UNIVERSITY OF CAMBRIDGE

Exploring Probability Measures with Markov Processes

Samuel Power

Churchill College

This dissertation is submitted on July 13, 2020 for the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or am concurrently submitting, for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or is being concurrently submitted, for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. This dissertation does not exceed the prescribed limit of 60 000 words.

Samuel Power
July 13, 2020

Abstract

Author: Samuel Power
Thesis Title: Exploring Probability Measures with Markov Processes

In many domains where mathematical modelling is applied, a deterministic description of the system at hand is insufficient, and so it is useful to model systems as being in some way stochastic. This is often achieved by modeling the state of the system as being drawn from a probability measure, which is usually given algebraically, i.e. as a formula. While this representation can be useful for deriving certain characteristics of the system, it is by now well-appreciated that many questions about stochastic systems are best-answered by looking at samples from the associated probability measure. In this thesis, we seek to develop and analyse efficient techniques for generating samples from a given probability measure, with a focus on algorithms which simulate a Markov process with the desired invariant measure.

The first work presented in this thesis considers the use of *Piecewise-Deterministic Markov Processes* (PDMPs) for generating samples. In contrast to usual approaches, PDMPs are i) defined as continuous-time processes, and ii) are typically *non-reversible* with respect to their invariant measure. These distinctions pose computational and theoretical challenges for the design, analysis, and implementation of PDMP-based samplers. The key contribution of this work is to develop a transparent characterisation of how one can construct a PDMP (within the class of *trajectorially-reversible* processes) which admits the desired invariant measure, and to offer actionable recommendations on how these processes should be designed in practice.

The second work presented in this thesis considers the task of sampling from a probability measure on a discrete space. While work in recent years has made it possible to apply sampling algorithms to probability measures with differentiable densities on continuous spaces in a reasonably generic way, samplers on discrete spaces are still largely derived on a case-by-case basis. The contention of this work is that this is not necessary, and that one can in fact define quite generally-applicable algorithms which can sample efficiently from discrete probability measures. The contributions are then to propose a small collection of algorithms for this task, and verify their efficiency empirically. Building

on the previous chapter's work, our samplers are again defined in continuous time and non-reversible, each of which offer noticeable benefits in efficiency.

The third work presented in this thesis concerns a theoretical study of a particular class of Markov Chain-based sampling algorithms which make use of parallel computing resources. The Markov Chains which are produced by this algorithm are mathematically equivalent to a standard Metropolis-Hastings chain, but their real-time convergence properties are affected nontrivially by the application of parallelism. The contribution of this work is to analyse the convergence behaviour of these chains, and to use the 'optimal scaling' framework (as developed by Roberts, Rosenthal, and others) to make recommendations concerning the tuning of such algorithms in practice.

The introductory chapters provide a general overview on the task of generating samples from a probability measure, with particular focus on methods involving Markov processes. There is also an interlude on the relative benefits of i) continuous-time and ii) non-reversible Markov processes for sampling, which are intended to provide additional context for the reading of the first two works.

Acknowledgements

I would first like to thank my supervisor, Dr. Sergio Bacallado, for his support and guidance over the past four years. The decision to come to Cambridge for my doctoral studies was very much influenced by his support in the early stages, and looking back at his arguments for coming to Cambridge, they were prescient in that they encapsulate many of the things which I have enjoyed so much about my time here. Since my arrival, we have never struggled to find interesting topics to discuss and work on together, and I have learned a great deal. I could not have asked for a better supervisor.

During my time in Cambridge, my experience has been enriched greatly by the various communities of which I have found myself a part.

Firstly, the CCA and its members provided me with a wonderful welcome back into the world of Mathematics, which was bolstered immensely by the diversity of topics studied by its members. The CAKE Seminar and CCA Coffee were well-loved components of my time at the CMS, which I will miss. Thanks to Kweku Abraham and Tessa Blackman in particular for contributing to this great atmosphere.

Secondly, the collegial nature of the CCIMI has been nourishing, and I am proud to count all members of our initial cohort as close friends. My warm thanks to Ed Ayers, Andrew Celsus, Eric Hanson, Ferdia Sherry, and Sven Wang, for being ever-present fixtures in my time at the CMS, from whom I have learned so much, and with whom I feel lucky to have overlapped. Further thanks are also due to Rachel Furner, Josh Stevens, and Carola Schoenlieb for their hard work in making the CCIMI what it is. I am also grateful to the CCIMI itself for generously funding my doctoral studies.

Thirdly, there have been a number of groups to which I do not strictly belong, but who have nevertheless been extremely welcoming to me over the years, and so I offer my appreciation to the Cambridge Image Analysis (CIA), Frequentist Bayesian Inference (FBI), and Signal Processing (SigProc) groups for each having treated me as one of their own. I would like to express special gratitude to Sam Duffield, Jacob Vorstrup Goldman, Torben Sell, and Sumeetpal Singh for welcoming me into the SigProc Reading Group, which has played an pivotal role in shaping my research over the past two years.

Beyond Cambridge, I would like to thank Marcelo Pereyra and Kostas Zygalakis for welcoming me to Edinburgh numerous times over the past few years; it has always been

interesting and rewarding, and I hope that this continues over the years to come. I would also like to thank Tony Lelievre and Gabriel Stoltz for welcoming me to Paris in 2019, and to Tony for his continued willingness to meet during his time in London. Thanks are also due to Michael Betancourt, for his patience and generosity with his insights.

Before coming to Cambridge, I was fortunate to study under the tutelage of Ruth Baker, Glenys Luke, and Tom Sanders, each of whom were instrumental to my mathematical development, and alongside Bill O'Donovan and Samuel Johnston, whose friendship made it so clear that mathematics is most rewarding when it is social. When making the decision to return to academic mathematics, the collective encouragement and advice which they provided made matters all the more obvious. Thanks are also due to Gabriel Chircu for having elevated my view of what mathematics could be in an exhilarating way at a formative stage.

I am also extremely thankful to my family for their role in all of this; from encouraging my interest in mathematics in the early years, to giving me the freedom to pursue that interest as fully as possible, even at quite (!) a distance. I could not have asked for more support.

Finally, my deepest gratitude goes to my partner Scarlett, whose encouragement and patience has allowed me to throw myself into research without fear, and whose support has allowed me to remain grounded. She has made me feel at home in a way which is hard to put into words, and for this, and much more, I dedicate this thesis to her.

Summary of Contributions

Chapter 1 of this thesis consists of review material and known results, cited where appropriate.

Chapter 2 consists of original work. The idea of studying PDMPs with time-enriched and split structures is my own, and the proofs that the proposed processes admit the correct invariant measure are my own. My supervisor Sergio Bacallado proposed the problem of showing 'completeness' of these processes, the solution to which and proofs of which are my own.

Chapter 3 consists of original work, produced in collaboration with Jacob Vorstrup Goldman. I developed the conceptual basis of the algorithms contained therein, and the proofs of their theoretical properties are my own. Jacob implemented the algorithms, and ran the numerical experiments shown in the paper, which we designed together. The work was originally written up jointly, and I have since independently edited it to fit the presentation of this thesis.

Chapter 4 consists of original work.

All chapters benefitted greatly from the supervision and guidance of Sergio Bacallado.

Contents

Chapter 1

Introduction

1.1 Understanding Probability Measures through Simulation

In mathematical modelling, one is often faced with trying to understand large, complex systems, involving numerous variables undergoing nontrivial interactions. As the complexity of such systems grows, a deterministic description of their behaviour may become insufficient, and so it is often enlightening to model them as stochastic in nature. In particular, it is common to specify the possible states of the system in terms of some probability measure π. Instead of talking about what *will* happen in the system, one now begins to talk about what *could* happen, what *is likely* to happen, and other such questions, couched in the language of uncertainty. This can enable a richer description of such systems, and has been an influential approach in many disparate domains. Below, we outline a few of these domains, with an eye towards providing context for the range of fields in which stochastic models have been adopted, and why they are a natural fit.

- **Statistics** - In modern statistical modelling, it is near-universal to model the behaviour of data, given model parameters, as being drawn from some appropriate probability distribution. Stochastic modelling allows statisticians to systematically account for variations in observations which are impossible or challenging to model in a purely mechanistic way. Exploring the generative process behind a statistical model allows us to understand what sort of behaviour we are encoding in our specification of the model, and is an invaluable tool for assessing whether a given statistical model is appropriate for a certain task. Bayesian statistics (see e.g. [Jay96, Rob07, RC13, GCS+13] for coverage of various aspects of this vast field) takes this approach one step further, and quantifies uncertainty about the *parameters of the model* in a probability measure as well, known as the *posterior distribution*. Exploring the posterior distribution of a statistical model is akin to exploring the

space of possible explanations of the data, weighted according to how well each explanation aligns with both the data, and our a priori understanding of the system. We note that a number of other non-'strictly-Bayesian' approaches to statistical learning often exploit similar perspectives; see e.g. [DT12, Gue19] for some examples of this framework.

- **Natural Sciences** - Across the natural sciences, stochastic modelling provides a lens through which to understand both dynamic and static systems. While ordinary differential equations provide a useful first approximation to a number of systems of interest, they are by their very nature *overconfident* about how a system will evolve in time. This makes them particularly unreliable for studying the long-time behaviour of systems which are close to deterministic, but under the influence of some perturbation (deterministic, stochastic, systematic, or otherwise), as there will typically be some bias incurred, whose effects become more pronounced over time. A useful strategy is to account for these perturbations, fluctuations, and drift as an intrinsic stochastic aspect of the system, acknowledging that while there may be some regular patterns to the evolution of the system, a purely deterministic model will become insufficient at some stage. Some concrete examples of dynamical models in the sciences which have fruitfully been modelled by stochastic techniques can be found in Systems Biology [Wil07, Wil18], Ecology [Woo10], Chemical Kinetics [AH12, Gal16, LBGY16], Genetics [BZB02, BCSJ12, GT94], and Particle Transport [JW95, Vea97, MMN$^+$13, RF13, Lux18].

  Static models of the natural world can also be viewed through a stochastic lens. One common source of such models comes from systems which evolve in continuous time, but on sufficiently fast timescales that, at any given time, they have relaxed to some equilibrium measure, and their behaviour can thus be summarised through that measure. This view is particularly natural in the context of Statistical Physics [VW77, Sok97, NB99, Kra06, KW09, BCH$^+$12, LB14a], Molecular Simulation [FS01, SRL10], and Chemical Physics [PR09], and also arises in areas somewhat further afoot, including Quantum Chemistry [Cre81, RCALJ82, HLR94] and Quantum Field Theory [DKPR87],

- **Signal Processing** - Various forms of online signal processing involve estimating the state of some partially-observed system, and it is natural to model uncertainty about both i) the dynamics of the system, and ii) the unknown state of the system. Modelling the dynamics and observations in a purely-deterministic way has a deficiency in that when the true system deviates from the model, many methods will exhibit a bias which gets worse and worse over time. Standard tasks of this form include (Nonlinear) Filtering [CR11b, DGA00, Fea98, Smi13] and Data

Assimilation* [Eve09, LSZ15, RC15, VLCR15], as well as Multi-Target Tracking [HLCP02, VSD03]. Similar comments apply to Control Theory and Stochastic Approximation [KY03, BMP12], which tend to operate with similar models, though with slightly different specific goals.

- **Finance and Operations Research** - While there exist simple models in which a deterministic treatment is sufficient, it is by now well-established that financial systems are best-modelled by structures which acknowledge uncertainty in the form of stochastic models. Finance is in some sense a natural candidate for this treatment, due to the impact of forces operating at widely-differing scales (individual behaviour, national and governmental actions, short- and long-term effects). We refer to [Jäc02, GG06, LEc09, McL11, Gla13] for various accounts of stochastic modelling and computation in the context of finance. For similar reasons, Econometrics [Gew89, Cre12] has often been treated in a stochastic fashion. Queueing Theory [AG07, Asm08, Bré13] deviates somewhat from the previous two examples, as it is typically couched in the terminology of discrete-event systems and jump processes, rather than continuous motion. As such, the randomness in question primarily models the intervals between different events taking places, as well as the nature of those events.

- **Discrete Structures and Counting** - A key observations in the study of discrete structures has been that in order to find an element with a particular property, instead of carrying out a careful, deterministic search for such an element, it can be instructive to instead consider generating an element randomly, and using probabilistic arguments to reason about how likely it is for that random element to possess that property. This is the so-called *probabilistic method* (surveyed and exposited beautifully in [AS04]), and has been highly influential in the study of combinatorics and the analysis of discrete structures more broadly; see also [SJ89, JS89, JS96, RK13] for related applications.

- **Uncertainty Quantification** - In recent years, there has been some coalescence between lines of research in Bayesian Statistics, Engineering, Inverse Problems, and Data Assimilation, into a community now united under the banner of '*Uncertainty Quantification*'. Broadly speaking, the problems at hand tend to involve real physical systems occurring in the natural sciences, often derived from a differential equation of some form. These systems are then typically observed indirectly, giving rise to an inverse problem (see e.g. [DS17, Stu10, BGL$^+$17]). Particular challenges in this region stem from the high-dimensional parameter spaces, computationally-intensive

*The terms are largely synonymous / refer to similar models; the communities corresponding to these terms do not necessarily overlap in accordance with that.

model evaluations, and nonlinear forward maps. The probabilistic treatment of these problems has become increasingly prevalent in recent years, as deterministic methods have shown themselves to be somewhat limited in fully capturing the subtleties induced by these complex models. One is often observing data of much lower dimension than the parameter of interest, with an observation map which is far from linear, and a dataset which is far from large enough for a localised approximation to the model to provide a satisfactory picture. Stochastic models force the scientist to confront these unpleasant truths in a more overt way, and encourage a more thorough approach to quantification of uncertainty.

The common threads are broadly as follows: there is some system of interest (physical, inferential, artificial, or otherwise) which it is appropriate to model as stochastic, due to some aleatoric or epistemic uncertainty about its state and evolution. The uncertainty in the stochastic system is something which we would like to understand the scope of; if a deterministic representation were sufficient, then one would typically use it directly. However, the source and interpretation of the randomness involved can be quite diverse. For example:

- In inferential contexts (e.g. Bayesian Statistics, Uncertainty Quantification, ...), the uncertainty is a way of encapsulating our beliefs about the parameters of some statistical problem.
- In stochastic models of real physical systems (e.g. Chemical Kinetics, Ecology, Molecular Dynamics), the uncertainty might represent fluctuations of the system at scales which we cannot resolve analytically or mechanistically, and be used to understand the stability of the overall system to perturbations at these levels.
- In more theoretical contexts (e.g. Statistical Physics), the uncertainty in the system is often taken as innate, and the interest is in understanding the typical behaviour of the system, the scales at which fluctuations occur, and the stability of these conclusions with respect to model parameters (e.g. temperature).

The first step in building models of this from is to derive some algebraic representation of the measure, usually in the form of a density with respect to some appropriate dominating measure. Given such an algebraic representation, one can then begin to ask analytical and structural questions about the measure: on which sets does this measure place mass? Does it satisfy certain smoothness and regularity conditions? By asking these questions first, we get a sense for which mathematical tools will be available for us when interacting with the measure going forward.

While qualitative questions of this form are useful for gaining an abstract understanding of the measure, they cannot always tell the whole story. When it comes time to ask

quantitative questions about the measure at hand, it is often the case that standard analytic and algebraic tools can fall short, or at least become more challenging to apply. Archetypal tasks of this form involve computing expectations under the measure, such as moments of functions, probabilities of specific events, marginal densities of given variables, and so on. When it comes to computing expectations under a given measure, the availability of analytically-tractable solutions is the exception, rather than the rule, and the same generally holds true for tasks involving a quantitative understanding of the measure.

One of the beautiful flexibilities of probability measures is that they are not purely analytical objects; they are also intimately tied to the notion of random variables, a rich connection which enables a more tangible representation of what the measure represents. Given a probability measure π, one can associate to it a random variable X, and by studying the properties of this random variable, one can begin to probe the properties of π.

Given access to a simulator which can generate realisations of the random variable X, a number of doors spring open, which can enable us to 'explore' the content of π. One can assess the typical properties of a sample from π, uncover patterns in how different samples vary (and do not), as well as using the samples to carry out concrete computations, as is done in the *Monte Carlo* method (see e.g. [Ham13]). While there is a well-understood appreciation for the latter application, the former are perhaps under-emphasised. Drawing samples can help us to answer questions about π which we already have, but it can also be used in a more exploratory fashion, by looking at samples, and using them to prompt new questions which would be difficult to dream up from a purely analytical standpoint[†]. Given the complexity of many stochastic systems of interest, it is an important challenge to derive methodological tools for understanding 'what is going on' with a given system, in a broad sense. Being able to draw samples from a probability measure is a well-posed first step in this direction, which enables the formation of a more coherent picture of the problem at hand, in a very general setting.

It is thus a primary contention of this thesis that it is of value to derive algorithmic techniques which facilitate the efficient exploration of probability measures. While the chief application domain for such methods is arguably in Monte Carlo computations, where a number of additional considerations (such as variance reduction, post-processing, and much more) come into play, the core focus of this thesis is to derive and analyse procedures which can deliver those samples in the first place. We hope that this presentation allows for a more focused exposition of the key concepts.

[†]For example, in the context of materials science, the use of molecular simulation as a scientific tool has been compared (see e.g. [SRL10]) to a 'computational microscope'.

1.2 Techniques for Exploration of Probability Measures

Settling on the task of exploring probability measures by simulation, the practical question arises of exactly how one ought to go about this. It is fair to say that the answer to this question cannot be answered with a single procedure, due to the diversity of stochastic systems which are encountered in practice. As is typical in computation, the algorithmic solution to a problem is contingent upon the resources which the problem affords you, and different tools are suitable under different problem settings. As such, it is useful to think of stochastic simulation less as a unified method, and more as a toolbox; a collection of methods which can be deployed in conjunction with one another, depending upon the precise nature of the problem at hand. Here, we present a survey of some of the most useful tools. We begin with tools which make strong assumptions, and can thus solve the simulation problem in a strong sense. From there, we gradually work towards tools which are more broadly-applicable, but deliver solutions at a greater cost, either computationally, or by inducing some bias.

1.2.1 Exact Sampling Techniques

1.2.1.1 Inversion Sampling

Perhaps the most widely-used ingredient of exact sampling methods is *inversion sampling*. For this algorithm to apply, it is necessary that both π and its cumulative distribution function (CDF) F can be manipulated directly. This typically restricts the applicability of the method to low-dimensional state spaces (in particular, most applications are to one-dimensional distributions), or countable discrete state spaces, and to measures π which are reasonably well-understood. The significance of this method is not that it allows for the direct solution of many hard problems, but that the solution of hard problems will typically involve sub-problems which *are* amenable to this treatment.

For probability measures π which are supported on the real line $\mathbf{R}$, one can sample realisations of $X \sim \pi$ directly by Algorithm 1.

Algorithm 1 Sampling $X \sim \pi$ by inversion, π supported on $\mathbf{R}$

1. Sample $u \sim \mathcal{U}[0, 1]$.
2. Let $x = \inf\{y \in \mathbf{R} : F(y) \geqslant u\}$, where $F(y) = \int_{-\infty}^{y} \pi(t)dt$.
3. Output x.

The same technique applies to π which are supported on some countable set $\mathcal{X}$. Supposing without a loss of generality that $\mathcal{X} = \mathbf{Z}_{\geqslant 1} = \{1, 2, \ldots\}$, one can apply Algorithm 2.

Algorithm 2 Sampling $X \sim \pi$ by inversion, π supported on $\mathbf{Z}$

1. Sample $u \sim \mathcal{U}[0,1]$.
2. Let $x = \min\{k \in \mathbf{Z}_{\geqslant 1} : F(k) \geqslant u\}$, where $F(k) = \sum_{j=1}^{k} \pi(j)$.
3. Output x.

One immediate roadblock to using Inversion Sampling is the requirement that the CDF of π be directly available, which is not always the case, even for low-dimensional problems.

1.2.1.2 Rejection Sampling

An easy way to extend the scope of Inversion Sampling is to consider π for which the CDF is not necessarily available, but such that $\pi \approx \nu$ for a ν which **is** amenable to inversion sampling. More precisely, suppose that $\pi(x) \leqslant M \cdot \nu(x)$ for all x, where ν can be sampled from exactly, and M is a known 'bounding constant'. One can then draw samples from π using the technique of *Rejection Sampling*, presented as Algorithm 3.

Algorithm 3 Rejection Sampling $X \sim \pi$ with proposal ν, bounding constant M

1. Set $a = 0$. Until $a = 1$, do:
   (a) Sample $x \sim \nu$ (using e.g. Algorithm 1 or 2).
   (b) Sample $u \sim \mathcal{U}[0,1]$.
   (c) If $u < \frac{\pi(x)}{M \cdot \nu(x)}$, set $a = 1$.
2. Output x.

This greatly extends the scope of exact sampling, as it allows one to sample from any measure π of which we have *pointwise* knowledge, and which we can globally upper-bound by some measure ν which we understand well; see [Dev06] for an encyclopaedic collection of examples to this effect. Crucially, once a bound is found, no integration is required. However, the algorithm now has a random running time, and if the bounding constant M is too large, this run time will grow.

One approach to reducing M is to use more flexible families of ν. For example, if $\nu(x)$ can be decomposed as a *mixture* of N components, i.e. we can write $\nu(x) = \sum_{i=1}^{N} p_i \nu_i(x)$ with $\mathbf{p}$ a probability vector, and each of the ν_i a probability measure for which exact sampling is possible, then one can sample from ν using Algorithm 4. By using N separate components, it is often possible to tune the parameters of ν such that M is much closer to 1 than would be possible with a single component.

Algorithm 4 Mixture Sampling from $\nu(x) = \sum_{i=1}^{N} p_i \nu_i(x)$

1. Sample $I \sim \mathbf{p}$, i.e. set $I = i$ with probability p_i.
2. Sample $x \sim \nu_i(x)$.
3. Output x.

One can also construct such mixtures adaptively, as in [GW92]. However, these methods can only help up to a point; as the dimension of the problem increases, one typically expects that even the optimal choice of M will grow exponentially with dimension (or that the complexity of representing ν will grow exponentially). This means that the scope of direct rejection sampling is largely restricted to low-dimensional problems, or to high-dimensional measures which are very close to solvable already; see e.g. [VPD19] for an example of the latter.

1.2.1.3 Recursive Sampling

The difficulty of rejection sampling in high dimension essentially stems from trying to get a correct sample in one hit, i.e. generating a d-dimensional vector $x \sim \nu$, and then checking whether it 'looks right' under π. A more sensible approach would be to generate the vector x in parts, making sure that each of the parts looks right in sequence, and then outputting the final vector. This is the idea behind *Recursive Sampling*, usually known as *Ancestral Sampling* (see e.g. [Bis06]).

To give a simple example, suppose that we want to sample a d-dimensional vector $x \sim \pi(x) = \pi(x_1, \ldots, x_d)$. In principle, one can always decompose this target measure into a product of marginal and conditional distributions, as

$$\pi(x_1, \ldots, x_d) = \pi(x_1) \cdot \pi(x_2|x_1) \cdot \pi(x_3|x_1, x_2) \cdot \cdots \cdot \pi(x_d|x_1, \ldots, x_{d-1}).$$

Provided that each of these conditional distributions can be sampled from, the whole measure can be sampled from, using Algorithm 5.

Algorithm 5 Sequential Recursive Sampling $X \sim \pi(x) = \pi(x_1, \ldots, x_d)$

1. For $i = 1, 2, \ldots, d$:
   (a) Sample $x_i \sim \pi(x_i|x_1, \ldots, x_{i-1})$ (using e.g. Algorithm 1, 2, or 3).
2. Output $x = (x_1, \ldots, x_d)$.

Note that the procedure above works (in principle) for any ordering of the variables

$(x_1, \ldots, x_d)$, though in practice, some orderings may be more efficient than others. Moreover, the decomposition need not be sequential *per se*. Suppose that the indices $\{1, 2, \ldots, d\}$ can be equipped with the structure of a partial order $\prec$ (directed acyclic graph; see [Lau96, KF09]), such that for each i, the conditional distribution $\pi(x_i|x_{\prec i}) = \pi\left(x_i|\{x_j\}_{j\prec i}\right)$ is possible to sample from. This structure is relatively common in the specification of prior models in Bayesian statistics, as well as in various generative models used in machine learning; see e.g. [GDM$^+$14, UML14, GGML15, KSJ$^+$16, HKLC18, WL20a, WL20b]. One can then use Algorithm 6 to generate samples from π.

Algorithm 6 General Recursive Sampling $X \sim \pi(x) = \pi(x_1, \ldots, x_d)$ with partial order $\prec$

1. Specify an ordering on indices such that $\{i_1, \ldots, i_d\} = \{1, \ldots, d\}$, and $i_1 \prec \cdots \prec i_d$ under the partial order $\prec$.
2. For $j = 1, \ldots, d$,
   (a) Sample $x_{i_j} \sim \pi(x_{i_j}|\{x_{i_k}\}_{k<j})$.
3. Output $x = (x_1, \ldots, x_d)$.

Note that the complexity of both Algorithms 5 and 6 is linear in d, i.e. linear in the complexity of representing the sample. This is a substantial improvement upon the typically-exponential complexity of trying to sample such an x 'in one hit'. Moreover, for certain partial ordering structures, the real-time cost of this algorithm can be reduced further when parallel processing is available, particularly when the associated directed acyclic graph has low depth.

At this point, it is worth remarking on the modular nature of these algorithms. In Algorithms 5 and 6, the method demands samples from certain conditional distributions, but is otherwise agnostic as to how they were generated; one can use inversion, rejection, or any other technique at one's disposal. This is a desirable feature, and one which a large majority of simulation algorithms favour. If a simulation method is modular in this way, it can easily and unbiasedly be embedded into other simulation methods. Interesting models are often high-dimensional and not amenable to explicit computation. As such, in order to make them surmountable, we need to find ways of handling them which are modular. This is one explanation for why certain communities within the field of simulation place such a high premium on unbiased and asymptotically-unbiased methods; by eliminating sources of bias, one can more safely nest methods inside more complicated algorithms. In what follows, it will be left implicit that the source of samples does not matter, unless specifically noted.

1.2.2 Importance Sampling, and Variants

The exact sampling techniques derived above are quite powerful, but also hinge upon quite strong assumptions. Inversion samplers are available only for a small class of very nice measures, rejection samplers require knowledge of certain global upper bounds, and recursive samplers require that the measure at hand admit a convenient decomposition into conditional distributions from which we can sample exactly.

Unfortunately, it is often the case that none of these conditions hold in a meaningful sense. One then needs to reassess how 'exact' a sample needs to be in order to be useful, and the nature of approximations which can be afforded.

One relaxation which is often used is the following: instead of demanding a collection of exact samples from π, consider taking a collection of samples from some other measure ν such that[‡] the support of ν contains the support of π, and then associate to each sample x a weight $w(x) \geqslant 0$, which assesses how close x is to being a draw from π, in some sense. This is the basis of *importance sampling*, which is typically implemented as in Algorithm 7.

Algorithm 7 Importance Sampling $X \sim \pi$ with proposal ν

1. Sample $x \sim \nu$.
2. Compute $w = w(x) = \frac{\pi(x)}{\nu(x)}$.
3. Output (x, w).

The premise of importance sampling is essentially that an appropriately-weighted collection of inexact samples can be 'as good' as an unweighted collection of exact samples, if one is primarily interested in computing expectations under π. Note that

$$\begin{aligned}\mathbf{E}_\nu[w(x)f(x)] &= \int \nu(x)w(x)f(x)dx \\ &= \int \nu(x)\left(\frac{\pi(x)}{\nu(x)}\right)f(x)dx \\ &= \int \pi(x)f(x)dx,\end{aligned}$$

and as such, that importance sampling enables unbiased estimation of expectations under π, without ever drawing samples from π. For certain applications, one is more interested in the computation of expectations than in drawing samples, and so importance sampling can be a useful tool.

[‡]In greatest precision and generality, it is necessary that the measure π be *absolutely continuous* with respect to ν; see e.g. [Bil08] for a precise definition of this notion. For most finite-dimensional, practical applications, it suffices that both measures admit densities with respect to Lebesgue measure, and that the support of π is contained in that of ν.

Note that importance sampling can also be implemented in a recursive fashion. For example, suppose that we want to generate weighted samples from $\pi(x) = \pi(x_{1:T})$. We will generally **not** have a clean decomposition of π into conditionals and marginals, as exploited in 5, but we can often still construct a sequence of interpolating measures which start with a low-dimensional target, and end up with the full π. Suppose then that we fix a sequence of measures $\{\pi_t(x_{1:t})\}_{t=1}^T$, such that for each t, π_t can be evaluated exactly, and $\pi_T = \pi$. Similarly to 5, we assume existence of a proposal distribution $q(x_{1:T})$ which *can* be decomposed into a sequence of proposal distributions $\{q_t(x_t|x_{1:t-1})\}_{t=1}^T$, all of which can be both sampled from and evaluated. Given these tools, we can generate a weighted sample from π using Algorithm 8, also known as *Sequential Importance Sampling* (SIS).

Algorithm 8 Sequential Importance Sampling $X \sim \pi$, interpolating sequence $\{\pi_t\}_{t=1}^T$, proposals $\{q_t\}_{t=1}^T$

1. Sample $x_1 \sim q_1(x_1)$, and set $w_1 = \frac{\pi_1(x_1)}{q_1(x_1)}$.
2. For $1 < t \leqslant T$, sample $x_t \sim q_t(x_t|x_{1:t-1})$, and set $w_t = w_{t-1} \cdot \frac{\pi_t(x_{1:t})}{\pi_{t-1}(x_{1:t-1}) \cdot q_t(x_t|x_{1:t-1})}$
3. Output (x, w_T).

It should be noted that SIS is equivalent to basic importance sampling with proposal $q(x_{1:T}) = q_1(x_1) \cdot \prod_{1<t\leqslant T} q_t(x_t|x_{1:t-1})$. However, the form of Algorithm 8 will be useful for certain extensions of IS which we present below.

A shortcoming of SIS is that the weights which are generated along the way can become highly variable. For even moderate T, if one generates N weighted samples using Algorithm 8, it will often be the case that one weight is orders of magnitude larger than all of the other weights. This is known as the *weight degeneracy* problem of importance sampling.

As written, these importance sampling methods require that the densities of π and ν be known exactly, and that ν can be sampled from exactly. While this is true in an interesting range of cases, the condition on π will often be too strong. In complex stochastic problems, it is typically the case that π is only known up to a multiplicative factor, i.e.

$$\pi(x) = \frac{\gamma(x)}{Z}$$

where γ can be evaluated exactly, but Z cannot. In this setting, the intractability of Z implies that one cannot evaluate $w(x)$. A common solution is to use the same samples

from ν to estimate Z, using the fact that

$$Z = \int \gamma(x)dx = \int \nu(x) \left(\frac{\gamma(x)}{\nu(x)} \right) dx.$$

Using $w(x)$ to now denote $\gamma(x)/\nu(x)$, one can apply Algorithm 9, usually known as *Self-Normalised Importance Sampling* (SNIS)[§].

Algorithm 9 Self-Normalised Importance Sampling $X \sim \pi(x) = \gamma(x)/Z$, with proposal ν, N particles

1. For $a = 1, \ldots, N$:
   (a) Sample $x^a \sim \nu$.
   (b) Compute $\hat{w}^a = w(x^a) = \frac{\gamma(x^a)}{\nu(x^a)}$.
2. Compute $\hat{Z} = \frac{1}{N} \sum_{a=1}^{N} w^a$.
3. For $a = 1, \ldots, N$, compute $w^a = \hat{w}^a / \hat{Z}$.
4. Output $(\{(x^a, w^a)\}_{a=1}^N, \hat{Z})$.

One can then estimate expectations under π as

$$\mathbf{E}_\pi[f(x)] \approx \frac{1}{N} \sum_{a=1}^{N} w^a f(x^a)$$

which is a consistent estimator. Further details on the convergence of estimators derived by importance sampling arguments can be found in [Liu08, APSAS17, CD18, HR19].

We note that the idea of SNIS can naturally be fused with Sequential Importance Sampling, i.e. suppose that that we fix a sequence of measures $\{\pi_t(x_{1:t})\}_{t=1}^T$, such that for each t, π_t can be evaluated up to a multiplicative factor, i.e.

$$\pi_t(x_{1:t}) = \frac{\gamma_t(x_{1:t})}{Z_t}$$

where γ_t can be evaluated exactly, Z_t cannot, and $\pi_T = \pi$. We again assume existence of a proposal distribution $q(x_{1:T})$ which can be decomposed into a sequence of proposal distributions $\{q_t(x_t|x_{1:t-1})\}_{t=1}^T$, each of which can be both sampled from and evaluated. One can then apply Algorithm 10.

The applicability of SNIS to situations where π is only available up to a constant represents an important widening of scope for these methods. However, as the method

[§]Note that in this presentation of the algorithm, the weights are scaled such that the typical size of a weight w is of order 1, uniformly in N.

Algorithm 10 Self-Normalised Sequential Importance Sampling $X \sim \pi$ with interpolating sequence $\{\pi_t\}_{t=1}^T$, proposals $\{q_t\}_{t=1}^T$, N particles

1. For $t = 1$,
   (a) For $a = 1, \ldots, N$,
      i. Sample $x_1^a \sim q_1(x_1)$.
      ii. Compute $\hat{w}_1^a = \frac{\gamma_1(x_1^a)}{q_1(x_1^a)}$.
   (b) Set $\hat{Z}_1 = \frac{1}{N} \sum_{a=1}^N \hat{w}_1^a$.
   (c) For $a = 1, \ldots, N$, compute $w_1^a = \hat{w}_1^a / \hat{Z}_1$.
2. For $1 < t \leqslant T$,
   (a) For $a = 1, \ldots, N$,
      i. Sample $x_t^a \sim q_t(x_t | x_{1:t-1}^a)$.
      ii. Set $x_{1:t}^a = (x_{1:t-1}^a, x_t^a)$.
      iii. Compute $\hat{w}_t^a = \hat{w}_{t-1}^a \cdot \frac{\gamma_t(x_t^a)}{\gamma_{t-1}(x_{1:t-1}^a) \cdot q_t(x_t^a | x_{1:t-1}^a)}$.
   (b) Set $\hat{Z}_t = \frac{1}{N} \sum_{a=1}^N \hat{w}_t^a$.
   (c) For $a = 1, \ldots, N$, compute $w_t^a = \hat{w}_t^a / \hat{Z}_t$.
3. Output $(\{(x_{1:T}^a, w_T^a)\}_{a=1}^N, \{\hat{Z}_t\}_{t=1}^T)$.

is still fundamentally an importance sampling approach, at least naively, it cannot be expected to behave well in higher-dimensional spaces, where the variability of the weights will again lead to degeneracy. In this setting, it is useful to take the ideas underlying Algorithms 8 and 9, and fuse them with another idea: *resampling*.

The key idea behind resampling is to consider the weights obtained by (Self-Normalised) Importance Sampling, and use them to focus attention on promising partial samples. For example, let $t \ll T$, and let $x = x_{1:t}$ be a partial sample with very low weight. It is often reasonable to assume that the event of x being evolved into a full sample $x_{1:T}$ which has a large weight has quite low probability, and so it is unlikely to develop into a sample which represents π well. By contrast, if the weight of x is large, then we might hope that it is on track to develop into a sample which is close to π. This intuition is put to use in *Sequential Importance Sampling with Resampling* (SISR), presented as Algorithm 11.

Algorithm 11 Sequential Importance Sampling with Resampling, Targets $\{\pi_t\}_{t=1}^T$, Proposals $\{q_t\}_{t=1}^T$, N particles

---

1. For $t = 1$,
   (a) For $a = 1, \ldots, N$,
      i. Sample $x_1^a \sim q_1(x_1)$.
      ii. Compute $\hat{w}_1^a = \frac{\gamma_1(x_1^a)}{q_1(x_1^a)}$.
   (b) Set $\hat{Z}_1 = \frac{1}{N} \sum_{a=1}^N \hat{w}_1^a$.
   (c) For $a = 1, \ldots, N$, compute $w_1^a = \hat{w}_1^a / \hat{Z}_1$.
2. For $1 < t \leqslant T$,
   (a) For $a = 1, \ldots, N$,
      i. Sample $a_t^a = \text{Categorical}\left(\{\frac{1}{N} w_t^b\}_{b=1}^N\right)$, and set $\hat{x}_{1:t-1}^a = x_{1:t-1}^{a_t^a}$
      ii. Sample $x_t^a \sim q_t(x_t | \hat{x}_{1:t-1}^a)$.
      iii. Set $x_{1:t}^a = (\hat{x}_{1:t-1}^a, x_t^a)$.
      iv. Compute $\hat{w}_t^a = \frac{\gamma_t(x_t^a)}{q_t(x_t^a | \hat{x}_{1:t-1}^a)}$.
   (b) Set $\hat{Z}_t = \hat{Z}_{t-1} \cdot \left(\frac{1}{N} \sum_{a=1}^N \hat{w}_t^a\right)$.
   (c) For $a = 1, \ldots, N$, compute $w_t^a = \hat{w}_t^a / \hat{Z}_t$.
3. Output $(\{(x_{1:T}^a, w_T^a)\}_{a=1}^N, \{\hat{Z}_t\}_{t=1}^T)$.

In SISR, as in SIS, one maintains a weighted collection of N particles, such that for

any $t \leqslant T$, one has the approximation

$$\pi_t(x_{1:t}) \approx \frac{1}{N} \sum_{a=1}^{N} w_t^a \delta(x_{1:t}^a, dx_{1:t}).$$

The distinction with SIS comes from the resampling step at the beginning of each time step. The resampling step takes as input a weighted system of N distinct particles, and outputs an equally-weighted system of $\leqslant N$ particles, possibly with repetitions. The downside of resampling is the loss of diversity in particles which it causes, but the upside is that each of the remaining particles is, in some sense, 'equally important'. In particular, this behaviour leads to more stable weights, mitigating the weight degeneracy problem and leading to more reliable estimates. It is in general hard to rigorously characterise the circumstances under which SISR should be strictly preferred to plain SIS (see [JD08] for some discussion on 'optimality' within certain forms of SISR), but in practice, it is well-understood that for challenging, high-dimensional models, resampling is extremely useful.

We note that SISR is typically better-known by the name *Sequential Monte Carlo* (SMC, see e.g. [Smi13]). SMC was historically developed for applications in filtering of state-space models [GSS93, Kit96, DM97], but over time, it has become clear that the SMC methodology is a 'first-class citizen' in the toolbox of sampling techniques, with applications which extend far beyond filtering. For this reason, our presentation has thus focused more on the generality of the SMC framework, rather than its historical roots in the context of state-space models. We refer the interested reader to [DMDJ06, NLS14, LJN$^+$17, NLS19] for some illustrative examples of how this general framework can be applied to a diverse range of tasks.

