minimising the number of target density evaluations by not requiring additional density evaluations at the bracket endpoints during the step-out routine. An alternative 'doubling' step-out routine was also proposed in [190]. This has the advantage of exponentially expanding the slice bracket compared to the linear growth of the step-out routine described in Algorithm 4 and so can be more efficient in target distributions where the typical scales of the density varies across several orders of magnitude. The doubling procedure requires a more complex subsequent procedure for sampling points in the resulting bracket however to ensure reversibility.

Once the initial bracket has been generated and potentially stepped out, the remainder of the algorithm consists of finding a point on the slice

Algorithm 5 Elliptical slice sampling.

Input: $\boldsymbol{x}_n$: current chain state, $\tilde{p}$: unnormalised target density,
$\boldsymbol{\mu}, \boldsymbol{\Sigma}$: mean and covariance of normal approximation to target.
Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p \implies \boldsymbol{x}_{n+1} \sim p$.

$h \sim \mathcal{U}(\cdot \mid 0, \tilde{p}(\boldsymbol{x}_n)/\mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}))$ ▷ Sample slice height
$\boldsymbol{v} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ ▷ Sample vector setting slice ellipse
$\theta_u \sim \mathcal{U}(\cdot \mid 0, 2\pi)$ ▷ Uniformly sample bracket around current state
$\theta_l \leftarrow \theta_u - 2\pi$
$\theta \leftarrow \theta_u$
while TRUE **do**
    $\boldsymbol{x}^* \leftarrow (\boldsymbol{x}_n - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{v} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu}$ ▷ Update proposed state
    if $\tilde{p}(\boldsymbol{x}^*)/\mathcal{N}(\boldsymbol{x}^* \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \leq h$ **then** ▷ Proposed point not on slice
        if $\theta < 0$ **then** $\theta_l \leftarrow \theta$ **else** $\theta_u \leftarrow \theta$ ▷ Shrink slice bracket
        $\theta \sim \mathcal{U}(\cdot \mid \theta_l, \theta_u)$ ▷ Sample uniformly from new bracket
    else
        return $\boldsymbol{x}^*$

line bracket which is within the slice S_h. This is done in an iterative manner by first sampling a point uniformly from the current bracket and checking if it is in the slice or not. If the proposed point is in the slice, the corresponding value for the target variables is returned at the new state. Otherwise the proposed point is set as the new upper or lower bound of the bracket such that the point corresponding to the current state remains in the bracket. This shrinks the bracket by removing an interval where it is known at least some points are not in the slice. A new point is then sampled uniformly from the smaller bracket and the procedure repeats until a point in the slice is found.

The iterative shrinking of the slice bracket implemented by this procedure introduces a further level of adaptivity in to the slice sampling algorithm, meaning that even if only a small proportion of the initial bracket lies within the slice only relatively few iterations will be needed still till the bracket is shrunk sufficiently for there to be a high probability of proposing a point within the bracket. By ensuring the point corresponding to the current state always remains within the current bracket, reversibility is maintained.

An alternative to the linear slice sampling procedure just described, is the *elliptical slice sampling* method proposed in [183] and described in Algorithm 5. As suggested by the name, in elliptical slice sampling rather than proposing points on a line instead an elliptical path in the target space is defined and new points proposed on this ellipse.

Elliptical slice sampling is intended for use in target distributions which can be approximated by a known normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. This distribution might correspond to a normal prior distribution on model latent variables where the dependence between the latent and observed variables is only weak and so the posterior remains well approximated by the prior or a normal approximation fitted directly to the target distribution using an optimisation based approximate inference scheme such as those discussed in Appendix C [194].

In each elliptical slice sampling transition an auxiliary vector $\boldsymbol{v}$ is independently sampled from the normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. If the target distribution was exactly described by the normal distribution we could use this independent draw directly as the new chain state (though obviously in this case there would be no advantage in formulating as an MCMC method). In reality the target distribution will only approximately described by $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and so we wish to instead use this independent draw to define a Markov transition operator that will potentially move the state to a point nearly independent of the current state, but is also able to back off to more conservative proposals closer to the current chain state. This is achieved by defining an elliptical path in target space centred at $\boldsymbol{\mu}$, passing through the current state $\boldsymbol{x}_n$ and the auxiliary vector $\boldsymbol{v}$ and parameterised by an angular variable θ

$$\boldsymbol{x}^*(\theta) = (\boldsymbol{x}_n - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{v} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu}. \tag{2.49}$$

If we generated θ uniformly from $\mathcal{U}(0, 2\pi)$ then the corresponding proposed transition $\boldsymbol{x}^*(\theta)$ would exactly leave the distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ invariant. As we instead wish to leave the target distribution invariant, a slice sampling algorithm is used to find a θ which accounts for the difference between the target distribution and normal approximation. An auxiliary slice height variable h is sampled uniformly from $\mathcal{U}(0, \tilde{p}(\boldsymbol{x}_n) / \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \Sigma))$ and used to define a slice

$$S_h = \left\{ \boldsymbol{x} \in X : \frac{\tilde{p}(\boldsymbol{x}_n)}{\mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \Sigma)} < h \right\}. \tag{2.50}$$

Similar to the linear slice sampling algorithm, a bracket $[\theta_l, \theta_u]$ on the elliptical path is randomly placed around $\theta = 0$ corresponding to the current state $\boldsymbol{x}_n$. Unlike the requirement to choose a suitable initial bracket width in linear slice sampling however, we can define the initial bracket in elliptical slice sampling to include the entire elliptical

path i.e. $\theta_l = \theta_u - 2\pi$; we only need to randomise the 'cut-point' defining the initial end-points of the bracket to ensure reversibility. This removes the need to choose an initial bracket width (defined by $|\boldsymbol{v}|$ in our description of the linear slice algorithm) and for any step out procedure, and so beyond choosing the multivariate normal approximation elliptical slice sampling does not have any free settings which need to be tuned.

Once the initial bracket is defined, a directly analogous iterative procedure to that used in the linear slice sampling algorithm is used to find a θ value corresponding to a point in the slice while using rejected proposed points to shrink the bracket. As with linear slice sampling, providing the target density is a smooth function and so the intersection of the elliptical path with the slice is a non-zero measure set, then the state moved to by the elliptical slice sampling transition operator will never be equal to the previous state.

2.3.2 Hamiltonian Monte Carlo

The MCMC algorithms discussed so far have required only the ability to evaluate a (unnormalised) density function for the target distribution of interest. For distributions defined on real-valued variables the target density function $\tilde{p}$ will often be differentiable - the gradient $\frac{\partial \tilde{p}}{\partial \boldsymbol{x}}$ exists P-almost everywhere. In these cases it is natural to consider using the gradient to help guide updates to the state. In particular we might hope to reduce the random-walk behaviour of simpler methods which leads to a slow diffusive exploration of high-dimensional spaces.

What we refer to as Hamiltonian Monte Carlo here was introduced in [81] as Hybrid Monte Carlo. *The alternative* Hamiltonian Monte Carlo *was suggested by MacKay [161] to emphasise the role of Hamiltonian dynamics in the method while maintaining the same acronym [34].*

A particularly powerful auxiliary variable MCMC method utilising gradient information is *Hamiltonian Monte Carlo* (HMC) [81, 192]. HMC introduces auxiliary *momentum* variables in to the chain state and then uses simulated Hamiltonian dynamics trajectories in the augmented space to generate proposed updates to the momentum–target variables state pair. The simulated Hamiltonian dynamics exhibit key geometric properties that make HMC well suited to performing MCMC in complex target distributions on high-dimensional spaces [37], with the method able to propose long-range moves with a high probability of acceptance. The reduced random-walk behaviour means that HMC often scales better to high-dimensional target distributions than simpler methods such as random-walk Metropolis and Gibbs sampling [33]. Under the assumption of a target distribution consisting of a product of independ-

ent factors on D variables, optimally tuned random-walk Metropolis transitions will take $O(D^2)$ computational effort to achieve near independence between states of the chain while an optimally tuned HMC transition can achieve the same with a $O(D^{\frac{5}{4}})$ cost [192].

Most implementations of HMC require the target density p is defined with respect to the Lebesgue measure on a Euclidean space $X = \mathbb{R}^D$. Target densities with bounded support on $\mathbb{R}^D$ add complication to the algorithm by requiring checks that proposed updates to the state remain within the support of the target distribution and reflecting at the boundaries of the support [192]. Often however a change of variables can be performed with a bijective transformation (using Equation 1.22) that maps to a density with unbounded support, for example taking a log-transform of a positive variable.

Rather than working directly with the unnormalised target density $\tilde{p}$ the HMC algorithm is more naturally described in terms of a *potential energy* function $\phi : \mathbb{R}^D \to \mathbb{R}$ which is related to $\tilde{p}$ by

$$\tilde{p}(\boldsymbol{x}) = \exp(-\phi(\boldsymbol{x})) \iff \phi(\boldsymbol{x}) = -\log \tilde{p}(\boldsymbol{x}). \tag{2.51}$$

The original *target variables* $\mathbf{x} \in \mathbb{R}^D$ are augmented with a vector of *momentum variables* $\mathbf{p} \in \mathbb{R}^D$. The conditional density on the momenta given the target variables $\mathrm{p}_{\mathbf{p}|\mathbf{x}}$ is defined in terms of a *kinetic energy* function $\tau : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ which is even in its first argument

$$\mathrm{p}_{\mathbf{p}|\mathbf{x}}(\boldsymbol{p}|\boldsymbol{x}) \propto \exp(-\tau(\boldsymbol{p} \mid \boldsymbol{x})). \tag{2.52}$$

The joint density on the momentum and target variables is then

$$\mathrm{p}_{\mathbf{x},\mathbf{p}}(\boldsymbol{x}, \boldsymbol{p}) \propto \exp(-\phi(\boldsymbol{x}) - \tau(\boldsymbol{p} \mid \boldsymbol{x})) = \exp(-h(\boldsymbol{x}, \boldsymbol{p})). \tag{2.53}$$

The function $h(\boldsymbol{x}, \boldsymbol{p}) = \phi(\boldsymbol{x}) + \tau(\boldsymbol{p} \mid \boldsymbol{x})$ is termed the *Hamiltonian* for the system. A common simplification is for the momenta to be chosen to be independent of the target variables with a marginal density defined by a kinetic energy $\tau : \mathbb{R}^D \to \mathbb{R}$

$$\mathrm{p}_{\mathbf{p}}(\boldsymbol{p}) \propto \exp(-\tau(\boldsymbol{p})). \tag{2.54}$$

In this case the Hamiltonian $h(\boldsymbol{x}, \boldsymbol{p}) = \phi(\boldsymbol{x}) + \tau(\boldsymbol{p})$ is *separable* - there are no terms jointly dependent on both $\boldsymbol{x}$ and $\boldsymbol{p}$.

Commonly a quadratic form $\tau(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{M}^{-1}\boldsymbol{p}$ is used for the kinetic energy where $\boldsymbol{M}$ is a positive definite matrix typically termed the *mass matrix*. A quadratic kinetic energy corresponds to assuming normally distributed momenta with zero-mean and covariance $\boldsymbol{M}$.

In classical mechanics, the Hamiltonian describes the total energy of a mechanical system, and can be used to define a *canonical Hamiltonian dynamic* via the set of *ordinary differential equations* (ODEs)

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial h^\mathsf{T}}{\partial \boldsymbol{p}}, \qquad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial h^\mathsf{T}}{\partial \boldsymbol{x}}. \tag{2.55}$$

We define the *flow map* corresponding to this dynamic as a family of mappings $\boldsymbol{\psi}_t : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D \times \mathbb{R}^D$ parameterised by a time $t \in \mathbb{R}$ such that if $(\boldsymbol{x}(t), \boldsymbol{p}(t))$ is the solution to the set of ODEs (2.55) at a time t given an initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0, \boldsymbol{p}(0) = \boldsymbol{p}_0$ then

$$\boldsymbol{\psi}_t(\boldsymbol{x}_0, \boldsymbol{p}_0) = (\boldsymbol{x}(t), \boldsymbol{p}(t)). \tag{2.56}$$

The Hamiltonian flow map has several desirable properties as a proposal generating mechanism for a MCMC method. The Hamiltonian is exactly conserved along the trajectories generated by the flow map, i.e. $h(\boldsymbol{x}, \boldsymbol{p}) = h \circ \boldsymbol{\psi}_t(\boldsymbol{x}, \boldsymbol{p})$ for all $t \in \mathbb{R}$ and for any initial $\boldsymbol{x}, \boldsymbol{p}$ pair. As $\mathrm{p}_{\mathbf{x},\mathbf{p}}(\boldsymbol{x}, \boldsymbol{p}) \propto \exp(-h(\boldsymbol{x}, \boldsymbol{p}))$ this means Hamiltonian trajectories remain confined to constant density surfaces in the augmented state space.

The Hamiltonian flow map is also *volume preserving* - the Jacobian of the flow map, $\mathbf{J}_{\boldsymbol{\psi}_t}$ has determinant one for all t and starting from any initial $(\boldsymbol{x}, \boldsymbol{p})$. This volume preservation is a consequence of a stronger geometric property of the dynamic - that the flow map is *symplectic* [152]. Symplecticity of the flow map is implied by the condition

$$\mathbf{J}_{\boldsymbol{\psi}_t}^\mathsf{T} \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{J}_{\boldsymbol{\psi}_t} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \tag{2.57}$$

being satisfied. The symplectic nature of Hamiltonian dynamics is central to efficient scaling of HMC to high-dimensional spaces [37, 192].

A final crucial property of the Hamiltonian flow map is that it exhibits a time-reversal symmetry under negation of the momenta

$$(\boldsymbol{x}', \boldsymbol{p}') = \boldsymbol{\psi}_t(\boldsymbol{x}, \boldsymbol{p}) \iff (\boldsymbol{x}, -\boldsymbol{p}) = \boldsymbol{\psi}_t(\boldsymbol{x}', -\boldsymbol{p}'). \tag{2.58}$$

If we define $\boldsymbol{r} : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D \times \mathbb{R}^D$ as a 'momentum-reversal' operator such that $\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{p}) = (\boldsymbol{x}, -\boldsymbol{p})$ then this time-reversal symmetry means that the composition $\boldsymbol{r} \circ \boldsymbol{\psi}_t$ is an involution. Further as $\boldsymbol{r}$ also has Jacobian determinant one, then the composition $\boldsymbol{r} \circ \boldsymbol{\psi}_t$ also itself has a unit Jacobian determinant.

Using the previous result for the Metropolis–Hastings accept ratio for the special case of a deterministic proposal formed by an involution (2.39), we have that a proposal generated by applying $\boldsymbol{r} \circ \boldsymbol{\psi}_t$ to the current state pair $(\boldsymbol{x}, \boldsymbol{p})$ for any t has an accept probability of one

$$\alpha(\boldsymbol{x}, \boldsymbol{p}) = \min\left\{1, \frac{\exp(-h \circ \boldsymbol{r} \circ \boldsymbol{\psi}_t(\boldsymbol{x}, \boldsymbol{p}))}{\exp(-h(\boldsymbol{x}, \boldsymbol{p}))} \left|\mathbf{J}_{\boldsymbol{r} \circ \boldsymbol{\psi}_t}(\boldsymbol{x}, \boldsymbol{p})\right|\right\} \tag{2.59}$$

$$= \min\{1, \exp(h(\boldsymbol{x}, \boldsymbol{p}) - h \circ \boldsymbol{r} \circ \boldsymbol{\psi}_t(\boldsymbol{x}, \boldsymbol{p}))\} = 1. \tag{2.60}$$

This is a result of the conservation of the Hamiltonian under the flow map (and momentum-reversal operator as τ is even in the momenta) and the composed map having unit Jacobian determinant. Therefore proposals formed by integrating the ODEs forward by some length of time from the current state and then reversing the momentum would always be accepted.

On its own the momentum reversal operator $\boldsymbol{r}$ is also an involution with unit Jacobian determinant which exactly conserves the Hamiltonian, and so can also be applied as a 'proposal' with probability of acceptance of one. If we sequentially alternate updates using $\boldsymbol{r} \circ \boldsymbol{\psi}_t$ and $\boldsymbol{r}$, each defines a valid Markov transition operator which leaves the (extended) target invariant, and in sequential composition the momentum reversals cancel. We can therefore construct a Markov chain which leaves the target distribution with density (2.53) invariant by repeatedly generating new states by integrating the Hamiltonian dynamic forward by arbitrary lengths of time. Note that though each of $\boldsymbol{r} \circ \boldsymbol{\psi}_t$ and $\boldsymbol{r}$ are individually reversible and so respect detailed balance, the sequential composition is no longer reversible.

There are two major problems with this scheme. Firstly for most ϕ and τ it is not possible to integrate the ODEs (2.55) exactly and so we cannot evaluate the exact flow map $\boldsymbol{\psi}_t$. Secondly the scheme as proposed would not be ergodic as the Hamiltonian is conserved by each application of the flow map, and so all states generated in this way would be confined to a constant Hamiltonian manifold in the joint space.

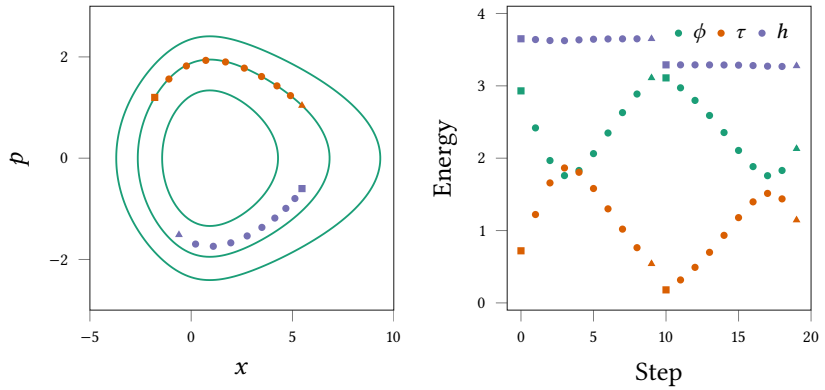


Figure 2.12.: Visualisation of Hamiltonian Monte Carlo applied to a univariate target density $p(x) = \alpha \exp(-\alpha x)(1 + \exp(-x))^{-1-\alpha}$ with $\alpha = 0.4$. The left axis shows contours (green curves) of the Hamiltonian function on the augmented (x, p) state space. The orange markers shows a Hamiltonian trajectory simulated using the leapfrog method, starting at the square marker and finishing at the triangular marker. The trajectory nearly exactly traces a Hamiltonian contour due to the approximate energy conservation of the simulated dynamic, with the proposed update (from square to triangular markers) therefore accepted with high probability. At the end of the orange trajectory the momentum is randomly resampled, giving a new initial state (purple square marker) for a second simulated trajectory shown by the purple markers. The right axis shows the variation in the Hamiltonian h, potential energy ϕ and kinetic energy τ over the two trajectories. In each trajectory the Hamiltonian is close to constant, with shifts in the potential energy matched by opposing shifts in the kinetic energy. There is a step change in the kinetic energy and Hamiltonian when then momentum is resampled at the end of first trajectory.

The first issue can be resolved by approximately integrating the ODEs. Importantly by using a *symplectic integrator* we are able to form approximate Hamiltonian flow maps which maintain the key volume-preservation and time-reversibility properties of the exact flow map dynamic and define, as the name suggests, symplectic maps. There is a large class of such symplectic integrators [152] however for separable Hamiltonians an appealingly simple scheme is the *Störmer–Verlet* or *leapfrog* integrator. If we first define the following component maps

$$\hat{\boldsymbol{\psi}}^{\mathrm{A}}_{\delta t}(\boldsymbol{x}, \boldsymbol{p}) = \left(\boldsymbol{x} + \delta t \nabla \tau(\boldsymbol{p})^{\mathsf{T}}, \boldsymbol{p}\right), \; \hat{\boldsymbol{\psi}}^{\mathrm{B}}_{\delta t}(\boldsymbol{x}, \boldsymbol{p}) = \left(\boldsymbol{x}, \boldsymbol{p} - \delta t \nabla \phi(\boldsymbol{x})^{\mathsf{T}}\right), \quad (2.61)$$

then a leapfrog step is defined by the symmetric composition

$$\hat{\boldsymbol{\psi}}^{\mathrm{LF}}_{\delta t} = \hat{\boldsymbol{\psi}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\psi}}^{\mathrm{A}}_{\delta t} \circ \hat{\boldsymbol{\psi}}^{\mathrm{B}}_{\frac{\delta t}{2}}. \quad (2.62)$$

Each leapfrog step is time-reversible and volume conserving. The composition of L leapfrog steps $\left(\hat{\boldsymbol{\psi}}^{\text{LF}}_{\delta t}\right)^L$ maintains these properties. The alternative symmetric composition $\hat{\boldsymbol{\psi}}^{\text{A}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\psi}}^{\text{B}}_{\delta t} \circ \hat{\boldsymbol{\psi}}^{\text{A}}_{\frac{\delta t}{2}}$ also defines a symplectic integrator and is also sometimes termed the leapfrog method. In practice when multiple leapfrog steps are composed together the intermediate half time-steps can be combined, for example using (2.62)

$$\left(\hat{\boldsymbol{\psi}}^{\text{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\psi}}^{\text{A}}_{\delta t} \circ \hat{\boldsymbol{\psi}}^{\text{B}}_{\frac{\delta t}{2}}\right) \circ \left(\hat{\boldsymbol{\psi}}^{\text{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\psi}}^{\text{A}}_{\delta t} \circ \hat{\boldsymbol{\psi}}^{\text{B}}_{\frac{\delta t}{2}}\right) = \hat{\boldsymbol{\psi}}^{\text{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\psi}}^{\text{A}}_{\delta t} \circ \hat{\boldsymbol{\psi}}^{\text{B}}_{\delta t} \circ \hat{\boldsymbol{\psi}}^{\text{A}}_{\delta t} \circ \hat{\boldsymbol{\psi}}^{\text{B}}_{\frac{\delta t}{2}}$$

and so the two different symmetric compositions only differ by whether initial and final half momentum or position time steps are taken.

Although an approximate flow map will no longer exactly conserve the Hamiltonian, a key property of symplectic integrators, including the leapfrog method, is that they correspond to the exact flow map of a modified Hamiltonian system. Providing the integrator step-size δt is below a stability threshold this modified Hamiltonian $\tilde{h}_{\delta t}$ will be close to the original target Hamiltonian: $\left|h(\boldsymbol{x},\boldsymbol{p}) - \tilde{h}_{\delta t}(\boldsymbol{x},\boldsymbol{p})\right| \leq O(\delta t^k)$ where k is the order of the integrator ($k = 2$ for the leapfrog method) [152]. As the approximate flow map exactly conserves this modified Hamiltonian, this means that the change in the Hamiltonian over long simulated trajectories will remain bounded. If we replace the exact flow map for the approximate flow map corresponding to L steps of the leapfrog integrator with step size δt in (2.59) then we have that the probability of accepting a proposal generated by approximately integrating the ODEs and then negating the momentum is

$$\alpha(\boldsymbol{x},\boldsymbol{p}) = \min\left\{1, \exp\left(h(\boldsymbol{x},\boldsymbol{p}) - h \circ \boldsymbol{r} \circ \left(\hat{\boldsymbol{\psi}}^{\text{LF}}_{\delta t}\right)^L(\boldsymbol{x},\boldsymbol{p})\right)\right\}. \tag{2.63}$$

As the change in Hamiltonian over the trajectory remains bounded, if the step-size is small enough the probability of acceptance will remain close to one even for a large number of integrator steps L. This means simulated Hamiltonian dynamics can be used to form long-range proposed moves which maintain a high probability of acceptance. An example of this approximate conservation of the Hamiltonian is shown in Figure 2.12 which shows a visualisation of trajectories simulated using the leapfrog method in a system with a one-dimensional target variable x and the variation in the Hamiltonian, potential and kinetic energies along these trajectories.

Algorithm 6 Hamiltonian Monte Carlo.

Input: $\boldsymbol{x}_n$: current target variables state, ϕ : differentiable potential energy function $= -\log \tilde{p}$, δt : leapfrog integrator step size, L : number of leapfrog integration steps, $\boldsymbol{M}$: mass matrix.

Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p \implies \boldsymbol{x}_{n+1} \sim p$.

$\boldsymbol{p} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{M})$ ▷ Independently sample new momentum vector.
$\boldsymbol{p}^* \leftarrow \boldsymbol{p} - \frac{\delta t}{2} \nabla \phi(\boldsymbol{x}_n)^\mathsf{T}$ ▷ Initial half momentum step $\hat{\boldsymbol{\psi}}^{\mathrm{B}}_{\frac{\delta t}{2}}$.
$\boldsymbol{x}^* \leftarrow \boldsymbol{x}_n + \delta t \boldsymbol{M}^{-1} \boldsymbol{p}^*$ ▷ $\hat{\boldsymbol{\psi}}^{\mathrm{A}}_{\delta t}$.
for $s \in \{1 \dots L-1\}$ **do**
    $\boldsymbol{p}^* \leftarrow \boldsymbol{p}^* - \delta t \nabla \phi(\boldsymbol{x}^*)^\mathsf{T}$ ▷ $\hat{\boldsymbol{\psi}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\psi}}^{\mathrm{B}}_{\frac{\delta t}{2}}$.
    $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^* + \delta t \boldsymbol{M}^{-1} \boldsymbol{p}^*$ ▷ $\hat{\boldsymbol{\psi}}^{\mathrm{A}}_{\delta t}$.
$\boldsymbol{p}^* \leftarrow \boldsymbol{p}^* - \frac{\delta t}{2} \nabla \phi(\boldsymbol{x}^*)^\mathsf{T}$ ▷ Final half momentum step $\hat{\boldsymbol{\psi}}^{\mathrm{B}}_{\frac{\delta t}{2}}$.
$u \sim \mathcal{U}(0, 1)$
$\alpha \leftarrow \exp\left(\phi(\boldsymbol{x}_n) + \frac{1}{2} \boldsymbol{p} \boldsymbol{M}^{-1} \boldsymbol{p} - \phi(\boldsymbol{x}^*) - \frac{1}{2} \boldsymbol{p}^* \boldsymbol{M}^{-1} \boldsymbol{p}^*\right)$ ▷ Accept probability.
if $u < \alpha$ **then**
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}^*$ ▷ Proposed move accepted.
else
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_n$ ▷ Proposed move rejected.
return $\boldsymbol{x}_{n+1}$

When using an approximate flow map as there is now a non-zero probability of rejection, upon rejecting the momentum will remain at its current state, before then being negated. The next simulated trajectory will therefore backtrack along a previous trajectory after a rejection. To prevent this backtracking behaviour and resolve the issue that Markov transitions consisting solely of simulated dynamics proposals and momentum-reversals would remain confined to a constant modified Hamiltonian manifold, a further update is introduced in to the overall HMC transition which changes the momenta while leaving the target variables fixed.

For the common case of a quadratic kinetic energy $\tau(\boldsymbol{p}) = \frac{1}{2} \boldsymbol{p}^\mathsf{T} \boldsymbol{M}^{-1} \boldsymbol{p}$ and so normal marginal distribution on the momenta, the simplest way to update the momenta is to independently sample a new vector from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{M})$ at the beginning of each HMC transition. This will both perturb the initial Hamiltonian of the system and also mean the initial direction of any simulated trajectory is randomised so that the negation of the previous momentum upon a rejection does not lead to backtracking. In this case as the momentum is independently resampled in each transition there is no need to store the momentum state between successive transitions and the overall HMC transition will be reversible.

Algorithm 6 describes the overall HMC transition corresponding to using a quadratic kinetic energy $\tau(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{M}^{-1}\boldsymbol{p}$, independent momentum resampling and a leapfrog integrator. The integrator step size δt and number of steps L together determine the total approximate integration time $T = L\delta t$. Intuitively we want to choose T to minimise the dependence of the generated proposals on the current point. In general however we will not know what this optimal integration time is and in most problems it will depend on the starting state. Choosing an appropriate integration time can therefore be challenging, and will often involve some level of trial and error with pilot runs [192]. Too small values lead to dynamics proposals which remain close to the current state which combined with the independent resampling of the momenta on each transition lead to random-walk like behaviour for the overall transition, with limited gain from using the HMC transition over simpler methods such as random-walk Metropolis.

The computational cost of each HMC transition will however scale linearly with L and so the integration time, therefore it is desirable to not increase the integration time beyond the point where there is any gain in decreased dependence between successive points; further as typically the simulated trajectories will be quasi-periodic increasing the integration time can in some cases lead to proposals moving closer to the original state. The integration time does not need to be the same for each transition and randomising it by for example uniformly sampling from an interval can be helpful in some problems to reduce pathological behaviour due to near periodicity of trajectories [192].

In combination with the integration time an appropriate value must also be chosen for the integrator step size δt. As for a fixed integration time T the step size determines the number integrator steps needed and so computational cost per transition, we ideally want to use as large a step size as possible. The step size however also controls how large the typical change in the Hamiltonian is across a simulated trajectory and so the accept rate for the proposed updates. As the step size increases the average accept rate will decrease and beyond some limit typically the dynamic will become unstable and the Hamiltonian error no longer remain bounded, leading to very low accept rates for large L. Typically this stability limit will vary depending on the starting state, so we may occasionally encounter unstable diverging trajectories even when the step size is small enough for most trajectories to remain stable.

Analogously to results for tuning the proposal width for random-walk Metropolis methods, guidelines have been derived for choosing the integrator step-size in HMC based on an optimal (in the sense of maximising some measure of computational efficiency) average accept probability for the Metropolis step. A target accept rate of 0.65 has been suggested [26, 192] under idealised assumptions of a high-dimensional target distribution in which the individual dimensions are independent and when using the leapfrog integrator. Under more general assumptions in [35] an accept rate range of 0.6 to 0.9 was instead recommended as giving close to optimal performance for symplectic integrators of order 2 including the leapfrog method.

To help address the challenges of tuning the free integrator step-size and integration time parameters of the standard HMC algorithm, adaptive variants which automatically tune these parameters have been proposed. Of particular note is the *no U-turn sampler* (NUTS) algorithm [130]. Rather than using a single fixed integration time, NUTS dynamically varies the length of the simulated trajectories, expanding the trajectories until a termination criterion corresponding intuitively to the trajectory 'turning back on itself' (hence the name) is met and then using a slice sampling scheme to select a new state from this dynamically generated trajectory. Maintaining reversibility in such a scheme is non-trivial and NUTS uses an elegant recursive method to do so: full details are beyond the scope of this review but both [130] and a subsequent review article [33] provide excellent visual explanations of the algorithm. The dynamic integration time scheme is combined in NUTS with a stochastic optimisation method for tuning the integrator step-size to achieve a target acceptance rate, with a vanishing adaptation rate ensuring convergence of the chains to stationarity [9].

Refinements to NUTS have been suggested including generalised termination criteria [29, 32] and an extension to use multinomial sampling of the final state from the generated trajectory instead of slice sampling [32, 33]. The original NUTS algorithm and these refinements have seen widespread empirical success through their implementation in the probabilistic programming framework *Stan* [55] which combines a general purpose probabilistic model specification language [247] and automatic differentiation library [54] with efficient implementations of approximate inference methods including NUTS.

We have so far neglected to mention how the mass matrix $\boldsymbol{M}$ is chosen. The simplest choice is to use $\boldsymbol{M} = \mathbf{I}$; this is a reasonable choice when the target density has a close to isotropic geometry with approximately equal scaling of each dimension and no strong correlations between variables. Using a non-identity mass matrix is equivalent to running HMC with a identity mass matrix in a linearly transformed reparameterisation of the target density [192] and so can be used to account for non-isotropic target densities by rescaling and decorrelating the variables of the target distribution. Ideally the mass matrix would be chosen based on the covariance of the target distribution [33, 192] however in practice this will typically not be available. One option is to estimate the target covariance during an initial adaptive phase in the chain which is used within the NUTS implementation in Stan [55].

In reality for complex target distributions the geometry of the density will vary across the state space with position-dependent curvature. In these cases a single constant mass matrix will be ineffective at locally decorrelating and normalising the scale of the variables corresponding to the different dimensions of the target distribution. This can degrade the performance of HMC methods, with no single step size appropriate in all regions of the state space and typically a trade-off needing to be made between choosing a small step size based on the scale of the most constrained directions and choosing a larger step size for efficiency with the possible result of the simulated dynamic being unable to enter tightly constrained regions of the target distribution [28].

For a quadratic kinetic energy function $\tau(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p}$, considering the kinetic energy as a random variable $\tau = \tau(\mathbf{p})$, as $\mathbf{p}$ has a multivariate normal marginal distribution, τ will have mean $D/2$ and standard deviation $\sqrt{D}$ [192]. As the kinetic energy is bounded below by zero and the Hamiltonian and so sum of kinetic and potential energies approximately conserved along simulated trajectories, the maximal increase in the potential energy along a trajectory is approximately upper bounded by the initial kinetic energy. The potential energy will therefore typically vary by an amount of order D over simulated trajectories. For complex target distributions with varying curvature and scales, the potential energy will often vary by much more than D across the typical set of the distribution and so any single trajectory will typically be only able to cover a small region in the typical set, with exploration of the full typical set of the distribution then degrading to a random-walk like

behaviour as only the momentum resampling steps allows movement up and down the full potential energy range [28, 33].

The naming of RMHMC arises from a connection to Riemannian geometry, with the metric defining a Riemannian manifold, with the length of shortest path between two points on the manifold, i.e. a geodesic, *locally defined by the metric.*

A suggested resolution to the issue of varying curvature across the target density is to use a mass matrix which depends on the target state $\mathbf{x}$. *Riemannian-manifold Hamiltonian Monte Carlo* (RMHMC) [107] defines a non-separable Hamiltonian using a kinetic energy function

$$\tau(\boldsymbol{p} \mid \boldsymbol{x}) = \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}(\boldsymbol{G}(\boldsymbol{x}))^{-1}\boldsymbol{p} + \frac{1}{2}\log|\boldsymbol{G}(\boldsymbol{x})| \tag{2.64}$$

corresponding to $\mathrm{p}_{\mathbf{p}|\mathbf{x}}(\boldsymbol{p} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{p} \mid \boldsymbol{0}, \boldsymbol{G}(\boldsymbol{x}))$ where $\boldsymbol{G} : \mathbb{R}^D \to \mathbb{R}^{D \times D}$ is a positive-definite matrix function termed the *metric*. In analogy to the earlier mentioned equivalence between using a non-identity constant mass matrix and running HMC with an identity mass matrix in a reparameterised target distribution in terms of a linear transformation of the original target variables [192], RMHMC can be shown to be equivalent to running HMC with an identity mass matrix in a reparameterisation of the target distribution in terms of a *non-linear* transformation of the target variables [195]. This non-linear reparameterisation can locally transform the target distribution so that the resulting density has a geometry more amenable to exploration by the HMC dynamic.

Various schemes have been proposed for choosing a metric for a particular target distribution. In [107] the *Fisher–Rao metric* [5] is suggested as it provides a natural description of the Riemannian geometry of parametric probability distributions and so is particularly relevant for target distributions corresponding to the posterior of Bayesian inference problems for models of IID datasets. The Fisher–Rao metric only has a closed form solution however for a limited set of distributions. An alternative more generally applicable metric based on a regularisation of the Hessian of the log target density to ensure positive-definiteness was suggested in [28]. A 'geometrically tempered' metric designed to help exploration of multimodal distributions was suggested in [195].

The non-separable nature of the Hamiltonian in RMHMC means that the standard leapfrog method cannot be employed to simulate the resulting dynamic, with alternative symplectic integrators such as the *generalised leapfrog method* [152] required. These integrators involve implicit steps which requires solving a set of non-linear equations on each iteration. Further evaluation of the inverse of the metric and its log determinant in general have a cost which scales cubically with D, there-

fore the overall computational cost of simulating the RMHMC dynamic is much higher per transition than for standard HMC. In some target distributions the gain in sampling efficiency over standard HMC from using RMHMC updates can significantly outweigh the increased computational cost per sample however [28, 107].

2.3.3 Simulated tempering

The final auxiliary variable method we consider, *simulated tempering* [164], simulates the dynamics of a thermodynamic system subject to a varying temperature. Simulated tempering was originally proposed to improve the exploration of highly-multimodal distributions defined by undirected models such as the Ising spin model.

A particle of systems with a state described by a vector $\boldsymbol{x} \in X$ and a total energy determined by a function $\phi : X \to \mathbb{R}$ will have an equilibrium distribution on the state at a temperature T which has a density proportional to $\exp(-\beta\phi(\boldsymbol{x}))$ where $\beta = (kT)^{-1}$ is the *inverse temperature* and k is Boltzmann's constant. If the energy function ϕ is 'rough' with multiple local minima, then as the temperature T tends to zero and $\beta \to \infty$ the corresponding peaks in the density function become increasingly sharp and the mass of the distribution more tightly concentrated around these peaks. Conversely as the temperature T increases and $\beta \to 0$, the density becomes increasingly flat across X.

Simulated annealing [1, 140], is a stochastic optimisation method which uses this intuition about the properties of thermodynamical systems to improve the probability of an optimisation routine converging to a global optima in highly multimodal objectives. The objective function to be minimised is identified with the energy function ϕ of the system and the variables being optimised with the state $\boldsymbol{x} \in X$. An increasing *schedule* of K inverse temperatures $\{\beta_k\}_{k=1}^K$ is chosen with $0 \leq \beta_1 \leq \beta_2 \leq \cdots \leq \beta_K \leq \infty$. An initial value for the target variables $\boldsymbol{x}_0$ is (randomly) chosen and new values for the target variables are then computed iteratively for each $k \in \{1 \ldots K\}$ by applying a Metropolis–Hastings transition operator $\boldsymbol{x}_k \sim \mathrm{T}_k(\cdot \mid \boldsymbol{x}_{k-1})$ which leaves the distribution with density proportional to $\exp(-\beta_k\phi(\boldsymbol{x}))$ invariant.

The hope is that the transitions at low inverse temperatures will be able to move freely around the target space due to the relatively flat form of the density function with lowered barriers between modes. Ideally

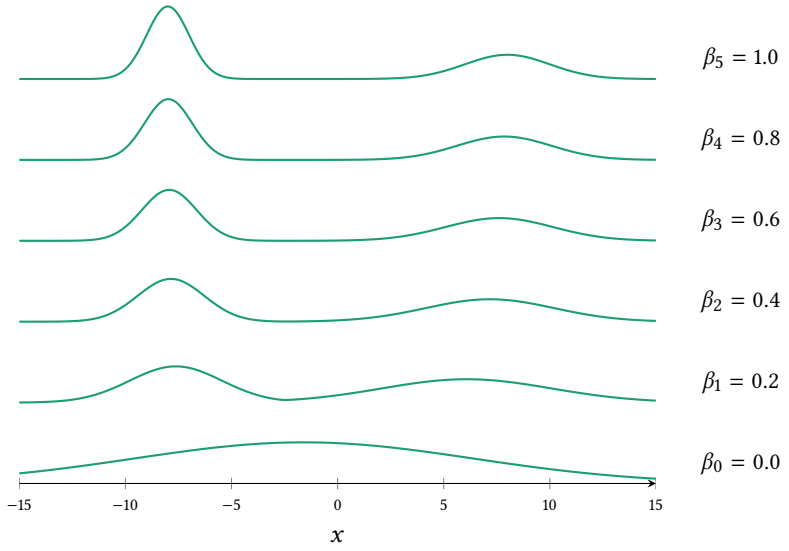


Figure 2.13.: Example of using an inverse temperature to geometrically bridge between unimodal base and bimodal target densities. Each curve shows the conditional density $\mathrm{p}_{\mathrm{x}|\mathrm{k}}$ corresponding to the joint target density (2.66) for $\mathrm{k} = 0$ to $\mathrm{k} = 5$ with $\beta_k = k/5$ in this case.

the state will therefore tend to converge towards the modes with the largest mass. As the inverse temperature is increased the density function becomes increasingly peaked and the updates will tend to remain confined to one mode and as $\beta \to \infty$ will become concentrated near to the maximum of this mode. Although there is no guarantee this heuristic will find a global optima, empirically it has been found to be useful in practice in a range of applications.

In simulated tempering, rather than using an inverse temperature to define an optimisation procedure instead a discrete index controlling the inverse temperature is introduced as an auxiliary variable in an MCMC method. The variables $\mathbf{x} \in X$ on which the target distribution P is defined are augmented with a discrete index variable $\mathrm{k} \in \{0 \ldots K\}$. A corresponding set of inverse temperature values $\{\beta_k\}_{k=0}^{K}$ are specified, as with simulated annealing these chosen to form an ordered sequence but in this case over the interval $[0, 1]$ with

$$0 = \beta_0 < \beta_1 < \beta_2 < \cdots < \beta_K = 1. \tag{2.65}$$

A joint density on the target variables $\mathbf{x}$ and temperature index k is then defined as

$$\mathrm{p}_{\mathbf{x},\mathrm{k}}(\boldsymbol{x}, k) = \frac{1}{C} \exp(-\beta_k \phi(\boldsymbol{x}) - (1 - \beta_k)\psi(\boldsymbol{x}) + w_k). \tag{2.66}$$

Algorithm 7 Simulated tempering.

Input: $(\boldsymbol{x}_n, k_n)$: current target variables – temperature index state pair, T_1 : transition operator updating only target variables $\mathbf{x}$ and leaving distribution with density in (2.66) invariant, T_2 : transition operator updating only temperature index k and leaving distribution with density in (2.66) invariant.

Output: $(\boldsymbol{x}_{n+1}, k_{n+1})$: new target variables – temperature index state pair.

1: $\boldsymbol{x}_{n+1} \sim \mathsf{T}_1(\cdot \,|\, \boldsymbol{x}_n, k_n)$
2: $k_{n+1} \sim \mathsf{T}_2(\cdot \,|\, \boldsymbol{x}_{n+1}, k_n)$
3: **return** $(\boldsymbol{x}_{n+1}, k_{n+1})$

The values $\{w_k\}_{k=0}^K$ are a set of 'prior' weights associated with each inverse temperature value, and which can be used to help shape the marginal distribution on the temperature index k. As in the preceding subsection the energy function $\phi : X \to \mathbb{R}$ is defined as the negative logarithm of the unnormalised target density i.e. $\phi(\boldsymbol{x}) = -\log \tilde{p}(\boldsymbol{x})$.

The function $\psi : X \to \mathbb{R}$ defines a corresponding energy function for a *base distribution* Q with normalised density $q(\boldsymbol{x}) = \exp(-\psi(\boldsymbol{x}))$ with respect to μ. The base distribution is typically chosen to have a simple unimodal density with mass covering as many of the regions of high density under the target density in X as possible. When the target distribution corresponds to the posterior in a Bayesian inference task, Q is often chosen as the prior distribution on the target variables which will typically have a simple unimodal form and be much more diffuse than the posterior. If the state space X consists of a finite set of values, the base distribution can be chosen to be uniform across X in which case $\psi(\boldsymbol{x})$ is constant and can be omitted from (2.66).

Importantly the conditional distribution $\mathsf{P}_{\mathbf{x}|\mathsf{k}}$ on the target variables $\mathbf{x}$ for $\mathsf{k} = 0$ ($\beta_0 = 0$) corresponds to the base distribution Q and to the target distribution P for $\mathsf{k} = K$ ($\beta_K = 1$). We can therefore use the $\mathbf{x}$ components of sampled chain states for which $\mathsf{k} = K$ to estimate expectations with respect to the target distribution P. For intermediate values of k the conditional distribution geometrically interpolates between P and Q. Figure 2.13 shows a simple example of this geometric bridging between a unimodal univariate base density and bimodal target density for $K = 5$ inverse temperature values.

In simulated tempering, a Markov chain with an invariant distribution corresponding to (2.66) is constructed by alternating updates of the target variables $\mathbf{x}$ given the current value of temperature index k, with updates of the temperature index k given the current value of the tar-

get variables $\mathbf{x}$ as summarised in Algorithm 7. For the transition operator T_1 updating the target variables, any of the previously discussed methods such as random-walk Metropolis, slice sampling or HMC can be used. In the case of Metropolis–Hastings based updates, it may be desirable to adjust the proposal generating mechanism to depend on the current temperature index k as for example we might generally expect for k corresponding to lower inverse temperatures β_{k} and so conditional densities $\mathsf{p}_{\mathbf{x}|\mathsf{k}}$ closer to the base density that larger moves can be made while maintaining reasonable accept rates; as k remains fixed this can validly be done without breaking reversibility.

In the original description of the simulated tempering algorithm in [164], the transitions to the index variable k given fixed values of the target variables $\mathbf{x}$ were performed using a random-walk Metropolis operator for T_2 which proposes to randomly increment or decrement k by one (except at the end-points $\mathsf{k} = 0$ and $\mathsf{k} = K$ where it always proposed to increment and decrement respectively). For large K this can lead to slow mixing up and down the inverse temperature scale - if the marginal density p_{k} is uniform we would expect $O(K^2)$ updates would be needed to traverse the full inverse temperature range. An alternative is to use a Gibbs sampling step with the conditional distribution $\mathsf{P}_{\mathsf{k}|\mathbf{x}}$ here being a multinomial distribution with density

$$\mathsf{p}_{\mathsf{k}|\mathbf{x}}(k \mid \boldsymbol{x}) = \frac{\exp(\beta_k(\psi(\boldsymbol{x}) - \phi(\boldsymbol{x})) + w_k)}{\sum_{k'=0}^{K} \exp(\beta_{k'}(\psi(\boldsymbol{x}) - \phi(\boldsymbol{x})) + w_{k'})} \tag{2.67}$$

which we can tractably generate independent samples from. For arbitrary $\{\beta_k, w_k\}_{k=0}^{K}$ this will require explicit summation over $K+1$ values to calculate the normalising constant and so the cost of generating an independent index will scale linearly with K.

For $\beta_k = \frac{k}{K}$ and $w_k = \alpha\beta_k \; \forall k \in \{0 \dots K\}$ for some $\alpha \in \mathbb{R}$, the normalising constant in (2.67) takes the form of a geometric series

$$\sum_{k=0}^{K} \exp\left(\frac{\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) + \alpha}{K}\right)^k = 1 + \frac{\exp(\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) + \alpha)}{1 - \exp\left(\frac{\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) + \alpha}{K}\right)}. \tag{2.68}$$

The conditional distribution $\mathsf{P}_{\mathsf{k}|\mathbf{x}}$ in this case has the form of a geometric distribution with parameter $\exp\left(\frac{\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) + \alpha}{K}\right)$ truncated to $\{0 \dots K\}$ which we can generate samples at a cost independent of K.

The marginal distribution P_{k} on the index variable k has density

$$\mathrm{p}_{\mathsf{k}}(k) = \frac{\exp(w_k)}{C} \int_X \exp(-\beta_k \phi(\boldsymbol{x}) - (1-\beta_k)\psi(\boldsymbol{x}))\, \mu(\mathrm{d}\boldsymbol{x}). \tag{2.69}$$

As $\int_X \exp(-\psi(\boldsymbol{x}))\, \mu(\mathrm{d}\boldsymbol{x}) = 1$ and $\int_X \exp(-\phi(\boldsymbol{x}))\, \mu(\mathrm{d}\boldsymbol{x}) = Z$ we have

$$\mathrm{p}_{\mathsf{k}}(0) = \frac{\exp(w_0)}{C} \qquad \text{and} \qquad \mathrm{p}_{\mathsf{k}}(K) = \frac{\exp(w_K) Z}{C}. \tag{2.70}$$

If Z is much more than one and $w_k = 0$ for all $k \in \{0 \ldots K\}$ then we would have $\mathrm{p}_{\mathsf{k}}(K) \gg \mathrm{p}_{\mathsf{k}}(0)$ and a simulated tempering chain will tend to spend many more iterations with $\mathsf{k} = K$ than $\mathsf{k} = 0$. This will give a large number of samples with which to estimate expectations with respect to P however it will also limit the gain from using simulated tempering over running a Markov chain in the original non-augmented target variable space, as the chain will rarely visit the lower inverse temperatures which aid exploration. Conversely if Z is much less than one, we have $\mathrm{p}_{\mathsf{k}}(K) \ll \mathrm{p}_{\mathsf{k}}(0)$. In this case the chain will tend to remain at k values corresponding to low inverse temperatures and so few samples are available for computing expectations with respect to P.

If we could set $w_0 - w_K = \log Z$ we would have $\mathrm{p}_{\mathsf{k}}(K) = \mathrm{p}_{\mathsf{k}}(0)$ however for the target distributions of interest we will generally not be able to evaluate Z and a similar result holds for the normalising constants of the conditional distributions $\mathrm{P}_{\mathbf{x}|\mathsf{k}}$ corresponding to intermediate inverse temperatures and so the appropriate values for $\{w_k\}_{k=1}^{K-1}$. In general therefore it will be difficult to identify reasonable values to set the weights $\{w_k\}_{k=0}^{K}$ to a-priori. This is typically solved in practice by using an iterative scheme [131, 164]: an initial pilot chain is run with $w_k = 0\ \forall k$ to estimate the marginal density p_{k} by constructing a histogram of counts of samples for each k and then this histogram used to set the weights so as to approximately flatten the marginal density.

The relationship between the marginal density p_{k} and Z although presenting challenges in terms of choosing the weights $\{w_k\}_{k=0}^{K}$ also however demonstrates that simulated tempering chains can be used to estimate Z. In particular we have that

$$Z = \exp(w_0 - w_K) \frac{\mathrm{p}_{\mathsf{k}}(K)}{\mathrm{p}_{\mathsf{k}}(0)}. \tag{2.71}$$

Given the sampled states $\{\boldsymbol{x}^{(n)}, k^{(n)}\}_{n=1}^{N}$ of a simulated tempering chain, one way to form a consistent estimate of Z is therefore to compute the ratio of the counts of samples with $\mathsf{k} = K$ to those with $\mathsf{k} = 0$,

$$Z = \lim_{N\to\infty} \exp(w_0 - w_K) \frac{\sum_{n=1}^{N} \mathbb{1}_{\{K\}}(k^{(n)})}{\sum_{n=1}^{N} \mathbb{1}_{\{0\}}(k^{(n)})}. \tag{2.72}$$

This estimate will typically have a high variance however as it uses information only from the subset of sampled states with $\mathsf{k} = 0$ or $\mathsf{k} = K$. Expanding p_{k} as a marginalisation integral of the joint density (2.66) we can reformulate the identity in (2.71) as

$$Z = \exp(w_0 - w_K) \frac{\int_X \mathrm{p}_{\mathsf{k}|\mathsf{x}}(K \mid \boldsymbol{x})\, \mathrm{p}_{\mathsf{x}}(\boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x})}{\int_X \mathrm{p}_{\mathsf{k}|\mathsf{x}}(0 \mid \boldsymbol{x})\, \mathrm{p}_{\mathsf{x}}(\boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x})}. \tag{2.73}$$

This is an example of what is sometimes termed *Rao-Blackwellisation* [56] and was used in [53] to suggest a *Rao-Blackwellised estimator* for the normalising constant Z from the samples $\{\boldsymbol{x}^{(s)}, k^{(s)}\}_{s=1}^{S}$ of a simulated tempering chain

$$Z = \lim_{N\to\infty} \exp(w_0 - w_K) \frac{\sum_{n=1}^{N} \mathrm{p}_{\mathsf{k}|\mathsf{x}}(K \mid \boldsymbol{x}^{(n)})}{\sum_{n=1}^{N} \mathrm{p}_{\mathsf{k}|\mathsf{x}}(0 \mid \boldsymbol{x}^{(n)})}. \tag{2.74}$$

This estimator uses all of the sampled chain states and will typically be lower variance than the count-based estimator (2.72). Importantly this estimator can still give reasonable estimates for Z when there are no sampled states for which $\mathsf{k} = 0$ (or $\mathsf{k} = K$) unlike the count-based estimator. This is particularly important when using an iterative scheme to choose the weights $\{w_k\}_{k=0}^{K}$ as if $Z \gg 1$ or $Z \ll 1$ an initial short pilot chain will typically remain confined to one end of the inverse temperature scale for all iterations, giving limited count-based information with which to update weights for subsequent iterations.

2.4 DISCUSSION

The sampling approaches to approximate inference described in this chapter allow tractable estimation of the integrals involved in many inference problems. In cases where we can generate independent samples from the target distribution, the $\frac{1}{N}$ scaling of the variance of Monte Carlo estimates of expectations with the number of samples N allows computation of estimates with sufficient accuracy for most practical

purposes without the exponential blow-up in computation of quadrature methods with dimensionality.

Generating independent samples from arbitrary distributions on high-dimensional spaces is often infeasible however. Transform sampling methods offer a scalable approach for only a few special cases such as the multivariate normal distribution. Rejection sampling is more generally applicable however the usually exponential decrease in the proportion of accepted samples with dimension means that it is only useful in relatively low-dimensional distributions. Simple importance sampling schemes similarly scale poorly with dimensionality, with mismatch between the proposal and target distribution in high-dimensions meaning the variance of the resulting estimators is impractically high.

Although these Monte Carlo methods are not directly applicable to performing inference in the complex probabilistic models of interest, they are still useful building blocks and will appear as components of the methods we will discuss in the rest of this thesis. In Chapter 3 we will discuss MCMC methods which use importance sampling estimators of the target density to construct the chain. The simulator models discussed in Chapter 4 can be considered an extension of the idea of transform sampling, with a complex series of deterministic operations transforming inputs from a pseudo-random number generator to simulated values for the variables in a probabilistic model. One of the standard approaches for performing approximate inference in simulator models is based on rejection sampling, and our discussion of the poor scaling of rejection sampling with dimensionality will be relevant when considering the limitations of these methods.

Markov chain Monte Carlo methods offer a more scalable approach to inference in complex probabilistic models and are the main focus of the work discussed in this thesis. The local perturbative updates typically employed in MCMC methods avoid the curse of dimensionality effects which lead to the exponential blow up in the computational effort required by methods such as rejection sampling as the dimension increases. MCMC methods such as random-walk Metropolis and Gibbs sampling typically require minimal implementation effort and have successfully applied in a wide range of settings.

For target distributions with more complex geometries however such as due to the non-linear relationships between variables often present

in hierarchical models or the multimodal distributions typically arising from inference in undirected models, MCMC methods such as random-walk Metropolis and Gibbs sampling can exhibit pathological behaviour that means impractically long chains are needed for MCMC estimators to give useful results. In these cases methods which exploit more information about the geometry of the distribution in each update can offer significant improvements in efficiency and robustness.

The introduction of auxiliary variables in to the chain state has proved a particularly successful approach for proposing MCMC methods which can accelerate the exploration of complex target distributions. We concluded this chapter by reviewing three auxiliary variable MCMC methods that will be central to the contributions made in this thesis: slice sampling, Hamiltonian Monte Carlo and simulated tempering.

Slice-sampling offers a very generally applicable approach for constructing Markov chains which are able to adapt the scale of proposed moves to the local geometry of the target distribution. The information controlling this adaptation comes from allowing multiple evaluations of the target density per update in slice sampling compared to for example the single target density evaluation per iteration of random-walk Metropolis methods. The overhead from these multiple density evaluations will mean that for target distribution in which the geometry of the density does not vary significantly across the space, well-tuned random-walk Metropolis updates will often be able to outperform slice sampling transition operators in terms of the computational cost per effective independent sample. However the ease of use of slice sampling methods, with typically minimal user tuning required of the free algorithmic parameters, and increased robustness to distributions with more complex geometries, are in our opinion often more important than a potential improvement in peak efficiency.

Hamiltonian Monte Carlo methods put a requirement of differentiability on the target density and so are not as widely applicable as slice sampling approaches. When available however gradient information can be a significant help in guiding the exploration of the target space by a MCMC dynamic. Using reverse-mode automatic differentiation (as described in Appendix B) code for evaluating the exact gradients of a density function can be automatically generated given just the definition of the original density function and the resulting gradient function evaluated at a cost which has only a constant factor overhead over

the cost of the original density function evaluations. When optimally tuned HMC methods can overcome the random-walk behaviour inherent to simpler MCMC methods and so offer significantly improved performance in complex high-dimensional target distributions. Although implementation of HMC algorithms is more complex than approaches such as Gibbs sampling and random-walk Metropolis and the tuning of the algorithm parameters can be vital for good performance, the availability of efficient, adaptive implementations in probabilistic programming frameworks such as Stan [55] and PyMC3 [236] has supported the use of HMC in a wide range of inference problems.

Simulated tempering offers a complementary approach to the improved local exploration afforded by slice sampling and HMC methods by potentially improving the global exploration of challenging multimodal target distributions. As the updates to the target variables at a fixed inverse temperature can be performed using any valid Markov transition operator applicable to the original target distribution, both slice sampling and HMC transition operators can be used within a simulated tempering chain and both potentially offer an improved ability to adapt to the varying geometry of the density on the target variables at different inverse temperatures compared to simpler methods such as random-walk Metropolis. In addition to the possible improved exploration of multimodal targets, the ability to use simulated tempering chains to estimate an unknown normalising constant of the target density, often corresponding to a model evidence term, offers a further distinct advantage over standard MCMC methods.

Although simulated tempering can provide several important benefits, use of the algorithm in statistical applications seems relatively rare in practice. This can perhaps be partially attributed to the need to tune the values of the free inverse temperature β_k and prior weight w_k parameters introduced in the algorithm, with any improvement in exploration of the target distribution strongly dependent on the simulated tempering chain being able to move up and down the inverse temperature range. Further the lack of standard implementations in frameworks such as Stan and PyMC3, and relative wastefulness of the standard approach of estimating expectations with respect to the target distribution by averaging over only sampled states corresponding to an inverse temperature of $\beta_K = 1$, add further discouragements to widespread use of the algorithm.

2.5 OUTLINE OF CONTRIBUTIONS

Having now completed our reviews of both the inference tasks and MCMC methods underlying the work in this thesis, we are now in a position to outline the contributions made in the rest of the thesis.

Inference in hierarchical models is often a challenging task for MCMC methods due to the strong dependencies between the global and local latent variables and resulting complex geometry of the target density on the model variables. We will sometimes only be directly interested in inferring plausible values for the global latent variables in the model but will typically be unable to analytically integrate out the local latent variables. The *pseudo-marginal* framework shows how an unbiased estimator of the marginal density on the global latent variables can be used to construct a Metropolis–Hastings method for sampling values of the global latent variables. Pseudo-marginal Metropolis–Hastings methods however suffer from 'sticking' pathologies where chains reject updates over long series of iterations and are challenging to tune.

In Chapter 3 we demonstrate that by including the auxiliary variables used in the density estimator in the chain state alternative transition operators can be used in a pseudo-marginal setting, including adaptive rejection-free methods like slice-sampling. The resulting *auxiliary pseudo-marginal* methods are able to prevent the sticking artifacts common to existing pseudo-marginal methods, are easier to tune and in some cases give significant improvements in sampling efficiency.

We described simulator models as a challenging setting for approximate inference methods in Chapter 1 due to the lack of an explicit target density on the model variables. *Approximate Bayesian computation* (ABC) is a class of methods for performing inference in such models by conditioning on simulated observations being 'close' rather than exactly equal to the observed data. ABC methods based on both rejection sampling and pseudo-marginal Metropolis–Hastings have been proposed, but both suffer from curse of dimensionality effects that mean further approximation is typically required by reducing the simulated observations and data to lower-dimensional summary statistics.

In Chapter 4 we show that any generative model can be considered as a deterministic transformation of a vector of auxiliary variables from a known distribution. We use this intuition to demonstrate how MCMC

methods such as slice sampling and HMC can be applied within an ABC setting, the improved performance of these methods compared to approaches based on pseudo-marginal Metropolis–Hastings meaning that in some cases ABC inference can be performed without the need for summary statistics. For a restricted class of *differentiable generative models* we derive an expression for conditional expectations under the model in terms of an integral against a distribution defined on an implicitly-defined manifold. We use this to propose a novel constrained HMC method for performing approximate inference in differentiable generative models without an explicit density function on the model variables. This method allows computationally tractable inference when conditioning high-dimensional simulated observations being arbitrarily close to observed data.

Simulated tempering provides an approach for tackling two of the key challenges identified in Chapter 1: performing inference in multimodal distributions such as those defined by undirected models like the Boltzmann machine; estimating the model evidence normalising constant terms required for model comparison. However as noted above simulated tempering is used relatively rarely in practice. In the above discussion we suggested factors which may have discouraged more widespread adoption of the algorithm: the difficulty in choosing the set of inverse temperature and prior weight values to use, the relative inefficiency of using only a small proportion of the samples in a chain to compute estimates and the lack of support for simulated tempering methods in existing inference packages.

In Chapter 5 we suggest approaches to overcome these issues. We propose using a continuous auxiliary variable to control the inverse temperature rather than a discrete index. This sidesteps the need to choose a set of inverse temperature values and allows the auxiliary variable to be jointly updated with the target variables in a HMC update making it straightforward to use tempering within existing HMC-based inference packages. Further we show how all of the samples in a tempered chain can be used to estimate expectations with respect to the target distribution. Finally we demonstrate that variational inference methods provide a natural approach for choosing the base distribution bridged to during tempering and show that cheap biased approximations to the normalising constant of the target density can be exploited to help flatten the marginal density on the inverse temperature.

3 PSEUDO-MARGINAL METHODS

The MCMC methods considered in Chapter 2 provide a widely applicable set of tools for performing inference in probabilistic models where we can evaluate a, potentially unnormalised, density function for the target distribution of interest. In some models we may not be able to directly evaluate such a function however but instead have access to an unbiased estimator of the target density. The pseudo-marginal framework [8] allows MCMC methods to be extended to such problems.

The typical setting for pseudo-marginal methods is that a distribution on an extended set of variables is constructed which has the target distribution as a marginal distribution. Values of a density function for the target distribution are then estimated by using a Monte Carlo method such as importance sampling to approximately marginalise out the additional variables. The variables which are marginalised out may correspond to latent variables specified in the model but that are not of direct interest for the inference task or variables introduced solely for computational reasons. In both cases it will usually be possible to specify a Markov transition operator which leaves the distribution on the extended set of variables invariant, with such schemes often being described as *data augmentation* [245, 257] or *auxiliary variable* [83, 129] methods. Here we will refer to any variables which are marginalised over as auxiliary variables and the variables of interest we wish to infer plausible values for as the target variables.

The density of the joint distribution on auxiliary and target variables will often have a complex geometry with strong dependencies between the variables and in some cases may be multimodal. This can lead to poor exploration of the extended space by simple MCMC schemes such as random-walk Metropolis–Hastings and Gibbs sampling [8]. The motivation for pseudo-marginal methods is that in some cases the density of the marginal distribution on the target variables will have a simpler geometry than the density of the joint distribution on the extended space and therefore be more amenable to exploration by standard MCMC methods.

Although in general we cannot analytically integrate out the auxiliary variables, the pseudo-marginal framework shows how an unbiased estimator of the marginal density can be used within a Metropolis–Hastings update while maintaining the asymptotic exactness of standard MCMC methods. Intuitively the lower the variance of the density estimator the closer the behaviour of the algorithm to the case where the auxiliary variables are analytically marginalised out. We can control the variance of the estimator both by varying the number of auxiliary variable samples used in the Monte Carlo estimate and by using variance reduction methods to increase the estimator efficiency.

By posing the problem of specifying an MCMC algorithm in terms of designing an efficient[1] unbiased estimator of the density of interest, the large literature on methods for constructing low-variance unbiased estimators can be exploited. For example comparatively cheap but biased optimisation-based inference approaches such as Laplace's method (see Appendix C) can be combined with an importance sampling 'debiasing' step to produce an unbiased estimator which can then be used in a pseudo-marginal MCMC update. This provides a way of exploiting cheap but biased approximate inference methods within a MCMC method which still gives guarantees of asymptotically exact results.

The pseudo-marginal framework has been applied to a wide range of probabilistic models where inference might otherwise be intractable. However the standard pseudo-marginal method, which is based on a Metropolis–Hastings transition operator, is susceptible to 'sticking' behaviour where proposed moves are repeatedly rejected for many iterations [8, 239]. The method can also be difficult to tune as it breaks some of the assumptions underlying standard heuristics for adapting the parameters of Metropolis–Hastings methods.

In this chapter we will discuss an alternative formulation of the pseudo-marginal framework which bridges between the approach of directly specifying a Markov transition operator on the extended state space which includes the auxiliary variables and the pseudo-marginal method where the auxiliary variables are approximately marginalised out. This *auxiliary pseudo-marginal* framework still allows the intuitive design of pseudo-marginal algorithms in terms of identifying low-variance unbiased estimators, while overcoming some of the issues of the pseudo-

1 We use 'efficient' in a general sense here rather than the notion of a minimum-variance unbiased estimator satisfying the Cramér-Rao lower bound [67, 220].

marginal Metropolis–Hastings method. In particular it shows how more flexible adaptive MCMC algorithms such as slice-sampling can be used within the pseudo-marginal setting, which can improve the robustness and ease of application of the approach by minimising the amount of user-tuning of free parameters required.

The work summarised in this chapter is based on a collaboration with Iain Murray which resulted in the published conference paper

- Pseudo-marginal slice sampling. Iain Murray and Matthew M. Graham. *The Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, JMLR W&CP 51:911-919*, 2016.

Iain Murray was the main contributor of the ideas proposed in that publication and responsible for the 'doubly-intractable' Gaussian and Ising model experiments in Sections 5.1 and 5.2 of the paper. We discussed the presentation and details of the work together. My individual contribution was implementing and analysing the Gaussian process classification experiments summarised in Section 5.3 of that work, an extended version of which is reproduced in Section 3.6.2 of this chapter. The Gaussian latent variable model experiments discussed in Section 3.6.1 were directly inspired by the experiments in Section 5.1 of the above paper, but we use a different latent variable model formulation for the model here and conduct additional empirical studies of the effect of the variance of the estimator on the relative performance of the algorithms and the sensitivity of the performance of the pseudo-marginal slice sampling algorithms to their free parameters. The text and figures in this chapter are all my own work, though inevitably some of the discussion and analysis is similar to sections of the above publication.

3.1 PROBLEM DEFINITION

As in the previous chapter our goal is to be able to compute estimates of expectations with respect to a target distribution of interest, that is integrals of the form

$$\bar{f} = \int_X f(\boldsymbol{x})\, P(\mathrm{d}\boldsymbol{x}) = \int_X f(\boldsymbol{x})\, p(\boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x}) \tag{3.1}$$

where $f : X \to \mathbb{R}$ is an arbitrary Lebesgue integrable function and P is a probability distribution on a space X with density $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$. We assume

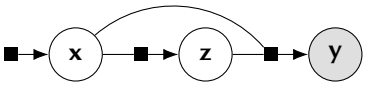


Figure 3.1.: Hierarchical model factor graph.

as previously that the density p may have an intractable normalising constant C that we cannot evaluate i.e. $p(\boldsymbol{x}) = \tilde{p}(\boldsymbol{x})/C$. We make the further assumption here that we cannot directly evaluate $\tilde{p}$ either but only compute an unbiased, non-negative estimate of it. More explicitly we assume we can generate values of a non-negative random variable $\hat{\mathrm{p}}$ from a regular conditional distribution $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}$ such that

$$\tilde{p}(\boldsymbol{x}) = \mathbb{E}[\hat{\mathrm{p}} \mid \mathbf{x} = \boldsymbol{x}] = \int_0^\infty \hat{p}\, \mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}(\mathrm{d}\hat{p} \mid \boldsymbol{x}) \quad \forall \boldsymbol{x} \in X. \tag{3.2}$$

Note that we only require that we can generate $\hat{\mathrm{p}}$ values for a given $\mathbf{x}$, not that we can evaluate a density for $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}$. For concreteness throughout the rest of this chapter we will assume that the target variables take values in a real-valued space $X = \mathbb{R}^D$ and that any density on these variables is defined with respect to the Lebesgue measure $\mu = \lambda^D$.

3.1.1 Example: hierarchical latent variable models

The application of pseudo-marginal methods we focus on is inference in hierarchical probabilistic models where the unobserved variables are split into global latent variables we are interested in inferring and local per datapoint latent variables that we wish to marginalise over the values of, as introduced in Section 1.3.1 in Chapter 1. For notational simplicity we here assume all observed variables are concatenated in a single vector $\mathbf{y}$ and likewise all associated local latent variables in a vector $\mathbf{z}$. The global latent variables, i.e. the target variables for inference, are then $\mathbf{x}$. A factor graph representing the factorisation across the model variables is shown in Figure 3.1.

The target distribution P is then the posterior distribution $\mathrm{P}_{\mathbf{x}|\mathbf{y}}$ given fixed observed values $\boldsymbol{y}$ and the unnormalised target density is chosen as the joint density $\tilde{p}(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y})$. We can express $\tilde{p}$ as a marginal of the joint density $\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}$, which assuming the latent variables $\mathbf{z}$ being marginalised over are real-valued and have a density with respect to the Lebesgue measure can be written

$$\tilde{p}(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y}) = \int_Z \mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{y}, z)\, \mathrm{d}z. \tag{3.3}$$

Generally this integral will not have an analytic solution. We can however form an unbiased estimate of (3.3) using importance sampling. We define a *importance distribution* Q which we can generate independent samples from and with a known density q which in general may depend on the values of the target variables $\boldsymbol{x}$ and observations $\boldsymbol{y}$ and which the target distribution P must be absolutely continuous with respect to. If $\{\mathbf{z}^{(n)}\}_{n=1}^{N}$ are a set of independent variables distributed according to Q then we can define a unbiased density estimator $\hat{\mathrm{p}}$ as

$$\hat{\mathrm{p}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\mathbf{x},\boldsymbol{y},\mathbf{z}^{(n)})}{q(\mathbf{z}^{(n)} \mid \mathbf{x},\boldsymbol{y})} \implies \mathbb{E}[\hat{\mathrm{p}} \mid \mathbf{x} = \boldsymbol{x}] = \tilde{p}(\boldsymbol{x}). \tag{3.4}$$

The variance $\mathbb{V}[\hat{\mathrm{p}}]$ is proportional to $\frac{1}{N}$ and so to decrease the estimator variance we can increase the number of importance samples used, however this comes with the trade-off of an increased computational cost of each density estimate. The estimator variance will also be dependent on the importance distribution used. The optimal choice in terms of minimising variance would be the conditional distribution $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$. Under this choice the density 'estimate' takes the form

$$\hat{\mathrm{p}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\mathbf{x},\boldsymbol{y},\mathbf{z}^{(n)})}{\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\mathbf{z}^{(n)} \mid \mathbf{x},\boldsymbol{y})} = \frac{1}{N}\sum_{n=1}^{N}\mathrm{p}_{\mathbf{x},\mathbf{y}}(\mathbf{x},\boldsymbol{y}) = \tilde{p}(\mathbf{x}) \tag{3.5}$$

and so is equal to the unnormalised target density independent of the sampled $\mathbf{z}^{(n)}$ values with zero variance. In reality however we will not be able to evaluate the density of $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ nor sample from it as this is equivalent to being able to analytically solve the integral in (3.3).

The conditional distribution $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ will often be tractable to sample from and to evaluate the density of and so is a possible choice for the importance distribution. Typically however $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ will be much less concentrated than $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$. This will mean samples from $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ will tend to fall in low density regions of $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$, with only occasionally sampled values being in regions with high density under $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ leading to a high variance estimator, with the problem becoming more severe as the dimension of $\mathbf{z}$ increases. This can mean a large number of importance samples are needed to achieve an estimator with a reasonable variance.

An alternative is to fit an approximation to $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ to use as the importance distribution using for example one of the optimisation-based approximate inference approaches discussed in Appendix C. For example

we could use Laplace's method to fit a multivariate normal approximation $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y}) \approx \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ and use this as the importance distribution. As $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ depends on $\boldsymbol{x}$ this involves fitting an approximation for each $\boldsymbol{x}$ value we wish to evaluate the density at. Although computationally costly the significant variance reduction brought by this approach can make this overhead worthwhile in practice [87].

Inference in hierarchical latent variable models using an importance sampling estimator for the marginal density is just one setting in which pseudo-marginal methods are applied. Other applications of the framework have included inference methods for dynamical state space models using a particle filter estimator [78, 112] for the marginal density of the observed state sequence given the model parameters [7, 61, 209], parameter inference in 'doubly-intractable' distributions [184] where an intractable normaliser depends on the variables of interest using density estimators based on exact sampling methods [178, 181, 216] and random series truncation [160] and approximate inference in simulator models where the density on the simulator outputs is only implicitly defined [165].

In the discussion and experiments in this chapter we will concentrate on latent variable models and importance sampling density estimators of the form described in this section. Examples of applying the methods discussed here to inference in a doubly intractable distribution were discussed in the associated conference paper [185]. Although particle filtering based methods are a major use case of the pseudo-marginal framework, the associated models and estimators tend to be more complex and we have chosen to avoid further expanding the theoretical background material in this thesis by concentrating on simpler cases here. The use of pseudo-marginal MCMC methods to perform inference in simulator models will be a major topic of the next chapter which specifically considers inference methods applicable in this setting so we will delay discussion of models of this form till then.

3.2 PSEUDO-MARGINAL METROPOLIS–HASTINGS

The pseudo-marginal Metropolis–Hastings method is summarised in Algorithm 8. The term *pseudo-marginal* was proposed by Andrieu and Roberts in [8] as part of an extensive theoretical analysis of the pseudo-marginal framework. Andrieu and Roberts cite Beaumont [18] as the

Algorithm 8 Pseudo-marginal Metropolis–Hastings.

Input: $(\boldsymbol{x}_n, \hat{p}_n)$: current target variables – density estimate state pair, $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}$: density estimate conditional distribution, r : proposal density for updates to target variables.

Output: $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1})$: new target variables – density estimate state pair.

$\boldsymbol{x}^* \sim r(\cdot \mid \boldsymbol{x}_n)$ ▷ Propose new values for target variables.
$\hat{p}^* \sim \mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}(\cdot \mid \boldsymbol{x}^*)$ ▷ Estimate density at proposed $\boldsymbol{x}^*$.
$u \sim \mathcal{U}(\cdot \mid 0, 1)$
if $u < \frac{r(\boldsymbol{x}_n \mid \boldsymbol{x}^*)\,\hat{p}^*}{r(\boldsymbol{x}^* \mid \boldsymbol{x}_n)\,\hat{p}_n}$ **then**
    $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1}) \leftarrow (\boldsymbol{x}^*, \hat{p}^*)$ ▷ Accept proposal.
else
    $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1}) \leftarrow (\boldsymbol{x}_n, \hat{p}_n)$ ▷ Reject proposal.
return $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1})$

original source of the algorithm. Special cases of the algorithm have also been independently proposed, for example in the statistical physics literature by Kennedy and Kuti [137] and a MCMC method for doubly intractable distributions by Moller et al. [178].

The algorithm takes an intuitive form, with a very similar structure to the standard Metropolis–Hastings method (Algorithm 2) except for the ratio of densities in the accept probability calculation being replaced with a ratio of the density estimates. Importantly the stochastic density estimates are maintained as part of the chain state: if we reject a proposed update on the next iteration of the algorithm we reuse the same density estimate for the current state as in the previous iteration. This is required for the correctness of the algorithm, but also helps explain the sticking behaviour sometimes encountered with pseudo-marginal Metropolis–Hastings chains. If the density estimator distribution is heavy-tailed occasionally a estimate $\hat{p}_n$ will be sampled for the current target state $\boldsymbol{x}_n$ which is much higher than the expected value $\tilde{p}(\boldsymbol{x}_n)$. Assuming for simplicity a symmetric proposal density r is used such that the accept probability ratio in Algorithm 8 reduces to $\hat{p}^*/\hat{p}_n$, for subsequent proposed $(\boldsymbol{x}^*, \hat{p}^*)$ pairs the $\hat{p}^*$ values will typically be much smaller than the outlier $\hat{p}_n$ value and so the accept probability low. This can cause a long sequence of proposed moves being rejected until a move is proposed to an $\boldsymbol{x}^*$ where the density is similar to $\hat{p}_n$ or another atypically high density estimate is proposed [8, 87, 239].

The efficiency of the pseudo-marginal Metropolis–Hastings update depends on how noisy the density estimates are and so the choice of the number of Monte Carlo samples N in the density estimate, for example

the number of importance samples in (3.4). As N increases, the variance decreases and the algorithm becomes increasingly similar to performing standard Metropolis–Hastings updates under the (marginal) target distribution. Generally a chain will therefore mix better for larger N, with fewer sticking events. Typically however the computational cost of density estimate and so Metropolis–Hastings updates also increases with N and so there is a trade off between this improved mixing and increased per-update cost. Several theoretical studies have suggested guidelines for how to tune the parameters of the algorithm to optimise overall efficiency.

For Monte Carlo estimators formed as an average of unbiased estimators (such as the importance sampling estimator discussed above) and under an assumption that the computational cost of each density estimate scales linearly with the number of Monte Carlo samples N, it has been shown [47, 238] that it is close to optimal to choose $N = 1$. Although the variance reduction in the density estimates for larger N generally gives higher acceptance rates and improved mixing, the gain in the number effective samples in this case is usually smaller than the increased computational cost per update.

As noted in [238] in many practical settings cases the assumption of a linear increase in cost with the number of importance samples N will not be valid, particularly for small N. For example most modern *central processing units* (CPUs) have some degree of parallel compute capability through multiple cores so (assuming the parallelism can be exploited) there will usually be a non-linear increase in cost until all cores are at full utilisation: a rough guideline in this case is to use one sample per core. Another situation in which the linear cost assumption may not hold is when there is a high fixed computational overhead in each density estimate independent of the number of samples. For example if an importance distribution is used which is dependent on the target variables there may be computational operations such as matrix decompositions that can be performed once and then their cost amortised over generation of multiple importance samples.

Particle filtering estimators do not take the form of a simple Monte Carlo average of independent unbiased estimates but are instead are formed as a product of (dependent) Monte Carlo estimates [238]. The result of [238] that using $N = 1$ is close to optimal (with N now the number of particles) is therefore not applicable in this case.

Under an alternative simplifying assumption relevant to the particle filtering setting that the noise in the logarithm of the density estimator is normally distributed and independent of the value of the target variables $\boldsymbol{x}$ and that the computational cost of each density estimate scales linearly with N, it is argued in [79] that N should be chosen so as to make the standard deviation of the logarithm of the density estimator approximately equal to 1.2. In [239] a more specific case is considered of pseudo-marginal Metropolis–Hastings methods using a isotropic Gaussian random-walk Metropolis proposal $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$ and the same assumptions as [79] made of additive normal noise in the logarithm of the density estimator which is independent of $\boldsymbol{x}$ and a computational cost for each density estimate which scales linearly with N. It is shown that for target distributions on a D dimensional space which obey certain regularity assumptions as $D \to \infty$ that computational efficiency is maximised for a choice of λ and N which gives an average accept rate of approximately 0.07 and a noise standard deviation for the logarithm of the density estimator of approximately 1.8.

3.3 REPARAMETERISING THE ESTIMATOR

As a first step in considering how to apply alternative transition operators to pseudo-marginal inference problems, we define a reparameterisation of the density estimator in terms of a deterministic function of the auxiliary random variables used in computing the estimate. An equivalent reparameterisation has also been used in other work analysing the pseudo-marginal framework, for example [79].

In general the computation of a density estimate will involve sampling values from known distributions using a pseudo-random number generator and then applying a series of deterministic operations to these auxiliary random variables. Under the simplifying assumption that the estimator uses a fixed number of auxiliary random variables, we can therefore define a non-negative deterministic function $\varepsilon : X \times U \to [0, \infty)$ and a distribution R with known density $\rho = \frac{\mathrm{d}R}{\mathrm{d}\nu}$ with respect to a reference measure ν such that if $\boldsymbol{u}$ is an independent sample from R, then $\hat{p} = \varepsilon(\boldsymbol{x}, \boldsymbol{u})$ is an independent sample from $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}(\cdot \mid \boldsymbol{x})$. Here R represents the known distribution of the auxiliary variables and ε the operations performed by the remaining estimator code given values for

the target and auxiliary variables. We can use this to reparameterise (3.2) as

$$\tilde{p}(\boldsymbol{x}) = \int_U \varepsilon(\boldsymbol{x}, \boldsymbol{u})\, R(\mathrm{d}\boldsymbol{u}) = \int_U \varepsilon(\boldsymbol{x}, \boldsymbol{u})\, \rho(\boldsymbol{u})\, \nu(\mathrm{d}\boldsymbol{u}) \quad \forall \boldsymbol{x} \in X. \tag{3.6}$$

For example considering the importance-sampling density estimator for a hierarchical latent variable model defined in (3.4), if we assume the importance distribution is chosen to be a multivariate normal with density $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ then defining $\boldsymbol{u} = \left[\boldsymbol{u}^{(1)}; \cdots; \boldsymbol{u}^{(n)}\right]$ as the concatenated vector of standard normal variables used to generate the importance distribution samples, we have $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \boldsymbol{0}, \boldsymbol{I})$ and

$$\varepsilon(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{N} \sum_{n=1}^{N} \frac{\mathsf{p}_{\mathsf{x},\mathsf{y},\mathsf{z}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}} \boldsymbol{u}^{(n)} + \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}})}{\mathcal{N}\left(\boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}} \boldsymbol{u}^{(n)} + \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}} \mid \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)}, \tag{3.7}$$

where $\boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}}$ is the lower triangular Cholesky factor of $\boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}$.

Rather than defining the chain state in the pseudo-marginal Metropolis–Hastings update as the target state – density estimate pair $(\boldsymbol{x}, \hat{p})$, we can instead replace the density estimate $\hat{p}$ with the auxiliary random variables $\boldsymbol{u}$ drawn from R used to compute the estimate. As $\hat{p}$ is a deterministic function of $\boldsymbol{x}$ and $\boldsymbol{u}$ these two parameterisations are equivalent. The implementation in Algorithm 8 can be considered a practically motivated variant that avoids the $\boldsymbol{u}$ values needing to be stored in memory and in fact means they do not need to be explicitly defined in the algorithm at all.

While the formulation of the update in Algorithm 8 is the more useful for implementation purposes, showing the correctness of the update is simpler when considering the chain state as $(\boldsymbol{x}, \boldsymbol{u})$. We will briefly go through this derivation now as it provides some useful insights in to the pseudo-marginal Metropolis–Hastings algorithm that will help motivate our alternative proposed approaches.

From (3.6) we know that a distribution on $X \times U$ with density

$$\pi(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{C}\, \varepsilon(\boldsymbol{x}, \boldsymbol{u})\, \rho(\boldsymbol{u}) \tag{3.8}$$

will have the target distribution on X as its marginal distribution. Showing that the transition operator defined by Algorithm 8 leaves a distribu-

tion with density corresponding to (3.8) invariant is therefore sufficient for ensuring the correctness of the algorithm.

The transition operator corresponding to Algorithm 8 has a density

$$\mathrm{t}(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u}) = r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')\alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u}) + \delta(\boldsymbol{x} - \boldsymbol{x}')\,\delta(\boldsymbol{u} - \boldsymbol{u}') \left(1 - \int_U \int_X r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')\alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u})\,\mu(\mathrm{d}\boldsymbol{x})\nu(\mathrm{d}\boldsymbol{u})\right),$$

with the accept probability α being defined here as

$$\alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u}) = \min\left\{1, \frac{r(\boldsymbol{x} \mid \boldsymbol{x}')\varepsilon(\boldsymbol{x}', \boldsymbol{u}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})\varepsilon(\boldsymbol{x}, \boldsymbol{u})}\right\}. \tag{3.9}$$

As in Chapter 2 it is sufficient to show the non self-transition term in this transition density satisfies detailed balance with respect to the target density (3.8) as self-transitions leave any distribution invariant. We have that for $\boldsymbol{x} \neq \boldsymbol{x}'$, $\boldsymbol{u} \neq \boldsymbol{u}'$

$$\begin{aligned}
&\mathrm{t}(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u})\,\pi(\boldsymbol{x}, \boldsymbol{u}) \\
&\quad = \frac{1}{C}\, r(\boldsymbol{x}' \mid \boldsymbol{x})\,\rho(\boldsymbol{u}')\,\alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u})\,\varepsilon(\boldsymbol{x}, \boldsymbol{u})\,\rho(\boldsymbol{u}) \\
&\quad = \frac{1}{C}\,\rho(\boldsymbol{u}')\,\rho(\boldsymbol{u})\,\min\{r(\boldsymbol{x}' \mid \boldsymbol{x})\,\varepsilon(\boldsymbol{x}, \boldsymbol{u}), r(\boldsymbol{x} \mid \boldsymbol{x}')\,\varepsilon(\boldsymbol{x}', \boldsymbol{u}')\} \\
&\quad = \frac{1}{C}\, r(\boldsymbol{x} \mid \boldsymbol{x}')\,\rho(\boldsymbol{u})\,\alpha(\boldsymbol{x}, \boldsymbol{u} \mid \boldsymbol{x}', \boldsymbol{u}')\,\varepsilon(\boldsymbol{x}', \boldsymbol{u}')\,\rho(\boldsymbol{u}') \\
&\quad = \mathrm{t}(\boldsymbol{x}, \boldsymbol{u} \mid \boldsymbol{x}', \boldsymbol{u}')\,\pi(\boldsymbol{x}', \boldsymbol{u}'),
\end{aligned} \tag{3.10}$$

and so the transition operator corresponding to Algorithm 8 leaves the target distribution invariant.

We can equivalently consider Algorithm 8 as a standard Metropolis–Hastings transition operator on a target distribution with density (3.8) using a proposal $r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')$ i.e. perturbatively updating the $\boldsymbol{x}$ values and independently resampling the $\boldsymbol{u}$ values. Substituting this proposal density and target density into the standard Metropolis–Hastings accept ratio recovers the form used in the pseudo-marginal variant,

$$\frac{r(\boldsymbol{x} \mid \boldsymbol{x}')\rho(\boldsymbol{u})\frac{1}{C}\,\varepsilon(\boldsymbol{x}', \boldsymbol{u}')\,\rho(\boldsymbol{u}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')\frac{1}{C}\,\varepsilon(\boldsymbol{x}, \boldsymbol{u})\,\rho(\boldsymbol{u})} = \frac{r(\boldsymbol{x} \mid \boldsymbol{x}')\varepsilon(\boldsymbol{x}', \boldsymbol{u}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})\varepsilon(\boldsymbol{x}, \boldsymbol{u})}. \tag{3.11}$$

This formulation highlights a potential source of some of the computational issues with the pseudo-marginal Metropolis–Hastings algorithm. In high-dimensional spaces generally we would expect independent

Algorithm 9 Auxiliary pseudo-marginal framework.

Input: $(\boldsymbol{x}_n, \boldsymbol{u}_n)$: current target variables – auxiliary variables pair, T_1 : transition operator updating only auxiliary variables $\boldsymbol{u}$ and leaving distribution with density in (3.8) invariant, T_2 : transition operator updating only target variables $\boldsymbol{x}$ and leaving distribution with density in (3.8) invariant.

Output: $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$: new target state – auxiliary variables pair.

$\boldsymbol{u}_{n+1} \sim \mathsf{T}_1(\cdot \mid \boldsymbol{x}_n, \boldsymbol{u}_n)$
$\boldsymbol{x}_{n+1} \sim \mathsf{T}_2(\cdot \mid \boldsymbol{x}_n, \boldsymbol{u}_{n+1})$
return $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$

Algorithm 10 Auxiliary pseudo-marginal MI + MH.

Input: $(\boldsymbol{x}_n, \boldsymbol{u}_n)$: current target – auxiliary variables state pair, ε : estimator function for density of target distribution, ρ : density of estimator's auxiliary variable distribution, r : proposal density for updates to target state.

Output: $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$: new target – auxiliary variables state pair.

$\boldsymbol{u}^* \sim \rho(\cdot)$ ▷ T_1: MI update to auxiliary variables.
$v \sim \mathcal{U}(\cdot \mid 0, 1)$
if $v < \frac{\varepsilon(\boldsymbol{x}_n, \boldsymbol{u}^*)}{\varepsilon(\boldsymbol{x}_n, \boldsymbol{u}_n)}$ **then**
    $\boldsymbol{u}_{n+1} \leftarrow \boldsymbol{u}^*$
else
    $\boldsymbol{u}_{n+1} \leftarrow \boldsymbol{u}_n$
$\boldsymbol{x}^* \sim r(\cdot \mid \boldsymbol{x}_n)$ ▷ T_2: MH update to target variables.
$w \sim \mathcal{U}(\cdot \mid 0, 1)$
if $w < \frac{r(\boldsymbol{x}_n \mid \boldsymbol{x}^*)\, \varepsilon(\boldsymbol{x}^*, \boldsymbol{u}_{n+1})}{r(\boldsymbol{x}^* \mid \boldsymbol{x}_n)\, \varepsilon(\boldsymbol{x}_n, \boldsymbol{u}_{n+1})}$ **then**
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}^*$
else
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_n$
return $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$

resampling of a subset of the variables in a Markov chain state from their marginal distribution for a proposed Metropolis–Hastings move to perform poorly [186]. Unless the variables being independently resampled have little or no dependency on the rest of the chain state, the marginal distribution will be significantly different from the conditional distribution given the remaining variables and proposed values from the marginal will be often be highly atypical under the conditional and so have a low probability of acceptance.

3.4 AUXILIARY PSEUDO-MARGINAL METHODS

The observation that the pseudo-marginal Metropolis–Hastings update corresponds to a special case of the standard Metropolis–Hastings algorithm with independent proposed updates to the auxiliary random variables suggests the possibility of using alternative transition operators within a pseudo-marginal context. A particularly simple frame-

work is to alternate updates to the target state $\boldsymbol{x}$ given the auxiliary variables $\boldsymbol{u}$ and to the auxiliary variables $\boldsymbol{u}$ given the target state $\boldsymbol{x}$. We refer to this scheme as the *auxiliary pseudo-marginal* (APM) framework and summarise it in Algorithm 9.

A simple example of an APM method is formed by alternating *Metropolis independence* (MI) updates to the auxiliary variables given the target variables using R as the proposal distribution with *Metropolis–Hastings* (MH) updates to the target variables given the current auxiliary variables; this variant is described in Algorithm 10. Following the convention of [185] we name this method APM MI+MH for short and will in general use the form APM [T1]+[T2] to name APM methods where [T1] and [T2] are abbreviations for the types of the transition operators T_1 and T_2 respectively.

The APM MI+MH method retains the black-box nature of the original *pseudo-marginal* (PM) MH algorithm by requiring no explicit knowledge of the auxiliary random variables used in the density estimate providing we can read and write the internal state of the PRNG used by the estimator. This can be achieved for example using the `.Random.seed` attribute in `R` and the `get_state` and `set_state` methods of a `NumPy` `RandomState` object. We then only need to store the PRNG state associated with each target density estimator evaluation and restore a previous state if we wish to estimate the density at a new target state with the same set of auxiliary variables as used for a previous evaluation.

Any PM MH implementation can easily be converted in to a APM MI+MH method as the two algorithms require exactly the same input objects with the APM MI+MH method simply splitting the original single MH step into two separate propose-accept steps. The APM MI+MH method introduces some overhead by requiring two new evaluations of the target density estimator per overall update (once for the new proposed auxiliary variables and once for the new proposed target variables) compared to the single evaluation required for the PM MH algorithm.

Importantly however the updates to the target variables in APM MI+MH take the form of a standard perturbative MH update. If we use a random-walk Metropolis update then this means we can automatically tune the step size of the updates by for example appealing to theoretical results suggesting tuning the step size to achieve an average acceptance rate of 0.234 is optimal (in terms of maximising the number of effect-

ive samples per computation time) when making perturbative moves in high-dimensions [94]. The tuning can either be done in an initial warm-up phase of the chain with the samples from this initial phase not included in the final Monte Carlo estimates or by using online approaches which use vanishing adaptation [9, 116].

As discussed earlier for particle filtering estimators, under certain simplifying assumptions an alternative average acceptance rate of 0.07 has shown to be optimal for PM MH with a isotropic normal random-walk proposal in high-dimensional target distributions [239]. While this does provide a target for tuning the step-size of a standard PM MH update in the cases where it is relevant, the APM MI+MH update may often be easier to tune in practice. The 0.07 target accept rate is predicated on the variance of the density estimator having been tuned, via the number of Monte Carlo samples, such that log density estimates have a standard deviation of approximately 1.8. In general tuning the density estimator variance can be non-straightforward as in real problems it will typically vary depending on $\boldsymbol{x}$ and it is not clear which value or values to use to measure the variance at, potentially requiring an additional preliminary run to find a suitable $\boldsymbol{x}$ value to tune at. Further the non-constant estimator variances found in practice will tend to give an accept rate which varies in mean and variance across the target space. This gives a noisy signal for adaptive algorithms to tune the step-size by, potentially requiring slow adaptation for stability.

In contrast the APM MI+MH method decouples the MI auxiliary updates, which have an acceptance rate controlled by the variance of the density estimate[2] and so N, and the MH target variables updates which have an acceptance rate which is controlled by the proposal step-size λ. The two distinct accept rates provide independent signals to tune the two free parameters N and λ by, and which individually will generally be less noisy than the single combined accept rate of the PM MH update.

In density estimators which are simple Monte Carlo averages and when the cost of the estimator scales linearly with the number of Monte Carlo samples N such that the results of [238] apply and a choice of $N = 1$ is close to optimal, the additional signal provided by the accept rate

2 During the MI update to the auxiliary variables the target variables $\boldsymbol{x}$ are held fixed and a proposed new set of auxiliary variable values $\boldsymbol{u}^*$ and so density estimate $\hat{p}^* = \varepsilon(\boldsymbol{x}, \boldsymbol{u}^*)$ independently sampled. If the variance of the density estimate tends to zero the ratio of $\hat{p}^*$ to the previous estimate $\hat{p}$ which determines the accept probability of the MI step tends to one.

of the MI updates to the auxiliary variables is of less direct relevance. However as noted previously, in practice often the linear estimator cost assumption will not hold for small N, due to utilisation of parallel computation or high fixed costs. In these cases we may still wish to use the MI accept rate to adjust N so that the accept rate is above some lower threshold: although a low N (and so high estimator variance and low MI step accept probability) may be preferable in the asymptotic regime as the number of samples tends to infinity, in practical settings with finite length chains it can be that an overly high density estimator variance can lead to very low accept rates for the auxiliary variable updates such that in a finite length chain the number of updates to the auxiliary variables is very low (or even zero), potentially leading to biases in the marginal distributions of the sampled target variables.

3.5 PSEUDO-MARGINAL SLICE SAMPLING

Rather than using a MH update to the target variables, the APM framework also makes it simple to apply alternative transition operators to pseudo-marginal inference problems. A particularly appealing option are the linear and elliptical *slice sampling* (SS) algorithms discussed in Chapter 2 (Algorithms 4 and 5); when combined with MI updates to the auxiliary variables we term such methods APM MI+SS. Slice sampling algorithms automatically adapt the scale of proposed moves and so will generally require less tuning than random-walk Metropolis to achieve reasonable performance and also cope better in target distributions where the geometry of the density and so appropriate scale for proposed updates varies across the target variable space.

Slice sampling updates will always lead to a non-zero move of the target variables on each update providing for fixed values of the auxiliary variables the estimator function ϵ is a smooth function of the target variables. In such cases APM MI+SS chains will not show the 'sticking' artefacts in the traces of the target variables common to PM MH chains. As the auxiliary variables are still being updated using Metropolis independence transitions however they will still be susceptible to having proposed moves rejected, therefore the accept rate (and traces if available) of the auxiliary variables updates should also be monitored to check for convergence issues.

The APM MI+MH and APM MI+SS methods, although offering advantages over the standard PM MH method, do not address the issue that proposing new auxiliary variable values independently of their previous values can perform poorly in high dimensions. Even weak dependence between the auxiliary variables and target variables will mean that in high-dimensions the typical set of the marginal auxiliary variable distribution R used as the proposal distribution will differ significantly from the typical set of the conditional distribution on the auxiliary variables given the target variables values. This conditional distribution is used to decide acceptances and so the accept probability of proposed updates to the auxiliary variables will be small.

One way of increasing the probability of proposed updates to the auxiliary variables from R being accepted is to increase the number of Monte Carlo samples N used in the estimator. For concreteness we will assume we use the importance sampling estimator (3.4) for inference in a hierarchical latent variable model with a multivariate normal importance distribution $q(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}\right)$ (in general $\boldsymbol{\mu}$ and $\boldsymbol{L}$ will depend on $\boldsymbol{x}$ and $\boldsymbol{y}$ but we leave this dependence implicit for notational simplicity). Using the reparameterisation of the estimator in (3.7), the target density (3.8) on the auxiliary and target variables takes the form

$$\pi(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{NC} \sum_{n=1}^{N} \frac{\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu})}{\mathcal{N}\left(\boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu} \mid \boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}\right)} \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right). \tag{3.12}$$

Using that $C = \mathrm{p}_{\mathbf{y}}(\boldsymbol{y})$ and $\mathcal{N}\left(\boldsymbol{L}\boldsymbol{u} + \boldsymbol{\mu} \mid \boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}\right) = |\boldsymbol{L}|^{-1}\mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$ this can be manipulated into the form

$$\pi(\boldsymbol{x}, \boldsymbol{u}) = \frac{\mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y})}{N|\boldsymbol{L}|^{-1}} \sum_{n=1}^{N} \frac{\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{y})}{\mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right)} \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right).$$

By separating out the terms involving a single auxiliary variable sample $\boldsymbol{u}^{(m)}$, the conditional density on $\boldsymbol{u}^{(m)}$ given the remaining auxiliary variable samples can be shown to take the form of a mixture

$$\begin{aligned} &\pi\left(\boldsymbol{u}^{(m)} \mid \boldsymbol{x}, \{\boldsymbol{u}^{(n)}\}_{n \neq m}\right) \propto \\ &\qquad \mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{L}\boldsymbol{u}^{(m)} + \boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{y}) + w\left(\boldsymbol{x}, \{\boldsymbol{u}^{(n)}\}_{n \neq m}\right) \mathcal{N}\left(\boldsymbol{u}^{(m)} \mid \mathbf{0}, \mathbf{I}\right) \end{aligned} \tag{3.13}$$

with $w\left(\boldsymbol{x}, \{\boldsymbol{u}^{(n)}\}_{n \neq m}\right) = \sum_{n \neq m} \left(\frac{\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{y})}{\mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right)} \right)$.

The sum of the importance weights in w will grow with N (for independent $\boldsymbol{u}^{(n)} \sim \mathcal{N}(\cdot \mid \mathbf{0}, \mathbf{I})$ $\forall n \neq m$ it would have an expected value $(N-1)|\boldsymbol{L}|$) and so for large N the second term in the mixture will increasingly dominate and the conditional density on $\boldsymbol{u}^{(m)}$ will tend to $\mathcal{N}\left(\boldsymbol{u}^{(m)} \mid \mathbf{0}, \mathbf{I}\right)$ and independence from $\boldsymbol{x}$. Therefore as we increase N we would expect independently re-sampling the auxiliary variables from R in a MI step to have an increasing probability of acceptance.

Although non-rigorous, this analysis also gives an intuition to why the pseudo-marginal method can provide an advantage over directly performing MCMC in the joint space of $\mathbf{x}$ and $\mathbf{z}$ in hierarchical latent variable models: if the conditional density on the local latent variables $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ has a challenging geometry, for example it is multimodal, then MCMC transition operators based on local moves working in the $(\mathbf{x}, \mathbf{z})$ space are likely to mix poorly for example by getting stuck in a single mode or only being able to make very small moves per update. If we instead reparameterise in terms of a set of auxiliary variables $\{\mathbf{u}^{(n)}\}_{n=1}^{N}$, then we are able to maintain the correct marginal distribution on the target variables $\mathbf{x}$ while working with a distribution on an extended space which becomes increasingly tractable to sample from as we increase N, with the individual auxiliary variable samples $\mathbf{u}^{(n)}$ individually having conditional densities which only weakly depend on $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$.

While we can always increase N to the point where independently proposing updates to the auxiliary variables from R will have a reasonable probability of acceptance, this will also increase the computational expense of each update. Rather than proposing new values for the auxiliary variables independently of their previous values, an obvious idea is to take a more standard MCMC approach by using local perturbative updates which leave the overall target distribution (3.8) invariant. For $N = 1$ this equivalent to performing MCMC directly in a non-centred reparameterisation [203] of the joint $(\mathbf{x}, \mathbf{z})$ space by alternating updates of the target and auxiliary (latent) variables. For $N > 1$ we potentially gain from the conditional distribution on the auxiliary variables being easier for MCMC algorithms to explore though with an increased computational cost per update.

One option is to use a MH method such as random-walk Metropolis to update the auxiliary variables. While with a well tuned proposal distribution this approach can work well, it adds further tuning burden to the user which might outweigh any efficiency gains. For problems in

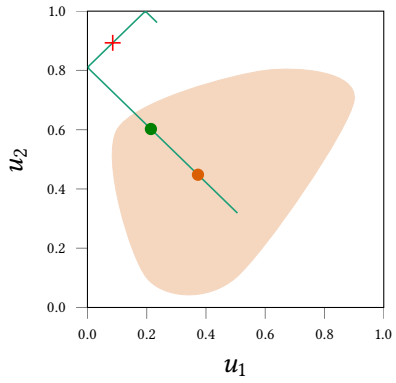


Figure 3.2.: Illustration of reflective linear slice sampling in two dimensions. The orange circular marker represents the current state and the light filled orange region the density slice at the sampled slice height (see explanation of Algorithm 4 in Chapter 2 for details). A random slice line direction vector $\boldsymbol{v}$ is sampled from some distribution as in Algorithm 4, for example with elements independently sampled from $\mathcal{N}(0, 1)$ or $\mathcal{U}(-1, 1)$. This defines a line passing through the current point (green-blue line in Figure), with importantly in this case the line *reflected* at the boundaries of the hypercube (square in this two-dimensional case). An initial bracket of a specified width is randomly placed around the current point on the line. The algorithm then proceeds as in the standard linear slice sampling algorithm by repeatedly proposing a point in the current bracket and accepting if on the slice (in orange region, for example the green circle) or rejecting and shrinking the bracket if off the slice (outside orange region, for example the red cross).

which we can reparameterise the density estimator as a deterministic function of a vector of standard normal draws so that $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$, an appealing option is to use elliptical slice sampling (Algorithm 5) to update the auxiliary variables. The elliptical slice sampling algorithm has no free parameters for the user to choose and initially proposes moves to points nearly independent of the current values [183] so if the conditional distribution of the auxiliary variables is well approximated by the normal marginal distribution R, elliptical slice sampling should perform similarly to a MI update. Using the adaptive bracket shrinking procedure discussed in Chapter 2 the elliptical slice sampler is also however able to exponentially back-off to smaller proposed moves around the current state if the bold initial proposal is not on the slice. Providing for fixed values of the target variables the target density (3.8) is a smooth function of the auxiliary variables, the slice sampling procedure will always lead to a non-zero update of the auxiliary variables.

If the auxiliary variables are instead marginally distributed as independent standard uniform variables[3] i.e. $\rho(\boldsymbol{u}) = \prod_i \mathcal{U}(u_i \mid 0, 1)$, one option is to reparameterise these as independent standard normal variables which are then mapped through the normal CDF. We can then run elliptical slice sampling in the transformed normal space. In general evaluation of the normal CDF is a relatively expensive operation and the distortion induced by pushing through the CDF may in some case map a distribution with a density with relatively simple geometry in the uniform space to a density with more complex geometry in the normal space. An alternative is to therefore perform linear slice sampling directly in the uniform auxiliary variable space.

A small subtlety is that the target distribution on the auxiliary variables will only have support on the unit hypercube in this case. We can adjust Algorithm 4 for this setting by replacing Line 8 in the Algorithm with $\boldsymbol{x}^* \leftarrow \textsc{Reflect}(\boldsymbol{x}_n + \lambda \boldsymbol{v})$ (and the likewise the corresponding equivalent expressions in the step-out routine in Lines 17 and 20), where the Reflect function is defined elementwise by

---

function Reflect(u)
  $v \leftarrow u \bmod 2$
  return $v\, \mathbb{1}_{[0,1)}(v) + (2 - v)\, \mathbb{1}_{[1,2)}(v)$

The reflection transformation defined by this function has a unit Jacobian determinant and maintains reversibility and so the reflective slice sampling transition leaves the uniform distribution on the slice invariant. An illustrative schematic of a reflective linear slice sampling transition in two dimension is shown in Figure 3.2. Reflective variants of slice sampling are discussed in [190] and [80].

3.6 NUMERICAL EXPERIMENTS

We will now discuss the results of two empirical studies in to the performance of the proposed auxiliary pseudo-marginal methods. Further experiments applying some of the proposed methods in a simulator model inference setting will be discussed in Chapter 4.

3 The auxiliary variables in this case could for example represent all the standard uniform draws from the PRNG that are used to generate random variables in the estimator using the rejection and transform sampling routines discussed in Chapter 2.

3.6.1 Gaussian latent variable model

As a first numerical example we consider inference in a hierarchical Gaussian latent variable model. In particular we assume a model with the factorisation structure shown in Figure 3.1 with

$$\begin{aligned} \mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \mathbf{0}, \mathbf{I}), \quad \mathrm{p}_{\mathbf{z}|\mathbf{x}}(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{z}^{(m)} \mid \boldsymbol{x}, \sigma^2 \mathbf{I}\right), \\ \text{and} \quad \mathrm{p}_{\mathbf{y}|\mathbf{x},\mathbf{z}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) = \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{y}^{(m)} \mid \boldsymbol{z}^{(m)}, \epsilon^2 \mathbf{I}\right). \end{aligned} \tag{3.14}$$

We used $\sigma = 1$ and $\epsilon = 2$ in the experiments and generate $M = 10$ simulated observed values $\{\boldsymbol{y}^{(m)}\}_{m=1}^{M}$, each of dimensionality $D = 10$. We assume we wish to infer plausible values for the D-dimensional vector $\mathbf{x}$ consistent with the observed $\mathbf{y}$ and so the target distribution for inference has density $p(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y})$. Here because of the self-conjugacy of the Gaussian distribution, the marginalisation over the local latent variables $\mathbf{z}$ can be performed analytically to give

$$\mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{x} \,\middle|\, \frac{1}{M + \sigma^2 + \epsilon^2} \sum_{m=1}^{M} \boldsymbol{y}^{(m)}, \frac{\sigma^2 + \epsilon^2}{N + \sigma^2 + \epsilon^2} \mathbf{I}\right). \tag{3.15}$$

Although exact inference is therefore tractable in this case, we apply pseudo-marginal MCMC methods to allow us to study the performance of the methods in a case where we have a ground-truth for the inferences to check convergence against.

We use an importance sampling estimator of the form given in (3.4) using $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ as the importance distribution i.e.

$$q\left(\{\boldsymbol{z}^{(m)}\}_{m=1}^{M} \mid \boldsymbol{x}, \{\boldsymbol{y}^{(m)}\}_{m=1}^{M}\right) = \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{z}^{(m)} \mid \boldsymbol{x}, \sigma^2 \mathbf{I}\right). \tag{3.16}$$

As this importance distribution does not take in to account the observed values $\{\boldsymbol{y}^{(m)}\}_{m=1}^{M}$ it results in a relatively high-variance importance sampling estimator of the target density with a variance which depends on the values of the target variables $\boldsymbol{x}$. Therefore although exact inference in this example is tractable and the target distribution has a simple isotropic geometry, in this pseudo-marginal formulation the model still has some of the key features which can pose challenges to pseudo-marginal inference algorithms.

For the auxiliary pseudo-marginal methods, we use a reparameterisation of the estimator equivalent to (3.7), using the standard normal variables used to generate samples from $\mathsf{P}_{\mathbf{z}|\mathbf{x}}$ as the auxiliary variables, resulting in an auxiliary variable marginal distribution with density $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$ and an estimator function ε

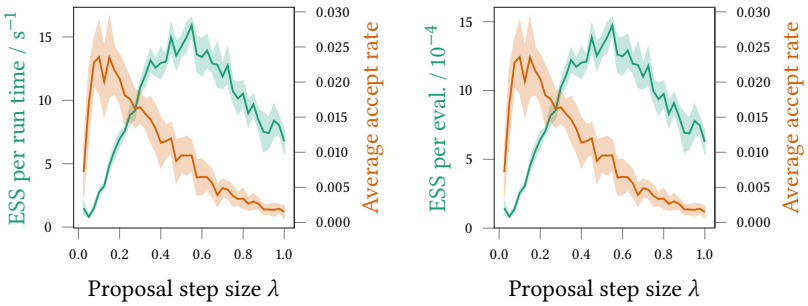
$$\varepsilon(\boldsymbol{x}, \boldsymbol{u}) = \frac{\mathcal{N}(\boldsymbol{x} \mid \mathbf{0}, \mathbf{I})}{N} \sum_{n=1}^{N} \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{y}^{(m)} \mid \sigma \boldsymbol{u}^{(n,m)} + \boldsymbol{x}, \epsilon^2 \mathbf{I}\right), \tag{3.17}$$

with $\boldsymbol{u} = \left[\boldsymbol{u}^{(1,1)}; \ldots \boldsymbol{u}^{(1,M)}; \boldsymbol{u}^{(2,1)} \ldots \boldsymbol{u}^{(N,M)}\right] \in \mathbb{R}^{NM}$.

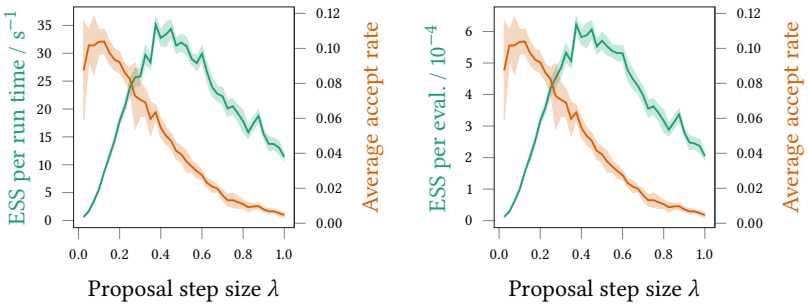
3.6.1.1 *Pseudo-marginal Metropolis–Hastings*

We first applied the PM MH algorithm to perform inference in this model, using an isotropic normal random-walk proposal distribution for the updates to the target variables, i.e. $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$. To assess the impact of the choice of the proposal step size parameter λ on sampling efficiency, we ran 10 independent chains initialised from the prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for λ values on a equispaced grid of 40 points between 0.025 and 1, running each chain for 50 000 iterations. We ran all experiments for the cases of density estimators using $N = 1$, $N = 8$ and $N = 32$ importance samples, with the logarithm of the density estimate at the value of the target variables $\boldsymbol{x}$ used to generate the observed values $\boldsymbol{y}$ having standard deviation 3.6 for $N = 1$, 1.8 for $N = 8$ and 1.2 for $N = 32$.

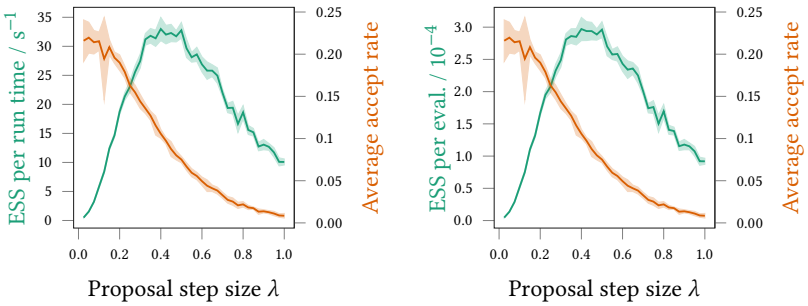
For all combinations of N and λ we estimated the *effective sample size* (ESS) (as defined for a geometrically ergodic Markov chain in Equation 2.25 of Chapter 2) for the posterior mean of each chain using the `R CODA` package [211]. We then derived two overall measures of computational efficiency from these ESS estimates by normalising either by the number of joint density evaluations in the density estimator (which increases per iteration with the number of importance samples N) or the wall clock run time of the chains in seconds. The results are plotted in Figure 3.3. Each pair of plots in a row corresponds to a particular number of importance samples. In each row the left column shows the ESSs normalised by the run time and the right column by the number of density evaluations, with the green curves representing the mean of these values across all the chains and the filled region plus and minus one standard deviation (note the standard deviation rather than standard error of mean was used as in some of the plots the standard error



(a) $N = 1$



(b) $N = 8$



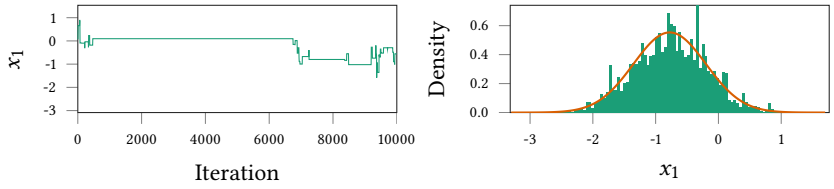
(c) $N = 32$

Figure 3.3.: Results of Gaussian latent variable model PM MH chains. The plots in each row show both the estimated ESS normalised by either the compute time (green, left column) or number of density estimator evaluations (green, right column) and average acceptance rate of MH updates (orange), versus the isotropic random-walk proposal step-size λ for the MH updates to the target variables. The top row shows the case for a density estimator using $N = 1$ importance sample, middle row for $N = 8$ and the bottom row for $N = 32$. In all cases the curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation.

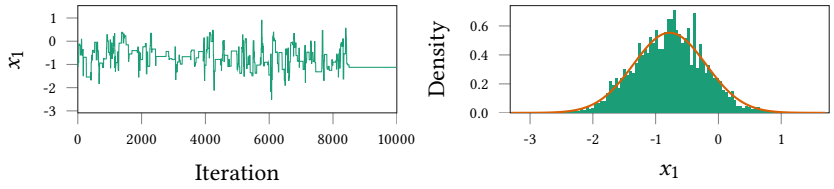
was too small to be easily visible). On each axis as well as the normalised ESS, the average accept rate across the chains is also plotted in orange (with scale shown on the right vertical axis), with again the curves showing the mean value across the chains and the filled regions plus and minus one standard deviation.

The results of [238] suggest that asymptotically using $N = 1$ importance sample should be optimal in this case assuming a linear increase in the cost of generating each sample with N. The measure of computational efficiency used in [238] therefore most closely corresponds to the estimated ESS normalised by the number of density evaluations (which scales linearly with N), and indeed on this measure (green curves in right column of Figure 3.3) we see that the chains using $N = 1$ outperforms the $N = 8$ and $N = 32$ cases.

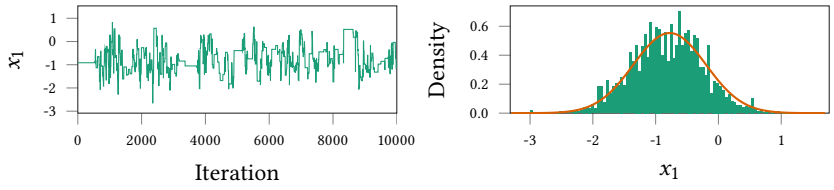
The plots in Figure 3.3a and to a lesser extent 3.3b show a spurious appearing behaviour for the smallest step sizes that the accept rate (orange curve) seems to initially *increase* as the step size is made larger, contrary to what we would reasonably expect. This anomaly can be ascribed to a lack of convergence in the chains with small step sizes due to the sticking behaviour discussed previously. For the $N = 1$ case, because of the relatively high density estimator variance, the chains are prone to getting stuck for thousands of iterations at a time. The estimator variance is dependent on the values of the target variables $\boldsymbol{x}$ and generally seems to be lower for values typical under the posterior. As the chains are initialised from the prior, they tend to therefore initialised in regions in which the estimator variance is higher than typical often leading to long sticking periods near the start of the chain. For the chains with small step sizes the chain is slower to 'warm-up' and converge towards the typical set of the posterior distribution on the target variables and so this propensity for sticking during the initial warm-up period has a larger effect, leading to some chains rejecting nearly all updates even though the step size is very small. This counter intuitive behaviour of the empirical accept rates for small step sizes and general noisiness of the dependency of the accept rate on the step size, particularly for small N, highlights the difficulty of tuning the PM MH updates: the low accept rates here would intuitively indicate the step size should be made smaller but in some cases this would actually make the measured accept rate even worse.



(a) $N = 1, \lambda = 0.550$



(b) $N = 8, \lambda = 0.375$



(c) $N = 32, \lambda = 0.400$

Figure 3.4.: Example traces and histograms of PM MH chains in Gaussian latent variable model inference task. In each row a trace of the sampled values for the x_1 target variable in the last 10000 iterations of a PM MH Markov chain using the optimal step size for the relevant N found from Figure 3.3 is shown in the left plot, while the right plot shows a histogram of the samples values from the full chain (green filled region) against the exact marginal posterior density (orange curve). In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the increased number of density evaluations for higher N, so the $N = 1$ plot is of a chain of 3.2×10^6 samples, the $N = 8$ plot is of 8×10^5 samples as the $N = 32$ plot of 1×10^5 samples.

Figure 3.4 shows example traces of the x_1 variable samples for chains using density estimates with $N = 1$, $N = 8$ and $N = 32$ importance samples. In each case the step size suggested to be optimal by the results in Figure 3.3 (in terms of effective samples per density evaluation) has been used, and the traces shown are the last 10 000 iterations of a longer run. Also shown are histogram estimates of the posterior marginal densities on the x_1 variable using the sampled states from the whole chain, with the total number of samples in each chain adjusted to account for the extra computational cost of using more importance samples, along with a curve showing the true posterior marginal density. The propensity of the chains to stick is clearly visible in the traces particularly for the $N = 1$ case, with long series of thousands of rejected updates at a time. This is also reflected in the noisiness of the marginal density estimates with spurious peaks appearing around the states where the chain gets stuck.

When comparing instead in terms of the estimated ESS normalised by actual chain run time (green curves in left column of Figure 3.3) the results no longer suggest $N = 1$ is optimal, with the $N = 8$ and $N = 32$ cases both performing better on this measure for all step sizes. This can be explained by the non-linear scaling of the computational cost per update with the number of importance samples due to both overhead from the implementation of the rest of the operations in the transition and only partial utilisation of the parallel compute resource available (the CPU used in the experiments had 4 cores). Although the increase in efficiency per actual run time for $N \neq 1$ is implementation and device dependent, a possibly stronger reason suggested by the results to use $N > 1$ is the less brittle nature of the chains behaviour, with the very low accept rates in the $N = 1$ case needing long runs to smooth out the effects of long series of rejections.

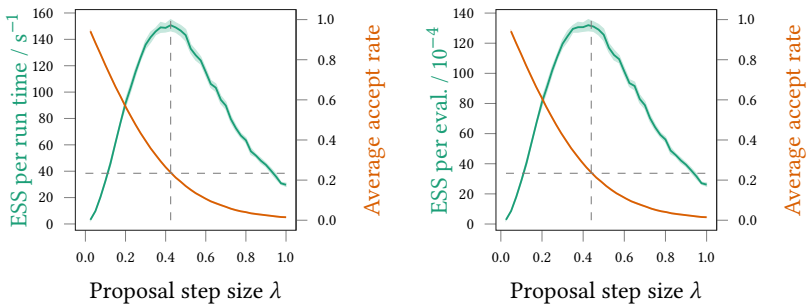
The results in Figure 3.3 also highlight the difficulty of tuning the proposal step size when using a random-walk Metropolis PM MH update. The optimal step size appears to possibly weakly depend on the number of importance samples used (though the noisiness of the curves make this difficult to determine). Further there is not a clear relationship between the average accept rate and optimal step size. As previously stated the result of [94] that a step size giving an accept rate of 0.234 is close to optimal is not applicable to the update here, with this confirmed empirically by the fact that only the chains with the smallest

step sizes for the $N = 32$ case are even able to achieve an accept rate close to 0.234 (and are far from optimal in efficiency). In practice we therefore we do not have an obvious signal to tune the step size by beyond running pilot chains and computing ESS estimates which is likely to add too much cost to justify any gain in efficiency from choosing a better step size for subsequent chains.

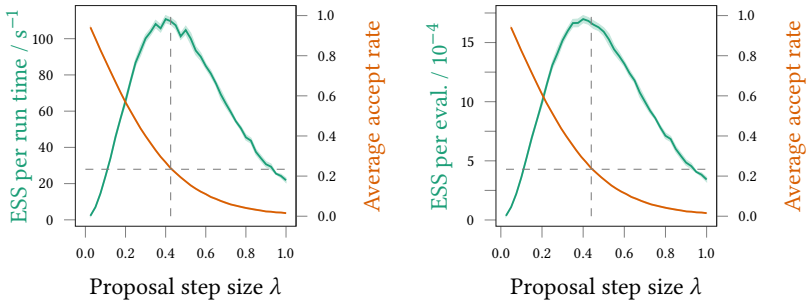
3.6.1.2 *Splitting the update*

We next applied the proposed APM MI+MH algorithm to perform inference in the Gaussian latent variable model. From an implementation perspective this simply requires the original combined update to the auxiliary and target variables in the PM MH case to be split in to separate MI updates of the auxiliary variables given fixed target variables and MH updates of the target variables for fixed auxiliary variables. Despite the seemingly minor change to the form of the update, the difference in the results is dramatic.

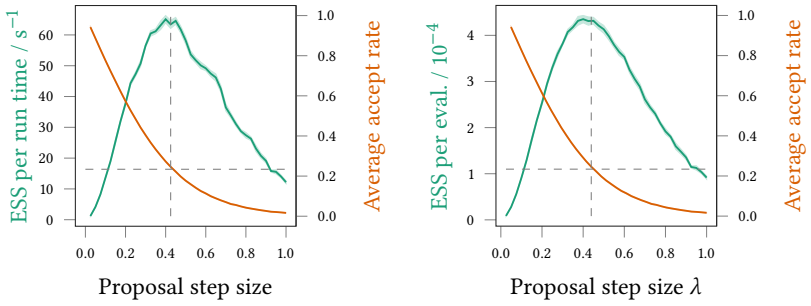
Figure 3.5 shows plots of results of an equivalent series of experiments as used to produce Figure 3.3. In this case the horizontal axes on the plots shows the proposal step size for the MH updates to the target variables which as previously use an random-walk Metropolis proposal $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$. Again 10 independent chains initialised from the prior were run for each step size λ and number of importance samples N pair, with in this case shorter chains of 20 000 iterations used (with the known posterior means and standard deviations used to establish that the chains had adequately converged). Again the estimated ESSs for estimates of the posterior mean were computed for each chain, with the green curves in the left column of plots in Figure 3.5 showing the mean of these estimated ESSs across the chains normalised by the total wall clock run time for the chain, and the right column the ESSs normalised by the number of joint density evaluations. The average accept rate shown by the orange curves in Figure 3.5 is for the MH update to the target variables. A separate average accept rate was recorded for the MI updates to the auxiliary variables and was found to not show any obvious dependency on the target variable proposal step size, with an average accept rate of approximately 0.025 for chains with $N = 1$ importance sample in the density estimates, an average accept rate of 0.11 for chains with $N = 8$ and an average accept rate of 0.23 for chains using $N = 32$.



(a) $N = 1$



(b) $N = 8$



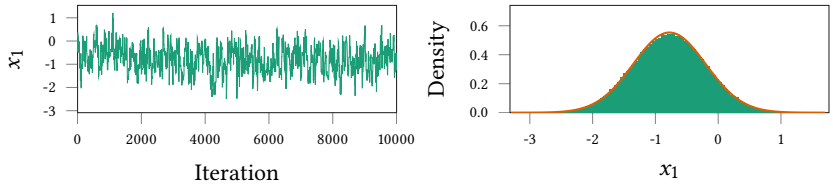
(c) $N = 32$

Figure 3.5.: Results of Gaussian latent variable model APM MI+MH chains. The plots in each row show both the estimated ESS normalised by either the total compute time (green, left column) or number of density estimator evaluations (green, right column) and average acceptance rate for the MH updates (orange), versus the isotropic random-walk proposal step-size for the MH updates to the target variables. The top row shows the case for a density estimator using $N = 1$ importance sample, middle row for $N = 8$ and the bottom row for $N = 32$. In all cases the curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation. The horizontal dashed lines indicate an accept rate of 0.234 and the vertical dashed lines the corresponding proposal step size.

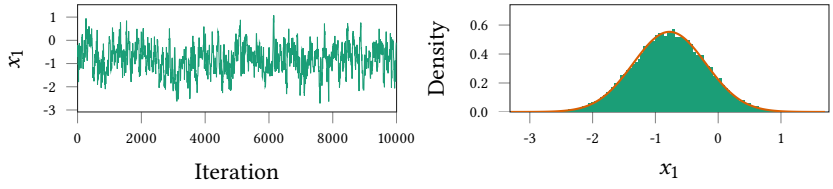
On both the time and density evaluation normalised measures of efficiency the APM MI+MH chains perform significantly better than the PM MH chains. The peak ESS per density evaluation value for the $N = 1$ and $\lambda = 0.425$ case is around a factor of ten higher than the corresponding peak value for the PM MH chains, while in terms of the ESS per run time metric the best APM MI+MH chains show around a factor four improvement over the PM MH chains. While other experiments have suggested this level of improvement is atypical, it seems reasonable to conclude that at least in some cases the extra overhead introduced by requiring two density estimates per overall update is worthwhile.

More importantly perhaps the curves in Figure 3.5 suggest the APM MI+MH update is significantly easier to tune. The average accept rate of the MH updates to the target variables shows the expected monotonically decreasing behaviour as the step size is increased and in general the measured accept rates are significantly less noisy than the corresponding accept rates for the PM MH updates. The horizontal dashed lines in Figure 3.5 indicate an average accept rate of 0.234 with the corresponding vertical dashed lines showing the estimated proposal step size corresponding to this acceptance rate. As can be seen by both the compute time and density evaluation normalised measures of sampling efficiency, the chains with proposal step sizes giving accept rates near to 0.234 are close to optimal in efficiency, suggesting the theoretical result of [94] holds here as suggested earlier. Further in this model at least, this relationship seems to hold for a range of different numbers of importance samples and so density estimator variances. This suggests it is valid to use standard adaptive approaches which use the average accept rate as a control signal to tune the step size of the target variables MH proposal distribution when using the APM MI+MH update.

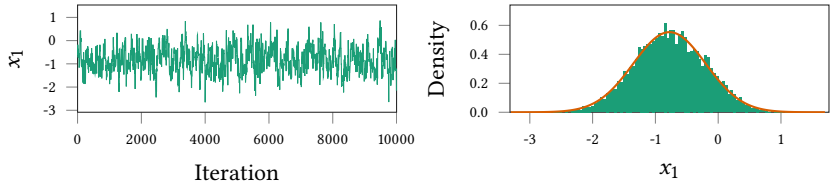
In further contrast to the PM MH results, the results for the APM MI+MH chains seem to unambiguously support using $N = 1$ importance sample. On both the computation time and density evaluation normalised measures of efficiency, the chains using one importance sample dominate over the $N = 8$ and $N = 32$ cases. The APM MI+MH chains using a single importance sample do not show the pathological sticking behaviour evident in the PM MH chains, with an example trace shown for a step size of $\lambda = 0.425$ (which Figure 3.5 suggests is close to optimal) in Figure 3.6a. Unlike the $N = 1$ PM MH trace, over the 10 000 iterations shown the APM MI+MH seems to mix well with no obvious sticking peri-



(a) $N = 1, \lambda = 0.425$



(b) $N = 8, \lambda = 0.425$



(c) $N = 32, \lambda = 0.425$

Figure 3.6.: Example traces and empirical histograms of APM MI+MH chains in Gaussian latent variable model inference task. In each row a trace of the sampled values for the x_1 target variable in the last 10 000 iterations of a APM MI+MH Markov chain using the optimal step size for the relevant N found from Figure 3.5 is shown in the left plot, while the right plot shows an empirical histogram of the samples values from the full chain (green filled region) against the exact marginal posterior density (orange curve). In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the increased number of density evaluations for higher N and to account for the 2 evaluations per update compared to PM MH to allow fair comparison with Figure 3.4, so the $N = 1$ plot is of a chain of 1.6×10^6 samples, the $N = 8$ plot is of 2×10^5 samples as the $N = 32$ plot of 5×10^4 samples.

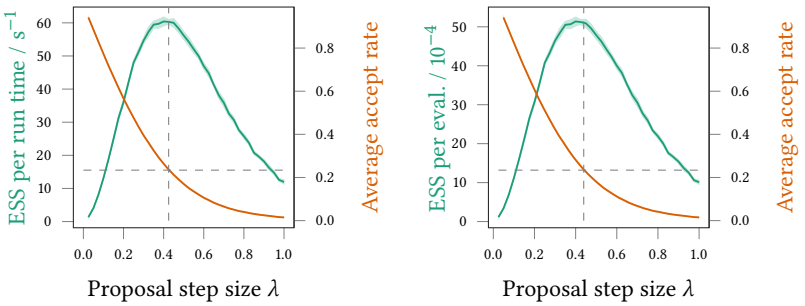


Figure 3.7.: Results of Gaussian latent variable model APM SS+MH chains (using $N = 1$ importance sample). The plots in each row show both the estimated ESS normalised by either the total compute time (green, left column) or number of density estimator evaluations (green, right column) and average acceptance rate for the MH updates (orange), versus the isotropic random-walk proposal step-size λ for the MH updates to the target variables. The curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation.

ods. The example traces for the $N = 8$ and $N = 32$ APM MI+MH chains in Figure 3.6 also seem to follow this pattern. Comparing the posterior marginal density estimates for the x_1 target variable shown in the right column of Figure 3.6, the marginal estimates for the $N = 1$ case appear the smoothest, almost indistinguishable from the curve of the true density (to normalise for the additional density evaluations required for the $N = 8$ and $N = 32$ cases the number of samples in the chains used to produce the histograms was reduced accordingly). This again suggests that any improvement in mixing by using $N > 1$ in this case is outweighed by the cost of the additional density evaluations.

3.6.1.3 Slice sampling the auxiliary variables

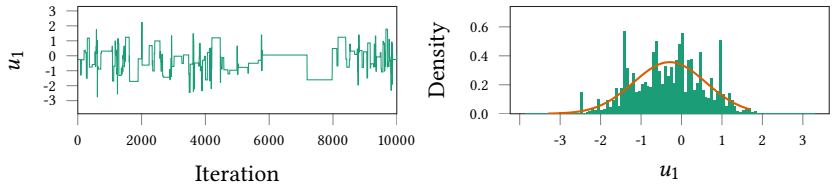
For the APM MI+MH chains discussed in the previous subsection, when using $N = 1$ importance sample the MI updates to the auxiliary variables were only accepted 2.5% of the time. Although this did not appear to impede convergence of the chain in this example, more generally low accept rates for the MI updates to the auxiliary variables may be a cause for concern as in shorter chains this will mean the auxiliary variables are only updated a small number of times across the chain. As convergence of the distribution on the target variables in the chain state to their marginal target distribution is reliant on the distribution of the auxiliary variables in the chain state also converging, very infrequent updates of the auxiliary variables could potentially lead to diffi-

cult to diagnose convergence issues in the chains. Although increasing the number of importance samples in the estimator can increase the MI step accept rate as seen in the APM MI+MH experiments above, there is a diminishing returns behaviour to the increase of acceptance rate with the number of samples.

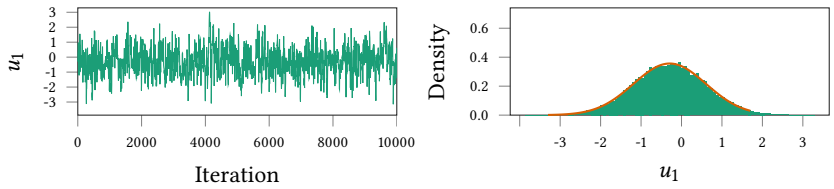
The earlier suggestion to use perturbative updates to the auxiliary variables provides an alternative approach to improve the auxiliary variable mixing. We test specifically here the proposal to use elliptical slice sampling updates to the auxiliary variables, which is a natural choice in this case due to their Gaussian marginal distribution. We use the same MH update to the target variables as in the experiments in the previous two subsections, and again measure sampling efficiency for different proposal step sizes λ. We only run chains using a estimator taking $N = 1$ importance sample in this case as we are mainly interested in using perturbative updates to the auxiliary variables as an alternative to having to increase the number of importance samples to achieve reasonable acceptance rates for MI updates to the auxiliary variables.

Results for an equivalent series of experiments as discussed in the previous two subsections for APM SS+MH chains using elliptical slice sampling updates to the auxiliary variables are shown in Figure 3.7. In this case as the MI updates to the auxiliary variables for the $N = 1$ case seemed to be sufficient to achieve convergence, the elliptical slice sampling updates do not seem to significantly improve mixing of the target variables. The extra overhead from the adaptive slice sampling updates means overall computational efficiency decreases by roughly a factor of two across all proposal step sizes λ compared to the corresponding APM MI+MH results for $N = 1$ in Figure 3.5a, with this consistent across both the density evaluation normalised efficiency metric and run time normalised measure.

Although the slice sampling updates do not help improve the sampling of the target variables here, the resulting auxiliary variables samples are much more representative of their true posterior distribution (which again can be found analytically) compared to when using MI updates. Figure 3.8 shows traces and histograms of one of the auxiliary variables for chains computed using both the APM MI+MH and APM SS+MH updates. The slice sampling updates give significantly better mixing of the auxiliary variables than the MI updates which due to the low accept rate remain fixed for many iterations. Although in this case this

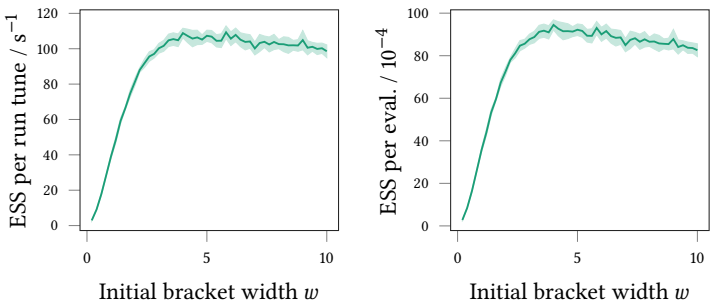


(a) APM MI+MH $N = 1, \lambda = 0.425$

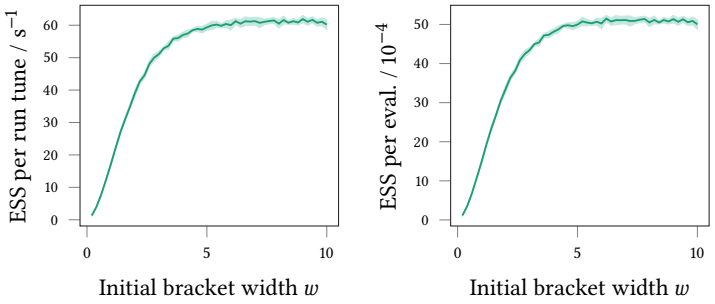


(b) APM SS+MH $N = 1, \lambda = 0.425$

Figure 3.8.: Example traces and histograms of an auxiliary variable in APM MI+MH and APM SS+MH chains in Gaussian latent variable model inference task. In each row a trace of the sampled values for the u_1 auxiliary variable in the last 10000 iterations of a Markov chain using the optimal step size $\lambda = 0.425$ and $N = 1$ is shown in the left plot, while the right plot shows a histogram of the sample values from the full chain (green filled region) against the exact marginal posterior density (orange curve). In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the roughly two times increase in the number of density evaluations per sample for the APM SS+MH updates compared to APM MI+MH, so the APM MI+MH plot is of a chain of 10^5 samples and the APM SS+MH plot is of 5×10^4 samples.



(a) APM MI+SS ($N = 1$)



(b) APM SS+SS ($N = 1$)

Figure 3.9.: Results of Gaussian latent variable model APM chains using linear SS to update target variables and either MI updates to auxiliary variables (top row) or elliptical SS updates (bottom row). The plots in each row show both the estimated ESS normalised by either the total compute time (left column) or number of density estimator evaluations (right column), versus the slice sampler initial bracket width for the linear SS updates to the target variables. The curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation.

does seem to translate to an obvious improvement in convergence of the target variables, more generally a factor two increase in run time for the added robustness of significantly improved mixing of the auxiliary variables seems like it will often be a worthwhile trade-off to avoid possible convergence issues.

3.6.1.4 Slice sampling the target variables

As a final set of experiments for this model, we explored the use of slice sampling updates to the target variables with an auxiliary pseudo-marginal framework, specifically linear slice sampling updates along an isotropically sampled direction. To test the claim that the efficiency of slice sampling updates is less sensitive to the choice of the free initial bracket width parameter w of the algorithm than random-walk Metropolis updates are to the choice of the proposal step size parameter λ, we

ran a similar series of experiments as in the previous sub-sections to analyse the dependency of sampling efficiency on λ by instead varying the initial bracket width w.

For each of 50 initial bracket width w values on an equispaced grid between 0.2 and 10, we ran 10 independent APM MI+SS and APM SS+SS chains (with elliptical slice sampling updates to the auxiliary variables) initialised from the prior of 20 000 iterations each. As previously for each set of chains for a particular w value we computed the estimated ESSs of the chains for the estimate of the posterior mean and normalised this value by both the total wall-clock run time in seconds and total number of joint density evaluations to give two measures of overall efficiency. The means and one standard deviation intervals of these values across the 10 chains are shown for the APM MI+SS chains in Figure 3.9a and for the APM SS+SS chains in Figure 3.9b. In all cases zero linear step-out iterations were used in the slice sampling updates to the target variables.

The peak efficiency achieved by the APM MI+SS chains on this problem is less than that for the best APM MI+MH chains by a factor of around 1.5 on both measures of efficiency. As the slice sampling updates do more work per iteration than the MH updates this is not unexpected as a well-tuned MH update will generally perform better than a slice sampling update when the geometry of the target distribution is simple (as is the case here). Importantly however the slice sampling updates maintain a computational efficiency that is within around 10% of the optimal efficiency across a wide range of initial bracket width values, with values from $w = 2$ to $w = 10$ all seeming to perform reasonably well in this problem. This is in contrast to the much tighter range of proposal step size values required to get good performance with MH updates to the target variables. The exponential back-off to smaller proposals provided by the adaptive bracket shrinking procedure in the slice sampling transition means that the penalty for using an overly large scale parameter w is much less severe than the corresponding situation for using an overly large λ in a MH update.

The APM SS+SS chains have around half the sample efficiency of the APM MI+SS chains for the same bracket width w due to the additional computational cost of the elliptical slice sampling updates to the auxiliary variables. As the cost of the elliptical SS updates to the auxiliary variables dominates over that of linear SS updates to the target variables here, the

APM SS+SS chains have a similar per sample computational cost as the APM SS+MH chains. The extra overhead of the SS rather than MH updates to the target variables is minimal and so the peak sample efficiencies of these two methods are similar. However the APM SS+SS chains remain close to this peak efficiency over a much wider range of scale parameter settings (bracket width w or proposal step-size λ).

3.6.2 Gaussian process probit regression

As a second experiment we consider a more challenging problem of inferring the parameters of the covariance function of a latent Gaussian process used to model the relationship between pairs of feature vectors and binary target outputs. The use of PM MH for this task was considered in [87] and shown to give significant improvements over competing MCMC methods.