1.2.3 Iterative Sampling

Direct sampling methods make strong assumptions about the target distribution, and produce exact samples. Importance sampling methods make weaker structural assumptions, and by producing samples which are not drawn from the true target, provide an incomplete if useful solution. The question stands as to whether, for a sufficiently wide class of probability measures π, there is a practical algorithmic solution which can generate samples which are distributed according to π.

The technique of *Markov Chain Monte Carlo* (MCMC) is aimed at closing this gap. MCMC accepts that while a 'true exact sample' might be out of reach, an approximate sample is still useful, particularly if one can iteratively process that approximate sample in a way which reduces the approximation error. At a high level, this is the approach of MCMC: begin by drawing a sample from some distribution and iteratively whittle away

the inexactness, thus pushing it closer in law to a true draw from π. The promise is that these iterations will get you close enough to π *eventually*; the cost is in just how long 'eventually' might take.

This approach seems appealing enough in principle, provided that one can find an appropriate iterative scheme for transforming samples. The usual perspective on this task is to find some stochastic channel or *Markov kernel* K, such that if one inputs an exact sample from π to the channel, then the output of the channel is also marginally a sample from π. That is, as measures,

$$\int_{x \in \mathcal{X}} \pi(x) K(x \to y) dx = \pi(y), \tag{1.1}$$

or more informally[¶], $\pi K = \pi$. If this condition holds, we say that the Markov kernel K leaves π invariant, or that K admits π as an invariant measure.

We have thus transformed the task of sampling from a probability measure into writing down a fixed-point equation which is solved by π. Provided that the Markov kernel K is contractive around π in some suitable sense, the route is clear: at the level of measures, start with some initial measure ν_0, evolve it by applying the kernel, i.e. iterate $\nu_t = \nu_{t-1} K$ for $t \geqslant 1$, and then hope that ν_t converges to π as $t \to \infty$. This can be made more concrete by viewing the iteration at the level of samples, where it corresponds to drawing a sample $x_0 \sim \nu_0$, and then sequentially generating a Markov chain, by iteratively drawing $x_t \sim K(x_{t-1} \to x_t)$. Under this procedure, x_t is then marginally distributed according to ν_t, and so it becomes closer and closer in law to π.

Algorithm 12 Markov Chain with Transition Kernel K, Initial Measure ν_0

1. Sample $x_0 \sim \nu_0$.
2. For $t \geqslant 1$, sample $x_t \sim K(x_{t-1} \to x_t)$.

The challenge is then to identify Markov kernels K which i) can be implemented practically, ii) admit π as an invariant measure, iii) actually converge towards π in the limit, and iv) ensure that this convergence is as fast as possible. Points i), ii), and iii) are essentially qualitative in nature; whereas point iv) is a quantitative demand, emphasising that the practical efficiency of these algorithms is characterised by the speed at which ν_t converges to π.

It is this strategy which will be pursued for the remainder of this thesis. We focus our attention on MCMC in particular because it is applicable under very general circumstances, it is robust to changes in model structure, and it has shown itself to be consistently capable

[¶]In this section, we will use somewhat 'colloquial' notation for measures, Markov kernels, and so on; for a proper measure-theoretic treatment of these notions, see e.g. [MT12].

of providing useful solutions in the context of complex, high-dimensional models across a range of application domains. It holds the promise of dynamically exploring probability measures of interest in principle, and is broadly feasible in practice.

1.3 Iterative Exploration, and MCMC

We begin the presentation of MCMC algorithms by outlining how one can develop practically-feasible Markov kernels K which admit a prescribed π as an invariant measure. By 'practically-feasible', we mean that it is possible to draw exact samples from the distribution $K(x \to dy)$ for any given x in a reasonable amount of time. A priori, this should not necessarily be easy, as the condition given in Equation (1.1) involves calculating an integral under π, which is generally challenging.

1.3.1 Reversibility

A first step towards a more tractable problem is to demand a condition which is more stringent than (1.1), but which is easier to verify for a given (π, K). In particular, it is desirable to identify a condition which depends only on *local* properties of (π, K). One such condition is that (π, K) satisfy

$$\pi(x)K(x \to y) = \pi(y)K(y \to x) \quad \text{for all } x, y \in \mathcal{X}. \tag{1.2}$$

When this holds, we say that K is π-reversible, or that (π, K) together satisfy *detailed balance*. We can see that this condition implies (1.1) by noting that if (1.2) holds, then

$$\begin{aligned}\int_{x\in\mathcal{X}} \pi(x)K(x \to y)dx &= \int_{x\in\mathcal{X}} \pi(y)K(y \to x)dx \\ &= \pi(y)\int_{x\in\mathcal{X}} K(y \to x)dx \\ &= \pi(y).\end{aligned}$$

As such, we have a condition on the pair (π, K) which implies that K admits π as an invariant measure, and is easily-checkable in practice.

We comment briefly that because we have narrowed our search space to considering only *reversible* kernels K, it might be natural to worry that we are missing out on something, i.e. that there could exist non-reversible kernels with nicer convergence properties. While the latter point is true, the former is somewhat subtler: by first understanding reversible kernels, it in fact becomes easier to systematically construct these non-reversible kernels. Indeed, compositions of π-reversible kernels need not be π-reversible, in much the same way that products of self-adjoint matrices need not be self adjoint. Fortunately, for our purposes, reversibility is just the means to the ends of π-invariance, and while reversibility isn't conserved under composition, π-invariance is. As such, one can easily construct non-reversible, π-invariant kernels simply by composing simple reversible kernels. With few exceptions, this is actually how most practical non-reversible kernels are constructed,

a point which will be discussed at length in Section 1.6 of this chapter. As such, we assure the reader that it is sufficient to initially focus our attention on reversible kernels.

Equipped with reversibility, we can now properly begin our search for practical kernels K. We will first present two elementary techniques for deriving π-reversible kernels, which can then be used as building blocks for constructing more advanced methods. We note preemptively that although each of the two techniques can, in principle, be viewed as a special case of the other (see e.g. [Fin15] for a discussion of this point, as well as several other similar connections arising in Monte Carlo), it is instructive to present them separately, as the philosophies behind them are in a sense quite distinct, and it is useful to be able to apply each mindset separately in a given scenario.

1.3.2 Gibbs Sampling

The first technique is known as 'Gibbs Sampling' (see e.g. [GG84, SR93, CG92] for its introduction, and early applications to Bayesian statistics), and is fundamentally a model-centric approach to navigating the target measure. At its heart, Gibbs sampling relies on the availability of

1. A meaningful partitioning of the variable $x = (x_i)_{i \in I}$, and
2. A tractable collection of conditional distributions, i.e. for $i \in I$, it must be possible to draw samples from $\pi\left(x_i | \{x_j\}_{j \in I \setminus \{i\}}\right) = \pi(x_i | x_{-i})$.

Note that the latter condition is far weaker than what is required of the earlier recursive algorithms (e.g. Algorithms 5, 6), as one only requires one-dimensional distributions, conditioned on **all** other variables. Going forward, we will denote the i^{th} complete conditional distribution, conditioned on $X_{-i} = x_{-i}$, by $\pi_i(\cdot | x_{-i})$.

For a given $i \in I$, consider now the Markov transition defined by

1. Sample $y_i \sim \pi_i(\cdot | x_{-i})$.
2. Set $y = (x_{-i}, y_i)$.

This corresponds to replacing the i^{th} coordinate of x by a random draw from its full conditional distribution. The associated transition kernel can then be written as

$$K_i(x \to dy) = \delta(x_{-i}, dy_{-i}) \cdot \pi_i(dy_i | x_{-i}).$$

One can then compute that

$$\begin{aligned}
\pi(dx) \cdot K_i(x \to dy) &= \pi(dx) \cdot \delta(x_{-i}, dy_{-i}) \cdot \pi_i(dy_i|x_{-i}) \\
&= \pi(dx_{-i}) \cdot \pi_i(dx_i|x_{-i}) \cdot \delta(x_{-i}, dy_{-i}) \cdot \pi_i(dy_i|x_{-i}) \\
&= \pi(dx_{-i}) \cdot \delta(x_{-i}, dy_{-i}) \cdot \pi_i(dx_i|x_{-i}) \cdot \pi_i(dy_i|x_{-i}) \\
&= \pi(dy_{-i}) \cdot \delta(y_{-i}, dx_{-i}) \cdot \pi_i(dx_i|y_{-i}) \cdot \pi_i(dy_i|y_{-i}) \\
&= \pi(dy_{-i}) \cdot \pi_i(dy_i|y_{-i}) \cdot \delta(y_{-i}, dx_{-i}) \cdot \pi_i(dx_i|y_{-i}) \\
&= \pi(dy) \cdot K_i(y \to dx),
\end{aligned}$$

and hence deduce that K_i is π-reversible. This is good news: if our target measure admits tractable full conditional distributions, then we have a collection of Markov kernels $\{K_i\}_{i \in I}$ which each leave π invariant. As highlighted earlier, although a composition of π-reversible Markov kernels needn't be reversible, a composition of π-invariant Markov kernels still is, and so composing the K_i still gives us a π-invariant Markov kernel. This gives rise to the *Systematic Scan Gibbs Sweep*, as presented in Algorithm 13.

Algorithm 13 Systematic Scan Gibbs Sweep for π, $I = \{1, \ldots, M\}$, start at x^0

1. For $i = 1, \ldots, M$,
   (a) Sample $x^i \sim K_i(x^{i-1} \to x^i)$.
2. Output x^M.

The term 'sweep' here refers to the fact that the update passes over each variable x_i at least once. There are also alternative protocols for such sweeps, e.g. one can consider alternative deterministic orderings of the variables, or one can even randomise the order in which the variables are updated, leading to the *Random Scan Gibbs Sweep*. Perhaps interestingly, there is no universal dominance between the two options; sometimes a given Systematic Scan is to be preferred, and sometimes a given Random Scan is to be preferred; see e.g. [LWK95, LC06, RR15, HDSMR16, MM17] for details, theoretical results, and further discussion on this topic. We note also that the Systematic Scan Gibbs sampler can also be made reversible relatively easily by symmetrising the process, i.e. update the variables in the order $1, 2, \ldots, (M-1), M, (M-1), \ldots, 2, 1$.

The significance of Gibbs sampling from the perspective of algorithm development is its fundamental modularity. It provides assurance that one can construct a procedure which iteratively performs only local updates (which are typically easier to both reason about and program), and still end up with a π-invariant Markov chain. Even when applying algorithms which are superficially quite different to Gibbs sampling, this concept may still be at work, as identifying and exploiting tractable structure within a complex,

high-dimensional measure is a key tool for building efficient algorithms.

1.3.3 Metropolis-Hastings Updates

The second technique is a device known as the *Metropolis-Hastings filter* (partially derived in [MRR+53], and then elaborated into its modern form in [Has70]), and in contrast to the Gibbs sampler, is comparatively model-agnostic as a construction. The essence of the MH filter is that one begins with some 'proposal' kernel q, which usually does not already admit π as an invariant measure, and then critiques the moves proposed by q according to a stochastic gating procedure. That is, when the chain is situated at x_t, the proposal kernel generates a candidate location $y_t \sim q(x_t \to y_t)$, and then one takes $x_{t+1} \in \{x_t, y_t\}$ according to a randomised decision rule.

Denoting the probability of 'accepting' a proposed move from x to y as $\alpha = \alpha(x \to y) \in [0, 1]$, one can write down the transition kernel corresponding to the above procedure as

$$\begin{aligned} K(x \to dy) &= q(x \to dy) \cdot \alpha(x \to y) + (1 - \bar{\alpha}(x)) \cdot \delta(x, dy) \\ \bar{\alpha}(x) &= \int_{y \in \mathcal{X}} q(x \to dy) \cdot \alpha(x \to y). \end{aligned}$$

$\bar{\alpha}(x)$ here represents the average rate at which proposed moves away from x are accepted, and is generally not known explicitly.

In order to guarantee that this kernel generates a π-invariant Markov chain, we demand that K be π-reversible. This will allow us to derive an expression for α, and hence an implementable algorithm. Writing down Equation (1.2) for $y \neq x$ (noting that it holds trivially for $y = x$), we obtain

$$\begin{aligned} \pi(x)K(x \to y) &= \pi(y)K(y \to x) \\ \pi(x)q(x \to y) \cdot \alpha(x \to y) &= \pi(y)q(y \to x) \cdot \alpha(y \to x) \\ \alpha(x \to y) &= \alpha(y \to x) \cdot \frac{\pi(y)q(y \to x)}{\pi(x)q(x \to y)}. \end{aligned}$$

Now, noting that we must have $\alpha \in [0, 1]$, we can deduce that

$$\alpha(x \to y) \leqslant \alpha^{\mathrm{MH}}(x \to y) = \min\left(1, \frac{\pi(y)q(y \to x)}{\pi(x)q(x \to y)}\right),$$

and in fact, taking $\alpha = \alpha^{\mathrm{MH}}$ is sufficient for (1.2) to hold. Moreover, although there are other valid choices of α which ensure π-reversibility, it was shown by [Pes73] that whenever α^{MH} can be computed, then it should be preferred. See e.g. [GŁR17, VGŁR20] for some specific applications in which α^{MH} is **not** available, and so other acceptance probabilities

should be considered.

It should be emphasised that the generality of the Metropolis-Hastings filter is quite remarkable, in terms of how little it demands of the user; the proposal kernel q can have nothing in particular to do with the structure of the target π, and the algorithm will still generate a π-invariant Markov chain. Of course, for practical purposes, it is important to design q judiciously (and approaches to doing so will be discussed in subsequent sections) to ensure favourable convergence behaviour, but even without that, the Metropolis-Hastings filter provides you with a foot in the door, a Markov chain with π as an invariant measure, and some hope of being able to approximately sample from an otherwise-intractable target measure. Pseudocode for a generic Metropolis-Hastings chain is presented in Algorithm 14.

Algorithm 14 Metropolis-Hastings Update for π, with proposal q

1. At x, propose a move to $y \sim q(x \to y)$.
2. Compute $r(x \to y) = \frac{\pi(y)q(y \to x)}{\pi(x)q(x \to y)}$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output y; otherwise, output x.

1.3.4 Convergence of Markov Chains

Before proceeding to more advanced algorithms, it is worth pausing to discuss the gap between constructing i) a Markov chain which leaves π invariant, and ii) a Markov chain which converges rapidly to π. In a sense, leaving π invariant only ensures that if the chain is initially distributed according to π, then it retains that property; it does not ensure that it gets closer to π if you started somewhere else. For example, the 'do nothing' Markov kernel $K(x \to y) = \delta(x, dy)$ admits any measure as an invariant measure, but will not converge to anything useful for nondegenerate targets. The purpose of the exposition which follows is thus to highlight some obstructions which can prevent a Markov chain from converging, rather than to give exposition to tools for showing convergence. When designing Markov chains for practical applications, the first step is to avoid falling into traps which will impede asymptotic convergence; this section is intended to help the reader avoid such traps.

One potential obstruction to convergence is known as *reducibility*. If the invariant measure π is supported on a set $\mathcal{S}$, it is important that, from any starting point x, the Markov chain be able to reach all of $\mathcal{S}$, eventually. If the chain is ultimately 'trapped' in some set $\mathcal{S}' \not\subseteq \mathcal{S}$, we call the chain *reducible*; if not, the chain is *irreducible*. To be more

precise, for $t \geqslant 1$, define the iterated Markov kernel K^t recursively by

$$K^t(x \to dy) = \int_{z \in \mathcal{X}} K^{t-1}(x \to dz) K(z \to dy).$$

K^t is the Markov kernel corresponding to t steps of the Markov chain defined by K. Given a positive measure φ on the space $\mathcal{X}$, we say that K is φ-irreducible if for all $x \in \mathcal{X}$ and for all set A such that $\varphi(A) > 0$, there exists a $t = t(x, A) > 0$ such that $\int_{y \in A} K^t(x \to dy) > 0$. Note that this is a *qualitative* statement, i.e. it states that the chain *can* enter a certain set eventually, and says nothing about how long this might take.

Irreducibility of a Markov chain is essentially sufficient to guarantee that the chain has a unique invariant measure (one can find details to this effect in e.g. [Gey98, MT12]). As such, if we run a Markov chain driven by some irreducible, π-invariant Markov kernel K, and the chain converges in law to some probability measure μ, then it must be the case that $\mu = \pi$.

The next obstruction is guaranteeing whether the chain does indeed converge to something. The barrier which one should have in mind here is that even though the chain might be able to reach any set eventually, there could be a positive probability that this never actually happens. For example, it is known that the simple random walk on $\mathbf{Z}^d$ is *transient* for $d \geqslant 3$, i.e. for any $x \in \mathbf{Z}^d, t > 0$,

$$\mathbf{P}\left(X_s = x \text{ for some } s \geqslant t\right) < 1.$$

Thus, with some positive probability, the walk never returns to the site x. Similar notions can be defined on general state spaces, corresponding to behaviour which is qualitatively similar to 'drifting off to infinity'.

A Markov chain is said to be *φ-recurrent* if it is not transient, i.e. if *every* set A of positive measure under φ is visited infinitely often. It is said to be *Harris recurrent* if there exists a 'small set' A and a probability measure ν such that

1. The hitting time

   $$\tau_A = \min\{t \geqslant 0 : X_t \in A\}$$

   satisfies $\mathbf{P}(\tau_A < \infty | X_0 = x) = 1$ for all $x \in \mathcal{X}$, and

2. For some $\lambda \in (0, 1), m \geqslant 1$, and for all $x \in A$, and $B \subset \mathcal{X}$, it holds that

   $$\int_{y \in B} K^m(x \to dy) \geqslant \lambda \cdot \nu(B).$$

The first condition ensures that the chain cannot drift off to infinity indefinitely, as it

must always return to the set A eventually; it is in some sense 'anchored' at A. The second condition ensures that from any location $x \in A$, the transition kernels K all look somewhat similar, and so the behaviour of the chain inside this set is relatively homogeneous. One should view A as being something like the 'bulk' of the measure π; as a hub to which the chain will return regularly.

Under the assumption of Harris recurrence, aperiodicity, and the following additional assumption on average hitting times

$$\sup_{x \in A} \mathbf{E}_x[\tau_A] < \infty,$$

it can be shown that the resulting Markov chain has a unique invariant measure, which it converges to. Note that on its face, this is again a qualitative result, though there is implicit quantitative information included in the constant λ.

Finally, one comes to the question of quantitative convergence of Markov chains to their equilibrium distribution. In cases of interest, for time-homogeneous Markov chains, one cannot expect better than exponential rates of convergence, i.e. if $\delta_x K^t$ is the law of a π-invariant Markov chain started at x and run for t steps, one cannot hope for faster convergence than

$$d\left(\delta_x K^t, \pi\right) \leqslant c(x) \cdot \exp(-\lambda t) \tag{1.3}$$

for some $c(x) \in (0, \infty), \lambda > 0$, where d is some appropriate metric on the space of probability measures (for example, it is common to work with the *total variation* metric, or the *Wasserstein / transport* metric; see [Dal17] for definitions of both.). If this holds, we say that the chain is *exponentially ergodic* (sometimes 'geometrically ergodic'). If the prefactor $c(x)$ is bounded from above uniformly in x, then the term 'uniformly ergodic' is used. These titles are essentially the gold standard for convergence of MCMC algorithms, and allow for a number of useful corollaries to be deduced, e.g. Laws of Large Numbers, Central Limit Theorems, concentration inequalities, etc. for functionals of the path of the chain. These are of particular interest in the context of Monte Carlo integration, where path functionals are used as a proxy for computing integrals under π.

Without probing into the precise mathematical details, we will quickly sketch what can cause an ergodic Markov chain to fail to be exponentially ergodic. One potential issue is the existence of poorly-behaved parts of the target measure, in which the chain can take a long time to move even a small distance. For example, suppose we are using a Metropolis-Hastings chain, and the state contains sets in which the average acceptance probability gets arbitrarily small. In these situations, ergodicity implies that the chain will *eventually* leave these sets, but that the exit times will become hard to control, e.g. they may have infinite expectation. This is generally a symptom of the Markov kernel

being poorly-adapted to the target measure, and tends to arise for very inhomogeneous targets. Some instructive examples of this can be found in [Liv15].

Another issue can be when the chain continues to make moves of a reasonable size, but is poorly-confined, e.g. the chain makes excursions into the tails of the distribution, and takes a long time to turn around and return to the 'bulk' of the distribution. This is relatively common when running the Metropolis-Hastings algorithm with local proposals on heavy-tailed targets, as the resulting Markov chain essentially devolves into a random walk when it ventures into the tails. Further details and intuition on this point can be found in [JH00, JT03, JR07]. While this can to some extent be ameliorated by the use of tailored proposal mechanisms, this issue is often symptomatic of the target measure being quite diffuse in its current parametrisation.

Broadly speaking, a Markov chain will tend to exhibit geometric ergodicity when i) the target measure has tails which are not too heavy (lighter than exponential tends to be sufficient), ii) the target measure is not too rough or inhomogeneous (in terms of the length scales on which the density varies), and iii) the chain is able to successfully make moves of a fixed magnitude, uniformly across the space, with reasonable probability. Pathologies in the convergence of Markov chains can often be identified through the failure of one of the above three conditions.

Finally, we note that it is of substantial interest, both theoretically and practically, to undertake a *quantitative* study of exponential ergodicity, i.e. to identify explicit values of λ such that Equation 1.3 holds. This is a challenging task, which almost always requires problem-specific tools, particularly when seeking sharp estimates of λ. We will not comment on this further in this work, and refer the interested reader to [Dal17, MS17, CCBJ18, DCWY19, SL19] for some recent developments on the topic.

1.4 Design and Implementation of Practical MCMC Algorithms

In designing a practical MCMC algorithm for sampling from a given target measure, there are a number of considerations which need to be taken into account. In this section, we outline one possible workflow for identifying and resolving these considerations in sequence. At a high level, the workflow is as follows:

1. Identify a class of moves which can be used to navigate the space on which the target measure resides.
2. Identify an ideal sub-class of moves which is well-adapted to the specific target measure.
3. Devise a practically-feasible approximation to these ideal moves, and determine how to satisfactorily resolve any approximation errors which are induced in doing so.
4. Having specified the set of moves from which the Markov chain will be constructed, detail how this chain will be simulated on a computer.
5. Finally, consider how any free parameters in the definition of the algorithm i) would be set ideally, and ii) should be tuned in practice.

This workflow emphasises some particular elements of the design process, and in many applications, some of these steps will be considerably more involved than others. For example, in intractable likelihood models (broadly interpreted), there may be little room for creativity in the first two steps, whereas the latter three necessitate far more subtle care and attention. By contrast, in high-dimensional models on continuous spaces with explicit, differentiable densities, the first three steps will often be comparatively important, and the latter two might be more automatic. Nevertheless, this framing is introduced as a strategy for highlighting the distinct phases of algorithm design, with the hope that the decisions made in each phase can, to some extent, be decoupled and resolved separately.

1.4.1 Phase 1: Markov Processes for Navigating a Space

When seeking a class of moves with which to navigate a given space, the first step is to catalog the types of motion which are well-defined on the space. When considering Markov processes, there are essentially only a few things which a single particle exploring a space can do, which are

- jump randomly from one place to another

- drift deterministically along the flow of some vector field
- diffuse randomly around the current location

and superpositions of these. We will focus on *continuous-time* processes, as in this setting, the distinction between the three types of motion is most unambiguous.

On a discrete space, these notions all essentially coincide, but it is standard to view a Markov process on a discrete space as a *Markov Jump Process* (MJP), i.e. the path of a particle which stays at each state x for a random duration $T \sim \text{Exp}(\Lambda(x))$ and then jumps randomly to one of its neighbours $y \sim q(x \to y)$. We call $\Lambda(x)$ the *total jump rate* out of x, and $q(x \to y)$ is the *transition probability* from x to y. It is also sometimes useful to define $\lambda(x \to y) = \Lambda(x)q(x \to y)$, the *jump rate* from x to y.

On a continuous space, MJPs can take many different flavours. Three which are of particular interest are i) global jumps, ii) coordinate-wise global jumps, and iii) local jumps.

A global jump process is specified by a *jump rate* function Λ, and a *jump distribution* ν, and can be implemented by Algorithm 15. Such processes are memoryless in a very strong sense; their next location is independent of even the chain's current location.

Algorithm 15 Global Jump Process

1. at x,
   (a) Sample $T \sim \text{Exp}(\Lambda(x))$.
   (b) Sample $y \sim \nu$.
   (c) Stay at x for time T, and then jump to y.

By contrast, a coordinate-wise global jump process is specified by a collection of coordinate-wise jump rate functions $\{\lambda_i\}_{i=1}^d$, and a collection of coordinate-wise jump distributions $\{\nu_i\}_{i=1}^d$. They can be implemented by Algorithm 16. These models can be viewed as an abstraction of the idea behind Gibbs sampling, i.e. make moves by changing one coordinate at a time, but with the option to make more general moves in that coordinate.

Finally, a local jump process can be specified by a jump rate function Λ as before, as well as some transition kernel $q(x \to y)$ which is 'local' to x, e.g. $q(x \to y) = \mathcal{U}(dy|\{y : |y - x| < r\})$, where $\mathcal{U}(dx|\mathcal{S})$ is the normalised uniform measure on the set $\mathcal{S}$. One then implements such a process via Algorithm 17. This is distinguished from the previous two classes of MJP by the implication that q should depend explicitly on the current location of the chain, x.

Markov Jump Processes are illustrative to study in the context of simulation, as they exist on very general spaces, which allows for their analysis to be relatively agnostic to

Algorithm 16 Coordinate-Wise Global Jump Process

1. at x,
   (a) Compute $\Lambda(x) = \sum_{i=1}^{d} \lambda_i(x)$.
   (b) Sample $T \sim \text{Exp}(\Lambda(x))$.
   (c) Set $I = i$ with probability $\lambda_i(x)/\Lambda(x)$.
   (d) Sample $y_i \sim \nu_i(y_i|x_{-i})$, and write $y = (x_{-i}, y_i)$.
   (e) Stay at x for time T, and then jump to y.

Algorithm 17 Local Jump Process

1. at x,
   (a) Sample $T \sim \text{Exp}(\Lambda(x))$.
   (b) Sample $y \sim q(x \to y)$.
   (c) Stay at x for time T, and then jump to y.

the details of the space in question. Moreover, given a discrete-time Markov chain, one can extend it to a continuous-time MJP by subordinating the chain to a unit-rate Poisson process, i.e. to include a random exponentially-distributed holding time in between jumps of the chain, while making the same moves. This is a useful trick, as certain analytical computations in the study of Markov processes are simplified when working in continuous time, relative to discrete-time Markov chains.

Beyond MJPs, continuous spaces can also accommodate dynamics with *drift* and *diffusion*. Drift is best understood in terms of *Ordinary Differential Equations* (ODEs), i.e. the particle follows the motion of some vector field b according to the dynamics $dx = b(x)dt$. As the solutions of ODEs are deterministic given their initial conditions, it is rare to use them for sampling in isolation, and so one typically injects some form of stochasticity into the system to generate exploration. While this can be done by adding in jumps (a hybrid which is known as a *Piecewise-Deterministic Markov Process*, or PDMP), a more common route is to perturb the path of the ODE with additive noise. This gives rise to a *Stochastic Differential Equation* (SDE). An SDE is specified not only by a drift function b, but also a *diffusion coefficient* σ, with which one writes

$$dx = b(x)dt + \sigma(x)dW,$$

where dW represents a stochastic noise term which perturbs the motion of the particle. This should be interpreted approximately as saying that over a small time-scale h, a

particle moves from x to $x + h \cdot b(x) + \sqrt{h} \cdot \sigma(x) \cdot \xi$, where $\xi \sim \mathcal{N}(0, I)$. This can be given a more rigorous interpretation (see e.g. [Pav14] for a practical introduction). A desirable feature of SDEs is that their evolution is purely *local* in nature, which makes their approximate simulation less challenging than jump processes. Moreover, when the diffusion matrix $\Sigma(x) = \sigma(x)\sigma(x)^T$ is full rank for all x, the transition probabilities of an SDE (and of most standard numerical approximations thereof) admit a density, which opens the doors to their use in Metropolis-Hastings algorithms.

Finally, we note that all three of these ingredients can be combined, giving rise to the so-called 'Jump SDEs'. Strictly speaking, even this does not fully cover the entire world of Markov processes (we have, for example, neglected to discuss Levy processes; see [App09] for an account), but it is fair to say that the vast majority of practical simulation algorithms are based around the core three elements of jumps, drift, and diffusion. We refer to [Sim17, SZTG20] for a couple of interesting, rare exceptions.

1.4.2 Phase 2: Markov Processes of a Given Invariant Measure

Having provided a rough taxonomy of Markov processes, the next task is to identify which of these would sample correctly from our target measure π *in principle*, deferring concerns of computation for the time being.

A key tool in establishing that a continuous-time Markov process admits a given invariant measure is to study the so-called *infinitesimal generator* of the process (which we will simply refer to as the 'generator'). Given a time-homogeneous Markov process X_t, the generator $\mathcal{L}$ is usually defined by its action on smooth, bounded functions f as

$$(\mathcal{L}f)(x) \triangleq \lim_{t \to 0^+} \frac{\mathbf{E}[f(X_t)|X_0 = x] - f(x)}{t}.$$

One can then extend the action of $\mathcal{L}$ to the space of all functions for which the above limit exists, the *domain* of $\mathcal{L}$, usually written as $\mathcal{D}(\mathcal{L})$. In all of the cases which we will consider, $\mathcal{L}$ acts linearly on functions. For ODEs, and SDEs, $\mathcal{L}$ takes the form of a first- and second-order differential operator respectively, whereas for MJPs, $\mathcal{L}$ takes the form of an integral operator. For processes which combine the behaviour of several dynamical processes, the generator is formed by additive superposition, e.g. a Piecewise-Deterministic Markov Process (PDMP) consists of jumps and deterministic drift, and so will have a generator which is the sum of an integral operator (corresponding to the jumps) and a first-order differential operator (corresponding to the drift).

The significance of the generator is that provides us with a tool for checking algebraically whether the process admits π as an invariant measure. The following can be found in e.g. [EK09].

Theorem 1. *Let $\mathcal{L}$ be the generator of a time-homogeneous Markov process on some space $\mathcal{X}$, and let π be a probability measure on $\mathcal{X}$.*

1. *If, for all $f \in \mathcal{D}(\mathcal{L})$, it holds that*

$$\mathbf{E}\left[(\mathcal{L}f)(x)\right] = 0 \tag{1.4}$$

   then the Markov process admits π as an invariant measure.

2. *Moreover, if for all $f, g \in \mathcal{D}(\mathcal{L}) \cap L^2(\pi)$, it holds that*

$$\mathbf{E}\left[(\mathcal{L}f)(x)g(x)\right] = \mathbf{E}\left[f(x)(\mathcal{L}g)(x)\right] \tag{1.5}$$

   then the Markov process is also π-reversible.

The utility of this condition is that it provides us with a way of establishing an analytic property of the Markov process (its invariant measure) by algebraic manipulation. In particular, for ODEs and SDEs, verifying either of Equations 1.4 or 1.5 will boil down to an exercise in integration by parts. As such, with minor exceptions for the occasional more-exotic Markov process, the generator method is typically the standard approach for verifying[‖] that a given Markov process admits the invariant measure which we desire.

To apply this method, we begin by studying Markov Jump Processes. Given an MJP with jump rates $\lambda(x \to y) = \Lambda(x)q(x \to y)$, its generator can be computed as

$$\begin{aligned}\mathcal{L}f(x) &= \int \lambda(x \to y)\left[f(y) - f(x)\right] dy \\ &= \left(\int \lambda(x \to y) f(y) dy\right) - \Lambda(x) f(x).\end{aligned}$$

Note that $\mathcal{L}$ is an integral operator, and in particular, acts non-locally on f. A particular consequence of this is that it becomes difficult to check that an MJP admits π as an invariant measure when it is not π-reversible. More precisely, to show π-invariance, we would want to show that $\mathbf{E}_\pi\left[\mathcal{L}f(x)\right] = 0$ for all f, and so we compute

$$\begin{aligned}\mathbf{E}_\pi\left[\mathcal{L}f(x)\right] &= \int \pi(x)\left(\int \lambda(x \to y) f(y) dy - \Lambda(x) f(x)\right) dx \\ &= \int\int \pi(x)\Lambda(x) q(x \to y) f(y) dy dx - \int \pi(x)\Lambda(x) f(x) dx.\end{aligned}$$

Now, supposing with mild[**] loss of generality that we can rescale the jump rates such that

[‖] We comment here that characterising $\mathcal{D}(\mathcal{L})$ is not always easy, and so a full justification that a given process admits a given invariant measure can become technical. However, for heuristic calculations, one tends to ignore such technicalities, which while non-rigorous, can nonetheless be useful as a sanity check.

[**] In principle, this expectation could be infinite. This is typically not the case for well-behaved jump processes, and is especially atypical for practical samplers.

$\mathbf{E}_\pi[\Lambda(x)] = 1$, we can apply Bayes' rule to the joint distribution $\Pi(x,y) = \pi(x)\Lambda(x)q(x \to y)$ to assert the existence of probability measures $\pi^{\Lambda,q}(y)$ and $q^{\mathrm{T}}_{\pi,\Lambda}(y \to x)$ such that

$$\pi(x)\Lambda(x)q(x \to y) = \pi^{\Lambda,q}(y)q^{\mathrm{T}}_{\pi,\Lambda}(y \to x).$$

We thus continue with our previous development:

$$\begin{aligned}
&\int\int \pi(x)\Lambda(x)q(x \to y)f(y)dydx - \int \pi(x)\Lambda(x)f(x)dx \\
&= \int\int \pi^{\Lambda,q}(y)q^{\mathrm{T}}_{\pi,\Lambda}(y \to x)f(y)dydx - \int \pi(x)\Lambda(x)f(x)dx \\
&= \int \pi^{\Lambda,q}(y)f(y)dy - \int \pi(x)\Lambda(x)f(x)dx \\
&= \int \left(\pi^{\Lambda,q}(x) - \pi(x)\Lambda(x)\right) f(x)dx,
\end{aligned}$$

from which we can deduce that the MJP leaves π invariant precisely when $\pi^{\Lambda,q}(x) = \pi(x)\Lambda(x)$. While true, this condition is of limited use in most practical scenarios, as characterising $\pi^{\Lambda,q}$ requires the calculation of integrals.

As such, for the remainder of this section, we will focus our attention only on MJPs which are reversible with respect to the desired target measure. Fortunately, this scenario is far more tractable. Noting from the earlier discussion that the MJP will be π-reversible precisely when $\mathbf{E}[(\mathcal{L}f)(x)g(x)] = \mathbf{E}[f(x)(\mathcal{L}g)(x)]$, we compute

$$\begin{aligned}
\mathbf{E}[(\mathcal{L}f)(x)g(x)] &= \int \pi(x)(\mathcal{L}f)(x)g(x)dx \\
&= \int \pi(x)\left(\int \lambda(x \to y)f(y)dy - \Lambda(x)f(x)\right)g(x)dx \\
&= \int\int \pi(x)\lambda(x \to y)f(y)g(x)dydx - \int \pi(x)\Lambda(x)f(x)g(x)dx \\
&= \int\int \pi(y)\lambda(y \to x)f(x)g(y)dydx - \int \pi(x)\Lambda(x)f(x)g(x)dx
\end{aligned}$$

Performing a similar calculation for $\mathbf{E}[f(x)(\mathcal{L}g)(x)]$, we see that this equality will hold iff for all $f, g \in \mathcal{D}(\mathcal{L})$,

$$\int\int \pi(y)\lambda(y \to x)f(x)g(y)dydx = \int\int \pi(x)\lambda(x \to y)f(x)g(y)dydx \tag{1.6}$$

$$\iff \pi(x)\lambda(x \to y) = \pi(y)\lambda(y \to x) \quad \text{for all } x, y. \tag{1.7}$$

We can now use this to design MJPs with invariant measure π.

As a first example, consider a global jump process, where the jumps are drawn from some fixed measure ν, and the jump rate is to be determined. The condition given

in equation 1.6 implies that we must have $\pi(x)\Lambda(x)\nu(y) = \pi(y)\Lambda(y)\nu(x)$, and hence $\Lambda(x) \propto \nu(x)/\pi(x)$. The intuition is as follows: if x is more likely to have been drawn from π than from ν, then it is a good point, and we should remain there longer; hence $\Lambda(x)$ will be smaller. A careful reading will reveal that this process is essentially equivalent to importance sampling, where instead of assigning a weight $w(x) = \pi(x)/\nu(x)$ to a particle at x, one instead waits at x for a random amount of time, which has expectation $w(x)$.

Moving onwards, consider now a coordinate-wise global jump process, where, for $i = 1, \ldots, d$, at rate $\lambda_i(x)$, the i^{th} coordinate jumps to $y_i \sim \nu_i(y_i|x_{-i})$. Checking Condition 1.6 with $y = (x_{-i}, y_i)$, we see that we must have

$$\lambda_i(x) \propto \frac{\nu_i(x_i|x_{-i})}{\pi(x_i|x_{-i})}$$

up to a multiplicative function of x_{-i}. The same intuition holds as in the fully-global case; any coordinate which is particularly 'out-of-equilibrium' in the sense that $\pi(x_i|x_{-i}) \ll \nu_i(x_i|x_{-i})$ should be encouraged to be resampled. In the case where $\nu_i(x_i|x_{-i}) = \pi(x_i|x_{-i})$ for all i, the rates λ_i can all be taken equal to 1, and this is precisely a random-scan Gibbs sampler with exponential holding times between updates. The case in which $(\nu_i(x_i|x_{-i}), \pi(x_i|x_{-i}))$ are unequal but close is studied as a 'Tempered Gibbs Sampler' in [ZR19] with some success.

Finally, consider the more general case of a local jump process based on accept/reject-type jumps, i.e. the jump rate from x to y is given by $q(x \to y)\alpha(x \to y)$, where $q(x \to y)$ is a Markov kernel, and $\alpha(x \to y) \in [0, 1]$ can be evaluated. Condition 1.6 now necessitates that

$$\pi(x)q(x \to y)\alpha(x \to y) = \pi(y)q(y \to x)\alpha(y \to x),$$

and the discussion from Subsection 1.3.3 of this chapter implies that taking $\alpha = \alpha^{\text{MH}}$ will again deliver us a π-reversible chain. Doing so will generate a continuous-time Markov process which is equivalent to a Metropolis-Hastings chain, but with independent and identically-distributed Exp(1) holding times in between each jump. We emphasise that this construction is more of a theoretical tool, rather than a practically-useful algorithmic modification of usual Metropolis-Hastings.

Moving onto ODEs, we begin by noting that the flow of a nontrivial ODE can essentially never be in detailed balance with respect to a measure, as if x flows to y in time t, it will be exceedingly rare that y also flows to x in time t, unless both points lie on opposite points of a closed orbit ('antipodal'), and it is essentially impossible for this to hold *for all* t. On the contrary, these flows are in some sense 'strictly irreversible'; x flows to y if and only if y flows to x *backwards in time*.