As an example data set we used the Wisconsin breast cancer prediction data set [162] from the UCI machine learning dataset repository [158] as also used for experiments in [87]. The data $\{\boldsymbol{d}^{(m)}, y_m\}_{m=1}^{M}$ consists of pairs of vectors $\boldsymbol{d}^{(m)}$ of $K = 9$ integer descriptors of individual cells found in a fine needle aspiration biopsy of suspect breast lumps, and a binary class y_m indicating whether the lump was later found to malignant or benign. The original dataset contains 699 data-points, however 17 data-points have missing attributes so $M = 682$ data-points were used in the experiments here.

To model the unknown relationship between the input descriptors and binary class label output, a zero-mean Gaussian process prior [221] was placed on a set of latent real-valued function values $\mathbf{z} \in \mathbb{R}^M$. A squared exponential covariance function was used with per-feature length scales $\boldsymbol{\ell} \in \mathbb{R}^K_{>0}$ and output scale $\mathrm{s} \in \mathbb{R}_{>0}$, with the covariance function specifically defined as

---

function SqExpCov($\{\boldsymbol{d}^{(m)}\}_{m=1}^{M}, \mathrm{s}, \boldsymbol{\ell}, \epsilon = 10^{-8}$)
  for $i \in \{1 \dots M\}$ **do**
    $\mathrm{C}_{i,i} \leftarrow \mathrm{s} + \epsilon$
    for $j \in \{1 \dots i-1\}$ **do**
      $\mathrm{C}_{i,j} \leftarrow \mathrm{s} \exp\left(-\frac{1}{2}\sum_{k=1}^{D}\left(\frac{d_k^{(i)} - d_k^{(j)}}{\ell_k}\right)^2\right)$
      $\mathrm{C}_{j,i} \leftarrow \mathrm{C}_{i,j}$
  return C

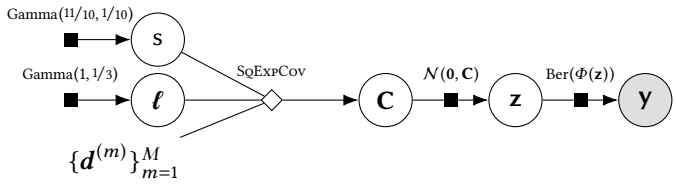


Figure 3.10.: Gaussian process probit regression factor graph.

The ϵ value is a 'jitter' parameter to improve numerical stability [221]. This covariance functions represents an assumption that nearby $\boldsymbol{d}^{(m)}$ points correspond to similar $\mathbf{z}$ values, with the typical length-scales in the feature space over which correlations are high determined by the values of the elements of $\boldsymbol{\ell}$. The latent variables $\mathbf{z}$ are assumed to determine the probability of the observed binary class outputs $\mathbf{y}$ being one or zero by a probit link function i.e. given $\mathbf{z} = z$ the binary outputs are modelled as having a Bernoulli distribution $\mathrm{Ber}(\Phi(z))$ where Φ is the standard normal CDF function. Following [87] Gamma prior distributions were placed on both the per-feature length-scales $\boldsymbol{\ell}$ and output-scale s covariance function parameters. The overall model is shown as a directed factor graph in Figure 3.10.

For inference we assume we are interested in inferring the posterior distribution on the $\boldsymbol{\ell}$ and s covariance function parameters given the observed input-output pairs, such that we could then use the inferred plausible $\boldsymbol{\ell}$ and s values to make predictions of the outputs corresponding to unlabelled inputs. We define the target variables for inference $\mathbf{x}$ as the logarithms of $\boldsymbol{\ell}$ and s so that the target distribution has support on an unbounded space i.e. $\mathbf{x} = [\log \mathrm{s}; \log \boldsymbol{\ell}]$ and $\mathbf{x} \in \mathbb{R}^{10}$, with a Jacobian determinant factor accounting for the change of variables being included in the transformed prior (marginal) density $\mathrm{p}_{\mathbf{x}}$

$$\mathrm{p}_{\boldsymbol{x}}(\boldsymbol{x}) \propto \exp\left(\frac{11x_1}{10} - \frac{\exp(x_1)}{10}\right) \prod_{i=2}^{10} \exp\left(x_i - \frac{\exp(x_i)}{3}\right). \tag{3.18}$$

The unnormalised target density is then $\tilde{p}(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y}) \propto \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y} \mid \boldsymbol{x})$. We cannot evaluate $\mathrm{p}_{\mathbf{y}|\mathbf{x}}$ as it involves an intractable marginalisation over the latent function values $\mathbf{z}$

$$\mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) = \int_Z \prod_{m=1}^{M} \left(\Phi(z_m)^{y_m}(1-\Phi(z_m))^{1-y_m}\right) \mathcal{N}(z \mid \mathbf{0}, \mathbf{C})\, \mathrm{d}z. \tag{3.19}$$

One option would be to construct a Markov chain on the joint $(\mathbf{x}, \mathbf{z})$ space with an unnormalised target density $\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}$, however strong dependencies between the (transformed) covariance function parameters $\mathbf{x}$ and the latent variables $\mathbf{z}$ makes the joint distribution difficult for MCMC methods to explore effectively [87]. As an alternative [87] proposes to use the pseudo-marginal framework to construct a Markov chain using an unbiased importance sampling estimator of $\tilde{p}$.

Though a Monte Carlo estimate of (3.19) can be formed by sampling latent values $\mathbf{z}$ from the Gaussian process prior $\mathrm{p}_{\mathbf{z}|\mathbf{x}}$, as this ignores the observed output values $\boldsymbol{y}$ this will tend to lead to a density estimator with an unusably high variance for the purposes of use in a pseudo-marginal update. A key insight in [87] was that much lower variance density estimates can be computed by using an optimisation-based approximate inference method to fit a Gaussian approximation to $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ (which as discussed previously is the optimal choice for the importance distribution in terms of minimising variance) to use as the importance distribution. In [87] both Laplace's method and expectation propagation are considered within this context; we concentrate on Laplace's method here for simplicity.

As discussed in Appendix C, Laplace's method involves finding the mode of the density being approximated and then evaluating the Hessian matrix of the log density at this point. An efficient and numerically stable implementation of a Newton–Raphson method can be used to find the mode of the latent posterior for this probit regression Gaussian process model [221, §3.4] with the latent posterior density guaranteed to have a unique mode. Each Newton–Raphson step involves computing a Cholesky factorisation of the Hessian of the log density at the current point which has a $O(M^3)$ computational cost. In the experiments around 10 Newton steps were needed to achieve convergence when finding the mode. Evaluating the density of the Gaussian process prior on the latent function values $\mathbf{z}$ also requires computing a Cholesky decomposition of the Gaussian process covariance matrix which again has $O(M^3)$ cost. As $M = 682$ these cubic cost operations will tend to be the dominant contributor to the overall run time. As the Gaussian process covariance and Laplace approximation to the latent posterior both depend on the value of the covariance function parameters, the cubic operations have to be performed each time a density estimate is computed at a new value for the target variables.

Once an approximate Gaussian latent posterior $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ has been fitted using Laplace's method, it can then be used as the importance distribution in an importance sampling estimator of the form shown in (3.4). The Cholesky factorisation $\boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}} = \text{chol}\,\boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}$ is computed as part of the Laplace's method iteration, and so can be reused to efficiently evaluate the importance distribution density at a $O(M^2)$ cost for each importance sample and to generate samples from the importance distribution using $\boldsymbol{z}^{(n)} = \boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}}\boldsymbol{u}^{(n)} + \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}$ where $\boldsymbol{u}^{(n)}$ is a sampled standard normal vector from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, this again having a $O(M^2)$ cost. This same expression can also be used to reparameterise the estimator as a deterministic function of a set of independent standard normal values $\boldsymbol{u} = [\boldsymbol{u}^{(1)}; \boldsymbol{u}^{(2)} \ \ldots \ \boldsymbol{u}^{(N)}]$ as shown previously in (3.7).

Due to the high overhead of the cubic operations the result of [238] that a choice of $N = 1$ is close to optimal does not apply here. In experiments in [238] with a similar Gaussian process classifier model (using a logistic link function and a dataset with $M = 144$) it was found computational efficiency was approximately maximised by using $N = 200$ importance samples with it noted that this is around the number required for the $O(M^2N)$ cost of sample generation to be of comparable magnitude to the cubic operation cost. In the example of [238] a non-iterative approach is used to find a Gaussian importance distribution hence only a single Cholesky decomposition of the importance distribution covariance matrix is required. The use of an iterative Laplace method approximation for the importance distribution here as proposed in [87] makes it unclear whether a similar choice of the number of importance samples is reasonable here. While the even higher overhead of the multiple cubic operations per estimator evaluation supports possibly using $N \geq M$, part of the justification of using an expensive procedure to fit the importance distribution is that it means fewer importance samples are needed to achieve a low-variance density estimator. In the experiments with the same dataset in [87], $N = 1$ importance sample was used and found to work well, though in that case an isotropic covariance function was used with a single length scale parameter such that the dimensionality of the target space was two rather than ten as here.

In preliminary runs we found that the PM MH update had very low accept rates however small we set the proposal step-size when using $N = 1$ importance sample in the density estimator. Increasing the num-

ber of importance samples to $N = 50$ gave a significant improvement in performance and overall stability with a negligible increase in run time per update. Increasing the number of importance samples further to $N = 500$ gave a further increase in efficiency but also increased the run time in our implementation by around one third which outweighed the per iteration sampling efficiency gains made. We therefore used $N = 50$ importance samples for the main experiments with all methods; given the limited number of values tested this is unlikely to be optimal but in most practical situations we would be unlikely to perform an exhaustive search for the optimal N. Interestingly the auxiliary pseudo-marginal methods appeared to still be able to mix when using $N = 1$ importance sample with APM MI+MH chains still able to achieve a target accept rate in the range $[0.15, 0.3]$ for the MH updates to the target variables. Due to the negligible increase in run time however when using $N = 50$ importance samples we performed the experiments for the APM methods with $N = 50$ also.

We generated Markov chains for the model using each of PM MH, APM MI+MH, APM SS+MH and APM SS+SS for the updates. For the MH updates to the target variables we used a Gaussian random-walk Metropolis proposal distribution $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$. To set the proposal step size λ we followed the adaptive approach used in [87], with the step size adjusted over an initial warm-up phase of 2000 iterations, with the average accept rate over every 100 iterations used as a control signal to decide whether to increase or decrease the step size. Also following [87] a target average accept rate range of $[0.15, 0.3]$ was used[4], with the step size made smaller or larger, if the average accept rate is below or above this range respectively during the adaptive phase. As noted in the previous experiments, while a target rate of 0.234 for the MH updates to the target variables in APM methods can be justified theoretically and empirically, it is not clear what the optimal choice is for PM MH updates, with this seeming to be dependent on the estimator variance and so number of importance samples N. While the $[0.15, 0.3]$ target accept rate range therefore seems reasonable for the APM methods it is unclear whether it is a good choice for the pseudo-marginal method, however as it was used with some success in [87] and given a lack of

4 Although in the published version of [87] it is stated a target range of $[0.2, 0.3]$ was used, the code accompanying the paper suggests a range of $[0.15, 0.3]$ was used in the experiments so we follow that instead.

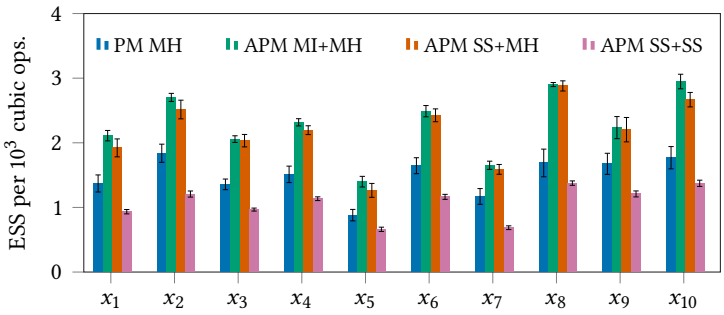
obvious alternative methods for choosing the target rate, we use it for the PM MH updates here.

For the APM SS+MH and APM SS+SS methods we used elliptical slice sampling for the updates to the auxiliary variables. For the slice sampling update to the target variables in the APM SS+SS chains we used linear slice sampling along a random direction vector (sampled isotropically) with a fixed initial bracket width of $w = 4$ and no linear step out iterations. To account for the 2000 adaptive warm up iterations performed before the main PM MH, APM MI+MH and APM SS+MH chains, for the APM SS+SS chains we ran 1000 warm up iterations before the main chain runs. For all four methods, 10 chains independently initialised from the prior were run for 10 000 iterations, with both the total number of cubic operations performed and overall run time recorded to allow for adjustment for different per iteration costs in the results.

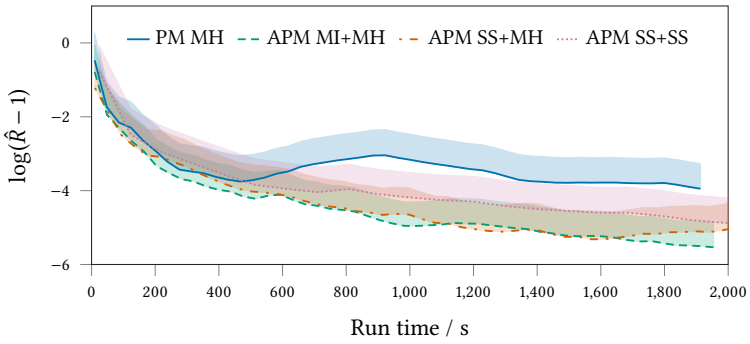
Results of the experiments are summarised in Figure 3.11. As a first measure of performance we consider the relative estimated sampling efficiency of the different methods. For each of the 10 target variables we estimated the ESS for the estimated mean of the variable using `R CODA` [211] and normalised these values by the total number of cubic operations performed in each chain[5]. The means of these values across the 10 chains per method (and standard errors) are shown for each of the target variables in the bar plot in Figure 3.11a.

By this ESS measure of efficiency, the APM MI+MH and APM SS+MH chains both consistently perform better than the PM MH chains, with they performing very similarly to each other, and the APM SS+SS chains perform worse than all other methods. Note that as the updates to the auxiliary variables to do not require any cubic operations (providing the Cholesky factorisations of the Gaussian process prior covariance and importance distribution covariance at the current target variable values are cached from the target variable update), there is little effect on the overall run time from using elliptical SS updates to the auxiliary variables as opposed to MI updates, hence the much closer performance here of APM MI+MH and APM SS+MH compared to the previous Gaussian latent variable experiments. The average accept rate of the MI updates

5 The mean chain run time per cubic operation performed was 0.0184 ± 0.00007 s for PM MH, 0.0187 ± 0.00014 s for APM MI+MH, 0.0196 ± 0.00012 s for APM SS+MH and 0.0184 ± 0.00013 s for APM SS+SS so using the cubic operation count as a proxy for overall computational cost seems reasonable here and removes the effect of any variable background system processes on the wall-clock run times.



(a) Sampling efficiency. ESS estimates for each of 10 target variables normalised by the number of cubic cost operations performed per chain. The bars show the means values across 10 independent chains with markers for ±1 standard error of mean.



(b) Chain convergence. Plots of PSRF $\hat{R}$ statistic on a log scale computed across 10 independent chains initialised from the prior for increasing number of chain iterations (normalised by mean total run time for each method to adjust for different per iteration run times) for each of four transition operators tested. Curves show median value and filled regions indicate confidence interval to upper 95th percentile of computed estimate. A $\hat{R}$ value of unity is indicative of chains having converged to stationarity, so for the plotted $\log(\hat{R}-1)$ values, more negative values indicate chains approaching convergence.

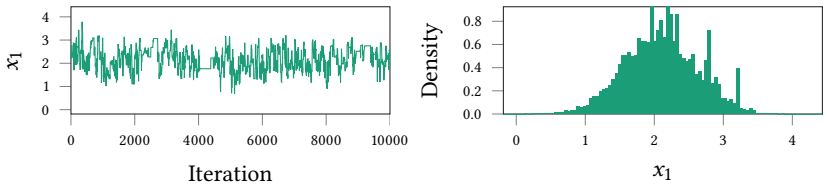
Figure 3.11.: Gaussian process probit regression results.

to the auxiliary variables in the APM MI+MH chains was 0.24 here suggesting there is probably a limited gain from using elliptical SS updates to the auxiliary variables in this case as the MI updates are likely to be mixing the auxiliary variables sufficiently well.

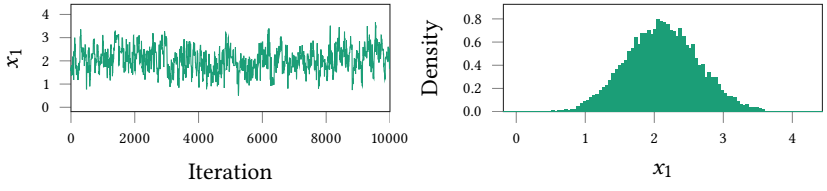
Although PM MH seems to outperform the APM SS+SS method here, other results suggest the estimated ESS measures of performance should be treated with some caution, with in general estimated ESSs being susceptible to giving misleading results when chains have poorly converged. Figure 3.11b shows plots of the *potential scale reduction factor* (PSRF) convergence diagnostic proposed by Gelman and Rubin in [98], also often

termed the $\hat{R}$ statistic. This is a heuristic measure of Markov chain convergence computed from multiple independent chains initialised from a distribution which should be over-dispersed compared to the (common) target distribution (we use the prior here). The diagnostic compares the between-chain and within-chain variance of each variable in the chain state, with a necessary but not sufficient condition for convergence being that these converge to being equal, corresponding to a $\hat{R}$ value of one. We used `CODA` to estimate the $\hat{R}$ values from the 10 independent chains run for each method as a function of an increasing number of iterations in the chain sequences used to compute the $\hat{R}$ estimates. We then accounted for the different per iteration run time of the different methods (in particular the APM SS+SS chains took on average $\sim 2.5\times$ longer per iteration than the other methods) by plotting these $\hat{R}$ values for increasing chain iterations against the estimated run time to complete that number of iterations, the resulting curves shown in Figure 3.11b. The darker coloured curves show the median of the estimated $\hat{R}$ interval and the lighter filled regions of the same colour show the 50th–95th percentile range of the estimate. To allow the curves to be more clearly distinguished, the $\hat{R}$ values are plotted on a shifted log scale i.e. $\log(\hat{R} - 1)$, with more negative values therefore corresponding to $\hat{R}$ values closer to one and so are indicative of the chains being closer to convergence.

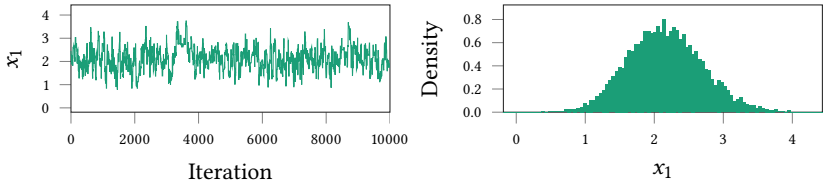
On this measure of performance the PM MH chains seem to perform more poorly, showing a slower convergence rate than the other methods, including the APM SS+SS chains. The non-monotonically decreasing behaviour seen in the $\hat{R}$ curve for the PM MH chains seems to be the result of the chains suffering the earlier discussed sticking behaviour, with one of the 10 chains found to have stuck for a run of over 2000 iterations and multiple incidents of sticking periods of hundreds of iterations in all of the chains. An example trace of one of the chains for the x_1 target variable is shown in Figure 3.12a where these sticking periods are clearly visible. The chains run using $N = 500$ importance samples (traces not shown here) also showed sticking behaviour though somewhat less frequently, suggesting that while increasing the number of importance samples can lessen the impact of these events, it does not seem to necessarily eliminate them. Figure 3.12 also shows example chain traces for the x_1 variable for each of the three other APM methods; in all cases here there are no visible long sticking periods and this was also reflected across the other chains not shown.



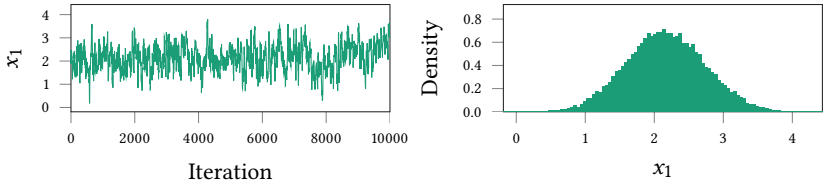
(a) PM MH



(b) APM MI+MH



(c) APM SS+MH



(d) APM SS+SS

Figure 3.12.: Example traces and histograms of target variable $x_1 = \log s$ from chains sampled using pseudo-marginal and auxiliary pseudo-marginal approaches in Gaussian process probit regression model inference task. In each row a trace of the sampled values for the x_1 variable for a single 10 000 iteration Markov chain is shown in the left plot, while the right plot shows a histogram of the sampled values from all 10 chains. In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the roughly 2.5 times increase in run time for the APM SS+SS chains compared to the other methods.

The right column of Figure 3.12 shows histograms for the x_1 target variable computed from the samples from all 10 chains for each method (in the case of the APM SS+SS chains only the first 4000 iterations from each chain were included to account for the roughly 2.5 times slower run time per chain in this case). Although we do not have a ground truth for the marginal posterior density here to compare against, it seems reasonable to assume that the spurious peaks in the histogram for the PM MH chains are not a reflection of the true marginal density but instead a result of the long sticking artefacts in the chains causing the states that the chain remains stuck at to be overly represented in the histograms. The APM methods produced much smoother marginal density estimates, with the APM SS+SS chains seeming to give a particularly smooth result here even with the run time adjustment meaning this histogram is computed from less than half the number of samples as used in the other methods. Although by no means conclusive, this provides a further suggestion that the relatively poor standing of the APM SS+SS chains on the ESS measure of performance is not an entirely accurate portrayal of the overall performance of the method.

3.7 DISCUSSION

The auxiliary pseudo-marginal methods discussed in this Chapter are a relatively simple extension to the existing pseudo-marginal MCMC framework which nonetheless offer some important benefits.

The simplest proposed approach of splitting the combined proposed update to both auxiliary and target variables in the standard PM MH algorithm into a separate Metropolis independence update to the auxiliary variables and Metropolis–Hastings update to the target variables (APM MI+MH) involves changing only a few lines of code in most implementations and adds no further free parameters to tune. Despite involving only a minor change, in the empirical studies performed this adjusted update was found to give significantly better computational cost normalised sampling efficiency over the standard pseudo-marginal Metropolis–Hastings update, despite in some cases doubling the computational effort per overall chain update. A simple intuition for understanding this improved performance is that for a fixed proposal distribution for the target variables, the accept rate of the MH update to the target variables in the APM MI+MH chains was typically more than double

the corresponding accept rate for the overall PM MH update. Therefore the doubling of the number of density estimates needed per iteration was more than outweighed by more than double the number of proposed target variable updates *from the same proposal distribution* (e.g. same Gaussian random-walk step-size) being accepted

The size of the increase in the accept rates for a fixed proposal distribution is dependent on how high the variance of the density estimator is or equivalently how dependent the target and auxiliary variables are under the auxiliary joint target. For high variances cases e.g. when using $N = 1$ importance sample, the increase in accept rates for the target variable updates in APM MI+MH chains over the accept rate of the PM MH updates is higher due to the poor performance of making independent proposed updates to the auxiliary variables in the PM MH having a strong deleterious effect on the PM MH accept rate. For example in the Gaussian latent variable model experiments when using $N = 1$ importance sample the accept rate of the MH updates in the APM MI+MH chains was typically around a factor of 20 higher than the accept rate for the corresponding PM MH chains using the same proposal step size. As the variance of the density estimator is decreased by increasing N, the difference in the accept rates for a fixed proposal step size becomes less marked with around a factor five difference for $N = 8$ and around a factor two difference for $N = 2$ between APM MI+MH and PM MH. So with a lower variance estimator the difference in performance between APM MI+MH and PM MH becomes less marked.

However the recommendation of [238] suggests that when the computational cost of each PM MH update scales linearly with N (and when using a density estimator formed as an average of unbiased Monte Carlo estimates) that using $N = 1$ is close to optimal for PM MH despite the higher estimator variance. As the low N, high density estimator variance cases are precisely when we expect to see the largest potential gains from using APM MI+MH over PM MH this suggests when this linear cost scaling argument is valid there will often be a computational gain from using APM MI+MH. In some cases as we saw in the Gaussian process experiments we can form a much lower variance density estimate by expending some computational effort to fit a good importance distribution. In these cases due to the additional overhead of the fitting procedure the linear cost scaling argument no longer applies. Further the density estimates in this case may be sufficiently low variance for

there to be little improvement in accept rates of updates to the target variables by splitting the PM MH in to separate MI and MH updates. However typically in these cases the overhead introduced by separately updating the auxiliary variables in an MI step will also be much less than the cost of the original PM MH update, as for fixed values of the target variables the importance distribution does not need to be refitted and any target variable dependent computations such as Cholesky factorisations of covariance matrices can be cached and reused. Therefore the overall cost per APM MI+MH update will be very close to that of each PM MH update and so even a small improvement in accept rate of the target variable updates can make it worthwhile to split the update.

Perhaps more important than the sampling efficiency gains seen from using APM MI+MH over PM MH in the experiments here was the significantly improved ability to tune the MH updates in the algorithm even when using a high-variance density estimator. By decoupling the dependency of the MH accept rate from the density estimator variance, theoretical guidelines for choosing a proposal step-size based on the average accept rate can be straightforwardly applied to tune APM MI+MH updates. The resulting increased ease of use of the algorithm and decreased requirement for user intervention to get good performance might often make APM MI+MH an attractive choice even when the extra run time overhead per update negates any sampling efficiency gains. Further the separate MI step accept rate of the APM MI+MH update provides a diagnostic already computed as part of the chain updates which can alert users to issues with poor mixing of the auxiliary variables due to low accept rates of the auxiliary updates. In contrast it will not always be clear if a poor accept rate of a PM MH chain is due to poor choice of the target variables proposal distribution or due to a high density estimator variance, and separately monitoring the density estimator variance as part of the update adds overhead while not being as directly interpretable as the MI step accept rate.

If initial runs using an APM MI+MH method do show a very low accept rate for the updates to the auxiliary variables which might lead to convergence issues, the proposed APM SS+MH approaches offer a simple 'plug-in' solution to improve mixing of the auxiliary variables without having to tune a separate proposal distribution for a MH update to the auxiliary variables. If the auxiliary variables can be naturally represented as being marginally distributed according to the standard normal

distribution, then elliptical slice sampling is a straightforward choice, having no free parameters to tune and still initially proposing bold moves to near independent points in the auxiliary space while able to back-off to more conservative updates to ensure a non-zero move to the auxiliary variables under weak smoothness conditions.

Another common case is auxiliary variables which are naturally parameterised as a vector of standard uniform draws, in which case reflective linear slice sampling offers analogous benefits. Although the linear slice sampling algorithm does have a free initial bracket width parameter to be chosen, in general (as seen in the experiments using this algorithm for updates to the target variables in the Gaussian latent variable model) the efficiency of the algorithm is not strongly dependent on the choice of this parameter providing it is set large enough to cover most of the intersection of the slice with the sampled line as the exponential shrinking of the bracket on proposing an off slice point will quickly reduce the bracket to a more appropriate size if set initially too large. For reflective slice sampling in the unit hypercube a fixed initial bracket width of one and a direction vector $\boldsymbol{v}$ sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ was found to work well in experiments applying APM SS+MH methods to inference in a doubly-intractable Ising model problem in [185].

Independently of and concurrently with the original conference publication [185] related to this work, both Dahlin et al. [68] and Deligiannidis et al. [73] considered related frameworks in which the auxiliary random variables of a pseudo-marginal density estimator are updated using a Metropolis–Hastings method which leaves the distribution defined by the density (3.8) on the joint auxiliary–target variable space invariant. Both assume a parameterisation in which the auxiliary variables have an isotropic standard normal marginal distribution $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$, and consider a Metropolis–Hastings update to the auxiliary variables with proposal density

$$r^*(\boldsymbol{u}' \mid \boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{u}' \mid \sqrt{1-\lambda^2}\boldsymbol{u}, \lambda^2\mathbf{I}\right) \tag{3.20}$$

which can be considered as a discretisation of a Ornstein-Uhlenbeck diffusion process or as a fixed step size update on an elliptical path that the elliptical slice sampling algorithm 5 generalises by adaptively setting the step size λ. This fixed step-size Metropolis–Hastings update is more amenable to analysis, with both [68] and [73] giving much more

extensive theoretical justifications for using perturbative updates to the auxiliary variables (or equivalently introducing correlations in between the auxiliary variable samples) than the mainly intuition based and empirical arguments made here. These theoretical insights are important for informing future development of these ideas. In practical settings however, though the above MH update with an optimally tuned choice of λ may give better sampling efficiency performance compared to the elliptical slice sampling updates proposed here, we would suggest that the additional tuning burden placed on the user and loss of robustness in cases where the appropriate step size varies across the state space, would suggest that elliptical slice sampling updates to the auxiliary variables are still often a good default choice.

The empirical evidence for using slice sampling updates to the target variables as in the proposed APM MI+SS and APM SS+SS methods is less strong, with in both of the models considered in the experiments here these methods having poorer run-time adjusted efficiency than the APM MI+MH and APM SS+MH methods respectively. If adapting an existing PM MH algorithm where some effort has already been extended to identify an appropriate proposal distribution for updates to the target variables or other information is available to inform this choice, the additional overhead of the slice sampling updates might not be worthwhile. In cases however where we have less prior knowledge about appropriate scales for updates to the target variables or are more concerned with overall robustness and ease of use, slice sampling updates to the target variables are likely to be more attractive.

Subsequent to the publication of the conference paper related to this work, Lindsten and Doucet proposed the use of HMC within an (auxiliary) pseudo-marginal framework [159]. Under the assumption that the joint auxiliary target density (3.8) is defined with respect to the Lebesgue measure and is differentiable, their *pseudo-marginal Hamiltonian Monte Carlo* algorithm proposes jointly updating the auxiliary and target variables using a HMC transition operator. In particular they assume the marginal distribution on the auxiliary variables R is standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and leverage this to propose an alternative symplectic integrator to the typical leapfrog scheme which improves the scaling of the method the dimensionality of the auxiliary variable space is high.

In numerical experiments with a hierarchical model of a diffraction process with a three target variables to infer, it was found that the proposed pseudo-marginal HMC algorithm gave similar performance to using a APM SS+MH update when normalised by the computational cost per update. In a second experiment with a generalised linear mixed model with a 13 dimensional target space, the proposed pseudo-marginal HMC algorithm was compared to a APM SS+MH update in which the update to the target variables is formed of a sequential scan of per-dimension random-walk Metropolis updates to each individual target variable. It is reported that attempts to jointly update all target variables in the MH step led to very poor acceptance rates. The traces for the pseudo-marginal HMC chain (Figure 4 in [159]) in this case indicate improved mixing compared to the APM SS+MH update, though as the run-time per sample of the pseudo-marginal HMC method is reported to be approximately 3.5 times higher in the implementation used and the traces do not appear to be run-time adjusted it is not clear what a cost normalised comparison would show. Autocorrelation plots for chains from the two approaches are also shown (Figure 13 in [159]), with the pseudo-marginal HMC method showing quicker decay of the autocorrelations per sample lag compared to APM SS+MH though again it is not clear if the autocorrelation plots are run-time adjusted. Both the pseudo-marginal HMC and APM SS+MH chains appear to mix significantly better than *Particle Gibbs* [8], an auxiliary variable approach based on a particle filter density estimator.

The use of HMC updates with an auxiliary pseudo-marginal framework seems an appealing idea when the required gradients are available due to the improved performance in complex high-dimensional target distributions often offered by HMC methods, and the integrator proposed by [159] is an elegant approach for exploiting structure in the auxiliary target distribution to give improved performance when the number of auxiliary variable dimensions is large. Though in the experiments in [159] it is not clear how significant the gain in performance is over using random-walk Metropolis updates to the target variables in a APM SS+MH method, it seems plausible that in models with higher-dimensional target space that HMC updates would start to increasingly outperform random-walk Metropolis updates to the target variables. In the next chapter we will discuss related methods which apply HMC updates to perform inference in simulator models; this work was performed concurrently and independently to [159].

4 IMPLICIT GENERATIVE MODELS

In the approximate inference methods considered in Chapter 2 a unifying element was the requirement to be able to evaluate an explicit probability density function for the target distribution of interest. In the previous chapter we considered the pseudo-marginal framework which allowed relaxing this requirement to being able to evaluate an unbiased (and non-negative) estimator for the target density. In this chapter we will consider probabilistic models specified by a generative process in which the density of the model variables is defined only *implicitly* [19, 77, 113] - that is we can generate sample values for the variables in the model, but we cannot tractably evaluate the probability distribution of those variables or more specifically its density with respect to an appropriate base measure.

Although models without an explicit density function are challenging to work with from an inferential perspective, they are ubiquitous in science and engineering in the form of probabilistic models defined by the computational simulation of a physical system. Typically simulator models are specified procedurally in code with any stochasticity introduced by drawing values from a pseudo-random number generator. The complexity of the function mapping from random inputs to simulated outputs typically makes calculating an explicit density on the outputs at best non-trivial and often intractable (as seen for a simple example in Figure 1.6 in Chapter 1).

There has also been a long history in statistics of using distributions defined by their quantile function [126, 254] from which we can easily generate independent samples using the inverse transform sampling method discussed in Chapter 2. Although these *quantile distributions* are often able to offer very flexible descriptions of shape of a distribution [104] often the quantile functions will not have an analytic inverse meaning their CDF and so density function cannot be evaluated analytically. Generative models in which the density of the model variables is only defined implicitly have also been the subject of substantial recent interest in the machine learning community due to the development

of effective training approaches which do not require evaluation of a density on the model variables [82, 110, 157], with there being significant gains in modelling flexibility by dropping the requirement to be able to compute an explicit density function [177, 252].

The focus of this chapter will therefore be methods for performing approximate inference in generative models where we do not necessarily have access to an explicit density on the model variables. A lack of an explicit density function makes it non-trivial to directly apply the approximate inference approaches that have been discussed so far in this thesis. This has spurred the development of inference approaches specifically targeted at implicit generative models such as indirect inference [113] and *approximate Bayesian computation* (ABC) [19].

In both indirect inference and ABC, inferences about plausible values of the unobserved variables are made by computing distances between simulated observed variables and observed data. At a qualitative level, values of the unobserved variables associated with simulated observations that are 'near' to the data are viewed to be more plausible. This approximation that the simulated observations are only close but not equal to the observed data makes the inference problem more tractable, but also biases the inference output. Further simple distance measures tend to become increasingly less informative as the dimensionality of a space increases, making it challenging to use these approaches to perform inference in models with large numbers of unobserved variables. This motivates the use of dimensionality reduction techniques to project the observations to a set of lower-dimensional summary statistics. Although through careful choice of summaries this approach can yield good results, identifying informative summaries is challenging and except for rare cases where sufficient statistics are available any reduction to summary statistics will entail a loss of information about the unobserved variables compared to conditioning on all observations.

We make two main contributions in this chapter. First we show that by reparameterising the approximate conditional expectations estimated in ABC approaches to inference in generative models it is possible to express them in the form of an expectation of a function of a random vector variable distributed according to a density which we can evaluate up to a normalising constant. This makes it possible to apply efficient general purpose approximate inference methods such as slice sampling and Hamiltonian Monte Carlo to implicit generative models

without the need to develop dedicated ABC variants. It is sometimes feasible to apply these methods when conditioning on all observations without the need to reduce dimensionality using summary statistics. The reparameterisation used is closely related to that applied to pseudo-marginal density estimators in the previous chapter, with existing ABC MCMC methods being a common special case of the pseudo-marginal Metropolis–Hastings update discussed there.

Secondly for a restricted class of generative models we term differentiable generative models and which we define in a following section, we show that it is possible to express exact conditional expectations under the model as integrals against a density we can evaluate pointwise across an implicitly defined manifold. We use this to propose a novel constrained HMC method for performing inference in differentiable generative models. Unlike ABC approaches, this method allows inference to be performed by conditioning the observed variables in the model to be within arbitrary small distances of the data values while remaining computationally tractable.

The contributions described in this chapter have previously appeared in the published conference paper

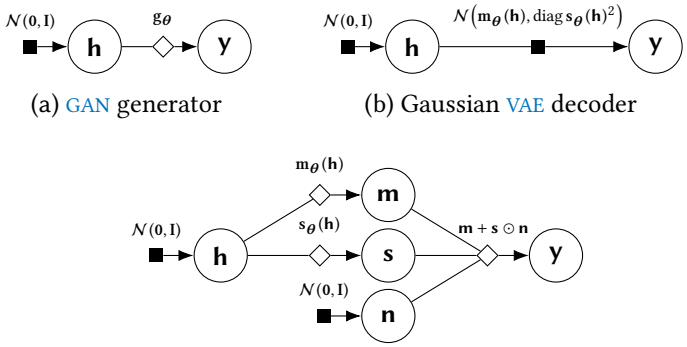
- Asymptotically exact inference in differentiable generative models. Matthew M. Graham and Amos J. Storkey. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR 54:499-508*, 2017.

I was the primary author of that work and proposed the novel contributions made in the paper. I also performed and analysed the numerical experiments described in the paper, some of which are reproduced in this chapter in Section 4.10. This chapter expands upon the analysis and discussion in the above publication and includes additional numerical experiments.

4.1 DIFFERENTIABLE GENERATOR NETWORKS

We first review two approaches to specifying generative models using differentiable networks[1], *generative-adversarial networks* (GANs) [110] and *variational autoencoders* (VAEs) [139, 224]. Although the methods

1 A a differentiable parametric function formed by interleaving 'layers' of affine transformations and elementwise non-linearities, also termed a *neural network*.



(a) GAN generator

(b) Gaussian VAE decoder

(c) Reparameterised Gaussian VAE decoder

Figure 4.1.: Example factor graphs for the generator of a GAN and decoder of a VAE. (a) The generator for a GAN with a standard normal distribution on the hidden code $\mathbf{h}$, this mapped through a differentiable network $\mathbf{g}_{\boldsymbol{\theta}}$, to generate the simulated output vector $\mathbf{y}$. (b) The decoder of a Gaussian VAE. Again a hidden code vector $\mathbf{h}$ with a standard normal distribution is used, differentiable network functions $\mathbf{m}_{\boldsymbol{\theta}}$ and $\mathbf{s}_{\boldsymbol{\theta}}$ then mapping from this code vector to mean and diagonal covariance parameters of a multivariate normal distribution on the output vector $\mathbf{y}$. (c) The same VAE decoder model as (b), with in this case the conditional factor on the outputs $\mathbf{y}$ given hidden code $\mathbf{h}$ reparameterised in terms of a deterministic transformation of a standard normal vector $\mathbf{n}$.

used for training these models differ significantly, their generative component have the same form of a function, specified by a differentiable network, which takes as input a vector of random variables from a known distribution and outputs a generated sample from an implicitly defined distribution. The overarching term *differentiable generator networks* has been suggested for generative models with this form [109]. We will use a VAE model in one of the later experiments in this chapter so this material is partly to provide the necessary background for our description of the models used in that experiment, however more broadly the structure of the generative models described here was a key inspiration for the ideas described in this chapter.

GANs [110] have become a popular approach in unsupervised machine learning for training models which can generate plausible simulated data points, typically images, given a large collection of data to learn from. The training procedure for GANs is posed as a minimax game between a *generator function*, a differentiable network $\mathbf{g}_{\boldsymbol{\theta}}$ which receives as input a vector of values $\mathbf{h}$ drawn from a simple known distribution such as the standard normal and outputs a simulated data point $\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{h})$, and an adversarial *discriminator function* $\mathbf{d}_{\boldsymbol{\phi}}$, which

predicts whether a presented vector input is a simulated or real data point drawn from the training data. Training proceeds by updating the generator parameters $\boldsymbol{\theta}$ to maximise the expected discriminator uncertainty, while the discriminator parameters $\boldsymbol{\phi}$ are updated to minimise the expected discriminator uncertainty.

Although there are many variants of this basic outline of the training procedure, for our purposes the main relevant factor is that most GAN models retain the same basic structure for the generator, which is visualised as a factor graph in Figure 4.1a. While $\mathrm{p}_{\mathbf{h}}$ is known, as $\mathbf{y}$ is a deterministic transformation of $\mathbf{h}$ there is not a well-defined joint density on $\mathbf{h}$ and $\mathbf{y}$. If $\mathrm{g}_{\boldsymbol{\theta}}$ were bijective we could apply the change of variables formula (1.22) to calculate the density $\mathrm{p}_{\mathbf{y}}$ in terms of $\mathrm{p}_{\mathbf{h}}$ and the Jacobian $\mathbf{J}_{\mathrm{g}_{\boldsymbol{\theta}}}$ however this will not usually be the case - typically in fact the dimensions of $\mathbf{y}$ and $\mathbf{h}$ will differ.

An alternative generative modelling approach using differentiable networks is the Gaussian VAE [139] or *deep latent Gaussian model* [224]. In a Gaussian VAE differentiable networks $\mathbf{m}_{\boldsymbol{\theta}}$ and $\mathbf{s}_{\boldsymbol{\theta}}$ are used to generate respectively the mean and per-dimension standard deviations, corresponding to a diagonal covariance matrix, of a conditional normal distribution on the outputs given a hidden code vector $\mathbf{h}$ drawn from a known distribution. The simulated output $\mathbf{y}$ can then be generated by sampling from the conditional distribution $\mathcal{N}\left(\mathbf{m}_{\boldsymbol{\theta}}(\boldsymbol{h}), \operatorname{diag} \mathbf{s}_{\boldsymbol{\theta}}(\boldsymbol{h})^2\right)$ given a sampled code vector $\boldsymbol{h}$. Unlike a GAN, in a Gaussian VAE the joint density on $\mathbf{y}$ and $\mathbf{h}$ is tractable to evaluate, for the case of a normally distributed code vector $\mathbf{h}$ corresponding to

$$\mathrm{p}_{\mathbf{y},\mathbf{h}}(\boldsymbol{y}, \boldsymbol{h}) = \mathcal{N}\left(\boldsymbol{y} \mid \mathbf{m}_{\boldsymbol{\theta}}(\boldsymbol{h}),\ \operatorname{diag} \mathbf{s}_{\boldsymbol{\theta}}(\boldsymbol{h})^2\right) \mathcal{N}(\boldsymbol{h} \mid \mathbf{0},\ \mathbf{I}). \tag{4.1}$$

Although typically we cannot marginalise out the hidden code vector $\mathbf{h}$ to get the marginal density on the generated outputs $\mathbf{y}$, having access to the joint density allows the use of standard approximate inference methods when training the model. In particular as suggested by their name variational autoencoders are trained using a parametric variational inference approach which uses a second *encoder* differentiable network to encode the parameters of a variational approximation to the posterior density $\mathrm{p}_{\mathbf{h}|\mathbf{y}}$ given a data point $\boldsymbol{y}$, with a lower bound on the log joint density of the data points then maximised with respect to the encoder and decoder network parameters. Once a VAE model is

trained, the joint density (4.1) also allows direct application of approximate inference methods such as MCMC to infer plausible values for a subset $\mathbf{y}_1$ of the decoder generated outputs $\mathbf{y}$ given observations of the remaining values $\mathbf{y}_2$ by jointly inferring $\mathbf{y}_1$ and $\mathbf{h}$ given $\mathbf{y}_2$.

By reparameterising the normal conditional factor $p_{\mathbf{y}|\mathbf{h}}$ in (4.1) as a deterministic transformation $\mathbf{y} = \mathbf{m}_\theta(\mathbf{h}) + \mathbf{s}_\theta(\mathbf{h}) \odot \mathbf{n}$ where $\mathbf{n}$ is a vector of standard normal variables we can express the generative process specified by the decoder of a VAE similarly to that of a GAN by considering the generator to be $\mathbf{g}_\theta(\mathbf{h}, \mathbf{n}) = \mathbf{m}_\theta(\mathbf{h}) + \mathbf{s}_\theta(\mathbf{h}) \odot \mathbf{n}$ with both $\mathbf{h}$ and $\mathbf{n}$ as inputs. These two parameterisations of a VAE decoder are shown as factor graphs in Figures 4.1b and 4.1c.

This definition of the 'generator' corresponding to a VAE decoder is helpful when using it as a building block in a larger generative model where it is composed with other functions. When composing together several generator modules like this, even if we are able to evaluate a density on the variables in an individual module it may not be possible to evaluate a density on the variables of interest in the overall model. However by defining each module in the standard form of a differentiable function from input variables to generated outputs, the overall model retains the same form allowing us to build up more complex models and still be able to apply the same inference methods.

4.2 GENERATIVE MODELS AS TRANSFORMATIONS

In the preceding section we saw that the generative process of both GAN and VAE models can be described as a transformation of a vector of random variables drawn from a known distribution. This formulation of a generative model in fact extends beyond these machine learning examples. Any probabilistic model that we can programmatically generate values from in a finite time can be expressed in the form of a deterministic function which takes as input a vector of random variables sampled from a known distribution. This observation just corresponds to stating that we can track all of the calls to a random number generator in a program, and that given the values sampled from the random number generator all of the operations then performed by the program

are deterministic[2]. The key idea we will exploit in this chapter is that we can perform inference in generative models by considering the distribution induced on the random inputs to the model when conditioning on partial observations of the generated output.

To formalise this idea we first introduce some notation. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space, and $(X, \mathcal{G})$, $(Z, \mathcal{H})$ be two measurable spaces. We denote the vector of observed random variables in the model of interest as $\mathbf{x} : S \to X$ and the vector of unobserved random variables that we wish to infer $\mathbf{z} : S \to Z$. Our objective is to be able to compute conditional expectations $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x}] : X \to F$ of arbitrary measurable functions $f : Z \to F$ of the unobserved variables given known values for the observed variables $\mathbf{x}$. We now give a concrete definition for what we will consider as constituting a generative model for $\mathbf{x}$ and $\mathbf{z}$.

DEFINITION 4.1 (Generative model): *Let $(U, \mathcal{F})$ be a measurable space and $\mathbf{u} : S \to U$ a random vector taking on values in this space. We require that $\mathsf{P}_{\mathbf{u}}$ has a density ρ that we can evaluate with respect to a base measure μ and that we can generate independent samples from $\mathsf{P}_{\mathbf{u}}$. Then if $\mathrm{g}_{\mathbf{x}} : U \to X$ and $\mathrm{g}_{\mathbf{z}} : U \to Z$ are measurable functions such that*

$$\mathbf{x}(s) = \mathrm{g}_{\mathbf{x}} \circ \mathbf{u}(s) \quad \textit{and} \quad \mathbf{z}(s) = \mathrm{g}_{\mathbf{z}} \circ \mathbf{u}(s) \quad \forall s \in S \tag{4.2}$$

we define $(U, \mathcal{F}, \rho, \mu, \mathrm{g}_{\mathbf{x}}, \mathrm{g}_{\mathbf{z}})$ as a generative model for $\mathbf{x}$ and $\mathbf{z}$. We call $(U, \mathcal{F})$ the input space *of the generative model, $(X, \mathcal{G})$ the* observed output space *and $(Z, \mathcal{G})$ the* unobserved output space. *Further we will refer to $\mathrm{g}_{\mathbf{x}}$ as the* generator of $\mathbf{x}$ *and likewise $\mathrm{g}_{\mathbf{z}}$ the* generator of $\mathbf{z}$. *The random vector $\mathbf{u}$ is the* random inputs *and the density ρ the* input density.

Intuitively the input vector $\mathbf{u}$ represents all of the values drawn from a random number generator in the code of a generative model and the generator functions $\mathrm{g}_{\mathbf{x}}$ and $\mathrm{g}_{\mathbf{z}}$ represent the operations used to generate values for $\mathbf{x}$ and $\mathbf{z}$ respectively given values for the random inputs $\mathbf{u}$. In some cases the number of random inputs used in a generator evaluation will depend on the values of the random inputs themselves, for example if there is a branching statement which depends on a random input and the operations in each branch use different random inputs. Although implementationally more challenging, we can still consider

2 For the purposes of clarity of exposition here we consider the outputs of a pseudo-random number generator as truly random, even though in reality as we saw in Chapter 2 they are deterministically computed.

this case within the above framework by enumerating the random inputs required in all possible control flow paths through the generator code and mapping each to a different element in $\mathbf{u}$. In interpreted languages, this can be done lazily by detecting if a call to a random number generator object has occurred at the same point in a execution trace previously and if so matching to same element in $\mathbf{u}$ as used previously otherwise matching to a new $\mathbf{u}$ element.

In this chapter we will mainly concentrate on a restricted class of generative models which we term *differentiable generative models*.

DEFINITION 4.2 (Differentiable generative model): *Let* $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ *be a generative model for* $\mathbf{x}$ *and* $\mathbf{z}$ *as specified in Definition 4.1. Then if the following conditions are satisfied*

1. $U \subseteq \mathbb{R}^M$, $\mathcal{F} = \mathscr{B}(U)$ *and* $X \subseteq \mathbb{R}^{N_\mathbf{x}}$, $\mathcal{G} = \mathscr{B}(X)$,
2. $\mathsf{P}_\mathbf{u}$ *has density* ρ *with respect to* $\mu = \lambda^M$,
3. *the input density gradient* $\nabla\rho(\boldsymbol{u}) = \frac{\partial \rho}{\partial \boldsymbol{u}}$ *exists* $\mathsf{P}_\mathbf{u}$*-almost everywhere,*
4. *the generator Jacobian* $\mathbf{J}_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u}) = \frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}}$ *exists* $\mathsf{P}_\mathbf{u}$*-almost everywhere.*

we describe $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ *as a differentiable generative model.*

These requirements are quite severe: for example they exclude any models with discrete random inputs and those in which branch statements in the generator code introduce discontinuities. However there are still a large class of interesting generative models which do meet these conditions: for example models based on approximate integration of partial or ordinary differential equations combined with a stochastic observation model or SDE models without a jump-process component. As differentiability with respect to model parameters is a requirement for training models such as GANs and VAEs using stochastic gradient descent, the corresponding generators will also usually be differentiable with respect to the random inputs and so fall in to this class.

A further restriction we will require in some cases is that the Jacobian $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ is full row-rank $\mathsf{P}_\mathbf{u}$-almost everywhere, which requires that $M \geq N_\mathbf{x}$ i.e the number of random inputs is at least as many as the number of observed variables that will be conditioned on. In cases where this does not hold the implicitly defined probability distribution $\mathsf{P}_\mathbf{x}$ will not be absolutely continuous with respect to the Lebesgue measure. Instead $\mathsf{P}_\mathbf{x}$ will only have support on a sub-manifold of dimension locally equal

to the rank of $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ and conditioning on arbitrary $\boldsymbol{x} \in X$ is not a well-defined operation. The GAN generator models trained in practice often do not meet this condition as it is typical to use a lower dimensional hidden input than the generated output dimension [10]. There is no fundamental requirement in adversarial training to use generators of this form however and theoretical results [10] suggest that the lack of absolute continuity of the implicit distribution on the generator outputs with respect to $\lambda^{N_\mathbf{x}}$ may contribute to the often unstable behaviour of GAN training.

Although we only required the existence of the input density gradient $\nabla\rho$ and generator Jacobian $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ in Definition 4.2, unsurprisingly this is motivated by the need to evaluate these terms in the proposed inference methods for differentiable generative models. Although this may seem a limiting requirement for complex models, the availability of efficient general-purpose *automatic differentiation* (AD) libraries [17] means it is possible to automatically calculate the necessary derivatives given just the code defining the forward functions ρ and $\mathbf{g}_\mathbf{x}$. For generative models implemented in existing code this will often require re-coding using an appropriate AD framework. In some cases however it is possible to use AD tools which automatically transform existing source code — for example given C or Fortran code for a function *Tapenade* [125] can generate code for computing the function's derivatives. By applying the reverse-mode accumulation AD (Algorithm 16 in Appendix B) the gradient $\nabla\rho$ can be evaluated at a $O(1)$ cost relative to evaluating the density itself and the Jacobian $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ can be evaluated at a $O(N_\mathbf{x})$ factor of the cost of a single evaluation of the generator $\mathbf{g}_\mathbf{x}$.

For a given joint distribution $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ there is not a single unique corresponding generative model $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ for $\mathbf{x}$ and $\mathbf{z}$. As a simple example if $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ is a differentiable generative model and $\mathbf{f} : V \to U$ is a diffeomorphism, then we can reparameterise the random inputs as $\mathbf{v} = \mathbf{f}^{-1}(\mathbf{u})$, and define an input density $\tilde{\rho}(\boldsymbol{v}) = |\mathbf{J}_\mathbf{f}(\boldsymbol{v})|\,\rho(\mathbf{f}(\boldsymbol{v}))$ using the change of variables formula for a diffeomorphism (1.22), with $(V, \mathscr{B}(V), \tilde{\rho}, \mu, \mathbf{g}_\mathbf{x} \circ \mathbf{f}, \mathbf{g}_\mathbf{z} \circ \mathbf{f})$ then also a generative model for $\mathbf{x}$ and $\mathbf{z}$. In Appendix D we discuss some of the issues involved in choosing a parameterisation of a generative model. We will typically parameterise a model such that $\mathsf{P}_\mathbf{u}$ has unbounded support and each of the dimensions of $\mathbf{u}$ has unit variance, as these properties simplify the implementation of the MCMC methods we propose later in the chapter.

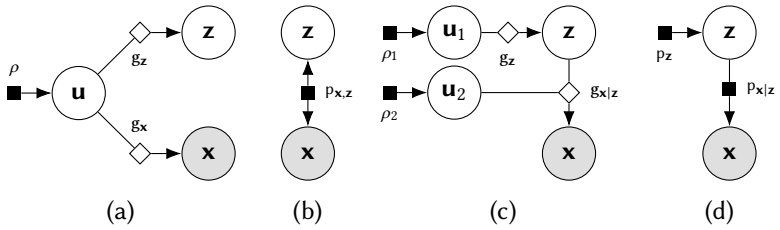


Figure 4.2.: Factor graphs of undirected and directed generative models. Panel (a) shows the more general undirected model case in which observed variables $\mathbf{x}$ and latent variables $\mathbf{z}$ are jointly generated from random inputs $\mathbf{u}$, with (b) showing equivalent factor graph after marginalising out the random inputs. Panel (c) shows the directed model case in which we first generate the latent variables $\mathbf{z}$ from a subset of the random inputs $\mathbf{u}_1$ then generate the observed variables $\mathbf{x}$ from $\mathbf{z}$ and the remaining random inputs $\mathbf{u}_2$, with (d) showing resulting natural directed factorisation of joint distribution when marginalising out $\mathbf{u}_1$ and $\mathbf{u}_2$.

4.3 DIRECTED AND UNDIRECTED MODELS

So far we have considered generative models where both the observed and unobserved variables are jointly generated from $\mathbf{u}$ without assuming any particular relationship between $\mathbf{z}$ and $\mathbf{x}$. This structure is shown as a factor graph in Figure 4.2a and a corresponding factor graph for just $\mathbf{x}$ and $\mathbf{z}$ with $\mathbf{u}$ marginalised out shown in Figure 4.2b.

A common special case is when the input space $U = U_1 \times U_2$ and the unobserved variables $\mathbf{z}$ are generated from a subset of the random inputs $\mathbf{u}_1 : S \to U_1$ (e.g. corresponding to sampling from a prior distribution over the parameters of a simulator model), with the observed variables $\mathbf{x}$ then generated from a function $\mathbf{g}_{\mathbf{x}|\mathbf{z}} : Z \times U_2 \to X$ which takes as input both the generated unobserved variables $\mathbf{z}$ and the remaining random variables $\mathbf{u}_2 : S \to U_2$, i.e. $\mathbf{x} = \mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z}, \mathbf{u}_2) = \mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{g}_{\mathbf{z}}(\mathbf{u}_1), \mathbf{u}_2)$. This is illustrated as a factor graph in Figure 4.2c. Again a corresponding factor graph with $\mathbf{u}$ marginalised out is shown in Figure 4.2d, with in this case the structure of the generator making a directed factorisation in terms $\mathrm{p}_{\mathbf{z}}$ and $\mathrm{p}_{\mathbf{x}|\mathbf{z}}$ natural.

We will therefore term models with this structure as *directed generative models* (with the more general case termed *undirected* for symmetry). The method we propose are equally applicable to undirected and directed generative models, though often the extra structure present in the directed case can allow computational gains. Most ABC inference methods concentrate on directed generative models. Typically the marginal

density $\mathrm{p}_{\mathbf{z}}$ will be tractable to explicitly compute in such cases, such that it is only the conditional density $\mathrm{p}_{\mathbf{x}|\mathbf{z}}$ which we cannot evaluate. As this conditional density is often referred to as the *likelihood*, an alternative designation of *likelihood-free inference* is sometimes used for ABC and related methods.

4.4 APPROXIMATE BAYESIAN COMPUTATION

We will now review the ABC approach to inference in generative models. We will assume here that the observed variables in the generative model of interest are real-valued, i.e. that $X \subseteq \mathbb{R}^{N_{\mathbf{x}}}$, with inference in generative models with discrete observations being in general simpler from a theoretical perspective (though not necessarily computationally). The auxiliary-variable description we give of ABC is non-standard, but is consistent with the algorithms used in practice and will help illustrate the relation of our approach to existing ABC methods.

We introduce an auxiliary X-valued random vector $\mathbf{y}$ which depends on the observed random vector $\mathbf{x}$ via a regular conditional distribution $\mathrm{P}_{\mathbf{y}|\mathbf{x}}$ we term the *kernel* which has a conditional density $k_\epsilon : X \times X \to [0, \infty)$ with respect to the Lebesgue measure,

$$\mathrm{P}_{\mathbf{y}|\mathbf{x}}(A \,|\, \boldsymbol{x}) = \int_A k_\epsilon(\boldsymbol{y} \,|\, \boldsymbol{x}) \,\mathrm{d}\boldsymbol{y} \qquad \forall A \in \mathscr{B}(X),\, \boldsymbol{x} \in X. \tag{4.3}$$

The kernel density k_ϵ is parameterised by a *tolerance* ϵ and chosen such that the following conditions holds for arbitrary Lebesgue measurable functions $f : X \to \mathbb{R}$

$$\lim_{\epsilon \to 0} \int_X f(\boldsymbol{y})\, k_\epsilon(\boldsymbol{y} \,|\, \boldsymbol{x}) \,\mathrm{d}\boldsymbol{y} = f(\boldsymbol{x}) \tag{4.4}$$

$$\text{and} \quad \lim_{\epsilon \to 0} \int_X f(\boldsymbol{x})\, k_\epsilon(\boldsymbol{y} \,|\, \boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} = f(\boldsymbol{y}). \tag{4.5}$$

Intuitively these requirements correspond to kernels which collapse to a Dirac delta in the limit of $\epsilon \to 0$. For kernels meeting these condition (4.4) we have that $\forall A \in \mathscr{B}(X)$

$$\begin{aligned} \lim_{\epsilon \to 0} \mathrm{P}_{\mathbf{y}}(A) &= \lim_{\epsilon \to 0} \int_X \mathrm{P}_{\mathbf{y}|\mathbf{x}}(A \,|\, \boldsymbol{x}) \,\mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \lim_{\epsilon \to 0} \int_X \int_X \mathbb{1}_A(\boldsymbol{y})\, k_\epsilon(\boldsymbol{y} \,|\, \boldsymbol{x}) \,\mathrm{d}\boldsymbol{y} \,\mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \int_X \mathbb{1}_A(\boldsymbol{x}) \,\mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \mathrm{P}_{\mathbf{x}}(A), \end{aligned} \tag{4.6}$$

i.e. that in the limit $\epsilon \to 0$, $\mathbf{y}$ has the same distribution as $\mathbf{x}$. Intuitively, as we decrease the tolerance ϵ we increasingly tightly constrain $\mathbf{y}$ and $\mathbf{x}$ to have similar distributions. Two common choices of kernels satisfying (4.4) and (4.5) are the *uniform ball* and *Gaussian* kernels which respectively have densities

$$k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\Gamma(\frac{N_\mathbf{x}}{2} + 1)}{\pi^{\frac{N_\mathbf{x}}{2}} \epsilon^{N_\mathbf{x}}} \mathbb{1}_{[0,\epsilon]}(\|\boldsymbol{y} - \boldsymbol{x}\|_2) \qquad \text{(uniform ball)}, \tag{4.7}$$

$$\text{and} \quad k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{x}, \epsilon^2 \mathbf{I}\right) \qquad \text{(Gaussian)}. \tag{4.8}$$

The marginal distribution of $\mathbf{y}$ can be written $\forall A \in \mathscr{B}(X)$ as

$$\mathsf{P}_\mathbf{y}(A) = \int_X \mathsf{P}_{\mathbf{y}|\mathbf{x}}(A \mid \boldsymbol{x})\, \mathsf{P}_\mathbf{x}(\mathrm{d}\boldsymbol{x}) = \int_A \int_X k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathsf{P}_\mathbf{x}(\mathrm{d}\boldsymbol{x})\, \mathrm{d}\boldsymbol{y}. \tag{4.9}$$

Therefore $\mathsf{P}_\mathbf{y}$ has a density with respect to the Lebesgue measure

$$\mathsf{p}_\mathbf{y}(\boldsymbol{y}) = \int_X k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathsf{P}_\mathbf{x}(\mathrm{d}\boldsymbol{x}) = \int_{X \times Z} k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z). \tag{4.10}$$

The density $\mathsf{p}_\mathbf{y}$ exists irrespective of whether $\mathsf{P}_\mathbf{x}$ has a density with respect to the Lebesgue measure (it may not for example if $\mathsf{P}_\mathbf{x}$ only has support on a sub-manifold of X). Using this definition of the density $\mathsf{p}_\mathbf{y}$ we have that for any measurable function $f : Z \to F$ of the unobserved variables and $\forall A \in \mathscr{B}(X)$ that

$$\begin{aligned} \int_{A \times Z} f(z)\, \mathsf{P}_{\mathbf{y},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}z) &= \int_{A \times X \times Z} f(z)\, \mathsf{P}_{\mathbf{y},\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \int_{X \times Z} \int_A f(z)\, k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{d}\boldsymbol{y}\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \int_A \int_{X \times Z} f(z)\, k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)\, \mathrm{d}\boldsymbol{y}. \end{aligned} \tag{4.11}$$

Using that $\mathsf{P}_\mathbf{y}$ has a density $\mathsf{p}_\mathbf{y}$ with respect to the Lebesgue measure, and that we can safely ignore the set for which $\mathsf{p}_\mathbf{y}(\boldsymbol{y}) = 0$ when integrating against $\mathsf{P}_\mathbf{y}$ as it is zero-measure, we have that $\forall A \in \mathscr{B}(X)$

$$\begin{aligned} &\int_{A \times Z} f(z)\, \mathsf{P}_{\mathbf{y},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}z) = \\ &\qquad \int_A \frac{1}{\mathsf{p}_\mathbf{y}(\boldsymbol{y})} \int_{X \times Z} f(z)\, k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)\, \mathsf{P}_\mathbf{y}(\mathrm{d}\boldsymbol{y}). \end{aligned} \tag{4.12}$$

Comparing this to the definition of the conditional expectation from Chapter 1 (1.30) therefore we have $\forall \boldsymbol{y} \in X : \mathrm{p}_{\mathbf{y}}(\boldsymbol{y}) > 0$

$$\begin{aligned}\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{X \times Z} f(z)\, k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \frac{\int_{X \times Z} f(z)\, k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)}{\int_{X \times Z} k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)}.\end{aligned} \tag{4.13}$$

For the case of a model in which $\mathrm{P}_{\mathbf{z}}$ has a density $\mathrm{p}_{\mathbf{z}}$ with respect to the Lebesgue measure, then if we use $f = \mathbb{1}_A$ for $A \in \mathscr{B}(Z)$ in (4.13) and the definition of a regular conditional distribution in (1.32) we have

$$\mathrm{P}_{\mathbf{z}|\mathbf{y}}(A \mid \boldsymbol{y}) = \int_A \frac{\int_X k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \mid z)\, \mathrm{p}_{\mathbf{z}}(z)}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\, \mathrm{d}z. \tag{4.14}$$

In this case the regular conditional distribution $\mathrm{P}_{\mathbf{z}|\mathbf{y}}$ has a conditional density $\mathrm{p}_{\mathbf{z}|\mathbf{y}}$ with respect to the Lebesgue measure,

$$\mathrm{p}_{\mathbf{z}|\mathbf{y}}(z \mid \boldsymbol{y}) = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_X k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \mid z)\, \mathrm{p}_{\mathbf{z}}(z). \tag{4.15}$$

In reference to terminology of Bayesian inference, the density $\mathrm{p}_{\mathbf{z}|\mathbf{y}}$ is termed the ABC posterior density, and therefore conditional expectations of the form of (4.13) which correspond to an integral with respect to this ABC posterior, are termed ABC posterior expectations.

We now consider how $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ is related to the conditional expectation we are interested in evaluating $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$. If we assume that $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ is absolutely continuous with respect to the Lebesgue measure with density $\mathrm{p}_{\mathbf{x},\mathbf{z}}$, using (4.5) we have that $\forall \boldsymbol{y} \in X : \mathrm{p}_{\mathbf{x}}(\boldsymbol{y}) > 0$

$$\begin{aligned}\lim_{\epsilon \to 0} \mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \lim_{\epsilon \to 0} \frac{\int_Z f(z) \int_X k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, z)\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}z}{\int_Z \int_X k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, z)\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}z} \\ &= \frac{\int_Z f(z)\, \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{y}, z)\, \mathrm{d}z}{\int_Z \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{y}, z)\, \mathrm{d}z} = \mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}].\end{aligned}$$

We therefore have that ABC posterior expectations $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ converge as $\epsilon \to 0$ to the exact posterior expectations we wish to be able to estimate $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$. Note this result requires that $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ is absolutely continuous with respect to the Lebesgue measure.

Crucially from a computational perspective the numerator and denominator of (4.13) both are expectations of known functions of $\mathbf{x}$ and $\mathbf{z}$,

$$\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\big] = \frac{\mathbb{E}\big[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y} \mid \mathbf{x})\big]}{\mathbb{E}\big[k_\epsilon(\boldsymbol{y} \mid \mathbf{x})\big]}. \tag{4.16}$$

Generating Monte Carlo estimates of these expectations only requires us to be able to generate samples from $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ without any requirement to be able to evaluate densities and therefore can be achieved in the implicit generative models of interest.

We can therefore estimate $\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\big]$ by generating a set of independent pairs of random vectors $\{\mathbf{x}_s,\, \mathbf{z}_s\}_{s=1}^S$ from $\mathrm{P}_{\mathbf{x},\mathbf{z}}$[3] and computing Monte Carlo estimates of the numerator and denominator in (4.16), which gives the following estimator

$$\hat{\mathrm{f}}_{S,\epsilon} = \frac{\sum_{s=1}^S (f(\mathbf{z}_s)\, k_\epsilon(\boldsymbol{y} \mid \mathbf{x}_s))}{\sum_{s=1}^S (k_\epsilon(\boldsymbol{y} \mid \mathbf{x}_s))}. \tag{4.17}$$

This directly corresponds to an importance sampling estimator for expectations with respect to $\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ using $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ as the proposal distribution. Therefore if both $f(\mathbf{z})\, k_\epsilon(\boldsymbol{y} \mid \mathbf{x})$ and $k_\epsilon(\boldsymbol{y} \mid \mathbf{x})$ have finite variance, then the estimator $\hat{\mathrm{f}}_{S,\epsilon}$ will be consistent,

$$\lim_{S\to\infty} \mathbb{E}\Big[\hat{\mathrm{f}}_{S,\epsilon}\Big] = \mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\big]. \tag{4.18}$$

If the kernel used is the uniform ball kernel (4.7), the estimator can be manipulated in to a particularly intuitive form

$$\hat{\mathrm{f}}_{S,\epsilon} = \frac{1}{|A|} \sum_{s\in A} (f(\mathbf{z}_s)) \;\text{ with }\; A = \{s \in \{1 \dots S\} : \|\boldsymbol{y} - \mathbf{x}_s\|_2 < \epsilon\} \tag{4.19}$$

which corresponds to averaging the values of sampled unobserved variables $\mathbf{z}_s$ where the corresponding samples of model observed variables $\mathbf{x}_s$ are within a distance ϵ of the observed data $\boldsymbol{y}$. The is the standard ABC rejection algorithm [92, 215, 232, 246, 259] , with A corresponding to the indices of the set of accepted samples, with the other samples being 'rejected' as the simulated observations $\mathbf{x}_s$ are more than a distance ϵ from the observed data $\boldsymbol{y}$. As an instance of a rejection sampler, conditioned on the acceptance set containing at least one sample, i.e. $|A| > 0$, (4.19) is an unbiased estimator for $\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\big]$.

3 As ABC is usually applied to directed models this is considered as generating $\mathbf{z}$ from a prior then simulating $\mathbf{x}$ given $\mathbf{z}$ however more generally we can sample from the joint.

Compared to the general rejection sampling scheme described in Algorithm 1 there is no probabilistic accept step. However the ratio of the target distribution $\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ to the proposal distribution $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ here is always equal to exactly zero or a constant $c(\epsilon)$ corresponding to the ratio of the volume of the ϵ radius ball, thus if we choose the bounding constant M in the rejection sampler as $c(\epsilon)$ the acceptance probabilities will always be zero or one and so no auxiliary u values are needed to perform a probabilistic accept. For more general kernels a rejection sampler with probabilistic accept is discussed in [262].

If we instead use a Gaussian (or other smoothly varying) kernel (4.8), then as for the general case for importance sampling, the estimator (4.17) is no longer unbiased. In the Gaussian kernel case we more highly weight samples if the simulated observed variables are closer to the data which may be viewed as preferable to equally weighting all values within a fixed tolerance as in ABC reject. However as it has support on all of X the Gaussian kernel also gives non-zero weights to all of the samples, with typically most making little contribution to the expectation which may be considered somewhat wasteful of computation versus the rejection scheme which creates a sparse set of samples to compute expectations over [19]. Kernels with bounded support but non-flat densities such as the *Epanechnikov kernel* [84] which has a parabolic density in a bounded region, offer some of the advantages of both the uniform ball and Gaussian kernels.

Irrespective of the kernel chosen, the estimate formed is only consistent for the ABC posterior expectation $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y}; \epsilon]$ rather than the actual posterior expectation $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$ we are directly interested in. As $\epsilon \to 0$, $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y}; \epsilon]$ converges to $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$, however for reject ABC we also have that as $\epsilon \to 0$ the proportion of accepted samples will tend to zero meaning that we need to expend increasing computational effort to get an estimator for $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y}; \epsilon]$ with a similar variance (which by a standard Monte Carlo argument is inversely proportional to the number of accepted samples). In the more general importance sampling case, although we do not explicitly reject any samples if using a kernel with unbounded support, we instead have that as $\epsilon \to 0$ that the kernel weightings in (4.17) will becoming increasingly dominated by the few samples closest to the observed data and so the contribution to the estimator (4.17) from all but a few will be negligible, again leading to an increasing number of samples being

needed to keep the variance of the estimator reasonable - i.e. the same issues which we discussed in the context of more general importance samplers in Chapter 2. For the exact $\epsilon = 0$ case we would only accept (or equivalently put non-zero weight on) samples for which $\mathbf{x}_s$ is exactly equal to $\boldsymbol{y}$. For $X \subseteq \mathbb{R}^{N_\mathbf{x}}$ if $\mathrm{P}_\mathbf{x}$ is absolutely continuous with respect to the Lebesgue measure, the event $\mathbf{x} = \boldsymbol{y}$ has zero measure under $\mathrm{P}_{\mathbf{x},\mathbf{z}}$[4] and so some degree of approximation due to non-zero ϵ is always required in practice in these simple Monte Carlo ABC schemes.

When the dimensionality of the observed variable vector $\mathbf{x}$ is high it quickly becomes impractical to reduce the variance of these naive Monte Carlo estimators for (4.13) to reasonable levels without using large ϵ which introduces significant approximation error. The ABC rejection method is well known to scale poorly with dimensionality due to curse of dimensionality effects [43, 163, 213]. Although often discussed specifically in the context of ABC, the issues faced are much the same as encountered when trying to use any simple rejection or importance sampling scheme to approximate expectations with respect to a probability distribution on a high-dimensional space. If the proposal distribution ($\mathrm{P}_{\mathbf{x},\mathbf{z}}$ here) is significantly more diffuse than the target distribution ($\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ here) an exponentially small proportion of the probability mass of the proposal distribution will lie in the typical set of the target distribution and so very few samples will be accepted or have non-negligible importance weights.

Rather than conditioning on the full observed data most ABC methods used in practice therefore instead use *summary statistics* to extract lower dimensional representations of the observed data [213]. A function $\mathbf{s} : X \to T$ is defined which computes summary statistics from simulated observed outputs $\mathbf{x}$ and observed data $\boldsymbol{y}$ with the dimensionality of the summaries, $\dim(T)$, typically much smaller than $N_\mathbf{x}$. The ABC posterior expectation is then computed using

$$\mathbb{E}\left[f(\mathbf{z}) \mid \mathbf{s} = \mathbf{s}(\boldsymbol{y});\, \epsilon\right] = \frac{\int_{X\times Z} f(\boldsymbol{z})\, k_\epsilon(\mathbf{s}(\boldsymbol{y}) \mid \mathbf{s}(\boldsymbol{x}))\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}{\int_{X\times Z} k_\epsilon(\mathbf{s}(\boldsymbol{y}) \mid \mathbf{s}(\boldsymbol{x}))\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}, \tag{4.20}$$

with now the variable conditioned on the T-valued variable $\mathbf{s}$ with

$$\mathrm{P}_{\mathbf{s}|\mathbf{x}}(A \mid \boldsymbol{x}) = \int_A k_\epsilon(\boldsymbol{s} \mid \mathbf{s}(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{s} \qquad \forall A \in \mathscr{B}(T),\, \boldsymbol{x} \in X. \tag{4.21}$$

4 In reality due the use of finite precision arithmetic the probability of generating values exactly consistent with data though vanishingly small is non-zero.

In general the statistics used will not be *sufficient* - the posterior distribution on $\mathbf{z}$ will differ when conditioning on $\mathbf{s}(\mathbf{x})$ compared to conditioning on $\mathbf{x}$ directly. By a data processing inequality argument we know that the mutual information between $\mathbf{z}$ and $\mathbf{s}(\mathbf{x})$ will be less than or equal to the mutual information between $\mathbf{z}$ and $\mathbf{x}$ therefore we would expect for the posterior distribution on $\mathbf{z}$ given $\mathbf{s}(\mathbf{x})$ to be less informative about $\mathbf{z}$ than the posterior distribution given $\mathbf{x}$ [12]. This means that even in the limit of $\epsilon \to 0$ estimates of the ABC summary statistics posterior expectation will generally not converge to the true posterior expectations of interest.

If a, b *and* c *are random variables and* $\mathbb{I}[\mathrm{a}, \mathrm{b}]$ *denotes the mutual information between* a *and* b *the data processing inequality states that if* $\mathrm{a} \perp \mathrm{c} \mid \mathrm{b}$ *then* $\mathbb{I}[\mathrm{a}, \mathrm{b}] \geq \mathbb{I}[\mathrm{a}, \mathrm{c}]$.

ABC methods therefore trade-off between the approximation errors introduced due to using summary statistics and a non-zero tolerance ϵ, and the Monte Carlo error from using a finite number of samples in the estimates. If informative summary statistics can be found then typically the approximation error can be kept to a more reasonable level compared to the conditioning on the full data without the Monte Carlo error becoming impractically large by allowing a smaller ϵ to be used while maintaining a reasonable accept rate. Finding informative low-dimensional summaries is often critical to getting ABC methods to work well in practice and there is a wide literature on methods for choosing summary statistics - see [213] and [44] for reviews.

In some cases use of summary statistics might not be viewed just as a computational convenience, but as a purposeful exercise in removing 'irrelevant' information from the data. For example if inferring plausible parameter values for a dynamic model of a system given observed sequences of the system state showing quasi-periodic behaviour, then we might view the phase of observed state sequences as an irrelevant artefact of the arbitrary point at which observations were started. In this case conditioning on the exact observed data could be viewed as over constraining the model to reproduce features of the data which are only incidental, and using summary statistics which are invariant to phase could be preferable to conditioning on the full data [263].

Similarly the introduction of a kernel in ABC need not be viewed as simply a method for making inference tractable, but instead as part of the modelling process [262]. In general we will expect any observed data to be subject to some amount of measurement noise (at the very least it will include some quantification noise) and so conditioning the model to reproduce the exact values of the data is not necessarily desir-

able. In this context we can consider $\mathbf{y}$ the noisy measured version of an underlying state $\mathbf{x}$ and the kernel $P_{\mathbf{y}|\mathbf{x}}$ as representing the measurement noise model. We might also instead view the kernel $P_{\mathbf{y}|\mathbf{x}}$ as accounting for the mismatch between our proposed model for how the observed values are generated and the true data generating process [222, 262]. In both these cases we could then consider ϵ as a further unobserved variable to be inferred.

These examples demonstrate that in some cases there may be a modelling motivation for introducing summary statistics or a 'noise' kernel. In practice however the summary statistics and tolerance ϵ seem to be more typically chosen on grounds of computational tractability [163, 213, 226]. Therefore inference methods which are able to maintain computational tractability when conditioning on higher-dimensional summaries or in some cases all observations, and when using smaller tolerance ϵ values are of significant practical interest.

4.5 ABC MCMC METHODS

The ABC inference methods considered so far correspond to simple Monte Carlo approaches that we previously claimed in Chapter 2 scale poorly to large complex probabilistic models. It is natural to consider therefore whether more scalable approximate inference methods can be applied instead. In this section we will discuss an ABC MCMC method [165, 240], which corresponds to an instance of the pseudo-marginal framework introduced in Chapter 3. The methods we propose in the following section are intended to address some of the shortcomings of this existing approach.

There has also been a significant amount of work on developing other more scalable ABC inference schemes, with in particular methods based on *sequential Monte Carlo* (SMC) [20, 72, 241, 251] having achieved significant empirical success. Typically however ABC SMC approaches make use of ABC MCMC moves as part of the overall algorithm therefore improved MCMC methods are also of direct relevance to those frameworks. More recently there has also been several approaches proposed for using optimisation-based approximate inference schemes in an ABC setting, including expectation propagation [15] and variational methods [180, 253]. These offer an interesting alternative to the standard

Monte Carlo based approaches, and the variational methods in particular share significant aspects with some of the ideas proposed here.

As is standard in ABC methods, the ABC MCMC approach proposed in [165] is targeted at directed generative models where the unobserved variables have a known marginal density $p_{\mathbf{z}}$ (the prior) but where we can only generate samples from the conditional distribution $P_{\mathbf{x}|\mathbf{z}}$ (the likelihood). The method constructs a Metropolis–Hastings transition operator which leaves the ABC posterior distribution $P_{\mathbf{z}|\mathbf{y}}$ invariant. We cannot evaluate the ABC posterior density, defined in (4.15), as it involves an intractable integral with respect to $P_{\mathbf{x}|\mathbf{z}}$. As we can generate samples from $P_{\mathbf{x}|\mathbf{z}}$ we can however compute an unbiased and non-negative estimate of $p_{\mathbf{z},\mathbf{y}}(z, \boldsymbol{y})$ which is proportional to the posterior density up to an unknown normalising constant $p_{\mathbf{y}}(\boldsymbol{y})$. Specifically if $\boldsymbol{x}$ is generated from $P_{\mathbf{x}|\mathbf{z}}(\cdot \mid z)$ then $p_{\mathbf{z}}(z)\, k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})$ is an unbiased estimate for $p_{\mathbf{z},\mathbf{y}}(z, \boldsymbol{y})$. We can therefore use this estimate of the (unnormalised) ABC posterior density within a pseudo-marginal Metropolis–Hastings transition as described in Algorithm 8, a proposal distribution used to generate perturbative updates to the unobserved variables $\mathbf{z}$ and these proposed updates accepted or rejected based on a Metropolis–Hastings accept ratio calculated using the density estimates.

By making small changes to the unobserved variables $\mathbf{z}$ and so making use of information from the previous state about plausible values for $\mathbf{z}$ under the ABC posterior $P_{\mathbf{z}|\mathbf{y}}$ rather than independently sampling them from $P_{\mathbf{z}}$ as in the simpler Monte Carlo schemes, ABC MCMC can often increase efficiency in generative models with large numbers of unobserved variables to infer [240]. This potential improved efficiency comes at a cost of introducing the usual difficulties associated with MCMC methods, with successive samples now dependent and it challenging to monitor convergence of the chain. Further as with the pseudo-marginal Metropolis–Hasting chains encountered in the previous chapter, ABC MCMC chains can be prone to 'sticking' pathologies, suffering long series of rejections. This can be considered a symptom of the density estimator being high variance as in the previous discussion of pseudo-marginal methods in Chapter 3, however it is instructive to consider more specifically the cause of the issue in this setting.

Though we propose small updates to $\mathbf{z}$ we independently sample proposed simulated observations $\mathbf{x}$ from $P_{\mathbf{x}|\mathbf{z}}$ in each transition to calculate the density estimate. The generated $\mathbf{x}$ values do not take in to ac-

count the observed data $\boldsymbol{y}$ and when the number of observed variables is high, typically the distance between the simulated observations and observed data will be high. This leads to the kernel term $k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x})$ in the density estimate at the new proposed $\mathbf{z}$ values being very small (or zero in the case of a uniform ball kernel) and so low probability of accepting the proposed $\mathbf{z}$ values, even if they are plausible under the ABC posterior distribution $\mathrm{P}_{\mathbf{z}|\mathbf{y}}$. Therefore though pseudo-marginal based ABC MCMC methods can help improve the scalability of ABC to models with larger numbers of unobserved variables $\mathbf{z}$, they still typically are limited in the dimensionality of the observed variables $\mathbf{x}$ that can be tractably supported and will usually require dimensionality reduction of the observations with summary statistics.

4.6 ABC INFERENCE IN THE INPUT SPACE

To try to overcome some of the limitations of the standard ABC MCMC approach, we now consider reparameterising the inference problem using the formulation of a generative model as a deterministic transformation of random inputs introduced in Definition 4.1 in Section 4.2. For a generative model $(U, \mathcal{F}, \rho, \mu, \mathrm{g}_\mathbf{x}, \mathrm{g}_\mathbf{z})$ for observed variables $\mathbf{x}$ and unobserved variables $\mathbf{z}$, the ABC posterior expectation (4.13) can be reparameterised using the *Law of the unconscious statistician* (1.27) as

$$\begin{aligned}\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \frac{1}{\mathrm{p}_\mathbf{y}(\boldsymbol{y})}\mathbb{E}[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y} \mid \mathbf{x})] \\ &= \frac{1}{\mathrm{p}_\mathbf{y}(\boldsymbol{y})}\mathbb{E}[f(\mathrm{g}_\mathbf{z}(\mathbf{u}))\, k_\epsilon(\boldsymbol{y} \mid \mathrm{g}_\mathbf{x}(\mathbf{u}))] \\ &= \frac{1}{\mathrm{p}_\mathbf{y}(\boldsymbol{y})}\int_U f \circ \mathrm{g}_\mathbf{z}(\boldsymbol{u})\, k_\epsilon(\boldsymbol{y} \mid \mathrm{g}_\mathbf{x}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mu(\mathrm{d}\boldsymbol{u}).\end{aligned} \tag{4.22}$$

Crucially this reparameterisation takes the form of an integral of a function $f \circ \mathrm{g}_\mathbf{z}$ against an explicitly defined probability density

$$\pi_\epsilon(\boldsymbol{u}) = \frac{1}{\mathrm{p}_\mathbf{y}(\boldsymbol{y})} k_\epsilon(\boldsymbol{y} \mid \mathrm{g}_\mathbf{x}(\boldsymbol{u}))\, \rho(\boldsymbol{u}), \tag{4.23}$$

that we can evaluate up to an unknown normalising constant $\mathrm{p}_\mathbf{y}(\boldsymbol{y})$. This is the typical setting for approximate inference in a probabilistic model, and so allows applying standard variants of MCMC methods to inference in generative models. Although the target distribution described by the density (4.23) will typically have a complex geometry

due to the complex non-linear dependency on the simulated observed variables $\mathbf{x}$ on the inputs $\mathbf{u}$ via the generator function $\mathbf{g}_{\mathbf{x}}$, we are free to use scalable MCMC methods which are better able to cope with such complex geometries.

Importantly as all variables in the model are now being perturbatively updated, we can avoid the curse of dimensionality effects that limit the performance of standard ABC approaches when conditioning on large numbers of observed variables. We will however still be susceptible to the standard challenges of ensuring convergence when applying MCMC methods in complex high-dimensional target distributions. The dimensionality of the observations and choice of kernel k_ϵ and tolerance will affect the geometry of the target density and so will have therefore have a bearing on how well chains are able to mix.

For generator functions which are not differentiable or it is non-trivial to calculate derivatives using AD, the slice-sampling methods discussed in Chapter 2 offer a relatively black-box approach to constructing transition operators which leave the distribution defined by the density (4.23) invariant. The adaptive nature of slice sampling updates will potentially give improved performance in target densities with a complex geometry than simpler approaches such as random-walk Metropolis. To apply slice-sampling to inference in a generative model we only require that the inputs $\mathbf{u}$ are real-valued (which does not preclude the inclusion of discrete latent variables in the generative model as these will typically be generated by transforms of standard uniform random inputs) and that we are able evaluate the generator $\mathbf{g}_{\mathbf{x}}$ and density ρ of the input distribution $\mathsf{P}_{\mathbf{u}}$. If the inputs $\mathbf{u}$ are marginally normally distributed, a natural choice is the elliptical slice sampling algorithm described in Algorithm 5. If the inputs $\mathbf{u}$ instead marginally have a standard uniform distribution, then we can instead apply the reflective variant of linear slice sampling described in Chapter 3, and for other input distributions with unbounded support we can use the standard linear slice sampling method (Algorithm 4) without reflections. If different subsets of the inputs have distinct properties, we may wish to combine different slice-sampling transition operators which each update only a subset of the inputs.

For the case of differentiable generative models as defined in Definition 4.2, a natural choice of MCMC method is Hamiltonian Monte Carlo. In general in this case it will be desirable to use a Gaussian kernel k_ϵ as

this is a smooth function of both inputs and if ρ has unbounded support so will the target density (4.23). For a Gaussian kernel, the potential energy function corresponding to (4.23) is

$$\phi(\boldsymbol{u}) = \frac{1}{2\epsilon^2}(\boldsymbol{x} - \mathbf{g}_{\mathbf{x}}(\boldsymbol{u}))^{\mathsf{T}}(\boldsymbol{x} - \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})) - \log \rho(\boldsymbol{u}), \tag{4.24}$$

The potential energy combines a term favouring inputs $\mathbf{u}$ which generate outputs close to the observed data $\boldsymbol{x}$ and prior term favouring input values which are plausible under $\mathsf{P}_{\mathbf{u}}$. The gradient of (4.24) is

$$\nabla\phi(\boldsymbol{u}) = \frac{1}{\epsilon^2}(\mathbf{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x})^{\mathsf{T}}\mathbf{J}_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u}) - \frac{1}{\rho(\boldsymbol{u})}\nabla\rho(\boldsymbol{u}). \tag{4.25}$$

Although this expression involves the generator Jacobian $\mathbf{J}_{\mathbf{g}_{\mathbf{x}}}$, in practice by using reverse-mode AD we can evaluate the gradient without explicitly evaluating the full Jacobian as it only appears as a matrix-vector product. Typically the density ρ will have a simple form e.g. standard normal $\mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$ in which case the main complexity in the target density arises from the term due to the Gaussian kernel k_ϵ and generator function $\mathbf{g}_{\mathbf{x}}$; this term puts high density (and low potential energy) on inputs close to the pre-image $\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})$ of the observed data $\boldsymbol{x}$ under the generator function $\mathbf{g}_{\mathbf{x}}$. For small ϵ this will mean the distribution in the input space is increasingly tightly concentrated in a narrow 'ridge' around the manifold embedded in the input space corresponding to $\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})$[5]. Although the gradient-based Hamiltonian dynamic is able to propose moves which remain within this high-density region, the strong gradients normal to the manifold tends to produce trajectories which oscillate back and forth across the ridge, limiting the motion tangential to the manifold and requiring a small integrator step-size for stability; this is illustrated in a simple model with a two dimensional input space in Figure 4.3. In practice this tends to limit how small ϵ can be made. We consider an alternative HMC approach for directly defining a dynamic on the $\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})$ manifold in a later section.

There is a direct link between the reparameterisation of a generative model used here to allow application of alternative MCMC transition operators to ABC inference and the reparameterisation of the density estimators used in pseudo-marginal MCMC methods proposed in Chapter 3. For the common special case (and typical ABC setting) of a directed

5 Such 'ridged densities' also arise in other contexts; a discussion of some of the geometric and computational issues involved in simulating Markov chains in such distributions is given in [27].

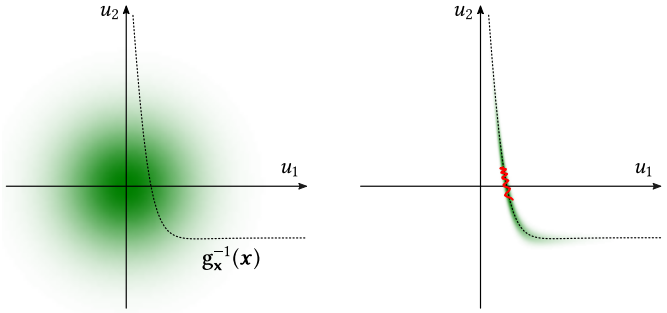


Figure 4.3.: Illustration of oscillatory behaviour in HMC trajectories when using an ABC target density (4.23) in the input space to a generative model. The left axis shows the two-dimensional input space U of a toy differentiable generative model with a Gaussian input density ρ (green shading). The dashed curve shows the one-dimensional manifold corresponding to the fibre under the generator function $\mathrm{g}_{\mathbf{x}}$ of an observed output x. The right axis shows the same input space with now the green shading showing the density proportional to $k_\epsilon(x \mid \mathrm{g}_{\mathbf{x}}(\boldsymbol{u}))\,\rho(\boldsymbol{u})$ with a Gaussian k_ϵ. The red curve shows a simulated HMC trajectory using this density as the target: the large magnitude density gradients normal to the manifold cause high-frequency oscillations and slows movement along the manifold (which corresponds to variation in the latent variable z).

generative model with a tractable marginal density on the unobserved variables $\mathrm{p}_{\mathbf{z}}$, we have that

$$\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\big] = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_Z \int_{U_2} f(z)\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}|\mathbf{z}}(z,\, \boldsymbol{u}_2))\, \mathrm{p}_{\mathbf{z}}(z) \rho_2(\boldsymbol{u}_2)\, \mathrm{d}\boldsymbol{u}_2\, \mathrm{d}z \tag{4.26}$$

and so the target density for inference is

$$\pi_\epsilon(z, \boldsymbol{u}_2) = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} k_\epsilon(\boldsymbol{y} \mid \mathrm{g}_{\mathbf{x}|\mathbf{z}}(z,\, \boldsymbol{u}_2))\, \mathrm{p}_{\mathbf{z}}(z)\, \rho_2(\boldsymbol{u}_2). \tag{4.27}$$

Identifying $Z = \mathrm{p}_{\mathbf{y}}(\boldsymbol{y})$, $\varepsilon(z, \boldsymbol{u}_2) = k_\epsilon(\boldsymbol{y} \mid \mathrm{g}_{\mathbf{x}|\mathbf{z}}(z,\, \boldsymbol{u}_2))\, \mathrm{p}_{\mathbf{z}}(z)$ and $\rho = \rho_2$, this has directly the same form as (3.8). The main difference in the target density (4.23) is therefore that all of the random inputs, both those being used to generate the unobserved variables $\mathbf{z}$ and to generate the observed variables $\mathbf{x}$ given the unobserved variables, are treated equivalently. Although a minor difference, the formulation used in this Chapter helps clarify a geometric interpretation of the distribution induced in the input space to a generator when conditioning on observed values of its outputs which we will exploit in the following section.

4.7 ASYMPTOTICALLY EXACT INFERENCE IN DIFFERENTIABLE GENERATIVE MODELS

In the reparameterising inference in terms of evaluating an integral over the input space we have still so far required the definition of a kernel k_ϵ and tolerance ϵ and the integral being estimated is the approximate expectation (4.13) rather than target conditional expectation $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x}]$ we are directly interested in. We now consider in the specific case of differentiable generative models how to perform inference without introducing an ABC kernel.

We begin an initial intuition for the approach, by considering taking the limit of $\epsilon \to 0$ in the integral (4.22) corresponding to evaluating the ABC conditional expectation in the generator input space. We previously showed in (4.4) that the approximate expectation $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ converges as $\epsilon \to 0$ to the conditional expectation of interest $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$, providing that the implicit distribution of the observed variables in the generative model $\mathsf{P}_\mathbf{x}$ is absolutely continuous with respect to the Lebesgue measure with density $\mathsf{p}_\mathbf{x}$. Informally for kernels meeting the conditions (4.4) and (4.5), in the limit of $\epsilon \to 0$ the kernel density $k_\epsilon(\boldsymbol{y};\, \mathsf{g}_\mathbf{x}(\boldsymbol{u}))$ tends to a Dirac delta $\delta(\boldsymbol{y} - \mathsf{g}_\mathbf{x}(\boldsymbol{u}))$ and so

$$\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}] = \lim_{\epsilon \to 0} \mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] \tag{4.28}$$

$$\simeq \frac{\int_U f \circ \mathsf{g}_\mathbf{z}(\boldsymbol{u})\, \delta(\boldsymbol{y} - \mathsf{g}_\mathbf{x}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}{\int_U \delta(\boldsymbol{y} - \mathsf{g}_\mathbf{x}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}. \tag{4.29}$$

The fibre $\mathbf{m}^{-1}(\boldsymbol{y})$ *of an element* $\boldsymbol{y} \in Y$ *under a map* $\mathbf{m} : X \to Y$ *is the pre-image of the singleton set* $\{\boldsymbol{y}\}$ *under* $\mathbf{m}$.

The Dirac delta term restricts the integral across the input space U to an embedded, $M - N_\mathbf{x}$ dimensional, implicitly-defined manifold corresponding to the fibre of $\boldsymbol{y}$ under $\mathsf{g}_\mathbf{x}$, $\mathsf{g}_\mathbf{x}^{-1}(\boldsymbol{y}) \equiv \{\boldsymbol{u} \in U : \mathsf{g}_\mathbf{x}(\boldsymbol{u}) = \boldsymbol{y}\}$. It is not necessarily immediately clear however how to define the required density on that manifold for arbitrary non-injective $\mathsf{g}_\mathbf{x}$.

In differentiable generative models we can however use a derivation equivalent to that given by Diaconis, Holmes and Shahshahani in [75] for the conditional density on a manifold to find an expression for the conditional expectation consistent with definition given in (1.30). The key result we use is a formula from geometric measure-theory, Federer's *co-area formula* [86, §3.2.12]. This generalises Fubini's theorem for iterated integrals on spaces defined by a Cartesian product to more general partitions of a space.

THEOREM 4.1 (Co-area formula): *Let $V \subseteq \mathbb{R}^L$ and $W \subseteq \mathbb{R}^K$ with $L \geq K$, and let $\mathbf{m} : V \to W$ be a Lipschitz function and $\mathrm{h} : V \to \mathbb{R}$ a Lebesgue measurable function. Then*

$$\int_V \mathrm{h}(\boldsymbol{v})\, D_{\mathbf{m}}(\boldsymbol{v})\, \lambda^L(\mathrm{d}\boldsymbol{v}) = \int_W \int_{\mathbf{m}^{-1}(\boldsymbol{w})} \mathrm{h}(\boldsymbol{v})\, \hbar^{L-K}(\mathrm{d}\boldsymbol{v})\, \lambda^K(\mathrm{d}\boldsymbol{w}) \tag{4.30}$$

with $\hbar^{L-K}$ denoting the $L-K$-dimensional Hausdorff measure and $D_{\mathbf{m}}(\boldsymbol{v})$ denoting the generalised Jacobian determinant for 'wide' rectangular Jacobian matrices

$$D_{\mathbf{m}}(\boldsymbol{v}) \equiv \left|\mathbf{J}_{\mathbf{m}}(\boldsymbol{u})\mathbf{J}_{\mathbf{m}}(\boldsymbol{u})^{\mathsf{T}}\right|^{\frac{1}{2}}. \tag{4.31}$$

The K-dimensional Hausdorff measure $\hbar^K$ on $\mathbb{R}^N$ for $K \in \mathbb{N}$, $0 < K < N$ formalises a measure of the 'volume' of K-dimensional submanifolds of $\mathbb{R}^N$ - e.g. for $K = 1$ it corresponds to the length of a curve in $\mathbb{R}^N$. Additionally $\hbar^N = \lambda^N$ and $\hbar^0 = \#$.

More immediately applicable in our case is the following corollary.

COROLLARY 4.1: *If Q is a probability measure on V with density q with respect to the Lebesgue measure λ^L and $\mathbf{J}_{\mathbf{m}}$ is full row-rank Q-almost everywhere, then for Lebesgue measurable $\mathrm{h}' : V \to \mathbb{R}$*

$$\int_V \mathrm{h}'(\boldsymbol{v})\, q(\boldsymbol{v})\, \lambda^L(\mathrm{d}\boldsymbol{v}) = \int_W \int_{\mathbf{m}^{-1}(\boldsymbol{w})} \mathrm{h}'(\boldsymbol{v})\, q(\boldsymbol{v})\, D_{\mathbf{m}}(\boldsymbol{v})^{-1}\, \hbar^{L-K}(\mathrm{d}\boldsymbol{v})\, \lambda^K(\mathrm{d}\boldsymbol{w}). \tag{4.32}$$

This can be shown by setting $h(\boldsymbol{v}) = h'(\boldsymbol{v})\, q(\boldsymbol{v}) D_{\mathrm{f}}(\boldsymbol{v})^{-1}$ in (4.30) and using the equivalence of Lebesgue integrals in which the integrand differs only zero-measure sets.

We first show that $\mathsf{P}_{\mathbf{x}}$ has a density $\mathrm{p}_{\mathbf{x}} = \frac{\mathrm{d}\mathsf{P}_{\mathbf{x}}}{\mathrm{d}\lambda^{N_{\mathbf{x}}}}$.

PROPOSITION 4.1 (Change of variables in a differentiable generative model): *For a differentiable generative model $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_{\mathbf{x}}, \mathbf{g}_{\mathbf{z}})$ as defined in Definition 4.2, then if the generator $\mathbf{g}_{\mathbf{x}}$ is Lipschitz and the Jacobian $\mathbf{J}_{\mathbf{g}_{\mathbf{x}}}$ has full row-rank $\mathsf{P}_{\mathbf{u}}$-almost everywhere, the observed vector $\mathbf{x}$ has a density with respect to the Lebesgue measure satisfying*

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \int_{\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})} \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u}) \qquad \forall \boldsymbol{x} \in X. \tag{4.33}$$

Proof. From Definition 4.2 we have that $\mathbf{x} = \mathbf{g}_{\mathbf{x}}(\mathbf{u})$ and $\frac{\mathrm{d}\mathsf{P}_{\mathbf{u}}}{\mathrm{d}\lambda^M} = \rho$ and so

$$\mathsf{P}_{\mathbf{x}}(A) = \int_U \mathbb{1}_A \circ \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, \lambda^M(\mathrm{d}\boldsymbol{u}) \qquad \forall A \in \mathscr{B}(X).$$

As $\mathbf{g}_\mathbf{x}$ is Lipschitz and $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ has full row-rank $\mathsf{P}_\mathbf{u}$-almost everywhere we can apply Corollary 4.1, and so we have that $\forall A \in \mathscr{B}(X)$

$$\mathsf{P}_\mathbf{x}(A) = \int_X \int_{\mathbf{g}_\mathbf{x}^{-1}(\boldsymbol{x})} \mathbb{1}_A \circ \mathbf{g}_\mathbf{x}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_\mathbf{x}}(\mathrm{d}\boldsymbol{u})\, \lambda^{N_\mathbf{x}}(\mathrm{d}\boldsymbol{x}).$$

The term $\mathbb{1}_A \circ \mathbf{g}_\mathbf{x}(\boldsymbol{u})$ inside the inner integral is equal to $\mathbb{1}_A(\boldsymbol{x})$ across all points in the fibre $\mathbf{g}_\mathbf{x}^{-1}(\boldsymbol{x})$ being integrated across and so can be taken outside the inner integral to give

$$\begin{aligned}\mathsf{P}_\mathbf{x}(A) &= \int_X \mathbb{1}_A(\boldsymbol{x}) \int_{\mathbf{g}_\mathbf{x}^{-1}(\boldsymbol{x})} \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_\mathbf{x}}(\mathrm{d}\boldsymbol{u})\, \lambda^{N_\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \int_A \int_{\mathbf{g}_\mathbf{x}^{-1}(\boldsymbol{x})} \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_\mathbf{x}}(\mathrm{d}\boldsymbol{u})\, \lambda^{N_\mathbf{x}}(\mathrm{d}\boldsymbol{x}).\end{aligned}$$

By definition the density $\mathsf{p}_\mathbf{x}$ of a probability measure $\mathsf{P}_\mathbf{x}$ with respect to the Lebesgue measure $\lambda^{N_\mathbf{x}}$ satisfies

$$\mathsf{P}_\mathbf{x}(A) = \int_A \mathsf{p}_\mathbf{x}(\boldsymbol{x})\, \lambda^{N_\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \qquad \forall A \in \mathscr{B}(X)$$

$\therefore$ $\mathsf{P}_\mathbf{x}$ has a density corresponding to (4.33) with respect to $\lambda^{N_\mathbf{x}}$. □

This is a generalisation of the change of variables formula under a diffeomorphism encountered previously in Chapter 1. We now derive a result for the conditional expectation.

PROPOSITION 4.2 (Conditional expectations in a differentiable generative model): *For a differentiable generative model* $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ *as defined in Definition 4.2 and satisfying the conditions in Proposition 4.1, then for Lebesgue measurable functions* $f : X \to \mathbb{R}$ *and* $\boldsymbol{x} \in X$ *such that* $\mathsf{p}_\mathbf{x}(\boldsymbol{x}) > 0$ *we have that*

$$\begin{aligned}&\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{x}] = \\ &\qquad \frac{1}{\mathsf{p}_\mathbf{x}(\boldsymbol{x})} \int_{\mathbf{g}_\mathbf{x}^{-1}(\boldsymbol{x})} f \circ \mathbf{g}_\mathbf{z}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_\mathbf{x}}(\mathrm{d}\boldsymbol{u}).\end{aligned} \tag{4.34}$$

Proof. Restating the general definition for a conditional expectation from Chapter 1, we need to find a measurable function $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x}] : X \to \mathbb{R}$ which $\forall A \in \mathscr{B}(X)$ satisfies

$$\int_A \mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{x}]\, \mathsf{P}_\mathbf{x}(\mathrm{d}\boldsymbol{x}) = \int_{A \times Z} f(z)\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z),$$

with this uniquely defining the conditional expectation up to $\mathsf{P}_{\mathbf{x}}$-null sets. Using $\mathbf{x} = \mathbf{g}_{\mathbf{x}}(\mathbf{u})$, $\mathbf{z} = \mathbf{g}_{\mathbf{z}}(\mathbf{u})$ and $\mathsf{p}_{\mathbf{u}} = \rho$ we have that $\forall A \in \mathscr{B}(X)$

$$\int_{A\times Z} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) = \int_{U} \mathbb{1}_A \circ \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})\, f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, \lambda^M(\mathrm{d}\boldsymbol{u}).$$

Applying the co-area corollary (4.32) to the right-hand side and again noting the indicator term $\mathbb{1}_A \circ \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})$ is constant across the fibre being integrated on, we have that $\forall A \in \mathscr{B}(X)$

$$\begin{aligned}
&\int_{A\times Z} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) \\
&= \int_X \int_{\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})} \mathbb{1}_A \circ \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})\, f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{N_{\mathbf{x}}}(\mathrm{d}\boldsymbol{x}) \\
&= \int_A \int_{\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})} f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{N_{\mathbf{x}}}(\mathrm{d}\boldsymbol{x}).
\end{aligned}$$

Finally using that $\mathsf{P}_{\mathbf{x}}$ has a density $\mathsf{p}_{\mathbf{x}}$ with respect to the Lebesgue measure as shown in the previous proposition, we have that

$$\begin{aligned}
&\int_{A\times Z} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) = \\
&\qquad \int_A \frac{1}{\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})} \int_{\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})} f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \hbar^{M-N_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}).
\end{aligned}$$

Note that as we are integrating against the probability measure $\mathsf{P}_{\mathbf{x}}$ we can safely ignore the points for which $\mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) = 0$ as the set of all such points naturally has zero measure under $\mathsf{P}_{\mathbf{x}}$ and so does not contribute to integral. Comparing to the definition of the conditional expectation we have that (4.34) satisfies the definition. □

The expression derived for the conditional expectation has the form of an integral of function $f \circ \mathbf{g}_{\mathbf{z}}$ integrated against a density

$$\pi(\boldsymbol{u}) = \frac{1}{\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})} D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \rho(\boldsymbol{u}) \tag{4.35}$$

which we can evaluate up to an unknown normalising constant $\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})$. The key complicating factor is that the integral is now not across a Euclidean space, but an implicitly defined manifold corresponding to the fibre $\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})$. However if we can construct a Markov transition operator which has an invariant distribution with density (4.35) with respect to

the Hausdorff measure on the manifold, then we can use samples of the chain states $\{\mathbf{u}^{(s)}\}_{s=1}^{S}$ to compute an estimate

$$\hat{\mathrm{f}}_S = \frac{1}{S}\sum_{s=1}^{S}\left(\mathrm{f} \circ \mathrm{g}_{\mathbf{z}}\left(\mathbf{u}^{(s)}\right)\right) \tag{4.36}$$

which providing the chain is also aperiodic and irreducible will be a consistent estimator for $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{x}]$. Although constructing a Markov transition operator with the required properties is non-trivial, there is a significant body of existing work on methods for defining Markov chains on manifolds. We propose here to use a constrained Hamiltonian Monte Carlo method.

4.8 CONSTRAINED HAMILTONIAN MONTE CARLO

The HMC method introduced in Chapter 2 defined a transition operator which leaves a target distribution on a Euclidean space invariant. In our case the target distribution is defined on an implicitly defined manifold $\mathrm{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})$ embedded in a Euclidean space $U = \mathbb{R}^M$. Intuitively we can consider the manifold as representing the allowable configurations of mechanical system subject to a constraint. By simulating a constrained Hamiltonian dynamic we can therefore construct a HMC transition operator analogous to those discussed in Chapter 2 but that generates chains on an implicitly defined manifold rather than an unconstrained Euclidean space.

The use of constrained Hamiltonian dynamics within a MCMC method has been proposed by multiple authors. In the molecular dynamics literature, Hartmann and Schutte [124] and Lelièvre, Rousset and Stoltz [154] used simulated constrained Hamiltonian dynamics within a HMC framework to estimate free-energy profiles of molecular systems. Most relevantly for our case, Brubaker, Salzmann and Urtasun [49] proposed a constrained HMC algorithm for performing inference in target distributions defined on implicitly defined embedded manifolds. We will concentrate on the algorithm proposed in [49] here.

To simplify notation and emphasise the generality of the approach beyond our specific setting, we define the following notation for the vector *constraint function* on the system and corresponding Jacobian

$$\mathbf{c}(\boldsymbol{u}) = \mathrm{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x}, \qquad \mathbf{J}_{\mathbf{c}}(\boldsymbol{u}) = \mathbf{J}_{\mathrm{g}_{\mathbf{x}}}(\boldsymbol{u}). \tag{4.37}$$

The *constraint manifold* is then defined as the zero level-set of $\mathbf{c}$, in our case corresponding to the fibre of $\boldsymbol{x}$ under the generator $\mathbf{g}_{\mathbf{x}}$

$$C = \{\boldsymbol{u} \in \mathbb{R}^M : \mathbf{c}(\boldsymbol{u}) = \mathbf{0}\} = \mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x}). \tag{4.38}$$

Defining as in Chapter 2 the potential energy ϕ as the negative logarithm of the unnormalised target density and the kinetic energy as a quadratic form $\frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p}$ where $\boldsymbol{p}$ is a vector of momenta, the Hamiltonian for the constrained system can be written as

$$h(\boldsymbol{u}, \boldsymbol{p}) = \phi(\boldsymbol{u}) + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p} + \mathbf{c}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}, \tag{4.39}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrangian multipliers for the constraints. The constrained Hamiltonian dynamic is then defined by

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{M}^{-1}\boldsymbol{p}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\nabla\phi(\boldsymbol{u})^{\mathsf{T}} - \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}, \tag{4.40}$$

with the Lagrange multipliers taking values to ensure the system constraints are satisfied. In addition to the configuration constraint $\mathbf{c}(\boldsymbol{u}) = \mathbf{0}$ there is a corresponding implied constraint on the momenta $\boldsymbol{p}$ requiring that the configuration velocity $\boldsymbol{M}^{-1}\boldsymbol{p}$ is always tangential to the constraint manifold at the current configuration, or equivalently that the momenta are in the *tangent space* to the constraint manifold. The tangent space $\mathcal{T}_{\boldsymbol{u}}C$ at a configuration $\boldsymbol{u}$ is defined as

$$\mathcal{T}_{\boldsymbol{u}}C = \left\{\boldsymbol{p} \in \mathbb{R}^M : \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}\boldsymbol{p} = \mathbf{0}\right\}. \tag{4.41}$$

The complete set of valid configuration–momentum state pairs is termed the *tangent bundle* $\mathcal{T}C$ of the constraint manifold and defined as

$$\mathcal{T}C = \left\{\boldsymbol{u}, \boldsymbol{p} \in \mathbb{R}^M \times \mathbb{R}^M : \mathbf{c}(\boldsymbol{u}) = \mathbf{0},\ \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}\boldsymbol{p} = \mathbf{0}\right\}. \tag{4.42}$$

The solution at time t to the initial value problem defined by the ODEs (4.40) defines a flow map $\boldsymbol{\gamma}_t : \mathcal{T}C \to \mathcal{T}C$ between states in the tangent bundle of the constraint manifold. As with the unconstrained Hamiltonian dynamics encountered previously in Chapter 2, this flow map exactly conserves the Hamiltonian and is time-reversible. Further the flow map of the constrained dynamic is symplectic and conserves the volume element of the constraint manifold tangent bundle [152].

Importantly there exist symplectic integrators which can be used to approximate the constrained Hamiltonian dynamic flow map and which map between states exactly in the constraint manifold tangent bundle (modulo numerical error due to finite precision arithmetic). The approximate flow maps defined by these integrators are time-reversible and conserve the tangent bundle volume element. They also exhibit the bounded change in the Hamiltonian over simulated trajectories discussed previously for the unconstrained case in Chapter 2.

A popular symplectic numerical integrator for constrained Hamiltonian dynamics is the RATTLE method [6, 153]. This a generalisation of the leapfrog integrator encountered in our previous discussion of HMC in Chapter 2 with additional steps to project the states on to the tangent bundle of the constraint manifold. A RATTLE step is composed of three component maps. The first map is defined by

$$\begin{aligned}\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t}(\boldsymbol{u}, \boldsymbol{p}) &= \Big(\boldsymbol{u} + \delta t \boldsymbol{M}^{-1}(\boldsymbol{p} - \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}),\, \boldsymbol{p} - \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}\Big) \\ &\quad \text{solving for } \boldsymbol{\lambda} \text{ such that } \mathbf{c}\Big(\boldsymbol{u} + \delta t \boldsymbol{M}^{-1}(\boldsymbol{p} - \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda})\Big) = \mathbf{0}.\end{aligned} \tag{4.43}$$

This defines an approximate *geodesic* step on the constraint manifold: the configuration $\boldsymbol{u}$ is incremented in the direction of the current velocity $\boldsymbol{M}^{-1}\boldsymbol{p}$ and then the new configuration state projected back on to the constraint manifold by solving a non-linear system of equations for the Lagrange multipliers $\boldsymbol{\lambda}$.

The second component map updates the momenta with a 'kick' in the direction of the potential energy gradient

$$\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\delta t}(\boldsymbol{u}, \boldsymbol{p}) = \Big(\boldsymbol{u},\, \boldsymbol{p} - \delta t \nabla \phi(\boldsymbol{u})^{\mathsf{T}}\Big). \tag{4.44}$$

Though both $\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t}$ and $\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\delta t}$ steps will map between configurations in the constraint manifold (trivially in the case of $\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\delta t}$ as the configurations are kept fixed), the corresponding momenta will not be confined to the tangent spaces to the manifold. The final component map projects the momenta in to the tangent space of the constraint manifold at the current configuration. It is defined by

$$\begin{aligned}\hat{\boldsymbol{\gamma}}^{\mathrm{P}}(\boldsymbol{u}, \boldsymbol{p}) &= \Big(\boldsymbol{u},\, \boldsymbol{p} - \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}\Big) \\ &\quad \text{solving for } \boldsymbol{\lambda} \text{ such that } \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}(\boldsymbol{p} - \mathbf{J}_{\mathbf{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}) = \mathbf{0}.\end{aligned} \tag{4.45}$$

In this case the system of equation needing to be solved is linear and has an analytic solution, giving the following closed-form definition

$$\hat{\boldsymbol{\gamma}}^{\mathrm{P}}(\boldsymbol{u}, \boldsymbol{p}) = \left(\boldsymbol{u},\ \boldsymbol{p} - \mathrm{J_c}(\boldsymbol{u})^{\mathsf{T}}(\mathrm{J_c}(\boldsymbol{u})\boldsymbol{M}^{-1}\mathrm{J_c}(\boldsymbol{u})^{\mathsf{T}})^{-1}\mathrm{J_c}(\boldsymbol{u})\boldsymbol{M}^{-1}\boldsymbol{p}\right). \tag{4.46}$$

An overall RATTLE step is then defined by the composition

$$\hat{\boldsymbol{\gamma}}^{\mathrm{R}}_{\delta t} = \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}. \tag{4.47}$$

In practice the intermediate momentum projection steps $\hat{\boldsymbol{\gamma}}^{\mathrm{P}}$ are redundant [168] and so typically the momentum is only projected back in to the tangent space at the end of the step, giving the following update

$$\hat{\boldsymbol{\gamma}}^{\mathrm{R}}_{\delta t} = \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}. \tag{4.48}$$

Solving the non-linear constraint equations in the geodesic step $\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t}$ is computationally challenging, with closed form solutions generally not available and so an iterative approach required. Further the system of equations are not guaranteed to have a unique solution: if the step size δt is too large there can be multiple or no solutions [152]. It is important therefore to keep the step size small enough to avoid the iterative solver converging to an incorrect solution or not converging at all. Often the resulting step size will be smaller than required however in terms of controlling the Hamiltonian error over a simulated trajectory. An alternative to the standard RATTLE integrator is therefore to perform $N_g > 1$ inner geodesic steps $\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\frac{\delta t}{N_g}}$ for each outer pair of momentum kick steps $\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}$

$$\hat{\boldsymbol{\gamma}}^{\mathrm{G}}_{\delta t} = \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \left(\hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\frac{\delta t}{N_g}}\right)^{N_g} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}. \tag{4.49}$$

This *geodesic integrator* [150, 151] scheme can reduce the number of potential energy gradient evaluations required by using a larger step size for the momentum kick updates while still maintaining a sufficiently small step size to avoid convergence issues in the geodesic step.

Assuming the iterative solving of the projections to constraint manifold in the geodesic steps converge correctly, the approximate flow map defined by iterating RATTLE or geodesic integrator steps preserves the volume element of $\mathcal{T}C$ and is reversible under negation of the momenta. We can therefore use the composition of the approximate flow

map with a momentum reversal operator to define a volume-preserving involution between states in $\mathcal{T}C$. We can then use this involution as a proposal generating mechanism for a Metropolis accept step (2.39) to correct for the Hamiltonian error in the approximate flow map.

As in the standard HMC algorithm, Metropolis updates with approximate flow map proposals are interleaved with updates in which the momenta are independently resampled. To ensure the momenta remain in the tangent space $\mathcal{T}_{\boldsymbol{u}}C$ to the constraint manifold after generating new values from $\mathcal{N}(\mathbf{0}, \boldsymbol{M})$, the momenta are projected in to the tangent space using the projection operator defined in (4.46). The overall constrained HMC transition operator defined by this combination of momentum resampling and Metropolis accept step with a constrained dynamic proposal, leaves invariant the distribution with negative log density defined by the Hamiltonian in (4.39) on the constraint manifold tangent bundle $\mathcal{T}C$, and so marginally leaves the target distribution with density proportional to $\exp(-\phi(\boldsymbol{u}))$ on C invariant.

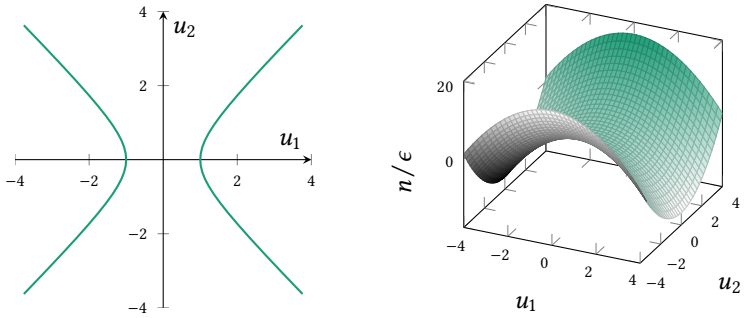
Ensuring ergodicity of chains generated by the constrained HMC transition operator is in general more challenging than for HMC on Euclidean spaces due to the often complex geometry of the constraint manifold C and potential for numerical issues when solving the non-linear equations in the projection steps. In [49] it is shown that if[6]

- C is a connected, smooth differentiable manifold,
- $\mathbf{J}_{\mathbf{c}}$ has full row-rank everywhere,
- and $\pi(\boldsymbol{u}) \propto \exp(\phi(\boldsymbol{u}))$ is smooth and strictly positive on C

for a constrained HMC transition using an approximate flow map defined by a symplectic integrator with step size δt, if the step-size δt is set sufficiently small such that there is a unique solution to the choice of Lagrange multipliers $\boldsymbol{\lambda}$ in each geodesic step (4.43) and the iterative method employed converges to this solution in every step, that the overall transition operator will be irreducible, aperiodic and leave the target distribution on C invariant.

These conditions put stricter requirements on the generator $\mathbf{g}_{\mathbf{x}}$ of a differentiable generative model than those specified in Definition 4.2 and Proposition 4.2 if we wish to use a constrained HMC method to

6 We give only a loose statement of the full conditions here for brevity; for complete details see Theorems 1 to 4 in [49].



(a) $\mathbf{g}_\mathbf{x}^{-1}(1)$ (b) $\mathbf{g}_\mathbf{y}^{-1}(1)$

Figure 4.4.: Visualisations of the hyperbola fibre $\mathbf{g}_\mathbf{x}^{-1}(1)$ of the toy generator $\mathbf{g}_\mathbf{x}$ defined in (4.51) consisting of two disconnected components and the corresponding connected hyperbolic paraboloid fibre $\mathbf{g}_\mathbf{y}^{-1}(1)$ of the noisy generator.

estimate conditional expectations under the model. The requirement that $C = \mathbf{g}_\mathbf{x}^{-1}(\boldsymbol{x})$ is a smooth and connected manifold is likely to be challenging to check for complex generators. If the fibre of $\boldsymbol{x}$ under the generator $\mathbf{g}_\mathbf{x}$ consists of multiple disconnected components then the constrained Hamiltonian dynamic will remain confined to just one of them. Although problematic, this issue is similar to that faced by other MCMC methods in target distributions with multiple separated modes. The requirement that the Jacobian $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ is defined and full row-rank everywhere is also stricter than previously required.

If we define an augmented 'noisy' generator

$$\mathbf{g}_\mathbf{y}(\boldsymbol{u}, \boldsymbol{n}) = \mathbf{g}_\mathbf{x}(\boldsymbol{u}) + \epsilon \boldsymbol{n} \tag{4.50}$$

with $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and ϵ a small positive constant, then if $\mathbf{g}_\mathbf{x}$ is differentiable everywhere then the Jacobian of the augmented generator $\mathbf{J}_{\mathbf{g}_\mathbf{y}}$ will be full row-rank everywhere. That $\mathbf{J}_{\mathbf{g}_\mathbf{y}}$ will have full-row rank everywhere is evident from its block structure $\mathbf{J}_{\mathbf{g}_\mathbf{y}}(\boldsymbol{u}, \boldsymbol{n}) = \left[\mathbf{J}_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u}) \,\middle|\, \epsilon \mathbf{I}\right]$. The Jacobian being full row-rank implies that $\mathbf{g}_\mathbf{y}$ is a submersion from $\mathbb{R}^{M+N_\mathbf{x}}$ to $\mathbb{R}^{N_\mathbf{x}}$.

If U and V are open subsets of $\mathbb{R}^M$ and $\mathbb{R}^N$ respectively with $N > M$ and $\boldsymbol{\phi} : U \to V$ is a differentiable map, then $\boldsymbol{\phi}$ is a submersion *if the Jacobian $\mathbf{J}_{\boldsymbol{\phi}}$ is full row-rank everywhere in U.*

In some cases the fibres under the noisy generator $\mathbf{g}_\mathbf{y}^{-1}(\boldsymbol{x})$ will be connected when the fibres under the original generator $\mathbf{g}_\mathbf{x}^{-1}(\boldsymbol{x})$ are not. As a simple example consider

$$\mathbf{g}_\mathbf{x}(\boldsymbol{u}) = u_1^2 - u_2^2, \qquad \mathbf{g}_\mathbf{x}(\boldsymbol{u}, n) = u_1^2 - u_2^2 + \epsilon n. \tag{4.51}$$

The fibres $\mathbf{g}_x^{-1}(x)$ are hyperbola in $\mathbb{R}^2$, for $x \neq 0$ consisting of two disconnected components as shown in Figure 4.4a. The fibres of $\mathbf{g}_y^{-1}(x)$ are connected hyperbolic paraboloids in $\mathbb{R}^3$ as shown in Figure 4.4b.

This noisy augmentation of the generator corresponds to using an ABC approach with a Gaussian kernel with tolerance ϵ, and so we could instead perform standard HMC using the potential energy (4.24). If ϵ is small however, the previously discussed tendency towards oscillatory trajectories when simulating an unconstrained Hamiltonian dynamic using the potential energy (4.24) corresponding to a Gaussian kernel (see Figure 4.3), can require use of a very small integrator step size. In some cases (examples of which will be shown in the numerical experiments) applying constrained HMC with the noisy generator $\mathbf{g}_y$ can therefore be more efficient than running standard HMC in the ABC target density, despite the much higher per-step costs, as constrained HMC updates are able to use a much larger integrator step size when using small ϵ.

An alternative approach would be to apply a RMHMC algorithm to the Gaussian kernel ABC target density in the input space (4.23) and use a metric exploiting the geometry of this target density to improve the behaviour of the simulated dynamic. For example the metric

$$\boldsymbol{G}(\boldsymbol{u}) = \frac{1}{\epsilon^2}\mathbf{J}_{\mathbf{g}_x}(\boldsymbol{u})^\mathsf{T}\mathbf{J}_{\mathbf{g}_x}(\boldsymbol{u}) + \mathbf{I} \tag{4.52}$$

is positive definite everywhere and equal to the Hessian of the potential energy (4.24) for $\boldsymbol{u} \in \mathbf{g}_{\boldsymbol{x}}^{-1}(\boldsymbol{x})$. Using this metric, for small ϵ and inputs $\boldsymbol{u}$ generating outputs close to the data $\boldsymbol{x}$ i.e. small values of $\frac{1}{\epsilon}\|\mathbf{g}_{\boldsymbol{x}}(\boldsymbol{u}) - \boldsymbol{x}\|_2$, the velocity in the RMHMC dynamic $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{G}(\boldsymbol{u})^{-1}\boldsymbol{p}$ will tend to be higher along the directions tangential to the fibre $\mathbf{g}_{\boldsymbol{x}}^{-1}(\boldsymbol{x})$, reducing the tendency for the dynamic to oscillate normal to the fibre. RMHMC requires use of a computationally costly implicit integrator due to the non-separable Hamiltonian and so like the constrained HMC method proposed here has a significantly higher computational cost per sample than the standard HMC algorithm. However as with constrained HMC the potential for improved exploration of the space for small ϵ may compensate for the more costly updates. We do not explore this idea further here but it may be an interesting avenue for future work.

Geodesic Monte Carlo [52] also considers applying a HMC scheme to sample from non-linear manifolds embedded in a Euclidean space. Sim-

ilarly to [49] however the motivation is performing inference with respect to distributions explicitly defined on a manifold such as directional statistics. The method presented in [52] uses an exact solution for the geodesic flow on the manifold. The use of a geodesic integration scheme within a constrained HMC update as discussed here can be considered an extension for cases when an exact geodesic solution is not available. Instead the geodesic flow is approximately simulated while still maintaining the required volume-preservation and reversibility properties for validity of the overall HMC scheme.

An alternative Metropolis method for sampling from distributions defined on manifolds embedded in a Euclidean space is proposed in [265]. Compared to constrained HMC this alleviates the requirements to calculate the gradient of (the logarithm of) the target density on the manifold, though still requiring evaluation of the constraint function Jacobian. As discussed in Appendix B, using reverse-mode AD the gradient of the target density can be computed at a constant factor overhead of evaluating the target density itself. In general we would expect exploiting the gradient of the target density on the manifold within a simulated Hamiltonian dynamic to lead to more coherent exploration of the target distribution, instead of the more random-walk behaviour of a non-gradient based Metropolis update, and so for the gradient evaluation overhead to be worthwhile.

There is extensive theoretical discussion of the issues involved in sampling from distributions defined on manifolds in [75], including a derivation of conditional densities on a manifold using the co-area formula which directly motivated our earlier derivations of expressions for conditional expectations under a differentiable generative model. The experiments in [75] are mainly concentrated on expository examples using simple parameterised manifolds such as a torus embedded in $\mathbb{R}^3$ and conditional testing in exponential family distributions.

4.9 IMPLEMENTATION DETAILS

The constrained HMC implementation we propose for performing inference in differentiable generative models is shown in Algorithm 11. This algorithm differs in some details from that proposed in [49] and we discuss these differences and some of the computational issues specific to our setting in the following subsections.

Algorithm 11 Constrained Hamiltonian Monte Carlo.

Input:

$\mathbf{g_x}$: observed variable generator function;
ϕ : potential energy function $\phi(\boldsymbol{u}) = -\log \rho(\boldsymbol{u}) + \frac{1}{2}\log|\mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u})\mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u})|$;
$\boldsymbol{x}$: observed data values being conditioned on;
$\boldsymbol{u}$: current chain state (model inputs) with $\|\mathbf{g_x}(\boldsymbol{u}) - \boldsymbol{x}\|_\infty < \epsilon$;
$(\varphi, \boldsymbol{J}, \boldsymbol{L})$: cached values of ϕ, $\mathbf{J}_{\mathbf{g_x}}$ and $\text{chol}\left(\mathbf{J}_{\mathbf{g_x}}\mathbf{J}_{\mathbf{g_x}}^\mathsf{T}\right)$ evaluated at $\boldsymbol{u}$;
ϵ : convergence tolerance for Newton iteration;
I : number of Newton iterations to try before rejecting for non-convergence;
δt : integrator time step; N_s : number of time steps to simulate;
N_g : number of geodesic steps per time step.

Output:

$\boldsymbol{u}_\text{n}$: new chain state with $\|\mathbf{g_x}(\boldsymbol{u}_\text{n}) - \boldsymbol{x}\|_\infty < \epsilon$;
$(\varphi_\text{n}, \boldsymbol{J}_\text{n}, \boldsymbol{L}_\text{n})$: values of ϕ, $\mathbf{J}_{\mathbf{g_x}}$ and $\text{chol}\left(\mathbf{J}_{\mathbf{g_x}}\mathbf{J}_{\mathbf{g_x}}^\mathsf{T}\right)$ evaluated at new $\boldsymbol{u}_\text{n}$.

$\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\boldsymbol{n}, \boldsymbol{J}, \boldsymbol{L})$
$\boldsymbol{u}_\text{p}, \boldsymbol{p}_\text{p}, \boldsymbol{J}_\text{p}, \boldsymbol{L}_\text{p} \leftarrow \textsc{SimDyn}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
$\varphi_\text{p} \leftarrow \phi(\boldsymbol{u})$
$r \sim \mathcal{U}(0, 1)$
$p_a \leftarrow \exp\left(\varphi + \frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{p} - \varphi_\text{p} - \frac{1}{2}\boldsymbol{p}_\text{p}^\mathsf{T}\boldsymbol{p}_\text{p}\right)$
if $r < p_a$ **then**
  $\boldsymbol{u}_\text{n}, \varphi_\text{n}, \boldsymbol{J}_\text{n}, \boldsymbol{L}_\text{n} \leftarrow \boldsymbol{u}_\text{p}, \varphi_\text{p}, \boldsymbol{J}_\text{p}, \boldsymbol{L}_\text{p}$
else
  $\boldsymbol{u}_\text{n}, \varphi_\text{n}, \boldsymbol{J}_\text{n}, \boldsymbol{L}_\text{n} \leftarrow \boldsymbol{u}, \varphi, \boldsymbol{J}, \boldsymbol{L}$

function $\textsc{SimDyn}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \frac{\delta t}{2}\nabla\phi(\boldsymbol{u})^\mathsf{T}$
  $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
  $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L} \leftarrow \textsc{SimGeo}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  for $s \in \{2 \dots N_s\}$ **do**
    $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \delta t\nabla\phi(\boldsymbol{u})^\mathsf{T}$
    $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L} \leftarrow \textsc{SimGeo}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \frac{\delta t}{2}\nabla\phi(\boldsymbol{u})^\mathsf{T}$
  $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
  return $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L}$

function $\textsc{ProjectMom}(\boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  return $\boldsymbol{p} - \boldsymbol{J}^\mathsf{T}\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{J}\boldsymbol{p}$

function $\textsc{ProjectPos}(\boldsymbol{u}, \boldsymbol{J}, \boldsymbol{L})$
  $\boldsymbol{\delta} \leftarrow \mathbf{g_x}(\boldsymbol{u}) - \boldsymbol{x}$
  $i \leftarrow 0$
  while $\|\boldsymbol{\delta}\|_\infty > \epsilon \wedge i < I$ **do**
    $\boldsymbol{u} \leftarrow \boldsymbol{u} - \boldsymbol{J}^\mathsf{T}\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{\delta}$
    $\boldsymbol{\delta} \leftarrow \mathbf{g_x}(\boldsymbol{u}) - \boldsymbol{x}$
    $i \leftarrow i + 1$
  if $i = I$ **then**
    raise RejectMove
  return $\boldsymbol{u}$

function $\textsc{SimGeo}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  for $i \in \left\{1 \dots N_g\right\}$ **do**
    $\tilde{\boldsymbol{u}} \leftarrow \boldsymbol{u} + \frac{\delta t}{N_g}\boldsymbol{p}$
    $\boldsymbol{u}' \leftarrow \textsc{ProjectPos}(\tilde{\boldsymbol{u}}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{J} \leftarrow \mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u}')$
    $\boldsymbol{L} \leftarrow \text{chol}\left(\boldsymbol{J}\boldsymbol{J}^\mathsf{T}\right)$
    $\tilde{\boldsymbol{p}} \leftarrow \frac{N_g}{\delta t}(\boldsymbol{u}' - \boldsymbol{u})$
    $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{u}_r \leftarrow \boldsymbol{u}' - \frac{\delta t}{N_g}\boldsymbol{p}$
    $\boldsymbol{u}_r \leftarrow \textsc{ProjectPos}(\boldsymbol{u}_r, \boldsymbol{J}, \boldsymbol{L})$
    if $\|\boldsymbol{u} - \boldsymbol{u}_r\|_\infty > \sqrt{\epsilon}$ **then**
      raise RejectMove
    $\boldsymbol{u} \leftarrow \boldsymbol{u}'$
  return $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L}$

4.9.1 Iterative solver for projection on to manifold

Rather than the RATTLE integrator used in [49], we use the geodesic integrator generalisation discussed in the previous section to simulate the dynamic. This gives increased flexibility in balancing the need for an appropriately small step-size to ensure convergence of the iterative solution of the equations projecting on to the constraint manifold and

using a more efficient larger step size for updates to the momentum due to the potential energy gradient. We have assumed $\boldsymbol{M} = \mathbf{I}$ here; other mass matrix choices can be implemented by reparameterising the model with an initial linear transformation stage in the generator.

The projection on to the constraint manifold in the geodesic steps corresponding to (4.43) is performed in the function ProjectPos in Algorithm 11. We use a quasi-Newton method for solving for $\boldsymbol{\lambda}$ the system of equations $\mathbf{g}_{\mathbf{x}}(\boldsymbol{u} + (\delta t/N_g)\boldsymbol{p} - \boldsymbol{J}^{\mathsf{T}}\boldsymbol{\lambda}) = \boldsymbol{x}$ where $\boldsymbol{J} = \mathbf{J}_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})$. The true Newton update would be

$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^{\mathsf{T}}\left(\mathbf{J}_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u}')\boldsymbol{J}^{\mathsf{T}}\right)^{-1}(\mathbf{g}_{\mathbf{x}}(\boldsymbol{u}') - \boldsymbol{x}). \tag{4.53}$$

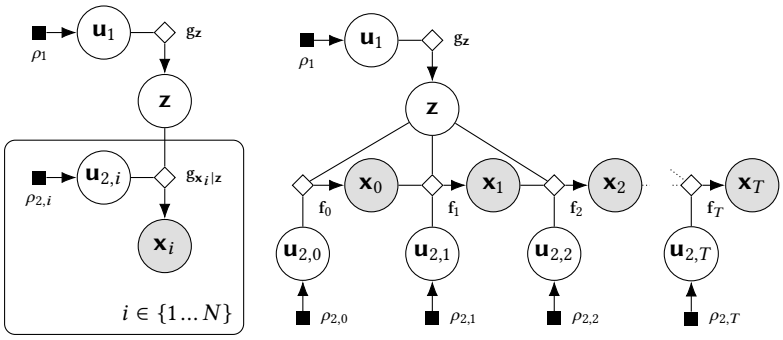
This requires recalculating the Jacobian and solving a dense linear system within the optimisation loop. Instead as proposed in [14] we use a symmetric quasi-Newton update,

$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^{\mathsf{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}\right)^{-1}(\mathbf{g}_{\mathbf{x}}(\boldsymbol{u}') - \boldsymbol{x}). \tag{4.54}$$

The Jacobian product $\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}$ evaluated at the previous state is used to condition the moves. This matrix is positive-definite and a Cholesky decomposition can be calculated outside the optimisation loop allowing cheaper quadratic cost solves within the loop.

Convergence of the quasi-Newton iteration is signalled when the maximum absolute difference between the generated observed variables and the observed data is below a tolerance ϵ, i.e. $\|\mathbf{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x}\|_\infty < \epsilon$. The tolerance is somewhat analogous to the ϵ parameter in ABC methods, however here we can set this value close to machine precision (with $\epsilon = 10^{-8}$ in the experiments) and so the error introduced is comparable to that otherwise incurred for using non-exact arithmetic.

In some cases the quasi-Newton iteration will fail to converge. We use a fixed upper limit on the number of iterations and reject the move (line 34 in Algorithm 11) if convergence is not achieved within this limit. To ensure reversibility, once we have solved for a forward geodesic step on the manifold in SimGeo, we then check if the corresponding reverse step (with the momentum negated) returns to the original position. This involves running a second Newton iteration, though as it reuses the same Jacobian $\boldsymbol{J}$ and Cholesky factor $\boldsymbol{L}$, the evaluation of which tend to be the dominant costs in the algorithm, we found the



(a) Independent $\mathbf{x}_i$ (b) Markovian $\mathbf{x}_i$

Figure 4.5.: Factor graphs of examples of structured directed generative models.

overhead introduced tended to be quite small (around a 20% increase in run-time compared to only performing the forward step). A similar scheme for ensuring reversibility is proposed in [265].

The square root of the tolerance ϵ used for the initial Newton convergence check *in the output space of generator* (line 29 in Algorithm 11) is used for the reverse-step check on *the inputs* (line 48 in Algorithm 11) based on standard recommendations for checking convergence in optimisation routines [62]. In the implementation we used in the experiments, we fall back to a MINPACK [179] implementation of the robust Powell's Hybrid method [212] if the quasi-Newton iteration diverges or fails to converge, with a rejection then only occurring if both iterative solvers fail. In practice we found if the step size δt and number of geodesic steps N_g is chosen appropriately then rejections due to non-convergence or non-reversible steps occur very rarely.

4.9.2 Exploiting model structure

For larger systems, the Cholesky decomposition of the Jacobian matrix product $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}$ (line 42) will become a dominant cost, generally scaling cubically with $N_\mathbf{x}$. In many models however conditional independency structure will mean that not all observed variables $\mathbf{x}$ are dependent on all of the input variables $\mathbf{u}$ and so the Jacobian $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ has a sparse structure which can be exploited to reduce this worst-case cost.

In particular two common cases are directed generative models in which the observed variables $\mathbf{x}$ can be split into groups $\{\mathbf{x}_i\}_{i=1}^G$ such that all of the $\mathbf{x}_i$ are either conditionally independent given the latent variables $\mathbf{z} = \mathbf{g}_\mathbf{z}(\mathbf{u}_1)$ (for example a model for a IID dataset), or each $\mathbf{x}_i$ is conditionally independent of all $\{\mathbf{x}_j\}_{j<i-1}$ given $\mathbf{x}_{i-1}$ and $\mathbf{z}$ (most commonly

Markov chains for example from simulation of a SDE model as shown in Figure 1.5 though models with more general tree structured dependencies can also be ordered into this form).

Figure 4.5 shows factor graphs for directed generative models with these two structures, with the conditional independencies corresponding to each $\mathbf{x}_i$ being generated as a function of only a subset $\mathbf{u}_{2,i}$ of the random input variables $\mathbf{u}_2$. Equivalently these structures can be considered as corresponding to generators which can be expressed in one of the two forms below

$$\mathbf{x}_i = \mathbf{g}_{\mathbf{x}_i|\mathbf{z}}(\mathbf{z}, \mathbf{u}_{2,i}) \qquad \text{(independent)} \tag{4.55}$$

$$\mathbf{x}_i = \mathbf{f}_i(\mathbf{z}, \mathbf{x}_{i-1}, \mathbf{u}_{2,i}) = \mathbf{g}_{\mathbf{x}_i|\mathbf{z}}\left(\mathbf{z}, \{\mathbf{u}_{2,j}\}_{j=1}^{i}\right) \qquad \text{(Markovian).} \tag{4.56}$$

For models with these structures the generator Jacobian

$$\mathbf{J}_{\mathbf{g}_\mathbf{x}} = \left[\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1} \middle| \frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}\right] \tag{4.57}$$

has a component $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$ which is either block-diagonal (independent) or block-triangular (Markovian). Considering first the simplest case where each $(\mathbf{x}_i, \mathbf{u}_{2,i})$ pair are single dimensional, the Cholesky decomposition of $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T} = \frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}{}^\mathsf{T} + \frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}{}^\mathsf{T}$ can then be computed by low-rank Cholesky updates of the triangular or diagonal matrix $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$ with each of the columns of $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}$. As $\dim(\boldsymbol{u}_1) = L$ is often significantly less than the number of observations being conditioned on $N_\mathbf{x}$, the resulting $O(LN_\mathbf{x}^2)$ cost of the low-rank Cholesky updates is a significant improvement over the original $O(N_\mathbf{x}^3)$. For cases in which each $(\mathbf{x}_i, \mathbf{u}_{2,i})$ pair are both vectors of dimension D (i.e. $N_\mathbf{x} = GD$) and so $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$ is block diagonal or triangular, then the Cholesky factorisation of $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}{}^\mathsf{T}$ can be computed at a cost $O(GD^3)$ for block diagonal, and $O(G^2D^3)$ for block triangular $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$, with then again $O(LN_\mathbf{x}^2)$ cost low-rank updates of this Cholesky factor by the columns of $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}$ performed.