To understand when the flow of an ODE can leave a measure invariant, begin by noting

that the 'generator' corresponding to the ODE $dx = b(x)dt$ is given by

$$\mathcal{L}f(x) = \langle b(x), \nabla f(x) \rangle.$$

As such, in order to leave π invariant, we examine the equality $\mathbf{E}_\pi [\mathcal{L}f(x)] = 0$, which can be rewritten (under appropriate smoothness and tail conditions on b, f, and π) as

$$\begin{aligned} \mathbf{E}_\pi [\mathcal{L}f(x)] &= \int \pi(x) \langle b(x), \nabla f(x) \rangle dx \\ &= -\int \operatorname{div}(\pi(x)b(x)) f(x) dx. \end{aligned}$$

We can thus deduce that to admit π as an invariant measure, it is necessary and sufficient that $\operatorname{div}(\pi b) = 0$. In [MCF15, MFCW19], an explicit construction is provided, which allows for π-invariant vector fields b to be generated systematically.

Theorem 2. *Let π be the density of a probability measure supported on all of $\mathbf{R}^d$, and let $b(x)$ be a vector field on $\mathbf{R}^d$. If the flow of this vector field leaves π invariant, then there exists a skew-symmetric matrix-valued function $Q(x)$ such that*

$$\begin{aligned} b(x) &= Q(x)\nabla \log \pi(x) + \Gamma(x) \\ \text{where } \Gamma_i(x) &= \sum_{j=1}^{d} \partial_{x_j} Q_{ij}(x) \quad \text{for } i = 1, \ldots, d. \end{aligned}$$

A particularly neat case is that of *divergence-free* or *volume-preserving* fields, i.e. vector fields for which $\operatorname{div} b \equiv 0$. In this special case, the condition for invariance reduces to $\langle \nabla \pi, b \rangle = 0$, i.e. that the density π is conserved along the flows of the ODE. A particularly prominent application of such flows has been the Hamiltonian Monte Carlo (HMC) algorithm [Nea11, HG14, Bet17], and its extension to Riemannian manifolds in [GC11].

Before moving on, we emphasise that as ODEs have deterministic flows, it is usually the case in practice that *only* using ODE dynamics will fail to generate an ergodic process, and one will not sample correctly from the desired measure. Nevertheless, this should not dissuade us from considering ODEs as a part of the simulation toolbox, as they can be fruitfully combined with other techniques, together with which one can sample correctly.

Finally, we come to SDEs. The Markov process corresponding to the diffusion

$$dx = b(x)dt + \sigma(x)dW$$

has generator given by

$$\mathcal{L}f(x) = \langle b(x), \nabla f(x) \rangle + \frac{1}{2}\mathbf{Tr}\left[\Sigma(x)\nabla^2 f(x)\right]$$
$$\text{where } \Sigma(x) = \sigma(x)\sigma(x)^T$$

Characterising the pairs of (b, Σ) which leave π invariant is more challenging than in the ODE case, as b and Σ interact nontrivially. Fortunately, this has also been addressed by the work of [MCF15, MFCW19], who provide a full characterisation of diffusion processes which admit a given π as invariant measure.

Theorem 3. *Let π be the density of a probability measure supported on all of $\mathbf{R}^d$, and consider the diffusion given by*

$$dx = b(x)dt + \sigma(x)dW.$$

- *If this diffusion is reversible with respect to π, then there exists a positive-semidefinite matrix-valued function $D(x)$ such that*

$$\begin{aligned} \sigma(x)\sigma(x)^T &= 2D(x) \\ b(x) &= D(x)\nabla \log \pi(x) + \Gamma(x) \\ \textit{where } \Gamma_i(x) &= \sum_{j=1}^{d} \partial_{x_j} D_{ij}(x) \quad \textit{for } i = 1, \ldots, d. \end{aligned}$$

- *If this diffusion admits π as an invariant measure, then there exist a positive-semidefinite matrix-valued function $D(x)$, and a skew-symmetric matrix-valued function $Q(x)$, such that*

$$\begin{aligned} \sigma(x)\sigma(x)^T &= 2D(x) \\ b(x) &= (D(x) + Q(x))\,\nabla \log \pi(x) + \Gamma(x) \\ \textit{where } \Gamma_i(x) &= \sum_{j=1}^{d} \partial_{x_j} \left(D_{ij}(x) + Q_{ij}(x)\right) \quad \textit{for } i = 1, \ldots, d. \end{aligned}$$

The key content of this theorem is twofold. The first conclusion is that any π-reversible diffusion can be specified by a positive-definite matrix-valued function $D(x)$, which can be viewed as a sort of preconditioner, emphasising important directions in the space. Dynamics of this form can thus be viewed as consisting of

1. purely dissipative behaviour, as the dynamics of $dx = D(x)\nabla \log \pi(x)dt$ try to dissipate the 'energy' $V(x) = -\log \pi(x)$

2. structured uniform exploration, through the action of the stochastic flow $dx = \Gamma(x)dt + \sqrt{2D(x)}dW$, which admits the uniform measure as an invariant measure.

The second conclusion is that any π-invariant diffusion is a superposition of a π-invariant ODE (as specified by Q) and a π-reversible SDE (as specified by D). As such, any π-invariant Markov process with continuous sample paths can be uniquely decomposed into a purely reversible part, which explores the space systematically, while dissipating excess energy, and a purely non-reversible part, which circulates mass around the space in a directed manner.

To make this discussion slightly more concrete, we present some examples of how this framework can be used to generate MCMC algorithms.

Example 1. *Taking $D(x) \equiv 1, Q(x) \equiv 0$ gives rise to the standard Overdamped Langevin Diffusion*

$$dx = \nabla \log \pi(x)dt + \sqrt{2}dW.$$

This is in some sense a 'canonical' diffusion for sampling purposes, and is well-studied (e.g. it is treated extensively in [Pav14]). It can be viewed as a gradient flow which is perturbed by additive noise; the SDE tries to move into regions of high probability by performing gradient ascent on the function $\log \pi(x)$, and then explores the target measure through the additional noise injection.

Example 2. *Taking $D(x) \equiv C, Q(x) \equiv 0$ for some fixed matrix $C \neq I$ gives rise to the standard Preconditioned Overdamped Langevin Diffusion*

$$dx = C\nabla \log \pi(x)dt + \sqrt{2C}dW.$$

At a high-level, this corresponds to the original Overdamped Langevin diffusion, applied in a different coordinate system. By applying a change of basis in this way, the diffusion can often be made to equilibrate more rapidly, by emphasising different directions in the state space.

Example 3. *It can be useful to design dynamics which operate in an extended space, which contains the original space as a subspace. A common choice is to augment the space with a 'momentum' variable p (usually equipped with a $\mathcal{N}(0, I)$ distribution), and let the particle dynamically explore this joint (x, p) space, often referred to as 'phase space'.*

Making this augmentation, one can set

$$D(x,p) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, Q(x,p) = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}$$

to obtain the dynamics

$$dx = p\,dt$$
$$dp = \nabla \log \pi(x)\,dt.$$

These are Hamilton's equations of motions, given the 'Hamiltonian' $\mathcal{H}(x,p) = -\log \pi(x) + \frac{1}{2}|p|^2$. Note that these dynamics are both volume-preserving and energy-preserving (i.e $\mathcal{H}$ is conserved along the flows of this ODE).

Example 4. *Working in phase space once more, we can take*

$$D(x,p) = \begin{pmatrix} 0 & 0 \\ 0 & \gamma I \end{pmatrix}, Q(x,p) = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}$$

to obtain the so-called Underdamped Langevin Dynamics

$$dx = p\,dt$$
$$dp = \nabla \log \pi(x)\,dt - \gamma p dt + \sqrt{2\gamma} dW.$$

Relative to Hamiltonian dynamics, this process dissipates momentum (through the $dp = -\gamma p dt$ term), while also adding noise to the momenta (through the $dp = \sqrt{2\gamma} dW$ term) to ensure exploration. Note that the diffusion matrix in this setting is ***not*** *full-rank.*

For the interested reader, a number of diverse other examples are given in [MCF15, MFCW19]. Some especially novel applications of this approach in recent years have involved constructing systems of interacting particles undergoing a 'collaborative diffusion' using this methodology; we refer to [NR19, GINR19, DNS19] for some recent advances in this burgeoning research area.

A reasonable question to ask here is whether an optimal choice for (D, Q) exists. A complication here is that this is really two questions; i) which choice of (D, Q) would give the optimal *continuous-time* dynamics, and ii) which dynamics can be efficiently discretised, such that the practically-implemented algorithm mixes quickly and correctly as well. In any case, both questions are quite challenging. There is a standing conjecture in the convex geometry community (implicit in the discussion in [Kol14, Kla14, KM16, KK17, KK19]) that in continuous time, the optimal choice of D, fixing $Q \equiv 0$, is given by the so-called *Kaehler-Einstein metric*. This metric is defined implicitly by the following construction:

1. For any probability measure π on $\mathbf{R}^d$ which i) is centered at the origin (i.e. $\mathbf{E}_\pi[x] = 0$) and ii) is not supported in a strict subspace of $\mathbf{R}^d$, there exists a *log-concave* measure $\mu(dy) = \exp(-\Phi(y))$ on $\mathbf{R}^d$ such that if $y \sim \mu$, then $x = \nabla\Phi(y)$ is marginally drawn according to π.

2. As Φ is convex, it admits a convex conjugate Φ^*.

3. The Kaehler-Einstein metric is then defined as

$$D^{\mathrm{KE}}(x) = \nabla^2 \Phi^*(x)$$

Unfortunately, the Kaehler-Einstein metric has the practical shortcoming that in almost all cases of interest, computing this D is intractable. A key barrier is that identifying and computing the so-called 'mirror map' Φ is extremely difficult, and essentially corresponds to the solution of a high-dimensional nonlinear PDE. As such, trying to work directly with the Kaehler-Einstein metric will generally not be an option, unless the target measure is extremely simple. However, if one can reproduce qualitative features of the Kaehler-Einstein metric with a tractable choice of D, some sort of practical 'approximate optimality' may be achievable.

In practice, $D(x)$ is typically taken to be either i) the identity matrix, ii) (an approximation to) the covariance matrix of the posterior, or iii) some other matrix which reflects the curvature of the posterior, either locally or globally. There exist a small number of highly-structured situations in which a bespoke choice of metric is both clear and easy to implement. For example, [HKRC18] construct a metric for sampling from Dirichlet measures on the simplex, such that the resulting process converges uniformly fast in time, for **all** settings of the parameters of the Dirichlet distribution. Nevertheless, despite the fact that variable-metric MCMC approaches have shown practical benefits in a number of applications, the present consensus seems to view them as perhaps too complicated for inexperienced users to apply. It is thus typically more common to rely on general-purpose gradient-based samplers like basic Langevin or Hamiltonian Monte Carlo instead, perhaps applying an appropriate reparametrisation along the way.

Implicit in the theorem of [MCF15, MFCW19] is the fact that if two Markov processes each admit π as an invariant measure, then the superposition of these two processes (in an appropriate sense) will also admit π as an invariant measure. This is a useful tool for constructing samplers; one can simply dip into the cookbook of π-invariant Markov processes, combine them additively in some proportions, and be done with it. What is not always clear is whether this provides the complete picture. In the case of diffusions, there is a particularly clean decomposition into 'reversible diffusion' and 'purely non-reversible ODE'. Such decompositions are not always apparent a priori; one goal of this thesis is to identify and elucidate such decompositions in novel settings.

1.4.3 Phase 3: Discretisation and Adjustment of Markov Processes

We now come to the question of how to simulate these processes, either exactly or approximately, in a manner which enables us to recover valid samples from our target distribution. The situation depends heavily on the class of moves under consideration, and so the exposition is divided up accordingly.

For Markov Jump Processes, one needs to be able to i) simulate the holding times, and ii) simulate the transition kernels. The extent to which these are difficult tasks depends on precisely how the process is specified. If the MJP is given in terms of a total jump rate $\Lambda(x)$ and normalised transition kernels $q(x \to y)$, then the holding times are exponentially distributed with rate $\Lambda(x)$, and can thus be simulated efficiently. Provided the transition kernels q can be simulated from in any form, e.g. rejection sampling, then one can simulate the full MJP without discretisation error.

On the other hand, if the MJP is only specified by state-to-state jump rates $\lambda(x \to y)$, a little more work is necessary. Suppose we can bound

$$\lambda(x \to y) \leqslant \bar{\Lambda}(x) \cdot r(x \to y)$$

uniformly in y, where $\bar{\Lambda}$ can be computed explicitly, and $r(x \to y)$ is a Markov kernel from which samples can be easily drawn. Defining $\alpha(x \to y) = \lambda(x \to y) / \left(\bar{\Lambda}(x) \cdot r(x \to y)\right) \in [0, 1]$, we can write $\lambda(x \to y) = \bar{\Lambda}(x) \cdot r(x \to y) \cdot \alpha(x \to y)$, which naturally suggests a rejection sampling-type approach, as in Algorithm 18.

Algorithm 18 Rejection Sampling Markov Jump Process

1. Sample $T \sim \text{Exp}(\bar{\Lambda}(x))$.
2. Sample $y \sim r(x \to y)$.
3. Sample $U \sim \mathcal{U}[0, 1]$.
4. Stay at x for time T, and if $U \leqslant \alpha(x \to y)$, then also jump to y.

One example of this method is the so-called 'Metropolis-Hastings Jump Process', i.e. a Metropolis-Hastings Markov chain with exponentially-distributed holding times in between jumps. In this case, $\Lambda(x) = \int q(x \to y)\alpha(x \to y)dy$ is not analytically available, and so the first method does not apply. Nevertheless, Algorithm 18 enables the process to be simulated without discretisation error.

For ODEs, exact simulation is typically intractable for nontrivial dynamics, i.e. if the dynamics are not those of a linear dynamical system, or reducible to something similarly

elementary, then one will need to resort to numerical discretisation. An extra difficulty pertaining to ODE proposals is that they are deterministic, and as such, the transition kernels will generally not admit a density with respect to the target measure.

Consider first using a proposal corresponding to the exact flow of some ODE, that is, fix some time-step $\Delta t > 0$, and for $t > 0$, let $\phi_t(x)$ be the 'flow map' of the ODE $dx = b(x)dt$, i.e. the location, at time t of a particle which started at $x_0 = x$. Extend this to $t < 0$ by defining $\phi_t(x)$ as the location at time $|t|$ of a particle initialised at $x_0 = x$, under the flow of the reversed vector field, i.e. $dx = -b(x)dt$. Note that $\{\phi_t(\cdot)\}_{t\in\mathbf{R}}$ then forms a group under composition, and in particular, that $(\phi_t \circ \phi_{-t})(x) = x$ for all (x, t).

With these definitions established, one can define the proposal kernel

$$q(x \to dy) = \frac{1}{2}\delta\left(\phi_{\Delta t}(x), dy\right) + \frac{1}{2}\delta\left(\phi_{-\Delta t}(x), dy\right),$$

i.e. with probability $1/2$, the particle follows the flow of the ODE forwards in time for Δt units of time; with the remaining probability, it instead follows the flow of the ODE backwards in time.

In order to make sense of this proposal in the Metropolis-Hastings filter, one needs to work with its most general formulation (see e.g. [Gre95, Tie98]), which is typically presented at the level of measures. More precisely, let π be the target measure, and let q be the proposal kernel. Under the condition that the measures $\Pi(dx, dy) = \pi(dx)q(x \to y)$ and $\Pi^T(dx, dy) = \pi(dy)q(y \to dx)$ are mutually absolutely continuous, it is possible to define the Radon-Nikodym derivative

$$r(x \to y) = \frac{d\Pi^T}{d\Pi}(x, y).$$

With this established, one can run the standard Metropolis-Hastings filter, using $\alpha(x \to y) = \min(1, r(x \to y))$, and thus generate a Markov chain with π as its invariant measure.

With this formulation in hand, it can be shown that the relevant Radon-Nikodym derivative for a proposed move from x to $y = \phi_{\Delta t}(x)$ is given by

$$r(x \to y) = \frac{\pi(y)}{\pi(x)} \cdot \left|\det\left(\frac{\partial y}{\partial x}\right)\right|,$$

where $\frac{\partial y}{\partial x}$ is the Jacobian matrix of the flow from x to y. In particular, r can be computed, enabling the use of ODE proposals in a Metropolis-Hastings scheme.

Suppose now that we are using a numerical discretisation of the ODE, giving us an approximate flow map $\hat{\phi}_{\Delta t}(x) \approx \phi_{\Delta t}(x)$. It is then essential that our approximate flow map be *exactly* invertible, i.e. $\left(\hat{\phi}_{\Delta t} \circ \hat{\phi}_{-\Delta t}\right)(x) = x$ for all x, as otherwise, the measures Π and Π^T could have disjoint support, and thus $r(x \to y)$ would have trivial behaviour, rendering our algorithm unable to move. This requirement is known as *reversibility* of the

discretisation scheme; we contrast this with the reversibility of the Markov process.

An immediate consequence of this is that, in the context of using ODE proposals in MCMC, one must be very careful in designing an appropriate numerical integrator. In particular, commonly-used approaches like Euler's methods (explicit and implicit), explicit multistep methods (e.g. Adams-Bashforth), and explicit multistage methods (Runge-Kutta) will generally **not** work (see e.g. [Ise09] for details on these methods, and more). Some implicit methods can be made to work - for example, the trapezoidal rule is symmetric in its endpoints by construction, and is thus immediately reversible - but the majority of off-the-shelf integrators are not constructed for use in MCMC, and as such, will not be suitable.

An approach which is generally more reliable is to additively decompose the vector field into simpler constituent vector fields, the flows of which can each be solved exactly. This is known as *splitting* in the field of numerical analysis. For the purposes of MCMC, it is often easiest to work with *symmetric* splittings, where the exact sub-flows are composed in a symmetric fashion. For example, consider the ODE given by

$$dx = (x - x^2)dt.$$

Although this ODE can in principle be solved exactly, it is much easier to solve its constituent parts. In particular, the flow of $dx = xdt$ for time t is given by $\phi^{(1)}_{\Delta t}(x) = \exp(\Delta t)x$, and the flow of $dx = -x^2dt$ is given by $\phi^{(2)}_{\Delta t}(x) = x/\left(1 + \Delta t \cdot x\right)$. One can then approximate the flow map by

$$\hat{\phi}_{\Delta t}(x) = \left(\phi^{(2)}_{\Delta t/2} \circ \phi^{(1)}_{\Delta t} \circ \phi^{(2)}_{\Delta t/2}\right)(x),$$

and standard methods suffice to show that this composition gives rise to a consistent approximation of the true flow map.

Note now that because both $\phi^{(1)}$ and $\phi^{(2)}$ are the *exact* flow maps of an ODE, they each form a group, and it is thus elementary to check that $\left(\hat{\phi}_{-\Delta t} \circ \hat{\phi}_{\Delta t}\right)(x) = x$, i.e. that the approximate flow map is a reversible discretisation of the ODE. It is a useful exercise to convince oneself that this would not be the case if we had instead taken $\hat{\phi}_{\Delta t} = \phi^{(1)}_{\Delta t} \circ \phi^{(2)}_{\Delta t}$.

The most commonly-used MCMC algorithm which directly incorporates ODE dynamics is Hamiltonian Monte Carlo (see [Nea11, Bet17] for reviews), which follows the dynamics

$$\begin{aligned} dx &= pdt \\ dp &= \nabla \log \pi(x) dt. \end{aligned}$$

Fortunately, these dynamics naturally suggest a splitting into tractable subsystems, namely

$$A : \begin{cases} dx &= p\,dt \\ dp &= 0\,dt \end{cases}$$
$$B : \begin{cases} dx &= 0\,dt \\ dp &= \nabla \log \pi(x)\,dt. \end{cases}$$

These systems admit exact flow maps, namely

$$\phi^{(1)}_{\Delta t}(x,p) = (x + \Delta t p, p)$$
$$\phi^{(2)}_{\Delta t}(x,p) = (x, p + \Delta t \nabla \log \pi(x)),$$

and composing them in the manner described above gives rise to the so-called *leapfrog integrator*, which is used in most standard implementations of HMC. See [SRL10, GC11, BPSSS11, LM16, CSASS19] for some scenarios in which more exotic splittings are either required or desirable.

When it comes to SDEs, there are different tools available, depending on the precise details of the diffusion at hand. Roughly speaking, from the point of view of MCMC simulation, there are three categories of SDE:

1. One-dimensional SDEs with constant diffusion coefficient and a specific drift form, which can be simulated *exactly*, using the rejection sampler of [BR05, BPRF06, BPR06].

2. SDEs with a full-rank diffusion matrix, which can be discretised with the Euler-Maruyama method, leading to numerical approximations which admit a tractable transition density.

3. Hypoelliptic SDEs with a rank-deficient diffusion matrix, which are generally treated with splitting methods.

In what follows, we will largely overlook the first category, as it has only been applied in quite a narrow context, and is not a tool for general-purpose MCMC in the same sense as the other two approaches.

For SDEs in the second category, one approximates the flow of the SDE $dx = b(x)dt + \sigma(x)dW$ for time Δt, started at x, by the so-called *Euler-Maruyama* method:

$$y \approx x + \Delta t \cdot b(x) + \sqrt{\Delta t} \cdot \sigma(x)\xi \quad \text{where } \xi \sim \mathcal{N}(0, I).$$

In general, the law of y will not be equal to the true law of $x(\Delta t)$ under the SDE, but for sufficiently small Δt, it will get increasingly close. For our purposes, this is not so

important; we do not win by approximating the SDE arbitrarily well; we win by generating a Markov chain which converges to the desired invariant measure as quickly as possible. As such, we take this numerical discretisation, and use it as the basis for a Metropolis-Hastings algorithm, i.e. we define

$$\begin{aligned}\Sigma(x) &= \sigma(x)\sigma(x)^T \\ q(x \to y) &= \mathcal{N}\left(y | x + \Delta t \cdot b(x), \Delta t \cdot \Sigma(x)\right),\end{aligned}$$

and use q as a proposal kernel. Note that because Σ has full rank throughout the space, q has positive density everywhere, and so the Metropolis-Hastings ratio $r(x \to y)$ is always well-defined. It is thus relatively straightforward to apply these methods for use in MCMC. They are primarily deployed in the context of simulating overdamped Langevin diffusions, see e.g. [RT96, Per15, DM17]. There has also been work on developing multi-stage variants of the Euler-Maruyama method (also known as *Runge-Kutta* methods), which are able to increase the range of Δt for which the integrator produces stable trajectories; see e.g. [VPZ19]. When used without Metropolis-Hastings corrections, these methods will incur a nonzero bias, but by increasing the range of feasible timesteps, they are able to equilibrate much more rapidly. In this context, there are alternative approaches for handling this induced bias (e.g. [Gil15]) which are perhaps more appropriate than Metropolis-Hastings corrections; we will revisit this point in greater depth later on.

In the third listed class of SDEs, for which the diffusion matrix is rank-deficient, a naive discretisation of the SDE can give rise to transition densities which are supported on a proper subspace of the state space, which can lead to degeneracies. The most common instance of this problem is the *Underdamped Langevin* diffusion, given by

$$\begin{aligned}dx &= pdt \\ dp &= \nabla \log \pi(x) dt - \gamma p dt + \sqrt{2\gamma} dW.\end{aligned}$$

In this setting, it is useful to take cues from the ODE setting, and split up the dynamics into tractable sub-problems whose dynamics can be solved exactly. Along the same lines

as the leapfrog integrator, we might decompose the system as

$$A : \begin{cases} dx &= p\,dt \\ dp &= 0\,dt \end{cases}$$
$$B : \begin{cases} dx &= 0\,dt \\ dp &= \nabla \log \pi(x)\,dt \end{cases}$$
$$O : \begin{cases} dx &= 0\,dt \\ dp &= -\gamma p\,dt + \sqrt{2\gamma} dW. \end{cases}$$

We recall from our earlier discussion that the first two systems can be solved exactly; readers who are familiar with SDEs will recognise that the third system is also amenable to exact treatment. This is the *Ornstein-Uhlenbeck* process, an autoregressive diffusion admitting $\mathcal{N}(0, I)$ as its invariant measure, and whose transition densities are given explicitly as

$$P_t^{\mathrm{OU}}(p \to dp') = \mathcal{N}\left(dp' | \exp(-\gamma t)p, (1 - \exp(-2\gamma t))\, I\right)$$

for $t > 0$. In particular, the transition densities are Gaussian, and can thus be simulated exactly. A standard symmetric splitting approach would then be to compose these exact flows, e.g. the 'ABOBA' method would consist of the following moves:

1. $x \leftarrow x + \frac{\Delta t}{2} p$
2. $p \leftarrow p + \frac{\Delta t}{2} \nabla \log \pi(x)$
3. $p' \sim P_{\Delta t}^{\mathrm{OU}}(p \to dp')$
4. $p \leftarrow p + \frac{\Delta t}{2} \nabla \log \pi(x)$
5. $x \leftarrow x + \frac{\Delta t}{2} p$

More details on the construction and analysis of splitting schemes can be found in [LM16]. We note that there are a wide range of SDEs to which these methods are applied; we point to [LR09, LNT09, JL11, MMW$^+$19, CKP20] for a taste of other complex diffusions which can be efficiently treated by splitting techniques.

For PDMPs, which are built as a hybrid of ODEs and MJPs, one has to consider additional details. In specifying a PDMP, one prescribes i) the vector field $b(x)$ which the process will deterministically follow, ii) the jump rate $\Lambda(x)$ which will specify when jumps occur, and iii) the jump kernel $Q(x \to dy)$ which specifies where the process will jump to at these events. This combination adds an additional complication, in that the

rates at which jumps occur are now time-varying, and so the event times now form an *inhomogeneous* Poisson process.

To simulate PDMPs directly in continuous time, we must first stipulate that the dynamics are tractable, i.e. our particle is moving under the motion of a vector field b, such that we explicitly know the flow map corresponding to b. Provided that the event rate at x is given by $\Lambda(x)$, the probability of **no** events happening in the next t units of time is given by

$$\begin{aligned}\mathbf{P}(\tau > t) &= \exp\left(-\int_0^t \Lambda(X_s)ds\right) \\ &= \exp\left(-\int_0^t \Lambda(\phi_s(X_0))ds\right),\end{aligned}$$

where we recall that ϕ_s is the flow map of the vector field b for s units of time. As such, unless we are blessed with a pair of (Λ, ϕ) such that the integral above can be computed analytically, we will need to find some upper bound, i.e. some $\bar{\Lambda}_x$ which satisfies $\bar{\Lambda}_x(t) \geqslant \Lambda(\phi_t(x))$ for $t \geqslant 0$, and whose integral can be computed and inverted quickly. We can then use a rejection procedure akin to that used in Algorithm 18 for MJPs. We will also need to sample from the transition kernel $q(x \to y)$ when events do occur; this is typically more straightforward in applications. We note also that there are certain discrete-time Markov chains which admit interpretations as discretisations of PDMPs; see [Gus98, VBCDD17, Mon19, PA20] for some illustrative examples.

Finally, Jump-SDEs involve all three key dynamics (drift, diffusion, and jumps), and as a result, are particularly challenging, which has arguably limited their direct application in the context of MCMC simulation thus far. We refer to [CR11a, Pol15, PJR16] for some examples of how their exact simulation can be tackled in certain highly-structured, low-dimensional scenarios, and to [PFJR16, WRS19, WPRS19] for some recent applications of Jump SDEs (specifically, killed diffusions with regeneration) to MCMC.

Before moving onto the next stage in the workflow, it is worth touching on the variety of inexact approaches to simulation, where the Metropolis-Hastings correction either plays a minor role, or is cast out of the picture entirely. The appeal of these methods is that one no longer needs to play by the rules of reversibility, exact π-invariance, and so on, any more. One might instead take the approach that the exact dynamics would provide an exact solution to our problem, and so we can focus our efforts on getting a good approximation to the ideal dynamics, rather than on carefully reproducing the invariant measure. This opens the doors to the use of more elaborate numerical integration schemes which can deliver improved stability, accuracy, and adaptivity; see e.g. [VPZ19, LWME19, Kle20] for some examples.

The cost is, of course, that the asymptotic exactness, for which we have otherwise

been striving, is now gone. This is particularly upsetting from a modularity point of view: once any of the Markov kernels involved in the overall Markov chain becomes inexact, the chain as a whole is liable to become inexact, and it can become difficult to obtain a full understanding of the biases induced by this inexactness. There are other approaches to bias correction, e.g. using Stein's method to select, prune, and reweight samples (e.g. [LL16, CMG+18, HSR20, RCC+20]), using Multi-Level Monte Carlo methods (reviewed with supreme clarity in [Gil15]) to reduce error (both bias and variance), reducing the step-size over time (as in e.g. [WT11]) such that the bias diminishes, or any of a number of other techniques. Ultimately, which of these is to be preferred will depend on the application at hand, and the extent to which different biases are tolerable to the user.

1.4.4 Phase 4: Model Structure and Implementation Details

The previous discussions can guide us toward a theoretical algorithm which, if implemented correctly, should be able to explore a target measure which we specify. However, there are a number of important considerations which will affect the details of how one implements such an algorithm, particularly with regard to the efficiency and complexity of each step. In this section, we outline some of these considerations, with particular care to how they arise for different families of sampler.

For a Gibbs sampler, perhaps the key consideration is to identify which blocks can be resampled in parallel without inducing a bias. Given a subset of indices $J \subset I$ such that $\{x_j\}_{j\in J}$ are separated in the relevant graphical model (an 'independent set', in graph-theoretic terminology), one has that

$$\pi\left(\{x_j\}_{j\in J}|\{x_i\}_{i\in I\setminus J}\right) = \prod_{j\in J}\pi(x_j|\{x_i\}_{i\in I\setminus J}),$$

enabling the parallel resampling of all variables with indices in J. As such, identifying independent sets within the graphical model allows for the application of parallel updates within Gibbs sampling, which can reap considerable practical benefits; some perspectives on this problem are collected in [GLGG11]. There is also a question of how to schedule these updates (deterministic, random, synchronous, asynchronous, etc.) for which the conclusion is currently not entirely clear; we refer again to [LWK95, LC06, RR15, HDSMR16, MM17] for a variety of perspectives on this matter. Similar comments apply to Metropolis-within-Gibbs samplers.

For Metropolis-type samplers, the key cost of implementation tends to be evaluations of the target density, and functionals thereof, e.g. the gradient of the log density (the negative of the 'potential', in physics parlance) is a key ingredient of MALA, HMC and other algorithms. While historically, users have coded up these gradients by hand, it has become increasingly common (and efficient) to outsource gradient computations to automatic

differentiation packages; two popular modern such packages are Tensorflow [AAB+15] and JAX [BFH+20]. An additional consideration is that these potentials are often formed as the sum of many simpler terms (this is particularly true in Bayesian inference and statistical physics), and so it can be beneficial to distribute the computation of these individual terms and their gradients as well; tools like CasADi [AGH+19] (developed in the context of optimal control) are noted to be particularly well-adapted to such challenges.

For MJPs, it is often the case that jump times and jump locations are decoupled; when appropriate, it can be useful to distribute these computations. For MJPs on discrete spaces, computing the total jump rate out of a given state x involves calculating the sum of the jump rates between x and its neighbours y; this is again possible to distribute. For ODEs and SDEs, there are few additional considerations, aside from how to efficiently compute the relevant gradients.

For PDMPs, the chief computational difficulty arises in computing the jump times, which occur according to an inhomogeneous Poisson process. As such, to be simulated exactly, one typically needs to identify some analytically-tractable upper bound to the event rate, which is difficult to automate, as any such bounds must hold globally. When the target measure is in some sense decomposable (e.g. factorises according to a graphical model), and each component is log-concave (or otherwise structured), this can be more manageable. A fully black-box implementation of any existing PDMP seems challenging at present.

Finally, there are certain model-specific challenges which can arise, essentially corresponding to when certain parts of the system are fast to compute, and other parts are more expensive. Standard examples include Uncertainty Quantification in Inverse Problems [Stu10, DS17], where evaluating the likelihood often involves the (expensive) numerical solution of a Partial Differential Equation, or in Approximate Bayesian Computation (ABC, [BZB02]), where likelihood access comes purely through the form of simulations from the forward model, which are again often expensive. In these situations, it is instructive to i) use computational resources to accelerate the expensive sub-routines of the algorithm as best possible, and ii) avoid calls to these expensive components where unnecessary. See e.g. [CF05, EHL06, Pra16, PB20] for some approaches to this problem.

1.4.5 Phase 5: Tuning and Refinement of MCMC Algorithms

Once the structure of the implementation of an MCMC algorithm is decided, there will still typically be a number of free parameters to be set in the algorithm, which we will collectively denote by θ. While it will often be the case that the algorithm retains its theoretical guarantees irrespective of the value of θ, the practical performance of an MCMC algorithm will often depend heavily on their setting.

A key first step is to identify which parameters θ remain free (at a suitable level of

abstraction), and to understand which aspects of the algorithm's performance they will influence. For example, a step-size parameter in a Metropolis-Hastings scheme typically induces a fairly transparent trade-off: small step-sizes will correspond to high acceptance rates, but slow exploration, whereas large step-sizes will accelerate the motion of the chain when moves are accepted, but these acceptances will become less frequent. By contrast, some free parameters will qualitatively change the pathwise behaviour of the chain; for example, the choice of diffusion matrix in an SDE will induce a certain geometry on the target space, analogously to the role of preconditioning in deterministic numerical computation.

Once the free parameters θ are identified and understood, it is natural to seek an 'optimal' setting of these parameters. To this end, the next step is usually to fix a working definition of what optimality might mean, derived either from some theoretical consideration, or from a tractable heuristic. Note that this is not always entirely straightforward. In reality, the key quantity which we are seeking to optimise is usually the speed at which the Markov chain converges to equilibrium, which i) can be hard to reliably estimate, and ii) can be *even harder* to use within an optimisation loop.

In practice, for efficient and robust adaptation of free parameters in MCMC algorithms, it is typically necessary to formalise the adaptation task in the form

$$\begin{aligned} &\text{optimise } L(\theta) \text{ over } \theta \in \Theta \quad \text{or} \\ &\text{solve } H(\theta) = 0 \text{ for } \theta \in \Theta, \end{aligned}$$

where L or H can be estimated directly from a run of the algorithm. In the optimisation setting, it is typically also necessary that the gradient of L with respect to θ can be estimated, though there some situations in which this requirement can be relaxed. The reasoning is clear: one needs some way of approximately measuring how close to optimal we are, and some information which can help us to update θ in the right direction. Specific choices of objectives will be discussed in later sections.

An important choice is how one adjusts these parameters in practice. If the parameters vary over the course of the algorithm, the Markov process in question is now *time-inhomogeneous*, which means that many of the theoretical aspects underpinning the MCMC approach do not necessarily continue to hold as-is. As such, a safe approach is to run the algorithm for some fixed amount of time, use the information from that run to set the parameters, and then fix those parameters for the remainder of the run. The remainder will then be a time-homogeneous Markov process, and standard theoretical considerations will apply.

A bolder approach is to simply continue adjusting parameters dynamically over the run of the chain. When done with appropriate care (see e.g. [AT08, RR09] for specifics), this

can be highly efficient, without sacrificing the theoretical properties of the original chain. This is conceptually appealing, in the sense that you might hope to eventually converge to the 'true' optimal parameter in some limit, leading to an optimal Markov process. This is the realm of *adaptive* algorithms, where the algorithm uses information accumulated over the course of its run to tune itself, allowing for performance which is comparable to having known and used the true optimal parameters, right from the start.[††]

We conclude by making some more general comments relating to the implementation and tuning of several common sampler types, as well as to their interplay with different model types.

To begin with, Gibbs samplers typically require few decisions to be made, so are quick to get up and running (as algorithms); the same cannot necessarily be said for their mixing behaviour, which can be quite poor when the variables are highly correlated. One also tends not to have many options when a Gibbs sampler goes wrong; occasionally a change of parametrisation will help things (see e.g. [GSC95, Pap03, PRS07, YM11, PRZ20] for useful examples), but there are not too many other options for improving the mixing of a Gibbs sampler, aside from switching to a different method.

Metropolis-Hastings algorithms with uninformed random walk proposals also typically require few user choices to be made. There is the step-size, and then perhaps the covariance matrix of the randomness as well. See [HST99, HST01, HST05, Vih12] for some adaptation strategies which apply in this context. Of course, even well-optimised random-walk proposals can only incorporate limited information about the target, and so the mixing of the chains can struggle as the complexity of the target grows.

For Metropolis-Hastings algorithms with proposals derived from ODEs and SDEs, there is usually a bit more work required in order to get the algorithm up and running. One needs to choose a discretisation scheme, and determine how to properly correct for discretisation error. As in the case of RWMH, there is typically also some freedom in choosing the step-size and a preconditioning matrix of some flavour. Note that in contrast to RWMH, the behaviour of such chains in the transient and stationary phase of the algorithm can be quite distinct (see e.g. [CRR05, KOS18, KOS19] for discussion on this point), as the use of gradient information allows the chain to move more aggressively in the tails. While broadly beneficial for mixing behaviour, it is worth commenting that this can complicate tuning. However, broadly speaking, one should expect these approaches to mix more rapidly, as the motion of the chain is well-informed by gradient information about the target density, allowing for a systematic exploration of the target.

PDMPs are a middle ground in terms of decisions and complexity. Typically, there

[††]We note that 'adaptive' is often used somewhat loosely to denote an algorithm which tunes its own parameters over the course of its run, without comment on (approximate) optimality of the resulting algorithm. See Section 4.3 of [Ora19] for some interesting comments on the semantic drift of this terminology over recent years.

are few free parameters to be set (this is arguably a more general virtue of working in continuous time), but the 'exact' parts of the process are more challenging to implement. In particular, computing event times requires the simulation of an inhomogeneous Poisson process, which requires explicit and tractable upper bounds on the event rate. This typically means more work for the user, as these bounds must (at present) be derived on a case-by-case basis. Once these algorithms are up and running, their mixing behaviour is relatively satisfactory; present consensus seems to be that they clearly out-perform methods like Gibbs sampling and Random Walk Metropolis-Hastings, and can be competitive with MALA and HMC, depending on the target measure.

For highly-structured, modular models, Gibbs sampling is naturally able to use this structure to its advantage, particularly when parallel computing is an option. When the mixing of Gibbs sampling is unsatisfactory, but one prefers not to switch directly to MALA or HMC, one can try middle-ground solutions like MALA-within-Gibbs and HMC-within-Gibbs, for which the benefits of structure can still be useful (see [Béd17, TMM20] for additional discussion). The Local Bouncy Particle Sampler of [BCVD18] shows some promise of potentially getting the best of both worlds here, for reasons which will be expanded upon in greater detail in subsequent sections.

For models in which the target is in some sense intractable (Pseudo-Marginal, Doubly-Intractable, ABC, Expensive Likelihood, 'Big Data' Problems, etc.), much of the difficulty arises in getting to a stage where it is possible for the chain to admit the correct invariant measure. It is possible to use more advanced proposal schemes in these settings (as in e.g. [ADL16, GS17, SBF18]), but it is fair to say that much of the foundational work thus far has focused on making any form of MCMC feasible for these models. This is of course changing gradually.

1.5 Roadmap

The preceding review first sought to outline why stochastic modelling is valuable, and why exploring stochastic models via simulation is a useful tool. Subsequently, we introduced several general simulation-based methods for exploring probability measures, before zooming in on Markov Process-based solutions to this problem. Restricting our attention to this approach, we then proceeded to outline a high-level workflow for constructing practical Markov process-based algorithms which will correctly sample from a given target measure, and how they can be made efficient.

This discussion notwithstanding, the general problem of designing efficient MCMC algorithms for a given probability measure is far from solved. There remain classes of models whose structure is challenging to exploit, interesting proposal mechanisms with poorly-understood behaviour, and challenges in how to optimally deploy computational resources. As such, there remains a persistent need for mathematical tools and techniques with which to address these gaps.

In this dissertation, we present three pieces of original work, targeted at expanding our understanding of the design and analysis of MCMC algorithms.

1. In Chapter 2, we study the application of *Piecewise-Deterministic Markov Processes* (PDMPs) to Markov chain-based simulation. PDMPs are a non-reversible, continuous-time family of Markov processes, and as such, stand in stark contrast to the ambient paradigm of reversible, discrete-time Markov chains, built on the foundations of Gibbs sampling and the Metropolis-Hastings algorithm. Some applications of PDMPs to Monte Carlo simulation have already shown promise, and although accompanying theory has made rapid progress in recent years, there remain a number of unanswered questions in the area. In this work, we study the versatility of PDMPs, with a focus on characterising how one can adapt the methodology underpinning existing samplers (e.g. the Bouncy Particle Sampler of [BCVD18], the Zig-Zag Process of [BFR19], the Coordinate Sampler of [WR20], etc.) to PDMPs which are driven by more general classes of vector fields. Attention is also paid to how such novel PDMPs can be designed to respect the modular structure which arises in many applications.

2. In Chapter 3, we continue on the theme of non-reversible sampling in continuous time, but with a focus on sampling from measures which are supported on discrete spaces. Such tasks are widespread across MCMC, and have a (perhaps unfair) reputation for requiring bespoke, model-specific algorithmic solutions. We contend that this need not be the case, and present a framework for constructing useful discrete samplers, which sample in continuous time and apply under quite general conditions. Moreover, we derive novel *non-reversible* discrete samplers, which outperform their reversible counterparts in practice. A key ingredient of the construction is to highlight

exploitable symmetry structures which recur across a range of practical discrete sampling tasks. Focusing on this common ground allows us to democratise the landscape of discrete sampling, enabling the simple application of efficient general-purpose samplers to generic problems on discrete spaces.

3. In Chapter 4, we devote our attention to the theoretical study of parallelism-enhanced MCMC algorithms, and to clarifying how parallel computing resources ought to be exploited in the context of sampling. The main contribution is to study a simple parallel version of the Metropolis-Hastings algorithm, and characterise its asymptotic efficiency via the '*Optimal Scaling*' paradigm of Roberts and coauthors (e.g. [RGG97, RR98, RR01, BPR$^+$13]. Our theoretical results provide predictions and guidance for how much of an efficiency gain should be expected for a given number of processors, which are in excellent agreement with our experimental study. These results provide concrete recommendations for the MCMC practitioner, and suggest a pathway towards the further theoretical study of other parallelism-enhanced MCMC algorithms.

Due to the key role played in both Chapter 1 and 2 by non-reversibility and continuous-time Markov processes, we will first provide a review of these topics, with a focus on their role in simulation-based exploration of probability measures. We then present the three main works in the order outlined above.

1.6 Interlude: Continuous-Time and Non-Reversible Sampling

In this section, we revisit the genesis of Markov process-based sampling algorithms in the discrete-time, reversible paradigm, and explore why this paradigm predominated for so long, why alternatives are worth considering in principle, what has enabled these alternatives to become practical in recent years, and which challenges remain.

1.6.1 Reversibility: Why?

To recapitulate, a Markov kernel K is *reversible* with respect to the measure π if

$$\pi(x)K(x \to y) = \pi(y)K(y \to x)$$

as measures. We initially motivated this condition as a tractable approach to constructing Markov processes which admit π as an invariant measure, where the tractability stems from the *local* nature of the condition. Moreover, two key building blocks of MCMC algorithms — the Gibbs sampler, and the Metropolis-Hastings filter — are reversible, and are easy to work with precisely because of this property. One result of this is that, to the sampling community at large, reversible algorithms are perceived as safe and familiar.

An additional contributing factor to the prevalence of reversible Markov processes in the MCMC literature is that they possess particularly nice analytical properties. In particular, defining the generator of a discrete-time Markov chain as

$$(L_K f)(x) = \int K(x \to y)\,[f(y) - f(x)]\,dy,$$

one can show that π-reversibility of K is equivalent to the self-adjointness of L_K as an operator on $L^2(\pi)$. The same connection holds true for continuous-time reversible Markov processes.

As a result of this connection, a number of functional-analytic tools become available for studying reversible Markov chains, many of which stem from *spectral theory*. This simplicity can be viewed as a higher-level analog of the phenomenon that symmetric matrices can be diagonalised with respect to an orthogonal basis, and are thus often more mathematically convenient to work with than general matrices. As such, a key convenience of reversible Markov processes is that they provide the theorist with a well-studied collection of tools for understanding their behaviour.

1.6.2 Reversibility: Why Not?

One simple if uninspiring reason to consider non-reversible Markov processes (for any task) is that they are a broader class of processes. A more convincing answer comes from considering the qualitative properties of reversible and non-reversible chains respectively. Examining the condition for reversibility, one notes that when the process has converged to stationarity, given any pair of states (x, y), one is equally likely to see a transition from x to y as from y to x, over any period of time. This suggests that the chain lacks a sense of direction, and that one might expect to observe backtracking behaviour. This is indeed borne out in practice.

To design Markov chains with improved mixing behaviour, it thus seems sensible to imbue them with some sense of directionality, such that the chain can systematically circulate around the state space, while also undergoing stochastic exploration. A priori, it is perhaps not clear that this is possible to do properly, or that it will yield nontrivial practical improvements. Fortunately, work in recent years has convincingly exhibited that it is both possible and worthwhile to study such non-reversible chains. Here, we will work through some examples of how this can be accomplished, in the hopes that it exposes the underlying principles of how one constructs non-reversible MCMC algorithms.

1.6.3 Non-Reversibility: How?

We begin by considering a simple example of a reversible MCMC algorithm, namely the ubiquitous Random-Walk Metropolis-Hastings algorithm. Fix a step-size $h > 0$, and consider the Metropolis-Hastings algorithm with proposal $q(x \to y) = \mathcal{N}(y|x, hI)$, with steps as in Algorithm 19. We note explicitly that this algorithm generates a π-reversible Markov chain.

Algorithm 19 Random Walk Metropolis-Hastings Update for π, with step-size h

1. At x, propose a move to $y \sim \mathcal{N}(y|x, hI)$.
2. Compute $r(x \to y) = \pi(y)/\pi(x)$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output y; otherwise, output x.

The standard description of this algorithm is that from x, a new location y is proposed, and then either accepted or rejected. However, one can reframe the algorithm as proposing a *direction* $v = y - x$, and then either taking a step in that direction, or not. Consider now the joint distribution $\Pi(x, v) = \pi(x) \cdot \mathcal{N}(v|0, hI)$, and the *deterministic* proposal $T : (x, v) \mapsto (x + v, -v)$. Note that T is both volume-preserving and an involution, i.e.

$(T \circ T)(x, v) = (x, v)$. One can then rewrite Algorithm 19 as a Metropolis-within-Gibbs chain, with Π as its invariant measure, as in Algorithm 20.

Algorithm 20 Random Walk Metropolis-Hastings Update for π, with step-size h

1. At (x, v), resample $v \sim \mathcal{N}(v|0, hI)$.
2. Propose a move to $(y, w) = T(x, v)$
3. Compute $r((x, v) \to (y, w)) = \pi(y)/\pi(x)$.
4. Sample $u \sim \mathcal{U}[0, 1]$.
5. If $u < r$, output (y, w); otherwise, output (x, v).

A key conceptual difference here is that we are now explicitly keeping track of the 'direction' variable v. The benefit of this approach is that we can thus seek to modify the algorithm in a way which encourages persistent behaviour in a given direction, rather than picking a completely random new direction at each step.

Note that if we omit step 1 in Algorithm 20, the direction is reversed after a successful move. This is perhaps counter-productive, as we would hope that continuing to move in an accepted direction would be fruitful. We can thus add in an additional deterministic 'flip' move to negate this behaviour, giving rise to the *non-reversible* Algorithm 21.

Algorithm 21 Non-Reversible Random Walk Metropolis-Hastings Update for π, with step-size h

1. At (x, v), propose a move to $(y, w) = T(x, v)$
2. Compute $r((x, v) \to (y, w)) = \pi(y)/\pi(x)$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output (y, w); otherwise, output (x, v).
5. Set $v = -v$.

In this setting, when we accept a move, we try to continue moving in the same direction, whereas if we reject a move, we instead reverse our direction and attempt to retrace our steps. A shortcoming of this approach is that the chain now becomes reducible, only moving back and forth along the line in direction v. A simple modification is to include a 'gentle refreshment of the direction', whereby v is blended with independent Gaussian noise at each step, as in Algorithm 22.

A similar algorithm is proposed in [Gus98]; see also [Bie16, Mai18] for some related constructions. Some analysis of this class of algorithms is provided in [DHN00, Hil00],

Algorithm 22 Non-Reversible Random Walk Metropolis-Hastings Update for π, with step-size h, refreshment parameter θ

1. At (x, v), propose a move to $(y, w) = T(x, v)$
2. Compute $r((x, v) \to (y, w)) = \pi(y)/\pi(x)$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output (y, w); otherwise, output (x, v).
5. Set $v \leftarrow -v$.
6. Sample $\xi \sim \mathcal{N}(\xi|0, hI)$, and set $v \leftarrow \cos\theta \cdot v + \sin\theta \cdot \xi$.

showing that their convergence behaviour can be considerably better than the original reversible algorithm in appropriate limiting regimes. A particularly illustrative result in [DHN00] establishes that when considering Markov chains on the set $\{1, 2, \ldots, N\}$, the natural reversible Markov chain takes $\tilde{O}(N^2)$ time to converge to equilibrium, whereas the corresponding non-reversible chain takes only $\tilde{O}(N)$. This gap between 'diffusive' and 'ballistic' scaling of mixing times is emblematic of what one hopes to gain when introducing non-reversibility.

While this is a simple example, it is nevertheless fairly illustrative of how many non-reversible MCMC algorithms are constructed. One first introduces some auxiliary variables which encode the 'directionality' of the proposals in some way, then rewrites the dynamics of the chain in terms of these extended variables, and finally modifies the dynamics such that the chain can follow a given direction in a coherent manner.

A second example begins by observing a specific Markov kernel which leaves a given distribution invariant, without being reversible, and then building from there. To be more specific, define the measure π, supported on $\mathbf{R}_+$, by

$$\pi_0(dx) = \exp(-x)dx \quad \text{for } x > 0,$$

i.e. an exponential distribution of unit rate. Now, let $h > 0$, and consider the Markov kernel $f(x \to y)$ which is generated by i) sampling $z \sim \text{Exp}(h)$, and then ii) setting $y = (1 + h) \cdot \min(x, z)$. One can compute an expression for this kernel as

$$\begin{aligned} f(x \to y) &= \exp(-hx) \cdot \delta\left((1+h)x, dy\right) \\ &\quad + (1 - \exp(-hx)) \cdot \left(\frac{\frac{h}{1+h} \exp\left(-\frac{h}{1+h} y\right) \cdot \mathbf{I}\left[0 < y < (1+h)x\right]}{1 - \exp(-hx)} \right) dy, \end{aligned}$$

and moreover, one can verify that f leaves π_0 invariant. However, it is clear that it cannot

be π_0-reversible; for small x, the chain essentially drifts deterministically to the right, and for large x, the chain jumps approximately-uniformly into the area to its left. This behaviour is clearly distinct from its time reversal[‡‡], which we can compute explicitly as

$$b(x \to y) = \frac{1}{1+h} \cdot \delta\left(\frac{x}{1+h}, dy\right) + \frac{h}{1+h} \cdot \exp\left(-\left\{y - \frac{x}{1+h}\right\}\right) \cdot \mathbf{I}\left[y > \frac{x}{1+h}\right] dy,$$

i.e. shrink x by a factor of $(1+h)$, and then with probability $\frac{h}{1+h}$, sample $z \sim \text{Exp}(1)$ and then add it to the previous value.

Now, one could attempt to use either of f or b in isolation to build a Metropolis–Hastings algorithm for sampling from distributions on $\mathbf{R}_+$, which are close in law to π_0, e.g. $\pi(x) = \pi_0(x)\ell(x)$, where ℓ is a slowly-varying term. While this is conceptually sound, if implemented naively, this will generally not be an effective strategy. The central issue is that by using only one of these two proposals, one will often propose moves from x to y such that the corresponding move from y to x has zero probability, thus leading to an acceptance rate of 0.

A solution is to consider *both* proposals at once. Write $q(x \to y; +1) = f(x \to y), q(x \to y; -1) = b(x \to y)$, and consider sampling from the target $\Pi(x, \tau) = \pi(x)R(\tau)$, where as before, $R(\tau)$ is the uniform or 'Rademacher' distribution on $\tau \in \{\pm 1\}$. One can then define a 'Forward-Backward' Metropolis-Hastings Markov chain, using Algorithm 23.

Algorithm 23 Reversible Forward-Backward Metropolis-Hastings for $\pi = \pi_0(x)\ell(x)$, with step-size h

1. At (x, τ), sample $y \sim q(x \to y; \tau)$, and propose a move to $(y, -\tau)$.
2. Compute $r((x, \tau) \to (y, -\tau)) = \ell(y)/\ell(x)$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output $(y, -\tau)$; otherwise, output (x, τ).

Note that because the proposal kernels are in balance with respect to the base measure π_0, the acceptance ratio simplifies to only involving the function ℓ. This is a recurrent feature for proposals which are designed with the prior in mind. Our initial construction again generates a reversible chain, but naturally suggests a means for inducing non-reversibility; namely, deterministically flipping τ at the end of each step. This gives rise to Algorithm 24, which is now genuinely non-reversible, and will naturally exhibit persistent behaviour in line with this.

This illustrates a second useful principle for designing non-reversible MCMC algorithms: if one is considering some ideal dynamics which leave π invariant, but are not π-reversible,

[‡‡]Defined by $\pi_0(x)f(x \to y) = \pi_0(y)b(y \to x)$.

Algorithm 24 Non-Reversible Forward-Backward Metropolis-Hastings for $\pi = \pi_0(x)\ell(x)$, with step-size h

1. At (x, τ), sample $y \sim q(x \to y; \tau)$, and propose a move to $(y, -\tau)$.
2. Compute $r((x, \tau) \to (y, -\tau)) = \ell(y)/\ell(x)$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output $(y, -\tau)$; otherwise, output (x, τ).
5. Flip $\tau \leftarrow -\tau$.

then it is instructive to study the time-reversal of those dynamics, and stitch them together with the original dynamics in some fashion. We will see further examples of this principle in later chapters.

A final, more advanced example of a non-reversible MCMC algorithm which stems from perturbing an existing, reversible algorithm is the 'non-reversible overdamped Langevin diffusion' (see e.g. [HHMS93, HHMS05, LNP13, RBS15]). By the results of [MCF15, MFCW19], one can show that for any skew-symmetric matrix J, the following diffusion admits π as an invariant measure

$$dx = (I + J)\nabla \log \pi(x)dt + \sqrt{2}dW. \tag{1.8}$$

For $J = 0$, one recovers the usual (π-reversible) overdamped Langevin diffusion; for $J \neq 0$, the diffusion is non-reversible, but still admits π as an invariant measure. In particular, taking the time-reversal of this diffusion corresponds to replacing J by $-J$. Theoretical results demonstrate that in continuous time, relative to taking $J = 0$, this can either improve convergence rates, reduce asymptotic variance, or both (see e.g. [HHMS93, HHMS05, LNP13]. Crucially, it never worsens either aspect.

There is thus a challenge: the ideal, continuous-time process is not π-reversible, but if we are to discretise this diffusion, and naively use it as a proposal within a Metropolis-Hastings algorithm, then the resulting Markov chain will be π-reversible. We are essentially coercing a proposal which is not π-reversible (even approximately) to be so, and one might expect this to cause problems.

To be more precise, for $h > 0$, write

$$q(x \to y; J) = \mathcal{N}(y|x + h(I + J)\nabla \log \pi(x), 2hI).$$

Then, when $J = 0$, one has that

$$\log \frac{\pi(y)q(y \to x; J)}{\pi(x)q(x \to y; J)} \asymp h^3$$

whereas when $J \neq 0$, instead

$$\log \frac{\pi(y)q(y \to x; J)}{\pi(x)q(x \to y; J)} \asymp h,$$

i.e. compared to the original MALA, we should expect to observe worse acceptance rates when using a comparable step-size, even though our numerical integrator is 'as good' as before. This is perhaps a concerning disparity.

A preferable approach is to note that the time-reversal of the diffusion in Equation 1.8 has the same form, but with J negated. With this in mind, one can compute that

$$\log \frac{\pi(y)q(y \to x; -J)}{\pi(x)q(x \to y; J)} \asymp h^3.$$

That is, although the non-reversible diffusion proposals fail to be in detailed balance with respect to π, there is a sort of 'skew-detailed balance' at play between this pair of proposals. This is a notion which will show itself to be a key ingredient of non-reversible MCMC in subsequent chapters.

The utility of this observation is that it encourages us to extend our state-space to include a variable which dictates whether to use J or $-J$ for our proposal. Let $R(\tau)$ again be the Rademacher distribution on $\tau \in \{\pm 1\}$. We can then define our extended target measure as $\Pi(x, \tau) = \pi(x)R(\tau)$, and construct a reversible Metropolis-Hastings chain according to Algorithm 25.

Algorithm 25 Non-Reversible Diffusion Proposal, Reversible Metropolis-Hastings Update for π, with step-size h

1. At (x, τ), sample $y \sim q(x \to y; \tau J)$, and propose a move to $(y, -\tau)$.
2. Compute $r((x, \tau) \to (y, -\tau)) = \frac{\pi(y)q(y \to x; -\tau J)}{\pi(x)q(x \to y; \tau J)}$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output $(y, -\tau)$; otherwise, output (x, τ).

Of course, we are in some sense back where we started: even though we have reasonable acceptance rates again, we are still generating a reversible Markov chain! Our dynamics will not see the benefit of the fleeting non-reversibility of the proposals either; if we accept a move, then we flip τ, and begin moving according to the opposite dynamics again.

Fortunately, there is a simple fix to this: at the end of each step, deterministically flip τ. This will give rise to Algorithm 26, generating a genuinely non-reversible Markov chain, still with Π as its invariant measure, but now with the persistent non-reversible dynamics we were initially hoping for. We note also that a related construction can be applied to the *underdamped* Langevin diffusion, as in [DNP17]. Note that this is non-obvious; the underdamped Langevin diffusion is already non-reversible, and so the construction is non-trivial in that it makes the dynamics 'even more' non-reversible, in a certain sense.

Algorithm 26 Non-Reversible MALA Update for π, with step-size h, skew-symmetric matrix J

1. At (x, τ), sample $y \sim q(x \to y; \tau J)$, and propose a move to $(y, -\tau)$.
2. Compute $r((x, \tau) \to (y, -\tau)) = \frac{\pi(y)q(y \to x; -\tau J)}{\pi(x)q(x \to y; \tau J)}$.
3. Sample $u \sim \mathcal{U}[0, 1]$.
4. If $u < r$, output $(y, -\tau)$; otherwise, output (x, τ).
5. Flip $\tau \leftarrow -\tau$.

A desired outcome of this sub-section is to have demonstrated some recurrent common features of useful non-reversible Markov chains as a primer towards more advanced applications in subsequent chapters.

1.6.4 Limitations of Non-Reversibility

Having established that the systematic construction of non-reversible MCMC algorithms is within reach, it is worth setting some reasonable expectations about what these new methods are able to bring us, and what they are not. What is well-established by now is that non-reversible methods are able to provide improved rates of convergence to equilibrium, and reduced asymptotic variance for estimates of expectations under the stationary measure. A slightly subtle question concerns whether the mixing behaviour is most improved in the transient phase of the chain, or once it has already converged to stationarity; see [VM20] for discussion on this point. One might also ask whether non-reversibility allows the resulting chain to overcome multimodality; this is true up to a point (see e.g. [GGZ18] for some results to this effect), but not arbitrarily; ultimately, Markov processes undergoing local exploration will struggle to overcome steep potential barriers. A final question might ask whether non-reversibility can confer an algorithm with additional robustness to high-dimensionality. At present, the answer appears to be no; robustness to dimension seems to generally be a function of how accurately the underlying stochastic process can be simulated, and this is not immediately impacted by modifying

the dynamics to be non-reversible[§§].

An interesting theoretical aspect of non-reversible Markov processes is that they often require new techniques. In particular, a generic strategy for establishing the convergence to equilibrium of a Markov process is to show that some 'divergence measure' between the law of the process at time t, denoted by μ_t, and the invariant measure of the process, denoted by π, tends to 0 as t grows, i.e. $D(t) = D(\mu_t, \pi) \to 0$ as $t \to \infty$. For reversible processes, it is often the case that $D'(t) \leqslant 0$, i.e. the process monotonically dissipates the divergence between the current law and the equilibrium measure, always growing closer to its target. With extra work, one then aims to show that $D(t)$ will in fact decrease all the way to 0. For non-reversible processes, this approach will not generally work directly; even if $D(t)$ is ultimately tending towards 0 - perhaps even at an asymptotically faster rate than its reversible counterpart - it will often be the case that, at least for small values of t, the divergence will actually *increase*. This counterintuitive phenomenon has necessitated novel techniques, involving the construction of non-standard divergence functions (e.g. [DMS15, DPBCD18, ADNR18]), time-averaging approaches (e.g. [CLW19]), reflective couplings (e.g. [BREZ18, EGZ19]), and more. It is perhaps fair to say that a unified theory for establishing the convergence of non-reversible Markov processes to equilibrium does not yet exist in the same way that it arguably does for reversible processes. The onslaught of new results in this area over recent years gives the author hope that a clearer theoretical picture is not too far over the horizon.

There are also some high-level questions about non-reversible MCMC algorithms which remain at least partially open. When does non-reversibility help most? When is it worth trying? Does it ever make things worse? Is there a systematic and optimal way of 'de-reversibilising' reversible MCMC algorithms? Should one tune non-reversible algorithms with the same policies with which one tunes reversible algorithms? Intuition for these problems is still in its genesis, without even speaking of a full theoretical resolution.

1.6.5 Discrete-Time: Why?

Discrete-time MCMC algorithms have been preferred historically partially due to their ease of implementation; conceptually, a discrete-time algorithm is usually easier to gain an intuition for, and the abstract algorithm corresponds fairly directly to the practical algorithm which one programs on a computer. More concretely, the natural implementation of both the Gibbs sampler and the Metropolis-Hastings algorithm takes place in discrete time. While these algorithms can in principle be embedded into continuous time, this approach has borne limited fruit in terms of practical improvements thus far. As such,

[§§]However, it is worth noting that several non-reversible MCMC schemes admit smoother sample paths than 'related' reversible schemes. This *can* make them easier to simulate more accurately in some respects; see [MMW+19, CKP20] for some examples to this effect.

discrete time has very much been the status quo for some time now. We point to [CRR03] for some early discussion on the use of continuous-time samplers in the context of trans-dimensional targets, i.e. models whose components lie in a space of varying dimension.

1.6.6 Discrete-Time: Why Not?

If discrete-time Markov processes are natural to consider because they typically reflect how an algorithm is implemented in practice, then continuous-time Markov processes are natural to consider because they often reflect some idealisation of the algorithm at hand, and can thus serve as a means of building intuition for how the discrete chain will behave. It is often the case that the convergence behaviour of a discrete-time Markov chain can be grappled with by first analysing the behaviour of the continuous-time analogue (which is usually simpler, due to the absence of numerical errors and discrete events), and then subsequently realising the discrete chain as an approximation to the continuous chain, tracking the approximation errors appropriately, and ultimately unraveling this story to deduce the behaviour of the original chain.

1.6.7 Limitations and Challenges of Continuous-Time

Of course, the preceding discussion essentially views continuous-time processes as pure idealisation. This erasure skips over the reality that, in fact, there **do** exist nontrivial, useful, continuous-time Markov processes which can be simulated without incurring discretisation error. Moreover, in addition to the theoretical and conceptual appeal of working directly in continuous time, there is often also a further practical benefit: if no discretisation needs to occur, then there are fewer algorithmic choices (step-size, numerical integration scheme, etc.) upon which the user needs to decide upon. As such, it is worth considering whether continuous-time Markov processes can actually be used practically for the exploration of probability measures.

To begin with, it is worth emphasising there is no free lunch. If we want our algorithm to use fast-mixing dynamics, to converge to the desired invariant measure, and to be exactly-implementable all at once, then the class of available algorithms is inevitably somewhat narrow. Moreover, even when we can achieve all of these features in principle, it might be computationally expensive or require substantial user input to do so in practice. A specific aspect of this which often arises is that for the exact simulation of continuous-time processes, one often requires some form of non-local information, e.g. bounds on some function related to the process dynamics. When such information is readily available, then it is rewarding to be able to exploit it algorithmically, but it should be acknowledged that acquiring such information can a priori be quite challenging.

To make this somewhat more concrete, consider the classes of process we described

in Section 1.4 of this chapter. Exact simulation of ODEs is rarely possible, and when possible, is even more rarely ergodic for the target measure. Exact simulation of SDEs is essentially only practical in low dimension, and existing methods require that both of $(\log \pi, \nabla \log \pi)$ can be explicitly bounded globally, which severely limits their applicability to practical problems. Similar comments apply to Jump-SDEs and Levy SDEs, though usually with additional, more severe complications.

This leaves us with MJPs and PDMPs. MJPs are particularly friendly creatures in the context of continuous-time simulation, as although they evolve directly in continuous time, they only actually move around at discrete instances in time. When situated at a location x, one only needs to calculate the jump rates to possible neighbours, which is a local procedure.

The situation with PDMPs is slightly more complicated. As with ODEs, when considering only dynamics which can be simulated exactly, there is a relatively limited range of interesting options available. Notably, in contrast to ODEs, one can generally build ergodic Markov processes out of PDMPs, and these can be made to converge to equilibrium relatively quickly. A slightly upsetting limitation remains, in that constructing an ODE which is both i) possible to simulate exactly and ii) capable of using information about the target distribution to productively guide its exploration, is usually not possible for interesting targets. Nevertheless, even fairly uninformed dynamics (e.g. straight lines, as in [BCVD18, BFR19, WR20] and elliptical orbits, as in [VBCDD17]) can be used as a basis for constructing efficient PDMP-based sampling algorithms.

Computationally, a key challenge of simulating PDMPs arises from the need to control the rate at which events occur, which takes place according to an inhomogeneous Poisson process with state-dependent event rate $\lambda(t) = \Lambda(X_t)$. In order to simulate these events, one needs to be able to invert the mapping $t \mapsto \int_0^t \lambda(s)ds$, or an appropriate upper bound thereof. The difficulty is thus twofold: one needs analytic control of i) how the event rate depends on the position, via $\Lambda(x)$, ii) how the position X depends on time t, via the flow of the ODE, and iii) how the two intertwine, i.e. how the composite $\lambda(t) = \Lambda(X_t)$ behaves over time. Depending on how structured $\Lambda(x)$ and X_t are, this task of finding appropriate bounds can be quite challenging. One common scenario is that Λ is naturally expressed as a sum of many terms. Even when each term can be bounded reasonably tightly, it will often be the case that aggregating individual bounds into a bound on the sum will lead to looseness. Typically this will manifest in the form of the resulting algorithm 'expecting' an event to happen sooner than is necessary, thus wasting computational time, and slowing the progress of the algorithm.

As ever, there is a tension between what we would like to be able to do, and what we are able to do efficiently in practice. When considering the allure of exact continuous-time simulation for exploration of probability measures on continuous space, PDMPs

currently seem to represent the most promising option. It is not yet possible to apply them automatically (relative to e.g. Metropolis-Hastings- or Gibbs sampling-type techniques), but gradually, model structures are being identified which are both widespread, and able to accommodate such automation, to some extent.

In closing, the adoption of both non-reversible and continuous-time sampling techniques seems to have initially been held at bay by virtue of their deviation from techniques with which the community is more familiar, with Metropolis-Hastings and Gibbs-type algorithms in particular representing the status quo. These novel classes of samplers are, in various ways, at odds with this traditional paradigm, which has lent them the impression of being harder to work with. Perhaps the key insight of the advances in these areas over recent years has been that although 'generic' non-reversible and continuous-time samplers might be intractable in some form, we are seldom working with generic processes. In contrast, practical sampling methods are highly-structured, regardless of the particular paradigm, due to the far-from-arbitrary nature of algorithm design. As such, it seems misleading to describe an entire class of processes as being 'intractable' or otherwise; the real marker of utility will be whether or not the class in question contains specific structured instances which can be handled naturally.

In the specific context of harnessing non-reversibility for MCMC sampling, the key substructure which renders them tractable is that they are often expressible as a composition of reversible parts which complement one another in a specific way. The usual recipe is i) a Markov kernel K which, when applied in isolation, would generate a proper π-reversible, ergodic Markov chain, ii) a highly-degenerate Markov kernel F, which although π-reversible, would generally produce a chain with some mixture of reducibility, periodicity, or other ergodicity-breaking behaviour, and iii) an interplay between K and F which, in some way, encourages the chain to move into new parts of the state space, rather than back-tracking into already-explored areas. In most examples of which the author is aware, F will typically involve some sort of flip-based symmetry, which can usually be interpreted as a time-reversal operator of sorts. It would be interesting to identify other symmetries which can be used to systematically generate non-reversible chains with useful properties.

With respect to the application of continuous-time processes to sampling, the essential observation has been that trying to reproduce generic continuous-time dynamics exactly will be challenging in most cases, but that a rich class of dynamics can be induced by considering simple combinations of simple dynamics. Advances have stemmed not from identifying increasingly complicated vector fields whose flows can be solved exactly, but from interweaving simple flows in simple ways, repeatedly, by splitting techniques and switching dynamics, as in PDMPs. At a high level, one can observe similar phenomena in Bayesian modelling, where useful, novel priors are often specified as the marginal distributions of simple hierarchical constructions involving elementary distributions (e.g.

[CPS10, LL10, PS10, ACD11, ADL13, PV17]), rather than by cooking up elaborate densities by writing down long formulas. Another striking example of this principle has been the deep learning boom of the past decade, whereby some of the most successful regression models are specified simply by the repeated composition of linear transformations and a single (!) nonlinear mapping, rather than by invoking special functions directly.

It has been eye-opening to see how, over the past few years, these classes of Markov processes have gone from seeming arcane, out-of-reach, and intractable, to becoming first-class citizens in the world of sampling algorithms, based essentially around identifying minimal structural properties which make them easy to work with. The author eagerly awaits seeing which other wild beasts in the world of sampling are tamed in the years to come, and which essential structures are identified in order to enable their practical utility.

Chapter 2

A Complete Characterisation of Trajectorially-Reversible PDMPs

Abstract

Piecewise-Deterministic Markov Processes (PDMPs) have arisen in recent years as an efficient means of sampling from a probability measure using *non-reversible* dynamics. Although the PDMP framework accommodates a wide range of underlying dynamics in principle, existing approaches have tended to use quite simple dynamics, such as straight lines and elliptical orbits.

In this work, we present a procedure which enables the application of general dynamical systems in the PDMP framework to sample from a given measure. The construction leverages the notion of *trajectorial reversibility*, which expands the domain of applicability of PDMPs to include dynamics which are **not** equipped with symmetries *a priori*. It is established that the procedure is both correct (i.e. generates Markov processes which admit the correct stationary measure) and complete (i.e. all correct processes take this form) in a general setting. Specific, constructive recommendations are also made for how to implement the resulting algorithms in practice.

2.1 Introduction

In this paper, we study the task of sampling from a probability measure using Markov chain Monte Carlo (MCMC). Historically, such algorithms have been built out of *reversible* Markov chains, with the Metropolis-Hastings algorithm (see [MRR$^+$53, Has70, Tie98, DSC98]) in particular providing a general-purpose framework for converting generic

proposal mechanisms into asymptotically-correct sampling schemes. As a result of the simplicity of the Metropolis-Hastings procedure, a vast proportion of the MCMC literature is built around constructing and studying reversible Markov chains.

There has been a persistent interest in non-reversible methods over the years as well. Much of this interest was driven by a number of works (e.g. [Gus98, CLP99, DHN00, Nea04, CH13, SDMD14, RBS15, DLP16, Bie16, MFCW19]) which demonstrated that introducing non-reversibility into a reversible chain can improve rates of convergence to equilibrium, often substantially. In recent years, non-reversible methods have moved into the spotlight as a result of the advent of *Piecewise-Deterministic Markov Processes* (hereafter, PDMPs). These are stochastic processes, evolving in *continuous time*, which use a combination of deterministic dynamics and jumps to navigate the state space. In particular, the introduction of generally-applicable schemes like the Bouncy Particle Sampler of [BCVD18] and the Zig-Zag Process of [BFR19] have exhibited that constructing non-reversible Markov processes which admit the correct stationary distribution is both possible and practical. Additional generalisations of these PDMPs have shown improved robustness and convergence properties under various scenarios (e.g. [MKK14, WR17, PGCP17, ST17, WR20]).

The construction of PDMPs often allows them to suppress backtracking behaviour and reach equilibrium rapidly. Although the PDMP framework accommodates a wide range of underlying dynamics in principle, existing approaches have tended to use quite simple dynamics, such as straight lines and elliptical orbits (though a notable exception is the recent work of [TT18]). In [VBCDD17], sufficient conditions are given for a PDMP using general ODE dynamics to admit the correct stationary measure. Here, this picture is completed, by supplying matching *necessary* conditions, under a natural symmetry assumption on the process. With this characterisation established, it is then possible to provide a constructive outline of how to build the sampler, given a choice of dynamics.

More precisely, suppose a target measure $\mu(dz)$ is specified, and one is able to simulate an ODE system of the form $dz = b(z)dt$ efficiently both forwards **and** backwards in time. This paper outlines how to explicitly specify an algorithm which uses the b-dynamics to produce a PDMP which admits μ as a stationary measure. Correctness of this procedure is verified in a general setting, and the construction is proven to be complete, in the sense that any correct PDMP (within the class of *trajectorially reversible* PDMPs, a classification which is described in Section 2.2) must take this specific form. Through this characterisation, it is then possible to contextualise a number of PDMP variants (e.g. the Zig-Zag Process, Local BPS, Coordinate Sampler) with the notion of a 'split' PDMP (see Section 2.3), which allows for a more transparent picture of how different event-generating mechanisms can fit together cohesively in the context of PDMPs.

Most closely related to this work is [VBCDD17], who in addition to providing sufficient

conditions for a range of processes to admit the correct stationary measure, also devote attention to the discrete-time and 'doubly-stochastic' families of PDMPs. The work presented here should be viewed as complementary; the framing allows for the removal of certain symmetry assumptions (e.g. the existence of certain measure-preserving involutions is automatic here, rather than assumed a priori), and by supplying necessary conditions for stationarity to match the existing sufficient conditions, it is more straightforward to sketch out a constructive recipe for designing PDMP algorithms. Particular attention is also devoted here to the role of trajectorial reversibility in designing correct PDMPs, as well as a more in-depth analysis of the relevant abstract quantities which govern the behaviour of a PDMP.

The structure of the paper is as follows. Section 2 reviews the basics of PDMPs, and introduces two structured classes of PDMPs, the *time-enriched* and *split time-enriched* PDMP. Section 3 introduces a new class of PDMP-type algorithms, and establishes that this class is, in an appropriate sense, necessary and sufficient for constructing μ-stationary PDMPs. Section 4 provides the proofs of this claim. Section 5 uses this characterisation of μ-stationary PDMPs to propose a collection of new processes, the relative merits of which are discussed from the point of view of convergence, ease of implementation, and the class of models to which they can be fruitfully applied. Finally, Section 6 recapitulates, and provides a discussion on the outlook for PDMPs in MCMC.

2.2 Piecewise-Deterministic Markov Processes

In this section, Piecewise-Deterministic Markov Processes are defined, and their algorithmic implementation is sketched. With this established, two variants of PDMPs, the *Time-Enriched* and *Split Time-Enriched* PDMPs are introduced, and are placed in the context of existing PDMPs by way of examples. Finally, *trajectorial reversibility*, a symmetry property satisfied by many PDMPs of practical interest, is introduced.

2.2.1 Basics of PDMPs

Piecewise-Deterministic Markov Processes (PDMPs) are a class of continuous-time stochastic processes, which were introduced in [Dav84]. For the purposes of this work, a PDMP can be thought of as the trajectory of a particle following the flow of an ordinary differential equation (ODE), interspersed with random jumps. Throughout, we work on the space $\mathcal{Z} = \mathbf{R}^N$ for some $N \in \mathbf{N}$, noting that the results of [DGM18b] allow much of what follows to be generalised to the case where $\mathcal{Z}$ is a smooth closed Riemannian sub-manifold of $\mathbf{R}^N$. Now, define:

Definition 2.2.1. *A $\mathcal{Z}$-valued PDMP $(Z_t)_{t\geqslant 0}$ is specified by*

1. *A vector field b on $\mathcal{Z}$, which is globally C^1 and grows at most linearly at infinity, i.e.*

$$\exists\, C > 0 \quad \text{such that} \quad \|b(z)\| \leqslant C(1 + \|z\|) \quad \text{for all } z$$

   Note that this ensures that the system $dz = b(z)dt$ has a solution for all $t > 0$.

2. *A rate function $\lambda : \mathcal{Z} \to [0, \infty)$, which is continuous and locally integrable along flows induced by b. That is, if $z(t)$ is a solution to $dz = b(z)dt$, then for any $T < \infty$, we have $\int_0^T \lambda(z(t))\, dt < \infty$.*

3. *A Markov Kernel $Q(\cdot, \cdot)$, that is, a map $Q : \mathcal{Z} \times \mathcal{B}(\mathcal{Z}) \to [0, 1]$ such that*

   - *$\forall A \in \mathcal{B}(\mathcal{Z})$, the map $z \mapsto Q(z, A)$ is measurable.*
   - *$\forall z \in \mathcal{Z}$, the map $A \mapsto Q(z, A)$ is a probability measure on $\mathcal{Z}$.*

   where $\mathcal{B}(\mathcal{Z})$ is the Borel sigma-algebra on $\mathcal{Z}$. Going forward, we will write the latter map as $Q(z \to dz')$, i.e.

$$Q(z, A) = \int_{z' \in A} Q(z \to dz').$$

We say that $(Z_t)_{t\geqslant 0}$ is a PDMP 'driven by b' with 'event rate' λ and 'jump kernel' Q.