When $\mathbf{x}_i$ and $\mathbf{u}_{2,i}$ are vectors of differing dimensions, with generally in this case $\dim(\mathbf{u}_{2,i}) > \dim(\mathbf{x}_i)$ due to the requirement the total number of random inputs M is at least $N_\mathbf{x}$, then though we could choose a subset of each $\mathbf{u}_{2,i}$ of equal dimension to $\mathbf{x}_i$ so as to identify a block-triangular component, generally any gain here will be minimal and it may be preferable to use efficient blocked algorithhms to compute the Cholesky of $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}$ directly.

4.9.3 Efficiently evaluating the potential energy and gradient

The Metropolis accept step and momentum updates in the SimDyn routine require evaluating the potential energy corresponding to (4.35) and its gradient respectively. Although this can by achieved by directly using the expression given in (4.35) (and applying reverse-mode AD to get the gradient), both the potential energy and its gradient can be more efficiently calculated by reusing the Cholesky decomposition of the constraint Jacobian Gram matrix computed in line 42.

Dropping the dependence of the Jacobian on $\boldsymbol{u}$ for brevity we have that the potential energy ϕ corresponding to the negative logarithm of the unnormalised target density on the manifold (4.35) is

$$\phi(\boldsymbol{u}) = \frac{1}{2}\log\left|\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\right| - \log\rho(\boldsymbol{u}) \tag{4.58}$$

In general evaluating the determinant $|\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}|$ has computational cost which scales as $O(MN_\mathbf{x}^2)$. However the lower-triangular Cholesky decomposition $\boldsymbol{L}$ of $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}$ is already calculated in the SimGeo routine in Algorithm 11. Using basic properties of the matrix determinant

$$\phi(\boldsymbol{u}) = \sum_{i=1}^{N_\mathbf{x}} \log(L_{ii}) - \log\rho(\boldsymbol{u}). \tag{4.59}$$

Given the Cholesky factor $\boldsymbol{L}$ we can therefore can evaluate the potential energy ϕ at a marginal computational cost that scales linearly with $N_\mathbf{x}$. For the gradient we can use reverse-mode AD to calculate the derivative of (4.59) with respect to $\boldsymbol{u}$. This requires propagating partial derivatives through the Cholesky decomposition [182]; implementations for this are present in many automatic differentiation frameworks.

Alternatively using the standard result for derivative of a log determinant and the invariance of the trace to cyclic permutations we have that the gradient of the log determinant term in (4.58) can be manipulated in to the form

$$\frac{1}{2}\frac{\partial}{\partial u_i}\log\left|\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\right| = \mathrm{Trace}\left(\left(\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\right)^{-1}\frac{\partial\mathbf{J}_{\mathbf{g}_\mathbf{x}}}{\partial u_i}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\right) \tag{4.60}$$

$$= \mathrm{Trace}\left(\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\left(\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\right)^{-1}\frac{\partial\mathbf{J}_{\mathbf{g}_\mathbf{x}}}{\partial u_i}\right) \tag{4.61}$$

We denote the matrix vectorisation operator vec such that for a $M \times N$ matrix $\boldsymbol{A}$, we have $\mathrm{vec}(\boldsymbol{A}) = [A_{1,1}, \ldots, A_{M,1}, A_{1,2}, \ldots, A_{N,M}]^\mathsf{T}$. Then as

the trace of a matrix product defines an inner product we have that $\text{Trace}(\boldsymbol{A}\boldsymbol{B}) = \text{vec}(\boldsymbol{A})^\mathsf{T}\text{vec}(\boldsymbol{B})$. We can therefore write the gradient of the log determinant term as

$$\frac{1}{2}\frac{\partial}{\partial \boldsymbol{u}}\log\left|\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\right| = \text{vec}\left(\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\left(\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}\right)^{-1}\right)^\mathsf{T}\frac{\partial \text{vec}\left(\mathbf{J}_{\mathbf{g}_\mathbf{x}}\right)}{\partial \boldsymbol{u}} \tag{4.62}$$

The matrix inside the left vec operator can be computed once by reusing the Cholesky factorisation of $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}$ to solve the system of equations by forward and backward substitution. We then have an expression in the form of a vector-Jacobian product which is provided as an efficient primitive in many AD frameworks, e.g. as `Lop` in Theano, and like the gradient (which is actually a special case) can be evaluated at cost which is a constant over head of evaluating the forward function (i.e. the cost of evaluating $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ here).

4.9.4 Initialising the state

A final implementation detail is the requirement to find an initial $\boldsymbol{u}$ satisfying $\mathbf{g}_\mathbf{x}(\boldsymbol{u}) = \boldsymbol{x}$ to initialise the chain at. In directed generative models with one of the structures described in Section 4.9.2, a method we found worked well in the experiments was to sample a $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ pair from $\mathrm{P}_\mathbf{u}$ and then keeping the $\boldsymbol{u}_1$ values fixed, solve $\mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{g}_\mathbf{z}(\boldsymbol{u}_1), \boldsymbol{u}_2) = \boldsymbol{x}$ for $\boldsymbol{u}_2$ using for example Newton's method or by directly minimising the Euclidean norm $\|\mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{g}_\mathbf{z}(\boldsymbol{u}_1), \boldsymbol{u}_2) - \boldsymbol{x}\|_2^2$ with respect to $\boldsymbol{u}_2$ by gradient descent. In more general cases one strategy is to randomly sample affine subspaces by generating a $M \times N_\mathbf{x}$ matrix $\boldsymbol{P}$ and M dimensional vector $\boldsymbol{b}$ and then attempting to find any intersections with the manifold by iteratively solving $\mathbf{g}_{\boldsymbol{x}}(\boldsymbol{P}\boldsymbol{v} + \boldsymbol{b})$ for $\boldsymbol{v}$, sampling a new subspace if no roots are found.

4.10 NUMERICAL EXPERIMENTS

To evaluate the performance of the MCMC methods proposed in Sections 4.6 and 4.8 we performed inference experiments with three implicit generative models: a quantile distribution model for an IID dataset, a Lotka–Volterra predator-prey SDE simulator model, and differentiable generator network models for human poses. In all experiments Theano [248], a Python computation graph framework providing reverse-mode AD, was used to specify the generator functions and compute their Jac-

obians. Although Theano has the ability to utilise *graphics processing units* (GPUs) for computation, to avoid complicating the comparison of algorithm run times all experiments were run on a CPU (Intel Core i5-2400 quad-core).

4.10.1 Quantile distribution inference

As a first example we consider inferring the parameters of quantile distribution model for a IID dataset of univariate values. The generalised Tukey lambda distribution [89, 218] is a four parameter family of distributions defined via its quantile function. It has very flexible form which can describe distributions with a range of shapes, including close approximations of standard distributions such as the normal but also allowing asymmetric distributions with more general skewness and kurtosis. This flexibility has supported it use for statistical modelling in a diverse range of settings, including for example finance [64], climatology [199], control engineering [201] and material science [39].

Using the inverse CDF method discussed in Chapter 2 it is simple to generate samples given a quantile function by mapping standard uniform samples through the quantile function. The quantile function does not have an analytic inverse however so the CDF and corresponding density function do not have explicit forms. The use of ABC to perform Bayesian inference using quantile distributions was suggested by Allingham, King and Mengersen in [4] with a pseudo-marginal ABC MCMC approach used based on order statistics of the observations in the experiments. McVinish [169] proposed a more efficient 'modified' ABC MCMC scheme specifically tailored to quantile distributions, with interval bisection used to identify an efficient proposal distribution for updates to the auxiliary uniform variables mapped through the quantile function.

We follow [169] in parameterising the quantile function of the generalised lambda distribution as

$$q_{\text{GL}}(p \mid \boldsymbol{z}) = z_1 + \frac{1}{z_2}\left(\frac{p^{z_3} - 1}{z_3} + \frac{(1-p)^{z_4} - 1}{z_4}\right), \tag{4.63}$$

with z_1 a location parameter, z_2 a positive scale parameter and z_3 and z_4 shape parameters. In the experiments in [169], a synthetic dataset $\boldsymbol{x}$ of $N = 250$ independent samples are generated from a generalised lambda distribution using the quantile function (4.63) with parameters

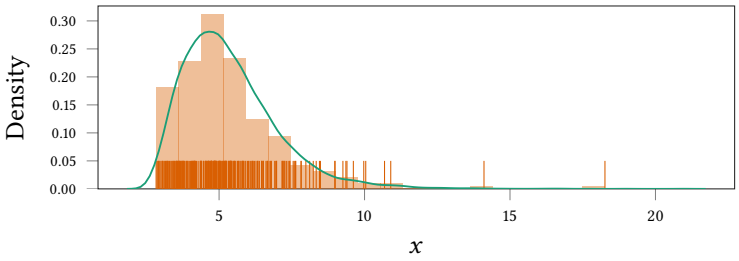


Figure 4.6.: Histogram of generated generalised lambda distribution dataset used in experiments with $N = 250$ points generated using the quantile function parameterisation in (4.63) with parameters $z_1 = 5$, $z_2 = 1$, $z_3 = 0.4$ and $z_4 = -0.1$. The light orange region shows the histogram of the generated data with the orange ticks along the x axis indicating the actual data points. The green curve shows a kernel density estimate of the density of the distribution using a separate set of 10 000 independent samples.

$z_1 = 5$, $z_2 = 1$, $z_3 = 0.4$ and $z_4 = -0.1$. The task considered in [169] is then inferring the posterior distribution on the parameters $\mathbf{z}$ given observed (synthetic) data $\boldsymbol{x}$. A prior density on $\mathbf{z}$ is defined as

$$\mathrm{p}_{\mathbf{z}}(\boldsymbol{z}) = \mathrm{Exp}(z_2 \mid \lambda) \prod_{i \in \{1,3,4\}} \mathcal{N}\left(z_i \mid 0, \sigma^2\right) \tag{4.64}$$

corresponding to independent normal priors on each of the location and shape parameters and an exponential prior on the scale parameter. In the experiments in [169] the prior hyper parameters are chosen as $\sigma = 10$ and $\lambda = 1/10$. In [169] the proposed modified ABC MCMC method is compared to a standard pseudo-marginal ABC MCMC approach and a population Monte Carlo ABC method [20]. The proposed modified ABC MCMC algorithm was found to significantly outperform the other two approaches, and so we focus on comparing to this method.

We compare a Cython [22] implementation of the modified ABC MCMC algorithm to two of the algorithms proposed in previous sections: an ABC approach with a Gaussian kernel k_ϵ, running HMC in the input space to a differentiable generator for the model as discussed in Section 4.6; the constrained HMC algorithm proposed in Section 4.8, conditioning the output of a differentiable generator to be exactly equal to observed data. As in [169] we use $N = 250$ generated data points using the parameters $\boldsymbol{z} = [5,\ 1,\ 0.4,\ -0.1]^\mathsf{T}$ with the generated data used in our experiments shown in Figure 4.6.

We formulate the quantile distribution model as a directed differentiable generative model as follows. We define a 'prior' generator $\mathbf{g_z}$ for the parameters $\mathbf{z}$ by

---

function $\mathbf{g_z}(\mathbf{u}_1)$

$z_1 \leftarrow \sigma u_{1,1}$

$z_3 \leftarrow \sigma u_{1,2}$

$z_4 \leftarrow \sigma u_{1,3}$

$z_2 \leftarrow \frac{1}{\lambda} \log\left(1 + \exp\left(\frac{\pi}{\sqrt{3}} u_{1,4}\right)\right)$

return $[z_1,\ z_2,\ z_3,\ z_4]^\mathsf{T}$

Here the input variables $u_{1,1}$, $u_{1,2}$ and $u_{1,3}$ are assumed to have independent standard normal distributions $\mathcal{N}(0, 1)$. The input variable $u_{1,4}$, which maps to the scale parameter z_2 with an exponential prior, has a unit-variance logistic distribution Logistic$(0,\ \sqrt{3}/\pi)$[7]. Given a vector of inputs $\mathbf{u}_1$ with these distributions, $\mathbf{g_z}$ outputs a parameter vector $\mathbf{z}$ distributed according to the prior density (4.64).

The generator for the observed variables $\mathbf{x}$ given the parameters $\mathbf{z}$ and additional random inputs $\mathbf{u}_2$ is then specified by

---

function $\mathbf{g_{x|z}}(\mathbf{z},\ \mathbf{u}_2)$

for $n \in \{1 \dots N\}$ **do**

$p_n \leftarrow \left(1 + \exp\left(-\frac{\pi}{\sqrt{3}} u_{2,n}\right)\right)^{-1}$

$x_n \leftarrow q_{\mathrm{GL}}(p_n \mid \mathbf{z})$

return $[x_1,\ x_2,\ \dots\ ,\ x_N]^\mathsf{T}$

Here the input variables $\mathbf{u}_2$ have independent unit-variance logistic distributions Logistic$(0,\ \sqrt{3}/\pi)$. These are transformed to standard uniform variables via a logistic sigmoid function, with these uniform variables then mapped through the quantile function to generate values from the generalised lambda quantile distribution given the provided parameter values $\mathbf{z}$.

As the generated observed variables $\mathbf{x}$ are conditionally independent given the parameter variables $\mathbf{z}$, the Jacobian of the overall generator $\mathbf{g_x}(\mathbf{u}) = \mathbf{g_{x|z}}(\mathbf{g_z}(\mathbf{u}_1), \mathbf{u}_2)$ has the block structure discussed in Section 4.9.2, with a dense matrix block corresponding to the partial derivatives of the generated $\mathbf{x}$ with respect to the inputs $\mathbf{u}_1$ mapping to parameters, and a diagonal matrix block corresponding to the partial derivatives of the generated $\mathbf{x}$ with respect to the inputs $\mathbf{u}_2$. As described in Section

7 The reparameterisation of the exponential distribution used here is described in Table D.1 in Appendix D

4.9.2 this allows efficient computation of the Jacobian product Cholesky factor in the constrained HMC algorithm.

The modified ABC MCMC method uses a proposal kernel to generate updates to the parameters $\mathbf{z}$ which are then accepted or rejected in a Metropolis–Hastings step. We follow the experiments of [169] and use a uniform random-walk proposal density $\mathcal{U}(z' \mid z - s, z + s)$ where s is a step-size parameter, which was tuned to give an average accept rate of ~ 0.25 in pilot runs, with $s = 0.075$ used in our experiments. The interval bisection method used to construct the proposed updates to the auxiliary uniform variables has a free parameter m defining the number of bisection iterations; following the method used in the experiments of [169] we use $m = 16$. The ABC kernel used in the modified ABC MCMC algorithm is uniform across a cubic region specified by an infinity norm tolerance

$$k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{\epsilon^D} \mathbb{1}_{[0,\epsilon]}(\|\boldsymbol{y} - \boldsymbol{x}\|_\infty) = \frac{1}{\epsilon^D} \prod_{d=1}^{D} \mathbb{1}_{[0,\epsilon]}(|y_i - x_i|) \tag{4.65}$$

with the product decomposition of this kernel being central to the proposed efficient update to the auxiliary variables in [169]. We follow [169] in using a tolerance of $\epsilon = 0.1$ in the experiments.

In pilot runs with the modified ABC MCMC algorithm, we found that when initialising chains from the normal–exponential prior (4.64) with hyperparameters $\sigma = 10$ and $\lambda = 1/10$, that some chains failed to converge, remaining at the initial state for long series of rejections even with very small step sizes and in some cases failing completely due to numerical overflow. By generating additional synthetic datasets using parameters sampled from a prior with $\sigma = 10$ and $\lambda = 1/10$ it was found that this prior choice put significant mass on settings leading to very extreme sampled values and in some cases producing values beyond the maximum range of double precision floating point. As such extreme variation in the target distribution seems implausible a-priori, we use a more informative choice of prior in our experiments with $\sigma = \lambda = 1$, with this choice giving a more plausible range of variation for simulated datasets. We found the regularisation provided by this choice to significantly improve the stability of all the methods tested while having a negligible impact on the inferred posteriors.

For all the approaches tested, the chains for the parameter values $\mathbf{z}$, or correspondingly the input variables $\mathbf{u}_1$ in the case of the methods parameterised in the generator input space, were initialised from values sampled from the prior, with the same 5 independently sampled initial states used for all chains. For the constrained HMC chains, the initial states of the remaining $\mathbf{u}_2$ input variables were set by using an optimisation routine to solve for values of these variables giving generated observed outputs within an maximum elementwise distance of 10^{-8} of the observed data values. These same optimised $\mathbf{u}_2$ initial states were also used for the unconstrained HMC chains. This optimisation was a negligible overhead (less than one second) and so not included in the run time estimates.

For the constrained HMC chains we used an integrator step size $\delta t = 0.6$ and $N_g = 4$ inner geodesic steps per overall time step. These values were chosen based on pilot runs to give an average accept rate in the range 0.6 to 0.9 [35] and to minimise the occurrence of any rejections due to non-reversible geodesic steps or convergence failure in the iterative solver. The number of integrator steps N_s for each constrained HMC update was uniformly sampled from $[5, 10]$.

For the unconstrained HMC chains using a Gaussian kernel ABC target density in the generator input space (4.23), we ran sets of chains for $\epsilon = 0.25$ and $\epsilon = 0.05$ (due to the different kernel from that used in the modified ABC MCMC method the tolerance values cannot be directly compared between the two methods). For $\epsilon = 0.25$ we used a integrator step size $\delta t = 2.5 \times 10^{-3}$ and for $\epsilon = 0.05$, $\delta t = 5 \times 10^{-4}$, again chosen based on trying to achieve a target accept rate in $[0.6, 0.9]$. We found however that the sensitivity of the stability of the updates to δt made it challenging to meet this requirement, with values for δt giving reasonable accept rates below 0.9 for some chains leading to other having very low accept rates, and so the chosen δt values gave accept rates closer to 0.95 in most cases. We sampled the number of leapfrog steps L for each update uniformly from $[20, 40]$ for the $\epsilon = 0.25$ chains and $[40, 80]$ for the $\epsilon = 0.05$ chains; these values were chosen relatively arbitrarily and performance could likely be improved by tuning these values or using the adaptive NUTS algorithm [130].

For all chains we ran initial warm-up phases which were excluded from the estimates to allow for convergence to the posterior typical set and reduce the estimator bias. The number of warm-up iterations for each

chain was hand-tuned based on visualising traces of the chains and setting the number of warm-up iterations to remove any obvious initial transient behaviour in the chains. For the constrained and unconstrained HMC chains we found it helped stability to use a smaller integrator step size and fewer integrator steps in the warm-up phase. The initial states have atypically high potential energy and so the momenta quickly grow large in the simulated dynamics in the early chain iterations, in some cases leading to stability issues with the step size. Using a smaller initial step size and smaller number of integration steps and so more frequent momentum resampling operations where the momenta are restored to values with more reasonable magnitudes helps to alleviate this issue.

We used $\delta t = 0.05$ and $N_s = 2$ in 200 warm up iterations for each constrained HMC chain; $\delta t = 10^{-3}$, $L = 10$ for 1000 warm up iterations for each $\epsilon = 0.25$ HMC chain; and $\delta t = 2.5 \times 10^{-4}$ and $L = 20$ for 5000 warm up iterations for each $\epsilon = 0.05$ HMC chain. For the modified ABC MCMC chains we used 5000 warm up iterations (using the same $s = 0.075$ step size as in the main runs). We ran the main sampling phase for 1000 iterations for the constrained HMC chains, 30 000 iterations for the $\epsilon = 0.25$ HMC chains, 15 000 iterations for the $\epsilon = 0.05$ HMC chains and 100 000 iterations for the modified ABC MCMC chains; in all cases this leading to chains taking roughly five minutes to run each in our implementations (we recorded exact run times for each chain *including the warm-up iterations* to use in normalising efficiency estimates). Although performance of the different methods is somewhat implementation dependent, in all cases the use of efficient compiled updates for the main computational bottlenecks (either via Cython for the modified ABC MCMC implementation or Theano for the two HMC algorithms) meant that the interpreter overhead from using Python was at least minimal, with all chains fully utilising a single CPU core when running.

The estimated parameter posterior distributions using the samples from all of the chains run for each of the approaches tested are shown in Figure 4.7. We can see that the marginal posteriors generally concentrate relatively tightly around the values of the parameters used to generate the data (shown by dashed lines), with the constrained HMC and modified ABC MCMC algorithms showing tighter estimated distributions than the Gaussian kernel HMC chains, with the $\epsilon = 0.25$ case being the most diffuse as expected. The estimated posterior marginals from the

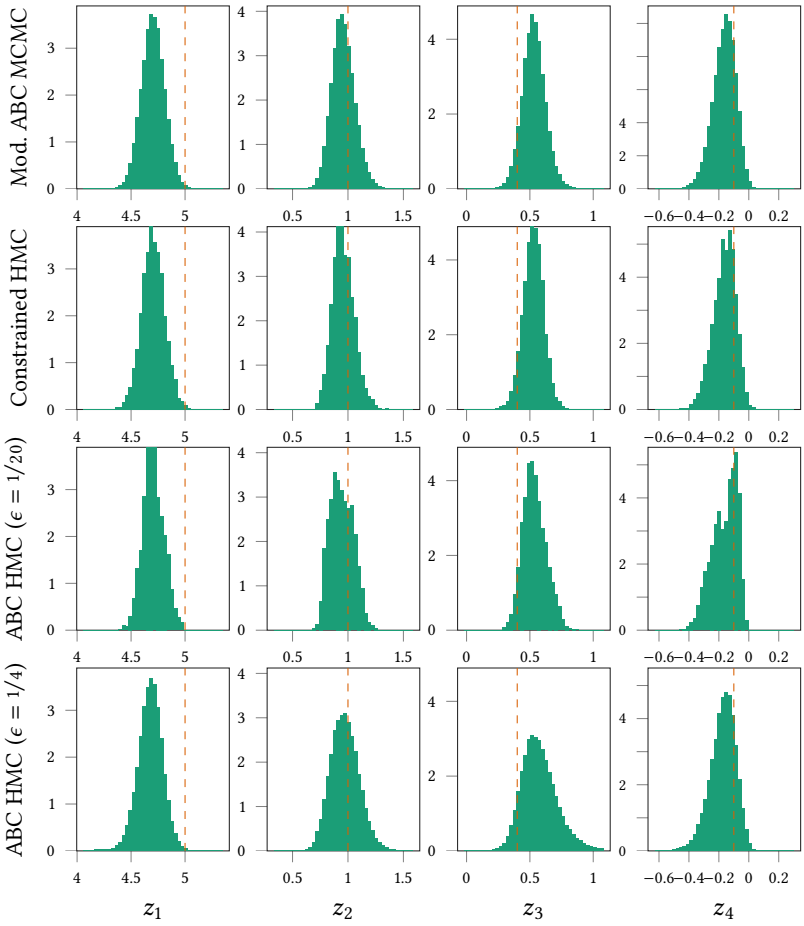


Figure 4.7.: Estimated marginal posterior distributions of generalised lambda model parameters. Each row corresponds to samples from five independent chains for MCMC method labelled to left of plot, while each column corresponds to one of the four distribution parameters, labelled to bottom of plot. The orange dashed line on each axis indicates the value of the parameter used to generate the data.

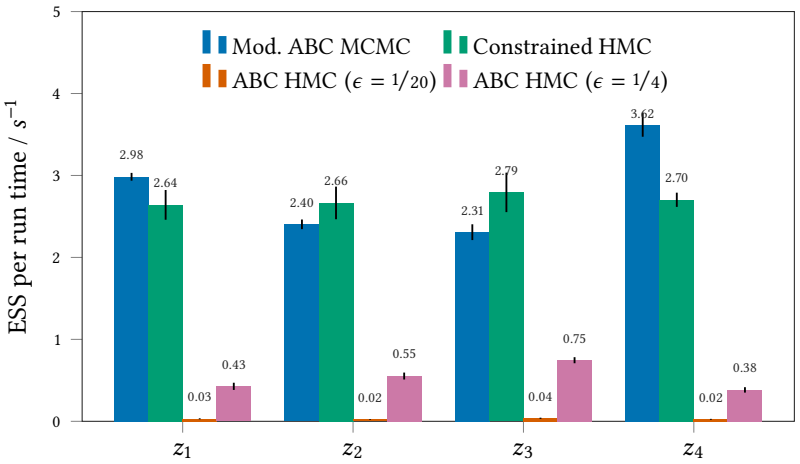


Figure 4.8.: Estimated ESS for posterior means of each generalised lambda model parameter normalised by chain run time. Each coloured set of bars corresponds to mean estimated ESS per run time across five independent chains for the method indicated in the legend. The ticks on the bars show ±1 standard error of mean.

$\epsilon = 0.05$ HMC chains show spurious appearing irregularities not evident in the results from the other chains, which is indicative of convergence issues in the chains. The estimated PSRF statistic for the $\epsilon = 0.05$ HMC chains was $\hat{R} = 1.21$ which also suggests convergence problems; for both the constrained HMC and modified ABC MCMC chains $\hat{R} = 1.00$ while for the $\epsilon = 0.25$ HMC chains $\hat{R} = 1.03$.

Figure 4.8 shows estimates of the ESS for the posterior means of each model parameter normalised by the chain run time in seconds for each of the four approaches tested. The coloured bars show the mean values across the five independent chains for each method and the black ticks plus or minus one standard error of mean. Although as noted above the real-time performance of the methods is somewhat implementation dependent, it seems that the proposed constrained HMC method performs broadly about as well as the modified ABC MCMC approach here in terms of sampling efficiency, while the methods performing HMC in the generator input space using a Gaussian ABC kernel are significantly less efficient.

That the proposed constrained HMC method works about as well as an algorithm custom tuned to this particular problem is encouraging. Further given the generally improved relative performance of HMC methods compared to random-walk Metropolis based methods as the dimension of the target distribution grows, it seems plausible that the

comparison would be even more positive towards the constrained HMC method in models with larger numbers of parameters. It is interesting to note that both approaches use iterative optimisation methods within the inner loop of the algorithms: in the modified ABC MCMC method interval bisection is used to find a relatively tight bounding box on the allowable values of the auxiliary uniform variables used to generate the simulated data given the current parameter values, while in our constrained HMC approach a quasi-Newton iteration is used to project on to the constraint manifold in the input space corresponding to the fibre of observed data under the generator function $\mathbf{g}_{\mathbf{x}}$. In both cases this helps overcome the curse of dimensionality effects typically experienced when conditioning on high-dimensional observed data in ABC inference problems.

The ABC Gaussian kernel HMC approaches also use gradient information to bias proposed updates to the generator inputs towards values producing outputs consistent with the data. However the complex 'narrow ridge' geometry of the target density in the input space (as illustrated for a simple example in Figure 4.3) tends to require small integrator step sizes which limits the overall efficiency of the approach.

4.10.2 Lotka–Volterra parameter inference

As a second test case we considered inferring the parameters of a SDE variant of the Lotka–Volterra predator–prey model, a common example problem in the ABC literature e.g. [171, 202]. In particular given (synthetic) observed predator–prey population data we consider inferring the model parameters of the following SDEs

$$\mathrm{d}r = (z_1 r - z_2 r f)\mathrm{d}t + \mathrm{d}n_r, \quad \mathrm{d}f = (z_4 r f - z_3 f)\mathrm{d}t + \mathrm{d}n_f, \tag{4.66}$$

where r represents the prey population, f the predator population, $\{z_i\}_{i=1}^4$ the system parameters and n_r and n_f are zero-mean white noise processes.

A simulator for these SDEs can be formed by using an Euler–Maruyuma [141] integration scheme to generate simulated realisations of the stochastic process at discrete time points. If the white-noise processes n_r and n_f have variances σ_r^2 and σ_f^2 respectively, then an Euler–Maruyama discretisation of N_s time points of the SDEs 4.66 with an integrator time step δt and initial system state (r_0, f_0) can be generated given a vec-

tor of standard normal random variates $\mathbf{u}_2$ as defined in the following pseudo-code.

---

function $\mathrm{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z},\, \mathbf{u}_2)$

  $\mathrm{r}_0 \leftarrow r_0$

  $\mathrm{f}_0 \leftarrow f_0$

  for $s \in \{1 \ldots N_s\}$ **do**

    $\mathrm{r}_s \leftarrow \mathrm{r}_{s-1} + \delta t(\mathrm{z}_1\mathrm{r}_{s-1} - \mathrm{z}_2\mathrm{r}_{s-1}\mathrm{f}_{s-1}) + \sqrt{\delta t}\sigma_r \mathrm{u}_{2,2s}$

    $\mathrm{f}_s \leftarrow \mathrm{f}_{s-1} + \delta t(\mathrm{z}_4\mathrm{r}_{s-1}\mathrm{f}_{s-1} - \mathrm{z}_3\mathrm{f}_{s-1}) + \sqrt{\delta t}\sigma_f \mathrm{u}_{2,2s+1}$

  $\mathbf{x} \leftarrow [\mathrm{r}_1, \mathrm{f}_1, \ldots \mathrm{r}_{N_s}, \mathrm{f}_{N_s}]$

  return x

As suggested by the notation we can consider this Euler–Maruyama integration as defining the observed generator of a directed generative model, mapping from the unobserved parameter variables $\mathbf{z}$ and an auxiliary vector of standard normal random inputs $\mathbf{u}_2$ to a vector formed by the concatenation of the simulated state sequences. This mapping is differentiable with respect to $\mathbf{z}$ and $\mathbf{u}_2$ and so defines a differentiable generative model. The generator in this case has the Markovian structure discussed in Section 4.9.2 allowing efficient computation of the Cholesky factor of the Jacobian matrix product $\mathbf{J}_{\mathrm{g}_\mathbf{x}}\mathbf{J}_{\mathrm{g}_\mathbf{x}}^\mathsf{T}$. This Markovian structure is present only when all simulated time steps of the SDEs are observed: if for example a smaller integrator time step $\delta t / M$ was used to decrease the approximation error in simulated realisations and simulated observations only recorded every M time points for some positive integer M, then the generator Jacobian would not longer have the structure discussed in Section 4.9.2.

In the parameterisation used in (4.66), all of the parameter variables $\mathbf{z}$ are required to be positive. A simple choice of a prior distribution on the parameters is therefore a log-normal distribution

$$\mathrm{p}_\mathbf{z}(z) = \prod_{i=1}^{4} \mathrm{LogNorm}(z_i \mid m_i, s_i) \tag{4.67}$$

A generator function for the parameters can then be defined by

---

function $\mathrm{g}_\mathbf{z}(\mathbf{u}_1)$

  $\mathbf{z} \leftarrow \exp(\boldsymbol{s} \odot \mathbf{u}_1 + \boldsymbol{m})$

  return z

---

where $\mathbf{u}_1$ is an input vector of standard normal variables.

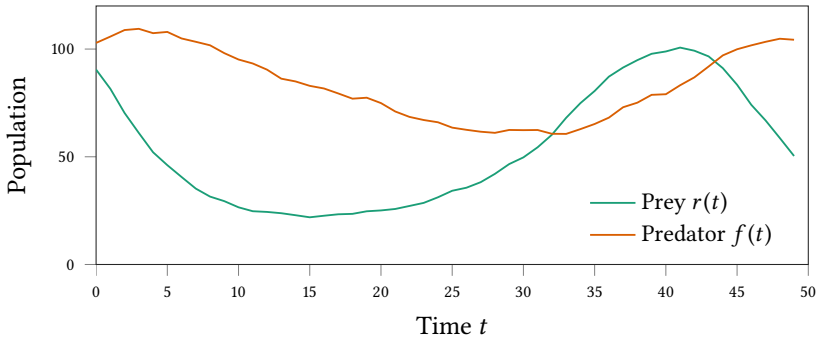


Figure 4.9.: Traces of generated realisations of Lotka–Volterra SDE model (4.66) used as the observations in the experiments.

For the experiments we generated a synthetic observed data set $\boldsymbol{x} = [r_1,\ f_1,\ \ldots\ r_{50},\ f_{50}]^{\mathsf{T}}$ of $N_s = 50$ simulated time points of predator–prey population state sequences using the Euler–Maruyama generator function defined above with an integrator time-step $\delta t = 1$, white noise process standard deviation $\sigma_f = \sigma_r = 1$, initial condition $r_0 = f_0 = 100$ and model parameter values $z_1 = 0.4$, $z_2 = 0.005$, $z_3 = 0.05$, $z_4 = 0.001$ (chosen to give stable, oscillatory dynamics). The generated sequences used in the experiments are shown in Figure 4.9. We then considered the problem of inferring the 'unknown' model parameters $\mathbf{z}$ (with the initial states, integrator time step and noise variances assumed to be known) given the observed data $\boldsymbol{x}$ and generative model.

For the parameter log-normal prior, we used location hyperparameters $m_i = -2\ \ \forall i$ and scale hyperparameters to $s_i = 1\ \ \forall i$. As in the generalised lambda distribution experiments in the previous section this choice of a relatively informative prior was motivated by trying to minimise the prior probability mass put on parameters corresponding to implausible generated sequences, with in particular in this case the Lotka–Volterra dynamics being unstable for many parameter settings, with an exponential blow-up in the prey population if the predator population 'dies off'. This exponential blow-up is both biologically implausible and causes computational issues as it can lead to numerical overflow. Biasing the prior towards smaller values was found to favour more plausible appearing sequences with stable dynamics.

We first tested several standard ABC approaches to perform inference, conditioning on the full observed data sequences i.e. without use of summary statistics. ABC rejection using a uniform ball kernel failed catastrophically, with no acceptances in 10^6 samples even with a very

large tolerance $\epsilon = 1000$. Pseudo-marginal ABC MCMC with a Gaussian random-walk proposal distribution also performed very poorly with the dynamic having zero acceptances over multiple runs of 10^5 updates for $\epsilon = 100$ and getting stuck at points in parameter space over thousands of iterations for larger $\epsilon = 1000$, even with very small proposal steps. Similar issues were also observed when attempting to run pseudo-marginal ABC MCMC chains using a Gaussian kernel. This poor performance is not unexpected, but highlights the challenges of working with high-dimensional observations in standard ABC approaches.

We next attempted to reduce the dimensionality of the observed data and generated observations by using a set of summary statistics. We used the nine summary statistics employed in a similar Lotka–Volterra inference problem in [202] - the means and log variances of the two sequences, lag one and lag two autocorrelation coefficients and cross-correlation coefficient of the sequences. To account for the differing scales of the nine statistics, they were normalised by dividing each by the empirical standard deviations of the summary statistics of 10^6 observed sequences generated from the prior, with any sequences in which numerical overflow occured or either population exceeded 10^5 excluded from the standard deviation estimates.

Even when reducing to this much lower dimensional space, ABC reject continued to have a very low accept rate, with only 22 accepted samples from 5×10^7 draws from the prior for a uniform ball kernel with $\epsilon = 2$ and no accepted samples from 5×10^7 draws from the prior using $\epsilon = 1$. Using this set of summary statistics we were however able to successfully run pseudo-marginal[8] ABC MCMC chains which appeared to converge (PSRF statistic of $\hat{R} = 1.00$ across ten independent chains of 500 000 samples) when using a uniform ball kernel with $\epsilon = 1$ on the nine-dimensional normalised summary statistics.

A histogram of the resulting estimated marginal posteriors on the model parameters from the last 250 000 samples of ten 500 000 sample chains is shown in Figure 4.10 with the orange dashed lines indicating the values of the parameters used to generate the data. The data generating parameters are located within the mass of the estimated marginal posteriors of z_1, z_2 and z_3 parameters, however this is not the case for the

8 We in fact used a 'split' auxiliary pseudo-marginal MI+MH form of update as discussed in Chapter 3, as this simplified tuning of the Gaussian random-walk proposal step-size to give an accept rate of ~ 0.234 as discussed previously.

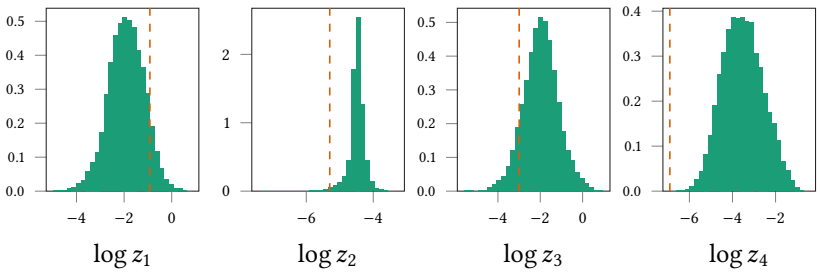


Figure 4.10.: Estimated marginal posterior distributions of Lotka–Volterra model parameters using random-walk Metropolis pseudo-marginal ABC MCMC chains with nine-dimensional summary statistics and a uniform ball kernel with $\epsilon = 1$. Each histogram corresponds to last 250 000 samples from ten independent chains of 500 000 samples. The orange dashed line on each axis indicates the value of the parameter used to generate the data.

estimated marginal posterior of the z_4 parameter with the data generating value outside the range of the posterior samples. Although there is nothing to guarantee that the true posterior is centred at the parameters used to generate the data[9] the discrepancy between where the posterior mass is located and the parameters used to generate the data is potentially concerning. We will see in later results that the posterior distributions conditioned on the summary statistics of generated observations exactly matching the data summary statistics appears to be concentrated around the data generating parameters as does the ABC posterior when conditioning on all of the data. It therefore seems that it may be the combined use of summary statistics and an ABC kernel which is leading to a non-representative posterior distribution estimate (in the sense of being representative of the true posterior we are interested in). These issues highlight the challenges in assessing the impact of the choice of summary statistics and tolerance on the inferred posterior in ABC methods.

As the generative model here is differentiable we are able to apply our proposed constrained HMC method in the input space of the generator to construct chains directly targeting the posterior distribution of interest, constraining the output of the generator to be equal to the observed data (to within a 10^{-8} infinity norm distance used as the convergence tolerance in the Newton iteration). We ran ten independent constrained HMC chains of 1000 samples, using an integrator time step

9 In general we would only expect the parameters to be a plausible sample under the posterior; if the parameters were sampled from the prior then they would represent an exact sample from the posterior given the generated data.

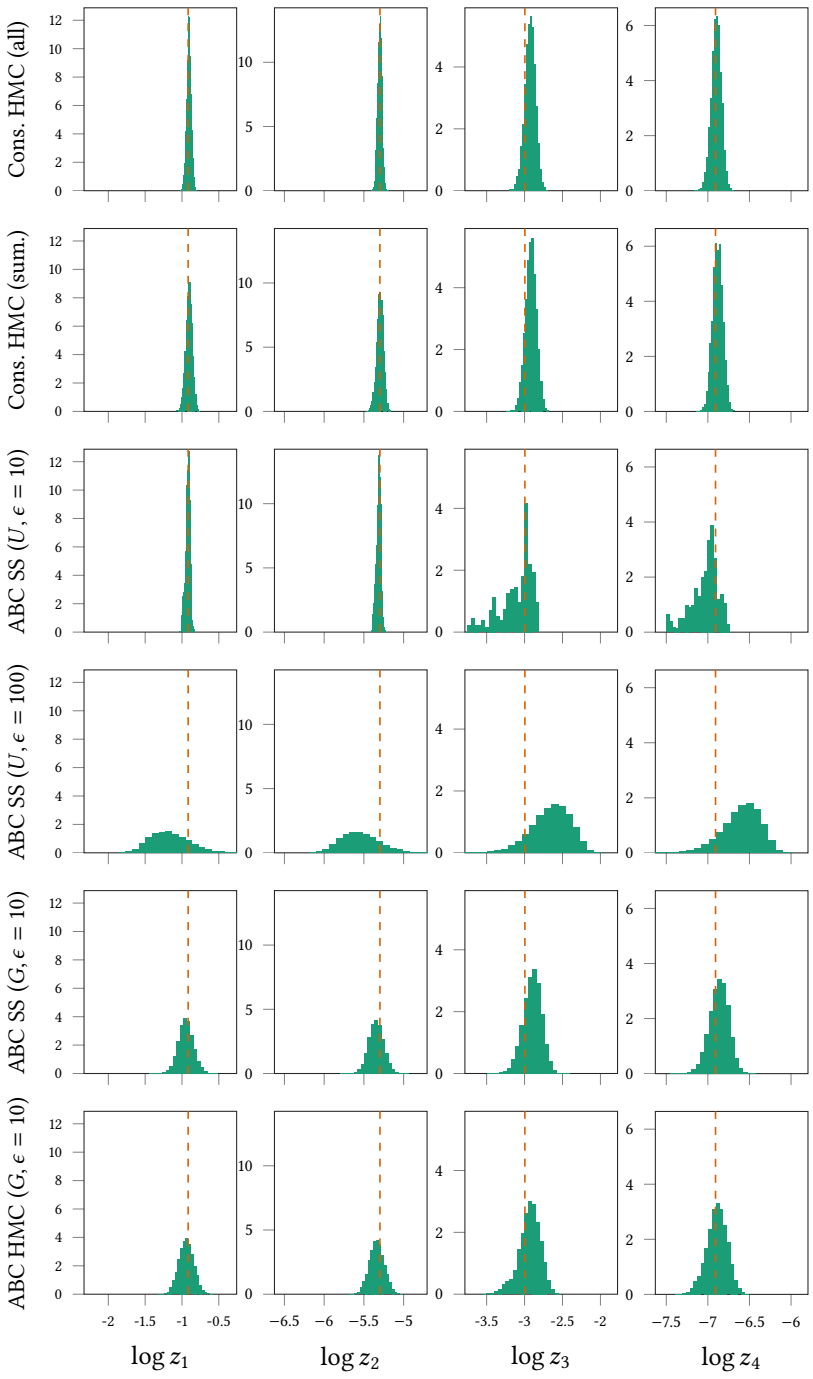


Figure 4.11.: Estimated marginal posterior distributions of Lotka–Volterra model parameters. Each row corresponds to samples from ten independent chains for MCMC method labelled to left of plot, while each column corresponds to one of the four distribution parameters, labelled to bottom of plot. The orange dashed line on each axis indicates the value of the parameter used to generate the data.

$\delta t = 0.25$, the number of integrator time steps per proposed update N_s uniformly sampled from $[4, 8]$ on each iteration, $N_i = 3$ inner geodesic steps per update and a Newton convergence tolerance of $\epsilon = 10^{-8}$. As in the experiments in the previous section, the initial states for the chains were computed by sampling random values of the input variables $\mathbf{u}_1$ corresponding to the model parameters and then solving for the values of the remaining random inputs $\mathbf{u}_2$ giving a generated output equal to the observed data using the `fsolve` optimisation routine in `SciPy` [134]. Based on visualisation of traces, the first ten iterations of each chain were removed as 'warm-up' iterations.

To test how informative the nine summary statistics used in the ABC MCMC methods are, we also ran constrained HMC chains in the posterior distribution formed by constraining the summary statistics of the generated observed variables $\mathbf{x}$ to be equal to the summary statistics of the observed data $\boldsymbol{x}$. As the summary statistics are all differentiable functions of the generated observations here, we can simply define an augmented generator which outputs summaries rather than full observations and use this in the constrained HMC update in Algorithm 11. We used the same initial input states and algorithm settings for these chains as for the full data case.

The resulting estimates of the marginal posteriors on the model parameters formed using the samples from the constrained HMC chains run in these two cases are shown in the top two rows in Figure 4.11, with the top row (labelled 'Cons. HMC (all)') corresponding to the posterior distribution conditioned on all of the data, and the second row (labelled 'Cons. HMC (sum.)') corresponding to the posterior distribution conditioned on just the summary statistics of the data. It can be seen that in both cases the estimated posteriors are concentrated around the parameter values used to generate the data (indicated by orange dashed lines), unlike the previous results in Figure 4.10. Interestingly there seems to be minimal loss of information about the parameters when conditioning on just the summary statistics rather than the full data here, with the estimated marginal posteriors for the summary statistics chains only barely noticeably more diffuse than the corresponding marginal posterior estimates for the full data chains. In both cases the estimated PSRF statistics across the 10 chains are $\hat{R} = 1.00$.

We also tested our proposed approach of performing ABC inference by running chains in the input space to the generator, as discussed in Sec-

tion 4.6. Encouragingly we found that by performing MCMC updates to the random inputs to the generator, we are able to tractably perform ABC inference when conditioning on the full observed data, even when using relatively simple non-gradient based MCMC methods. In particular based on the auxiliary pseudo-marginal slice sampling methods discussed in Chapter 3, we tried using alternating elliptical slice sampling updates of the random inputs $\mathbf{u}_1$ used to generate the parameters, i.e. $\mathbf{z} = \mathbf{g}_{\mathbf{z}}(\mathbf{u}_1)$, and remaining random inputs $\mathbf{u}_2$ used to generate the simulated observations given parameters, i.e. $\mathbf{x} = \mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z}, \mathbf{u}_2)$. Using this method, which has zero free parameters to tune, we were able to construct chains which appeared to converge to a reasonable approximate posterior both when using a uniform ball kernel with $\epsilon = 100$ and $\epsilon = 10$ and when using a Gaussian kernel with $\epsilon = 10$. We also ran HMC chains with a $\epsilon = 10$ Gaussian kernel, using an integrator time step $\delta t = 2.5 \times 10^{-3}$ and a number of leapfrog steps per update L sampled uniformly from $[10, 20]$. For the slice sampling approaches we ran 10 independent chains of 60 000 samples for each kernel and tolerance combination, discarding the first 30 000 samples as warm-up iterations, and for the HMC case we ran 10 independent chains of 10 000 samples, discarding the first 5000 samples as warm-up iterations.

Estimates of the marginal posterior distributions on the parameters for these chains are shown in the last four rows of Figure 4.11, with the label 'ABC SS' indicating elliptical slice sampling chains and 'ABC HMC' the HMC chains and a U in parenthesis in the label indicating use of uniform ball kernel and a G in parenthesis in the label G a Gaussian kernel, with in both cases the corresponding ϵ tolerance also given in the parentheses. It is immediately evident that the estimated ABC marginal posteriors here are more diffuse than for the constrained HMC chains, particularly for the slice sampling chains using a uniform ball kernel with $\epsilon = 100$ (fourth row), though the estimated posteriors are still significantly more concentrated than for the summary statistic based ABC posterior shown in Figure 4.10 (note the difference in the horizontal scales compared to Figure 4.11). Unlike the summary-statistic based pseudo-marginal ABC MCMC estimates in Figure 4.10 though, all of the marginal posterior estimates for these ABC methods using the full set of observations are concentrated around the region of the parameters used to generate the data, and seem to broadly consistent, albeit more diffuse, with the constrained HMC estimates of the true posterior.

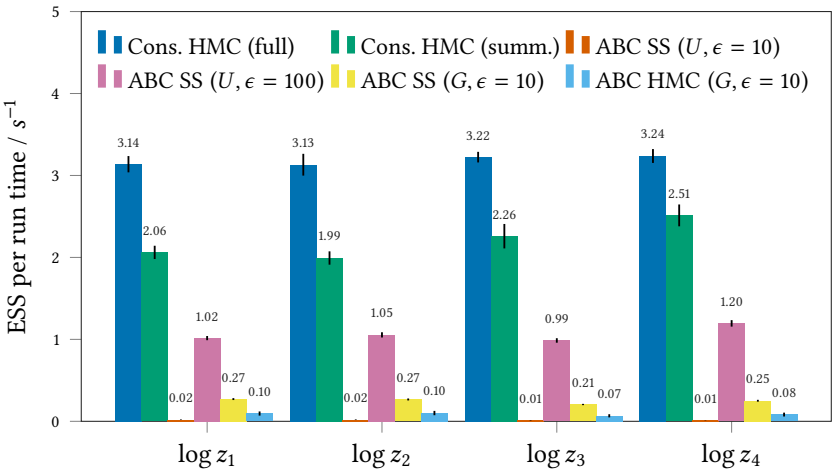


Figure 4.12.: Estimated ESS for posterior means of each Lotka–Volterra model parameters normalised by chain run time. Each coloured set of bars corresponds to mean estimated ESS per run time across ten independent chains for the method indicated in the legend (key: SS - slice sampling, summ. - summary statistics, G - Gaussian kernel, U - uniform ball kernel). The ticks on the bars show ±1 standard error of mean.

Between the four different approaches, the estimates of the ABC posterior using a uniform ball kernel with $\epsilon = 10$ (third row) are closest to the constrained HMC estimates of the true posterior (first row), with in particular the marginal estimates for z_1 and z_2 visually very similar. For the z_3 and z_4 marginal posterior estimates in the uniform ball kernel $\epsilon = 10$ case there are spurious appearing peaks however suggesting possible convergence issues and this is backed up by a PSRF statistic of $\hat{R} = 1.89$ across the 10 chains. Although not as visible in the marginal posterior estimates, the Gaussian kernel HMC chains also suffered convergence issues, with a PSRF statistic of $\hat{R} = 1.33$ across the 10 chains. The $\epsilon = 10$ Gaussian kernel and $\epsilon = 100$ uniform ball kernel slice sampling chains both had estimated PSRF statistics of $\hat{R} = 1.01$.

We also measured the sampling efficiency of the chains generated using the different approaches by computing ESS estimates for each parameter and normalising by the total chain run-time (including warm-up iterations), with the results shown in Figure 4.12. As there are quite significant differences between the distributions being targeted by the chains of most of the methods, as well as potential differences in the relative efficiencies of the implementations, the run time normalised ESS estimates can only give a rough indication of relative performance.

Subject to those provisos however, the results suggest the constrained HMC methods are potentially significantly more efficient than the alternative approaches here, despite also giving in some sense the closest estimates of the true posterior.

Somewhat counter-intuitively perhaps, the constrained HMC chains conditioned on the full data showed higher sampling efficiency than those conditioned on the reduced dimension summary statistics. Given the earlier statement that a dominant cost in the constrained HMC algorithm is the computation of the Cholesky decomposition of the generator Jacobian product, $\text{chol}\,\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}^\mathsf{T}$, which in general will scale cubically with the dimension of the generator output, it might be expected that projecting the generator output to a lower-dimensional space would lead to lower-cost updates and so improved efficiency. While this may be the case in some settings, here there is the additional factor that the generator Jacobian for the full observations case has the triangular block structure discussed in Section 4.9.2, which allows a more efficient computation of the Cholesky factor using low-rank updates. This structure is lost when projecting down to lower dimensions, and so a standard cubic-cost Cholesky factorisation routine needs to be used. In models without such structure however and with large numbers of observed variables, projecting down to lower-dimensional summaries could be an important method for improving the scalability of the constrained HMC approach (providing the summaries are differentiable functions of the observations), and as shown in the example here, in some cases may entail minimal loss of information about the unobserved variables being inferred.

4.10.3 Human pose and camera model inference

For the final experiment in this Chapter we consider inference in an differentiable generative model for human poses. In particular we consider the task of inferring a three-dimensional human pose given binocular two-dimensional projections of joint positions, using a learnt prior model of poses from motion capture and anthropometric data, and a simple projective pin-hole camera model.

We parameterised the poses used a 19 joint skeleton model, with degrees of freedom specifying the angular configurations of each joint and the lengths of the bones between joints. In total the model has 47 local joint angles $\mathbf{z}_a$ (with some joints, for example those correspond-

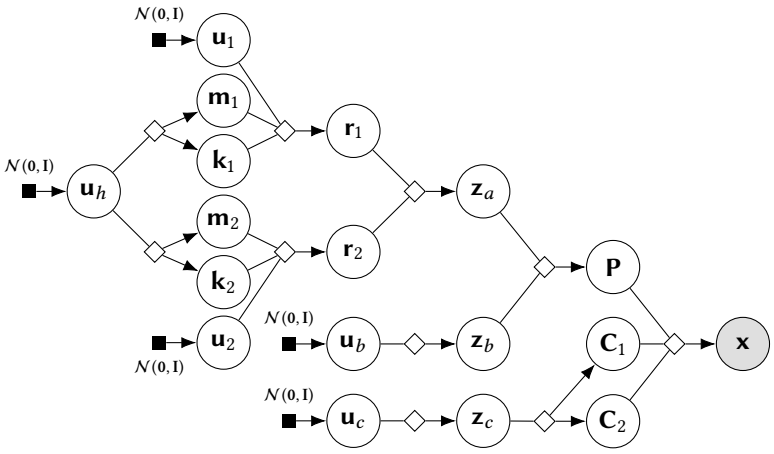


Figure 4.13.: Factor graph of human pose differentiable generative model. The operations corresponding to the deterministic nodes ($\diamond$) in the graph are described in Algorithm 12.

Algorithm 12 Human pose model generator functions.

Input:

$\{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=0}^{L}$: parameters of pose angle differentiable network;
$\boldsymbol{\mu}_b$, $\boldsymbol{\Sigma}$: mean and covariance of skeleton bone lengths;
$\boldsymbol{\mu}_{c,:2}$, $\boldsymbol{\sigma}_{c,:2}$: camera x, y coordinates normal prior parameters;
$\mu_{c,2}$, $\sigma_{c,2}$: camera z coordinate log-normal prior parameters;
JointPositions : maps pose angles and bone lengths to joint positions;
CameraMatrices : maps camera parameters to a pair of camera matrices;
Project : uses camera matrix to map world to image coordinates;
Partition : partitions a vector in a specified number of equal length parts;
Flatten : flattens a multidimensional array to a vector.

function $\mathbf{g}_\mathbf{z}([\mathbf{u}_h; \mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_b; \mathbf{u}_c])$
  $\mathbf{h}_L \leftarrow$ DifferentiableNetwork$(\mathbf{u}_h)$
  $\mathbf{m}_1, \mathbf{k}_1, \mathbf{m}_2, \mathbf{k}_2 \leftarrow$ Partition$(\mathbf{h}_L, 4)$
  $\mathbf{r}_1 \leftarrow \exp(\mathbf{k}_1) \odot \mathbf{u}_1 + \mathbf{m}_1$
  $\mathbf{r}_2 \leftarrow \exp(\mathbf{k}_2) \odot \mathbf{u}_2 + \mathbf{m}_2$
  $\mathbf{z}_a \leftarrow \text{atan2}(\mathbf{r}_2, \mathbf{r}_1)$
  $\mathbf{z}_b \leftarrow \exp(\boldsymbol{\mu}_b + \boldsymbol{\Sigma}_b \mathbf{u}_b)$
  $\mathbf{z}_{c,:2} \leftarrow \boldsymbol{\sigma}_{c,:2} \odot \mathbf{u}_{c,:2} + \boldsymbol{\mu}_{c,:2}$
  $\mathrm{z}_{c,2} \leftarrow \exp(\sigma_{c,2} \mathrm{u}_{c,2} + \mu_{c,2})$
  return $[\mathbf{z}_a; \mathbf{z}_b; \mathbf{z}_c]$
function DifferentiableNetwork$(\mathbf{u}_h)$
  $\mathbf{h}_0 \leftarrow \tanh(\boldsymbol{W}_0 \mathbf{u}_h + \boldsymbol{b}_0)$
  for $\ell \in \{1 \ldots L-1\}$ **do**
    $\mathbf{h}_\ell \leftarrow \tanh(\boldsymbol{W}_\ell \mathbf{h}_{\ell-1} + \boldsymbol{b}_\ell) + \mathbf{h}_{\ell-1}$
  return $\boldsymbol{W}_L \mathbf{h}_{L-1} + \boldsymbol{b}_L$
function $\mathbf{g}_{\mathbf{x}|\mathbf{z}}([\mathbf{z}_a; \mathbf{z}_b; \mathbf{z}_c])$
  $\mathbf{P} \leftarrow$ JointPositions$(\mathbf{z}_a, \mathbf{z}_b)$
  $\mathbf{C}_1, \mathbf{C}_2 \leftarrow$ CameraMatrices$(\mathbf{z}_c)$
  $\mathbf{X}_1 \leftarrow$ Project$(\mathbf{C}_1, \mathbf{P})$
  $\mathbf{X}_2 \leftarrow$ Project$(\mathbf{C}_2, \mathbf{P})$
  return Flatten$([\mathbf{X}_1; \mathbf{X}_2])$

ing to knees and elbows, not having a full three degrees of freedom). The prior over the joint angles was specified by a Gaussian VAE model trained on the *PosePrior* motion caption dataset [3]. The circular topology of the angular data is poorly matched by the Euclidean space a Gaussian VAE typically learns a distribution on, and simply 'unwrapping' the angles to e.g. $[-\pi, \pi)$ leads to unnatural discontinuities at the $\pm\pi$ cut-point, this both making the initial learning problem challenging (as there is no in-built prior knowledge of continuity across the cut-point) and tending to lead to a learned latent space less amenable to MCMC inference as 'nearby' poses with one or more joint angles on opposite sides of the cut-point will likely end up corresponding to points far apart in the latent space.

During training we therefore mapped each vector of 47 joint angles $\boldsymbol{z}_a^{(i)}$ (corresponding to a single motion capture datapoint) to a pair of 47-dimensional vectors $(\boldsymbol{r}_1^{(i)}, \boldsymbol{r}_2^{(i)})$ by sampling a Gaussian random vector $\boldsymbol{n}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then computing $\boldsymbol{r}_1^{(i)} = \exp \boldsymbol{n}^{(i)} \odot \cos \boldsymbol{z}_a^{(i)}$ and $\boldsymbol{r}_2^{(i)} = \exp \boldsymbol{n}^{(i)} \odot \sin \boldsymbol{z}_a^{(i)}$ and training the VAE to maximise (a variational lower bound) on the joint marginal density of the $\{\boldsymbol{r}_1^{(i)}, \boldsymbol{r}_2^{(i)}\}_i$ pairs. At the cost of doubling the dimension, this leads to an embedding in a Euclidean space which does not introduce any arbitrary cut-points and empirically seemed to lead to better sample quality from the learned generative model compared to learning the angles directly. Given the trained model we can generate a vector of angles $\mathbf{z}_a$ using the model by sampling a Gaussian code (latent representation) vector $\mathbf{u}_h$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ then sampling a pair of 47-dimensional vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ from the learnt Gaussian decoder model given $\mathbf{u}_h$ (and further Gaussian random input vectors $\mathbf{u}_1$ and $\mathbf{u}_2$), and finally recovering an angle by computing $\mathbf{z}_a = \text{atan2}(\mathbf{r}_2, \mathbf{r}_1)$. The resulting distribution on $\mathbf{z}_a$ is only implicitly defined, but the overall generative model is differentiable with respect to the input vectors $\mathbf{u}_h$, $\mathbf{u}_1$ and $\mathbf{u}_2$.

The *PosePrior* motion capture data includes recordings from only a relatively small number of distinct actors and so limited variation in the 'bone-lengths' of the skeleton model. Therefore a separate log-normal model for the bone lengths $\mathbf{z}_b$ was fitted using data from the *ANSUR* anthropometric data-set [111], due to symmetry in the skeleton thirteen independent lengths being specified.

A simple pin-hole projective camera model with three position parameters $\mathbf{z}_c$ and fixed focal-length was used[10]. A log-normal prior distribution was placed on the depth co-ordinate $z_{c,2}$ to enforce positivity with normal priors on the other two co-ordinates $z_{c,0}$ and $z_{c,1}$.

Given a generated triplet of joint-angles, bone length and camera parameters $\mathbf{z}_a$, $\mathbf{z}_b$ and $\mathbf{z}_c$, a binocular pair of simulated two-dimensional projection of the skeleton $\mathbf{x}$ are generated by first mapping the joint-angles and bone-lengths to a 4×19 matrix of joint positions $\mathbf{P}$ in (homogeneous) world-coordinates by recursing through the skeleton tree. Two 3×4 projective camera matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ are generated from $\mathbf{z}_c$ (with a fixed known offset between the camera centres) and then used to project the world-coordinate joint positions to two 2×19 matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ of joint positions in two-dimensional image-coordinates. The projected positions matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ are flattened to a vector to give the $19 \times 2 \times 2 = 76$ dimensional observed vector $\mathbf{x}$. The overall corresponding model generator functions $\mathbf{g}_{\mathbf{x}|\mathbf{z}}$ and $\mathbf{g}_{\mathbf{z}}$ are described procedurally in Algorithm 12 and a factor graph summarising the relationships between the variables in the model shown in Figure 4.13.

The generator $\mathbf{g}_{\mathbf{x}}$ here has a complex form, not corresponding to either the independent or Markovian observations structures discussed in Section 4.9.2. The total input dimension is $M = 140$ and output dimension $N_{\mathbf{x}} = 76$. The resulting generator Jacobian $\mathbf{J}_{\mathbf{g}_{\mathbf{x}}}$ is not full row-rank across the input space; for this binocular observation case this is not surprising as there are a maximum of $19 \times 3 = 57$ true degrees of freedom in the skeleton model (three degrees of freedom for each joint) versus the $N_{\mathbf{x}} = 76$ observed dimensions. We therefore define an 'augmented' noisy generator $\mathbf{g}_{\mathbf{y}}(\mathbf{u}, \mathbf{n}) = \mathbf{g}_{\mathbf{x}}(\mathbf{u}) + \epsilon\mathbf{n}$ to use to perform inference with as discussed in Section 4.6. We set the noise standard deviation $\epsilon = 0.01$ which produces a non-obvious level of perturbation in visualisations of the generated two-dimensional projections. Similar to the earlier discussion of ABC kernels, we can either consider this additional noise as a computational approximation or as part of the model, representing for example the measurement noise that would be present in two-dimensional joint positions derived from image data.

10 The camera orientation was assumed fixed to avoid replicating the degrees of freedom specified by the angular orientation of the root joint of the skeleton: only the relative camera–skeleton orientation is important.

Under this noisy generator model, the joint density on the generated outputs $\mathbf{y}$ and inputs $\mathbf{u}$ has an explicit form

$$\mathrm{p}_{\mathbf{y},\mathbf{u}}(\boldsymbol{y}, \boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{y} \mid \mathrm{g}_{\mathbf{x}}(\boldsymbol{u}),\, \epsilon^2 \boldsymbol{I}\right) \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \boldsymbol{I}), \tag{4.68}$$

and so the resulting posterior density $\mathrm{p}_{\mathbf{u}|\mathbf{y}}$ on the model inputs $\mathbf{u}$ given observed $\mathbf{y}$ values can be evaluated up to a normalising constant. This is directly equivalent to an ABC posterior density in the input space (4.23) when using a Gaussian kernel.

We generated three sets of binocular two-dimensional joint position projections to use as the observed data for the inference experiments using the *noisy* generator function; these are shown in the left column of Figure 4.14. Given each of these observed binocular joint projections, we then attempted to infer the corresponding plausible values for the model inputs $\mathbf{u}$ and so by consequence as they are a deterministic function of the inputs, the latent variables $\mathbf{z}_a$, $\mathbf{z}_b$ and $\mathbf{z}_c$ defining the three-dimensional scene information.

We ran chains using the proposed constrained HMC method with the noisy generator $\mathrm{g}_{\mathbf{y}}$, the chain state in this case being defined as both $\mathbf{u}$ and $\mathbf{n}$, and chains using standard HMC transitions on the posterior density $\mathrm{p}_{\mathbf{u}|\mathbf{y}}$ on the model inputs $\mathbf{u}$. We used a integrator step size of $\delta t = 0.05$ for the constrained HMC chains, $N_g = 8$ inner geodesic steps and a number of integrator steps per proposed update N_s uniformly sampled from $[10, 20]$. We ran five chains of 200 samples each from independent initialisations. To compute the initial states for the chains, we generated $\mathbf{u}$ values independently from the $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ prior and then set the $\mathbf{n}$ values to $\frac{1}{\epsilon}\left(\boldsymbol{x} - \mathrm{g}_{\mathbf{y}}(\boldsymbol{u})\right)$ where $\boldsymbol{u}$ are the values sampled from the prior and $\boldsymbol{x}$ is the observed data being conditioned on.

For the standard HMC chains we used an integrator step size of $\delta t = 2 \times 10^{-4}$ and a number of leapfrog steps L per proposed update randomly sampled from $[100, 200]$. We again ran five chains, independently sampling the initial $\mathbf{u}$ state from the normal prior, and running each chain for 1200 samples. The small integrator step size used here was the result of higher step sizes leading to some chains accepting very few or in some cases no updates. This issue was also encountered when using smaller numbers of leapfrog steps per update, including just one integrator step, with these chains however showing marginally slower overall run-time adjusted convergence. Given the small size of ϵ here

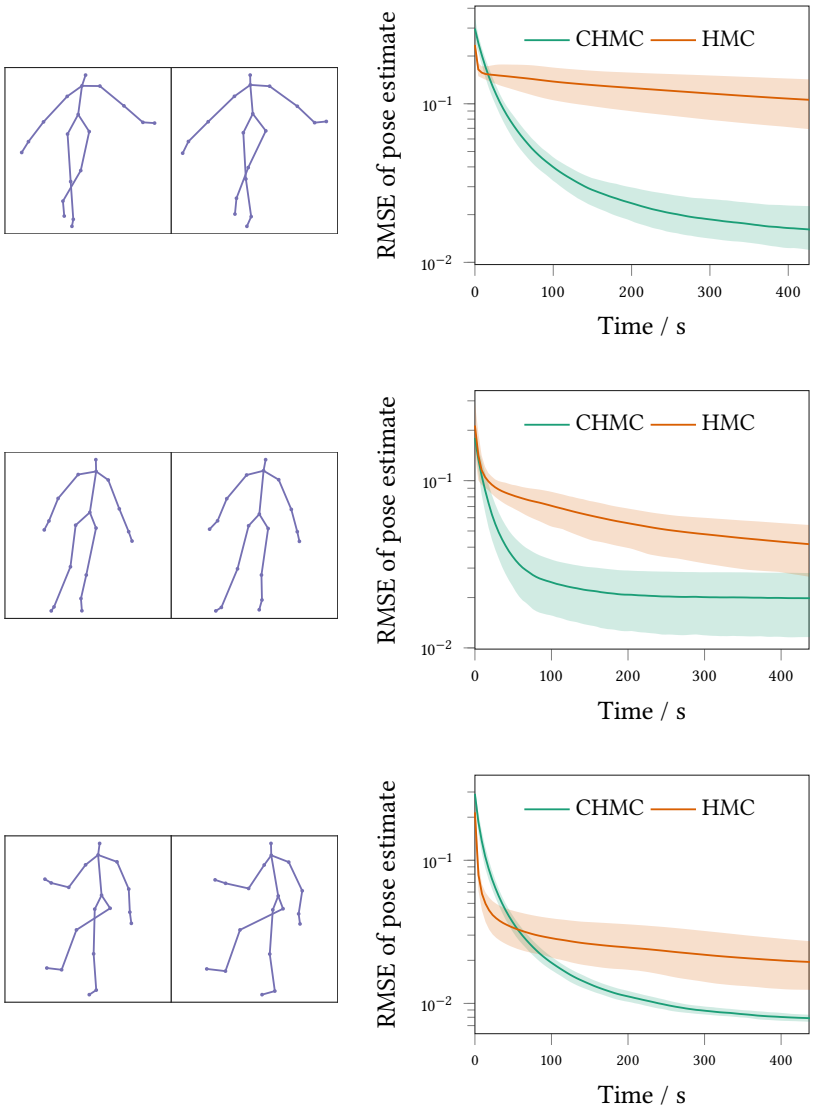


Figure 4.14.: Results of binocular pose inference experiments. In each row the left column shows the generated binocular two-dimensional joint position projections used as the observed data for inference. The right column shows plots of the RMSEs of posterior mean estimates of true 3D pose used to generate the binocular projections, using samples from a constrained HMC chains (green) versus standard HMC chains (orange). The horizontal axes of the plot show computation time to produce number of samples in the estimate. Solid curves are averages of RMSEs over five chains independently initialised from the prior and shaded regions show ±3 standard deviations.

and the resulting tight concentration of the posterior mass around the fibre $\mathbf{g}_{\mathbf{x}}^{-1}(\boldsymbol{x})$, the need for a small step size when using an unconstrained HMC approach is not surprising, however the ability of the constrained HMC updates to support a much larger step size is encouraging.

Due to the high-dimensional nature of the latent variables being inferred it is challenging to assess the convergence of the chains here. As a proxy measure for convergence, we computed the *root mean squared error* (RMSE) of an estimate of the three-dimensional joint positions compared the true positions, computed using the mean of the three-joint positions generated from the posterior $\mathbf{u}$ input samples. In this binocular case, the disparity in projected joint positions between the two projections gives information about the distances of the corresponding joints from the image plane in the depth direction and so we would expect the posterior distribution on the three-dimensional pose to be concentrated around the true values used to generate the observations.

The right column of Figure 4.14 shows the RMSE values as a function of the computation time taken to generate the number of samples included in the estimate, for each of the three generated scenes (with the projection visualisations corresponding to the RMSE plots in each row). The constrained HMC chains (green curves) consistently give position estimates which converge more quickly towards the true positions. In this case standard HMC performs relatively poorly despite the significantly cheaper cost of each integrator step compared to the constrained dynamics. The posterior distribution on the model inputs appeared to be multimodal here, with the chains often appearing to converge to slightly different modes. Visually inspecting the sampled poses and individual run traces (not shown) it seems that there are a number of local modes corresponding to a small subset of joints being 'incorrectly' positioned; both the HMC and constrained HMC chains are likely to be equally susceptible to getting stuck in local modes, as neither dynamic is likely to overcome large energy barriers.