It bears commenting that in some other works (e.g. [DGM18b]), it is preferred to characterise the dynamics by their flow maps, rather than the underlying vector field.

This allows for a slightly greater generality of underlying dynamics to be considered. In this work, the chosen convention is to work directly with the vector field, as many of the calculations presented herein involve the infinitesimal generator of the process at hand, and by working directly with the vector field, many of these calculations become more transparent.

Informally, one can think of Z_t as a process which deterministically follows the vector field b until an 'event' occurs. These events occur according to an inhomogeneous Poisson process of rate $\lambda(z)$, i.e. the intensity *varies* along the path of Z. If an event occurs at z, then z is replaced by a draw from $Q(z \to dz')$, and the process then resumes following the vector field.

Pseudocode for implementing such a PDMP is outlined in Algorithm 27, borrowing the presentation from [VBCDD17].

Algorithm 27 PDMP

---

1. Initialize Z_0 arbitrarily on $\mathcal{Z}$, and set $t_0 = 0$.
2. for $k = 1, 2, \ldots$ do
   (a) For $s \geqslant 0$, let z be the solution to

$$\begin{aligned} dz &= b(z)dt \\ z(0) &= Z_{t_{k-1}} \end{aligned}$$

   (b) Sample an inter-event time T_k, where T_k is a non-negative random variable such that

$$\mathbf{P}\left(T_k \geq t\right) = \exp\left[-\int_{s=0}^{t} \lambda\left(z(s)\right) ds\right].$$

   (c) For $s \in (0, T_k)$, set $Z_{t_{k-1}+s} = z(s)$.
   (d) Set $t_k = t_{k-1} + T_k$ and sample

$$Z_{t_k} \sim Q\left(Z_{t_k^-}, dZ_{t_k}\right).$$

Recall that the *infinitesimal generator* (hereafter, simply 'generator') $\mathcal{L}$ of a stochastic process $(X_t)_{t \geqslant 0}$ acts on functions u by

$$\left(\mathcal{L}u\right)(x) = \lim_{t \to 0^+} \frac{\mathbf{E}[u(X_t)|X_0 = x] - u(x)}{t}$$

for all functions u such that the limit exists. In what follows, the form of the generator of a process will be asserted without additional comment; for the processes considered herein, one can verify the correctness of these claims by checking the conditions from Section 7 of [DGM18b].

For PDMPs of the form described above, the generator is given by

$$\mathcal{L}u(z) = \langle b(z), \nabla_z u(z) \rangle + \lambda(z) \int_{z' \in \mathcal{Z}} Q(z \to dz') \left[u(z') - u(z) \right].$$

An eminent example of a PDMP is the Bouncy Particle Sampler (BPS). In this algorithm, one is interested in sampling from the distribution $\pi(dx) = \exp(-U(x))dx$, but constructs a PDMP on an extended state space with coordinates $z = (x, v)$, where v is an auxiliary variable representing velocity. The PDMP is specified as follows:

$$\begin{aligned} z &= (x, v) \\ b(z) &= (v, 0) \\ \lambda(z) &= \max(\langle v, \nabla U(x) \rangle, 0) \\ Q(z \to dz') &= \delta(x, dx') \cdot \delta(R_x(v), dv') \\ \text{where } R_x(v) &= \left(I - 2 \frac{(\nabla U(x)) \left(\nabla U(x)\right)^T}{\left(\nabla U(x)\right)^T (\nabla U(x))} \right) v. \end{aligned}$$

This process can be shown to admit $\mu(dz) = \pi(dx)\psi(dv)$ as a stationary measure, where ψ is any spherically-symmetric distribution of finite first moment (e.g. uniform on the sphere, isotropic Gaussian). In this process, the variable of interest x follows a piecewise-linear path, with its velocity jumping upon the occurrence of an event. In effect, events happen when the particle is heading towards regions of lower probability, discouraging the process from spending too much time in regions of low probability under μ. In Section 3, it will be shown that for systems with incompressible dynamics (i.e. $\operatorname{div} b = 0.$), the event rate essentially **must** behave in this way; for compressible systems the situation is slightly more subtle.

It bears emphasising that in applications, almost all PDMPs follow the BPS in being defined on an extended state space. Typically, the variable of interest x is augmented with an auxiliary variable v which represents some sort of velocity or momentum (as will be seen in several examples later in this work). In principle, though, this need not be the case, and one can certainly imagine scenarios in which the auxiliary variable corresponds to something else entirely. This has not been fully explored in practice yet. In any case, the theory in this paper is presented as agnostic to the choice of augmentation, i.e. all results are presented directly in terms of the joint coordinate z. As the analysis in Sections 3 and 4 will show, choosing to work in this setting is sufficiently general to recover the corresponding statements in those more specialised cases.

2.2.2 Time-Enriched PDMPs

We introduce here our first extension of PDMPs, the notion of a *time-enriched* PDMP. These are PDMPs for which the state of the process contains both i) the position of a particle, and ii) a discrete random variable $\tau \in \{\pm 1\}$. As such, the implied state space is thus $\mathcal{Z} \times \{\pm 1\}$. The interpretation throughout will be that τ represents the 'direction of time', in the sense that when $\tau = 1$, the particle follows the flow of the ODE *forwards* in time, and when $\tau = -1$, the particle follows the flow of the ODE *backwards* in time. Note that this **does not** correspond to the usual extended variable augmentations (e.g. velocity, momentum); the continuous part of the process lives in the same state space as before, but now has an additional discrete component attached.

More precisely, we define

Definition 2.2.2. *A $\mathcal{Z} \times \{\pm 1\}$-valued time-enriched PDMP (TE-PDMP) $(Z_t, \tau_t)_{t \geqslant 0}$ is specified by*

1. *A vector field b on $\mathcal{Z}$, which is globally C^1 and grows at most linearly at infinity, i.e.*

$$\exists C > 0 \quad \text{such that} \quad \|b(z)\| \leqslant C(1 + \|z\|) \quad \text{for all } z$$

   Note that this ensures that the system $dz = b(z)dt$ has a solution for all $t > 0$.

2. *A rate function $\lambda : \mathcal{Z} \to [0, \infty)$, which is continuous and locally integrable along flows induced by b both forwards **and** backwards in time. That is, if $z(t)$ is a solution to $dz = b(z)dt$, then for $T < \infty$, we have that both*

$$\int_0^T \lambda(z(t))\, dt < \infty, \quad \int_{-T}^0 \lambda(z(t))\, dt < \infty$$

   for any $z(0)$.

3. *A pair of $\mathcal{Z}$-valued Markov Transition kernels, $\{Q^\tau(z \to dz')\}_{\tau \in \{\pm 1\}}$.*

Having specified a TE-PDMP with the above information, it is now possible to describe their dynamical behaviour. Informally, TE-PDMPs deviate from standard PDMPs by being able to follow the flows of their vector field both forwards and backwards in time, that is, by solving

$$dz = \tau b(z)dt$$

for both $\tau = 1$ and $\tau = -1$. The new discrete variable τ dictates which direction of time the process is adhering to and remains constant in between jumps. Events happen in much the same way as for regular PDMPs, noting that now the event rate λ can depend on both of (z, τ). Finally, a more substantial deviation is that the jump dynamics Q have

a specific structure, namely, one first resamples z according to $Q^\tau(z, dz')$, and then τ is *deterministically flipped* to $-\tau$, i.e.

$$Q^{\text{Joint}}((z,\tau) \to (dz', d\tau')) = Q^\tau(z \to dz') \cdot \delta(-\tau, d\tau').$$

As such, the process evolves by the z-coordinate following the flow of the vector field in the direction of time τ, until it experiences an event. When the event occurs, the z coordinate jumps according to Q^τ, and τ jumps to $-\tau$.

The pseudocode in Algorithm 28 outlines how to simulate such a process. For TE-

Algorithm 28 TE-PDMP

1. Initialize (Z_0, τ_0) arbitrarily on $\mathcal{Z} \times \{\pm 1\}$, and set $t_0 = 0$.
2. for $k = 1, 2, \ldots$ do
   (a) For $s \geqslant 0$, let z be the solution to
   $$\begin{aligned} dz &= \tau_{k-1} b(z) dt \\ z(0) &= Z_{t_{k-1}} \end{aligned}$$
   (b) Sample an inter-event time T_k, where T_k is a non-negative random variable such that
   $$\mathbf{P}\left(T_k \geq t\right) = \exp\left[-\int_{s=0}^{t} \lambda\left(z(s), \tau_{k-1}\right) ds\right].$$
   (c) For $s \in (0, T_k)$, set $Z_{t_{k-1}+s} = z(s)$.
   (d) Set $t_k = t_{k-1} + T_k$ and sample
   $$Z_{t_k} \sim Q^{\tau_{k-1}}\left(Z_{t_k^-}, dZ_{t_k}\right).$$
   (e) Set $\tau_k = -\tau_{k-1}$.

PDMPs of the form described above, the generator is given by

$$\mathcal{L}u(z,\tau) = \tau \langle b(z), \nabla_z u(z,\tau)\rangle + \lambda(z,\tau) \int_{z' \in \mathcal{Z}} Q^\tau(z \to dz') \left[u(z', -\tau) - u(z,\tau)\right].$$

It is worth pausing to remark that although the concept of a TE-PDMP does not subsume all possible PDMPs, it is relatively natural in light of the existing PDMPs which are currently used for sampling, many of which are easily expressed in the TE-PDMP framework. For example, returning to the previous example of the Bouncy Particle Sampler,

when $\mu(dx) = \exp(-U(x))dx$, one can define a TE-PDMP with

$$\begin{aligned} b(x,v) &= (v,0) \\ \lambda(z,\tau) &= \max\left(\tau\langle v, \nabla U(x)\rangle, 0\right) \\ Q^{\tau}(z \to dz') &= \delta(x, dx') \cdot \delta(-R_x(v), dv'), \end{aligned}$$

which produces the same algorithm. Similar constructions exist for other algorithms in the literature, examples of which will be provided later in this paper.

2.2.2.1 Trajectorial Reversibility

We also extend the definition of trajectorial reversibility (see e.g. [DM13]) to TE-PDMPs.

Definition 2.2.3. *Fix a time-enriched PDMP $(Z_t, \tau_t)_{t\geqslant 0}$ with a unique stationary measure, and let $(Z'_t, \tau'_t)_{t\geqslant 0}$ denote the law of this PDMP as evolved backwards in time from stationarity. Say that $(Z_t, \tau_t)_{t\geqslant 0}$ is trajectorially reversible if*

$$(Z'_t, \tau'_t)_{t\geqslant 0} \stackrel{d}{=} (Z_t, -\tau_t)_{t\geqslant 0},$$

i.e. the time reversal of the process is equal in law to the original process, with the direction of time flipped.

One utility of this concept is that it elucidates the sense in which many existing PDMPs are 'almost' reversible, i.e. modulo a simple involution. It is reasonable to expect that for general dynamics*, it will typically be necessary for non-reversible methods to obey a condition of this form, in the same way that discrete-time MCMC algorithms are typically built by composing reversible kernels. It is easy to check that reversible Markov chains have the correct stationary distribution precisely because reversibility (i.e. the detailed balance condition) is local in nature, and thus easily verified. In the same way, it is typically straightforward to verify that trajectorially-reversible Markov chains admit the desired stationary measures. We note that this concept is closely linked to (μ, Q) self-adjointness, as described in [AL19].

*For an interesting deviation from this trend, note that in [MDS17], the authors seem able to circumvent this restriction somewhat by restricting to a setting in which the dynamics are linear and the velocity distributions are spherically-symmetric.

2.2.3 Split Time-Enriched PDMPs

We describe here the *Split* Time-Enriched PDMP (Split-TE-PDMP). This construction is initially motivated by the observation that when considering PDMPs on high-dimensional spaces, there is often an incentive to allow the process to act in distinct ways on the different coordinates. To this end, define the following:

Definition 2.2.4. *Given a space $\mathcal{Z}$ and a positive integer D, a State Space Decomposition of $\mathcal{Z}$ into D parts is a D-tuple of subspaces $(\mathcal{Z}_1, \dots, \mathcal{Z}_d)$ such that $\mathcal{Z} = \prod_{i=1}^{D} \mathcal{Z}_i$. We call D the 'size' of the decomposition.*

In contrast to basic TE-PDMPs, the idea for Split TE-PDMPs is that some coordinates of the vector z might follow the vector field b forwards in time, with the others following backwards in time. Thus instead of requiring a single 'direction of time' variable τ, we will now need $\tau \in \{\pm 1\}^D$, and allow the process to evolve under the flow of

$$dz = \tau \odot b(z)dt$$

where $\odot$ is the elementwise (Hadamard) product, i.e. $\tau \odot b(z) = (\tau_1 b_1(z), \cdots, \tau_D b_D(z))$.

In the basic TE-PDMP, upon the occurrence of an event, the value of τ is deterministically flipped to its negative. In the case of higher-dimensional τ, this approach will be limited: even though there are 2^D possible values for τ, over the course of the process, it would only cycle between two values, namely, its initial value and its negative. As such, it is reasonable to consider more general classes of updates for the τ variable, with an eye towards applying them after a jump occurs in a Split-TE-PDMP.

Definition 2.2.5. *Given a space $\mathcal{Z}$ equipped with a state space decomposition of size D, a Flipping Operator is an involutive map $\mathcal{F} : \{\pm 1\}^D \to \{\pm 1\}^D$.*

Given a suitably rich class of flipping operators, it will be possible to explore a wider range of values of τ by applying flips in different combinations. However, if there is only the one type of event, it is less clear how one should choose which flipping operator should be applied after such an event.

A useful solution is to consider PDMPs which accommodate *multiple* event types, each of which is tied to a unique flipping operator. That is, take some positive integer M, and consider a PDMP with M event rates, $\{\lambda_j\}_{j=1}^{M}$, each of which is associated to a flipping operator $\mathcal{F}_j$. Upon the occurrence of an event of type j, the state z jumps according to some prescribed jump dynamics Q_j^{τ}, and then the value of τ is replaced by $\mathcal{F}_j\tau$. This can be made more precise as follows:

Definition 2.2.6. *Fix a space $\mathcal{Z}$, and fix a state space decomposition into D parts as $\mathcal{Z} = \prod_{i=1}^{D} \mathcal{Z}_i$. A split time-enriched PDMP (Split TE-PDMP) compatible with this*

decomposition is a stochastic process (Z_t, τ_t) *taking values in* $\prod_{i=1}^{D} \mathcal{Z}_i \times \{\pm 1\}^D$*, and is specified by*

1. *D vector fields* $\{b_i\}_{i=1}^{D}$ *which take* $\mathcal{Z}$*-valued inputs, and produce* $\mathcal{Z}_i$*-valued vectors as outputs. We stipulate that each of the* b_i *is globally* C^1 *and grows at most linearly at infinity. Note that if we write*

$$b = (b_1, \cdots, b_D),$$

   then this guarantees that for all $\tau \in \{\pm 1\}^D$*, the system* $\dot{z} = \tau \odot b(z)$ *has solutions for all time.*

2. *M rate functions* $\{\lambda_j\}_{j=1}^{M} : \mathcal{Z} \times \{\pm 1\}^D \to [0, \infty)$*, each of which is continuous and locally integrable along flows induced by b both forwards and backwards in* ***each time direction****. That is, if* $z(t)$ *is a solution to* $\dot{z} = \tau \odot b(z)$ *for some* $\tau \in \{\pm 1\}^D$*, then for each* j *and for all* $T < \infty$*, we have that* $\int_0^T \lambda_j(z(t), \tau) dt < \infty$*.*

3. *M families of* $\mathcal{Z}$*-valued Markov Transition kernels,* $Q_j^{\tau}(z \to dz')$*, where* τ *ranges over* $\{\pm 1\}^D$*.*

4. *M flipping operators* $\mathcal{F}_j : \{\pm 1\}^D \to \{\pm 1\}^D$

The main differences between Split TE-PDMPs and the previous two settings are the following:

1. There are now multiple τ variables, allowing for finer control of the dynamics of the system. In particular, the particle can be following the flow of the ODE forwards in time on *some* of the subspaces, whilst following it backwards in the others.

2. There are now multiple distinguished event types, and associated to each, a specific involution. More precisely, events of type j occur at the rate λ_j. When such an event occurs, z is then resampled according to the Markov kernel Q_j^{τ}, and τ jumps to $\mathcal{F}_j \tau$. This distinction allows for the sources and effects of qualitatively different event types to be analysed in a disentangled fashion.

The pseudocode in Algorithm 29 outlines how to simulate such a process. For Split TE-PDMPs of the form described above, the generator is given by

$$\begin{aligned} \mathcal{L}u(z, \tau) = &\sum_{i=1}^{D} \tau_i \langle b_i(z), \nabla_{z_i} u(z, \tau) \rangle \\ &+ \sum_{j=1}^{M} \lambda_j(z, \tau) \int_{z' \in \mathcal{Z}} Q_j^{\tau}(z \to dz') \left[u(z', \mathcal{F}_j \tau) - u(z, \tau) \right] \end{aligned}$$

To make this construction more concrete, we offer a few specific examples of Split TE-PDMPs, with an eye towards giving some intuition as to how such splittings arise

Algorithm 29 Split TE-PDMP

1. Initialize (Z_0, τ_0) arbitrarily on $\mathcal{Z} \times \{\pm 1\}^D$, and set $t_0 = 0$.
2. for $k = 1, 2, \ldots$ do
   (a) For $s \geqslant 0$, let z be the solution to

$$\begin{aligned} dz &= \tau_{k-1} \odot b(z)dt \\ z(0) &= Z_{t_{k-1}} \end{aligned}$$

   (b) For $j = 1, \cdots, M$, sample an inter-event time $T_{j,k}$, where $T_{j,k}$ is a non-negative random variable such that

$$\mathbf{P}\left(T_{j,k} \geq t\right) = \exp\left[-\int_{s=0}^{t} \lambda_j\left(z(s), \tau_{k-1}\right) ds\right].$$

   Let $j_0 = \arg\min_j T_{j,k}$, and let $T_k = T_{j_0,k}$.
   (c) For $s \in (0, T_k)$, set $Z_{t_{k-1}+s} = z(s)$.
   (d) Set $t_k = t_{k-1} + T_k$.
   (e) Set $\tau_k = \mathcal{F}_{j_0}\tau_{k-1}$, and sample

$$Z_{t_k} \sim Q_{j_0}^{\tau_{k-1}}\left(Z_{t_k^-}, dZ_{t_k}\right).$$

naturally in applications.

Example 5 (The Zig Zag Process)**.** *In the Zig Zag Process (as presented in [BFR19]), assume that our variable of interest can be written as $x = (x_1, \cdots, x_D)$, and adjoin a velocity variable $v = (v_1, \cdots, v_D)$. Consider again straight-line dynamics, i.e. take $b(x, v) = (v, 0)$, as in the BPS. In the Zig Zag Process, however, there are now $M = D$ types of event — one for each dimension — and thus one must specify D event rates and D jump dynamics. Define now*

$$\begin{aligned} \lambda_i(z, \tau) &= \max\left(\tau_i \langle v_i, \partial_{x_i} U(x)\rangle, 0\right) \\ Q_i^{\tau}(z \to dz') &= \delta(z, dz') \\ \mathcal{F}_i\tau &= (\tau_{-i}, -\tau_i) \end{aligned}$$

Informally, in the Zig-Zag Process, an event of type i occurs when the particle is travelling into regions of lower probability in the i^{th} coordinate. After an event of a given type, the particle changes its direction in the corresponding coordinate. In the standard formulation of the Zig Zag Process, this manifests throught the jump $v_i \mapsto -v_i$, but in the Split TE-PDMP setup, the jump kernel is instead trivial with respect to z, with all of the jump dynamics absorbed into flipping τ_i. Perhaps surprisingly, this means that the distribution of the velocity v is now degenerate, i.e. $\psi(dv) = \delta(\mathbf{1}_D, dv)$ rather than uniform on $\{\pm 1\}^D$,

as in the original presentation of the Zig Zag Process.

While this might initially appear to be a fairly trivial equivalence (i.e. we have 'just' shifted the jumps from v to τ), it is worth noting that when ψ is nondegenerate (e.g. has continuous support), this construction makes it far simpler to design jump dynamics which lead to the correct stationary measure, as well as accommodating other variations on the Zig Zag Process. This is studied further in Section 5.

Example 6 (Local BPS / Factor Graph BPS)**.** *In the Local BPS (introduced in [BCVD18]), assume instead that the target measure can be written in the form of a factor graph, i.e.*

$$\pi(dx) = \left(\prod_{a \in F} \exp\left(-U_a(x_{\partial a})\right) \right) dx$$

i.e. the measure has a density which can be written as a product of many 'factor potentials' $\exp(-U_a(x_{\partial a}))$ (see e.g. [FKLW97] for an introduction to this topic), each of which depends only on a subset of the x-variables. The Local BPS uses this structure to allow for different types of event corresponding to each factor of this density. In effect, one defines the events such that an event of type a occurs when the term U_a is growing too quickly. More precisely, one has

$$\begin{aligned}
b_i(x,v) &= (v_i, 0) \quad \text{for } i = 1, \cdots, D \\
\lambda_a(x,v,\tau) &= \max\left(\sum_{i \in \partial a} \tau_i \langle v_i, \nabla U_a(x) \rangle, 0 \right) \\
Q_a^\tau(z \to dz') &= \delta(x, dx') \cdot \delta(v_{-\partial a}, dv'_{-\partial a}) \cdot \delta(-R_x^a(v_{\partial a}), dv'_{\partial a}) \\
\text{where } R_x^a(v) &= \left(I - 2 \frac{(\nabla U_a(x_{\partial a}))\,(\nabla U_a(x_{\partial a}))^T}{(\nabla U_a(x_{\partial a}))^T\,(\nabla U_a(x_{\partial a}))} \right) v \\
\mathcal{F}_a \tau &= (\tau_{-\partial a}, -\tau_{\partial a}).
\end{aligned}$$

Note that the gradients of U_a can be understood as being taken with respect to $x_{\partial a}$. We thus have that when factor a incites an event, the velocities adjacent to that factor jump according to the operator R_x^a, and then the time variables τ adjacent to that factor are flipped. A potential benefit of this formulation over the original BPS is that each event modifies only the most-relevant subset of the velocity coordinates.

Example 7 (BPS with Refreshment)**.** *In the naive implementation of the BPS, one only observes events when the particle is heading into regions of lower probability. In some examples, this process will suffer from reducibility issues, e.g. when applied to a spherical Gaussian target, there is a ball around the origin which the process cannot enter. As such, it is typical to add in a 'refreshment' event type, which allows a probability of completely*

resampling the velocity from its marginal, independently of location. This can be written as

$$\begin{aligned}
\lambda_{Natural}(x,v) &= \max\left(\tau\langle v, \nabla U(x)\rangle, 0\right) \\
\lambda_{Refresh}(x,v) &= \gamma > 0 \\
Q^{\tau}_{Natural}(z \to dz') &= \delta(x, dx')\delta(-R_x(v), dv') \\
Q^{\tau}_{Refresh}(z \to dz') &= \delta(x, dx')\psi(dv') \\
\mathcal{F}_{Natural}\tau = \mathcal{F}_{Refresh}\tau &= -\tau.
\end{aligned}$$

While one could, in principle, use mixture distributions to write this algorithm with a single event type and jump dynamics, it can be argued that distinguishing between the different event types further elucidates the structure of the algorithm. This also simplifies the analysis and comparison of algorithms.

Finally, for completeness, we also generalise the definition of trajectorial reversibility to split TE-PDMPs, noting that τ_t will now be vector-valued.

Definition 2.2.7. *Fix a split time-enriched PDMP $(Z_t, \tau_t)_{t\geqslant 0}$, with associated involutions $\{\mathcal{F}_j\}_{j=1}^M$ and a unique stationary measure, and let $(Z'_t, \tau'_t)_{t\geqslant 0}$ denote the law of this PDMP as evolved backwards in time from stationarity. Then we say $(Z_t, \tau_t)_{t\geqslant 0}$ is $\mathcal{F}_j$-trajectorially reversible if*

$$(Z'_t, \tau'_t)_{t\geqslant 0} \overset{d}{=} (Z_t, -\tau_t)_{t\geqslant 0},$$

i.e. the time reversal of the process is equal in law to the original process, with ***all*** *directions of time flipped.*

2.3 PDMPs for Monte Carlo

2.3.1 Main Algorithms and Theorems

In this section, PDMPs are contextualised with regard to their applications in Monte Carlo methods. The focus will thus be on constructing PDMPs which admit a prescribed probability measure μ as their stationary measure.

The tactic pursued in this work is to instead construct a TE-PDMP with stationary measure $\tilde{\mu}$ given by

$$\tilde{\mu}(dz, d\tau) = \mu(dz)R(d\tau),$$

where R is the uniform ('Rademacher') distribution on $\{\pm 1\}$. It is assumed that μ has a C^1 density with respect to Lebesgue measure, given by

$$\mu(dz) = \exp\left(-H(z)\right) dz.$$

In the sequel, it will be useful to work in terms of the following natural objects associated to a given TE-PDMP.

Definition 2.3.1. *For a given driving vector field $b(z)$ and a given target measure $\mu(dz) = \exp(-H(z))dz$, the Raw Event Rate of b with respect to μ is defined as*

$$r(z, \tau) = \tau \left[\langle b(z), \nabla H(z)\rangle - div\, b(z)\right]$$

The Natural Event Rate of b with respect to μ is then defined as

$$\lambda^0(z, \tau) = \sigma(r(z, \tau))$$

where $\sigma(u) = \max(0, u)$.

The raw event rate r is a real-valued quantity which describes the propensity for our process to experience an event at (z, τ). It comprises two terms, an *energy gain* term $\langle b(z, \tau), \nabla H(z)\rangle$, and a *compressibility penalty* term $\operatorname{div} b(z, \tau)$. In effect, the energy gain term encourages events when the process is heading towards regions of lower probability, whereas the compressibility penalty prevents the process from squeezing too much probability mass into any area.

Equally, by virtue of the equality

$$r(z, \tau) \exp(-H(z)) = -\operatorname{div}\left(b(z, \tau) \exp(-H(z))\right), \tag{2.1}$$

one can also think of r as a measure of the extent to which the dynamics are evenly

circulating probability mass around the target measure.

Given r, the natural event rate λ^0 is obtained by taking the positive part, i.e. $\lambda^0 = \sigma(r)$. This allows us to interpret λ^0 as the rate of a bona fide Poisson process.

Definition 2.3.2. *A Refreshment Rate on the space $\mathcal{Z}$ is a function $\gamma : \mathcal{Z} \to [0, \infty)$.*

As discussed in Section 2.3, the refreshment rate γ allows events to occur, even when the dynamics are circulating probability mass well. Typically, γ can be thought of as a tool for introducing additional randomness into the system, which often improves the ergodic properties of the process, allowing it to explore the space by jumping between different flows of the ODE.

Definition 2.3.3. *For a a given driving vector field $b(z)$, a given target measure $\mu(dz) = \exp(-H(z))dz$, and a given refreshment rate $\gamma(z)$, the (Total) Event Rate is defined as*

$$\lambda(z, \tau) = \lambda^0(z, \tau) + \gamma(z)$$

where λ^0 is the natural event rate of b with respect to μ.

The total event rate λ is then obtained as a superposition of the natural event rate and the refreshment rates, and dictates the rate at which events actually occur under the law of the process.

Definition 2.3.4. *For a a given target measure $\mu(dz) = \exp(-H(z))dz$, event rate $\lambda(z, \tau)$, and $\tau \in \{\pm 1\}$, the Jump Measure of (μ, λ, τ) is defined as*

$$J^\tau(dz) \propto \mu(dz)\lambda(z, \tau).$$

Finally, the jump measure J^τ denotes the law of the Z-value of the process, at stationarity, conditioned on a jump just having occurred, with the direction of time just before the jump being τ. This measure allows for the behaviour of the chain to be tracked *at* jumps, rather than only between them. This allows for a simplified analysis of the time-reversed process.

In what follows, a number of conditions will be assumed of our process, which are spelt out more explicitly in Section 4. Informally, the conditions impose that the target measure is smooth, that the flows of the associated ODE are well-behaved, and that the event rates behave sensibly along those flows, such that events happen neither too frequently (i.e. explosively so) nor too infrequently (i.e. an event should happen in finite time).

With these definitions in mind, one can consider in earnest the question "Given a vector field b and a target measure μ, is it possible to design a TE-PDMP which uses the dynamics of b to draw samples from μ?". The first conclusion of this work is to answer this question in the affirmative.

Informal Theorem 1: Let $\mu(dz)$ be a target measure, and let b be a driving vector field. Define an event rate λ by $\lambda = \lambda^0 + \gamma$, where λ^0 is the natural event rate of b with respect to μ, and γ is a refreshment rate. For each $\tau \in \{\pm 1\}$, take Q^τ to be a J^τ-reversible Markov kernel, where J^τ is the jump measure of (μ, λ, τ). Then, the TE-PDMP driven by b, with event rate λ, and with jump dynamics Q^τ, is trajectorially-reversible, and is stationary with respect to $\tilde{\mu}$.

Moreover, a subsequent conclusion is that this characterisation is complete, i.e. that any trajectorially-reversible TE-PDMP with the correct stationary measure must take this form.

Informal Theorem 2: Suppose that $(Z_t, \tau_t)_{t \geqslant 0}$ is a trajectorially-reversible TE-PDMP driven by the vector field b, with event rate λ, jump dynamics Q^τ, and admitting $\tilde{\mu}$ as a stationary measure. Then there exists a refreshment rate γ such that $\lambda = \lambda^0 + \gamma$, where λ^0 is the natural event rate of b with respect to μ, and the kernels Q^τ are reversible with respect to the jump measures J^τ of (μ, λ, τ).

Furthermore, if one wishes to use a split TE-PDMP to draw samples from μ, this is also possible. In this case, one must specify a priori a state space decomposition $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$ and posit a vector field $b = (b_1, \cdots, b_D)$. It will also be helpful to define the following notion.

Definition 2.3.5. *Consider a given driving vector field $b(z)$ and a given target measure $\mu(dz) = \exp(-H(z))\, dz$, both defined on a space $\mathcal{Z}$ admitting a state space decomposition into D parts as $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$, and let M be a positive integer. An Event Rate Decomposition of b with respect to $(\mu, \{\mathcal{Z}_i\}_{i=1}^D)$ into M parts is given by a collection of M functions $(r_1, \ldots, r_M)$, each mapping $\mathcal{Z} \times \{\pm 1\}^D \to \mathbf{R}$, and a collection of M flipping operators, $\{\mathcal{F}_j\}_{j=1}^M$, satisfying*

$$\begin{aligned} r(z, \tau) &= \sum_{j=1}^M r_j(z, \tau) \\ \text{for all } j, \quad r_j(z, -\tau) &= -r_j(z, \tau) \\ \text{for all } j, \quad r_j(z, \mathcal{F}_j \tau) &= -r_j(z, \tau). \end{aligned}$$

Abbreviate such a decomposition of r by $r \vdash \{r_j\}_{j=1}^M$.

By analogy with the previous section, we introduce the following notation and terminology:

Definition 2.3.6. *For a given driving vector field $b(z)$, a given target measure $\mu(dz) = \exp(-H(z))dz$, a state space decomposition into D parts as $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$, and an event rate decomposition $\left(\{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M\right)$, define the the j^{th} Raw Event Rate of b with respect to $(\mu, \{\mathcal{Z}_i\}_{i=1}^D, \{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M)$ as $r_j(z, \tau)$. The j^{th} Natural Event Rate of b with respect*

to $(\mu, \{\mathcal{Z}_i\}_{i=1}^D, \{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M)$ *is then defined as*

$$\lambda_j^0(z,\tau) = \sigma(r_j(z,\tau))$$

where $\sigma(u) = \max(0,u)$.

Definition 2.3.7. *Given a a state space decomposition into* D *parts as* $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$ *and an event rate decomposition* $\left(\{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M\right)$*, a Refreshment Rate Collection on the space* $\mathcal{Z} \times \{\pm 1\}^D$ *is a collection of* M *functions* $\{\gamma_j\}_{j=1}^M : \mathcal{Z} \times \{\pm 1\}^D \to [0,\infty)$.

If it holds in addition that

$$\begin{aligned} &\text{for all } j, \quad \gamma_j(z,-\tau) = \gamma_j(z,\tau) \\ &\text{for all } j, \quad \gamma_j(z,\mathcal{F}_j\tau) = \gamma_j(z,\tau) \end{aligned}$$

then we say that the refreshment rate collection is compatible with the state space and event rate decompositions.

Definition 2.3.8. *For a given driving vector field* $b(z)$*, a given target measure* $\mu(dz) = \exp(-H(z))dz$*, a state space decomposition into* D *parts as* $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$*, an event rate decomposition* $\left(\{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M\right)$*, and a refreshment rate collection* $\{\gamma_j\}_{j=1}^M$*, the* j^{th} *(Total) Event Rate is defined as*

$$\lambda_j(z,\tau) = \lambda_j^0(z,\tau) + \gamma_j(z,\tau)$$

where λ_j^0 *is the* j^{th} *natural event rate of* b *with respect to* $(\mu, \{\mathcal{Z}_i\}_{i=1}^D, \{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M)$.

Definition 2.3.9. *For a given target measure* $\mu(dz) = \exp(-H(z))dz$*, a collection of event rates* $\{\lambda_j(z,\tau)\}_{j=1}^M$*, and* $\tau \in \{\pm 1\}^D$*, the* j^{th} *Jump Measure of* $(\mu, \{\lambda_j\}_{j=1}^M, \tau)$ *is defined as*

$$J_j^\tau(dz) \propto \mu(dz)\lambda_j(z,\tau).$$

Define now a modified extended target measure

$$\tilde{\mu}(dz, d\tau) = \mu(dz) \prod_{i=1}^D R(d\tau_i).$$

The corresponding (informal) theorems are as follows:

Informal Theorem 3: Let $\mu(dz)$ be a target measure, and let b be a driving vector field. Fix a state space decomposition into D parts as $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$, an event rate decomposition $\left(\{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M\right)$, and a refreshment rate collection $\{\gamma_j\}_{j=1}^M$ which is compatible with these decompositions. Define a sequence of event rates $\{\lambda_j\}_{j=1}^M$ by

$\lambda_j(z,\tau) = \lambda_j^0(z,\tau) + \gamma_j(z,\tau)$, where λ_j^0 is the j^{th} natural event rate of b with respect to $(\mu, \{\mathcal{Z}_i\}_{i=1}^D, \{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M)$. For each $\tau \in \{\pm 1\}^D, j \in \{1, \cdots, M\}$, take Q_j^τ to be a J_j^τ-reversible Markov kernel, where J_j^τ is the j^{th} *Jump Measure* of $(\mu, \{\lambda_j\}_{j=1}^M, \tau)$, and such that for all $\tau \in \{\pm 1\}^D, Q_j^\tau = Q_j^{-\mathcal{F}_j\tau}$.

Then, the Split TE-PDMP driven by b, with event rates $\{\lambda_j\}_{j=1}^M$, and with jump dynamics $\{Q_j^\tau\}_{j=1}^M$, is trajectorially-reversible and stationary with respect to $\tilde{\mu}$. Moreover, the event rates of this process satisfy $\lambda_j(z,\tau) = \lambda_j(z, -\mathcal{F}_j\tau)$ for all j.

Informal Theorem 4: Let $\mu(dz)$ be a target measure, and b be a driving vector field. Fix a state space decomposition into D parts as $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$, and a sequence of flipping operators $\{\mathcal{F}_j\}_{j=1}^M : \{\pm 1\}^D \to \{\pm 1\}^D$. Suppose that $(Z_t, \tau_t)_{t \geqslant 0}$ is a trajectorially-reversible Split TE-PDMP, driven by the vector field b, with event rates $\{\lambda_j\}_{j=1}^M$, with jump dynamics $\{Q_j^\tau\}_{j=1}^M$, using $\{\mathcal{F}_j\}_{j=1}^M : \{\pm 1\}^D \to \{\pm 1\}^D$ as flipping operators, and admitting $\tilde{\mu}$ as a stationary measure. Assume also that for all j, it holds that $\lambda_j(z,\tau) = \lambda_j(z, -\mathcal{F}_j\tau)$ and $Q_j^\tau = Q_j^{-\mathcal{F}_j\tau}$.

Then,

1. There exists an event rate decomposition $r \vdash \{r_j\}_{j=1}^M$ which is compatible with the collection of flipping operators $\{\mathcal{F}_j\}_{j=1}^M$.
2. There exists a refreshment rate collection $\{\gamma_j\}_{j=1}^M$ which is compatible with $\{\mathcal{F}_j\}_{j=1}^M$, such that

$$\text{for all } j, \quad \lambda_j = \lambda_j^0 + \gamma_j,$$

   where λ_j^0 is the j^{th} natural event rate of b with respect to $(\mu, \{\mathcal{Z}_i\}_{i=1}^D, \{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M)$.
3. For all j, τ, the jump dynamics Q_j^τ is reversible with respect to the j^{th} *Jump Measure* of $(\mu, \{\lambda_j\}_{j=1}^M, \tau)$.

Formal statements and rigorous proofs of the above informal theorems are provided in Section 4.

2.3.2 Discussion

We pause here to discuss some features of the processes described in the foregoing results, and situate them in the context of existing work and future directions for this field.

2.3.2.1 Convergence to Stationary Measure

It is worth highlighting that the statements of our theorems only guarantee that the PDMPs admit $\tilde{\mu}$ as a stationary measure. In particular, we have not claimed that the PDMPs must converge to this stationary measure, nor have we made quantitative claims

about this convergence. Such claims are typically quite involved, and are addressed in other lines of work (see for example [DBCD19, DGM18a, ADNR18]).

2.3.2.2 Design of Refreshment Rate

A key design choice when constructing a PDMP is the refreshment rate, γ. Although taking $\gamma = 0$ still leads to a process with μ as a stationary measure, it is understood that the resulting process can have issues with reducibility and ergodicity — informally, the chain becomes 'too deterministic', and despite decorrelating quickly, may still be slow to reach equilibrium. At the other end of the spectrum, taking γ too large leads to the process instead being 'more random than necessary', inducing diffusive behaviour and slowing convergence to equilibrium.