To highlight the generality of the approach, we also considered inferring three-dimensional scene information from a single two-dimensional projection. Monocular projection is inherently information destroying with significant uncertainty to the true pose and camera parameters which generated the observations. Figure 4.15 shows pairs of orthographic projections of 3D poses: the left most column is the pose used to generate the projection conditioned on and the right three columns

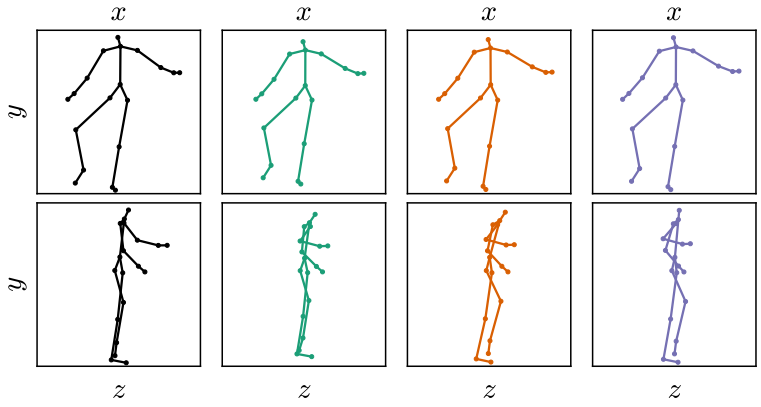


Figure 4.15.: Orthographic projections (top: front view, bottom: side view) of 3D poses consistent with monocular projections. Left most pair (black) shows pose used to generate observations, right three show constrained HMC samples.

are poses sampled using constrained HMC consistent with the observations. The top row shows front x–y views, corresponding to the camera view though with a orthographic rather than perspective projection, the bottom row shows side z–y views with the z axis the depth from the camera. The dynamic is able to move between a range of plausible poses consistent with the observations while reflecting the inherent depth ambiguity from the monocular projection.

4.11 DISCUSSION

The formulation of generative models as deterministic functions of random input variables discussed at the beginning of this chapter proved a useful intuition for considering more efficient MCMC approaches to performing approximate inference in implicit generative models. Under this description of a generative model, the task of MCMC inference can be reposed from the problem of constructing a Markov chain exploring the posterior distribution on the latent variables $\mathbf{z}$ of direct interest in the downstream task, to constructing a chain targeting the posterior distribution on all the random inputs $\mathbf{u}$ to the model generator.

From a computational perspective this may seem a potentially rash idea. The dimensionality of the inputs $\mathbf{u}$ will typically be significantly higher than the dimensionality of the latent variables $\mathbf{x}$. Trying to infer all of $\mathbf{u}$ when we are only directly interested in the subset which are used to generate $\mathbf{z}$ might appear as if it is increasing the dimensionality of the inference problem for little gain. However as we saw in the previous

chapter pseudo-marginal MCMC methods, which include the standard ABC MCMC algorithm, can be considered as Metropolis–Hastings transition operator on an augmented state space including auxiliary random inputs, in which updates to these auxiliary variables are proposed independently of the previous values. Although independently proposing updates to the auxiliary variables by drawing values directly from a random number generator conceals their presence in the algorithm by not requiring them to be explicitly enumerated, the succession of values drawn still implicitly defines a Markov chain on these variables.

The approaches we propose in this chapter make the construction of a chain on the joint space of all random inputs explicit. The explicit form of the ABC posterior density on the generator inputs (4.23) allows the application of a much greater range of MCMC methods to ABC inference problems than the standard pseudo-marginal Metropolis–Hastings algorithm. In particular it opens up the door to adaptive methods such as slice sampling and gradient-based approaches such as HMC. As we saw in the Lotka–Volterra experiments, in some cases by using these more efficient transition operators, we are able to tractably condition on all observed data when performing inference in a simulator model when standard ABC methods are only able to function at all when reducing to summary statistics. The use of slice sampling methods within ABC inference problems seems particularly promising, with the algorithms having minimal or no free parameters to tune and requiring no model gradients meaning there is a relatively low implementation overhead required to apply them to existing models.

For the restricted class of *differentiable generative models*, the constrained HMC approach we propose in some cases allows asymptotically exact inference where ABC methods might otherwise be used. The proposed algorithm helps deal with two of the key issues in ABC methods — enabling inference in continuous spaces as ϵ collapses to zero and allowing efficient inference when conditioning on high-dimensional observations without the need for dimensionality reduction with summary statistics (and the resulting task of choosing appropriate summary statistics). As well as being of practical importance itself, this approach should be useful in providing 'ground truth' inferences in more complex models to assess the affect of the approximations used in ABC methods on the quality of the inferences. The application of constrained HMC to a generator function outputting summary statistics in the Lotka–

Volterra experiments was an interesting example of this: that the posterior conditioning exactly on the summaries without use of an ABC kernel would be nearly as informative about the parameters as conditioning on all of the data was not obvious a-priori, and highlighted the complex interaction between the use of summaries and the kernel and tolerance choice in the resulting ABC posterior.

In molecular simulations, constrained dynamics are often used to improve efficiency. Intra-molecular motion is removed by fixing bond lengths. This allows a larger time-step to be used due to the removal of high-frequency bond oscillations [150]. An analogous effect is present when performing inference in an ABC setting with a ϵ kernel 'soft-constraint' to enforce consistency between the inputs and observed outputs. As $\epsilon \to 0$ the scales over which the inputs density changes value in directions orthogonal to the constraint manifold and along directions tangential to the manifold increasingly differ. To stay within the soft constraint a very small step-size needs to be used. Using a constrained dynamic decouples the motion on the constraint manifold from the steps to project on to it, allowing more efficient larger steps to be used for moving on the manifold. This was evident in the relative performances of the constrained HMC and unconstrained HMC methods in the experiments in all three models, with the use of HMC in the ABC posterior on the generator inputs typically limited in the efficiency achieved by the requirement to use a small integrator step size.

A strong limitation of the constrained HMC method is the requirement of differentiability of the generator. This prevents using the approach with generative models which use discontinuous operations or discrete random inputs. In some cases conditioned on fixed values of discrete random inputs the generator may still be differentiable and so the proposed method can be used to update the continuous random inputs given values of the discrete inputs. This would need to be alternated with updates to the discrete inputs, which would require devising methods for updating the discrete inputs to the generator while constraining its output to exactly match observations.

4.12 RELATED WORK

Similar methods to those explored in this chapter have been proposed by various authors. The *pseudo-marginal HMC* algorithm of [159] dis-

cussed at the end of the last chapter is particularly closely related, using a HMC transition operator to jointly update the target and auxiliary variables in pseudo-marginal settings, with the authors discussing the relevance of their approach to an ABC setting. Their use of a symplectic integrator that leverages additional structure in the models considered, in particular a normal marginal distribution on the auxiliary variables, contrasts to the more basic application of a standard HMC approach to ABC inference problems here.

Hamiltonian ABC [170], also proposes applying HMC to perform inference in simulator models. Rather than using reverse-mode AD to exactly calculate gradients of the generator function, *Hamiltonian ABC* uses a stochastic gradient estimator. This is based on previous work considering methods for using a stochastic gradients within HMC [60, 260]. It has been suggested however that the use of stochastic gradients can destroy the favourable properties of Hamiltonian dynamics which enable coherent exploration of high dimensional state spaces [31].

The authors of *Hamiltonian ABC* also observe that representing the generative model as a deterministic function by fixing the random inputs to the generator is a useful method for improving exploration of the state space. This is achieved by including the state (~ seed) of the PRNG in the chain state however rather than directly updating the random inputs. As pointed out by the authors, this formulation puts minimal requirements on the model implementation with most numerical computing libraries having some facility to control the internal state of the PRNG being used, simplifying the application of the method with existing legacy code. In comparison the approaches we consider will often require some re-implementation, at a minimum to explicitly enumerate the random inputs used by the generator, and for methods requiring derivatives potentially requiring re-coding in a framework supporting reverse-mode AD. This additional implementation effort comes with an associated gain however, with the ability to control the updates to all the random inputs and to make perturbative moves rather than independently resampling values, being central to the improved performance of the algorithms.

Also related is *Optimisation Monte Carlo* [171]. The authors propose using an optimiser to find parameters of a simulator model consistent with observed data (to within some tolerance ϵ) given fixed random inputs sampled independently. The optimisation is not volume-

preserving and so the Jacobian of the map is approximated with finite differences to weight the samples. Our proposed constrained HMC method also uses an optimiser to find inputs consistent with the observations, however by using a volume-preserving dynamic we avoid having to re-weight samples.

Our method also differs in treating all inputs to a generator equivalently; while the *Optimisation Monte Carlo* authors similarly identify the simulator models as deterministic functions they distinguish between parameters and random inputs, optimising the first and independently sampling the latter. This can lead to random inputs being sampled for which no parameters can be found consistent with the observations (even with a within ϵ constraint). Although optimisation failure is also potentially an issue for our method, we found this occurred rarely in practice if an appropriate step size is chosen. Our method can also be applied in cases were the number of unobserved variables is greater than the number of observed variables unlike *Optimization Monte Carlo*.

5 CONTINUOUS TEMPERING

In Chapter 2 we introduced simulated tempering [164] as an approach for dealing with two key issues with MCMC methods: improving exploration of multimodal distributions and allowing estimation of the normalising constant of the target distribution, which is often an important quantity for model comparison in Bayesian inference problems. Simulated tempering augments the Markov chain state with a discrete index variable controlling the *inverse temperature* of the system. As the inverse temperature varies, the chain moves between exploring the typically complex target distribution at high inverse temperatures to performing updates in a simpler unimodal *base distribution* at low inverse temperatures. The increased ability of the chain to move to new regions of the state space at low inverse temperatures helps improve the probability of the chain transitioning between separate modes in the target distribution. Further by computing the ratio of time spent at high and low inverse temperatures the normalising constant of the target distribution can be estimated.

Although the improved exploration of the state space and ability to estimate normalising constants offered by simulated tempering are important benefits, the algorithm can be challenging to use in practice due to sensitivity of the performance of the method to the various free parameters that need to be set. The schedule of inverse temperature values that the chain moves between must be selected, with the number of inverse temperatures chosen and their distribution across the inverse temperature range both important to getting the algorithm to perform well. More finely spaced inverse temperatures improve the ability of the chain to explore the inverse temperature space but also increase the computational demands of the method. The choice of transition operator for the updates to the inverse temperature index variable is also important, with random-walk updates tending to give a slow diffusive exploration of the inverse temperature range which can be particularly problematic when a large set of inverse temperatures is used.

It is also necessary to choose an appropriate base distribution to bridge to at low inverse temperatures, with this choice key in getting simulating tempering to perform well in high dimensional spaces. A base distribution which poorly matches the target distribution makes it unlikely the chain will mix between the low and high ends of the temperature range. Further if the volume under the target distribution (corresponding to the unknown normalising constant) differs significantly from that under the base distribution, the chain will tend to remain confined to one end of the inverse temperature range. This is typically tackled by putting prior weights on the different inverse temperature index variable values to attempt to flatten out the marginal distribution on the inverse temperatures. As appropriate values for these weights will not usually be known a-priori, generally an iterative approach is needed, with the inverse temperature distribution of initial pilot chains used to estimate the weights to use. If the mass at different inverse temperatures is very imbalanced, multiple iterations of running chains and then readjusting the weights may be required.

In the chapter we propose approaches to overcoming some of these computational issues with simulated tempering. We show how the discrete index variable used to control the inverse temperature in simulated tempering can be replaced with continuous control variable, eliminating the need to choose an appropriate number of inverse temperatures. Further the continuous nature of this auxiliary control variable means that we can apply more efficient transition operators when performing updates to the variable in the chain, including importantly in target distributions on real-valued variables allowing the control variable to be jointly updated with the target model variables with efficient HMC transition operators. This allows the proposed approach to be easily applied in probabilistic programming frameworks such as Stan [55] and PyMC3 [236] which use HMC based inference algorithms.

We also demonstrate that a principled and effective approach for choosing appropriate base distributions for high dimensional target distributions is to fit a simple approximation to the target distribution using variational inference methods. The lower bound to the target distribution normalising constant which is maximised by variational approaches can also be used to help flatten the marginal distribution on the inverse temperatures, reducing the need for multiple iterations of pilot chains to estimate weights to flatten out the inverse temperat-

ure distribution. By using flexible parametric variational inference approaches such as *automatic differentiation variational inference* (ADVI) [144], we can fit appropriate base distributions for high-dimensional target distributions with minimal need for user intervention.

The work described in this chapter has previously appeared in the conference publication

- Continuously tempered Hamiltonian Monte Carlo. Matthew M. Graham and Amos J. Storkey. *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

I was the primary author of that publication and responsible for the contributions made in the paper, as well as performing and analysing the numerical experiments described in the paper, which are reproduced in this chapter in Section 5.6.

5.1 PROBLEM NOTATION

As in the previous chapters, our goal is to be able estimate the integrals on high-dimensional spaces which arise when performing inference in probabilistic models. We assume in this chapter that the target distribution of interest P, is specified by an unnormalised density function $\tilde{p} : X \to [0, \infty)$ defined on a real-valued space $X = \mathbb{R}^D$ with respect to the Lebesgue measure. We further assume that target distribution has unbounded support on X - this is non-restrictive as distributions with bounded support can typically be transformed to distributions with unbounded support using a suitable bijective map and the change of variables formula (1.19).

The unnormalised density function $\tilde{p}$ can be related to an energy function $\phi : X \to \mathbb{R}$ by $\phi(\boldsymbol{x}) = -\log \tilde{p}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in X$. The normalising constant Z of the target density is then defined as

$$Z = \int_X \exp(-\phi(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{x}. \tag{5.1}$$

In the chapter we will consider methods both for estimating expectations with respect the target distribution and estimating the normalising constant Z, which for target distributions corresponding to the posterior distribution in a Bayesian inference problem, will correspond to the model evidence term or marginal likelihood.

Algorithm 13 Annealed importance sampling.

Input: ϕ : energy function corresponding to target distribution P, ψ : energy function corresponding to base distribution Q, $\{\beta_k\}_{k=0}^{K}$: increasing sequence of inverse temperatures satisfying (5.2), $\{\overrightarrow{\mathsf{T}}_k\}_{k=0}^{K-1}$: set of transition operators with T_k leaving the distribution with density $\pi(\boldsymbol{x} \mid \beta_k)$ invariant.
Output: $(\boldsymbol{x}_{K-1}, \ell_{K-1})$: importance sample and log importance weight.

$\boldsymbol{x}_0 \sim Q(\cdot)$
$\ell_0 \leftarrow \beta_1(\psi(\boldsymbol{x}_0) - \phi(\boldsymbol{x}_0))$
for $k \in \{1 \ldots K-1\}$ **do**
  $\boldsymbol{x}_k \sim \mathsf{T}_k(\cdot \mid \boldsymbol{x}_{k-1})$
  $\ell_k \leftarrow \ell_{k-1} + (\beta_{k+1} - \beta_k)(\psi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_k))$
return $(\boldsymbol{x}_{K-1}, \ell_{K-1})$

We will briefly reintroduce the notation and terminology we used to describe simulated tempering in Section 2.3.3 in Chapter 2. An ordered sequence of *inverse temperatures* $\{\beta\}_{k=0}^{K}$ are chosen such that

$$0 = \beta_0 < \beta_1 < \beta_2 < \cdots < \beta_{K-1} < \beta_K = 1. \tag{5.2}$$

A discrete auxiliary *temperature index* variable $\mathsf{k} \in \{1 \ldots K\}$ is introduce in addition to the vector of *target variables* $\mathbf{x} \in X$. The joint density on k and $\mathbf{x}$ is then defined as

$$\mathrm{p}_{\mathbf{x},\mathsf{k}}(\boldsymbol{x}, k) = \frac{1}{C} \exp(-\beta_k \phi(\boldsymbol{x}) - (1 - \beta_k)\psi(\boldsymbol{x}) + w_k). \tag{5.3}$$

The values $\{w_k\}_{k=0}^{K}$ are the *prior weights* associated with each inverse temperature value and $\psi : X \to \mathbb{R}$ defines an energy function for the *base distribution* Q with normalised density $q(\boldsymbol{x}) = \exp(-\psi(\boldsymbol{x}))$ with respect to the Lebesgue measure on X. Simulated tempering then constructs a Markov chain on $X \times \{1 \ldots K\}$ which has the distribution defined by the joint density (5.3) as its unique invariant distribution, by alternating updates of the index variable k given the target variables $\mathbf{x}$ and updates of the target variables $\mathbf{x}$ given the index variable k, as described in Algorithm 7.

5.2 ANNEALED IMPORTANCE SAMPLING

An alternative approach to simulated tempering for using an ensemble of distributions corresponding to different inverse temperatures within a Monte Carlo approximate inference method, is the *annealed importance sampling* (AIS) algorithm proposed by Neal [189]. AIS is a popu-

lar method in the machine learning literature for normalising constant estimation, e.g. [233, 264]. We will briefly review the basics of the algorithm and some related methods, as we compare AIS to our proposed approaches in the numerical experiments in Section 5.6.

As in simulated tempering, AIS specifies a base distribution Q on the target space X with a tractable normalised density $q(\boldsymbol{x}) = \exp(-\psi(\boldsymbol{x}))$ and which we can tractably draw independent samples from. An increasing sequence of inverse temperatures $\{\beta_k\}_{k=0}^K$ defined as in (5.2) are chosen, and used to define a series of distributions geometrically bridging between the densities of the base and target distributions. The distribution corresponding to an inverse temperature β is defined as having density

$$\pi(\boldsymbol{x} \,|\, \beta) = \frac{1}{z(\beta)} \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x})) \tag{5.4}$$

with the inverse temperature dependent normaliser $z(\beta)$ being termed the *partition function* and defined as

$$z(\beta) = \int_X \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}. \tag{5.5}$$

A sequence of $K-1$ Markov transition operators $\{\vec{\mathsf{T}}_k\}_{k=1}^{K-1}$ are defined, with the k^{th} transition operator leaving the distribution with density $\pi(\boldsymbol{x} \,|\, \beta_k)$ invariant. If $\vec{\mathsf{t}}_k$ is the transition density associated with the transition operator $\vec{\mathsf{T}}_k$ then this means that

$$\pi(\boldsymbol{x}' \,|\, \beta_k) = \int_X \vec{\mathsf{t}}_k(\boldsymbol{x}' \,|\, \boldsymbol{x})\, \pi(\boldsymbol{x} \,|\, \beta_k)\,\mathrm{d}\boldsymbol{x} \qquad \forall \boldsymbol{x}' \in X. \tag{5.6}$$

From our discussion of Markov transition operators in Chapter 2, we know that an equivalent condition to (5.6) is that there exists a reverse transition operator $\overleftarrow{\mathsf{T}}_k$ with transition density $\overleftarrow{\mathsf{t}}_k$ that respects the generalised balance condition

$$\vec{\mathsf{t}}_k(\boldsymbol{x}' \,|\, \boldsymbol{x})\, \pi(\boldsymbol{x} \,|\, \beta_k) = \overleftarrow{\mathsf{t}}_k(\boldsymbol{x} \,|\, \boldsymbol{x}')\, \pi(\boldsymbol{x}' \,|\, \beta_k) \quad \forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X. \tag{5.7}$$

AIS draws an initial state $\boldsymbol{x}_0$ from the base distribution Q and sequentially applies the transition operators to generate a sequence of states, for each $k \in \{1 \ldots K-1\}$ generating a state $\boldsymbol{x}_k$ by applying the transition operator $\vec{\mathsf{T}}_k$ to the previous state $\boldsymbol{x}_{k-1}$, i.e. $\boldsymbol{x}_k \sim \vec{\mathsf{T}}(\cdot \,|\, \boldsymbol{x}_{k-1})$. The

sequence of K generated states $\{\boldsymbol{x}_k\}_{k=0}^{K-1}$ forms a sample from a joint distribution with normalised density

$$\bar{q}(\boldsymbol{x}_0,\, \boldsymbol{x}_1,\, \ldots\, , \boldsymbol{x}_{K-1}) = q(\boldsymbol{x}_0) \prod_{k=1}^{K-1} \vec{\mathrm{t}}_k(\boldsymbol{x}_k \,|\, \boldsymbol{x}_{k-1}). \tag{5.8}$$

We could also consider an analogous process which draws an initial state $\boldsymbol{x}_K$ from the target distribution P and then sequentially applies the reverse transition operators, for each $k \in \{K-1 \ldots 1\}$ generating a state $\boldsymbol{x}_{k-1}$ by applying the reverse transition operator $\overleftarrow{\mathrm{T}}_k$ to the state $\boldsymbol{x}_k$, i.e. $\boldsymbol{x}_{k-1} \sim \overleftarrow{\mathrm{T}}(\cdot \,|\, \boldsymbol{x}_k)$. The sequence of states generated by this reverse process would have a distribution with unnormalised density

$$\bar{p}(\boldsymbol{x}_0,\, \boldsymbol{x}_1,\, \ldots\, , \boldsymbol{x}_K) = \tilde{p}(\boldsymbol{x}_K) \prod_{k=1}^{K} \overleftarrow{\mathrm{t}}_k(\boldsymbol{x}_{k-1} \,|\, \boldsymbol{x}_k), \tag{5.9}$$

and a normalising constant equal to Z, i.e. equal that of the target distribution. This reverse sequence will be in general intractable to generate as we cannot draw independent samples from the target distribution. However if we consider a state sequence $\{\boldsymbol{x}_k\}_{k=0}^{K-1}$ generated using the forward process as a sample from an importance sampling proposal distribution with density $\bar{q}$ for the distribution defined by the reverse process with density $\bar{p}$, we have that the corresponding importance weight formed by the ratio of these two densities is

$$\begin{aligned}
\vec{w}(\boldsymbol{x}_0,\, \boldsymbol{x}_1,\, \ldots\, , \boldsymbol{x}_{K-1}) &= \frac{\bar{p}(\boldsymbol{x}_0,\, \boldsymbol{x}_1,\, \ldots\, , \boldsymbol{x}_{K-1})}{\bar{q}(\boldsymbol{x}_0,\, \boldsymbol{x}_1,\, \ldots\, , \boldsymbol{x}_{K-1})} && (5.10)\\
&= \frac{\tilde{p}(\boldsymbol{x}_{K-1})}{q(\boldsymbol{x}_0)} \prod_{k=1}^{K-1} \frac{\overleftarrow{\mathrm{t}}_k(\boldsymbol{x}_{k-1} \,|\, \boldsymbol{x}_k)}{\vec{\mathrm{t}}_k(\boldsymbol{x}_k \,|\, \boldsymbol{x}_{k-1})} && (5.11)\\
&= \frac{\tilde{p}(\boldsymbol{x}_{K-1})}{q(\boldsymbol{x}_0)} \prod_{k=1}^{K-1} \frac{\pi(\boldsymbol{x}_{k-1} \,|\, \beta_k)}{\pi(\boldsymbol{x}_k \,|\, \beta_k)} && (5.12)
\end{aligned}$$

with the last line resulting from the condition that (5.7) holds.

We can evaluate the value of this importance weight as it only involves functions we can compute (with the partition function terms $z(\beta_k)$ in the product of ratios cancelling). In particular using the definition (5.4) in the log domain it can be shown to be equal to

$$\log \vec{w}(\boldsymbol{x}_0,\, \boldsymbol{x}_1,\, \ldots\, , \boldsymbol{x}_{K-1}) = \sum_{k=0}^{K-1} (\beta_{k+1} - \beta_k)(\psi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_k)). \tag{5.13}$$

This can be sequentially computed during the forward generation process by initialising $\ell_0 = (\beta_1 - \beta_0)(\psi(\boldsymbol{x}_0) - \phi(\boldsymbol{x}_0))$ and then for each k calculating $\ell_k = \ell_{k-1} + (\beta_{k+1} - \beta_k)(\psi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_k))$. The overall AIS algorithm, including this sequential computation of the log importance weight, is described in Algorithm 13.

By the same argument as given in the discussion of importance sampling in Chapter 2, the importance weight is an unbiased estimate of the normalising constant of $\bar{p}$, i.e. Z. Further as $\boldsymbol{x}_{K-1}$ is marginally distributed according to P under the distribution of the reverse process, the final state $\boldsymbol{x}_{K-1}$ generated from the proposal distribution in the forward process can be used as an importance sample for P. By performing multiple independent runs of the AIS algorithm, we can generate a set of importance samples and weights which we can both use to estimate expectations with respect to the target distribution P and compute unbiased estimates of the normalising constant Z.

We previously argued that importance sampling typically scales poorly to high-dimensional target distributions as mismatch between the proposal and target distributions means that few samples fall in to the target distribution typical set, leading to importance weights which vary widely in magnitude and high variance estimates of both expectations with respect to the target distribution and its normalising constant. The AIS algorithm helps overcome these issues by using Markov transition operators to define a proposal distribution which, providing the number of inverse temperatures is high enough and the transition operators mix well, should be a good match to the target.

In the case of $K = 1$ where we have just $\beta_0 = 0$ and $\beta_1 = 1$, the AIS algorithm reverts to the standard importance algorithm described in Chapter 2 using Q as the proposal distribution. For $K > 1$, the Markov transition operators will perturb the initial samples from the base distribution towards regions of high probability under the target distribution. As the number of inverse temperatures is increased, the difference between invariant distributions of the transition operators corresponding to successive inverse temperatures will become smaller. If the Markov transition operators explore the space well, then for small changes in the inverse temperature the transition operators should be able to keep the state close to equilibrium for the invariant distribution for the current inverse temperature. In general as $K \to \infty$ the final importance sample will have a distribution increasingly close to the

target distribution and the importance weights will give increasingly low variance estimates of Z.

If we were able to generate samples from the reverse process described to motivate the definition of the AIS extended target distribution, then we could use this sampled state sequence to compute importance sampling estimates with respect to the distribution defined by the forward process with density $\bar{p}$. The resulting inverted importance weight

$$\overleftarrow{w}(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{K-1}) = \frac{\bar{q}(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{K-1})}{\bar{p}(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{K-1})} \tag{5.14}$$

would then be an unbiased estimate of Z^{-1}. The *bidirectional Monte Carlo* (BDMC) algorithm of Grosse, Ghahramani and Adams [118] uses this idea to define a scheme for upper and lower bounding $\log Z$. Due to Jensen's inequality an unbiased estimate of Z is a stochastic lower bound on $\log Z$ and an unbiased estimate of Z^{-1} is a stochastic upper bound on $\log Z$, with the bounds becoming tighter on average as the number of inverse temperatures K used in the AIS runs is increased and so the variance of the estimators decreased. If we run both standard 'forward' AIS runs from base to target distributions, and also 'reverse' AIS runs from target to base distributions we can therefore stochastically upper and lower bound $\log Z$, and by using long AIS runs with large K we can make these bounds increasingly tight.

The key computational issue with this scheme is that generally computing an exact sample from the target distribution, as required to initialise a reverse AIS run, will be infeasible. In [118] it is observed however, that for directed generative models with observed variables $\mathbf{y}$ and latent variables $\mathbf{x}$, if a synthetic data set $\boldsymbol{y}$ is generated from $\mathrm{P}_{\mathbf{y}|\mathbf{x}}$ conditioned on latent variables $\boldsymbol{z}$ sampled from the prior distribution $\mathrm{P}_{\mathbf{x}}$, then as $(\boldsymbol{x}, \boldsymbol{y})$ is an exact sample from the joint distribution $\mathrm{P}_{\mathbf{x},\mathbf{y}}$, then $\boldsymbol{x}$ will be an exact sample from the posterior distribution $\mathrm{P}_{\mathbf{x}|\mathbf{y}}$ given the synthetic data $\boldsymbol{y}$. If we perform Bayesian inference experiments with synthetic data we can therefore use this approach to estimate upper and lower bounds on the model evidence term $\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})$. We use this BDMC approach in one of the later experiments in Section 5.6 to bound the value of the model evidence term for a synthetic image dataset under a generative image model, using these bounds a substitute for the intractable ground truth value to assess the quality of the estimates computed using our proposed approach and the other methods we test.

5.3 CONTINUOUS TEMPERATURE APPROACHES

In both AIS and simulated tempering the choices of the number and spacing of the discrete inverse temperature values are key to getting the methods to perform well in complex high dimensional distributions [23, 30, 187]. To get reasonable performance it may be necessary to do preliminary pilot runs to help identify the number of inverse temperatures to use, adding to the computational and user effort burdens of using these methods. It is natural therefore to consider whether it is possible to use a continuously varying inverse temperature variable.

Path sampling [97] proposes this approach, specifically in the context of estimation of normalising constants Z. A *path* is defined as a function parametrised by a variable $\beta \in [0, 1]$ which continuously maps between the target density $\exp(-\phi(\boldsymbol{x}))/Z$ and a base density $\exp(-\psi(\boldsymbol{x}))$, with the geometric bridge in (5.4) a particular example, and the one that we concentrate on here. The main proposal in [97] of direct relevance here is the suggestion to construct a Markov chain which leaves invariant a joint distribution on a state $(\mathbf{x}, \beta)$ with density

$$\mathrm{p}_{\mathbf{x},\beta}(\boldsymbol{x}, \beta) \propto \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x}))\,\rho(\beta). \tag{5.15}$$

with $\rho(\beta)$ a prior density on the inverse temperature variable analogous to the weights $\{w_n\}_{n=0}^N$ in simulated tempering. The authors of [97] suggest that the 'most natural approach' to do this is to use Metropolis or Gibbs transition operator to alternate updates of $\beta \,|\, \mathbf{x}$ and $\mathbf{x} \,|\, \beta$ as an obvious analogue to simulated tempering. In the numerical experiments a random-walk Metropolis scheme is used to perform the updates to β given $\mathbf{x}$.

The authors of [97] suggest a method to use the samples of a chain with an invariant distribution with density (5.15) to estimate $\log Z$ by utilising the identity

$$\log Z = \int_0^1 \int_X \frac{\mathrm{p}_{\mathbf{x},\beta}(\boldsymbol{x}, \beta)}{\mathrm{p}_\beta(\beta)} \left(\psi(\boldsymbol{x}) - \phi(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}\, \mathrm{d}\beta. \tag{5.16}$$

A hybrid numerical integration and Monte Carlo scheme that applies the trapezium rule on a grid specified by the sampled β values is suggested to approximate this integral. As the focus is normalising constant estimation there is no discussion of how to use the sampled $\mathbf{x}$ states to estimate expectations with respect to the target distribution.

Adiabatic Monte Carlo [30] also proposes using a continuously varying inverse temperature variable, here specifically in the context of HMC. The original Hamiltonian system $(\mathbf{x}, \mathbf{p})$ is further augmented with a continuous inverse temperature coordinate $\beta \in [0, 1]$.

A *contact Hamiltonian* is defined on the augmented system,

$$h_c(\boldsymbol{x}, \boldsymbol{p}, \beta) = \beta\phi(\boldsymbol{x}) + (1-\beta)\psi(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p} + \log z(\beta) + h_0 \quad (5.17)$$

this defining a corresponding contact Hamiltonian dynamic,

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial h_c}{\partial \boldsymbol{p}}^{\mathsf{T}}, \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = \frac{\partial h_c}{\partial \beta}\boldsymbol{p} - \frac{\partial h_c}{\partial \boldsymbol{x}}^{\mathsf{T}}, \frac{\mathrm{d}\beta}{\mathrm{d}t} = h_c - \frac{\partial h_c}{\partial \boldsymbol{p}}\boldsymbol{p}. \quad (5.18)$$

The flow map of this dynamic restricted to the zero level-set of the contact Hamiltonian (which the initial state can always be arranged to lie in by appropriately choosing the arbitrary constant h_0) generates trajectories which exactly conserve the contact Hamiltonian and extended state space volume element, and correspond to the thermodynamical concept of a reversible adiabatic process.

For a quadratic kinetic energy, $\frac{\mathrm{d}\beta}{\mathrm{d}t}$ is always non-positive and so forward simulation of the contact Hamiltonian flow generates non-increasing trajectories in β (and backwards simulation generates non-decreasing trajectories in β). In the ideal case this allows the inverse temperature range $[0, 1]$ to be coherently traversed without the random-walk exploration inherent to simulated tempering.

Simulating the contact Hamiltonian flow is non-trivial in practice however: the contact Hamiltonian (5.17) depends on the log partition function $\log z(\beta)$, the partial derivatives of which require computing expectations with respect to $\pi(\boldsymbol{x} \mid \beta)$ which for most problems is intractable to do exactly. Moreover the contact flow can encounter meta-stabilities whereby $\frac{\mathrm{d}\beta}{\mathrm{d}t}$ becomes zero and the flow halts at an intermediate β meaning the flow no longer defines a bijection between $\beta = 0$ and $\beta = 1$. This can be ameliorated by regular resampling of the momenta at a cost of increasing the random-walk behaviour of the dynamic.

An alternative *extended Hamiltonian approach* for simulating a system with a continuously varying inverse temperature was proposed in the statistical physics literature [108]. The inverse temperature of the system is indirectly set via an auxiliary variable, which we will term the

temperature control variable $\mathsf{u} \in \mathbb{R}$. This control variable is mapped to an interval $[s, 1]$, $0 < s < 1$ via a smooth piecewise defined function $\beta : \mathbb{R} \to [s, 1]$, with the conditions that for a pair of thresholds (θ_1, θ_2) with $0 < \theta_1 < \theta_2$, $\beta(u) = 1 \; \forall \; |u| \leq \theta_1$, $\beta(u) = s \; \forall \; |u| \geq \theta_2$ and $s < \beta(u) < 1 \; \forall \; \theta_1 < |u| < \theta_2$.

Unlike Adiabatic Monte Carlo, an additional momentum variable v corresponding to u is also introduced. Although seemingly a minor difference this simplifies the implementation of the approach significantly as the system retains a symplectic structure and can continue to be viewed within the usual Hamiltonian dynamics framework. An *extended Hamiltonian* is defined on the augmented system

$$\tilde{h}(\boldsymbol{x}, u, \boldsymbol{p}, v) = \beta(u)\phi(\boldsymbol{x}) + \omega(u) + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p} + \frac{v^2}{2m} \tag{5.19}$$

where ω is a 'confining potential' on u and m is the mass (marginal variance) associated with v. This extended Hamiltonian is separable with respect to the extended configuration $(\mathbf{x}, \mathsf{u})$ and extended momentum $(\mathbf{p}, \mathsf{v})$ and so can be efficiently simulated using a standard leapfrog integrator. In [108] the extended Hamiltonian dynamics are integrated using a Langevin scheme without Metropolis adjustment and shown to improve mixing in several molecular dynamics problems.

Due to the condition $\beta(u) = 1 \; \forall \; |u| < \theta_1$, the set of sampled configuration states $\mathbf{x}$ which have associated $|\mathsf{u}| < \theta_1$ will (assuming the dynamic is ergodic and Metropolis adjustment were used) asymptotically converge in distribution to the target, and so can be used to estimate expectations without any importance re-weighting. The β function is required to be bounded below by a $s > 0$ in [108] due to the base density being bridged to being an improper uniform density on X. The partition function $z(\beta) \to \infty$ as $\beta \to 0$ in this case, which would imply an infinite density for regions in the extended state space where $\beta(\mathsf{u}) = 0$. Even with a non-zero lower bound on β, the large variations in $z(\beta(\mathsf{u}))$ across different u values can lead to the dynamic poorly exploring the u dimension. In [108] this issue is tackled by introducing an adaptive history-dependent biasing potential on u to try to achieve a flat density across a bounded interval $|\mathsf{u}| < \theta_2$, using for example metadynamics [146]. The resulting non-Markovian updates bias the invariant distribution of the target state however this can be accounted for either by a re-weighting scheme [46], or using a vanishing adaptation.

Algorithm 14 Gibbs continuous tempering.

Input: $(\boldsymbol{x}_n, \beta_n)$: current target variables – inverse temperature state pair, ϕ : energy function of target distribution, ψ : energy function of base distribution, ζ : target distribution normalising constant estimate, T : transition operator updating only target variables $\mathbf{x}$ and leaving distribution with density in (5.20) invariant.

Output: $(\boldsymbol{x}_{n+1}, \beta_{n+1})$: new target variables – inverse temperature state pair, $(w_{0,n+1}, w_{1,n+1})$: new importance weight pair.

$\boldsymbol{x}_{n+1} \sim \mathsf{T}(\cdot \,|\, \boldsymbol{x}_n, \beta_n)$ ▷ Update $\mathbf{x}$ given current β.
$\Delta \leftarrow \phi(\boldsymbol{x}) + \log\zeta - \psi(\boldsymbol{x})$ ▷ Calculate energy difference at new state.
$u \sim \mathcal{U}(|0,1)$ ▷ Independently sampled new $\beta \,|\, \mathbf{x}$.
if $\Delta < 0$ **then**
    $\beta_{n+1} \leftarrow 1 + \frac{\log(1-u+\exp(-|\Delta|)u)}{|\Delta|}$
else if $\Delta = 0$ **then**
    $\beta_{n+1} \leftarrow u$
else
    $\beta_{n+1} \leftarrow -\frac{\log(1-u+\exp(-|\Delta|)u)}{|\Delta|}$
$w_{0,n+1} \leftarrow \frac{-\Delta}{\exp(-\Delta)-1}$ ▷ Calculate target distribution importance weight.
$w_{1,n+1} \leftarrow \frac{\Delta}{\exp(\Delta)-1}$ ▷ Calculate base distribution importance weight.
return $(\boldsymbol{x}_{n+1}, \beta_{n+1}), (w_{0,n+1}, w_{1,n+1})$

5.4 CONTINUOUS TEMPERING

We now describe our proposed continuous tempering approach. The two methods we suggest combine aspects of several of the existing algorithms we have reviewed, in particular we utilising ideas from *Rao–Blackwellised tempered sampling* [53] described in Section 2.3.3 in Chapter 2, *path sampling* [97] and the *extended Hamiltonian* approach of [108]. One of the key motivations of our approaches is to minimise the number of free parameters requiring tuning by the user as far as possible.

We define a joint density on an augmented state consisting of the target variable $\mathbf{x} \in X$ and an inverse temperature variable $\beta \in [0,1]$

$$\mathrm{p}_{\mathbf{x},\beta}(\boldsymbol{x}, \beta) = \frac{1}{C}\exp(-\beta\phi(\boldsymbol{x}) - \beta\log\zeta - (1-\beta)\psi(\boldsymbol{x})). \tag{5.20}$$

As previously ϕ and ψ are the energy functions of the target and base distributions. The term ζ sets a simple prior on the inverse temperature variable and is intended to help balance the marginal densities $\mathrm{p}_\beta(0)$ and $\mathrm{p}_\beta(1)$. It should ideally be set as close to Z as possible - we will motivate this in the next section.

Importantly the conditional density on β given $\mathbf{x}$ corresponding to this joint density has a tractable normalised form

$$p_{\beta|\mathbf{x}}(\beta \mid \boldsymbol{x}) = \frac{\exp(-\beta\Delta(\boldsymbol{x}))\Delta(\boldsymbol{x})}{1-\exp(-\Delta(\boldsymbol{x}))}, \tag{5.21}$$

with $\Delta(\boldsymbol{x}) = \phi(\boldsymbol{x}) + \log\zeta - \psi(\boldsymbol{x})$. For $\Delta(\boldsymbol{x}) > 0$ this corresponds to an exponential distribution with rate parameter $\Delta(\boldsymbol{x})$ truncated to $[0, 1]$. We can efficiently generate independent samples from this distribution and so an obvious method of constructing a Markov chain on the joint space is to alternate Gibbs sampling steps where new values for β are independently sampled from the conditional distribution $P_{\beta|\mathbf{x}}$ with updates to the target variables $\mathbf{x}$ with the inverse temperature β fixed, for example using a Hamiltonian Monte Carlo or slice sampling transition operator. We term this approach *Gibbs continuous tempering* and describe the method, including an approach for generating independent samples from the conditional distribution $P_{\beta|\mathbf{x}}$ in Algorithm 14. We will discuss the details and motivation for the additional values calculated by the algorithm shortly.

Instead of alternating separate updates of the inverse temperature β and target variables $\mathbf{x}$, with a continuous inverse temperature it is natural to consider jointly updating $\mathbf{x}$ and β. In general it is easier to define transition operators on densities with unbounded support so we first define a reparameterisation of the joint target density (5.20) using a *temperature control variable* $u \in \mathbb{R}$ related to β by a standard logistic sigmoid transformation

$$\beta = \frac{1}{1+\exp(-u)} = \sigma(u) \iff u = \log\frac{\beta}{1-\beta}. \tag{5.22}$$

Under this bijective transformation we can use the change of variables formula (1.22) to define the joint density on u and $\mathbf{x}$ as

$$p_{\mathbf{x},u}(\boldsymbol{x}, u) = \frac{\sigma(u)(1-\sigma(u))}{C}\exp(-\sigma(u)(\phi(\boldsymbol{x}) + \log\zeta) - (1-\sigma(u))\psi(\boldsymbol{x})). \tag{5.23}$$

If the energy functions of the target and base distributions are differentiable, then this target density function is also differentiable with respect to all inputs and has unbounded support on $\mathbb{R}^{D+1}$. In this case an obvious choice of transition operator to use is a HMC approach.

Algorithm 15 Continuously tempered Hamiltonian Monte Carlo.

Input: $(\boldsymbol{x}_n, u_n)$: current target variables – temperature control state pair, ϕ : energy function of target distribution, ψ : energy function of base distribution, ζ : target distribution normalising constant estimate, $\mathrm{T}_{\mathrm{HMC}}$: HMC transition operator as described in Algorithm 6 using Hamiltonian defined in (5.24).

Output: $(\boldsymbol{x}_{n+1}, u_{n+1})$: new target variables – temperature control state pair, $(w_{0,n+1}, w_{1,n+1})$: new importance weight pair.

1: $\boldsymbol{x}_{n+1}, u_{n+1} \sim \mathrm{T}_{\mathrm{HMC}}(\cdot \mid \boldsymbol{x}_n, u_n)$ ▷ Jointly update $(\mathbf{x}, \beta)$ using HMC.
2: $\Delta \leftarrow \phi(\boldsymbol{x}) + \log \zeta - \psi(\boldsymbol{x})$ ▷ Calculate energy difference at new state.
3: $w_{0,n+1} \leftarrow \frac{-\Delta}{\exp(-\Delta)-1}$ ▷ Calculate target distribution importance weight.
4: $w_{1,n+1} \leftarrow \frac{\Delta}{\exp(\Delta)-1}$ ▷ Calculate base distribution importance weight.
5: **return** $(\boldsymbol{x}_{n+1}, u_{n+1}), (w_{0,n+1}, w_{1,n+1})$

A Hamiltonian for the extended state $(\mathbf{x}, \mathrm{u})$ is constructed by augmenting with associated momenta variables $(\mathbf{p}, \mathrm{v})$ and then defining

$$\begin{aligned}\tilde{h}(\boldsymbol{x}, u, \boldsymbol{p}, v) =& \sigma(u)(\phi(\boldsymbol{x}) + \log \zeta) + (1 - \sigma(u))\psi(\boldsymbol{x}) \\ & - \log \sigma(u) - \log(1 - \sigma(u)) + \frac{1}{2}\boldsymbol{p}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{p} + \frac{v^2}{2m}. \end{aligned} \tag{5.24}$$

As with the approach of [108], this Hamiltonian is separable and the corresponding dynamic can be efficiently simulated with a leapfrog integrator. The reversible and volume-preserving simulated dynamic can then be used as a proposal generating mechanism on the joint space $(\mathbf{x}, \mathrm{u}, \mathbf{p}, \mathrm{v})$ for a Metropolis–Hastings step as in standard HMC transition described in Algorithm 6. We term the approach of running HMC in the extended joint space *continuously tempered Hamiltonian Monte Carlo*. An outline of the method is given in Algorithm 15; as with the previous Gibbs continuous tempering algorithm this contains steps for calculation of additional values which we will motivate shortly.

Importantly we can also equally apply more efficient adaptive HMC variants such as the NUTS algorithm [130] which dynamically set the number of integration steps. In particular we can easily define models with the augmented density (5.23) in probabilistic programming frameworks such as Stan and PyMC3 and exploit their efficient in-built NUTS based inference algorithms and automatic differentiation capabilities.

We have so far neglected to discuss the key issue of how to use the sampled continuous tempering chain states to compute the expectations with respect to the target distribution P of interest and its normalising constant Z. As described in Chapter 2, in simulated tempering,

typically expectations are estimated by computing averages over only the chain states for which $\mathsf{k} = K$ corresponding to $\beta_K = 1$, as at convergence these states will be distributed according to the target distribution. Although simple to implement, we previously suggested that this scheme is somewhat wasteful as it means only a small proportion of generated states are used to compute estimates and this issue becomes worse as the number of inverse temperatures is increased as fewer and fewer states will corresponding to $\mathsf{k} = K$.

As the inverse temperature β (or equivalently temperature control variable u) is continuous in our case, there seems to potentially be even more of a problem as the probability of chains generating states with a corresponding β exactly equal to one will be zero: the regions of the state space for which $\beta = 1$ is zero-measure under the joint distribution $\mathsf{P}_{\mathbf{x},\beta}$. We could form an approximate scheme by using states with β values within some small tolerance of one (analogous to an ABC approach); this would however introduce bias into the results which it would be hard to characterise, and further would require tuning the choice of tolerance. Further this method would still mean only a small subset of the chains states are utilised to estimate expectations.

In the *extended Hamiltonian* approach of [108], this issue is overcome by using a piecewise defined inverse temperature control function $\beta(u)$ which maps an *interval* $[-\theta_1, \theta_1]$ of u values to a corresponding inverse temperature value $\beta(\mathsf{u}) = 1$. As the region of space with $\mathsf{u} \in [-\theta_1, \theta_1]$ has a non-zero measure this means a non-zero number of sampled states will at convergence correspond to samples from the target distribution. Although an interesting approach, this method requires tuning the choice of the width interval θ_1 of u values mapping to $\beta(\mathsf{u}) = 1$. Additionally when the chain is at states with $|\mathsf{u}| \leq \theta_1$ and so the corresponding inverse temperature is $\beta(\mathsf{u}) = 1$, the chain is effectively exploring the original target distribution, which if multimodal will mean in these periods the chain will typically mix poorly. There is therefore a tension between the requirement to keep θ_1 small such that the chain maximises the time spent at low and intermediate inverse temperatures to allow for improved mixing, and the need for θ_1 to not made too small so that a reasonable number of sampled chain states can be used to compute expectations with respect to the target distribution. However this tradeoff is balanced in practice, the basic scheme described

will also mean that only a subset of the chain states will be used in computing expectations.

In Chapter 2 we discussed simulated tempering in the broader context of auxiliary variable methods where the target distribution corresponds to a conditional distribution given a value or set of values of the auxiliary variables. We made the observation there that the standard estimator used to compute expectation in these methods was closely related to rejection sampling. We at that point suggested that alternative approaches could be considered which are instead based on importance sampling. We use that intuition here to suggest an estimator for expectations with respect to the target distribution using continuous tempering chains. The key idea is to use the marginal distribution $\mathrm{P}_{\mathbf{x}}$ on the target variables under the joint density (5.20) as the proposal distribution for an importance sampling estimator for expectations with respect to the target distribution. As the generated target variable states of continuous tempering chains will be, at convergence, distributed according to $\mathrm{P}_{\mathbf{x}}$ we can use (all) the sampled $\mathbf{x}$ values to estimate expectations with respect to $\mathrm{P}_{\mathbf{x}}$.

The target distribution P is equal to $\mathrm{P}_{\mathbf{x}|\beta}(\cdot \mid 1)$ and so expectations with respect to P correspond to

$$\mathbb{E}\big[f(\mathbf{x}) \mid \beta = 1\big] = \int_X f(\boldsymbol{x})\, \mathrm{p}_{\mathbf{x}|\beta}(\boldsymbol{x} \mid 1)\, \mathrm{d}\boldsymbol{x}. \tag{5.25}$$

We can use an importance sampling approach to re-express this integral with respect to the conditional distribution $\mathrm{P}_{\mathbf{x}|\beta}$ as expectations with respect to the marginal distribution $\mathrm{P}_{\mathbf{x}}$, giving the following

$$\mathbb{E}\big[f(\mathbf{x}) \mid \beta = 1\big] = \frac{\int_X f(\boldsymbol{x}) \frac{\mathrm{p}_{\mathbf{x}|\beta}(\boldsymbol{x} \mid 1)}{\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})}\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}{\int_X \frac{\mathrm{p}_{\mathbf{x}|\beta}(\boldsymbol{x} \mid 1)}{\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})}\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}} \tag{5.26}$$

$$= \frac{\int_X f(\boldsymbol{x})\, \mathrm{p}_{\beta|\mathbf{x}}(1 \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}{\int_X \mathrm{p}_{\beta|\mathbf{x}}(1 \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}. \tag{5.27}$$

As at convergence the $\mathbf{x}$ components of the chain states will be marginally distributed according to $\mathrm{p}_{\mathbf{x}}$ we can therefore construct consistent estimators for (5.27) from the sampled target variable states $\{\boldsymbol{x}_n\}_{n=1}^N$ of a continuous tempering chain, by computing weighted averages

$$\mathbb{E}\big[f(\mathbf{x}) \mid \beta = 1\big] = \lim_{N\to\infty} \frac{\sum_{n=1}^N (w_1(\boldsymbol{x}_n) f(\boldsymbol{x}_n))}{\sum_{n=1}^N (w_1(\boldsymbol{x}_n))}. \tag{5.28}$$

The importance weights take values according to $\mathrm{p}_{\beta|\mathbf{x}}$ (5.21)

$$w_1(\boldsymbol{x}) = \mathrm{p}_{\beta|\mathbf{x}}(1 \,|\, \boldsymbol{x}) = \frac{\Delta(\boldsymbol{x})}{\exp(\Delta(\boldsymbol{x})) - 1}. \tag{5.29}$$

Intuitively this estimator (5.28) seems reasonable, more highly weighting sampled target variables $\boldsymbol{x}$ for which $\mathrm{p}_{\beta|\mathbf{x}}(1 \,|\, \boldsymbol{x})$ is high. Importantly unlike the standard rejection sampling type estimator it does not depend on the actual sampled inverse temperature state values and can be validly computed even if no sampled state pair ever has $\beta = 1$ which will generally be the case for continuous tempering, and uses information from all of the chain target variable states to compute estimates.

By an analogous construction, we can also estimate expectations with respect to the base distribution Q, corresponding to $\mathrm{P}_{\mathbf{x}|\beta}(\cdot|0)$ using

$$\mathbb{E}\big[f(\mathbf{x}) \,|\, \beta = 0\big] = \lim_{N\to\infty} \frac{\sum_{n=1}^{N}(w_0(\boldsymbol{x}_n) f(\boldsymbol{x}_n))}{\sum_{n=1}^{N}(w_0(\boldsymbol{x}_n))} \tag{5.30}$$

with the corresponding importance weights defined by

$$w_0(\boldsymbol{x}) = \mathrm{p}_{\beta|\mathbf{x}}(0 \,|\, \boldsymbol{x}) = \frac{\Delta(\boldsymbol{x})}{1 - \exp(-\Delta(\boldsymbol{x}))}. \tag{5.31}$$

Typically the base distribution will have known moments (e.g. mean and covariance of a Gaussian base distribution) which can be compared to the estimates calculated using this estimator (5.30) to check for convergence problems. Convergence of the estimates to the true moments is not a sufficient condition for convergence of the chain to the distribution defined by (5.20) but is necessary.

Although we can calculate the importance weights (5.29) and (5.31) in a post-processing step after running the chain, the energy difference terms $\Delta(\boldsymbol{x})$ required to compute them will typically already be computed as part of the chain state updates - for example $\Delta(\boldsymbol{x})$ is computed to independently generate a new β value from $\mathrm{P}_{\beta|\mathbf{x}}$ in Gibbs continuous tempering, and for continuously tempered HMC the final Metropolis accept step will require calculating the Hamiltonian (5.24) at the new state which involves evaluating $\Delta(\boldsymbol{x})$. It will therefore generally be more efficient to compute these weights as part of each update; we show such a scheme in Algorithms 14 and 15.

We can also use the importance weights to construct an estimator for the normalising constant Z of the target density. The inverse temperature marginal density p_β corresponding to (5.20) is

$$\mathrm{p}_\beta(\beta) = \frac{1}{C\zeta^\beta}\int_X \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x} = \frac{z(\beta)}{C\zeta^\beta}. \tag{5.32}$$

We therefore have that

$$\mathrm{p}_\beta(0) = \frac{1}{C},\ \mathrm{p}_\beta(1) = \frac{Z}{C\zeta} \implies Z = \frac{\mathrm{p}_\beta(1)}{\mathrm{p}_\beta(0)}\zeta. \tag{5.33}$$

We can use (5.21) to write the marginal density p_β

$$\mathrm{p}_\beta(\beta) = \int_X \mathrm{p}_{\beta|\mathbf{x}}(\beta \mid \boldsymbol{x})\,\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}\left[\mathrm{p}_{\beta|\mathbf{x}}(\beta \mid \mathbf{x})\right] \tag{5.34}$$

and so the marginal densities of $\beta = 0$ and $\beta = 1$ can be written

$$\mathrm{p}_\beta(0) = \mathbb{E}[w_0(\mathbf{x})] \quad \text{and} \quad \mathrm{p}_\beta(1) = \mathbb{E}[w_1(\mathbf{x})]. \tag{5.35}$$

Therefore given target variable samples $\{\boldsymbol{x}_n\}_{n=1}^N$ from a continuous tempering chain we have the following consistent estimator for Z

$$Z = \frac{\mathbb{E}[w_1(\mathbf{x})]}{\mathbb{E}[w_0(\mathbf{x})]}\zeta = \lim_{N\to\infty}\frac{\sum_{n=1}^N (w_1(\boldsymbol{x}_n))}{\sum_{n=1}^N (w_0(\boldsymbol{x}_n))}\zeta, \tag{5.36}$$

This can be seen to be a continuous analogue of the *Rao-Blackwellised* estimator (2.74) proposed in [53] and discussed in Section 2.3.3.

5.5 CHOOSING A BASE DISTRIBUTION

We now consider criteria for selecting an appropriate base distribution Q to use with tempering methods. We claimed in the introduction that the choice of base distribution was key to the performance of tempering methods in complex high-dimensional targets, with the degree of match between the base and target determining the ability of the chain to mix between low and high inverse temperatures. In this section we formalise this statement by deriving some basic properties of the logarithm of the inverse temperature marginal density $\log \mathrm{p}_\beta$ and its relation to the *Kullback–Leibler* (KL) divergences between Q and P. We will then use these properties to motivate specific computational schemes for fitting a base distribution.

Taking logarithms of (5.32) and recalling the previous definition of the geometric bridge density $\pi(\boldsymbol{x} \mid \beta)$ (5.4)

$$\begin{aligned}\log \mathrm{p}_\beta(\beta) &= \log \int_X \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x} - \log C - \beta \log \zeta \\ &= \log \int_X \pi(\boldsymbol{x} \mid \beta)\,\mathrm{d}\boldsymbol{x} - \log C - \beta \log \zeta. \end{aligned} \tag{5.37}$$

We will first show that $\log \mathrm{p}_\beta$ is convex. Differentiating (5.37) with respect to the inverse temperature and using the earlier definition of the partition function $z(\beta)$ (5.5) gives

$$\begin{aligned}\frac{\partial \log \mathrm{p}_\beta}{\partial \beta} &= \int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) - \log \zeta) \frac{\exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x}))}{z(\beta)}\,\mathrm{d}\boldsymbol{x} \\ &= \int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) - \log \zeta)\pi(\boldsymbol{x} \mid \beta)\,\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{5.38}$$

Differentiating again then gives after some manipulation

$$\begin{aligned}\frac{\partial^2 \log \mathrm{p}_\beta}{\partial \beta^2} = \int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}))^2 \pi(\boldsymbol{x} \mid \beta)\,\mathrm{d}\boldsymbol{x} - \\ \left(\int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}))\pi(\boldsymbol{x} \mid \beta)\,\mathrm{d}\boldsymbol{x}\right)^2. \end{aligned} \tag{5.39}$$

As the square function is convex by Jensen's inequality we have that

$$\left(\int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}))\pi(\boldsymbol{x} \mid \beta)\,\mathrm{d}\boldsymbol{x}\right)^2 \leq \int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}))^2 \pi(\boldsymbol{x} \mid \beta)\,\mathrm{d}\boldsymbol{x}.$$

Therefore $\frac{\partial^2 \log \mathrm{p}_\beta}{\partial \beta^2} \geq 0$ and so $\log \mathrm{p}_\beta$ is convex.

We now show that gradients of $\log \mathrm{p}_\beta$ at the end-points $\beta = 0$ and $\beta = 1$ are related to the KL divergences between the base and target distributions. We first recall the definition of the KL divergence between two probability distributions P and Q on a space X with P absolutely continuous with respect to Q as

$$\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] = \int_X \log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right)\mathrm{d}P, \tag{5.40}$$

which is read as the KL divergence from P to Q. For distributions P and Q on a real-valued space $X = \mathbb{R}^D$ defined by densities p and q with respect to the Lebesgue measure we further have that

$$\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] = \mathbb{D}^{\lambda}_{\mathrm{KL}}[p \,\|\, q] = \int_X p(\boldsymbol{x})\, \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}. \tag{5.41}$$

Considering first the value of $\frac{\partial \log \mathrm{p}_\beta}{\partial \beta}$ at the upper end point $\beta = 1$

$$\begin{aligned}
\frac{\partial \log \mathrm{p}_\beta}{\partial \beta}(1) &= \int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) - \log \zeta) \frac{1}{Z} \exp(-\phi(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{x} && (5.42)\\
&= \int_X (-\log q(\boldsymbol{x}) + \log p(\boldsymbol{x}) + \log Z - \log \zeta) p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} && (5.43)\\
&= \int_X \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} + \log Z - \log \zeta && (5.44)\\
&= \mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] + \log Z - \log \zeta. && (5.45)
\end{aligned}$$

Equivalently for the lower end point $\beta = 0$

$$\begin{aligned}
\frac{\partial \log \mathrm{p}_\beta}{\partial \beta}(0) &= \int_X (\psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) - \log \zeta) \exp(-\psi(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{x} && (5.46)\\
&= \int_X (-\log q(\boldsymbol{x}) + \log p(\boldsymbol{x}) + \log Z - \log \zeta) q(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} && (5.47)\\
&= -\int_X \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} q(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} + \log Z - \log \zeta && (5.48)\\
&= -\mathbb{D}_{\mathrm{KL}}[Q \,\|\, P] + \log Z - \log \zeta. && (5.49)
\end{aligned}$$

The convexity of the log marginal density means that gradients of $\log \mathrm{p}_\beta$ at the end-points give a lot of information about the shape of the log marginal density. For the purposes of improving mixing of tempered chains across the inverse temperature range, we ideally want the marginal density on the inverse temperatures p_β to be as flat as possible. As $\log \mathrm{p}_\beta$ is convex, if we set $\log \zeta = \log Z$ and chose a base distribution Q such that $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] = \mathbb{D}_{\mathrm{KL}}[Q \,\|\, P] = 0$ we would have $\log \mathrm{p}_\beta$ is constant and so the marginal density would be uniform across $[0, 1]$. Of course in reality we do not know $\log Z$ and $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] = \mathbb{D}_{\mathrm{KL}}[Q \,\|\, P] = 0$ would only be satisfied if we used the target distribution as the base distribution, which would negate any benefit from using a tempering approach.

More practically we need to constrain Q to be in a family of distributions amenable to exploration, such that tempered chains gain from an improved ability to move around the space when at low inverse temperatures. Under that constraint, we then ideally want the gradients of the (logarithm of the) marginal density p_β to be as close to zero as possible at $\beta = 0$ and $\beta = 1$ such that the marginal density is as flat as possible. This suggests we want to minimise $\mathbb{D}_{\mathrm{KL}}[Q \,\|\, P]$ or $\mathbb{D}_{\mathrm{KL}}[Q \,\|\, P]$ or some combination of thereof, subject to Q being in a 'simple' family.

This is exactly the task considered by the variational inference methods discussed in Appendix C. The standard variational inference approach involves fitting an approximate distribution Q in a fixed family to a target distribution P by minimising the *evidence lower bound* (ELBO) variational objective $\log Z - \mathbb{D}_{\mathrm{KL}}[Q \,||\, P]$ (so-called as in Bayesian inference problems Z corresponds to the model evidence and $\mathbb{D}_{\mathrm{KL}}[Q \,||\, P] \geq 0$ therefore the ELBO lower bounds the evidence). If we were to fit a base distribution Q in this manner and set $\log \zeta$ to the final value of the (estimated) variational objective, the gradient of the log marginal density at $\beta = 0$ should be close to zero.

In terms of the choice of variational inference approach, in some cases we may be able to use variational methods specifically designed for the target distribution family. More generally methods such as ADVI [144] provide a black-box framework for fitting variational approximations to differentiable target densities. In models such as VAE [139, 224] a parametric variational approximation to the target density of interest (e.g. posterior on latent space) is fitted during training of the original model, in which case it provides a natural choice for the base distribution as observed in [264].

A potential problem with this approach is that the classes of target distribution that we are particularly interested in applying our approach to — those with multiple isolated modes — are precisely the same distributions that simple variational approximations will tend to fit poorly, the divergence $\mathbb{D}_{\mathrm{KL}}[Q \,||\, P]$ being minimised favouring 'mode-seeking' solutions which usually fit only one mode well as illustrated in C.2 in Appendix C. This both limits how small the divergence from the base to target can be made, but also crucially is undesirable as we wish to use the base distribution to move between modes in the target.

One option would be to instead to minimise the reversed KL divergence $\mathbb{D}_{\mathrm{KL}}[P \,||\, Q]$, this tending to produce 'mode-covering' solutions that match global moments. Methods such as *expectation propagation* (EP) [176] do allow moment-matching approximations to be found and EP methods may be a good option in the cases were they are applicable to the distribution of interest . Another possibility would be to use an alternative divergence in a variational framework that favours mode-covering solutions, with methods using the χ-divergence [76] and Rényi divergence [156] having been recently proposed in this context and discussed in Section C.2 in Appendix C.

A further alternative is to fit multiple local variational approximations $\{q_i(\boldsymbol{x})\}_{i=1}^L$ by minimising the ELBO variational objective from multiple random parameter initialisations (discarding any duplicate solutions as measured by some distance tolerance between the variational parameters), each approximating a single mode well. We can then combine these local approximations into a global approximation $q(\boldsymbol{x})$, for example using a mixture model

$$m(\boldsymbol{x}) = \frac{1}{\zeta}\sum_{i=1}^L(\exp(\ell_i)q_i(\boldsymbol{x})), \quad \zeta = \sum_{i=1}^L \exp(\ell_i), \tag{5.50}$$

with ℓ_i the final variational objective value for q_i. If the local approximations had non-overlapping support this would lead to a global approximation which is guaranteed to be at least as close in KL divergence as any of the local approximations and a $\log \zeta$ which is at least as tight a lower bound on $\log Z$ as any of the individual ℓ_i [267]. Often we may wish to use local approximations with overlapping support (e.g. Gaussian) where the guarantee does not apply. However for the cases of target distributions with multiple isolated modes the 'overlap' (regions with high density under multiple q_i) between local Gaussian approximations to each mode will often be minimal and so the method is still a useful heuristic. A mixture distribution is unlikely to itself be a good choice of base distribution however, as it will tend to be multimodal. We can therefore instead use a base distribution with moments matched to the fitted mixture, e.g. $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean and covariance matched to the mean and covariance of the mixture $m(\boldsymbol{x})$.

For complex target distributions typically even with a variational fitting approach, the KL divergences between base and target distributions will remain relatively large. For approaches based on minimising $\mathbb{D}_{\text{KL}}[Q \,\|\, P]$ and setting $\log \zeta$ to the final variational objective, then as $\log \zeta$ will be a lower bound on $\log Z$ as a result of (5.33) in this case we will have $\mathrm{p}_\beta(1) \geq \mathrm{p}_\beta(0)$ (this can also be seen from the convexity of the log marginal density and that the log marginal density gradient at $\beta = 0$ should be approximately zero), with the degree of discrepancy depending on the divergence between the base and target distributions. If the divergence remains large, in this case tempered chains will tend to remain at inverse temperatures close to one, limiting the gain from the temperature augmentation. Equally if an approach is used which gives $\log Z < \log \zeta$ and so $\mathrm{p}_\beta(1) < \mathrm{p}_\beta(0)$ then if the discrepancy is large tempered chains will tend to remain confined to low inverse tem-

peratures, and the variance of the estimator (5.28) will become high as performance will be similar to using the base distribution Q directly in a simple importance sampling estimator (without the benefit of drawing independent samples).

A natural approach to try to ameliorate these effects is to use an iterative method analogous to the approaches often used in *simulated tempering* (ST) to choose the prior weights on the inverse temperatures [53, 103]. An initial Q and $\log \zeta$ are chosen for example using a Gaussian variational approximation to the target distribution as described above. An initial pilot tempered chain is then run. The sampled states from this chain can then be used to both compute an estimate for $\log Z$ using (5.36) and updated estimates of the target density mean and covariance using (5.28). A new Gaussian base distribution Q can then be specified with the updated mean and covariance estimates and $\log \zeta$ chosen to be the updated $\log Z$ estimate. Samples of the new joint density can then be used to update $\log \zeta$ and Q again and so on. Although this adds significantly to the computational and user effort burden, the use of an initial variational approach can potentially reduce the amount of iteration used significantly compared to using a simple non-informative base distribution such as the prior distribution of a posterior target distribution.

A difficulty presented by the use of a continuous inverse temperature formulation is that it is not as trivial as the discrete case to attempt to flatten out the overall marginal distribution on the inverse temperature by using estimates from an initial run of the relative probabilities (densities) of intermediate $\beta \in (0, 1)$ (with the $\log \zeta$ value controlling the relative densities of $\beta = 1$ and $\beta = 0$). While in the discrete case the weights $\{w_n\}_{n=0}^{N}$ can be used to arbitrarily reweight the marginal $P_{\beta=\beta_n}$, in the continuous case a suitably expressive parametric family across the interval $[0, 1]$ would need to be chosen for the prior density. We do not investigate this issue further here but it could form an important line of future developments of the algorithm.

5.6 NUMERICAL EXPERIMENTS

We now move on to discussing the results of three numerical experiments comparing the proposed continuous tempering approaches to simulated tempering and AIS.

5.6.1 Boltzmann machine relaxations

In Chapter 1 we introduced Boltzmann machines as an example of a undirected graphical model which often define highly multimodal distributions on a binary vector state space which it can challenging to sample from. It was shown in [266], that a Boltzmann machine distribution (1.45) with weight parameters $\boldsymbol{W} \in \mathbb{R}^{D_B \times D_B}$ and biases $\boldsymbol{b} \in \mathbb{R}^D_B$ can be relaxed to a closely related distribution with density on a real valued vector state $\mathbf{x} \in \mathbb{R}^D$

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \frac{2^{D_B} \exp\left(-\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}\right)}{(2\pi)^{D/2} Z_B \exp\left(\frac{1}{2}\operatorname{Tr}(\boldsymbol{D})\right)} \prod_{i=1}^{D_B} \left(\cosh\left(\boldsymbol{q}_i^{\mathsf{T}}\boldsymbol{x} + b_i\right)\right), \tag{5.51}$$

where $\left\{\boldsymbol{q}_i^{\mathsf{T}}\right\}_{i=1}^{D_B}$ are the D_B rows of a $D_B \times D$ matrix $\boldsymbol{Q}$ such that $\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}} = \boldsymbol{W} + \boldsymbol{D}$, with $\boldsymbol{D}$ chosen such that $\boldsymbol{W} + \boldsymbol{D}$ is semi positive-definite. The moments of this *Boltzmann machine relaxation distribution* are related to the moments of the corresponding binary state-space Boltzmann distribution by

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{Q}^{\mathsf{T}}\mathbb{E}[\mathbf{s}] \qquad \text{and} \qquad \mathbb{E}\left[\mathbf{x}\mathbf{x}^{\mathsf{T}}\right] = \boldsymbol{Q}^{\mathsf{T}}\mathbb{E}\left[\mathbf{s}\mathbf{s}^{\mathsf{T}}\right]\boldsymbol{Q} + \mathbf{I}. \tag{5.52}$$

Derivations of these relationships and further details of the parameterisation we use are shown in Appendix E.

The relaxation density (5.51) corresponds to a mixture of 2^{D_B} Gaussian components, with frustrated, multimodal Boltzmann machine distributions on the discrete state space corresponding to highly multimodal relaxation densities with large separations between the Gaussian components. Importantly the relationships in (5.52) mean that the moments of a relaxation distributions can be calculated from the moments of the original discrete Boltzmann machine distribution, which for models with a small number of binary units D_B (30 in our experiments) can be computed exactly by exhaustive iteration across the 2^{D_B} discrete states. This allows ground truth moments to be calculated against which convergence can be checked, making the Boltzmann machine relaxation distribution a useful test case for evaluating the ability of the proposed approaches to improve exploration of challenging multimodal distributions as claimed.

As a first experiment we therefore performed inference in relaxations of a set of ten synthetic Boltzmann machine distributions. The paramet-

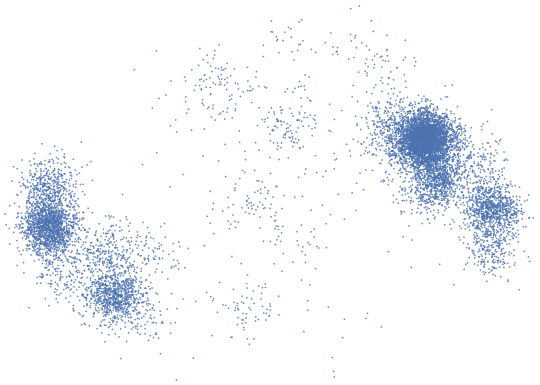


Figure 5.1.: Two-dimensional projection of 10 000 independent samples from a Gaussian mixture relaxation of a Boltzmann machine distribution. The parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ of the Boltzmann machine distribution were generated as described in the main text, with here $D_B = 28$ (rather than $D_B = 30$ as in the experiments) as independent sampling from larger systems exceeded the memory available on the workstation used. The two components shown correspond to the two eigenvectors of the generated basis $\boldsymbol{R}$ with the largest corresponding eigenvalues.

ers of the Boltzmann machine distributions were randomly generated so that the corresponding relaxations are highly multimodal and so challenging to explore well. The weight parameters $\boldsymbol{W}$ were generated using an eigendecomposition based method. A uniformly distributed (with respect to the Haar measure) random orthogonal matrix $\boldsymbol{R}$ was sampled. A vector of eigenvalues $\boldsymbol{e}$ was generated by sampling independent zero-mean unit-variance normal variates $n_i \sim \mathcal{N}(\cdot;\, 0,\, 1)\ \forall i \in \{1, \ldots D_B\}$ and then setting $e_i = s_1 \tanh(s_2 n_i)\ \forall i \in \{1, \ldots D_B\}$, with $s_1 = 6$ and $s_2 = 2$ in the experiments. This generates eigenvalues concentrated near $\pm s_1$ with this empirically observed to lead to systems which tended to be highly multimodal. A symmetric matrix $\boldsymbol{V} = \boldsymbol{R}\,\mathrm{diag}(\boldsymbol{e})\boldsymbol{R}^\mathsf{T}$ was then computed and the weights $\boldsymbol{W}$ set such that $W_{i,j} = V_{i,j}\ \forall i \neq j$ and $W_{i,i} = 0\ \forall i$. The biases $\boldsymbol{b}$ were generated using $b_i \sim \mathcal{N}\left(\cdot;\, 0, 0.1^2\right)\ \forall i$. A 2D projection of samples from a generated distribution illustrating the resulting multimodality is shown in Figure 5.1. Running HMC directly in these target distributions performed very poorly with the chains often getting stuck in one Gaussian component mode over thousands of iterations.

A Gaussian base distribution and approximate normalising constant ζ were fitted to each the 10 relaxation target densities by matching moments to a mixture of variational Gaussian approximations (indi-

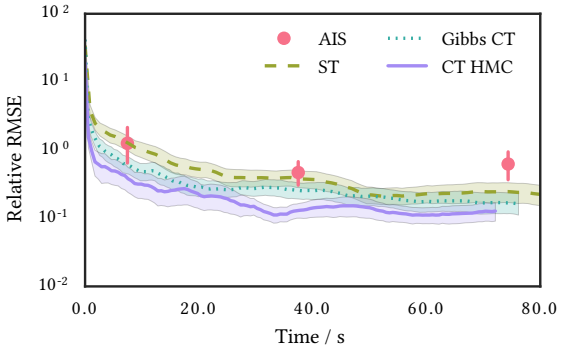
vidually fitted using a mean-field approach based on the underlying Boltzmann machine distribution) as described in Section 5.5. All methods used the same base distribution and ζ was used to set the prior on the inverse temperatures in all methods (equivalent to $w_k = -\beta_k \zeta$ in the earlier description of ST).

For ST, a *Rao Blackwellised* estimate of the normalising constant Z was used as described in [53] and an estimator equivalent to (5.28) used to allow estimation of the moments from all of the samples rather than just those for which $\mathrm{k} = K$ (with this estimator found to always give better results than the standard ST estimator in these experiments). For each of ST, Gibbs *continuous tempering* (CT) and AIS, for the updates to the target variables $\mathbf{x}$ given a fixed inverse temperature (index in the case of ST), a HMC transition operator was used with $\delta t = 0.5$, $L = 20$. For ST the inverse temperature values used a sigmoidal spacing

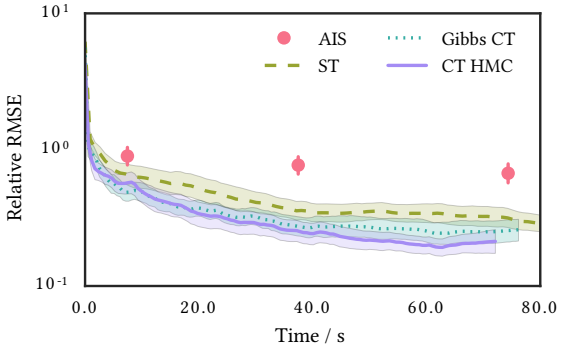
$$\tilde{\beta}_k = \left(1 + \exp\left(-4\frac{2k - K}{K}\right)\right)^{-1} \quad \forall k \in \{0 \dots K\}, \quad \beta_k = \frac{\tilde{\beta}_k - \tilde{\beta}_0}{\tilde{\beta}_K - \tilde{\beta}_0} \tag{5.53}$$

with $K = 1000$ used based on initial pilot runs, and independent multinomial resampling from $\mathrm{P}_{\mathrm{k}|\mathbf{x}}$ for the updates to the temperature index. For AIS separate runs with $K = 1000$, $K = 5000$ and $K = 10000$ inverse temperature β_k values were used to obtain estimates at different run times, using the same sigmoidal spacing (5.53) as for ST and the estimate for each K based on $N = 100$ AIS independent runs to produce N importance samples. For the *continuously tempered Hamiltonian Monte Carlo* (CT-HMC) chains a HMC transition operator with $\delta t = 0.5$ and $L = 20$ was used with the mass value for the temperature control variable set to $m = 1$.

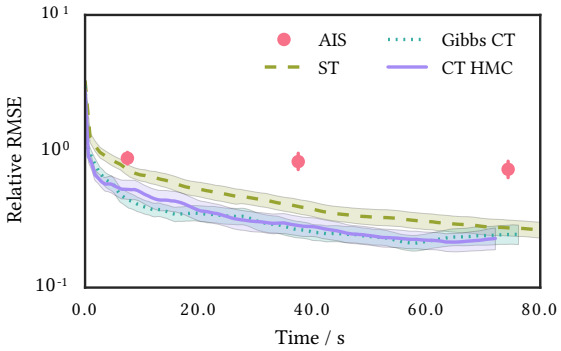
Plots showing the RMSE in estimates of $\log Z$ and the mean and covariance of the relaxation distribution against computational run time for different sampling methods are shown in Figure 5.2. The RMSE values are normalised by the residual errors of the corresponding estimated moments used in the base density (and $\log \zeta$) such that values below unity indicate an improvement in accuracy over the variational approximation. The curves shown are RMSEs averaged over 10 independent chains (or set of runs for AIS) for each of the 10 generated parameter sets, with the filled regions indicating plus or minus three standard errors of the mean. All methods used a shared Theano [248] implementation



(a) $\log Z$



(b) $\mathbb{E}[\mathbf{x}]$



(c) $\mathbb{E}\left[\mathbf{x}\mathbf{x}^{\mathsf{T}}\right] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^{\mathsf{T}}$

Figure 5.2.: RMSEs in empirical moments estimated from MCMC samples against run time for various thermodynamic ensemble MCMC methods run on Gaussian Boltzmann machine relaxation target distributions. All RMSEs are relative to the RMSE of the corresponding approximate moments calculated using the moment-matched variational mixtures, so values below 1 represent improvement on deterministic approximation. For AIS points across time axis represent increasing number of inverse temperatures: $(1,\ 5,\ 10) \times 10^3$. For ST, Gibbs CT and CT-HMC curves show RMSEs for expectations calculated with increasing number of samples from chains. All curves / points show mean across 10 runs for each of 10 generated parameter sets. Filled regions / error bars show ±3 standard errors of mean.

running on a Intel Core i5-2400 quad-core CPU for the HMC updates and so run times are roughly comparable.

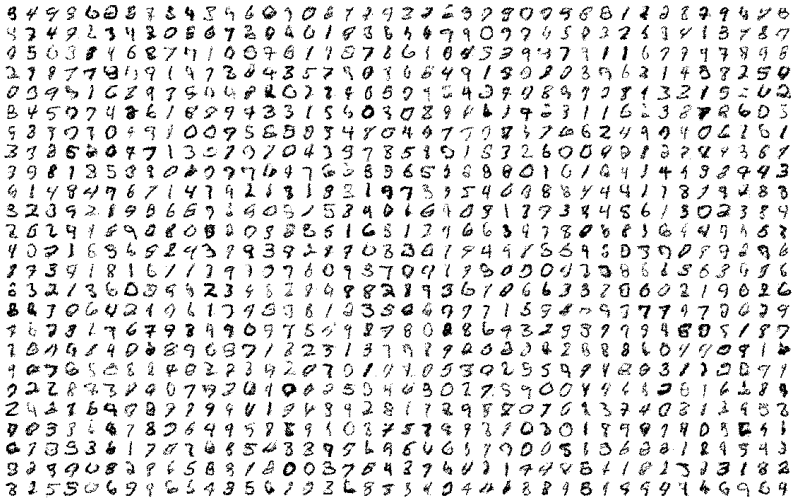
The two CT approaches, Gibbs CT and CT-HMC, both dominate in terms of having lower average RMSE in all three moment estimates across all run times, with CT-HMC showing slightly better performance on estimates of $\log Z$ and $\mathbb{E}[\mathbf{x}]$ than Gibbs CT. The tempering approaches seem to outperform AIS here, possibly as the highly multimodal nature of the target densities favours the ability of tempered dynamics to move the inverse temperature both up and down and so in and out of modes in the target density, unlike AIS where the fixed sequence of temperature updates are more likely to end up with chains confined to a single mode after the initial transitions for low β_k.

5.6.2 Generative image models

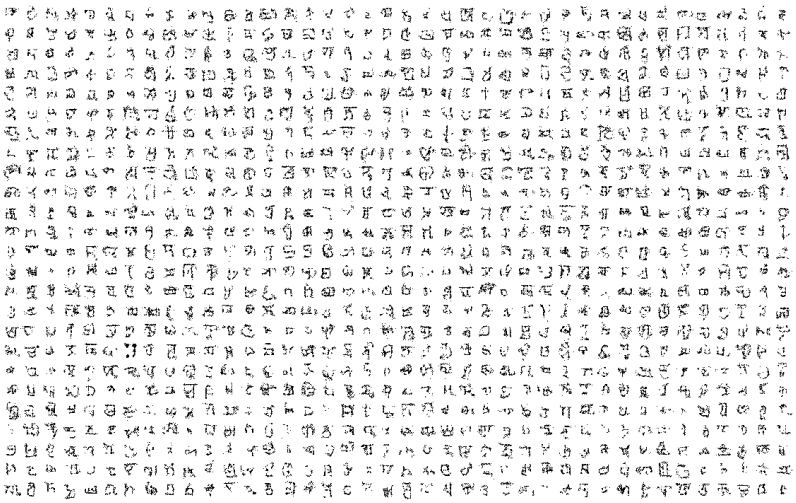
For the next experiments, we compared the efficiency of our CT-HMC and Gibbs CT approaches to ST and AIS for marginal likelihood estimation in decoder-based generative models for images. Use of AIS in this context was recently proposed in [264]. Specifically we estimate the joint marginal likelihood of 1000 generated binary images under the Bernoulli decoder distribution of two *importance weighted autoencoder* (IWAE) [51] models. Each IWAE has one stochastic hidden layer and a 50-dimensional latent space, with the two models trained on binarised versions of the MNIST [148] and Omniglot [147] datasets using the code at `https://github.com/yburda/iwae`. The generated images used are shown in Figure 5.3.

By performing inference on the per-image posterior densities on the latent representation given image, the joint marginal likelihood of the images can be estimated as the product of estimates of the normalising constants of the individual latent posterior densities. The use of generated images allows BDMC [118] to be used to stochastically bound the marginal likelihood as described in Section 5.2 with stochastic upper and lower bounds formed with long forward and backward AIS runs (averages over 16 independent runs with 10 000 inverse temperatures as used in [264]).

As the per-image latent representations are conditionally independent given the images, chains on all the posterior densities can be run in parallel, with the experiments in this section run on a NVIDIA Tesla



(a) Generated MNIST test images.



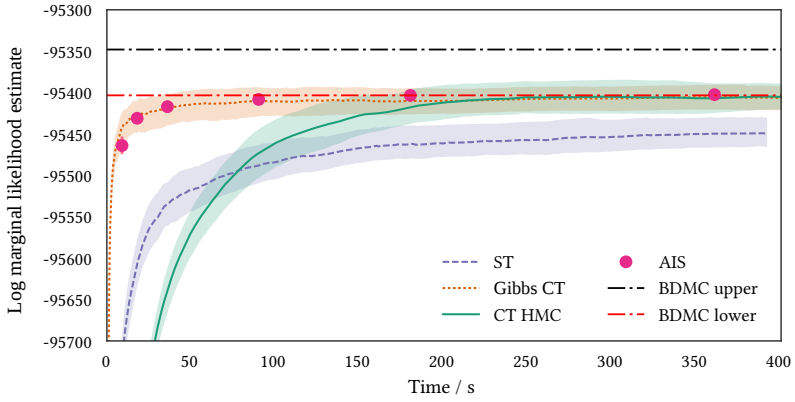
(b) Generated Omniglot test images.

Figure 5.3.: Generated image datasets from IWAE models that were used as the observed data for marginal likelihood estimates.

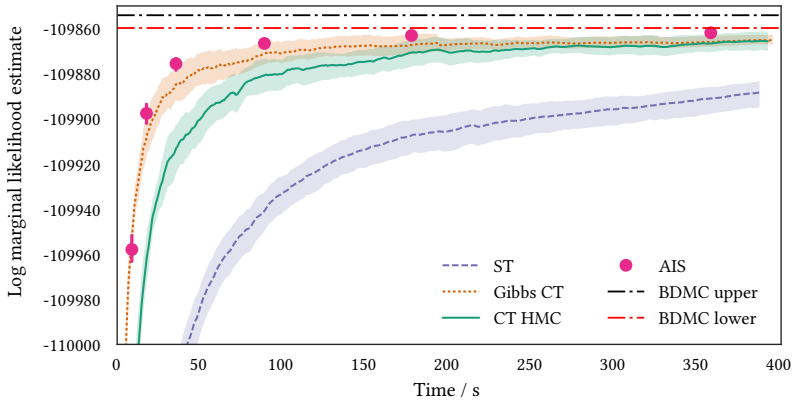
K40 GPU to exploit this inherent parallelism. The encoder of the trained IWAE models is an inference network which outputs the mean and diagonal covariance of a Gaussian variational approximation to the posterior density on the latent representation given an image and so was used to define per-image Gaussian base densities as suggested in [264]. Similarly the per-image $\log \zeta$ values were set using variational lower bound estimates for the per-image marginal likelihoods.

The results are shown in Figure 5.4, with the curves / points showing average results across 10 independent runs and filled regions / bars ± 3 standard error of means for the estimates. Here Gibbs CT and AIS perform similarly, with CT-HMC converging less quickly and simulated tempering significantly less efficient. The quick convergence of AIS and Gibbs CT here suggests the posterior densities are relatively easy for the dynamics to explore and well matched by the Gaussian base densities, limiting the gains from any more coherent exploration of the extended space by the CT-HMC updates. The higher per-leapfrog-step costs of the HMC updates in the extended space therefore mean the CT-HMC approach is less efficient overall here. The poorer performance of simulated tempering here is in part due to the generation of the discrete random indices becoming a bottleneck in the GPU implementation.

A possible partial reason for the better relative performance of AIS here compared to the experiments in Section 5.6.1 is its more effective utilisation of the parallel compute cores available when running for example on a GPU. Multiple AIS chains can be run for each data point and then the resulting unbiased estimates for each data points marginal likelihood averaged (reducing the per data point variance) before taking their product for the joint marginal likelihood estimate. While it is also possible to run multiple tempered chains per data point and similarly combine the estimates, empirically we found that greater gains in estimation accuracy came from running a single longer chain rather than multiple shorter chains of total length equivalent to the longer chain. This can be explained by the initial warm-up transients of each shorter chain having a greater biasing effect on the overall estimate compared to running longer chains. Therefore an increase in the number of parallel compute cores available seems to give greater gains for AIS versus the tempering methods.



(a) MNIST log marginal likelihood estimates.



(b) Omniglot log marginal likelihood estimates.

Figure 5.4.: Estimates of the log joint marginal likelihood of 1000 generated images under the Bernoulli decoder distributions of two IWAE models trained on the MNIST and Omniglot datasets against computation time. The black / red dashed lines show stochastic upper / lower bounds calculated using long BDMC runs. For AIS points across time axis represent increasing number of inverse temperatures: (50, 100, 200, 500, 1000, 2000). For ST, Gibbs CT and joint CT curves show estimates calculated with an increasing number of samples from chains. All curves / points show mean across 10 runs. Filled regions / error bars show ±3 standard errors of mean.

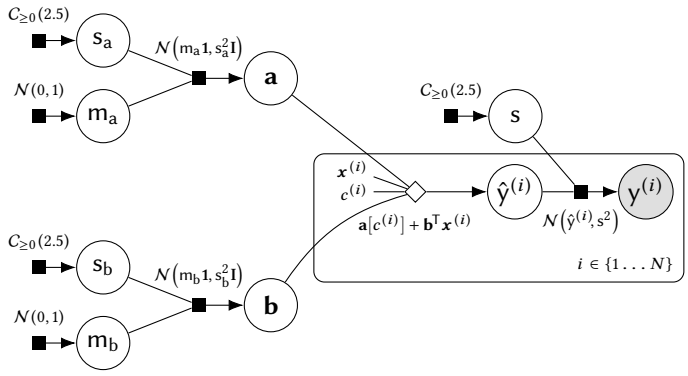


Figure 5.5.: Radon hierarchical regression model.

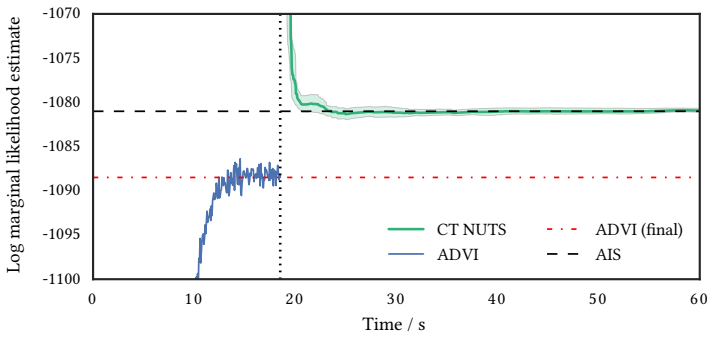


Figure 5.6.: Log marginal likelihood estimates against run time for hierarchical regression model. Black dashed line shows estimated log marginal likelihood from a long AIS run which is used as a proxy ground truth. The noisy blue curve shows the *evidence lower bound* ADVI objective over training and the red dot-dashed line the final converged value used for $\log \zeta$. The green curve shows log marginal likelihood estimates using samples from NUTS chains running on the extended joint density in the estimator (5.36), with the run time corresponding to increasing samples being included in the estimator (offset by initial ADVI run time). Curve shows mean over 10 runs and filled region ±3 standard errors of mean.

5.6.3 Hierarchical regression model

As a final experiment, we apply our CT-HMC approach to perform inference in a hierarchical regression model for predicting indoor radon measurements [96]. To illustrate the ease of integrating our approach in existing HMC-based inference software, this experiment was performed with the Python package PyMC3 [236], with its ADVI feature used to fit the base density and its implementation of the adaptive NUTS [130] HMC variant used to sample from the extended space.

The regression target in the dataset is measurements of the amount of radon gas $y^{(i)}$ in $N = 919$ households. Two continuous regressors $\boldsymbol{x}^{(i)}$ and one categorical regressor $c^{(i)}$ are provided per household. A multilevel regression model defined by the factor graph in 5.5 was used. The model includes five scalar parameters ($\sigma_{\mathbf{a}}$, $\mu_{\mathbf{a}}$, $\sigma_{\mathbf{b}}$, $\mu_{\mathbf{b}}$, ϵ), an 85-dimensional intercept vector $\mathbf{a}$ and a two-dimensional regressor coefficients vector $\mathbf{b}$, giving 92 parameters in total. As an example task, we consider inferring the model evidence (marginal likelihood) of the data under the model.

As our 'ground truth' we use a large batch of long AIS runs (average across 100 runs of 10000 inverse temperatures) on a separate Theano implementation of the model. We use ADVI to fit a diagonal covariance Gaussian variational approximation to the target density and use this as the base density. NUTS chains, initialised at samples from the base density, were then run on the extended space for 2500 iterations. The samples from these chain were used to compute estimates of the normalising constant (marginal likelihood) using the estimator (5.36). The results are shown in Figure 5.6. It can be seen that estimates from the NUTS chains in the extended continuously tempered space quickly converge to a marginal likelihood estimate very close to the AIS estimate, and significantly improve over the final lower bound on the marginal likelihood that ADVI converges to.

5.7 DISCUSSION

The continuous tempering approaches we have proposed in this chapter are a simple but powerful extension to existing simulated tempering methods which can both help exploration of distributions with multiple modes and allow estimation of the normalisation constant of the target

distribution. We propose an importance sampling based method for using all of a tempered chain's samples to estimate expectations with respect to the target distribution, overcoming the waste in the standard simulated tempering estimator which computes estimates using only a subset of the sampled states. This importance sampling estimator also allows use of a continuous inverse temperature formulation that removes the need to choose a set of inverse temperatures values.

Related importance sampling estimators have been proposed before in the context of tempering methods. Most relevant is the previously discussed *Rao–Blackwellized tempered sampling* [53], which uses an estimator directly analagous to that in (5.28) within a standard (discrete inverse temperature) simulated tempering scheme. The method in [53] is only discussed in the context of normalising constant estimation (with the direct continuous tempering analogue being (5.36)) however applying the same approach to estimation of expectations with respect to the target distribution was an obvious extension.

Importance tempering [114] also proposes use of an importance weighting scheme within a simulated tempering framework to allow using all the sampled chain states to estimate expectations with respect to the target distribution. Unlike the estimator used here, the estimator proposed in [114] uses importance weights which depend on the sampled inverse temperature indices k as well as the sampled target states $\mathbf{x}$. The sampled states $\{\mathbf{x}_s, \mathrm{k}_s\}_{s=1}^S$ are used to define a set of K per inverse temperature consistent estimators $\{\hat{\mathrm{f}}_{k,S}\}_{k=1}^K$ for expectations of functions f with respect to the target distribution P with

$$\hat{\mathrm{f}}_k = \frac{\sum_{s=1}^S w_k(\mathbf{x}_s, \mathrm{k}_s)\, f(\boldsymbol{x}_s)}{\sum_{s=1}^S w_k(\mathbf{x}_s, \mathrm{k}_s)}, \quad w_k(\boldsymbol{x}, k') = \frac{\mathrm{p}_{\mathbf{x},\mathrm{k}}(\boldsymbol{x}, 1)}{\mathrm{p}_{\mathbf{x},\mathrm{k}}(\boldsymbol{x}, k')}\, \mathbb{1}_{\{k\}}(k'). \quad (5.54)$$

These individual estimators can then be combined in to an overall consistent estimator using a convex combination

$$\hat{\mathrm{f}}_{\mathrm{IT}} = \sum_{k=1}^K \lambda_k \hat{\mathrm{f}}_{k,S} \quad \text{with} \quad \sum_{k=1}^K \lambda_k = 1. \quad (5.55)$$

The authors of [114] propose setting the free weighting parameters $\{\lambda_k\}_{k=0}^K$ using an estimate of the optimal weights in the sense of maximising the effective sample size of the overall estimator. This estimator is

compared to what is termed in [114] as a 'naive' estimator with weights $\lambda_k \propto \sum_{s=1}^{S} w_k(\boldsymbol{x}_s, k)$, which corresponds to an overall estimator

$$\hat{\mathrm{f}}_{\mathrm{NAIVE}} = \frac{\sum_{s=1}^{S} w_{\mathrm{NAIVE}}(\mathbf{x}_s, \mathrm{k}_s) f(\mathbf{x}_s))}{\sum_{s=1}^{S} w_{\mathrm{NAIVE}}(\mathbf{x}_s, \mathrm{k}_s)}, \quad w_{\mathrm{NAIVE}}(\boldsymbol{x}, k) = \frac{\mathrm{p}_{\mathrm{k}|\mathbf{x}}(1|\boldsymbol{x})}{\mathrm{p}_{\mathrm{k}|\mathbf{x}}(k \mid \boldsymbol{x})}. \tag{5.56}$$

In contrast the discrete inverse temperature analogue of the estimator we propose here in (5.28) has the form

$$\hat{\mathrm{f}}_{\mathrm{DT}} = \frac{\sum_{s=1}^{S} w_{\mathrm{DT}}(\mathbf{x}_s) f(\mathbf{x}_s))}{\sum_{s=1}^{S} w_{\mathrm{DT}}(\mathbf{x}_s)}, \quad w_{\mathrm{DT}}(\boldsymbol{x}) = \mathrm{p}_{\mathrm{k}|\mathbf{x}}(1|\boldsymbol{x}) \tag{5.57}$$

The weights in the naive estimator can be seen to have an additional division by a term depending on the sampled inverse temperature index. This introduces extra noise compared to the estimator in (5.57) which effectively analytically integrates out this inverse temperature dependence resulting in a lower variance estimator. Therefore though the 'optimal' weighting scheme proposed in [114] is demonstrated to give lower variance estimates compared to the naive estimator, it is not clear if the scheme offers an advantage over the form of estimator we propose here. Further as the estimator in [114] is reliant on binning the sampled states based on the discrete inverse temperature, it is non-trivial to extend to a continuous inverse temperature setting.

A key advantage of the proposed CT-HMC method is its ease of implementation - it simply requires running HMC in an extended state space and so can easily be used for example within existing probabilistic programming software such as PyMC3 [236] and Stan [55] as seen in the last experiment in Section 5.6. By updating the temperature jointly with the target state, it is also possible to leverage adaptive HMC variants such as NUTS [130] to perform tempering in a 'black-box' manner without a need to separately tune the updates of the inverse temperature variable.

The Gibbs CT method also provides a relatively black-box framework for tempering. Compared to ST it removes the need to choose the number and spacing of discrete inverse temperatures and also replaces generation of a discrete random variate from a categorical distribution when updating β given $\mathbf{x}$ (which as seen in Section 5.6.2 can become a computational bottleneck) with generation of a truncated exponential variate (which can be performed efficiently by inverse transform sampling). Compared to the CT-HMC approach, the Gibbs approach is

less simple to integrate in to existing HMC code due to the separate β updates, but eliminates the need to tune the temperature control mass value m and achieved similar or better sampling efficiency in the experiments in Section 5.6.

In models we considered in experiments, the sampling efficiency using the proposed continuous tempering approaches was always as good or better than simulated tempering. The comparison with AIS was less clear cut with the continuous tempering approaches outperforming AIS in the Boltzmann machine relaxation experiments but AIS generally doing better in the generative image model experiments (though Gibbs CT was competitive). The tempering approaches and AIS are less directly comparable due to the different basic inference approach being used to compute expectations — importance sampling with independent proposals — versus the Markov chain approaches of the tempering methods. This for example gave different tradeoffs in the utilisation of parallel compute, with it generally simpler to efficiently exploit more parallelism in AIS. Our proposed use of an importance sampling estimator with samples generated using a Markov chain does bear some interesting similarities with AIS, however the approaches remain quite different, with AIS using a Markov chain to construct each independent proposal rather than to sample from the proposal distribution, and the AIS normalising constant estimate is unbiased unlike the consistent estimator of our approach.

Our proposal to use variational approximations to select the base distribution helps improve the ability to scale tempering methods to complex high-dimensional target distributions were simple uninformative base distributions can lead to poor performance. Approaches such as ADVI [144] can be applied to a wide range of models with differentiable densities with minimal need for user input and efficient implementations are available in frameworks such as Stan and PyMC3.

Our use of variational inference within an MCMC framework can be viewed within the context of several existing approaches which suggest combining variational and MCMC inference methods. *Variational MCMC* [71] proposes using a variational approximation as the basis for a proposal distribution in a Metropolis-Hastings MCMC method. *MCMC and Variational Inference: Bridging the Gap* [234] includes parametrised MCMC transitions within a (stochastic) variational approximation and optimises the variational bound over these (and a base distribution's)

parameters. Here we exploit cheap but biased variational approximations to a target distribution and its normalising constant, and use them within an MCMC method which gives asymptotically exact results to help improve sampling efficiency.

6 SUMMARY

This thesis has been concerned with the development of methods for performing inference in complex probabilistic models and in particular the associated key computational challenge of approximating high-dimensional integrals. In particular we have contributed several novel Markov chain Monte Carlo approaches based on the introduction of auxiliary variables in to the chain state.

In Chapter 1 we introduced the probabilistic modelling theory and tools which we used to describe the models and methods discussed in the rest of the thesis, and concluded with a discussion of some classes of probabilistic models with key challenging features for existing inference methods: hierarchical latent variable models with both global latent variables affecting all observations and local per data point variables; simulator models which are defined by a generative process for producing samples from the model rather than by an explicit density function on model variables; and undirected models such as Boltzmann machines which tend to produce highly multimodal distributions.

Chapter 2 reviewed some the key existing approximate inference approaches for estimating high-dimensional integrals. After introducing the basic Monte Carlo integration method, we discussed some simple approaches for generating and using samples to approximate integrals and identified that the complexities of the geometry of distributions on high-dimensional spaces typically mean such methods are typically limited to only very small models or as building blocks within more efficient algorithms. We then introduced the main class of methods we focused on in the thesis - Markov chain Monte Carlo methods, and motivated the ability of this approach to provide a scalable, general purpose framework for approximating integrals with respect to probability distributions on high-dimensional spaces. The chapter concluded with a discussion of auxiliary variable approaches to constructing efficient Markov chains, and we described three instantiations of this idea - slice sampling, Hamiltonian Monte Carlo, with these methods underlying much of work discussed in the remainder of the thesis.

In Chapter 3 we introduced the *pseudo-marginal* method for constructing Markov chains using noisy unbiased estimators of the target density of interest. We concentrated in particular on the application of these approaches to inferring the posterior distribution on global variables in hierarchical latent variable models. Markov chains produced using standard pseudo-marginal methods are susceptible to getting stuck at points in the state space of many iterations and are difficult to tune using standard heuristics. By considering a reparameterisation of the density estimator as a deterministic function of the auxiliary random inputs used to compute the estimate, we proposed a *auxiliary pseudo-marginal* framework which applies separate updates to the auxiliary random inputs used in the estimator and the original target variables.

The simplest instantiation of this framework just corresponds to splitting the standard pseudo-marginal update in to two separate Metropolis–Hastings accept steps, but even this minor change is able to produce chains which are significantly easier to tune and in some cases give improved sampling efficiency. The auxiliary pseudo-marginal framework proposed also allows the use of alternative transition operators such as slice sampling algorithms, which as we demonstrated empirically are significantly less sensitive to the choices made for their free parameters than random-walk Metropolis methods. Although we found the slice sampling approaches proposed were generally less efficient than optimally tune random-walk Metropolis updates, we argue that extra robustness provided by the adaptivity of the slice sampling algorithm and minimal need for user tuning may often be more important benefits. Our emphasis was on empirical evaluation of the methods we proposed and a possible useful direction for future work would be to try to put the methods on a more firm theoretical footing, for example characterising when the proposed 'split' auxiliary pseudo-marginal update would be expected to give improved performance over the standard pseudo-marginal Metropolis–Hastings update and when it would be expected to be less beneficial.

In Chapter 4 we considered methods for performing inference in generative models where we can generate samples from the model but the density on the model variables is only implicitly defined. We demonstrated that a general framework for describing generative models is as a deterministic mapping from a vector of random inputs drawn from a known distribution, and defined a restricted class of *differentiable gener-*

ative models where the generator function which transforms inputs to simulated outputs is differentiable. Using this formulation of a generative model we showed how the *approximate Bayesian computation* (ABC) approach for inference in the parameter space of implicit generative models could be reposed in the input space to the generator. Similarly to the methods proposed in Chapter 3, this reparameterisation enables the application of more efficient MCMC transition operators such as slice sampling and Hamiltonian Monte Carlo to ABC inference problems. As we demonstrated empirically, in some cases the resulting methods are able to support inference when conditioning on the full set of observed data when standard ABC approaches are forced to used reduced dimensionality summary statistics.

For differentiable generative models we showed that computing expectations conditioned on observed values of the output of the generator corresponded to integrating against a distribution defined by an implicitly defined manifold in the generator input space. Based on this insight, we proposed a novel constrained Hamiltonian Monte Carlo method for performing inference in differentiable generative models while conditioning the model outputs to be arbitrarily close to observed values. This offers an asymptotically exact alternative to ABC in some models and, as we demonstrated empirically, the method can provide improved sampling efficiency over competing approaches by better exploiting the geometric structure of the problem, while also preserving more information about the variables being inferred.

We found that application of the standard HMC algorithm to inference in the input space of a generator based on conditioning with via a Gaussian ABC kernel did not perform particularly well, with the resulting simulated dynamics showing highly oscillatory behaviour which limited coherent exploration of the target distribution. A potential resolution to this issue would be to instead use a Riemannian-manifold Hamiltonian dynamic, using a metric exploiting the tangent space structure defined by the generator Jacobian. Although the requirement to use an implicit solver in the integrator in this case would increase the computational cost per update, it would interesting to see if as with the more costly constrained HMC dynamic considered here, whether any resulting improved exploration of the space was able to outweigh the higher computational costs.

An assumption of differentiability of the generative model is restrictive, with many of the models ABC approaches are commonly applied to involving discrete latent variables or having generators with branching control flow logic that introduce discontinuities. It would be interesting to consider if the general approach of parameterising a generative model as a deterministic map can also be utilised in these setting to suggest more efficient inference methods. From a software engineering perspective, it would also be useful to develop approaches for automating the tracking of calls to a pseudo-random number generator in simulator code, to allow for models to be more easily parameterised in a form suitable for the inference approaches we suggest.

In Chapter 5 we introduced a novel tempering method which introduces an auxiliary *inverse temperature* variable in to the state of a Markov chain to improve the exploration of challenging multimodal distributions and allow for normalising constants to be estimated. Unlike existing simulated tempering methods, the approach we proposed uses a continuous temperature. This reduces the tuning burden on users by eliminating the need to choose a set of of inverse temperature values. Further using a continuous formulation allows the inverse temperature variable to be jointly updated with the original variables in a chain, making it simple to apply efficient gradient based Hamiltonian Monte Carlo transition operators to the augmented space. We also proposed a novel importance sampling inspired approach for using all of the sampled states of a tempered chain to estimate expectations with respect to the target distribution, this making more effective usage of the computation performed than the standard estimator used in simulated tempering which computes averages over only a small subset of sampled states. We also demonstrated that variational inference is a natural approach for fitting the base distribution bridged to by the tempered dynamic, with the use of more informative base distributions key to scaling tempering approaches to large-scale problems by helping to flatten the marginal density on the inverse temperature.

In practice in complex target distributions even when using a base distribution fit using a variational approach, the tempered dynamic will still typically struggle to move up and down the inverse temperature range. We briefly outlined the basics of an iterative approach for trying to exploit information from initial pilot chains to improve mixing in subsequent chains by iteratively improving the fit of the base distribu-

tion and making use of improved estimates of the normalising constant. Ideally this would be done in an online manner to prevent the need to do multiple distinct runs, however maintaining reversibility in such a scheme would challenging. It may be fruitful to consider however if adaptive MCMC approaches could be employed in this setting.

We employed a standard HMC transition operator when performing joint updates on the extended space. The distribution on the extended space has a rich geometric structure with for example the smoothness of the distribution typically increasing in regions corresponding to low inverse temperatures. It may be possible to exploit this structure to construct more efficient HMC updates in the extended space, for example using a mass metric which depends on the current inverse temperature value within a Riemannian-manifold HMC framework.

A DISTRIBUTION DEFINITIONS

Name	Parameters	Shorthand	Density	Support
Bernoulli	$\pi \in [0,1]$	$\mathrm{Ber}(x \mid \pi)$	$\pi^x(1-\pi)^{(1-x)}$	$x \in \{0,1\}$
Categorical	$\boldsymbol{\pi} \in \mathbb{S}^K$	$\mathrm{Cat}(x \mid \boldsymbol{\pi})$	$\sum_{k=1}^{K}(\mathbb{1}_{\{k\}}(x)\pi_k)$	$x \in \{1 \dots K\}$

Table A.1.: Definitions of densities of parameteric distributions for discrete random variables used in this thesis.

Name	Parameters	Shorthand	Density
Normal	$\mu \in \mathbb{R}$: mean $\sigma > 0$: standard deviation	$\mathcal{N}\left(x \mid \mu, \sigma^2\right)$	$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
Multivariate normal	$\boldsymbol{\mu} \in \mathbb{R}^D$: mean vector $\boldsymbol{\Sigma} \in \mathcal{S}_{++}^D$: covariance matrix	$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\frac{1}{\sqrt{(2\pi)^D\lvert\boldsymbol{\Sigma}\rvert}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$
Logistic	$\mu \in \mathbb{R}$: location $\sigma > 0$: scale	$\mathrm{Logistic}(x \mid \mu, \sigma)$	$\frac{1}{4\sigma}\cosh\left(\frac{x-\mu}{2\sigma}\right)^{-2}$
Inverse cosh	$\mu \in \mathbb{R}$: location $\sigma > 0$: scale	$\mathrm{InvCosh}(x \mid \mu, \sigma)$	$\frac{1}{2\sigma}\cosh\left(\frac{\pi(x-\mu)}{2\sigma}\right)^{-1}$

Table A.2.: Definitions of densities of parameteric distributions for unbounded real random variables used in this thesis.

Name	Parameters	Shorthand	Density	Support
Log-normal	$\mu \in \mathbb{R}$: log mean $\sigma > 0$: log standard deviation	$\mathrm{LogNorm}(x \mid \mu, \sigma^2)$	$\frac{1}{x\sqrt{2\pi}\sigma} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right)$	$x > 0$
Multivariate log-normal	$\boldsymbol{\mu} \in \mathbb{R}^D$: log mean $\boldsymbol{\Sigma} \in \mathcal{S}^D_{++}$: log covariance	$\mathrm{LogNorm}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\frac{\exp\left(-\frac{1}{2}(\log \boldsymbol{x} - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\log \boldsymbol{x} - \boldsymbol{\mu})\right)}{\prod_{d=1}^D (x_d) \sqrt{(2\pi)^D \lvert\boldsymbol{\Sigma}\rvert}}$	$\boldsymbol{x} \in [0, \infty)^D$
Exponential	$\lambda > 0$: rate	$\mathrm{Exp}(x \mid \lambda)$	$\lambda \exp(-\lambda x)$	$x \geq 0$
Uniform	$a \in \mathbb{R}$: minimum $b \in \mathbb{R}$: maximum, $b > a$	$\mathcal{U}(x \mid a, b)$	$\frac{1}{b-a} \mathbb{1}_{[a,b]}(x)$	$a \leq x \leq b$
Half-Cauchy	$\gamma > 0$: scale	$C_{\geq 0}(x \mid \gamma)$	$\frac{2}{\pi\gamma}\left(1 + \frac{x^2}{\gamma^2}\right)^{-1}$	$x \geq 0$

Table A.3.: Definitions of densities of parameteric distributions for bounded real random variables used in this thesis.

B COMPUTATION GRAPHS

In Chapter 1 we introduced graphical models as a compact way of representing the structure in probabilistic models. Directed factor graphs in particular offer a natural approach for representing generative models. A directed graph can be used to specify a generative process via *ancestral sampling*, with values for the variables in the graph successively calculated in a forward pass consisting of a combination of deterministic operations, represented by deterministic factors, and stochastic sampling operations, represented by probabilistic factors.

Computation graphs [16] are a directed graph based representation of the operations involved in evaluating a mathematical expression. Similarly to a directed factor graph, a computation graph can be considered as specifying a generative process - generation of the expression outputs given inputs - computed via forward pass through the graph. The main difference of a forward pass through a computation graph compared to an ancestral sampling pass through a directed factor graph is that the inputs to a computation graph are assumed to be given rather than sampled from marginal densities and the intermediate operations are all deterministic. In this appendix we briefly review the key concepts of computation graphs and in particular their application to perform automatic differentiation.

Here we will distinguish between two types of nodes in a computation graph. *Variable nodes* correspond to variables which hold either inputs to the computation or intermediate results corresponding to the outputs of sub-expressions. *Operation nodes* describe how non-input variable nodes are computed as functions of other variable nodes. In other presentations of computation graphs often the operation nodes are instead implicitly represented by directed edges between variable nodes. However analogously to the more explicit factorisation afforded by directed factor graphs compared to directed graphical models, directly representing operations as nodes allows finer grained information about the decomposition of the operations associated with a computation graph to be included.

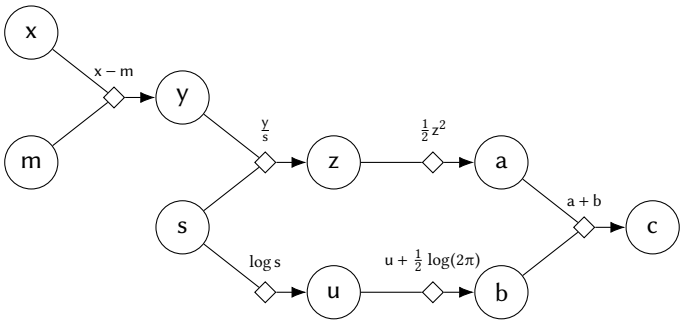


Figure B.1.: Example computation graph corresponding to calculation of the negative log density of a univariate normal distribution.

The direct overlap in our notation to represent variable and operation nodes in computation graphs and that used to represent (random) variable nodes and deterministic factor nodes in factor graphs is intentional. Although often the operations associated with a deterministic node in a factor graph will be more complex than the operations usually represented by nodes in a computation graph, this is only a matter of granularity of representations - fundamentally they perform the same role. Importantly this means we can treat subgraphs of a factor graphs consisting of only variable and deterministic factor nodes as computation graphs and if the operations performed by the deterministic nodes are differentiable, use reverse-mode automatic differentiation to efficiently propagate derivatives through these sub-graphs.

As with directed factor graphs, computation graphs cannot contain directed cycles. This does not preclude recursive and recurrent computations however as these can always be unrolled to form a directed acyclic graph. The 'mathematical expressions' a computation graph is constructed to evaluate can be arbitrarily complex - a computation graph corresponding to the evaluation of any numerical algorithm can always be constructed including use of arbitrary nested flow control and branching statements.

An example of a computation graph representing the calculation of the negative log density of a univariate normal distribution, i.e.

$$c = \frac{1}{2}\left(\frac{x - m}{s}\right)^2 + \log s + \frac{1}{2}\log(2\pi) \tag{B.1}$$

is shown in Figure B.1. The graph inputs have chosen to be the value of the random variable (x) to evaluate the density at and the mean (m) and the standard deviation (s) parameters of the density.

Variable nodes in the computation graph have been represented by labelled circles and operation nodes with labelled diamonds. Undirected edges connecting from a variable node to an operation node correspond to the inputs to the operation, and directed edges from an operation node to variable nodes to the outputs of the operation.

The computation graph associated with an expression is not uniquely defined. There will usually be multiple possible orderings in which operations can be applied to achieve the same result (up to differences due to non-exact floating point computation). Similarly what should be considered a single operation to be represented by a node in the computation graph as opposed to being split up into a sub-graph of multiple operations is a matter of choice. For example in Figure B.1 the addition of the constant $\frac{1}{2}\log(2\pi)$ could have been included at various other points in the graph and the operation $\frac{1}{2}z^2$ could have been split in to separate multiplication and exponentiation operations.

B.1 AUTOMATIC DIFFERENTIATION

The main motivation for representing an expression as a computation graph is to formalise an efficient general procedure termed automatic differentiation for automatically calculating derivatives of the output of an expression with respect to its inputs [21, 196]. The key ideas in automatic differentiation are to use the chain rule to decompose the derivatives into products and sums of the partial derivatives of the output of each individual operation in the expression with respect to its input, and to use an efficient recursive accumulation of these partial derivative sum-products corresponding to a traversal of the computation graph such that multiple derivatives can be efficiently calculated together.

Depending on how the computation graph is traversed to accumulate the derivative terms, different modes of automatic differentiation are possible. Of most use in this thesis will be *reverse-mode accumulation* [242], in which the derivatives of an output node with respect to all input nodes are accumulated by a reverse pass through the computation graph from the output node to inputs.

As an example the partial derivatives of the expression for univariate normal log density given in (B.1) with respect to x, m and s can be

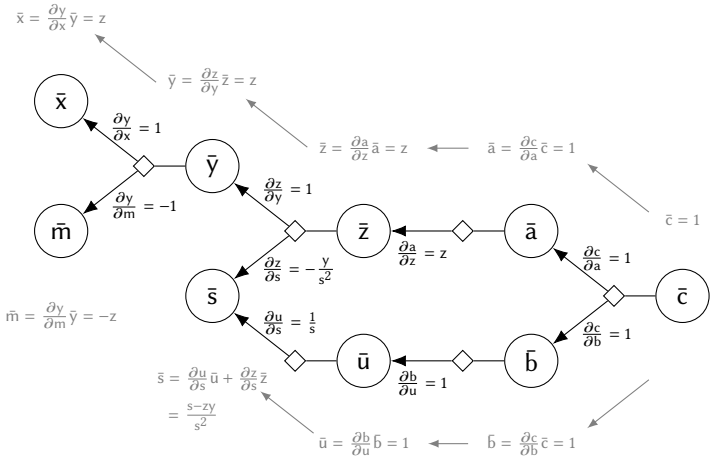


Figure B.2.: Visualisation of applying reverse-mode automatic differentiation to the computation graph in Figure B.1 to calculate the derivatives of the negative log density of a univariate normal distribution.

decomposed using the chain rule in terms of the intermediate variables in the computation graph shown in Figure B.1 as

$$\frac{\partial c}{\partial x} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial x}, \tag{B.2}$$

$$\frac{\partial c}{\partial m} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial m}, \tag{B.3}$$

$$\frac{\partial c}{\partial s} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s} + \frac{\partial c}{\partial b}\frac{\partial b}{\partial u}\frac{\partial u}{\partial s}. \tag{B.4}$$

We can immediately see that some of the chains of products of partial derivatives are repeated in the different derivative expressions - for example $\frac{\partial c}{\partial a}\frac{\partial a}{\partial z}$ appears in the expressions for all three derivatives. Reverse-mode accumulation is effectively an automatic way of exploiting these possibilities for reusing calculations.

Figure B.2 shows a visualisation of reverse-mode accumulation applied to the computation graph in Figure B.1. The first step is for a *forward pass* through the graph to be performed, i.e. values are provided for each of the input variables and then each of the intermediate and output variables calculated from the incoming operation applied to their parent values. Importantly the values of all variables in the graph calculated during the forward pass must be maintained in memory.

The *reverse pass* recursively calculates the values of the partial derivatives of the relevant output node with respect to each variable node in the graph - we will term these intermediate derivatives *accumulators*

Algorithm 16 Reverse-mode automatic differentiation.

Input: $\{x_i\}_{i=1}^{M}$: computation graph input variables,
Pa : indices of parent variables to an operation given its index,
Ch : indices to child operations of a variable given its index,
$\{f_i\}_{i=M+1}^{N}$: computation graph operations in topological order,
$\left\{\{\partial_j f_i\}_{j\in \mathrm{Pa}(i)}\right\}_{i=M+1}^{N}$: operation partial derivatives WRT parent variables.
Output: x_N : function output,
$\{\bar{x}_i\}_{i=1}^{M}$: partial derivatives of function output WRT input variables.

for $i \in \{M+1 \dots N\}$ **do** ▷ Forward pass
    $x_i \leftarrow f_i\left(\{x_j\}_{j\in \mathrm{Pa}(i)}\right)$
$\bar{x}_N \leftarrow 1$ ▷ Reverse pass
for $i \in \{N-1 \dots 1\}$ **do**
    $\bar{x}_i \leftarrow \sum_{j\in \mathrm{Ch}(i)} \bar{x}_j \partial_i f_j(x_i)$
return x_N, $\{\bar{x}_i\}_{i=1}^{M}$

denoted with barred symbols in Figure B.2 e.g. $\bar{a} = \frac{\partial c}{\partial a}$. The reverse pass begins by seeding an accumulator for the output node to one (i.e. $\bar{c} = \frac{\partial c}{\partial c} = 1$ in Figure B.2).

Accumulators for the input variables of an operation are calculated by multiplying the accumulator for the operation output by the partial derivatives of the operation output with respect to each input variable. For non-linear operations multiplying by the operator partial derivatives will require access to the value of the input variables calculated in the forward pass. If a variable is an input to multiple operations, the derivative terms from each operation are added together in the relevant accumulator, as for example shown for $\bar{s}$ in Figure B.2. By recursively applying these product and sum operations, the derivatives of the output with respect to all variables in the graph can be calculated. A general description of the method for computation graphs with a single output node and multiple inputs is given in Algorithm 16.

This reverse accumulation method allows computation of numerically exact (up to floating point error) derivatives of a single output variable in a computation graph with respect to *all input variables* with a computational cost, in terms of the number of atomic operations which need to be performed, that is a constant factor of the cost of the evaluation of the original expression represented by the computation graph in the forward pass. The constant factor is typically two to three and at most six [17]. This efficient computational cost is balanced by the requirement that the values of all intermediate variables in the computation graph evaluated in the forward pass through the graph must be stored

in memory for the derivative accumulation in a reverse pass, which for large computational graphs can become a bottleneck.

To calculate the full Jacobian from a computation graph representing a function with M inputs $\{\mathsf{x}_i\}_{i=1}^{M}$ and N outputs $\{\mathsf{y}_i\}_{i=1}^{N}$, i.e. the $N \times M$ matrix $\boldsymbol{J}$ with entries $J_{i,j} = \frac{\partial \mathsf{y}_i}{\partial \mathsf{x}_j}$, we can do a single forward pass and N reverse passes each time accumulating the derivatives of one output variable with respect to all inputs. This leads to an overall computational cost that is $O(N)$ times the cost of a single (forward) function evaluation to evaluate the full Jacobian. As each of the reverse passes can trivially be run in parallel (in addition to any parallelisation of the operations in the forward and reverse passes themselves), this $O(N)$ factor in the operation count need not corresponds to an equivalent increase in compute time.

An alternative to reverse-mode accumulation is *forward-mode accumulation* [261], which insteads accumulates partial derivatives with respect to a single input variable alongside the forward pass through the graph. In contrast to reverse-mode, this allows calculation of the partial derivatives of all output variables with respect to a single input variable at a computational cost that is a constant factor of the cost of the evaluation of the original expression in the forward pass. Forward-mode accumulation therefore allows evaluation of the Jacobian of a function with M inputs and N outputs at an overall computational cost that is $O(M)$ times the cost of a single function evaluation.

For functions with $M \gg N$, e.g. scalar valued functions of multiple inputs, reverse-mode accumulation is generally therefore significantly more efficient at computing the Jacobian. Forward-mode accumulation is however useful for evaluating the Jacobian of functions with $N \gg M$, and also has the advantage over reverse-mode accumulation of avoiding the requirement to store the values of intermediate variables from the forward pass for the reverse pass(es).

C OPTIMISATION-BASED APPROXIMATE INFERENCE

The sampling-based approaches to approximate inference discussed in Chapter 2 although a significant improvement in terms of computational complexity over quadrature methods can still be computationally demanding. In particular the MCMC methods which were identified as most suitable for inference in large, complex probabilistic models, will involve a minimum of one evaluation of the target distribution density per generated MCMC sample if using for example a simple random-walk Metropolis method and potentially tens or hundreds of density evaluation per sample if using more complex schemes such as slice sampling or the Hamiltonian Monte Carlo. Typically chains need to be run on the order of 10^2 to 10^4 iterations to ensure adequate converge of to the target distribution and to give a sufficient number of effective samples to get reasonable estimates of the expectations of interest, with generally running multiple chains preferred to give additional robustness and to allow convergence diagnostics.

In large complex models each target density evaluation may be computationally expensive. In particular when performing inference conditioned on large sets of observed data the target density typically factorise into a product (or sum in log space) of per datapoint factors. This means the cost of each target density evaluation scales with the number of datapoints and so can become appreciable for large datasets. Alongside the increase in computational demands for large (in the sense of number of datapoints) datasets, for common forms of probabilistic models such as observed variables which are *independently and identically distributed* (IID) given a set of fixed dimension unobserved variables (parameters), local asymptotic normality results means that the target (posterior) distribution is increasingly well approximated by a multivariate normal distribution as the number of observed data points increases.

In this appendix we review an alternative class of approximate inference methods which tradeoff a generally lower computational cost than

MCMC approaches for a loss of the ability to represent integrals across complex distributions with arbitrary accuracy and so asymptotic exactness guarantees. The central idea of these methods is to try to find a normalised probability density $q(\boldsymbol{x})$ from a 'simple' family that in some sense approximates the target density, i.e. $p(\boldsymbol{x}) \approx q(\boldsymbol{x})$. Depending on the family chosen for q, integrals of some functions f against the target density p, can be approximated by analytic solutions to integrals of f against q e.g. if $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma)$ then we can approximate the mean of the target density as $\boldsymbol{\mu}$ and the covariance as Σ. To compute integrals of more general functions f we typically still need to resort to using a Monte Carlo approach; generally it is possible to directly generate independent samples from q however while usually this is not be the case for p hence this two-step approach still offers (computational) advantages over directly applying a Monte Carlo approach. Often the approaches we discuss also allow estimation of the normalising constant Z which may be needed for model comparison.

C.1 LAPLACE'S METHOD

For target densities p defined with respect to a D-dimensional Lebesgue measure λ^D, a simple approach for computing a multivariate normal approximation q to p is *Laplace's method*. Although not always strictly required, in general the method will work better for target densities with unbounded support, and more generally for targets which are as 'close to normal' as possible. Therefore a useful initial step will often be to apply a change of variables to the target density, such that the density on the transformed space has unbounded support, for example working with the density on the logarithm of a random variable with support only on positive values.

The key idea in Laplace's method is to form a truncated Taylor series approximation to the logarithm of the unnormalised target density

$$\begin{aligned}\log \tilde{p}(\boldsymbol{x}) \approx \log \tilde{p}(\boldsymbol{x}^*) + \boldsymbol{g}(\boldsymbol{x}^*)^\mathsf{T}(\boldsymbol{x} - \boldsymbol{x}^*) \\ + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^\mathsf{T}\boldsymbol{H}(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*),\end{aligned} \tag{C.1}$$

where the *gradient* and *Hessian* of $\log \tilde{p}$ are defined respectively as

$$\boldsymbol{g}(\boldsymbol{x}) = \frac{\partial \log \tilde{p}(\boldsymbol{x})}{\partial \boldsymbol{x}}^\mathsf{T} \quad \text{and} \quad \boldsymbol{H}(\boldsymbol{x}) = \frac{\partial^2 \log \tilde{p}(\boldsymbol{x})}{\partial \boldsymbol{x} \partial \boldsymbol{x}^\mathsf{T}}. \tag{C.2}$$

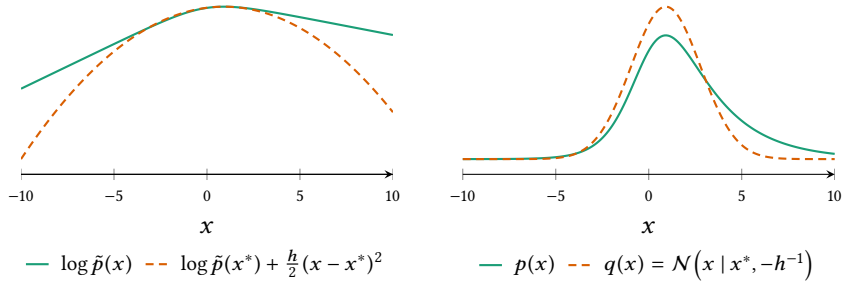


Figure C.1.: Univariate example of Laplace's method. Left axis shows the logarithm of the unnormalised target density $\log \tilde{p}(x)$ (green curve) and the corresponding quadratic Taylor series approximation $\log \tilde{p}(x^*) + \frac{h}{2}(x - x^*)^2$ (dashed orange curve) around the maxima x^* with $h = (\partial^2 \log \tilde{p} / \partial x^2)|_{x^*}$. The right axis shows the corresponding normalised target density $p(x)$ (green curve) and approximate density $q(x) = \mathcal{N}\left(x \mid x^*, -h^{-1}\right)$ (dashed orange curve).

If the point $\boldsymbol{x}^*$ the expansion is formed around is chosen to be a (local) maxima of $\log \tilde{p}$, which necessarily means that the gradient is zero, $\boldsymbol{g}(\boldsymbol{x}^*) = \boldsymbol{0}$, and the Hessian is negative definite, $\boldsymbol{H}(\boldsymbol{x}^*) \prec 0$, then

$$\log \tilde{p}(\boldsymbol{x}) \approx \log \tilde{p}(\boldsymbol{x}^*) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^\mathsf{T} \boldsymbol{H}(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*). \tag{C.3}$$

Taking the exponential of both sides we therefore have that

$$\tilde{p}(\boldsymbol{x}) \approx \tilde{p}(\boldsymbol{x}^*) \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^\mathsf{T}(-\boldsymbol{H}(\boldsymbol{x}^*))(\boldsymbol{x} - \boldsymbol{x}^*)\right). \tag{C.4}$$

This has the form of an unnormalised multivariate normal density with mean $\boldsymbol{x}^*$ and inverse covariance (precision) $-\boldsymbol{H}(\boldsymbol{x}^*)$.

This suggests setting the approximate density q to a multivariate normal density $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{x}^*, \boldsymbol{C})$ with $\boldsymbol{C} = -\boldsymbol{H}(\boldsymbol{x}^*)^{-1}$, i.e.

$$q(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{C}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^\mathsf{T} \boldsymbol{C}^{-1}(\boldsymbol{x} - \boldsymbol{x}^*)\right). \tag{C.5}$$

An example of applying Laplace's method to fit a normal approximation to a univariate generalised logistic target is shown in Figure C.1.

As $q(\boldsymbol{x}^*) \approx p(\boldsymbol{x}^*) = \tilde{p}(\boldsymbol{x}^*)/Z$ we can also form an approximation $\tilde{Z}$ to the normalising constant Z for the target density

$$Z \approx \tilde{Z} = (2\pi)^{\frac{D}{2}} |\boldsymbol{C}|^{\frac{1}{2}} \tilde{p}(\boldsymbol{x}^*). \tag{C.6}$$

A matrix $\boldsymbol{M} \in \mathbb{R}^{D \times D}$ is positive semi definite, *denoted $\boldsymbol{M} \succeq 0$, iff $\boldsymbol{x}^\mathsf{T}\boldsymbol{M}\boldsymbol{x} \geq 0$ $\forall \boldsymbol{x} \in \mathbb{R}^D$ and* positive definite, *denoted $\boldsymbol{M} \succ 0$, if the inequality is made strict. Corresponding definitions for a* negative semi definite *matrices, $\boldsymbol{M} \preceq 0$, and* negative definite *matrices, $\boldsymbol{M} \prec 0$, are formed by reversing the sign of the inequality.*

To use Laplace's method we need to be able to find a maxima of $\log \tilde{p}$ and evaluate the Hessian at this point. For simple unimodal target densities it may be possible to find the maxima and corresponding Hessian analytically. More generally if the gradient of $\log \tilde{p}$ can be calculated (using for example reverse-mode automatic differentiation), then a maxima can be found by performing iterative gradient ascent. The Hessian can then be evaluated at this point using analytic expressions for the second partial derivatives or again by using automatic differentiation (by computing the Jacobian of the gradient of $\log \tilde{p}$).

Though relatively simple to calculate, Laplace's method will often result in an approximate density which fits poorly to the target. As it only uses local information about the curvature of the (log) target density at the mode, away from the mode the approximate density can behave very differently from the target density, for instance observe the poor fit to the tails of the target of the example shown in Figure C.1. For multimodal densities, several different Laplace approximations can be calculated, each likely to at best capture a single mode well. For target densities which are well approximated by a normal distribution, for instance due to asymptotic convergence to normality of a posterior for IID data, Laplace's method can give reasonable results however.

C.2 VARIATIONAL METHODS

Laplace's method is limited by using information about the target density evaluated at only one point to fit the approximation. An alternative approach is to instead try to fit the approximate density based on minimising a global measure of 'goodness of fit' to the target; this is the strategy employed in *variational inference*.

The naming of variational inference arises from its roots in the *calculus of variations*, which is concerned with functionals (loosely a function of a function, often defined by a definite integral) and their derivatives. In particular it is natural to define the measure of the 'goodness of fit' of the approximate density to the target as a functional of the approximate density. The value of this functional is then minimised with respect to the approximate density function.

The most common functional used to define goodness of fit in variational inference is the KL divergence [145]. The KL divergence in its

most general form is defined for a pair of probability measures P and Q on a space X with P absolutely continuous with respect to Q as

$$\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] = \int_X \log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right) \mathrm{d}P, \tag{C.7}$$

which is read as the KL divergence from P to Q. The KL divergence is always non-negative $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] \geq 0$, with equality if and only if $P = Q$ almost everywhere. Intuitively the KL divergence gives a measure of how 'close' two measures are[1], however it is not a true distance as it is asymmetric: in general $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] \neq \mathbb{D}_{\mathrm{KL}}[Q \,\|\, P]$.

Generally we will work with probability densities rather than underlying probability measures. If p and q are the densities of two probability measures P and Q defined with respect to the same base measure μ on a space X, i.e. $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$ and $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$, then we will denote the KL divergence from P to Q in terms of the densities p and q by $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q] = \mathbb{D}_{\mathrm{KL}}[P \,\|\, Q]$, and from the definition (C.7) we have that

$$\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q] = \int_X p(x) \, \log \frac{p(x)}{q(x)} \, \mathrm{d}\mu(x), \tag{C.8}$$

with absolute continuity of P with respect to Q corresponding to a requirement that $p(x) = 0 \; \forall x \in X : q(x) = 0$. Somewhat loosely, we will refer to $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$ as the KL divergence from the (density) p to the (density) q rather than referring to the underlying measures.

When used without further qualification, variational inference is generally intended to mean inference performed by minimising a variational objective corresponding to the KL divergence from an approximate density q to the target density p. More specifically using the decomposition of the target density into an unnormalised density $\tilde{p}$ and normalising constant Z we have that

$$\mathbb{L}[q] = \log Z - \mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p] = \int_X q(\boldsymbol{x}) \, \log \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})} \, \mathrm{d}\mu(\boldsymbol{x}), \tag{C.9}$$

with $\mathbb{L}[q]$ the specific objective usually maximised in variational inference problems, with all terms in the integrand being evaluable pointwise. As $\log Z$ is constant with respect to the approximate density, maximising $\mathbb{L}$ with respect to q is equivalent to minimising $\mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p]$. Due

1 From an information theory perspective $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q]$ is typically termed the *relative entropy of P with respect to Q* and measures the expected information loss (in *nats* for base-e logarithms or *bits* for base-2 logarithms) of using Q to model samples from P.

to the non-negativity of the KL divergence we have that the following inequality holds

$$\mathbb{L}[q] \leq \log Z. \tag{C.10}$$

When the target density p corresponds to a posterior $\mathrm{p}_{\mathbf{x}|\mathbf{y}}$ on latent variables $\mathbf{x}$ given observed variables $\mathbf{y}$ and $\tilde{p}$ the corresponding joint density $\mathrm{p}_{\mathbf{x},\mathbf{y}}$, the normalising constant Z is equal to the model evidence term $\mathrm{p}_{\mathbf{x}}$ in Bayes' theorem. As $\mathbb{L}$ is a lower bound on $\log Z$ and so the (log) model evidence, the variational objective $\mathbb{L}$ is therefore sometimes termed the ELBO in this context.

Using the KL divergence from the approximate to target density as the variational objective is not the only choice available. One obvious alternative is the reversed form of the KL divergence, $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$ from the target density to the approximate density. In general as this form of the divergence involves evaluating an integral with respect to the target density, precisely the intractable computational task we are hoping to find an approximate solution, direct applications of this approach are limited to toy problems were this integral can be solved exactly or efficiently approximated.

An approach called EP [176] however locally optimises an objective closely related to $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$. EP is generally applied to target distributions with a density which factorise in to a product of (often per datapoint) factors

$$\tilde{p}(\boldsymbol{x}) = \prod_{i \in I} \tilde{p}_i(\boldsymbol{x}). \tag{C.11}$$

An approximate density is defined with an equivalent factorisation

$$q(\boldsymbol{x}) = \prod_{i \in I} q_i(\boldsymbol{x}), \tag{C.12}$$

with each q_i factor restricted to be the density of an exponential family distribution. EP then fits the individual approximate factors by iteratively for each $j \in I$ minimising

$$\min_{q_j} \mathbb{D}^{\mu}_{\mathrm{KL}}\left[\tilde{p}_j(\boldsymbol{x}) \prod_{i \in I \setminus \{j\}} q_i(\boldsymbol{x}) \,\middle\|\, q_j(\boldsymbol{x}) \prod_{i \in I \setminus \{j\}} q_i(\boldsymbol{x})\right]. \tag{C.13}$$

This is similar to minimising the KL divergence from the individual target factor $\tilde{p}_j$ to the corresponding approximate factor q_j, i.e. $\mathbb{D}^{\mu}_{\mathrm{KL}}[\tilde{p}_j \,\|\, q_j]$, but (C.13) instead weights the integral by the density of the 'cavity distribution' formed by current approximation of the product of the re-

maining target factors. Ideally as training proceeds the cavity distribution becomes an increasingly good approximation to the product of the true remaining factors and so EP locally minimises an objective increasingly close to $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$. The additional context provided by weighting by the cavity distribution density favours approximate factors q_j which fit well to the true factor $\tilde{p}_j$ where the mass of the current global approximation is concentrated. This is usually a significant improvement over simply fitting each q_j individually by minimising $\mathbb{D}^{\mu}_{\mathrm{KL}}[\tilde{p}_j \,\|\, q_j]$ which will often fit a very poor global approximation.

The KL divergence can be considered as a special case of a broader class of α-divergences. In particular the *Rényi divergence* [85, 223] of order $\alpha > 0, \alpha \neq 1$ between two probability measures P and Q with probability densities $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$ and $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$ on a space X is defined as

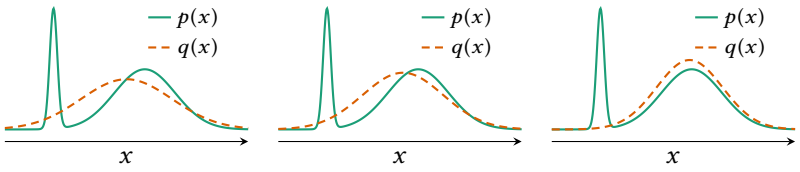
$$\mathbb{D}_{\alpha}[P \,\|\, Q] = \mathbb{D}^{\mu}_{\alpha}[p \,\|\, q] = \frac{1}{\alpha - 1} \log\left(\int_X p(\boldsymbol{x})^{\alpha} \, q(\boldsymbol{x})^{1-\alpha} \, \mathrm{d}\mu(\boldsymbol{x}) \right). \quad \text{(C.14)}$$

For $\alpha > 0$, $\mathbb{D}_{\alpha}[P \,\|\, Q]$ is a valid divergence, that is $\mathbb{D}_{\alpha}[P \,\|\, Q] \geq 0$ with equality if and only if $P = Q$ almost everywhere. The definition can also be extended to the cases $\alpha = 1$ and $\alpha = 0$ by considering limits of (C.14). Using L'Hôpital's rule it can be shown that $\lim_{\alpha \to 1} \mathbb{D}_{\alpha}[P \,\|\, Q] = \mathbb{D}_{\mathrm{KL}}[P \,\|\, Q]$. For $\alpha \to 0$, we have that $\mathbb{D}_{\alpha}[P \,\|\, Q] \to -\log P(\mathrm{supp}(Q))$ where $\mathrm{supp}(Q)$ represents the support of the probability measure Q; in this case $\mathbb{D}_{\alpha}[P \,\|\, Q]$ is no longer a valid divergence as it is equal to zero whenever $\mathrm{supp}(P) = \mathrm{supp}(Q)$. It can also be shown that for $\alpha \notin \{0, 1\}$ that $\mathbb{D}_{\alpha}[P \,\|\, Q] = \frac{\alpha}{1-\alpha} \mathbb{D}_{1-\alpha}[Q \,\|\, P]$. This motivates extending the definition in (C.14) for $\alpha < 0$, in which case we have that $\mathbb{D}_{\alpha}[P \,\|\, Q] = \frac{\alpha}{1-\alpha} \mathbb{D}_{1-\alpha}[Q \,\|\, P] \leq 0$ [156].