In practice, it is observed that PDMPs are generally quite robust to the choice of γ, and taking γ between 0.5 and 1 tends to be a sensible choice across a range of examples, see the comment in [RBC15]. Nevertheless, note that when the target measure in question has either light (lighter than a Gaussian) or heavy (heavier than a double-exponential), then the situation is more subtle, but still manageable; see [DBCD19] for details.

Note also that [AL19] give some Peskun-type ordering results for comparing PDMPs. In the case of the Zig-Zag process, they indicate that in order to minimise asymptotic variance, one should refresh as little as possible. However, it is not immediate that this is the best thing to do given a finite computational budget; the reader is referred to [VM20] for some enlightening examples related to this point.

2.3.2.3 Design of Augmentations

In applications, it is rare to construct a PDMP directly on the original state space, and is more common to first attach an auxiliary variable v with law ψ, write $z = (x, v)$, and ultimately target the extended measure given by

$$\mu(dx, dv) = \pi(dx) \cdot \psi(dv).$$

Note briefly that, in principle, the law of v could also depend on x (c.f. Riemannian Manifold HMC, [GC11]), though to the best of the authors' knowledge, this is yet to be explored. Thus far, the augmentation has generally come from a physical motivation, with v representing a momentum or velocity. While these augmentations have found a great deal of success thus far, it seems clear that there is substantial room for further creativity in this area. The choice of augmentation directly dictates μ, and is closely tied to the choice of dynamics b. As such, this choice will be instrumental to the success of the resulting PDMP.

Some principal considerations are then i) what the augmentation v represents, ii)

which family of dynamics will be used on the extended state space, and iii) how v will be distributed. Typically, the latter of these considerations tends to be most flexible, particularly once the first two have been decided upon.

When working in extended state spaces, the jump dynamics are typically designed to fix x, so that the sample paths of x are continuous, i.e.

$$Q^\tau((x,v) \to (dx', dv')) = \delta(x, dx') \cdot q^\tau(v \to dv'|x)$$

where $q^\tau(\cdot|x)$ must be reversible with respect to $j_x^\tau(dv) \propto \psi(dv)\lambda(x, v, \tau)$.

When working with non-standard dynamics, the process of designing such a q is not always immediate. Here, some general strategies for this task are presented:

1. In some cases, it will be tractable to draw exact samples from j_x^τ, using e.g. rejection sampling. When possible, this is generally a good choice. In fact, there is reason to suspect that this might be the optimal choice, in the sense of Peskun.

2. When this is not possible, one can instead attempt to sample from the restriction of j_x^τ to some simpler set. In particular, if the simpler set is finite and of moderate size, this resampling step effectively becomes trivial. A more general formulation of this construction is the following. Suppose that for each τ, there is a symmetric relation $\sim_\tau$ on $\mathcal{Z}$ such that if $z \sim_\tau z'$, then $\lambda(z, \tau) = \lambda(z', \tau)$. Writing $\mathcal{R}_{z,\tau} = \{z' : z \sim_\tau z'\}$, one can define the kernel

$$Q^\tau(z \to dz') = \mu(dz') \restriction_{z' \in \mathcal{R}_{z,\tau}},$$

   i.e. the kernel $Q^\tau(z \to dz')$, conditioned to land in $\mathcal{R}_{z,\tau}$. This can be shown to be J^τ-reversible, and thus leads to a process with the correct stationary measure. Some examples of this include the Bouncy Particle Sampler, which takes

$$\mathcal{R}_{(x,v),\tau} = \{(x, v')\} \quad \text{where } v' = -v + 2 \cdot \frac{\langle v, \nabla U(x)\rangle}{\|\nabla U(x)\|_2^2} \nabla U(x)$$

   and the Generalised BPS of [WR17], which takes

$$\mathcal{R}_{(x,v),\tau} = \{(\tilde{x}, \tilde{v}) : \tilde{x} = x, \langle v, \nabla U(x)\rangle = \langle \tilde{v}, \nabla U(x)\rangle\}.$$

   In the first case, the resampling step is deterministic; in the second case, it corresponds to sampling the restriction of a Gaussian measure to a linear subspace. In both cases, this is straightforward.

3. When such symmetries and uniformities are not available, one can simply use a Metropolis-Hastings step which targets j_x^τ. It should be noted, however, that when such steps are rejected, the PDMP will directly backtrack on its previous path, which

is typically suboptimal.

2.3.2.4 Computational Implementation Considerations

On the topic of implementing such methods, it should be kept in mind that essentially, one only needs to be able to carry out the following procedures: i) simulate the dynamics of b for all values of τ, ii) simulate events from the relevant Poisson process(es), and iii) simulate transitions from the jump kernels.

Exact simulation of dynamics has been achieved thus far by using straight-line and elliptical dynamics; we hope that more flexible choices of dynamics become tractable going forward. Note that in discrete-time MCMC, many algorithmic advances have come by making use of more efficient *target-informed* dynamics, and we envision that the same will ultimately be true for PDMPs.

While fairly straightforward from the theoretical perspective, simulating the relevant Poisson processes for PDMPs appears (at present) to be the implementation-side bottleneck for the widespread adoption of PDMPs. As the events are usually simulated through a combination of superposing and thinning simpler Poisson processes, a key element ends up being finding tractable upper bounds to $\sigma(r_j(z,\tau))$ as z moves along the flows of the ODE. Typically, these must be derived on a case-by-case basis. It would be of great public utility to construct generic dynamics b for which such bounds are easily derived.

Finally, simulation of the jump dynamics should always be designed so as to be fairly straightforward; the previous section provides some constructive recommendations as to how to make this possible.

2.3.2.5 Optimality and Ordering

Having established in reasonable generality what the conditions are for a TE-PDMP to be trajectorially-reversible with respect to a given measure μ, one may begin to discuss notions of optimality and ordering between PDMPs. Being able to compare the different options is practically important, as it allows for informed decisions to be made on which processes should be expected to converge most rapidly, and thus provide the most useful results.

A first question is how this comparison ought to even be framed. In the discrete-time, reversible setting, a common tool is to use *Dirichlet forms* and the *Peskun ordering*. For K a π-reversible Markov kernel, one can define the Dirichlet form of K, acting on pairs of functions in $L^2(\pi)$ by

$$\mathcal{E}_K(f,g) = \mathbf{E}_{x\sim\pi}\left[\mathbf{E}_{y\sim K(x\to y)}\left[f(x)\cdot\{g(x)-g(y)\}\right]\right] \tag{2.2}$$

$$= \frac{1}{2}\mathbf{E}_{x\sim\pi}\left[\mathbf{E}_{y\sim K(x\to y)}\left[\{f(x)-f(y)\}\cdot\{g(x)-g(y)\}\right]\right]. \tag{2.3}$$

$\mathcal{E}_K$ is then a positive-semidefinite bilinear form, with intimate ties to the spectral decomposition of the Markov kernel K, and thus to the convergence behaviour of the associated Markov chain. Similar notions can be derived for continuous-time, π-reversible Markov processes, through an appropriate re-definition in term of the infinitesimal generator of the process.

The Peskun ordering (see e.g. [Pes73, Tie98, LM08] is based on a comparison principle, which states that if K_1, K_2 are both π-reversible Markov kernels, and if $\mathcal{E}_{K_1}(f,f) \geqslant \mathcal{E}_{K_2}(f,f)$ for all f, then the asymptotic variance of estimators derived from the chain driven by K_1 are *no greater than* the asymptotic variance of estimators derived from the chain driven by K_2. One then says that K_1 'dominates K_2 in the sense of Peskun'. This indicates that, from the Monte Carlo perspective, the chain driven by K_1 should be preferred. Note the slightly subtle point that the Peskun ordering does not establish that either chain converges to equilibrium at a faster rate.

For general non-reversible chains, the Peskun ordering is not directly applicable, and so there is a necessity for new tools with which to study these processes. Fortunately, in the restricted setting of *trajectorially-reversible* processes, there is recent work making inroads in this direction. The work of [AL19] exploits trajectorial reversibility in order to define a generalised Dirichlet form, which allows for a similar comparison principle to be applied. The picture is particularly simple in the discrete-time case, and avoids some of the functional-analytic details which can impede mathematical progress in the continuous-time case. The reader is encouraged to consult the cited work for additional detail on the technical aspects of this construction.

With these tools, it is natural to ask whether one can derive optimal choices of (b, λ, Q) for time-enriched PDMPs, and their split variants. At present, with respect to continuous-time PDMPs, the techniques of [AL19] have only been able to provide rigorous ordering results in a fairly limited setting (comparing two Zig-Zag Processes with distinct event rates, in dimension at most 2) thus far. The present author has not yet been able to extend this any further. Nevertheless, it seems appropriate to conjecture what the optimal settings might be, in the language of this work.

Consider first the most restrictive case: fix a target measure μ, a driving vector field b, and an event rate λ; how should the jump dynamics Q be chosen? As any such Q^τ must be reversible with respect to the jump measure J^τ, our first conjecture is that the optimal jump dynamics are given by the 'most random' such Q^τ, i.e. resampling z' from J^τ itself:

Conjecture 1. *For a given target measure μ, driving vector field b, and event rate λ, the optimal jump dynamics are given by $Q^\tau(z \to dz') = J^\tau(dz') \propto \mu(dz')\lambda(z', \tau)$.*

Conjecture 2. *For a given target measure μ, driving vector field b, and event rate λ, if the position can be partitioned as $z = (x, v)$, and the jump dynamics are required to fix x, then the optimal jump dynamics for v are given by $Q^\tau(v \to dv'|x) \propto \psi(dv')\lambda(x, v', \tau)$.*

The author suspects that, when practically feasible, these choices would genuinely be very efficient, i.e. they are not simply an artifact of the choice to use the Peskun ordering.

Taking another step back, one can fix a target measure μ, and a driving vector field b; how should the event rate λ be chosen? Given that the set of admissible choices is given by the halfspace $\{\lambda : \lambda \geqslant \lambda^0\}$, it is perhaps natural to conjecture that the optimal event rate is the *minimal* choice:

Conjecture 3. *For a given target measure μ and driving vector field b, the optimal event rate is given by $\lambda(z,\tau) = \lambda^0(z,\tau)$, the natural event rate of b with respect to μ.*

In contrast to the previous case, using this choice of event rate in practice requires caution. As described earlier, in some cases (e.g. isotropic Gaussian target), this apparently 'optimal' choice can actually fail to be ergodic for the target measure. This can be highlighted as a potential blind spot of the Peskun ordering; while it is able to compare processes which leave a given measure stationary, it cannot necessarily reason about whether the process will actually *converge* to that measure.

Going one step further, given only a target measure μ, one may want to reason about whether an optimal choice of dynamics exists. One complication is that one can always linearly rescale b to obtain a faster process, and so it is natural to impose some constraints on the class of b we optimise over. One could choose to normalise b by imposing a constraint, e.g. $b \in B \triangleq \{b : \mathbf{E}_\mu[\|b(z)\|_2^2] \leqslant 1\}$, but it is not clear that this is a meaningful choice of normalisation. As such, the author is not yet prepared to speculate further on what an optimal setting of b for a given target measure might be.

Finally, in the light of the Split TE-PDMP formalism, it is natural to wonder whether an optimal splitting exists. The final conjecture of this section that in the continuous-time setting, it is in fact optimal to *not split at all*, i.e.

Conjecture 4. *For a given target measure μ and driving vector field b, the optimal decomposition of r is to take $M = 1$ and set $r_1 = r$.*

Note that a split- and non-split TE-PDMP do not strictly evolve on the same state space, and so some care should be taken in formulating this claim appropriately.

It should also be emphasised that this claim is a statement about ordering processes in the time-scale of the process itself, and not necessarily in terms of computational time. In particular, splittings can often lead to tighter bounds on the corresponding event rates, and hence to more efficient simulation of event times, which can make a substantial impact on how rapidly the chain mixes in real time.

2.3.2.6 Discrete-Time PDMPs

Note that all of the above results are stated in continuous time. Although a number of discrete-time PDMPs (see [Bie16, Gus98, VBCDD17, ST17, WRS18]) have been proposed

and analysed, this work chooses to focus exclusively on the continuous-time setting.

One reason for this is that the continuous-time setting simplifies certain concerns dramatically. For example, in continuous time, the splitting perspective allows for the Local BPS and Zig-Zag Process to be treated in a unified way, as they both simply correspond to decomposing the expression for the raw event rate r. In contrast, the discrete-time analog of the Local BPS (the factorised filter of [Mic16]) does not naturally hint at a discrete-time Zig-Zag Process. The event rates for the Local BPS hinge upon an additive decomposition of the energy H, which works just as easily in discrete time. In contrast, for the Zig-Zag Process, the event rates are based around a coordinate-based decomposition of the rate of change of the energy along trajectories, which does not translate as neatly.

With this being said, it should be remarked that the recent work in this area has greatly improved our understanding of discrete-time PDMPs, and it seems conceivable that extending them to more general dynamics may not be too far away. Indeed, in [Mon19], the author successfully constructs a discrete-time Zig-Zag process, making use of a novel class of Markov chains termed 'kinetic walks'. It will be interesting to see whether this construction can be naturally generalised to systematically translate other PDMPs into the discrete-time setting.

2.3.2.7 Subsampling

One major selling point for PDMPs has been their promise for allowing 'exact' MCMC while using subsampling techniques (see e.g. Sections 4 and 5 of [BFR19]). This aspect has largely been glossed over thus far in this work. The reason for this is that from a theoretical perspective, the subsampling mechanism effectively corresponds to a specific choice of splitting, e.g. in the BPS case

$$\begin{aligned} r(z,\tau) &= \tau \, \langle v, \nabla U(x) \, \rangle \\ &= \tau \, \langle v, \nabla \sum_i U_i(x) \, \rangle \\ &= \sum_i \tau \, \langle v, \nabla U_i(x) \, \rangle \\ \rightarrow r_i(z,\tau) &\triangleq \tau \, \langle v, \nabla U_i(x) \, \rangle, \end{aligned}$$

and similar splittings can be derived for the case of subsampling with control variates. As in the general case, practical success here will depend strongly on finding practical upper bounds on the resulting event rates.

2.3.2.8 A Pipeline for Devising PDMPs

A byproduct of the characterisation provided in the previous sections is that it suggests a modular recipe for devising PDMPs with desired properties. We detail this below, in the hopes that it will be instructive for readers with an interest in devising their own PDMPs for sampling.

When beginning the sampling task in earnest, one must first specify a target measure $\mu(dz)$, and thus implicitly, an energy function $H(z)$. If a PDMP of any sort is to be involved, one must then specify the vector field $b(z)$ which will be driving the dynamics. The raw event rate $r(z, \tau)$ is thus specified automatically, and can be computed explicitly. One can then form a valid event rate λ by combining the natural event rate $\lambda^0(z, \tau) = \sigma(r(z, \tau))$ together with a chosen refreshment rate, $\gamma(z)$. Finally, one computes the jump measure $J^\tau(dz) = \mu(dz) \cdot \lambda(z, \tau)$ using the previous information, and can thus begin to choose a jump kernel Q^τ which is in detailed balance with respect to this jump measure. This workflow is detailed visually in Figure 2.1.

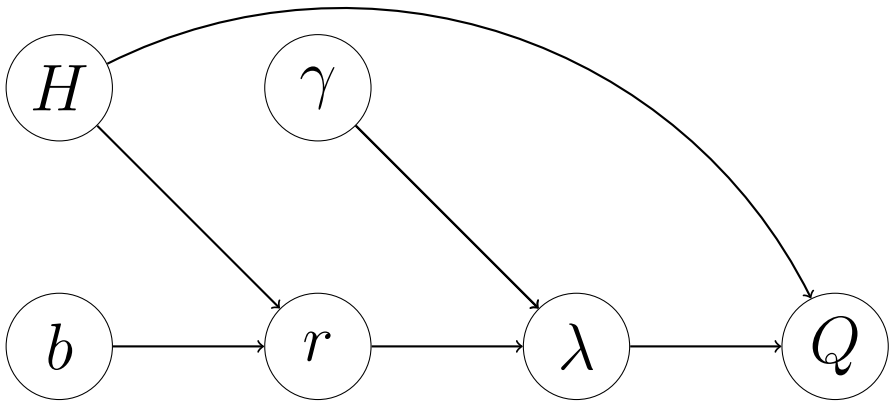

Figure 2.1: Flow Chart for Constructing a TE-PDMP

The workflow for a Split TE-PDMP is similar, but branches off partially after the raw event rate is specified. The branching point is that one must now exhibit an additive decomposition of this raw event rate as

$$r(z, \tau) = \sum_{j=1}^{M} r_j(z, \tau)$$

and decorate each of these summands with an involution $\mathcal{F}_j : \{\pm 1\}^D \to \{\pm 1\}^D$ such that $r_j(z, \mathcal{F}_j\tau) = -r_j(z, \tau)$. Once this is established, the remainder of the workflow is quite similar; one constructs event rates by adding the natural event rates together with a chosen refreshment rate, and then chooses each of the jump kernels to be reversible with respect to the corresponding, fully-specified jump measure. This is detailed visually in Figure 2.2,

noting that the inner plate denotes repetition (i.e. the inner workflow should be carried out for $j = 1, \ldots, M$).

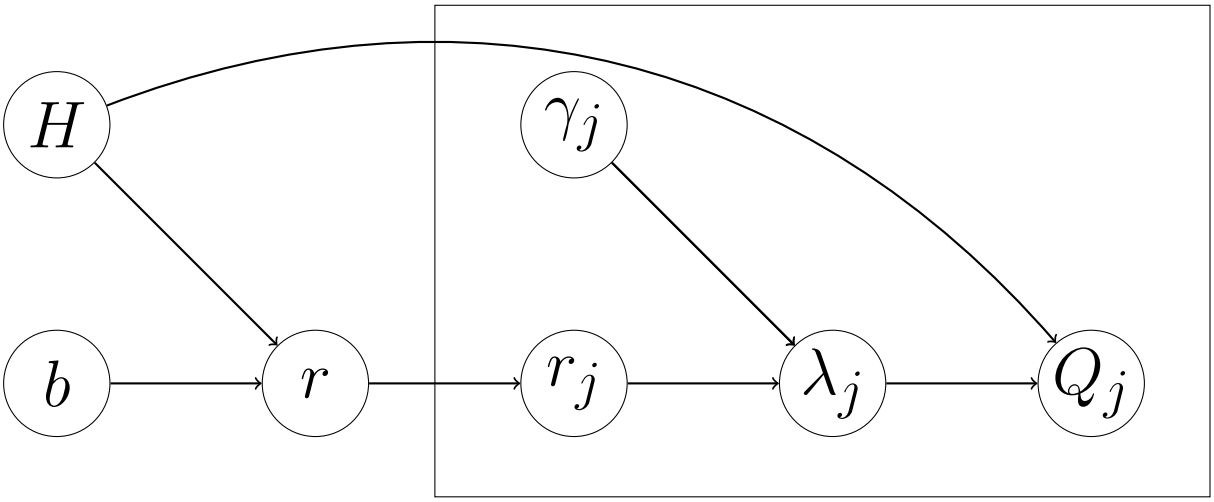

Figure 2.2: Flow Chart for Constructing a Split TE-PDMP

2.4 Theory

2.4.1 Assumptions

In this section, the theorems of the previous section are proven. Throughout, we make the following assumptions, noting that there may be some redundancy among them.

Assumption 1. *Our target measure $\mu(dz)$ is a probability measure with density $\exp(-H(z))$ with respect to Lebesgue measure, where $H(z)$ is C^1 on $\mathcal{Z}$, and the driving vector field b satisfies $\|b(z)\| \leqslant C(1 + \|z\|)$ for some $C \in (0, \infty)$.*

The condition on μ guarantees that the target measure has full support, and thus no boundary complications arise. The condition on b is used to guarantee the existence of flows of the ODE for all time.

Assumption 2. *The process (Z_t, τ_t) admits some stationary measure, $\tilde{\mu}$, and the total event rate λ satisfies the following conditions:*

- *The total event rate $\lambda(z, \tau)$ is continuous in z for each fixed τ.*
- *Under the stationary measure $\tilde{\mu}$, the total event rate has finite expectation, i.e. $\mathbf{E}_{\tilde{\mu}}[\lambda(z, \tau)] < \infty$.*
- *Along the flows of b, λ is locally integrable, and has infinite integral over $(0, \infty)$, i.e. if $z(t)$ is a solution to $dz = b(z)dt$, then for any $T < \infty$, we have $\int_0^T \lambda(z(t))\, dt < \infty$ and $\int_0^\infty \lambda(z(t))\, dt = \infty$. As such, with probability 1, a jump always occurs eventually.*
- *Let N_T be the number of events observed in the period $0 \leqslant t \leqslant T$. For all (z, τ, T), it holds that $\mathbf{E}^{(z,\tau)}[N_T] < \infty$, i.e. from any starting conditions, the chain is non-explosive.*
- *After a jump, the expected waiting time until the next jump is finite, i.e. $\mathbf{E}_{J^\tau}[T_1] < \infty$.*

These conditions are largely lifted from [LP13], with appropriate adaptations and omissions. Note in particular that many of their conditions can be dropped due to other assumptions which are made here; further details relating to this are provided in the appendix. The utility of these conditions is to reason about the time-reversal of the processes we discuss; in particular, to guarantee that they exist.

Assumption 3. *The jump dynamics Q^τ are continuous with respect to their starting location, i.e. for Borel sets A, $Q^\tau(z' \to A) \to Q^\tau(z \to A)$ as $z' \to z$ along the flows of b.*

This condition is also taken from [LP13]; we separate it as an assumption because it involves the jump dynamics, rather than only considering the behaviour of the flow or event rates.

Now, writing $C_c^1(\mathcal{Z})$ for the space of continuously-differentiable functions $u : \mathcal{Z} \to \mathbf{R}$, assume:

Assumption 4. *For $t \geqslant 0$, $u \in C_c^1(\mathcal{Z})$, define the linear operator P_t by*

$$P_t u(z,\tau) = \mathbf{E}[u(Z_t,\tau_t)|Z_0 = z, \tau_0 = \tau].$$

Assume also that the semigroup $(P_t)_{t\geqslant 0}$ is smoothly, compactly approximable (the definition of this is taken from [DGM18b], and is provided in the appendix).

Together with the assumption that the chain is non-explosive and that $\mathbf{E}_{\tilde{\mu}}[\lambda(z,\tau)] < \infty$, this allows for Corollary 22 of [DGM18b] to be applied. For the purposes of this work, the significance of this corollary is that it guarantees that

$$\mathbf{E}_{\tilde{\mu}}[\mathcal{L}u(z,\tau)] = 0 \quad \forall u \in C_c^1(\mathcal{Z}) \implies (Z_t,\tau_t) \text{ admits } \tilde{\mu} \text{ as a stationary measure.}$$

A priori, to show that $\tilde{\mu}$ is a stationary measure for the PDMP, one would have to show that these expectations vanish for all $u \in \mathcal{D}(\mathcal{L})$, which can be considerably more involved.

2.4.2 Correctness of Algorithms and Trajectorial Reversibility

The following theorems concern the formal statement of Informal Theorem 2.3.1 and Informal Theorem 2.3.1, namely, that the processes described are trajectorially reversible, and admit the desired stationary measure.

Theorem 4. *Let (μ, b) be a measure and a vector field satisfying **Assumption 1** and let γ be a refreshment rate. Let the raw event rate r be given by $r(z,\tau) = \tau\left[\langle b(z), \nabla H(z)\rangle - div_z b(z)\right]$, the natural event rate λ^0 be given by $\lambda^0(z,\tau) = \sigma(r(z,\tau))$, and set $\lambda = \lambda^0 + \gamma$. Suppose that λ then satisfies **Assumption 2**. Suppose that for each τ, the kernel Q^τ satisfies **Assumption 3** and is reversible with respect to J^τ, where $J^\tau(dz) \propto \mu(dz)\lambda(z,\tau)$ is the jump measure of the process. Suppose that the semigroup $(P_t)_{t\geqslant 0}$ associated to the process satisfies **Assumption 4**.*

Then the TE-PDMP driven by b, with event rate $\lambda = \lambda^0 + \gamma$, and jump dynamics Q^τ is $\tilde{\mu}$-stationary, where $\tilde{\mu}(dz, d\tau) = \mu(dz)R(d\tau)$, and moreover, is trajectorially reversible with respect to this measure.

Theorem 5. *Let (μ, b) be a measure and a vector field satisfying **Assumption 1**. Fix a state space decomposition into D parts as $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$, an event rate decomposition $\left(\{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M\right)$, and a refreshment rate collection $\{\gamma_j\}_{j=1}^M$ which is compatible with these decompositions.*

*Define a sequence of event rates $\{\lambda_j\}_{j=1}^M$ by $\lambda_j(z,\tau) = \lambda_j^0(z,\tau) + \gamma_j(z,\tau)$, where λ_j^0 is the j^{th} natural event rate of b with respect to $(\mu, \{\mathcal{Z}_i\}_{i=1}^D, \{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M)$. Suppose that for each j, λ_j then satisfies **Assumption 2**.*

For each $\tau \in \{\pm 1\}^D, j \in \{1, \cdots, M\}$, take Q_j^τ to be a J_j^τ-reversible Markov kernel, where J_j^τ is the j^{th} jump measure of $(\mu, \{r_j\}_{j=1}^M, b, \lambda)$, and such that for all $\tau \in \{\pm 1\}^D, Q_j^\tau = Q_j^{-\mathcal{F}_j\tau}$. Suppose that for each j, τ, the kernel Q_j^τ satisfies ***Assumption 3***

Suppose that the semigroup $(P_t)_{t\geqslant 0}$ associated to the process satisfies ***Assumption 4****.*

Then, the Split TE-PDMP driven by b, with event rates $\{\lambda_j\}_{j=1}^M$, and with jump dynamics $\{Q_j^\tau\}_{j=1}^M$, is trajectorially-reversible and stationary with respect to $\tilde{\mu}$. Moreover, the event rates of this process satisfy $\lambda_j(z, \tau) = \lambda_j(z, -\mathcal{F}_j\tau)$ for all j.

We will first prove that the processes are stationary with respect to the desired measure. Then, we can verify that they are trajectorially reversible with respect to that measure.

To this end, we begin with some preliminary lemmas, proofs of which are provided in the appendix. We abuse notation slightly and write $C_c^1(\mathcal{Z})$ for all functions $u : \mathcal{Z} \times \{\pm 1\}^D$ such that for all $\tau \in \{\pm 1\}^D$, we have that $u(\cdot, \tau)$ is compactly supported and continuously differentiable on $\mathcal{Z}$.

Lemma 6. *For all $u \in C_c^1(\mathcal{Z})$,*

$$\mathbf{E}_{\tilde{\mu}}[\langle b(z,\tau), \nabla_z u(z,\tau)\rangle] = \mathbf{E}_{\tilde{\mu}}[r(z,\tau)u(z,\tau)]$$

Lemma 7. *For all $u \in C_c^1(\mathcal{Z})$,*

$$\mathbf{E}_{\tilde{\mu}}\left[\lambda(z,\tau)\int_{z'\in\mathcal{Z}} Q^\tau(z \to dz')u(z', -\tau)\right] = \mathbf{E}_{\tilde{\mu}}[\lambda(z,-\tau)u(z,\tau)]$$

Lemma 8. *For all $u \in C_c^1(\mathcal{Z})$, for all $j = 1, \cdots, M$,*

$$\mathbf{E}_{\tilde{\mu}}\left[\lambda_j(z,\tau)\int_{z'\in\mathcal{Z}} Q_j^\tau(z \to dz')u(z', \mathcal{F}_j\tau)\right] = \mathbf{E}_{\tilde{\mu}}[\lambda_j(z,\mathcal{F}_j\tau)u(z,\tau)]$$

Now, by **Assumption 4**, to establish that the process is $\tilde{\mu}$-stationary, it suffices to show that

$$\forall u \in C_c^1(\mathcal{Z}), \quad \mathbf{E}_{\tilde{\mu}}[(\mathcal{L}u)(z,\tau)] = 0.$$

Since

$$(\mathcal{L}u)(z,\tau) = \langle b(z,\tau), \nabla_z u(z,\tau)\rangle + \lambda(z,\tau)\left(\int_{z'\in\mathcal{Z}} Q^\tau(z \to dz')u(z', -\tau) - u(z,\tau)\right),$$

we can write

$$\begin{aligned}
\mathbf{E}_{\tilde{\mu}}[(\mathcal{L}u)(z,\tau)] &= \mathbf{E}_{\tilde{\mu}}[\langle b(z,\tau), \nabla_z u(z,\tau)\rangle] \\
&\quad + \mathbf{E}_{\tilde{\mu}}\left[\lambda(z,\tau)\int_{z'\in\mathcal{Z}} Q^{\tau}(z \to dz')u(z',-\tau)\right] \\
&\quad - \mathbf{E}_{\tilde{\mu}}[\lambda(z,\tau)u(z,\tau)] \\
&= \mathbf{E}_{\tilde{\mu}}[r(z,\tau)u(z,\tau)] + \mathbf{E}_{\tilde{\mu}}[\lambda(z,-\tau)u(z,\tau)] - \mathbf{E}_{\tilde{\mu}}[\lambda(z,\tau)u(z,\tau)] \\
&= \mathbf{E}_{\tilde{\mu}}[\{r(z,\tau) - [\lambda(z,\tau) - \lambda(z,-\tau)]\}\, u(z,\tau)] \\
&= 0
\end{aligned}$$

where the first equality is by definition, the second by applying the preceding lemmata, the third by rearranging terms, and the fourth by calculating that $\lambda(z,\tau) - \lambda(z,-\tau) = r(z,\tau)$. An entirely analogous argument supplies the proof for the Split TE-PDMP case.

Next, we establish trajectorial reversibility. In the proof, we directly address the basic TE-PDMP case, noting that the same argument applies in the Split TE-PDMP case.

Proof. Let (Z_t, τ_t) be a TE-PDMP driven by the vector field b, with stationary measure $\tilde{\mu}$ given by $\tilde{\mu}(dz, d\tau) = \mu(dz)R(d\tau)$, event rate given by $\lambda = \lambda^0 + \gamma$, where λ^0 is the natural event rate of b with respect to μ, and jump dynamics Q^{τ}.

By **Assumptions 2, 3**, the results of [LP13] (hereafter LP) can be applied.

Define the 'embedded chain' $((Z_k^-, \tau_k^-), (Z_k^+, \tau_k^+))_{k\geqslant 1}$ as the state of the process immediately before and after the k^{th} jump occurs, respectively. Applying Proposition 2 and Theorem 1 of LP, deduce that $((Z_k^-, \tau_k^-), (Z_k^+, \tau_k^+))$ has stationary measure given by

$$\Pi\left((dz^-, d\tau^-), (dz^+, d\tau^+)\right) \propto \mu(dz^-)R(d\tau^-)\lambda(z^-,\tau^-)Q^{\tau^-}(z^- \to dz^+)\delta(-\tau^-, d\tau^+).$$

Define $\Pi^+(dz^+, d\tau^+), \Pi^-(dz^-, d\tau^-)$ as the marginal distributions under Π of (z^+, τ^+) and (z^-, τ^-) respectively, noting that $\Pi^-(dz^-, d\tau^-) \propto \mu(dz^-)R(d\tau^-)\lambda(z^-,\tau^-) = J^{\tau^-}(dz^-)$ is equal to the jump measure, as defined earlier. Moreover, using the J^{τ}-reversibility of the jump dynamics Q^{τ}, see that

$$\begin{aligned}
\Pi\left((dz^-, d\tau^-), (dz^+, d\tau^+)\right) &\propto \mu(dz^-)R(d\tau^-)\lambda(z^-,\tau^-)Q^{\tau^-}(z^- \to dz^+)\delta(-\tau^-, d\tau^+) \\
&= \mu(dz^+)R(d\tau^-)\lambda(z^+,\tau^-)Q^{\tau^-}(z^+ \to dz^-)\delta(-\tau^-, d\tau^+) \\
&= \mu(dz^+)R(d\tau^+)\lambda(z^+,-\tau^+)Q^{-\tau^+}(z^+ \to dz^-)\delta(-\tau^+, d\tau^-)
\end{aligned}$$

where the latter equality comes by noting that the joint distribution constrains that $\tau^- = -\tau^+$. As $Q^{-\tau^+}(z^+ \to dz^-)\delta(-\tau^+, d\tau^-)$ is a Markov kernel, one can marginalise out

(z^-, τ^-) to obtain that

$$\Pi^+(dz^+, d\tau^+) \propto \mu(dz^+)R(d\tau^+)\lambda^*(z^+, \tau^+),$$

where λ^* is defined by $\lambda^*(z, \tau) = \lambda(z, -\tau)$.

Applying Proposition 4 of LP, we can calculate that the time reversal of our process is also a PDMP, with event rate given by $\lambda^*(z, \tau)$. Moreover, unpacking Proposition 5 of LP, the jump dynamics of the reversed process Q^* must satisfy

$$\Pi^+(dz^+, d\tau^+)Q^*((z^+, \tau^+) \to (dz^-, d\tau^-)) = Q((z^-, \tau^-) \to (dz^+, d\tau^+))\Pi^-(dz^-, d\tau^-).$$

Expand the left-hand side as

$$\begin{aligned}
&\Pi^+(dz^+, d\tau^+)Q^*((z^+, \tau^+) \to (dz^-, d\tau^-)) \\
=&\mu(dz^+)R(d\tau^+)\lambda^*(z^+, \tau^+)Q^*((z^+, \tau^+) \to (dz^-, d\tau^-)) \\
=&\mu(dz^+)R(d\tau^+)\lambda^*(z^+, \tau^+)Q^*((z^+, \tau^+) \to (dz^-, d\tau^-)),
\end{aligned}$$

and the right-hand side as

$$\begin{aligned}
&Q((z^-, \tau^-) \to (dz^+, d\tau^+))\Pi^-(dz^-, d\tau^-) \\
=&\mu(dz^-)R(d\tau^-)\lambda(z^-, \tau^-)Q^{\tau^-}(z^- \to dz^+)\delta(-\tau^-, d\tau^+) \\
=&\mu(dz^+)R(d\tau^+)\lambda^*(z^+, \tau^+)Q^{-\tau^+}(z^+ \to dz^-)\delta(-\tau^+, d\tau^-),
\end{aligned}$$

this equality can be rewritten as

$$\begin{aligned}
&\mu(dz^+)R(d\tau^+)\lambda^*(z^+, \tau^+)Q^*((z^+, \tau^+) \to (dz^-, d\tau^-)) \\
=&\mu(dz^+)R(d\tau^+)\lambda^*(z^+, \tau^+)Q^{-\tau^+}(z^+ \to dz^-)\delta(-\tau^+, d\tau^-),
\end{aligned}$$

again using the J^τ-reversibility of Q^τ, the constraint that $\tau^- = -\tau^+$, and the definition of λ^*. Taking regular conditional probabilities of (z^-, τ^-) with respect to (z^+, τ^+), we obtain that

$$Q^*((z^+, \tau^+) \to (dz^-, d\tau^-)) = Q^{-\tau^+}(z^+ \to dz^-)\delta(-\tau^+, d\tau^-),$$

or

$$Q^*((z^+, \tau^+) \to (dz^-, d\tau^-)) = (Q^*)^{\tau^+}(z^+ \to dz^-)\delta(-\tau^+, d\tau^-),$$

where $(Q^*)^\tau(z \to dz') = Q^{-\tau}(z \to dz')$ for each τ.

Finally, the dynamics of the forward process are driven by $b(z, \tau) = \tau b(z)$; it is

immediate that the dynamics of the backward process are driven by $b^*(z,\tau) = -b(z,\tau) = b(z,-\tau)$. We thus know that the time-reversal of the process is a PDMP, driven by b^*, with event rate λ^* and jump dynamics Q^*.

To establish trajectorial reversibility, it must be shown that the original TE-PDMP $(Z_t,\tau_t)_{t\geqslant 0}$ and its time-reversal with τ flipped, $(Z_t^*,-\tau_t^*)_{t\geqslant 0}$, when each initialised at stationarity (i.e. from $\tilde{\mu}$), have the same law. We will do so by exhibiting a coupling.

As both chains admit the same stationary measure, $\tilde{\mu}$, we can couple them exactly at $t=0$, by

$$\begin{aligned}(z,\tau) &\sim \tilde{\mu}\\ (Z_0,\tau_0) &= (z,\tau)\\ (Z_0^*,\tau_0^*) &= (z,-\tau).\end{aligned}$$

We will then show, by induction, that the processes can be coupled so that i) jumps occur for both processes at the same times, ii) the processes remain equal after the jumps occur, and iii) the processes stick together in between jumps. By **Assumption 2**, the process is non-explosive, and so such a coupling will be well-defined for all time, proving equality in law.

For i), suppose that the k^{th} jump occurred at time t_k, and that the processes are equal immediately after the jump, i.e. we have $Z_{t_k}^+ = (Z^*)_{t_k}^+, \tau_{t_k}^+ = -(\tau^*)_{t_k}^+$. The initial value problems defining the flows for both processes are then identical, and thus so are their flows, i.e. until the next jump occurs, $Z_t = Z_t^*$ and $\tau_t = -\tau_t^*$. Moreover, the event rates for the two processes are synchronised, as $\lambda^*(Z_t^*,\tau_t^*) = \lambda(Z_t^*,-\tau_t^*) = \lambda(Z_t,\tau_t)$. As the two Poisson processes governing the jump times are identical, we couple them so that the next jump occurs at the same time for both processes, i.e. $t_{k+1} = t_{k+1}^*$. Now, because $(Q^*)^{\tau}(z\to dz') = Q^{-\tau}(z\to dz')$, we have that

$$\begin{aligned}(Q^*)^{(\tau^*)_{t_{k+1}}^-}((Z^*)_{t_{k+1}}^- \to dz') &= Q^{(-\tau^*)_{t_{k+1}}^-}((Z^*)_{t_{k+1}}^- \to dz')\\ &= Q^{\tau_{t_{k+1}}^-}((Z^*)_{t_{k+1}}^- \to dz')\\ &= Q^{\tau_{t_{k+1}}^-}(Z_{t_{k+1}}^- \to dz'),\end{aligned}$$

i.e. the laws of $Z_{t_{k+1}}^+$ and $(Z^*)_{t_{k+1}}^+$ are identical. As such, we again couple them to be exactly equal. Finally, the jumps of τ for both processes are deterministic flips. As $\tau_{t_{k+1}}^- = -(\tau^*)_{t_{k+1}}^-$ before the jump, it then holds deterministically that $\tau_{t_{k+1}}^+ = -(\tau^*)_{t_{k+1}}^+$, i.e. they remain opposed after the jump. As such, the two processes can be coupled exactly for all time and are thus equal in law.

□

2.4.3 Completeness of Characterisation

These theorems represent the formal statement of Informal Theorem 2.3.1 and Informal Theorem 2.3.1.