Analogously to using the decomposition of the target density p in to an unnormalised density $\tilde{p}$ and unknown normaliser Z when defining the previous variational objective in (C.9), it is observed in [156] that a *variational Rényi bound*, $\mathbb{L}_{\alpha}$, can be defined as

$$\mathbb{L}_{\alpha}[q] = \log Z - \mathbb{D}^{\mu}_{\alpha}[q \,\|\, p] = \frac{1}{1 - \alpha} \log \int_X q(\boldsymbol{x}) \left(\frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})} \right)^{1-\alpha} \mathrm{d}\mu(\boldsymbol{x}). \quad \text{(C.15)}$$

For $\alpha > 0$, we have that $\mathbb{D}^{\mu}_{\alpha}[q \,\|\, p] \geq 0$ and so $\mathbb{L}_{\alpha}$ is a lower bound on the $\log Z$, analogously to the ELBO, and we should maximise $\mathbb{L}_{\alpha}$ with respect to q to minimise $\mathbb{D}^{\mu}_{\alpha}[q \,\|\, p]$. For $\alpha < 0$ we have instead that $\mathbb{D}^{\mu}_{\alpha}[q \,\|\, p] \leq 0$ and so $\mathbb{L}_{\alpha}$ is an upper bound on $\log Z$ and that we should



(a) $\mathbb{D}^{\lambda}_{\mathrm{KL}}[p \,\|\, q]$ (b) $\mathbb{D}^{\lambda}_{\alpha}[p \,\|\, q],\ \alpha = \frac{1}{2}$ (c) $\mathbb{D}^{\lambda}_{\mathrm{KL}}[q \,\|\, p]$

Figure C.2.: Comparison of approximate densities fitted under different variational objectives. Each plot shows a bimodal target density $p(x)$ and a normal approximate density $q(x) = \mathcal{N}\left(x \,|\, \mu, \sigma^2\right)$ where μ and σ have been set to values which minimise the variational objective shown in the caption.

minimise $\mathbb{L}_{\alpha}$ to minimise $\mathbb{D}^{\mu}_{1-\alpha}[p \,\|\, q]$ (note the swapped order of the density arguments). An equivalent observation of the possibility of upper bounding $\log Z$ is made in [76] with a reparameterised version of (C.15) in terms of $n = 1 - \alpha > 1$.

As generally the family chosen for the approximate density q will not include the target density as a member, the choice of variational objective is important in determining the properties of how q approximates the target density [41]. The standard variational objective corresponding to $\mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p]$ strongly penalises regions in X where $\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \ll 1$, therefore the approximate densities fitted using this objective tend to be underdispersed compared to the target density, and in the case of target densities with multiple separated modes fitted with a unimodal approximate density, the approximate density will tend to fit only one mode well (with fits to the different modes corresponding to different local optima in the objective). Conversely using the reversed KL divergence $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$ as the variational objective penalises approximate densities where $\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \ll 1$ in regions with significant mass under the target density, therefore the approximate densities fitted using this objective tend to be overdispersed compared to the target density, and in the case of multimodal target densities, the approximate densities will tend to 'cover' multiple modes. Using a variational objective corresponding to a Rényi divergence with $0 < \alpha < 1$, allows interpolating between these two behaviours (with α close to one favouring underdispersed approximate densities similar to $\mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p]$, with the solutions becoming increasingly dispersed as α becomes lower).

Figure C.2 gives examples of normal approximate densities fitted to a bimodal target with three variational objectives to illustrate the ef-

fect of the different objectives on the fitted approximation. In Figure C.2a the approximate density q was fitted by minimising $\mathbb{D}^{\lambda}_{\mathrm{KL}}[p \,\|\, q]$, the resulting q putting mass on both modes in the target (and significant mass on the region of low density between the two target modes). The approximate density q in Figure C.2c was instead fitted by minimising $\mathbb{D}^{\lambda}_{\mathrm{KL}}[q \,\|\, p]$, with the result that q concentrates its mass around one of the modes. Finally Figure C.2b shows an approximate density fitted by minimising the Rényi divergence (C.14) with $\alpha = \frac{1}{2}$ for which $\mathbb{D}^{\lambda}_{\alpha}[p \,\|\, q] = \mathbb{D}^{\lambda}_{\alpha}[q \,\|\, p]$ and which interpolates between the behaviours of the two objectives used in Figures C.2a and C.2c. The approximate density here is less dispersed than in the $\mathbb{D}^{\lambda}_{\mathrm{KL}}[p \,\|\, q]$ case, but still places more mass on the minor mode than the $\mathbb{D}^{\lambda}_{\mathrm{KL}}[q \,\|\, p]$ case.

Once the variational objective has been defined, it still remains to choose the family of the approximate density q and optimisation scheme. A very common choice is to use an approximate density in the *mean-field variational family*; this assumes that the variables the target density is defined on can be grouped in to a set of mutually independent vectors $\{\mathbf{x}_i\}_{i \in I}$ and so the approximate density can be factorised as

$$q(\boldsymbol{x}) = \prod_{i \in I} q_i(\boldsymbol{x}_i). \tag{C.16}$$

This assumption can significantly reduce the computational demands of variational inference and facilitates simple evaluation of the approximate marginal density q_i of each variable group once fitted. However the mutual independence assumption prevents the approximate density q from being able to represent any of the dependencies between the variable groups in the target density. The early development of variational inference was largely based around mean-field family approximations [208, 237], with the naming arising from its origins in *mean-field theory*, used to study the behaviour of systems such as the Ising spin model in statistical physics [205]. Despite the limitations in representational capacity imposed by the independence assumption, because of its computational tractability mean-field methods remain very popular [42], with mean-field approximations allowing use of a particularly simple algorithm for optimising the standard variational objective (C.9), *co-ordinate ascent variational inference* [41, 42].

A more recent alternative to traditional mean-field variational methods, is to assume a fixed parametric form for the approximate density, i.e.

$q(\boldsymbol{x}) = q_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $q_{\boldsymbol{\theta}}$ is a density with respect to the measure μ of a fixed parametric family with a vector of parameters $\boldsymbol{\theta}$ [115, 144, 200, 219, 235]. Under this parametric assumption, rather than a variational optimisation problem we can now consider the variational objective functional $\mathbb{L}[q]$ as instead a function of the parameters $\ell(\boldsymbol{\theta}) = \mathbb{L}[q_{\boldsymbol{\theta}}]$. For the standard variational objective (C.9) we have that

$$\ell(\boldsymbol{\theta}) = \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \log \frac{\tilde{p}(\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{x})} \mu(\mathrm{d}\boldsymbol{x}). \tag{C.17}$$

Using the identities that for any $q_{\boldsymbol{\theta}}$ which is differentiable with respect to $\boldsymbol{\theta}$ we have that

$$\frac{\partial q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} = q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \tag{C.18}$$

$$\text{and} \qquad \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \mu(\mathrm{d}\boldsymbol{x}) = 0, \tag{C.19}$$

the gradient of (C.17) with respect $\boldsymbol{\theta}$ can be expressed as

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \log \frac{\tilde{p}(\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{x})} \mu(\mathrm{d}\boldsymbol{x}). \tag{C.20}$$

Typically both of the integrals in (C.17) and (C.20) defining the variational objective and its gradient will not have analytic solutions. However both take the forms of expectations of a random vector with distribution defined by the approximate density $q_{\boldsymbol{\theta}}$. If we can generate independent samples from $q_{\boldsymbol{\theta}}$ we can therefore form unbiased Monte Carlo estimates of the objective and its gradient.

The unbiased gradient estimates can then be used in a stochastic gradient ascent method [225] to maximise $\ell(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. This basic framework is applicable to a much broader class of target distributions than the previously discussed variational inference approaches, requiring only that we can pointwise evaluate a, potentially unnormalised, density function $\tilde{p}$ for the target distribution. Likewise the only restrictions on the approximating distribution are that we can evaluate a density function $q_{\boldsymbol{\theta}}$ which is differentiable with respect to its parameters $\boldsymbol{\theta}$ and that we can generate independent samples from this distribution to form the Monte Carlo estimates.

For target distributions on a real-valued space, a simple choice for $q_{\boldsymbol{\theta}}$ meeting these requirements is a multivariate normal density $q_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma)$ with the mean $\boldsymbol{\mu}$ and covariance Σ forming the parameters

$\boldsymbol{\theta}$ maximised with respect to. Using a diagonal covariance $\boldsymbol{\Sigma}$ would correspond to a mean-field factorisation assumption for the approximate density, however we can also use more general covariances including a full dense matrix allowing for arbitrary covariance structure in the approximate density, or a sparse covariance matrix which exploits known conditional independencies in the target distribution.

Although appealingly simple and flexible, the basic scheme as described so far has a major pitfall which is that the variance of the gradients estimate computed by forming the obvious Monte Carlo estimator from (C.20) typically has a very high variance for the complex target distributions of interest. This necessitates either taking a very large number of Monte Carlo samples to estimate the gradient for each parameter update with sufficient accuracy or taking very small gradient steps to allow stable optimisation. Therefore practical schemes based on this idea generally require the use of variance reduction methods to estimate the variational objective parameter gradient.

The *black box variational inference* (BBVI) algorithm of [219] proposes using two forms of variance reduction to compute more efficient gradient estimates - Rao-Blackwellisation and control variates. The Rao-Blackwellisation method relies on being able to decompose the approximate density $q_{\boldsymbol{\theta}}$ into a product of factors with per factor variational parameters and is so restricted to cases such as mean-field approximations where this is the case. The control variate method is more general and can be applied to non mean-field approximate densities.

An alternative variance reduction approach is proposed in the ADVI algorithm of [144] which instead uses a reparameterisation of the approximating distribution to produce a lower variance gradient estimator for a more restricted class of target distributions which have a differentiable density with respect to the Lebesgue measure. It is assumed that the samples from the approximating distribution can be generating using a transform sampling method, more specifically that there exists a differentiable bijective function $\mathbf{g}_{\boldsymbol{\theta}} : U \to X$ and a distribution R on U which has a density ρ which does not depend on $\boldsymbol{\theta}$ such that

$$q_{\boldsymbol{\theta}}(\boldsymbol{x}) = \rho\left(\mathbf{g}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x})\right)\left|\frac{\partial \mathbf{g}_{\boldsymbol{\theta}}^{-1}}{\partial \boldsymbol{x}}\right|. \tag{C.21}$$

In this case by the change of variables formula (1.22) discussed in Chapter 1 if $\boldsymbol{u}$ is an independent sample from R then $\boldsymbol{x} = \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{u})$ will be an independent sample from the approximate distribution with density $q_{\boldsymbol{\theta}}$. This transformation can be used to reparameterise the variational objective integral as

$$\ell(\boldsymbol{\theta}) = \int_U \rho(\boldsymbol{u}) \Big(\log(\tilde{p} \circ \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{u})) + \log \left| \frac{\partial \mathbf{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{u}} \right| - \log \rho(\boldsymbol{u}) \Big) \, \mathrm{d}\boldsymbol{u} \tag{C.22}$$

with a corresponding gradient expression

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \int_U \rho(\boldsymbol{u}) \left(\frac{\partial}{\partial \boldsymbol{\theta}} \log(\tilde{p} \circ \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{u})) + \frac{\partial}{\partial \boldsymbol{\theta}} \log \left| \frac{\partial \mathbf{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{u}} \right| \right) \mathrm{d}\boldsymbol{u}. \tag{C.23}$$

As suggested by the name ADVI uses automatic differentiation to calculate the gradient expression inside the parentheses in (C.23), with importantly this requiring propagation of derivatives through the target density function $\tilde{p}$ as well as the transformation $\mathbf{g}_{\boldsymbol{\theta}}$. To form a Monte Carlo estimate of the gradient using this reparameterisation we therefore require the target model density to be differentiable and so it is less general than the method used in BBVI, however as shown empirically in [144] the resulting gradient estimator will tend to be significantly more efficient requiring fewer samples to bring the variance to a reasonable level, with often gradients computed with a single sample being sufficient for stable optimisation.

D GENERATIVE MODEL PARAMETERISATION

As the inference methods we propose in Chapter 4 work in the generator input space, we can reparameterise the input space to endow it with a favourable form for inference. For example we will generally reparametrise input variables with bounded support to transformed variables with unbounded support, for example reparameterising in terms of the logarithm of a strictly positive variable. In general working with unbounded variables will simplify MCMC inference by preventing the need to check transitions respect bounding constraints. Probabilistic programming frameworks such as Stan [55] and PyMC3 [236] use a range of such transformations when performing inference.

As well as transforming to variables with unbounded support, another useful heuristic is to parameterise a model as far as possible in terms of inputs variables which have unit variance. Three examples of potentially suitable distributions with unit variance and unbounded support are the standard normal $\mathcal{N}(0, 1)$, inverse hyperbolic cosine (or hyperbolic secant) distribution $\text{InvCosh}(0, 1)$ and the logistic distribution $\text{Logistic}(0, \sqrt{3}/\pi)$. The densities for all three shown for comparison in Figure D.1 and Table D.1 shows reparameterisations for some common distributions in terms of variables distributed according to these standard densities. Normalising the marginal variances of variables in ρ typically makes it easier to choose an appropriate scale parameters for the MCMC transitions.

Also note that although we motivated our definition of the random inputs $\mathbf{u}$ in Chapter 4 by saying it could be constructed by tracking all the draws from a random number generator, in general we will not want to parameterise $\mathbf{u}$ in terms of low-level uniform draws, but instead use the output of higher-level functions for producing samples from standard densities using the transform and rejection sampling methods discussed in Chapter 2. This is important as if for example we defined as inputs the uniform draws used in the rejection sampling routines used to generate Gamma random variables, we would both require dealing

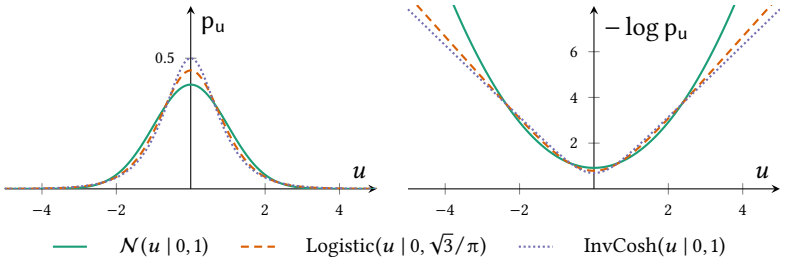


Figure D.1.: Unit variance densities with unbounded support.

Original factor	Reparametrisation
$\mathcal{N}(\mu, \sigma^2)$ → v	$\mathcal{N}(0,1)$ → u, $\mu + \sigma u$ → v
$\text{LogNorm}(\mu, \sigma^2)$ → v	$\mathcal{N}(0,1)$ → u, $\exp(\mu + \sigma u)$ → v
$\text{Exp}(\lambda)$ → v	$\text{Logistic}\left(0, \frac{\sqrt{3}}{\pi}\right)$ → u, $\frac{1}{\lambda}\log\left(1+\exp\left(\frac{\pi u}{\sqrt{3}}\right)\right)$ → v
$\mathcal{U}(a, b)$ → v	$\text{Logistic}\left(0, \frac{\sqrt{3}}{\pi}\right)$ → u, $a + (b-a)\left(1+\exp\left(\frac{\pi u}{\sqrt{3}}\right)\right)^{-1}$ → v
$C_{\geq 0}(\gamma)$ → v	$\text{InvCosh}(0,1)$ → u, $\gamma \exp\left(\frac{\pi u}{2}\right)$ → v

Table D.1.: Reparameterisations of random variables with some common parametric distributions as deterministic transformations of unit-variance unbounded support random variables.

with the complications involved with generators using variable numbers of random inputs as described in Chapter 4 and also have that $\mathbf{g}_\mathbf{x}$ would be non-differentiable with respect to the rejection sampling inputs even if the all of the operations performed with the Gamma variable to produce the generated outputs are themselves differentiable. If we instead use the generated Gamma variable itself as the input by including an appropriate Gamma density factor in ρ we side step these issues.

In some cases using the outputs of higher-level random number generator routines as the input variables will introduce dependencies between the variables in the input density ρ. In particular if u_i is drawn from a distribution with parameters depending on one or more previous random inputs $\{\mathrm{u}_j\}_{j\in\mathcal{J}}$, then an appropriate conditional density factor on u_i given $\{\mathrm{u}_j\}_{j\in\mathcal{J}}$ will need to be included in ρ. By using alternative para-

meterisations it may be possible to avoid introducing such dependencies; for example a random input v_i generated from a normal distribution with mean μ and standard deviation σ which depend on previous random inputs $\{\mathsf{u}_j\}_{j \in \mathcal{J}}$ can instead be parameterised in terms of an independent random variable u_i distributed with a standard normal density $\mathcal{N}(0, 1)$ and v_i computed as $\sigma \mathsf{u}_i + \mu$ in the generator.

Such *non-centred parameterisations* [45, 204, 214] are available for example for all location-scale family distributions. The reparametrisation of the Gaussian VAE decoder discussed in Chapter 4 also uses this same identity, and the term 'reparameterisation trick' is often used in the machine learning literature to describe this idea [139]. Whether it is necessarily helpful to remove dependencies in ρ like this for the methods discussed in Chapter 4 is an open question and will likely be model specific; it has previously been found that non-centred parameterisations can be beneficial when performing MCMC inference in hierarchical models when the unobserved variables are only weakly identified by observations [36, 203, 204].

E BOLTZMANN MACHINE RELAXATIONS

In this appendix we describe the continuous relaxation of the Boltzmann machine distribution introduced in [266] and derive some basic results which we use in the experiments with this model in Chapter 5.

The *Boltzmann machine distribution* on $\mathbf{s} \in \{-1, +1\}^{D_B} = S$ is defined

$$\begin{aligned} \mathrm{P}_{\mathbf{s}}(\boldsymbol{s}) &= \frac{1}{Z_B} \exp\left(\frac{1}{2}\boldsymbol{s}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^{\mathsf{T}}\boldsymbol{b}\right) \\ Z_B &= \sum_{\boldsymbol{s}\in S}\left(\exp\left(\frac{1}{2}\boldsymbol{s}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^{\mathsf{T}}\boldsymbol{b}\right)\right). \end{aligned} \tag{E.1}$$

We introduce an auxiliary real-valued vector random variable $\mathbf{x} \in \mathbb{R}^D$ with a Gaussian conditional distribution

$$\mathrm{p}_{\mathbf{x}|\mathbf{s}}(\boldsymbol{x} \mid \boldsymbol{s}) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{s}\right)^{\mathsf{T}}\left(\boldsymbol{x} - \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{s}\right)\right) \tag{E.2}$$

with $\boldsymbol{Q}$ a $D_B \times D$ matrix such that $\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}} = \boldsymbol{W} + \boldsymbol{D}$ for a diagonal $\boldsymbol{D}$ such that $\boldsymbol{W} + \boldsymbol{D} \succeq 0$. In our experiments, based on the observation in [266] that minimising the maximum eigenvalue of $\boldsymbol{W} + \boldsymbol{D}$ decreases the maximal separation between the Gaussian components in the relaxation, we set $\boldsymbol{D}$ as the solution to the semi-definite programme

$$\min_{\boldsymbol{D}}(\lambda_{\text{MAX}}(\boldsymbol{W} + \boldsymbol{D})) : \boldsymbol{W} + \boldsymbol{D} \succeq 0 \tag{E.3}$$

where λ_{MAX} denotes the maximal eigenvalue. In general the optimised $\boldsymbol{W} + \boldsymbol{D}$ lies on the semi-definite cone and so has rank less than D_B hence a $\boldsymbol{Q}$ can be found such that $D < D_B$.

The resulting joint distribution on $(\mathbf{x}, \mathbf{s})$ is

$$\begin{aligned} &\mathrm{p}_{\mathbf{x},\mathbf{s}}(\boldsymbol{x}, \boldsymbol{s}) = \\ &\quad \frac{1}{(2\pi)^{D/2} Z_B} \exp\left(-\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{s}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{x} - \frac{1}{2}\boldsymbol{s}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}}\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^{\mathsf{T}}\boldsymbol{b}\right) \end{aligned} \tag{E.4}$$

Using that $\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}} = \boldsymbol{W} + \boldsymbol{D}$ this can be simplified to

$$\mathrm{p}_{\mathbf{x},\mathbf{s}}(\boldsymbol{x}, \boldsymbol{s}) = \frac{1}{(2\pi)^{D/2} Z_B} \exp\Big(-\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{s}^{\mathsf{T}}(\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}) - \frac{1}{2}\boldsymbol{s}^{\mathsf{T}}\boldsymbol{D}\boldsymbol{s}\Big) \tag{E.5}$$

$$= \frac{\exp\Big(-\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}\Big)}{(2\pi)^{D/2} Z_B \exp\Big(\frac{1}{2}\operatorname{Tr}(\boldsymbol{D})\Big)} \prod_{i=1}^{D_B} \Big(\exp\Big(s_i\Big(\boldsymbol{q}_i^{\mathsf{T}}\boldsymbol{x} + b_i\Big)\Big)\Big), \tag{E.6}$$

where $\left\{\boldsymbol{q}_i^{\mathsf{T}}\right\}_{i=1}^{D_B}$ are the D_B rows of $\boldsymbol{Q}$. We can marginalise over the binary state $\mathbf{s}$ as each s_i is conditionally independent of all the others given $\mathbf{x}$ in the joint distribution. This gives the *Boltzmann machine relaxation* density on $\mathbf{x}$

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \frac{2^{D_B} \exp\Big(-\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}\Big)}{(2\pi)^{D/2} Z_B \exp\Big(\frac{1}{2}\operatorname{Tr}(\boldsymbol{D})\Big)} \prod_{i=1}^{D_B} \Big(\cosh\Big(\boldsymbol{q}_i^{\mathsf{T}}\boldsymbol{x} + b_i\Big)\Big), \tag{E.7}$$

which is a structured Gaussian mixture density with 2^{D_B} components. If we define $\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{Z}\exp(-\phi(\boldsymbol{x}))$ with

$$\phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x} - \sum_{i=1}^{D_B} \Big(\log\cosh\Big(\boldsymbol{q}_i^{\mathsf{T}}\boldsymbol{x} + b_i\Big)\Big), \tag{E.8}$$

then the normalisation constant Z of the relaxation density can be related to the normalising constant of the corresponding Boltzmann machine distribution by

$$\log Z = \log Z_B + \frac{1}{2}\operatorname{Tr}(\boldsymbol{D}) + \frac{D}{2}\log(2\pi) - D_B \log 2. \tag{E.9}$$

It can also be shown that the first and second moments of the relaxation distribution are related to the first and second moments of the corresponding Boltzmann machine distribution by

$$\begin{aligned} \mathbb{E}[\mathbf{x}] &= \int_X \boldsymbol{x} \sum_{\boldsymbol{s}\in S} (\mathrm{p}_{\mathbf{x}|\mathbf{s}}(\boldsymbol{x} \mid \boldsymbol{s})\, \mathrm{P}_{\mathbf{s}}(\boldsymbol{s}))\, \mathrm{d}\boldsymbol{x} \\ &= \sum_{\boldsymbol{s}\in S} \left(\int_X \boldsymbol{x}\, \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{s}, \mathbf{I}\right) \mathrm{d}\boldsymbol{x}\, \mathrm{P}_{\mathbf{s}}(\boldsymbol{s}) \right) \\ &= \mathbb{E}\left[\boldsymbol{Q}^{\mathsf{T}}\mathbf{s}\right] = \boldsymbol{Q}^{\mathsf{T}}\mathbb{E}[\mathbf{s}], \end{aligned} \tag{E.10}$$

$$\begin{aligned} \mathbb{E}\left[\mathbf{x}\mathbf{x}^{\mathsf{T}}\right] &= \sum_{\boldsymbol{s}\in S} \left(\int_X \boldsymbol{x}\boldsymbol{x}^{\mathsf{T}}\, \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{s}, \mathbf{I}\right) \mathrm{d}\boldsymbol{x}\, \mathrm{P}_{\mathbf{s}}(\boldsymbol{s}) \right) \\ &= \mathbb{E}\left[\boldsymbol{Q}^{\mathsf{T}}\mathbf{s}\mathbf{s}\boldsymbol{Q} + \mathbf{I}\right] = \boldsymbol{Q}^{\mathsf{T}}\mathbb{E}\left[\mathbf{s}\mathbf{s}^{\mathsf{T}}\right]\boldsymbol{Q} + \mathbf{I}. \end{aligned} \tag{E.11}$$

BIBLIOGRAPHY

[1] David H. Ackley, Geoffrey E. Hinton and Terrence J. Sejnowski. 'A learning algorithm for Boltzmann machines'. In: *Cognitive science* 9.1 (1985), pp. 147–169.

[2] Murray Aitkin. 'Posterior Bayes factors'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1991), pp. 111–142.

[3] Ijaz Akhter and Michael J. Black. 'Pose-Conditioned Joint Angle Limits for 3D Human Pose Reconstruction'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[4] David Allingham, Robert A.R. King and Kerrie L. Mengersen. 'Bayesian estimation of quantile distributions'. In: *Statistics and Computing* 19.2 (2009), pp. 189–201.

[5] Shun-Ichi Amari. 'Differential geometry of curved exponential families-curvatures and information loss'. In: *The Annals of Statistics* (1982), pp. 357–385.

[6] Hans C. Andersen. 'RATTLE: A velocity version of the SHAKE algorithm for molecular dynamics calculations'. In: *Journal of Computational Physics* (1983).

[7] Christophe Andrieu, Arnaud Doucet and Roman Holenstein. 'Particle Markov chain Monte Carlo methods'. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3 (2010), pp. 269–342.

[8] Christophe Andrieu and Gareth O. Roberts. 'The pseudo-marginal approach for efficient Monte Carlo computations'. In: *The Annals of Statistics* (2009).

[9] Christophe Andrieu and Johannes Thoms. 'A tutorial on adaptive MCMC'. In: *Statistics and computing* 18.4 (2008), pp. 343–373.

[10] Martín Arjovsky and Léon Bottou. 'Towards Principled Methods for Training Generative Adversarial Networks'. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[11] IEEE Standards Association. 'Standard for Floating-Point Arithmetic'. In: *IEEE 754-2008* (2008).

[12] Chris P. Barnes, Sarah Filippi, Michael P. H. Stumpf and Thomas Thorne. 'Considerate approaches to constructing summary statistics for ABC model selection'. In: *Statistics and Computing* 22.6 (2012), pp. 1181–1197. URL: http://dx.doi.org/10.1007/s11222-012-9335-7.

[13] Alessandro Barp, Francois-Xavier Briol, Anthony D Kennedy and Mark Girolami. 'Geometry and Dynamics for Markov Chain Monte Carlo'. In: *arXiv preprint 1705.02891* (2017).

[14] Eric Barth, Krzysztof Kuczera, Benedict Leimkuhler and Robert D. Skeel. 'Algorithms for constrained molecular dynamics'. In: *Journal of computational chemistry* (1995).

[15] Simon Barthelmé and Nicolas Chopin. 'Expectation propagation for likelihood-free inference'. In: *Journal of the American Statistical Association* 109.505 (2014), pp. 315–333.

[16] Friedrich L. Bauer. 'Computational graphs and rounding error'. In: *SIAM Journal on Numerical Analysis* 11.1 (1974), pp. 87–96.

[17] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul and Jeffrey Mark Siskind. 'Automatic differentiation in machine learning: a survey'. In: *arXiv preprint 1502.05767* (2015).

[18] Mark A. Beaumont. 'Estimation of population growth or decline in genetically monitored populations'. In: *Genetics* 164.3 (2003), pp. 1139–1160.

[19] Mark A. Beaumont, Wenyang Zhang and David J Balding. 'Approximate Bayesian computation in population genetics'. In: *Genetics* (2002).

[20] Mark A. Beaumont, Jean-Marie Cornuet, Jean-Michel Marin and Christian P Robert. 'Adaptive approximate Bayesian computation'. In: *Biometrika* 96.4 (2009), pp. 983–990.

[21] L. M. Beda, L. N. Korolev, N. V. Sukkikh and T. S. Frolova. *Programs for automatic differentiation for the machine BESM*. Technical Report. (In Russian). Moscow, USSR: Institute for Precise Mechanics and Computation Techniques, Academy of Science, 1959.

[22] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn and Kurt Smith. 'Cython: The best of both worlds'. In: *Computing in Science & Engineering* 13.2 (2011), pp. 31–39.

[23] Gundula Behrens, Nial Friel and Merrilee Hurn. 'Tuning tempered transitions'. In: *Statistics and computing* (2012).

[24] José M. Bernardo and Adrian F.M. Smith. *Bayesian Theory*. Wiley Series in Probability and Statistics. Wiley, 2009.

[25] Jarle Berntsen, Terje O. Espelid and Alan Genz. 'An adaptive algorithm for the approximate calculation of multiple integrals'. In: *ACM Transactions on Mathematical Software (TOMS)* 17.4 (1991), pp. 437–451.

[26] Alexandros Beskos, Natesh Pillai, Gareth Roberts, Jesus-Maria Sanz-Serna and Andrew Stuart. 'Optimal tuning of the hybrid Monte Carlo algorithm'. In: *Bernoulli* 19.5A (2013), pp. 1501–1534.

[27] Alexandros Beskos, Gareth Roberts, Alexandre Thiery and Natesh Pillai. 'Asymptotic Analysis of the Random-Walk Metropolis Algorithm on Ridged Densities'. In: *arXiv preprint 1510.02577* (2015).

[28] Michael Betancourt. 'A general metric for Riemannian manifold Hamiltonian Monte Carlo'. In: *Geometric science of information*. Springer, 2013.

[29] Michael Betancourt. 'Generalizing the no-U-turn sampler to Riemannian manifolds'. In: *arXiv preprint 1304.1920* (2013).

[30] Michael Betancourt. 'Adiabatic Monte Carlo'. In: *arXiv preprint 1405.3489* (2014).

[31] Michael Betancourt. 'The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling'. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[32] Michael Betancourt. 'Identifying the Optimal Integration Time in Hamiltonian Monte Carlo'. In: *arXiv preprint 1601.00225* (2016).

[33] Michael Betancourt. 'A Conceptual Introduction to Hamiltonian Monte Carlo'. In: *arXiv preprint 1701.02434* (2017).

[34] Michael Betancourt. 'The Convergence of Markov chain Monte Carlo Methods: From the Metropolis method to Hamiltonian Monte Carlo'. In: *arXiv preprint 1706.01520* (2017).

[35] Michael Betancourt, Simon Byrne and Mark Girolami. 'Optimizing the integrator step size for Hamiltonian Monte Carlo'. In: *arXiv preprint 1411.6669* (2014).

[36] Michael Betancourt and Mark Girolami. 'Hamiltonian Monte Carlo for hierarchical models'. In: *Current trends in Bayesian methodology with applications* 79 (2015), p. 30.

[37] Michael Betancourt, Simon Byrne, Sam Livingstone and Mark Girolami. 'The geometric foundations of Hamiltonian Monte Carlo'. In: *Bernoulli* 23.4A (2017), pp. 2257–2298.

[38] Joris Bierkens. 'Non-reversible Metropolis–Hastings'. In: *Statistics and Computing* 26.6 (2016), pp. 1213–1228.

[39] Maxence Bigerelle, D Najjar, Benjamin Fournier, Nicolas Rupin and Alain Iost. 'Application of lambda distributions and bootstrap analysis to the prediction of fatigue lifetime and confidence intervals'. In: *International Journal of Fatigue* 28.3 (2006), pp. 223–236.

[40] George D. Birkhoff. 'Proof of the ergodic theorem'. In: *Proceedings of the National Academy of Sciences* 17.12 (1931), pp. 656–660.

[41] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN: 9780387310732.

[42] David M. Blei, Alp Kucukelbir and Jon D. McAuliffe. 'Variational inference: A review for statisticians'. In: *Journal of the American Statistical Association* just-accepted (2017).

[43] Michael G.B. Blum. 'Approximate Bayesian computation: a nonparametric perspective'. In: *Journal of the American Statistical Association* 105.491 (2010), pp. 1178–1187.

[44] Michael G.B. Blum, Matthew A. Nunes, Dennis Prangle and Scott A Sisson. 'A comparative review of dimension reduction methods in approximate Bayesian computation'. In: *Statistical Science* 28.2 (2013), pp. 189–208.

[45] Georges Bonnet. 'Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire'. In: *Annals of Telecommunications* 19.9 (1964), pp. 203–220.

[46] Massimiliano Bonomi, Alessandro Barducci and Michele Parrinello. 'Reconstructing the equilibrium Boltzmann distribution from well-tempered metadynamics'. In: *Journal of computational chemistry* (2009).

[47] Luke Bornn, Natesh S. Pillai, Aaron Smith and Dawn Woodard. 'The use of a single pseudo-sample in approximate Bayesian computation'. In: *Statistics and Computing* 27.3 (2017), pp. 583–590.

[48] George E.P. Box and Mervin E. Muller. 'A note on the generation of random normal deviates'. In: *The Annals of Mathematical Statistics* 29.2 (1958), pp. 610–611.

[49] Marcus A. Brubaker, Mathieu Salzmann and Raquel Urtasun. 'A Family of MCMC Methods on Implicitly Defined Manifolds.' In: *International Conference on Artificial Intelligence and Statistics*. 2012.

[50] Wray L. Buntine. 'Operations for learning with graphical models'. In: *Journal of artificial intelligence research* (1994).

[51] Yuri Burda, Roger Grosse and Ruslan Salakhutdinov. 'Importance weighted autoencoders'. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016.

[52] Simon Byrne and Mark Girolami. 'Geodesic Monte Carlo on embedded manifolds'. In: *Scandinavian Journal of Statistics* (2013).

[53] David Carlson, Patrick Stinson, Ari Pakman and Liam Paninski. 'Partition Functions from Rao-Blackwellized Tempered Sampling'. In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016.

[54] Bob Carpenter, Matthew D. Hoffman, Marcus Brubaker, Daniel Lee, Peter Li and Michael Betancourt. 'The Stan Math Library: Reverse-Mode Automatic Differentiation in C++'. In: *arXiv preprint 1509.07164* (2015).

[55] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Michael A. Brubaker, Jiqiang Guo, Peter Li and Allen Riddell. 'Stan: A probabilistic programming language'. In: *Journal of Statistical Software* (2016).

[56] George Casella and Christian P. Robert. 'Rao-Blackwellisation of sampling schemes'. In: *Biometrika* 83.1 (1996), pp. 81–94. DOI: 10.1093/biomet/83.1.81. URL: +http://dx.doi.org/10.1093/biomet/83.1.81.

[57] Kung-Sik Chan. 'Asymptotic behavior of the Gibbs sampler'. In: *Journal of the American Statistical Association* 88.421 (1993), pp. 320–326.

[58] Kung Sik Chan and Charles J. Geyer. 'Discussion: Markov chains for exploring posterior distributions'. In: *The Annals of Statistics* 22.4 (1994), pp. 1747–1758.

[59] Ming-Hui Chen and Bruce Schmeiser. 'Toward black-box sampling: A random-direction interior-point Markov chain approach'. In: *Journal of Computational and Graphical Statistics* 7.1 (1998), pp. 1–22.

[60] Tianqi Chen, Emily Fox and Carlos Guestrin. 'Stochastic Gradient Hamiltonian Monte Carlo'. In: *Proceedings of the 31st International Conference on Machine Learning*. 2014.

[61] Nicolas Chopin and Sumeetpal S. Singh. 'On particle Gibbs sampling'. In: *Bernoulli* 21.3 (2015), pp. 1855–1883.

[62] Timothy M. Christensen, Stan Hurn and Kenneth A. Lindsay. 'The devil is in the detail: hints for practical optimisation'. In: *Economic Analysis and Policy* 38.2 (2008), pp. 345–368.

[63] Merlise A. Clyde and Edwin S. Iversen. 'Bayesian model averaging in the M-open framework'. In: *Bayesian Theory and Applications* (2013).

[64] Charles J. Corrado. 'Option pricing based on the generalized lambda distribution'. In: *Journal of Futures Markets* 21.3 (2001), pp. 213–236.

[65] Richard T. Cox. 'Probability, frequency and reasonable expectation'. In: *American Journal of Physics* 14.1 (1946), pp. 1–13. URL: http://dx.doi.org/10.1119/1.1990764.

[66] Richard T. Cox. 'The algebra of probable inference'. In: *American Journal of Physics* 31.1 (1963), pp. 66–67. URL: http://dx.doi.org/10.1119/1.1969248.

[67] Harald Cramér. *Mathematical Methods of Statistics*. Princeton University Press, 1946.

[68] Johan Dahlin, Fredrik Lindsten, Joel Kronander and Thomas B. Schön. 'Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables'. In: *arXiv preprint 1511.05483* (2015).

[69] Paul Damien, Jon Wakefield and Stephen Walker. 'Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables'. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.2 (1999), pp. 331–344.

[70] Philip J. Davis and Philip Rabinowitz. *Numerical Integration*. Blaisdell Publishing Company, 1967.

[71] Nando De Freitas, Pedro Højen-Sørensen, Michael I. Jordan and Stuart Russell. 'Variational MCMC'. In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. 2001.

[72] Pierre Del Moral, Arnaud Doucet and Ajay Jasra. 'An adaptive sequential Monte Carlo method for approximate Bayesian computation'. In: *Statistics and Computing* 22.5 (2012), pp. 1009–1020.

[73] George Deligiannidis, Arnaud Doucet, Michael K Pitt and Robert Kohn. 'The Correlated Pseudo-Marginal Method'. In: *arXiv preprint 1511.04992* (2015).

[74] Persi Diaconis, Susan Holmes and Radford M. Neal. 'Analysis of a nonreversible Markov chain sampler'. In: *Annals of Applied Probability* (2000), pp. 726–752.

[75] Persi Diaconis, Susan Holmes and Mehrdad Shahshahani. 'Sampling from a manifold'. In: *Advances in Modern Statistical Theory and Applications*. Institute of Mathematical Statistics, 2013, pp. 102–125.

[76] Adji B. Dieng, Dustin Tran, Rajesh Ranganath, John Paisley and David M. Blei. 'The χ-Divergence for Approximate Inference'. In: *arXiv preprint 1611.00328* (2016).

[77] Peter J. Diggle and Richard J. Gratton. 'Monte Carlo methods of inference for implicit statistical models'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1984), pp. 193–227.

[78] Arnaud Doucet, Nando de Freitas and Neil Gordon, eds. *Sequential Monte Carlo Methods in Practice*. Information Science and Statistics. Springer New York, 2001. ISBN: 9780387951461.

[79] Arnaud Doucet, M.K. Pitt, George Deligiannidis and Robert Kohn. 'Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator'. In: *Biometrika* 102.2 (2015), pp. 295–313.

[80] Oliver B. Downs, David J.C. MacKay and Daniel D. Lee. 'The nonnegative Boltzmann machine'. In: *Advances in Neural Information Processing Systems*. 2000, pp. 428–434.

[81] Simon Duane, Anthony D. Kennedy, Brian J. Pendleton and Duncan Roweth. 'Hybrid Monte Carlo'. In: *Physics Letters B* (1987).

[82] Gintare Karolina Dziugaite, Daniel M. Roy and Zoubin Ghahramani. 'Training generative neural networks via maximum mean discrepancy optimization'. In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2015, pp. 258–267.

[83] Robert G. Edwards and Alan D. Sokal. 'Generalization of the Fortuin–Kasteleyn–Swendsen–Wang representation and Monte Carlo algorithm'. In: *Physical review D* 38.6 (1988), p. 2009.

[84] Vassiliy A. Epanechnikov. 'Non-parametric estimation of a multivariate probability density'. In: *Theory of Probability & Its Applications* 14.1 (1969), pp. 153–158.

[85] Tim van Erven and Peter Harremos. 'Rényi divergence and Kullback-Leibler divergence'. In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 3797–3820.

[86] Herbert Federer. *Geometric measure theory*. Springer, 1969.

[87] Maurizio Filippone and Mark Girolami. 'Pseudo-marginal Bayesian inference for Gaussian processes'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2214–2226.

[88] Bruno de Finetti. 'Foresight: its logical laws, its subjective sources'. In: *Studies in Subjective Probability*. Ed. by H. E. Kyburg. English translation of original 1937 French article *La Prévision: ses lois logiques, ses sources subjectives*. Springer, 1992, pp. 134–174.

[89] Marshall Freimer, Georgia Kollia, Govind S. Mudholkar and C. Thomas Lin. 'A study of the generalized Tukey lambda family'. In: *Communications in Statistics-Theory and Methods* 17.10 (1988), pp. 3547–3567.

[90] Brendan J. Frey. 'Extending factor graphs so as to unify directed and undirected graphical models'. In: *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 2002, pp. 257–264.

[91] Brendan J. Frey, Frank R. Kschischang, Hans-Andrea Loeliger and Niclas Wiberg. 'Factor graphs and algorithms'. In: *Proceedings of the 35th Annual Allerton Conference on Communication Control and Computing*. 1997.

[92] Yun-Xin Fu and Wen-Hsiung Li. 'Estimating the age of the common ancestor of a sample of DNA sequences.' In: *Molecular biology and evolution* 14.2 (1997), pp. 195–199.

[93] Alan E. Gelfand and Adrian F.M. Smith. 'Sampling-based approaches to calculating marginal densities'. In: *Journal of the American Statistical Association* (1990).

[94] Andrew Gelman, Walter R. Gilks and Gareth O. Roberts. 'Weak convergence and optimal scaling of random walk Metropolis algorithms'. In: *The annals of applied probability* 7.1 (1997), pp. 110–120.

[95] Andrew Gelman and Christian Hennig. 'Beyond subjective and objective in statistics'. In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* (). ISSN: 1467-985X. DOI: 10.1111/rssa.12276. URL: http://dx.doi.org/10.1111/rssa.12276.

[96] Andrew Gelman and Jennifer Hill. 'Data analysis using regression and multilevel/hierarchical models'. In: Cambridge University Press, 2006. Chap. 12: Multilevel Linear Models: the Basics.

[97] Andrew Gelman and Xiao-Li Meng. 'Simulating normalizing constants: From importance sampling to bridge sampling to path sampling'. In: *Statistical science* (1998).

[98] Andrew Gelman and Donald B. Rubin. 'Inference from iterative simulation using multiple sequences'. In: *Statistical science* (1992), pp. 457–472.

[99] Andrew Gelman, Daniel Simpson and Michael Betancourt. 'The prior can generally only be understood in the context of the likelihood'. In: *arXiv preprint 1708.07487* (2017).

[100] Stuart Geman and Donald Geman. 'Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images'. In: *IEEE Transactions on pattern analysis and machine intelligence* (1984).

[101] Charles J. Geyer. *Markov chain Monte Carlo lecture notes (Statistics 8931)*. Tech. rep. University of Minnesota, 1998.

[102] Charles J. Geyer. 'The Metropolis-Hastings-Green Algorithm'. 2003. URL: http://www.stat.umn.edu/geyer/f05/8931/bmhg.pdf.

[103] Charles J. Geyer and Elizabeth A. Thompson. 'Annealing Markov chain Monte Carlo with applications to ancestral inference'. In: *Journal of the American Statistical Association* (1995).

[104] Warren Gilchrist. *Statistical Modelling with Quantile Functions*. CRC Press, 2000.

[105] Walter R. Gilks, Andrew Thomas and David J Spiegelhalter. 'A language and program for complex Bayesian modelling'. In: *The Statistician* (1994), pp. 169–177.

[106] Walter R. Gilks and Pascal Wild. 'Adaptive rejection sampling for Gibbs sampling'. In: *Applied Statistics* (1992), pp. 337–348.

[107] Mark Girolami and Ben Calderhead. 'Riemann-manifold Langevin and Hamiltonian Monte Carlo methods'. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214.

[108] Gianpaolo Gobbo and Benedict J. Leimkuhler. 'Extended Hamiltonian approach to continuous tempering'. In: *Physical Review E* (2015).

[109] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[110] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. 'Generative adversarial nets'. In: *Advances in Neural Information Processing Systems*. 2014.

[111] Claire C. Gordon, Thomas Churchill, Charles E. Clauser, Bruce Bradtmiller, John T. McConville, Ilse Tebbets and Robert A. Walker. *Anthropometric Survey of US Army Personell: Final Report*. Tech. rep. United States Army, 1988.

[112] Neil J. Gordon, David J. Salmond and Adrian F.M. Smith. 'Novel approach to nonlinear/non-Gaussian Bayesian state estimation'. In: *IEE Proceedings F (Radar and Signal Processing)*. Vol. 140. 2. IET. 1993, pp. 107–113.

[113] Christian Gourieroux, Alain Monfort and Eric Renault. 'Indirect inference'. In: *Journal of applied econometrics* 8.S1 (1993), S85–S118.

[114] Robert Gramacy, Richard Samworth and Ruth King. 'Importance tempering'. In: *Statistics and Computing* 20.1 (2010), pp. 1–7.

[115] Alex Graves. 'Practical Variational Inference for Neural Networks'. In: *Advances in Neural Information Processing Systems 24*. 2011, pp. 2348–2356.

[116] Todd L. Graves. 'Automatic step size selection in random walk Metropolis algorithms'. In: *arXiv preprint 1103.5986* (2011).

[117] Peter J. Green. 'Reversible jump Markov chain Monte Carlo computation and Bayesian model determination'. In: *Biometrika* (1995), pp. 711–732.

[118] Roger Grosse, Zoubin Ghahramani and Ryan P. Adams. 'Sandwiching the marginal likelihood using bidirectional Monte Carlo'. In: *arXiv preprint 1511.02543* (2015).

[119] James E. Gubernatis. 'Marshall Rosenbluth and the Metropolis algorithm'. In: *Physics of Plasmas* 12.5 (2005), p. 057303. URL: http://dx.doi.org/10.1063/1.1887186.

[120] Eduardo Gutiérrez-Peña and Stephen G. Walker. 'Statistical decision problems and Bayesian nonparametric methods'. In: *International Statistical Review* 73.3 (2005), pp. 309–330.

[121] Heikki Haario, Eero Saksman and Johanna Tamminen. 'An adaptive Metropolis algorithm'. In: *Bernoulli* (2001), pp. 223–242.

[122] Theodore E. Harris. 'The existence of stationary measures for certain Markov processes'. In: *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 2. 1956, pp. 113–124.

[123] John A. Hartigan. *Bayes theory*. Springer series in statistics. Springer-Verlag, 1983. ISBN: 9783540908838.

[124] Carsten Hartmann and Christof Schutte. 'A constrained hybrid Monte-Carlo algorithm and the problem of calculating the free energy in several variables'. In: *ZAMM-Zeitschrift fur Angewandte Mathematik und Mechanik* (2005).

[125] Laurent Hascoët and Valérie Pascual. 'The Tapenade Automatic Differentiation tool: Principles, Model, and Specification'. In: *ACM Transactions On Mathematical Software* 39.3 (2013). URL: http://dx.doi.org/10.1145/2450153.2450158.

[126] Cecil Hastings Jr., Frederick Mosteller, John W. Tukey and Charles P. Winsor. 'Low moments for small samples: a comparative study of order statistics'. In: *The Annals of Mathematical Statistics* (1947), pp. 413–426.

[127] W. Keith Hastings. 'Monte Carlo sampling methods using Markov chains and their applications'. In: *Biometrika* (1970).

[128] Bryan D. He, Christopher M. De Sa, Ioannis Mitliagkas and Christopher Ré. 'Scan Order in Gibbs Sampling: Models in Which it Matters and Bounds on How Much'. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1–9.

[129] David M. Higdon. 'Auxiliary variable methods for Markov chain Monte Carlo with applications'. In: *Journal of the American Statistical Association* 93.442 (1998), pp. 585–595.

[130] Matthew D Hoffman and Andrew Gelman. 'The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.' In: *Journal of Machine Learning Research* (2014).

[131] Yukito Iba. 'Extended ensemble Monte Carlo'. In: *International Journal of Modern Physics C* 12.05 (2001), pp. 623–656.

[132] Akihisa Ichiki and Masayuki Ohzeki. 'Violation of detailed balance accelerates relaxation'. In: *Physical Review E* 88.2 (2013), p. 020101.

[133] Ernst Ising. 'Beitrag zur theorie des ferromagnetismus'. In: *Zeitschrift für Physik A Hadrons and Nuclei* 31.1 (1925), pp. 253–258.

[134] Eric Jones, Travis Oliphant and Pearu Peterson. *SciPy: Open source scientific tools for Python*. [Online; accessed <today>]. 2001–. URL: http://www.scipy.org/.

[135] Herman Kahn and Theodore E. Harris. 'Estimation of particle transmission by random sampling'. In: *National Bureau of Standards applied mathematics series* 12 (1951), pp. 27–30.

[136] Robert E. Kass and Adrian E. Raftery. 'Bayes factors'. In: *Journal of the American Statistical Association* 90.430 (1995), pp. 773–795.

[137] Anthony D. Kennedy and Julius Kuti. 'Noise without noise: a new Monte Carlo method'. In: *Physical review letters* 54.23 (1985), p. 2473.

[138] Ross Kindermann and Laurie Snell. *Markov random fields and their applications*. American Mathematical Society, 1980.

[139] Diederik P. Kingma and Max Welling. 'Auto-Encoding Variational Bayes'. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. 2013.

[140] Scott Kirkpatrick, C. Daniel Gelatt and Mario P. Vecchi. 'Optimization by simulated annealing'. In: *Science* (1983).

[141] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Applications of Mathematics. Springer-Verlag, 1992. ISBN: 9783540540625.

[142] Andreĭ Nikolaevich Kolmogorov. *Foundations of the Theory of Probability*. Ed. by Nathan Morrison. 2nd English Edition. English translation of original 1933 German monograph, *Grundbegriffe der Wahrscheinlichkeitrechnung*. Chelsea Publishing Company, 1956. URL: https://pdfs.semanticscholar.org/c3e1/51f71168a5f348bdebfde11752ca603fa6d0.pdf.

[143] Dirk P. Kroese, Thomas Taimre and Zdravko I. Botev. 'Variance Reduction'. In: *Handbook of Monte Carlo Methods*. John Wiley & Sons, Inc., 2011, pp. 347–380. ISBN: 9781118014967. URL: http://dx.doi.org/10.1002/9781118014967.ch9.

[144] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman and David M. Blei. 'Automatic Differentiation Variational Inference'. In: *arXiv preprint 1603.00788* (2016).

[145] Solomon Kullback and Richard A. Leibler. 'On information and sufficiency'. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[146] Alessandro Laio and Michele Parrinello. 'Escaping free-energy minima'. In: *Proceedings of the National Academy of Sciences* (2002).

[147] Brenden M. Lake, Ruslan Salakhutdinov and Joshua B. Tenenbaum. 'Human-level concept learning through probabilistic program induction'. In: *Science* (2015).

[148] Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* (1998).

[149] Derrick H. Lehmer. 'Mathematical methods in large-scale computing units'. In: *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery (1949)*. 1951, pp. 141–146.

[150] Benedict J. Leimkuhler and Charles Matthews. 'Efficient molecular dynamics using geodesic integration and solvent–solute splitting'. In: *Proc. R. Soc. A*. The Royal Society. 2016.

[151] Benedict J. Leimkuhler and George W. Patrick. 'A symplectic integrator for Riemannian manifolds'. In: *Journal of Nonlinear Science* 6.4 (1996), pp. 367–384.

[152] Benedict J. Leimkuhler and Sebastian Reich. *Simulating Hamiltonian dynamics*. Cambridge University Press, 2004.

[153] Benedict J. Leimkuhler and Robert D. Skeel. 'Symplectic numerical integrators in constrained Hamiltonian systems'. In: *Journal of Computational Physics* (1994).

[154] Tony Lelièvre, Mathias Rousset and Gabriel Stoltz. 'Langevin dynamics with constraints and computation of free energy differences'. In: *Mathematics of computation* (2012).

[155] Meng Li and David Dunson. 'A framework for probabilistic inferences from imperfect models'. In: *arXiv preprint 1611.01241* (2016).

[156] Yingzhen Li and Richard E. Turner. 'Rényi divergence variational inference'. In: *Advances in Neural Information Processing Systems*. 2016.

[157] Yujia Li, Kevin Swersky and Rich Zemel. 'Generative moment matching networks'. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 1718–1727.

[158] Moshe Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[159] Fredrik Lindsten and Arnaud Doucet. 'Pseudo-Marginal Hamiltonian Monte Carlo'. In: *arXiv preprint 1607.02516* (2016).

[160] Anne-Marie Lyne, Mark Girolami, Yves Atchadé, Heiko Strathmann and Daniel Simpson. 'On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods'. In: *Statistical science* 30.4 (2015), pp. 443–467.

[161] David J.C. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[162] Olvi L. Mangasarian, W. Nick Street and William H. Wolberg. 'Breast cancer diagnosis and prognosis via linear programming'. In: *Operations Research* 43.4 (1995), pp. 570–577.

[163] Jean-Michel Marin, Pierre Pudlo, Christian P Robert and Robin J Ryder. 'Approximate Bayesian computational methods'. In: *Statistics and Computing* (2012).

[164] Enzo Marinari and Giorgio Parisi. 'Simulated tempering: a new Monte Carlo scheme'. In: *Europhysics Letters* (1992).

[165] Paul Marjoram, John Molitor, Vincent Plagnol and Simon Tavaré. 'Markov chain Monte Carlo without likelihoods'. In: *Proceedings of the National Academy of Sciences* (2003).

[166] George Marsaglia. 'Random numbers fall mainly in the planes'. In: *Proceedings of the National Academy of Sciences* 61.1 (1968), pp. 25–28.

[167] Makoto Matsumoto and Takuji Nishimura. 'Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator'. In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[168] Robert I. McLachlan, Klas Modin, Olivier Verdier and Matt Wilkins. 'Geometric generalisations of SHAKE and RATTLE'. In: *Foundations of Computational Mathematics* 14.2 (2014), pp. 339–370.

[169] Ross McVinish. 'Improving ABC for quantile distributions'. In: *Statistics and Computing* 22.6 (2012), pp. 1199–1207.

[170] Edward Meeds, Robert Leenders and Max Welling. 'Hamiltonian ABC'. In: *Proceedings of 31st Conference of Uncertainty in Artificial Intelligence*. 2015.

[171] Edward Meeds and Max Welling. 'Optimization Monte Carlo: Efficient and Embarrassingly Parallel Likelihood-Free Inference'. In: *Advances in Neural Information Processing Systems*. 2015.

[172] Kerrie L. Mengersen and Richard L. Tweedie. 'Rates of convergence of the Hastings and Metropolis algorithms'. In: *The annals of Statistics* 24.1 (1996), pp. 101–121.

[173] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller and Edward Teller. 'Equation of state calculations by fast computing machines'. In: *The Journal of Chemical Physics* (1953).

[174] Sean P. Meyn and Richard L. Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 1993.

[175] Grigori N. Milstein. 'Approximate integration of stochastic differential equations'. In: *Theory of Probability & Its Applications* 19.3 (1975), pp. 557–562.

[176] Thomas P. Minka. 'Expectation propagation for approximate Bayesian inference'. In: *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence* (2001).

[177] Shakir Mohamed and Balaji Lakshminarayanan. 'Learning in Implicit Generative Models'. In: *Proceedings of the International Conference on Learning Representations*. 2017.

[178] Jesper Møller, Anthony N. Pettitt, R. Reeves and Kasper K. Berthelsen. 'An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants'. In: *Biometrika* 93.2 (2006), pp. 451–458.

[179] Jorge J. Moré, Burton S. Garbow and Kenneth E. Hillstrom. *User Guide for MINPACK-1*. ANL-80-74, Argonne National Laboratory. 1980.

[180] Alexander Moreno, Tameem Adel, Edward Meeds, James M. Rehg and Max Welling. 'Automatic Variational ABC'. In: *arXiv preprint 1606.08549* (2016).

[181] Iain Murray. 'Advances in Markov chain Monte Carlo methods'. PhD thesis. University College London, University of London, 2007.

[182] Iain Murray. 'Differentiation of the Cholesky decomposition'. In: *arXiv preprint 1602.07527* (2016).

[183] Iain Murray, Ryan P. Adams and David J.C. MacKay. 'Elliptical slice sampling'. In: JMLR: W&CP 9 (2010), pp. 541–548.

[184] Iain Murray, Zoubin Ghahramani and David J. C. MacKay. 'MCMC for doubly-intractable distributions'. In: *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. AUAI Press, 2006, pp. 359–366.

[185] Iain Murray and Matthew M. Graham. 'Pseudo-marginal slice sampling'. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. 2016, pp. 911–919.

[186] Peter Neal and Clement Lee. 'Optimal scaling of the independence sampler: Theory and Practice'. In: *arXiv preprint 1511.04334* (2015).

[187] Radford M. Neal. 'Sampling from multimodal distributions using tempered transitions'. In: *Statistics and Computing* (1996).

[188] Radford M. Neal. *Markov chain Monte Carlo methods based on 'slicing' the density function*. Tech. rep. 9722. Department of Statistics, University of Toronto, 1997.

[189] Radford M. Neal. 'Annealed importance sampling'. In: *Statistics and Computing* (2001).

[190] Radford M. Neal. 'Slice sampling'. In: *Annals of statistics* (2003).

[191] Radford M. Neal. 'Improving asymptotic variance of MCMC estimators: Non-reversible chains are better'. In: *arXiv preprint 0407281* (2004).

[192] Radford M. Neal. 'MCMC using Hamiltonian dynamics'. In: *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011. Chap. 5, pp. 113–162.

[193] John von Neumann. 'Various techniques used in connection with random digits'. In: *National Bureau of Standards applied mathematics series* 3 (1951), pp. 36–38.

[194] Robert Nishihara, Iain Murray and Ryan P. Adams. 'Parallel MCMC with generalized elliptical slice sampling.' In: *Journal of Machine Learning Research* 15.1 (2014), pp. 2087–2112.

[195] Akihiko Nishimura and David Dunson. 'Geometrically Tempered Hamiltonian Monte Carlo'. In: *arXiv preprint 1604.00872* (2016).

[196] John F. Nolan. 'Analytical differentiation on a digital computer'. PhD thesis. Massachusetts Institute of Technology, 1953.

[197] Sheehan Olver and Alex Townsend. 'Fast inverse transform sampling in one and two dimensions'. In: *arXiv preprint 1307.1223* (2013).

[198] Art B. Owen. 'Importance sampling'. In: *Monte Carlo theory, methods and examples.* 2013. URL: http://statweb.stanford.edu/~owen/mc/Ch-var-is.pdf.

[199] A. Öztürk and R.F. Dale. 'A study of fitting the generalized lambda distribution to solar radiation data'. In: *Journal of Applied Meteorology* 21.7 (1982), pp. 995–1004.

[200] John W. Paisley, David M. Blei and Michael I. Jordan. 'Variational Bayesian Inference with Stochastic Search'. In: *Proceedings of the 29th International Conference on Machine Learning*. 2012, pp. 1367–1374.

[201] Surajit Pal. 'Evaluation of nonnormal process capability indices using generalized lambda distribution'. In: *Quality Engineering* 17.1 (2004), pp. 77–85.

[202] George Papamakarios and Iain Murray. 'Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation'. In: *Advances in Neural Information Processing Systems 29* (2016).

[203] Omiros Papaspiliopoulos, Gareth O. Roberts and Martin Sköld. 'Non-centered parameterisations for hierarchical models and data augmentation'. In: *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*. Vol. 307. Oxford University Press, USA. 2003.

[204] Omiros Papaspiliopoulos, Gareth O. Roberts and Martin Sköld. 'A general framework for the parametrization of hierarchical models'. In: *Statistical Science* (2007), pp. 59–73.

[205] Giorgio Parisi. *Statistical Field Theory*. Advanced Book Classics. Avalon Publishing, 1998. ISBN: 9780738200514.

[206] Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

[207] Peter H. Peskun. 'Optimum Monte-Carlo sampling using Markov chains'. In: *Biometrika* 60.3 (1973), pp. 607–612.

[208] Carsten Peterson and James R. Anderson. 'A mean field theory learning algorithm for neural networks'. In: *Complex systems* (1987).

[209] Michael Pitt, Ralph Silva, Paolo Giordani and Robert Kohn. 'Auxiliary particle filtering within adaptive Metropolis-Hastings sampling'. In: *arXiv preprint 1006.1914* (2010).

[210] Martyn Plummer. 'JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling'. In: *Proceedings of the 3rd international workshop on distributed statistical computing*. Vol. 124. Vienna. 2003, p. 125.

[211] Martyn Plummer, Nicky Best, Kate Cowles and Karen Vines. 'CODA: Convergence Diagnosis and Output Analysis for MCMC'. In: *R News* 6.1 (2006), pp. 7–11. URL: http://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

[212] Michael J.D. Powell. 'Numerical Methods for Nonlinear Algebraic Equations'. In: Gordon and Breach, 1970. Chap. A Hybrid Method for Nonlinear Equations.

[213] Dennis Prangle. 'Summary statistics in approximate Bayesian computation'. In: *arXiv preprint 1512.05633* (2015).

[214] Robert Price. 'A useful theorem for nonlinear devices having Gaussian inputs'. In: *IRE Transactions on Information Theory* 4.2 (1958), pp. 69–72.

[215] Jonathan K. Pritchard, Mark T. Seielstad, Anna Perez-Lezaun and Marcus W. Feldman. 'Population growth of human Y chromosomes: a study of Y chromosome microsatellites.' In: *Molecular biology and evolution* 16.12 (1999), pp. 1791–1798.

[216] James Gary Propp and David Bruce Wilson. 'Exact sampling with coupled Markov chains and applications to statistical mechanics'. In: *Random structures and Algorithms* 9.1-2 (1996), pp. 223–252.

[217] Adrian E. Raftery and Steven Lewis. *How many iterations in the Gibbs sampler?* Tech. rep. Washington University, Seattle, Department of Statistics, 1991.

[218] John S. Ramberg and Bruce W. Schmeiser. 'An approximate method for generating asymmetric random variables'. In: *Communications of the ACM* 17.2 (1974), pp. 78–82.

[219] Rajesh Ranganath, Sean Gerrish and David Blei. 'Black Box Variational Inference'. In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. 2014.

[220] Calyampudi R. Rao. 'Information and the accuracy attainable in the estimation of statistical parameters'. In: *Breakthroughs in statistics*. Springer, 1992, pp. 235–247.

[221] Carl E. Rasmussen and Chris K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006. ISBN: 9780262182539.

[222] Oliver Ratmann, Christophe Andrieu, Carsten Wiuf and Sylvia Richardson. 'Model criticism based on likelihood-free inference, with an application to protein network evolution'. In: *Proceedings of the National Academy of Sciences* (2009).

[223] Alfréd Rényi. 'On measures of entropy and information'. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. 1961, pp. 547–561.

[224] Danilo Jimenez Rezende, Shakir Mohamed and Daan Wierstra. 'Stochastic Backpropagation and Approximate Inference in Deep Generative Models'. In: *Proceedings of The 31st International Conference on Machine Learning*. 2014, pp. 1278–1286.

[225] Herbert Robbins and Sutton Monro. 'A stochastic approximation method'. In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[226] Christian P. Robert, Kerrie Mengersen and Carla Chen. 'Model choice versus model criticism'. In: *Proceedings of the National Academy of Sciences of the United States of America* (2010).

[227] Gareth O. Roberts and Jeffrey S. Rosenthal. 'Optimal scaling for various Metropolis-Hastings algorithms'. In: *Statistical science* 16.4 (2001), pp. 351–367.

[228] Gareth O. Roberts and Jeffrey S. Rosenthal. 'General state space Markov chains and MCMC algorithms'. In: *Probability Surveys* 1 (2004), pp. 20–71.

[229] Gareth O. Roberts and Adrian F.M. Smith. 'Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms'. In: *Stochastic processes and their applications* 49.2 (1994), pp. 207–216.

[230] Gareth O. Roberts and Richard L. Tweedie. 'Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms'. In: *Biometrika* (1996), pp. 95–110.

[231] Jeffrey S. Rosenthal. 'Optimal proposal distributions and adaptive MCMC'. In: *Handbook of Markov Chain Monte Carlo* (2011), pp. 93–112.

[232] Donald B. Rubin. 'Bayesianly justifiable and relevant frequency calculations for the applied statistician'. In: *The Annals of Statistics* 12.4 (1984), pp. 1151–1172.

[233] Ruslan Salakhutdinov and Iain Murray. 'On the quantitative analysis of deep belief networks'. In: *Proceedings of the 25th International Conference on Machine learning*. 2008.

[234] Tim Salimans, Diederik P. Kingma and Max Welling. 'Markov chain Monte Carlo and variational inference: Bridging the gap'. In: *International Conference on Machine Learning*. 2015.

[235] Tim Salimans and David A. Knowles. 'Fixed-form variational posterior approximation through stochastic linear regression'. In: *Bayesian Analysis* 8.4 (2013), pp. 837–882.

[236] John Salvatier, Thomas V. Wiecki and Christopher Fonnesbeck. 'Probabilistic programming in Python using PyMC3'. In: *PeerJ Computer Science* (2016).

[237] Lawrence K. Saul, Tommi Jaakkola and Michael I Jordan. 'Mean field theory for sigmoid belief networks'. In: *Journal of Artificial Intelligence Research* (1996).

[238] Chris Sherlock, Alexandre Thiery and Anthony Lee. 'Pseudo-marginal Metropolis–Hastings using averages of unbiased estimators'. In: *arXiv preprint 1610.09788* (2016).

[239] Chris Sherlock, Alexandre H. Thiery, Gareth O. Roberts and Jeffrey S Rosenthal. 'On the efficiency of pseudo-marginal random walk Metropolis algorithms'. In: *The Annals of Statistics* 43.1 (2015), pp. 238–275.

[240] Scott A. Sisson and Yanan Fan. 'Likelihood-free MCMC'. In: *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011. Chap. 12, pp. 313–333.

[241] Scott A. Sisson, Yanan Fan and Mark M. Tanaka. 'Sequential Monte Carlo without likelihoods'. In: *Proceedings of the National Academy of Sciences* 104.6 (2007), pp. 1760–1765.

[242] Bert Speelpenning. 'Compiling Fast Partial Derivatives of Functions Given by Algorithms'. PhD thesis. University of Illinois at Urbana-Champaign, 1980.

[243] Yi Sun, Jürgen Schmidhuber and Faustino J Gomez. 'Improving the asymptotic performance of Markov chain Monte-Carlo by inserting vortices'. In: *Advances in Neural Information Processing Systems*. 2010, pp. 2235–2243.

[244] Hidemaro Suwa and Synge Todo. 'Markov chain Monte Carlo method without detailed balance'. In: *Physical review letters* 105.12 (2010), p. 120603.

[245] Martin A. Tanner and Wing Hung Wong. 'The calculation of posterior distributions by data augmentation'. In: *Journal of the American statistical Association* 82.398 (1987), pp. 528–540.

[246] Simon Tavaré, David J. Balding, Robert C Griffiths and Peter Donnelly. 'Inferring coalescence times from DNA sequence data'. In: *Genetics* 145.2 (1997), pp. 505–518.

[247] Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual*. Version 2.16.0. URL: http://mc-stan.org.

[248] Theano development team. 'Theano: A Python framework for fast computation of mathematical expressions'. In: *arXiv preprint 1605.02688* (2016).

[249] Madeleine B. Thompson. 'A comparison of methods for computing autocorrelation time'. In: *arXiv preprint 1011.0175* (2010).

[250] Luke Tierney. 'Markov chains for exploring posterior distributions'. In: *The Annals of Statistics* (1994), pp. 1701–1728.

[251] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen and Michael P.H. Stumpf. 'Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems'. In: *Journal of the Royal Society Interface* 6.31 (2009), pp. 187–202.

[252] Dustin Tran, Rajesh Ranganath and David M. Blei. 'Deep and Hierarchical Implicit Models'. In: *arXiv preprint 1702.08896* (2017).

[253] Minh-Ngoc Tran, David J. Nott and Robert Kohn. 'Variational Bayes with intractable likelihood'. In: *Journal of Computational and Graphical Statistics* (2017).

[254] John W. Tukey. *Practical Relationship Between the Common Transformations of Percentages or Fractions and of Amounts*. Technical Report 36. Statistical Research Group,Princeton, 1960.

[255] Konstantin S. Turitsyn, Michael Chertkov and Marija Vucelja. 'Irreversible Monte Carlo algorithms for efficient sampling'. In: *Physica D: Nonlinear Phenomena* 240.4 (2011), pp. 410–414.

[256] Stanislaw Ulam and Nicholas Metropolis. 'The Monte Carlo method'. In: *Journal of the American Statistical Association* 44.247 (1949), pp. 335–341.

[257] David A. Van Dyk and Xiao-Li Meng. 'The art of data augmentation'. In: *Journal of Computational and Graphical Statistics* 10.1 (2001), pp. 1–50.

[258] Wolf Vanpaemel. 'Prior sensitivity in theory testing: An apologia for the Bayes factor'. In: *Journal of Mathematical Psychology* 54.6 (2010), pp. 491–498.

[259] Gunter Weiss and Arndt von Haeseler. 'Inference of population history using a likelihood approach'. In: *Genetics* 149.3 (1998), pp. 1539–1546.

[260] Max Welling and Yee W. Teh. 'Bayesian learning via stochastic gradient Langevin dynamics'. In: *Proceedings of the 28th International Conference on Machine Learning*. 2011.

[261] Robert Edwin Wengert. 'A simple automatic derivative evaluation program'. In: *Communications of the ACM* 7.8 (1964), pp. 463–464.

[262] Richard David Wilkinson. 'Approximate Bayesian computation (ABC) gives exact results under the assumption of model error'. In: *Statistical applications in genetics and molecular biology* (2013).

[263] Simon N. Wood. 'Statistical inference for noisy nonlinear ecological dynamic systems'. In: *Nature* 466.7310 (2010), pp. 1102–1104.

[264] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov and Roger Grosse. 'On the Quantitative Analysis of Decoder-Based Generative Models'. In: *arXiv preprint 1611.04273* (2016).

[265] Emilio Zappa, Miranda Holmes-Cerfon and Jonathan Goodman. 'Monte Carlo on manifolds: sampling densities and integrating functions'. In: *arXiv preprint 1702.08446* (2017).

[266] Yichuan Zhang, Zoubin Ghahramani, Amos J. Storkey and Charles A Sutton. 'Continuous relaxations for discrete Hamiltonian Monte Carlo'. In: *Advances in Neural Information Processing Systems*. 2012.

[267] Oliver Zobay. 'Mean field inference for the Dirichlet process mixture model'. In: *Electronic Journal of Statistics* (2009).