Theorem 9. *Let (μ, b) be a measure and vector field satisfying **Assumption 1**. Suppose $(Z_t, \tau_t)_{t\geqslant 0}$ is a trajectorially-reversible TE-PDMP driven by b, with event rate λ satisfying **Assumption 2**, stationary measure $\tilde{\mu}$, where $\tilde{\mu}(dz, d\tau) = \mu(dz)R(d\tau)$, and jump kernels Q^τ satisfying **Assumption 3**.*

Then,

1. *For $\tau \in \{\pm 1\}$, Q^τ is J^τ-reversible, where $J^\tau(dz) \propto \mu(dz)\lambda(z,\tau)$ is the jump measure of the process, and*
2. *There exists a refreshment rate $\gamma : \mathcal{Z} \to [0,\infty)$ such that the event rate is given by $\lambda = \lambda^0 + \gamma$, where λ^0 is the natural event rate of the vector field b with respect to the measure μ.*

Proof. Under the assumption of trajectorial reversibility, we have that the pre-/post-jump measure Π, given by

$$\Pi((dz^-, d\tau^-), (dz^+, d\tau^+)) \propto \mu(dz^-)R(d\tau^-)\lambda(z^-,\tau^-)Q^{\tau^-}(z^- \to dz^+)\delta(-\tau^-, d\tau^+)$$

is symmetric up to a sign flip in the τ variables, i.e. $S_\# \Pi = \Pi$, where $S((z^-,\tau^-), z^+, \tau^+)) = ((z^+, -\tau^+), z^-, -\tau^-))$. This can be written as

$$\begin{aligned} & \mu(dz^-)R(d\tau^-)\lambda(z^-,\tau^-)Q^{\tau^-}(z^- \to dz^+)\delta(-\tau^-, d\tau^+) \\ = \;& \mu(dz^+)R(d(-\tau^+))\lambda(z^+,-\tau^+)Q^{-\tau^+}(z^+ \to dz^-)\delta(\tau^+, d(-\tau^-)) \end{aligned}$$

Taking regular conditional probabilities of (z^-, τ^-, z^+) with respect to τ^+ on both sides, and using the symmetry of R, this becomes

$$\mu(dz^-)R(d\tau^-)\lambda(z^-,\tau^-)Q^{\tau^-}(z^- \to dz^+) = \mu(dz^+)R(d\tau^-)\lambda(z^+,\tau^-)Q^{\tau^-}(z^+ \to dz^-),$$

and taking regular conditional probabilities once more, this time of (z^-, z^+) with respect to τ^-, see that

$$\begin{aligned} \mu(dz^-)\lambda(z^-,\tau^-)Q^{\tau^-}(z^- \to dz^+) &= \mu(dz^+)\lambda(z^+,\tau^-)Q^{\tau^-}(z^+ \to dz^-), \\ \text{i.e.} \quad J^{\tau^-}(dz^-)Q^{\tau^-}(z^- \to dz^+) &= J^{\tau^-}(dz^+)Q^{\tau^-}(z^+ \to dz^-). \end{aligned}$$

Thus, any TE-PDMP of the form described must have jump dynamics which are reversible with respect to the jump measure.

To characterise the form of the event rate, we first note that

$$\forall u \in C_c^1(\mathcal{Z}), \quad \mathbf{E}_{\tilde{\mu}}[(\mathcal{L}u)(z,\tau)] = 0.$$

Retracing the steps in the proof of Theorem 4, and noting that Lemma 7 applies due to the J^τ-reversibility of Q^τ, we can deduce that

$$\lambda(z,\tau) - \lambda(z,-\tau) = r(z,\tau).$$

If we write $\lambda(z,\tau) - \lambda^0(z,\tau) = \gamma(z,\tau)$, this gives that $\gamma(z,\tau) - \gamma(z,-\tau) = 0$, and hence that γ is independent of τ, i.e. there exists a function $\gamma : \mathcal{Z} \to \mathbf{R}$ such that

$$\lambda(z,\tau) = \lambda^0(z,\tau) + \gamma(z).$$

To deduce positivity of γ, we note that λ must be nonnegative. Now, for each z, there exists a τ such that $r(z,\tau) \leqslant 0 \implies \lambda^0(z,\tau) = 0$. As such,

$$\begin{aligned}
\forall \tau, \quad \lambda(z,\tau) \geqslant 0 &\implies \inf_\tau \lambda(z,\tau) \geqslant 0 \\
&\implies \inf_\tau \left\{\lambda^0(z,\tau) + \gamma(z)\right\} \geqslant 0 \\
&\implies \inf_\tau \left\{\lambda^0(z,\tau)\right\} + \gamma(z) \geqslant 0 \\
&\implies \gamma(z) \geqslant 0.
\end{aligned}$$

Thus, we have that

$$\lambda(z,\tau) = \lambda^0(z,\tau) + \gamma(z).$$

for some nonnegative function γ, and the proof is complete. □

Theorem 10. *Let $\mu(dz)$ be a target measure, and b be a driving vector field satisfying **Assumption 1**.*

Fix a state space decomposition into D parts as $\mathcal{Z} = \prod_{i=1}^D \mathcal{Z}_i$, and a sequence of flipping operators $\{\mathcal{F}_j\}_{j=1}^M : \{\pm 1\}^D \to \{\pm 1\}^D$.

*Suppose that $(Z_t, \tau_t)_{t \geqslant 0}$ is a trajectorially-reversible Split TE-PDMP, driven by the vector field b, with event rates $\{\lambda_j\}_{j=1}^M$, with jump dynamics $\{Q_j^\tau\}_{j=1}^M$, using $\{\mathcal{F}_j\}_{j=1}^M : \{\pm 1\}^D \to \{\pm 1\}^D$ as flipping operators, and admitting $\tilde{\mu}$ as a stationary measure. Suppose that the event rates $\{\lambda_j\}_{j=1}^M$ all satisfy **Assumption 2**, and the jump kernels Q_j^τ all satisfy **Assumption 3**.*

Assume also that for all j, it holds that $\lambda_j(z,\tau) = \lambda_j(z, -\mathcal{F}_j\tau)$ and $Q_j^\tau = Q_j^{-\mathcal{F}_j\tau}$.

Then,

1. *There exists an event rate decomposition* $r \vdash \{r_j\}_{j=1}^M$ *which is compatible with the collection of flipping operators* $\{\mathcal{F}_j\}_{j=1}^M$.
2. *There exists a refreshment rate collection* $\{\gamma_j\}_{j=1}^M$ *which is compatible with* $\{\mathcal{F}_j\}_{j=1}^M$, *such that*

$$\text{for all } j, \quad \lambda_j = \lambda_j^0 + \gamma_j,$$

where λ_j^0 *is the* j^{th} *natural event rate of* b *with respect to* $(\mu, \{\mathcal{Z}_i\}_{i=1}^D, \{r_j\}_{j=1}^M, \{\mathcal{F}_j\}_{j=1}^M)$.
3. *For all* j, τ, *the jump dynamics* Q_j^τ *is reversible with respect to the jump measure* J_j^τ.

Proof. The proof that each set of jump dynamics is reversible with respect to its corresponding jump measure proceeds similarly to before, using the assumption of trajectorial reversibility together with the techniques of [LP13], but instead looking at the embedded chains corresponding to events of type j. Specifically, the same steps establish that the invariant measure of the embedded chain of pre-post events of type j can be written as

$$\Pi_j((dz^-, d\tau^-), (dz^+, d\tau^+)) \propto \mu(dz^-)R(d\tau^-)\lambda_j(z^-, \tau^-)Q_j^{\tau^-}(z^- \to dz^+)\delta(\mathcal{F}_j\tau^-, d\tau^+).$$

By trajectorial reversibility, this measure is symmetric up to a flip in the τ variables, i.e.

$$\begin{aligned}&\mu(dz^-)R(d\tau^-)\lambda_j(z^-, \tau^-)Q_j^{\tau^-}(z^- \to dz^+)\delta(\mathcal{F}_j\tau^-, d\tau^+)\\ =&\mu(dz^+)R(d(-\tau^+))\lambda_j(z^+, -\tau^+)Q_j^{-\tau^+}(z^+ \to dz-)\delta(-\mathcal{F}_j\tau^+, d(-\tau^-)).\end{aligned}$$

Observing the constraint that $\tau^+ = \mathcal{F}_j\tau^-$ and taking regular conditionals of $z^\pm$ given $\tau^\pm$, it thus holds that

$$\mu(dz^-)\lambda_j(z^-, \tau^-)Q_j^{\tau^-}(z^- \to dz^+) = \mu(dz^+)\lambda_j(z^+, -\mathcal{F}_j\tau^-)Q_j^{-\mathcal{F}_j\tau^-}(z^+ \to dz-).$$

By assumption, for all (j, τ), it holds that $\lambda_j(z, \tau) = \lambda_j(z, -\mathcal{F}_j\tau)$ and $Q_j^\tau = Q_j^{-\mathcal{F}_j\tau}$. This allows the above to be rewritten as

$$\mu(dz^-)\lambda_j(z^-, \tau^-)Q_j^{\tau^-}(z^- \to dz^+) = \mu(dz^+)\lambda_j(z^+, \tau^-)Q_j^{\tau^-}(z^+ \to dz-),$$

which is equivalent to the statement that $Q_j^{\tau^-}$ is reversible with respect to the jump measure $J_j^{\tau^-}$

To establish the existence of an event rate decomposition of r which is compatible with $\{\mathcal{F}_j\}_{j=1}^M$, first note that the same argument from the non-split case leads to

$$\sum_{j=1}^M [\lambda_j(z, \tau) - \lambda_j(z, \mathcal{F}_j\tau)] = r(z, \tau),$$

and so taking $r_j(z,\tau) = \lambda_j(z,\tau) - \lambda_j(z,\mathcal{F}_j\tau)$ yields a valid decomposition. As in the non-split case, one can then deduce that for each j, the function $\lambda_j(z,\tau) - \lambda_j^0(z,\tau)$ is invariant under the action of $\mathcal{F}_j$, thus implying that we can write $\lambda_j(z,\tau) = \lambda_j^0(z,\tau) + \gamma_j(z,\tau)$ for some $\mathcal{F}_j$-invariant function γ_j.

Now, recalling that $r_j(z,\mathcal{F}_j\tau) = -r_j(z,\tau)$, observe that for any (z,τ), it holds that $\min\{\lambda^0(z,\tau), \lambda^0(z,\mathcal{F}_j\tau)\} = 0$. Hence,

$$\begin{aligned}\forall\tau, \quad \lambda(z,\tau) \geqslant 0 &\implies \min\{\lambda(z,\tau), \lambda(z,\mathcal{F}_j\tau)\} \geqslant 0 \\ &\implies \min\left\{\lambda^0(z,\tau), \lambda^0(z,\mathcal{F}_j\tau)\right\} + \gamma_j(z,\tau) \geqslant 0 \\ &\implies \gamma_j(z,\tau) \geqslant 0.\end{aligned}$$

We can thus write that

$$\lambda_j(z,\tau) = \lambda_j^0(z,\tau) + \gamma_j(z,\tau).$$

for some nonnegative and $\mathcal{F}_j$-invariant function γ_j. Recalling our earlier assumption that $\lambda_j(z,\tau) = \lambda_j(z,-\mathcal{F}_j\tau)$, it can be deduced also that γ_j must satisfy $\gamma_j(z,\tau) = \gamma_j(z,-\tau)$, confirming that the putative collection of event rates $\{\gamma_j\}_{j=1}^M$ is compatible with the collection of flipping operators. The proof is thus complete. □

2.5 Applications and New Processes

In this section, a number of new PDMPs are introduced. The focus is on exposition, highlighting how the framework developed in this paper lends itself to the clean development of new processes. Some of the examples focus on accommodating new classes of dynamics, whereas others focus on how model structures can be exploited through appropriate splittings.

2.5.1 Zig-Zag Process with Alternative Velocity Distributions

In the standard presentation of the Zig-Zag Process, it is fixed a priori that the velocity of each coordinate takes values in $\{\pm 1\}$. One application of the results in the previous section is to demonstrate how the Zig-Zag Process can be generalised to other velocity distributions. For now, assume the velocity distribution is of product form, i.e.

$$\psi(dv) = \prod_{i=1}^{D} \psi_i(dv_i).$$

Under the standard Zig-Zag dynamics with $b(x, v) = (v, 0)$, one recovers the same raw event rates for all choices of ψ, as the dynamics do not affect the velocity. However, the jump dynamics must change. In particular, the i^{th} jump measure becomes

$$\begin{aligned} J_i^\tau(x, v) &= \left\{ \pi(dx) \cdot \prod_{j \neq i} \psi_j(dv_j) \right\} \cdot \psi_i(dv_i) \cdot \lambda_i(x, v, \tau) \\ \implies j_{x,i}^\tau(dv_i) &\propto \psi_i(dv_i) \cdot \sigma\left(\tau_i v_i \frac{\partial U}{\partial x_i} \right). \end{aligned}$$

In many cases of interest, one will be able to sample from the restricted jump measure $j_{x,i}^\tau(dv_i)$ directly. For example, assuming with mild loss of generality that $\tau_i \frac{\partial U}{\partial x_i} > 0$, it can be verified that:

- If $\psi_i(dv_i) = \mathcal{N}(dv_i|0, 1)$ has a Gaussian distribution, then the new velocity should be drawn from a Rayleigh distribution, $\hat{\psi}_i(dv_i) \propto v_i \exp\left(-\frac{1}{2} v_i^2\right), v_i > 0$.
- If $\psi_i(dv_i) = \text{Unif}(dv_i|(-1, 1))$ has a Uniform distribution, then the new velocity should be drawn from a Beta distribution, $\hat{\psi}_i(dv_i) \propto v_i \cdot \mathbf{I}[0 \leqslant v_i \leqslant 1]$.

Allowing for these variations in velocity distribution could be useful in developing preconditioned variants of the Zig-Zag process and enable other forms of adaptation.

2.5.2 A Non-Uniform Coordinate Sampler

In the Coordinate Sampler of [WR20], the distribution over velocities is uniform over each of the $2d$ axis-aligned unit vectors. This indicates that the process will eventually spend an equal amount of time travelling in each direction. In some situations, this may not be appropriate, particularly when the target measure only varies significantly in few directions. As such, one may consider taking the distribution over velocities to be non-uniform, e.g.

$$\psi(dv) = \sum_{i=1}^{D} p_i \cdot \left(\frac{\delta(e_i, dv) + \delta(-e_i, dv)}{2} \right)$$

where $\{p_i\}_{i=1}^D$ is a vector of probabilities, summing to 1, and $\{e_i\}_{i=1}^D$ are the coordinate vectors. This formulation allows for the emphasis of different directions.

Extending the Coordinate Sampler to this setting is straightforward. The dynamics remain the same, the event rates remain the same, and all that needs to change is the jump kernel. Computing the jump measure as

$$\begin{aligned} J^\tau(x, v) &= \pi(dx)\psi(dv)\lambda(x, v, \tau) \\ \text{where } \lambda(x, v, \tau) &= \left(\tau \langle v, \nabla U(x) \rangle\right)_+, \end{aligned}$$

it can be seen that it suffices to set the new velocity to $\pm e_i$ with probability proportional to $p_i \left(\pm\tau \frac{\partial U}{\partial x_i} \right)_+$ in order to sample from the correct invariant measure. Note that this resampling can be done exactly in time $O(D)$, as in the original scheme. An interesting open problem is to derive suitable heuristics and justifications for non-uniform settings of p. This could allow for efficient adaptive variants of the Coordinate Sampler which are able to focus their attention on the most challenging coordinates in the target distribution.

2.5.3 Dynamics with Acceleration

Consider the setting of the Bouncy Particle Sampler, i.e. one wants to sample from $\pi(x) = \exp(-U(x))$, using an auxiliary velocity variable $v \sim \psi(dv) = \mathcal{N}(dv|0, I)$. A number of existing PDMPs in this setting use straight-line dynamics, i.e. $b(x, v) = (v, 0)$. A benefit of this setting is that the paths of the process can be expressed as polynomials in t, which allows for comparatively simple bounds on the event rate.

With this in mind, one could begin to consider 'higher-order' PDMPs whose sample paths evolve as a polynomial in t. For example, consider a variant of the BPS in which the velocity of the particle is also changing at a constant rate, i.e. $b(x, v) = (v, a)$ with $a \in \mathbf{R}^d$ fixed. The flows of this vector field can be given explicitly as polynomials in t, and

one can then compute the resulting raw event rate as

$$\begin{aligned} r(x,v) &= \langle v, \nabla U(x) \rangle + \langle a, v \rangle \\ &= \langle v, \nabla U(x) + a \rangle. \end{aligned}$$

One can interpret this as a standard BPS, applied to the biased potential $U_a(x) = U(x) + \langle a, x \rangle$. One can even take this construction a step further and allow a to vary as well, e.g. assert that $a \sim \mathcal{N}(da|0, I)$, and take $b(x, v, a) = (v, a, 0)$. One would obtain the same event rates again, though now with the tacit assumption that the acceleration a should also be resampled at event times.

A potential benefit of this process would be that it accommodates more interesting paths than straight lines, while also admitting tractable expressions for the paths. Due to the faster movement of the paths, however, it might be expected that events happen more frequently, or that existing bounds on event rates might become quite loose. The extent to which these tradeoffs can be navigated effectively will govern the practicality of simulating such processes.

2.5.4 Nonlinear Dynamics

PDMPs with linear dynamics, such as the BPS and the Zig-Zag process, have the property that when sampling from distributions on constrained spaces, they may run into boundary complications. This may be considered either a bug or a feature. If it is preferred not to worry about such issues, it may be desirable to directly construct dynamics which remain on the interior of the space. For example, when sampling variables which are constrained to remain positive, one could use the vector field $b(x, v) = (xv, 0)$, with paths $x(t) = x(0) \cdot \exp(vt)$, which remain positive for all times. In this case, the flow is no longer incompressible, and so the divergence term in the raw event rate surfaces:

$$r(x,v) = x \cdot v \cdot U'(x) - v.$$

Similarly, for variables constrained to lie in $(-1, 1)$, one could define the vector field $b(x, v) = ((1 - x^2) \cdot v, 0)$, which admits sample paths $x(t) = \tanh\big(\tanh^{-1}(x(0)) + vt\big)$. Again, the flow is compressible, and one computes the raw event rate as

$$r(x,v) = (1 - x^2) \cdot v \cdot U'(x) + 2xv.$$

Similar constructions may be useful for variables which are constrained to lie on manifolds, where it is preferable to let the variables evolve directly on the manifold. One challenge associated with these flows is that computing bounds on the event rates may be especially difficult.

2.5.5 Zig-Zag Process with Localised Events

The Zig-Zag Process simplifies this computation by decomposing the natural event rate according to contributions from different coordinates in the state space. By contrast, the Local BPS simplifies the computation of event rates by decomposing the natural event rate according to contributions from distinct factors in the target measure. The relative benefits of each of these approaches depends on the scenario: the Zig-Zag Process is naturally geared towards simplifying high-dimensional problems, whereas the Local BPS is adapted to the setting in which the target measure is composed as a product of many terms, each of which depends on few variables. There are reasonable parallels to the distinction between coordinate descent and stochastic gradient descent methods in optimisation.

In practice, it is common to encounter both of these issues at once, that is, a high-dimensional target measure composed of many terms. It is thus natural to consider hybrid approaches, with the aim of recovering the best of both worlds. One such approach[†] would be to take the contributions from each factor (as in the Local BPS), and then decompose them according to contributions from each coordinate. That is, write

$$\begin{aligned} r(x, v, \tau) &= \langle \tau \odot v, \nabla U(x) \rangle \\ &= \sum_{i \in V} \tau_i \langle v_i, \nabla U(x) \rangle \\ &= \sum_{i \in V} \tau_i \left\langle v_i, \sum_{a \in \partial i} \nabla U_a(x) \right\rangle \\ &= \sum_{i \in V} \sum_{a \in \partial i} \tau_i \langle v_i, \nabla U_a(x) \rangle \end{aligned}$$

and define $r_{i,a}(x, v, \tau) = \tau_i \langle v_i, \nabla U_a(x) \rangle$. For all variable-factor pairs (i, a), one can then define $\mathcal{F}_{i,a} = \mathcal{F}_i$ to be the map from $\{\pm 1\}^V$ to itself which flips only τ_i. Theorem 2 then indicates that by prescribing that events of type (i, a) occur at rate $\lambda^0_{i,a}(x, v, \tau) = \sigma\left(r_{i,a}(x, v, \tau)\right)$, and that upon the occurrence of such a jump, the i^{th} velocity is resampled from $v_i \sim \psi(v_i) \cdot \lambda^0_{i,a}(x, v, \tau)$, and the i^{th} direction of time variable is flipped to $-\tau_i$, the resulting process will admit the desired invariant measure.

A downside of this formulation is that there are now more event types to monitor (as many types as there are edges in the associated factor graph), but an upside is that each of these event types should be simpler to control. There is also the somewhat ambiguous effect that each event now affects only a small subset of the velocity variables. To the best of the author's knowledge, this is empirically observed to be beneficial, but a theoretical justification is not yet forthcoming.

[†]During the completion of this work, a related construction was used in [BGvdMS20].

2.5.6 Blocked Bouncy Particle Sampler

In some cases, even when the target measure does not admit a clean factorisation into distinct terms, one might still want to update certain blocks of variables in a coupled manner. Consider now a slight generalisation of the BPS setup, where the velocity is now written as w, and its components come from possibly-distinct Gaussian[‡] distributions, i.e.

$$\psi_i(w_i) = \mathcal{N}(w_i|0, \Sigma_i).$$

Suppose now that we decorate the variables V with a block structure B, where each block $b \in B$ represents a subset of variables in V. The blocks are allowed to overlap. For a variable i, write ∂i for the set of blocks which contain i. Similarly, for a block b, write ∂b for the set of variables which are contained in b.

Assume now that for each variable i, we fix a probability distribution p over the blocks in which it lies, i.e. for each $i \in V$, it holds that

$$\sum_{b \in \partial i} p_{i,b} = 1.$$

One can then develop the decomposition

$$\begin{aligned} r(x, w, \tau) &= \langle \tau \odot w, \nabla U(x) \rangle \\ &= \sum_{i \in V} \tau_i \left\langle w_i, \frac{\partial U}{\partial x_i} \right\rangle \\ &= \sum_{i \in V} \tau_i \left\langle w_i, \left(\sum_{b \in \partial i} p_{i,b} \right) \cdot \frac{\partial U}{\partial x_i} \right\rangle \\ &= \sum_{b \in B} \left(\sum_{i \in \partial b} p_{i,b} \cdot \tau_i \left\langle w_i, \frac{\partial U}{\partial x_i} \right\rangle \right). \end{aligned}$$

Now, define

$$r_b(x, w, \tau) = \sum_{i \in \partial b} p_{i,b} \cdot \tau_i \left\langle w_i, \frac{\partial U}{\partial x_i} \right\rangle,$$

and denote by $\mathcal{F}_b$ the map from $\{\pm 1\}^V$ to itself which flips only $\{\tau_i\}_{i \in \partial b}$. Theorem 2 then suggests taking $\lambda_b(x, w, \tau) = \sigma\left(r_b(x, w, \tau)\right)$ (possibly supplemented by some refreshment

[‡]Other distributions can be accommodated, but working with Gaussian measures streamlines the presentation.

term), and a natural choice for Q_b^τ is to deterministically set

$$v'_{\partial b} = -v_{\partial b} + 2 \cdot \frac{\langle v_{\partial b}, \nabla U(x) \rangle}{\|\nabla U(x)\|_2^2} \nabla U(x)$$
$$v'_{-\partial b} = v_{-\partial b},$$

i.e. a specular reflection, in the same vein as the Local BPS. In this setting, an event of type b indicates that the block $x_{\partial b}$ is heading in an unproductive direction, and so one should modify only the corresponding velocities $v_{\partial b}$. The $p_{i,b}$ parameters denote the extent to which the variable i 'belongs' to the block b.

In [VGS20], this construction is carried out with $\Sigma_i = |\partial i|^2 \cdot I_{d_i}, p_{i,b} \equiv |\partial i|^{-1}$, and then reparametrising $v_i = w_i/|\partial i|$. Note that the choice of Σ_i may lead to anisotropic velocity distributions; the extent to which this is desirable will depend on the nature of the target distribution. An interesting direction would be to explore the potential benefits of choosing non-uniform $p_{i,b}$, perhaps with the goal of making the distribution of the event rates $\{\lambda_b\}_{b \in B}$ more flat, or otherwise easier to control.

2.5.7 PDMPs by Vector Field Switching

In some situations, it may desirable to not restrict oneself to using a single vector field to drive the dynamics of the process, and instead consider a collection of vector fields. For example, suppose that one can exactly evaluate the flows of the vector field b_ω for all ω in some index set Ω. Positing a probability distribution $\psi(d\omega)$ over this index set, one can consider constructing a time-enriched PDMP with invariant measure $\mu(dz) \cdot \psi(d\omega) \cdot R(d\tau)$, where the dynamics are given by

$$dz = \tau \cdot b_\omega(z) dt,$$

with (ω, τ) held constant in between jumps. Following Theorem 2, it is then natural to specify event rates as

$$\lambda(z, \omega, \tau) = \sigma\left(\tau \cdot r_\omega(z)\right)$$
$$\text{where } r_\omega(z) = \langle b_\omega(z), \nabla H(z) \rangle - \text{div} b_\omega(z).$$

Restricting to jump kernels which leave z fixed, any resampling of ω must be reversible with respect to the constrained jump measure

$$j_z^\tau(d\omega) = \psi(d\omega) \cdot \lambda(z, \omega, \tau).$$

In the case where the index set Ω is finite, a sensible solution is to sample exactly from $J_z^\tau(d\omega)$, which can be done in time $O(|\Omega|)$.

A benefit of this construction is that it allows for more interesting dynamics to be obtained by composing elementary flows, rather than working only with a single, simple vector field.

2.5.8 The Leapfrog PDMP

A construction which is closely related to the paradigm of vector field switching is that of vector field *splitting*. Consider the task of sampling from $\pi(dx) = \exp(-U(x))$ while using an auxiliary momentum variable $p \sim \mathcal{N}(0, I)$. In Hamiltonian Monte Carlo, one seeks to approximate the flow of the vector field $b(x, p) = (p, -\nabla U(x))$, which is typically not exactly solvable. The standard numerical approach to approximating the flow is to use a splitting approach, by writing

$$\begin{aligned} b(x,p) &= b_{+1}(x,p) + b_{-1}(x,p) \\ \text{where } b_{+1}(x,p) &= (p, 0) \\ b_{-1}(x,p) &= (0, -\nabla U(x)) \end{aligned}$$

and noting that each of $b_{\pm 1}$ have explicitly solvable, linear flows. The standard 'Leapfrog' integrator then alternates between solving the dynamics of each of these two flows for a fixed unit of time. As this unit tends to 0, the approximate flow which is generated in this way will converge to the true flow.

From the point of view of this work, one can instead consider constructing a continuous-time process which alternates between solving each of these flows for a random amount of time. The PDMP framework provides a simple means for realising this procedure. More precisely, it can be computed that for $\omega \in \{\pm 1\}$, the raw event rate is given by $r_\omega(x, p) = \omega \cdot \langle p, \nabla U(x) \rangle$. One thus defines an event rate λ by

$$\lambda(x, p, \omega) = \sigma\left(r_\omega(x, p)\right)$$

and stipulates that, at jumps, all that happens is that ω flips to $-\omega$, i.e. we switch to following the other flow. In particular, neither x nor p change at a jump. Using a modification of the results presented earlier, one can again show that the dynamics of this process preserve the desired invariant measure; the proof of this can be found in the appendix.

A key aspect of this process is that although the only dynamics which it uses are straight lines (as in the Bouncy Particle Sampler, Zig Zag Process, and other examples), there are still two distinct flows at play, namely, the flow which updates the position, and

the flow which updates the momentum. By alternating between the two types of flow in this structured way, one can expect more interesting dynamical behaviour. Moreover, the momentum updates incorporate information about the gradient of the target measure far more explicitly than in alternative PDMPs, which should lead to better-informed dynamics. The author is currently investigating this further.

Finally, it bears mentioning that in the case where $\pi(x)$ can be written as a change-of-measure from a Gaussian measure, i.e.

$$\frac{d\pi}{d\pi_0}(x) \propto \exp\left(-\Psi(x)\right) \quad \text{where } \pi_0(dx) = \mathcal{N}(x|0, \mathcal{C}),$$

then it is common to define the law of the momentum variable as being $\mathcal{N}(p|0, \mathcal{C}^{-1})$. This can be viewed as a preconditioning procedure. The resulting Hamiltonian dynamics can thus be written as

$$\begin{aligned} dx &= \mathcal{C}p dt \\ dp &= -\mathcal{C}^{-1}x dt - \nabla\Psi(x) dt. \end{aligned}$$

One can observe that when $\nabla\Psi \equiv 0$, the system is a simple harmonic oscillator, whose dynamics can be solved explicitly. This suggests an alternative splitting scheme, namely

$$\begin{aligned} b(x,p) &= b_{+1}(x,p) + b_{-1}(x,p) \\ \text{where } b_{+1}(x,p) &= (\mathcal{C}p, -\mathcal{C}^{-1}x) \\ b_{-1}(x,p) &= (0, -\nabla\Psi(x)). \end{aligned}$$

One can again compute the raw event rate, which is now given by $r_\omega(x,p) = \omega \cdot \langle p, \nabla\Psi(x)\rangle$. A PDMP can be constructed in the same way as before, that is, jumps occur at rate $\lambda(x,p,\omega) = \sigma\left(r_\omega(x,p)\right)$, and at their occurrence, one flips ω. A distinction from the previous case is that one is now alternating between elliptical and linear dynamics, rather than between linear and linear.

2.6 Discussion

This work has explored a general-purpose procedure for constructing PDMPs which admit a given invariant measure. The construction is valid under widely-applicable assumptions and makes the algorithm design pipeline as transparent as possible, giving concrete recommendations for how to design event rates and jump dynamics correctly.

A key element of the construction is the role of *trajectorial reversibility*, a relaxation of the standard detailed balance condition. By making this structural assumption on the process, one can often obtain more rapidly-mixing dynamics, while still retaining a local flavour which allows for simple verifications of the algorithms' correctness. This can be viewed as a natural interpretation of 'lifting', as explored in [CLP99, Vuc16].

Some key challenges related to PDMPs which are particularly worthy of further attention concern optimality and design of improved algorithms. The theoretical work carried out over recent years has allowed for a much clearer understanding of how to construct and study PDMPs for Monte Carlo simulation, and the community is now in a position to ask questions about when a certain PDMP may be *optimal* among a certain class of processes. Developing answers to this question will doubtless guide practical and algorithmic developments.

Moreover, it should be reiterated that many of the most striking advances in designing fast-mixing MCMC algorithms in recent years have been driven by well-chosen variable augmentations and target-informed proposals. It is remarkable that many existing PDMPs have found such success while using fairly generic augmentations and dynamics; it seems likely that with more careful choices, PDMPs could offer great potential benefits. It is hoped that the work presented herein can support and guide such developments.

2.7 Appendix

2.7.1 Proof of Lemmas

2.7.1.1 Proof of Lemma 6

Proof.

$$\begin{aligned}
\mathbf{E}_{\tilde{\mu}}[\langle b(z,\tau), \nabla_z u(z,\tau)\rangle] &= \mathbf{E}_R\left[\mathbf{E}_\mu[\langle b(z,\tau), \nabla_z u(z,\tau)\rangle]\right] \\
&= \mathbf{E}_R\left[\int_{\mathcal{Z}} \langle b(z,\tau), \nabla_z u(z,\tau)\rangle \exp(-H(z))dz\right] \\
&= \mathbf{E}_R\left[\int_{\mathcal{Z}} u(z,\tau)\mathrm{div}_z\left(-b(z,\tau)\exp(-H(z))\right)dz\right] \\
&= \mathbf{E}_R\left[\int_{\mathcal{Z}} u(z,\tau)r(z,\tau)b(z,\tau)\exp(-H(z))dz\right] \\
&= \mathbf{E}_R\left[\mathbf{E}_\mu[r(z,\tau)b(z,\tau)]\right] \\
&= \mathbf{E}_{\tilde{\mu}}[r(z,\tau)u(z,\tau)]
\end{aligned}$$

where the third equality uses the divergence theorem, and the fourth uses the expression for r given by the formula in equation 2.1. □

2.7.1.2 Proof of Lemma 7

Proof.

$$\begin{aligned}
&\mathbf{E}_{\tilde{\mu}}\left[\lambda(z,\tau)\int_{z'\in\mathcal{Z}} Q^\tau(z\to dz')u(z',-\tau)\right] \\
&= \mathbf{E}_R\left[\mathbf{E}_\mu[\lambda(z,\tau)\int_{z'\in\mathcal{Z}} Q^\tau(z\to dz')u(z',-\tau)]\right] \\
&= \mathbf{E}_R\left[\int_{z\in\mathcal{Z}}\mu(dz)\lambda(z,\tau)\int_{z'\in\mathcal{Z}} Q^\tau(z\to dz')u(z',-\tau)\right] \\
&= \mathbf{E}_R\left[\int_{z\in\mathcal{Z}}\int_{z'\in\mathcal{Z}}\mu(dz)\lambda(z,\tau)Q^\tau(z\to dz')u(z',-\tau)\right] \\
&= \mathbf{E}_R\left[\int_{z\in\mathcal{Z}}\int_{z'\in\mathcal{Z}}\mu(dz')\lambda(z',\tau)Q^\tau(z'\to dz)u(z',-\tau)\right] \\
&= \mathbf{E}_R\left[\int_{z'\in\mathcal{Z}}\mu(dz')\lambda(z',\tau)u(z',-\tau)\right] \\
&= \mathbf{E}_R\left[\int_{z'\in\mathcal{Z}}\mu(dz')\lambda(z',-\tau)u(z',\tau)\right] \\
&= \mathbf{E}_{\tilde{\mu}}\left[\lambda(z',-\tau)u(z',\tau)\right].
\end{aligned}$$

where the fourth equality uses the J^τ-reversibility of Q^τ, the fifth uses that $Q^\tau(z'\to dz)$

is a probability measure for each z', and the sixth uses the symmetry of R. □

2.7.1.3 Proof of Lemma 8

Lemma 11. *For all $u \in C_c^1(\mathcal{Z})$, for all $j = 1, \cdots, M$,*

$$\mathbf{E}_{\tilde{\mu}}\left[\lambda_j(z,\tau)\int_{z'\in\mathcal{Z}} Q_j^\tau(z \to dz')u(z', \mathcal{F}_j\tau)\right] = \mathbf{E}_{\tilde{\mu}}[\lambda_j(z, \mathcal{F}_j\tau)u(z,\tau)]$$

Proof.

$$\begin{aligned}
&\mathbf{E}_{\tilde{\mu}}\left[\lambda_j(z,\tau)\int_{z'\in\mathcal{Z}} Q_j^\tau(z \to dz')u(z', \mathcal{F}_j\tau)\right] \\
&= \mathbf{E}_{R^D}\left[\mathbf{E}_\mu[\lambda_j(z,\tau)\int_{z'\in\mathcal{Z}} Q_j^\tau(z \to dz')u(z', \mathcal{F}_j\tau)]\right] \\
&= \mathbf{E}_{R^D}\left[\int_{z\in\mathcal{Z}} \mu(dz)\lambda_j(z,\tau)\int_{z'\in\mathcal{Z}} Q_j^\tau(z \to dz')u(z', \mathcal{F}_j\tau)\right] \\
&= \mathbf{E}_{R^D}\left[\int_{z\in\mathcal{Z}}\int_{z'\in\mathcal{Z}} \mu(dz)\lambda_j(z,\tau)Q_j^\tau(z \to dz')u(z', \mathcal{F}_j\tau)\right] \\
&= \mathbf{E}_{R^D}\left[\int_{z\in\mathcal{Z}}\int_{z'\in\mathcal{Z}} \mu(dz')\lambda_j(z',\tau)Q_j^\tau(z' \to dz)u(z', \mathcal{F}_j\tau)\right] \\
&= \mathbf{E}_{R^D}\left[\int_{z'\in\mathcal{Z}} \mu(dz')\lambda_j(z',\tau)u(z', \mathcal{F}_j\tau)\right] \\
&= \mathbf{E}_{R^D}\left[\int_{z'\in\mathcal{Z}} \mu(dz')\lambda_j(z', \mathcal{F}_j\tau)u(z',\tau)\right] \\
&= \mathbf{E}_{\tilde{\mu}}\left[\lambda_j(z', \mathcal{F}_j\tau)u(z',\tau)\right].
\end{aligned}$$

As in the previous lemma, the fourth equality uses the J_j^τ-reversibility of Q_j^τ, the fifth uses that $Q_j^\tau(z' \to dz)$ is a probability measure for each z', and the sixth uses the symmetry of R^M under any flipping operator $\mathcal{F}_j$. □

2.7.2 Conditions from Löpker and Palmowski

Here, the relevant definitions from [LP13] are reproduced, as **Assumptions 2 & 3** in Section 4 are small modifications of their setup. We translate their conditions into the terminology of this paper, and adapt them to the TE-PDMP setting in a natural way. Throughout, work with a TE-PDMP with driving vector field b, event rate λ, and transition dynamics Q.

Definition 2.7.1. *(LP Condition A) Let $z(t)$ be the solution to $\dot{z} = b(z)$, and denote the map from $z(0) = z$ to $z(t) = z'$ by $z' = \varphi(z,t)$. Let N_t be the number of events which occur in the time interval $[0,t]$. We assume that:*

1. *For all $z \in \mathcal{Z}, \tau \in \{\pm 1\}$, there is an $\epsilon(z,\tau) > 0$ such that $\int_0^{\epsilon(z,\tau)} \lambda(\varphi(z,t),\tau)dt < \infty$.*

2. *For all* $z \in \mathcal{Z}, \tau \in \{\pm 1\}$, $\int_0^\infty \lambda(\varphi(z,t),\tau)dt = \infty$.

3. *For all* $z \in \mathcal{Z}, \tau \in \{\pm 1\}, t > 0$, *we have that* $\mathbf{E}[N_t | Z_0 = z, \tau_0 = \tau] < \infty$.

4. *For all* $z \in \mathcal{Z}, \tau \in \{\pm 1\}$, *we have that* $Q((z,\tau),(z,\tau)) = 0$.

These are all implicit in **Assumption 2**; (1) is strengthened to λ having a finite integral for all times, (2) and (3) are reproduced exactly, and (4) is not needed. This is because in the case of TE-PDMPs, the τ variable always jumps at an event. As such, there are no such 'phantom jumps' where a jump nominally occurs, but the chain does not move.

Definition 2.7.2. *(LP Condition B) Let* $z(t)$ *be the solution to* $\dot{z} = b(z)$, *and denote the map from* $z(0) = z$ *to* $z(t) = z'$ *by* $z' = \varphi(z,t)$. *Let* $\mathcal{B}(\mathcal{Z})$ *denote the Borel sets in* $\mathcal{Z}$. *We assume that:*

1. *The process* $(Z_t, \tau_t)_{t \geqslant 0}$ *has a stationary measure* $\tilde{\mu}$ *on* $\mathcal{Z} \times \{\pm 1\}$.

2. *The event rate* λ *satisfies* $\mathbf{E}_{\tilde{\mu}}[\lambda(z,\tau)] < \infty$.

3. *For each* $\tau \in \{\pm 1\}$, *the map* $z \mapsto \lambda(z,\tau)$ *is continuous.*

4. *For all* $A \in \mathcal{B}(\mathcal{Z})$, $z \in \mathcal{Z}$, *and* $\tau \in \{\pm 1\}$, *we have* $Q^\tau(\varphi(z,t),A) \to Q^\tau(z,A)$ *as* $t \to 0$.

5. *Let* $\partial\mathcal{Z}$ *denote the boundary of the space* $\mathcal{Z}$, *and define* $\partial_h\mathcal{Z} = \{z \in \mathcal{Z} : dist\,(z,\partial\mathcal{Z}) \leqslant h\}$. *We then assume that* $Q((z,\tau) \to \partial_h\mathcal{Z} \times \{\pm 1\}) \to 0$ *uniformly over* $z \in \mathcal{Z}$, *and* $\tau \in \{\pm 1\}$.

(1), (2), and (3) are included as part of **Assumption 2**, and Condition (4) is **Assumption 3**. (5) can be dropped, as by **Assumption 1**, the target measure μ has full measure on $\mathbf{R}^d$, and thus the boundary $\partial\mathcal{Z}$ does not exist.

Definition 2.7.3. *(LP Condition C) Let* Π_Q *denote the stationary measure of the PDMP immediately after a jump. Define* T_1 *as the time it takes for the first jump to occur. We then assume that* $\mathbf{E}_{\Pi_Q}[T_1] < \infty$.

The authors introduce this condition to ensure that the stationary measure of their process is absolutely continuous; in this work, this condition is assumed directly in **Assumption 1**, and is thus omitted.

Definition 2.7.4. *(LP Condition D) Let* $z(t)$ *be the solution to* $\dot{z} = b(z)$, *and denote the map from* $z(0) = z$ *to* $z(t) = z'$ *by* $z' = \varphi(z,t)$. *Let* Π_Q *denote the stationary measure of the PDMP immediately after a jump. Let* $\tilde{\mu}$ *denote the stationary measure of the PDMP. We assume that:*

1. Π_Q *is absolutely continuous with respect to* $\tilde{\mu}$, *with Radon-Nikodym derivative* $\beta(z,\tau) = \frac{d\Pi_Q}{d\tilde{\mu}}(z,\tau)$.

2. *For all* $z \in \mathcal{Z}, \tau \in \{\pm 1\}$*, there is an* $\epsilon(z,\tau) > 0$ *such that* $\int_0^{\epsilon(x)} \beta(\varphi(z,-t),\tau)dt < \infty$.

These conditions can be deduced from our other assumptions; for our processes, we can directly calculate both Π_Q and $\tilde{\mu}$, and thus verify (1) with an explicit formula for β. (2) can be deduced by comparing the form of this β to the original λ and then applying our strengthened variant of LP's condition A1.

2.7.3 Definition from Durmus, Guillin, Monmarché

Here, the relevant definitions from [DGM18b] are reproduced, in order to give a precise and maximally self-contained statement of their Definition 20, that is, their condition for a PDMP semigroup to be 'smoothly and compactly approximable'.

In this work, the authors say that a PDMP semi-group has 'characteristics (φ, λ, Q)' where, in this work, we would say that the process is driven by the vector field b which has flow maps given by φ, has event rate λ, and jump dynamics Q. In this section, we adopt their terminology, where the PDMP in question is named $(X_t)_{t\geqslant 0}$ and lives on the manifold $\mathcal{M}$.

Definition 2.7.5. *(DGM Definition 16) We say that a differential flow* φ *on* $\mathcal{M}$ *and a Markov kernel* Q *are compactly compatible if for all compact* $K \subset \mathcal{M}$*,* $T \geqslant 0$*, there exists a compact set* $\tilde{K} \subset \mathcal{M}$ *satisfying: for all* $n \in \mathbf{N}$*,* $\{t_i\}_{i=1}^n$*,* $\sum_{i=1}^n t_i \leqslant T$*, there exists a sequence of compact sets* $\{K_i\}_{i=1}^n \subset \mathcal{M}$ *such that, setting* $K_0 = K$*:*

1. *For all* $1 \leqslant i \leqslant n$*,* K_i *depends only on* $\{t_j\}_{j=1}^i$
2. $\cup_{i=0}^n K_i \subset \tilde{K}$
3. *For all* $0 \leqslant i \leqslant n-1$*,* $0 \leqslant s_{i+1} \leqslant t_{i+1}$*,* $0 \leqslant s_{n+1} \leqslant T - \sum_{j=1}^n T_j$*, it holds that*

$$\begin{aligned} \cup_{x\in K_i} \textit{supp}\ Q(\varphi_{t_{i+1}}(x),\cdot) &\subset K_{i+1} \\ \varphi_{s_{i+1}}(K_i) &\subset \tilde{K} \\ \varphi_{s_{n+1}}(K_n) &\subset \tilde{K} \end{aligned}$$

Definition 2.7.6. *(DGM Assumption 2) Let* $(P_t)_{t\geqslant 0}$ *be a non-explosive PDMP semi-group with characteristics* (φ, λ, Q)*. Assume that for all* $T \geqslant 0$*, there exists* $M \geqslant 0$ *such that for all* $x \in \mathcal{M}$ *and* $0 \leqslant t \leqslant T$*, supp* $P_t(x,\cdot) \subset B(x,M)$*.*

Definition 2.7.7. *(DGM Assumption 3) The characteristics* (φ, λ, Q) *satisfy*

1. *the flow* φ *and the Markov kernel* Q *are compactly compatible;*
2. $\lambda \in C^1(\mathcal{M})$*, and for all* $f \in C^1(\mathcal{M})$*,* $\lambda Q f \in C^1(\mathcal{M})$*, and there exists a locally bounded function* $\Psi : \mathcal{M} \to \mathbf{R}_+$ *such that for all* $x \in K$*,*

$$\|\nabla(\lambda Q f)(x)\| \leqslant \|\Psi\|_{\infty,K} \cdot \sup\{|f(y)| + \|\nabla f(y)\| : y \in \textit{supp}\ \{Q(x,\cdot)\}\}$$

where

$$\|\Psi\|_{\infty,K} = \sup_{x\in K} \|\Psi(x)\|$$
$$(Qf)(x) = \mathbf{E}_{y\sim Q(x\to dy)}[f(y)]$$
$$(\lambda Qf)(x) = \lambda(x)\cdot(Qf)(x)$$

3. *The mapping $(t,x)\mapsto\varphi_t(x)$ is in $C^1(\mathbf{R}_+\times\mathcal{M})$, and for all compact $K\subset\mathcal{M}$ and $t\geqslant 0$, it holds that*

$$\sup\{\|\nabla\varphi_s(x)\| : 0\leqslant s\leqslant t, x\in K\} < \infty$$

Definition 2.7.8. *(DGM Definition 20) We say that the PDMP semigroup $(P_t)_{t\geqslant 0}$ with characteristics (φ,λ,Q) is smoothly and compactly approximable if for all $\epsilon>0$, there exist characteristics $(\varphi,\lambda^\epsilon,Q^\epsilon)$ satisfying (DGM A2, DGM A3) and*

$$\sup_{x\in\mathcal{M},A\in\mathcal{B}(\mathcal{M})} \{(\lambda^\epsilon(x)\wedge\lambda(x))\cdot|Q^\epsilon(x,A)-Q(x,A)|+|\lambda^\epsilon(x)-\lambda(x)|\}\leqslant\epsilon.$$

2.7.3.1 Proof of Correctness for the Leapfrog PDMP

The generator for the Leapfrog PDMP can be written as

$$\mathcal{L}u(x,p,\omega) = \langle b_\omega(x,p),\nabla_{x,p}u(x,p,\omega)\rangle + \sigma\left(r_\omega(x,p)\right)\cdot(u(x,p,-\omega)-u(x,p,\omega))$$
$$\text{where}\quad r_\omega(x,p)=\omega\cdot\langle p,\nabla U(x)\rangle.$$

We will show that for all $u\in C_c^1(\mathcal{Z})$, it holds that $\mathbf{E}_{\tilde\mu}\left[\mathcal{L}u(x,p,\omega)\right]=0$, where $\tilde\mu(dx,dp,d\omega) = \exp(-U(x))dx\cdot\mathcal{N}(p|0,I)\cdot R(d\omega)$, where R is again the Rademacher distribution on $\{\pm1\}$.

Begin by computing that

$$\begin{aligned}
\mathbf{E}_\mu\left[\langle b_1(x,p),\nabla_{x,p}u(x,p,\omega)\rangle\right] &= \int\int\mu(x,p)\langle p,\nabla_x u(x,p,\omega)\rangle dxdp\\
&= \int\mathcal{N}(p|0,I)\int\exp\left(-U(x)\right)\langle p,\nabla_x u(x,p,\omega)\rangle dxdp\\
&= \int\mathcal{N}(p|0,I)\int\exp\left(-U(x)\right)\langle p,\nabla_x U(x)\rangle\cdot u(x,p,\omega)dxdp,\\
&= \mathbf{E}_\mu\left[\langle p,\nabla U(x)\rangle\cdot u(x,p,\omega)\right]
\end{aligned}$$

using integration by parts. Similarly, compute that

$$\begin{aligned}\mathbf{E}_\mu\left[\langle b_{-1}(x,p), \nabla_{x,p} u(x,p,\omega)\rangle\right] &= \int\int \mu(x,p)\langle -\nabla_x U(x), \nabla_p u(x,p,\omega)\rangle dxdp \\ &= \int \exp\left(-U(x)\right) \int \mathcal{N}(p|0,I)\langle -\nabla_x U(x), \nabla_p u(x,p,\omega)\rangle dpdx \\ &= \int \exp\left(-U(x)\right) \int \mathcal{N}(p|0,I)\langle -\nabla_x U(x), p\rangle \cdot u(x,p,\omega) dpdx \\ &= \mathbf{E}_\mu\left[\langle -\nabla_x U(x), p\rangle \cdot u(x,p,\omega)\right].\end{aligned}$$

Deduce thus that

$$\mathbf{E}_\mu\left[\langle b_\omega(x,p), \nabla_{x,p} u(x,p,\omega)\rangle\right] = \omega \cdot \mathbf{E}_\mu\left[\langle p, \nabla U(x)\rangle \cdot u(x,p,\omega)\right] = \mathbf{E}_\mu\left[r_\omega(x,p) \cdot u(x,p,\omega)\right]$$

Now, consider

$$\begin{aligned}&\mathbf{E}_{\tilde{\mu}}\left[\sigma\left(r_\omega(x,p)\right) \cdot \left(u(x,p,-\omega) - u(x,p,\omega)\right)\right] \\ =&\mathbf{E}_{\tilde{\mu}}\left[\sigma\left(r_\omega(x,p)\right) \cdot u(x,p,-\omega)\right] - \mathbf{E}_{\tilde{\mu}}\left[\sigma\left(r_\omega(x,p)\right) \cdot u(x,p,\omega)\right] \\ =&\mathbf{E}_{\tilde{\mu}}\left[\sigma\left(r_{-\omega}(x,p)\right) \cdot u(x,p,\omega)\right] - \mathbf{E}_{\tilde{\mu}}\left[\sigma\left(r_\omega(x,p)\right) \cdot u(x,p,\omega)\right] \\ =&\mathbf{E}_{\tilde{\mu}}\left[\sigma\left(r_{-\omega}(x,p)\right) - \sigma\left(r_\omega(x,p)\right) \cdot u(x,p,\omega)\right],\end{aligned}$$

using only that R is invariant under the flipping of ω. Now, compute explicitly that

$$\sigma\left(r_{-\omega}(x,p)\right) - \sigma\left(r_\omega(x,p)\right) = -r_\omega(x,p).$$

Finally, collect these observations to see that

$$\begin{aligned}&\mathbf{E}_{\tilde{\mu}}\left[\mathcal{L}u(x,p,\omega)\right] \\ =&\mathbf{E}_{\tilde{\mu}}\left[\langle b_\omega(x,p), \nabla_{x,p} u(x,p,\omega)\rangle\right] + \mathbf{E}_{\tilde{\mu}}\left[\sigma\left(r_\omega(x,p)\right) \cdot \left(u(x,p,-\omega) - u(x,p,\omega)\right)\right] \\ =&\mathbf{E}_{\tilde{\mu}}\left[r_\omega(x,p) \cdot u(x,p,\omega)\right] + \mathbf{E}_{\tilde{\mu}}\left[-r_\omega(x,p) \cdot u(x,p,\omega)\right] \\ =&0,\end{aligned}$$

and thus deduce the required result. Apply the results from [DGM18b] to extend this to all u in the domain of $\mathcal{L}$, and deduce that $\tilde{\mu}$ is an invariant measure for the process.

Chapter 3

Accelerated Sampling on Discrete Spaces with Non-Reversible Markov Processes

Abstract

We consider the task of MCMC sampling from a distribution defined on a discrete space. Building on recent insights provided in [Zan19], we devise a class of efficient continuous-time, non-reversible algorithms which make active use of the structure of the underlying space. Particular emphasis is placed on how symmetries and other group-theoretic notions can be used to improve exploration of the space. We test our algorithms on a range of examples from statistics, computational physics, machine learning, and cryptography, which show improvement on alternative algorithms. We provide practical recommendations on how to design and implement these algorithms, and close with remarks on the outlook for both discrete sampling and continuous-time Monte Carlo more broadly.

3.1 Introduction

In this work, we concern ourselves with Markov Chain Monte Carlo (MCMC) simulation of distributions defined on discrete state spaces.

On continuous spaces, there has been a great deal of successful work on how to construct efficient MCMC proposals which work in great generality. Much of this success has stemmed from identifying continuous-time dynamical processes (ODEs, SDEs, PDMPs)

which admit the desired invariant measure, and then discretising those processes to form tractable discrete-time chains.

This approach has apparently seen less use in the discrete setting. A reasonable justification for this is that differential equations do not exist per se on discrete spaces, and so extending this approach to derive appropriate dynamics for discrete spaces seems to pose some challenges at first. However, one can note the following: it is essentially the case that, on discrete spaces, the various notions of Markov process on continuous spaces (ODEs, SDEs, PDMPs, etc.) all collapse to the same notion, that of a *Markov Jump Process* (MJP).

From the perspective of algorithm design, this presents a great simplification: any continuous time sampler on a discrete space must arise as an MJP. The first contribution of this work is to explicitly construct a family of MJPs which admit a desired invariant measure. Moreover, in contrast to the usual scenario on continuous state spaces, it is possibly to dispense entirely with discretisation and to simulate the process exactly in continuous time. In addition to motivating this general-purpose sampling algorithm for discrete spaces, which we term* the *Locally-Balanced Jump Process* (LBJP), this construction will allow us to retroactively justify the success of certain existing approaches to discrete sampling.

A second aspect of discrete sampling which is worthy of attention is how one can improve the computational efficiency of a sampler by making use of the symmetric and algebraic structure available in a state space. In particular, it is often the case that a state space $\mathcal{X}$ is naturally acted upon by a *group* of symmetries G. Given such structures, which abound in applications, the natural question is how to best exploit them algorithmically. The second contribution of this work is to present a trio of algorithms which offer a generic solution to this question. The first, which we term the *Tabu Sampler* is adapted to settings where G is generated by low-order elements, and encourages the process to avoid re-using generators across short-to-medium timescales, thus preventing backtracking behaviour.

In contrast, the latter two algorithms focus on the scenario where the group is generated by elements of high or infinite order. They are thus constructed to instead encourage persistent motion across the state space, re-using generators so long as they are driving the dynamics in productive directions. The two schemes presented are the *discrete Coordinate Sampler* (dCS) and *discrete Zig-Zag Process* (dZZ), each named by analogy with corresponding algorithms in the literature on Piecewise-Deterministic Markov Processes (PDMPs), namely the works of [BFR19, WR20].

All three of the algorithms are devised to be non-reversible, and as been witnessed in a range of applications, this appears to have a beneficial effect upon convergence behaviour.

*An earlier version of this worked named this process the 'Zanella process', due to its genesis in the work of [Zan19]. While we still wish to celebrate the author's contribution, we have since opted to change the name to be more descriptive.

We demonstrate that all of the schemes above are asymptotically exact in the standard MCMC sense, in each case by establishing either reversibility or skew-reversibility. We explain how to implement these schemes in practice and provide a number of numerical experiments, as well as code. We also offer concrete recommendations on how and when each of these samplers should be applied in practice. We close with some discussion of potential future work in this area.

3.2 Continuous-time Algorithms on Discrete Spaces

3.2.1 Motivation: Locally-Balanced MCMC in Continuous Time

In this section, we construct the *Locally-Balanced Jump Process* (LBJP) algorithm for sampling from a distribution π supported on a discrete space $\mathcal{X}$. We begin by introducing the notion of a *Markov Jump Process* (MJP).

Definition 3.2.1. *A Markov Jump Process is a continuous-time Markov process taking values in a countable state space $\mathcal{X}$. Such a process is characterised by its jump rates, $\lambda(x \to y)$, defined such that, as $h \to 0^+$*

$$\begin{aligned}\mathbf{P}(X_{t+h} = y | X_t = x) &= h \cdot \lambda(x \to y) + o(h) \quad \text{for } y \neq x \\ \mathbf{P}(X_{t+h} = x | X_t = x) &= 1 - h \cdot \Lambda(x) + o(h)\end{aligned}$$

where $\Lambda(x) = \sum_y \lambda(x \to y)$. The generator of such a Markov process is given by

$$\mathcal{L}f(x) = \sum_y \lambda(x \to y) \left[f(y) - f(x)\right].$$

We will also require the notion of *reversibility* as it pertains to MJPs.

Definition 3.2.2. *A Markov Jump Process with generator $\mathcal{L}$ is reversible with respect to the measure π if, for all $f, g \in L^2(\pi)$, it holds that*

$$\mathbf{E}_\pi \left[(\mathcal{L}f)(x) g(x)\right] = \mathbf{E}_\pi \left[f(x)(\mathcal{L}g)(x)\right].$$

By considering $f(x) = \mathbf{I}[x = a], g(x) = \mathbf{I}[x = b]$, one can show that this is equivalent to

$$\pi(x)\lambda(x \to y) = \pi(y)\lambda(y \to x) \quad \text{for all } x, y \in \mathcal{X}$$

The utility of reversibility is that it is a sufficient condition for the MJP to admit π as an invariant measure, and moreover, it is a local condition, and is hence easily verifiable. As such, a standard approach to constructing Markov processes with a given invariant

measure is to explicitly construct a process which is reversible with respect to that measure. We now proceed along these lines.

In order to make our search for such a process more tractable, we make two simplifications. Firstly, we assume that our discrete space $\mathcal{X}$ admits the structure of a *graph*, that is, there is some set of *edges* $\mathcal{E} \subset \mathcal{X} \times \mathcal{X}$ which encodes the notion of locality in the space. We assume throughout that the graph is undirected, i.e. if $(x, y) \in \mathcal{E}$, then $(y, x) \in \mathcal{E}$ also. If $(x, y) \in \mathcal{E}$, we will call x and y *neighbours*, and interpret these points as being 'close', in some suitable sense. We use the shorthand $\partial x = \{y \in \mathcal{X} : (x, y) \in \mathcal{E}\}$ to denote the *neighbourhood* of x, i.e. the set of all points in $\mathcal{X}$ which are adjacent to x; we assume throughout that the graph is *locally finite*, i.e. that each vertex has finitely-many neighbours..

Secondly, we will restrict ourselves to considering Markov Jump Processes (MJPs) $(X_t)_{t \geqslant 0}$ on $\mathcal{X}$ which satisfy the following desiderata:

1. The process can only jump from x to y if $y \in \partial x$.
2. For $x \in \mathcal{X}, y \in \partial x$, the jump rate from x to y, $\lambda(x \to y)$, is purely a function of $\frac{\pi(y)}{\pi(x)}$.
3. X_t is in detailed balance with respect to π.

The first of these ensures that X_t respects the graphical structure of the model. The second is a simplifying assumption which streamlines the analysis, and can be weakened[†]. The third is sufficient to ensure that X_t be ergodic with respect to π (under mild additional assumptions), and is crucial for the validity of the procedure as a Monte Carlo method.

As established earlier, the third condition can be written as

$$\forall (x, y) \in E, \quad \pi(x)\lambda(x \to y) = \pi(y)\lambda(y \to x).$$

Applying the second condition, we write $\lambda(x \to y) = g\left(\frac{\pi(y)}{\pi(x)}\right)$ for some function g, and thus see that

$$\begin{aligned} \pi(x) g\left(\frac{\pi(y)}{\pi(x)}\right) &= \pi(y) \cdot g\left(\frac{\pi(x)}{\pi(y)}\right) \\ g\left(\frac{\pi(y)}{\pi(x)}\right) &= \frac{\pi(y)}{\pi(x)} \cdot g\left(\frac{\pi(x)}{\pi(y)}\right) \\ g(t) &= t \cdot g(1/t) \quad \text{for } t = \frac{\pi(y)}{\pi(x)}. \end{aligned}$$

As such, we deduce that for an algorithm of this form to satisfy all of our desiderata, irrespective of the target distribution π, it is necessary and sufficient that $\lambda(x \to y) = g\left(\frac{\pi(y)}{\pi(x)}\right)$ for some function g satisfying $g(t) = t \cdot g(1/t)$. We refer to such g as *balancing*

[†]For greatest generality, one could take $\lambda(x \to y)$ to be the product of a function of $\frac{\pi(y)}{\pi(x)}$ with a symmetric function $S(x, y)$.

functions. As such, for the remainder of this section, we fix a balancing function and study the resulting process, which we term the *Locally Balanced Jump Process*.

Algorithm 30 Locally-Balanced Jump Process for sampling from $\pi(x), x \in \mathcal{X}$

1. At $x \in \mathcal{X}$,
   (a) For $y \in \partial x$, compute $\lambda(x \to y) = g\left(\frac{\pi(y)}{\pi(x)}\right)$.
   (b) Compute $\Lambda(x) = \sum_{y \in \partial x} \lambda(x \to y)$.
   (c) Sample a waiting time $T \sim \text{Exponential}(\text{rate } = \Lambda(x))$.
   (d) Sample a new location $y \in \partial x$ with probability $\frac{\lambda(x \to y)}{\Lambda(x)}$.
   (e) Advance time by T.
   (f) Jump to y.

Informally, the process can be seen as a random walk which observes its neighbours, weighs up which of the neighbours is most preferable as a next location, and then chooses to jump there after a random amount of time. Note that by operating directly in continuous time, the algorithm is 'rejection-free'.

A reasonable analogy in continuous state spaces is the (overdamped) Langevin diffusion process, in the sense that both processes are 'weakly greedy'; they naturally gravitate towards regions of higher probability, while retaining the ability to explore regions of lower probability from time to time. This analogy is developed further in the work [LZ19].

The optimal choice of g is not necessarily clear a priori. Empirically, some sensible choices include $g(t) \in \left\{\sqrt{t}, \min(1, t), \frac{t}{1+t}\right\}$. We note that in [Zan19], there seems not to be a universally-optimal choice of g; it appears to be genuinely task-dependent. We are unsure of whether the same reasoning holds for the continuous-time process. Moreover, as is often the case with continuous-time processes, if one wants to compare different choices of g, it is necessary choose a normalisation of some sort. This is because for any balancing function g, one could equally take $2g$ as the balancing function, and obtain a process which converges twice as fast — by the clock of the process, but certainly not in real life! Upon selecting a normalisation (e.g. $g(1) = 1$), it may be possible to derive an optimality result, in the spirit of [Pes73].

The chief cost of this algorithm is the repeated computation of expressions of the form $\frac{\pi(y)}{\pi(x)}$. Many target distributions of interest admit convenient factorisation structures, and in these settings, computing $\frac{\pi(y)}{\pi(x)}$ can be considerably cheaper than computing either of $\pi(x), \pi(y)$ (even up to a normalising constant). As such, when wall-clock time is a concern, it is imperative to carry out these computations judiciously. In situations where one does not have such simplifications, or where the chosen graphical structure is dense (i.e. each state has many direct neighbours), then the cost can grow somewhat. We expect that it is still generally worthwhile to work with the LBJP (as opposed to Random Walk-based algorithms), but it is difficult to say anything concrete at this level of generality. We

present the LBJP as a worthy baseline for general discrete sampling tasks, relative to random-walk based samplers. We have found it to be simple, transparent, and reliable on unimodal tasks, in particular for Bayesian sampling with highly-informative posterior distributions.

In terms of related work, the LBJP bears a strong connection to *Kinetic Monte Carlo* algorithms (also 'KMC', see e.g. [ELVE19]) which have long been used in the computational physics community, but which may not be well-known to statisticians. An early example of this is the 'N-Fold Way' of [BKL75], which can be viewed as a discrete-time analog of the LBJP. The 'Waiting Time Method' (WTM) of [DS01], was derived by embedding the N-Fold Way into continuous time, and is almost directly identical to the LBJP. We learned of these references through the recent work of [Bal17], which essentially puts the WTM approach back into discrete time, but with some useful algorithmic simplifications for target measures of a given structure. Our use of balancing functions was motivated by the work of [Zan19], which also prompted our interest in structured discrete sampling more broadly.

3.2.2 Simulation on Spaces with Algebraic Structure

In the design of the LBJP above, the implicit assumption of a graphical structure of the space $\mathcal{X}$ is made. It is often clear what the graphical structure is, but there is no guarantee that it facilitates sampling some arbitrary distribution π on $\mathcal{X}$. Often the form of π implies that a subset of edges are much more natural to travel along than others, and this distinction is often not as clear when working at the abstract level of a graph. In this section we will observe that in a number of applications, the state space $\mathcal{X}$ is, in addition, also naturally furnished with an algebraic structure. That is, there exist a group G which acts on $\mathcal{X}$. Informally, this means that there is a set G of 'actions' g such that for any $x \in \mathcal{X}$, we can make sense of what it means to perform the action g upon the state x and obtain a new state $y = `g \star x'$, where we by the latter mean applying the action g to x. To make this more rigorous, we first recall the definition of a group.

Definition 3.2.3. *A group is a set G, equipped with a binary operation $\bullet : G \times G \to G$, satisfying the four group axioms:*

1. *For all $g, h \in G$, $g \bullet h \in G$. (Closure)*
2. *For all $g, h, k \in G$, $(g \bullet h) \bullet k = g \bullet (h \bullet k)$. (Associativity)*
3. *There is an element, the identity element, $id \in G$ such that for all $g \in G$, $g \bullet id = id \bullet g = g$.*
4. *For all $g \in G$, there is an element $h = g^{-1} \in G$ such that $g \bullet h = h \bullet g = id$.*

We denote a group by $(G, \bullet)$.

In what follows, we will generally write the composition of group elements as $g \cdot h$ rather than $g \bullet h$ to reduce clutter, in practice overloading the multiplication operator.

To motivate our approach, we will consider a simple but relevant case which illustrates how a space can be associated with a group. Begin by considering a set $\mathcal{S}$ of n elements, i.e., a set isomorphic to $[n] \equiv \{1, 2, \cdots, n-1, n\}$ and consider the set of all one-to-one functions from $\mathcal{S}$ to itself. If we equip this set with the operation of function composition, this is known as the symmetric group $\Sigma_n \equiv (\Sigma_n, \circ)$. The elements in Σ_n corresponds to all permutations of the objects in $\mathcal{S}$, so the symmetry group gives us a natural way to travel everywhere between points of $\mathcal{S}$ by selecting the components of Σ_n appropriately.

Nonetheless, from an algorithmic perspective, the symmetric group is often vast and unmanageable, as the order (number of elements) of Σ_n is $n!$. It is therefore much more useful to identify, if possible, a subset consisting of the 'fundamental' elements, i.e. the building blocks of the group, allowing us to decompose any permutation into a series of simple steps. This is akin to the notion of a basis for a vector space, and such a collection is known as a *generating set*:

Definition 3.2.4. *Let $(G, \cdot)$ be a group, and let $\Gamma \subset G$ be a subset. Define $\langle \Gamma \rangle$ to be the smallest subgroup of G which contains Γ. If $\langle \Gamma \rangle = G$, then we say that Γ is a generating set for G and call the elements of Γ generators. A generating set Γ is furthermore* SYMMETRIC *if $\gamma \in \Gamma \implies \gamma^{-1} \in \Gamma$.*

Thus with a generator set Γ, it is possible to decompose any element $g \in G$ to a series of simple group operations. As such, we might consider designing a sampling procedure which moves around the state space by applying elements of Γ to the current state, without ever needing to consider the entirety of G all at once. The concept of a *group action* makes this heuristic formal:

Definition 3.2.5. *Given a set $\mathcal{X}$ and a group $(G, \cdot)$, a group action is defined as an operation $\star$ mapping $G \times \mathcal{X}$ to $\mathcal{X}$ satisfying:*

1. $\forall x \in \mathcal{X}, \quad id \star x = x.$
2. $\forall g, h \in G, \quad g \star (h \star x) = (g \cdot h) \star x.$

From the definition, we can see that group actions offer a convenient language with which to formulate motion between the states of a space $\mathcal{X}$. The samplers we introduce below will in every case carry out exploration on the target space via repeated applications of group actions to $\mathcal{X}$. Since objects in Γ generate G, we can furthermore restrict ourselves to only consider group actions of the generators. However, the form of the generating set can often be determined by the specific application we consider, rather than being determined automatically by the state space in isolation. In order to make these notions somewhat more concrete, we first provide two examples equivalent to the space $\mathcal{X} = [n]$

we considered above, but where the target distributions of interest induce significantly different generator sets:

Example 8 (Matching/Linkage (e.g. [BWBS19, Zan19]))**.** *Suppose we have two sets of covariates,* $\{Y_i\}_{i=1}^N$ *and* $\{Z_j\}_{j=1}^N$*, which correspond to the same individuals, but in an unknown order. A natural task is to align these two sets, i.e. to identify a bijective mapping* $\sigma : [n] \to [n]$ *such that* Y_i *and* $Z_{\sigma(i)}$ *correspond to the same individual. This set of maps is just the symmetric group on* n *elements* Σ_n*, introduced above. As* $\mathcal{X}$ *is already a group in this case, the natural choice is to take* G *to be the same group, i.e. to set* $G = \Sigma_n$*. Since there is no ordering structure to the covariates, a reasonable choice of generating set* Γ *is the set of all transpositions, i.e. mappings* $\tau_{i,j}$ *which results in group actions given by*

$$y = \tau_{i,j} \star x \iff \begin{cases} y_j = x_i \\ y_i = x_j \\ y_k = x_k \quad \text{for } k \neq i, j. \end{cases}$$

In words, applying the action $\tau_{i,j}$ *to* x *would thus correspond to swapping our beliefs about which* Z *covariates correspond to* Y_i *and* Y_j *respectively. The cardinality of* $|\Gamma|$ *in this case is* $n(n-1)/2$*.*

Example 9 (Ranking)**.** *Suppose now that we have* n *individuals, and we want to rank them in some way. As in the matching case, the underlying state space is* $\mathcal{X} = \Sigma_n$*, and it is natural to take* $G = \Sigma_n$ *as well. However, because the ordering of* $[n]$ *is now meaningful for our problem, it is more natural to use the set of all* ADJACENT *transpositions as a generating set for our task, i.e. to use* $\{\tau_{i,j}\}_{|i-j|=1}$*. Applying the mapping* $\tau_{i,i+1}$ *then corresponds to swapping our beliefs about the relative rankings of the individuals who were previously ranked in* i^{th} *and* $(i+1)^{th}$ *position. This generator set is just of cardinality* $(n-1)$*.*

The above examples illustrates how different algebraic structures can be exploited to serve our goal of sampling a particular distribution, and also be used to make the set of possible directions much more manageable in size. We now consider other spaces where the group is significantly different from the symmetric group.

Example 10 (Binary Spin Systems)**.** *In statistical physics, one commonly studies binary spin systems on graphs. Here, there is an underlying graph* $G_0 = (V, E)$*, and for each vertex* i*, there is a binary 'spin',* $\sigma_i \in \{\pm 1\}$*. The state space of interest is then given by* $\mathcal{X} = \{\pm 1\}^{|V|}$*. The simplest group action one can define for such a system is generated by picking a vertex* i*, and flipping the spin at that vertex, i.e. setting* $\sigma_i' = -\sigma_i$*. If we call that flipping move* γ_i*, then* $\Gamma = \{\gamma_i\}_{i \in V}$ *with cardinality* $|V|$ *generates a group* $(G, \cdot)$

which acts on $\mathcal{X}$. This group is isomorphic to the product of cyclic groups, $\mathbb{Z}_2^V$, and is often directly identified as such.

Example 11 (Subset Selection)**.** *Fix a finite set S, and consider the task of selecting a subset $A \subset S$. The state space is then given by the power set of S, which is naturally isomorphic to $\{0,1\}^S$, by taking a 1 in the i^{th} coordinate to mean 'element i is included in the subset A'. Mathematically, this is now essentially equivalent to the spin system setting; one can define γ_i to act as*

$$\gamma_i \star A = \begin{cases} A \backslash \{i\} & \text{if } i \in A \\ A \cup \{i\} & \text{if } i \notin A, \end{cases}$$

i.e. delete i if it is already in A, otherwise include it. Again, taking $\Gamma = \{\gamma_i\}_{i \in S}$ generates a group $(G, \cdot)$ which acts on $\mathcal{X}$ and is isomorphic to $\mathbb{Z}_2^V$.

Example 12 (Lattice Distributions)**.** *Consider a lattice $L \subseteq \mathbb{Z}^n$ for some integer n, and a distribution π defined over all points of $\mathcal{X} = L$. Many classical discrete distributions used in probability and statistics are of this form. In this case, for a given point $x \in L$, we can define a move as either increasing or decreasing the value of a particular index x_i, $i = 1, 2, \ldots, n$:*

$$y = \gamma_i \star x \iff \begin{cases} y_i = x_i + 1 \\ y_j = x_j, \quad j \neq i \end{cases}$$

In this case it is clear that the inverse γ_i^{-1} corresponds to subtracting 1 from the i^{th} index of x. Then the symmetric generating set $\Gamma = \{(\gamma_i, \gamma_i^{-1})\}_{i \in \{1,2,\ldots,n\}}$ generates a group which acts on $\mathcal{X}$.

Example 13 (Lattice Polymers)**.** *In computational chemistry, one toy model for protein folding is given by lattice polymers, see e.g. [PGT94]. Here, one fixes a lattice $L \subset \mathbb{Z}^3$, a positive integer N, and considers the space $\mathcal{X}_N$ of length-N walks on L, i.e. sequences $(x_0, \cdots, x_N) \in L^{N+1}$ such that $d(x_i, x_{i+1}) = 1$ for all i. One also defines a potential function $V : \mathcal{X}_N \to \mathbb{R} \cup \{\infty\}$, and then seeks to sample from*

$$\pi(x) \propto \exp\left(-V(x)\right) \quad \text{where } x = (x_0, \cdots, x_N).$$

A polymer is then defined as a walk which never passes through the same lattice site twice, i.e. $i \neq j \implies x_i \neq x_j$. In this context, it is useful to then think of the potential function $V(x)$ as being infinite whenever x is a walk which is NOT *a polymer.*

In this context, there is a commonly-used groupoid[‡] which acts on $\mathcal{X}_N$. This group is

[‡]A groupoid is a group in which it the group composition operation need only be partially-defined, i.e.

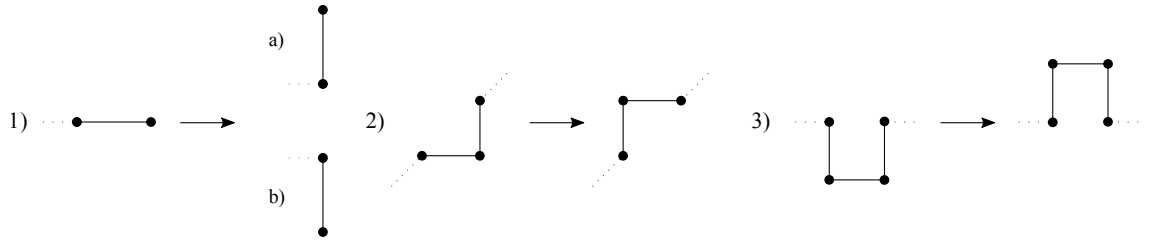

Figure 3.1: Illustration of the three types of generators for building polymers in protein folding.

simplest to describe by specifying its generators.

The first class of generators are those which arise by moving the end link of a walk, i.e. by taking x_0 and moving it to a different neighbour of x_1, or taking x_N and moving it to a different neighbour of x_{N-1}, see Figure 3.1, 1).

The second class of generators come from finding subsequences (x_i, x_{i+1}, x_{i+2}) which form an L-shape, and then flipping the orientation of the L as in Figure 3.1, 2). These are known as 'L-flips'.

The third class of generators comes from finding subsequences $(x_i, x_{i+1}, x_{i+2}, x_{i+3})$ which make up 3 of the 4 edges of a square, and then rotating these edges around the missing edge, as shown in Figure 3.1, 3). These are known as 'crank-shaft' moves.

We remark that the algebraic structure generated by these 3 types of move is non-Abelian, and quite complex. However, as in the cyclic group, the order of the elements in the generating set can be determined easily, and is low.

We note for completeness that given a space $\mathcal{X}$, a group $(G, \cdot)$ which acts upon $\mathcal{X}$, and a symmetric generating set Γ, it is possible to imbue $\mathcal{X}$ with a graphical structure directly via the generators, namely, one declares that x and y are neighbours precisely when there exists a generator $\gamma \in \Gamma$ such that $y = \gamma \star x$.

We re-emphasize that the benefit of this group-centred formulation over the general discrete space setting is that the space now 'looks the same' from all states x; no matter where in the space you are, the set of directions in which you are able to move remains the same (in graph-theoretic terms, this is known as *transitivity*). In particular, even though the graph may now have many edges, the 'effective' number of edges is now fixed at $|\Gamma|$. This opens us up to constructing more structured stochastic processes with which to sample from distributions defined on $\mathcal{X}$, as notions such as velocity and directionality can now be made sense of in a principled way.

there may exist elements which cannot be composed with one another.

3.2.2.1 Designing Non-Reversible Markov Processes on Discrete Spaces

In this section, we will outline some general principles for designing non-reversible Markov processes of a given invariant measure, on a structured discrete space. A core benefit of non-reversibility is that it allows for the construction of Markov processes which avoid back-tracking behaviour, which is often wasteful.

An observation we have found to be useful is that on discrete spaces with algebraic structure, the benefits of non-reversibility are qualitatively different, depending on whether the generators of the associated group are low-order, or high-order.

Definition 3.2.6. *Let $(G, \cdot)$ be a group, and let $g \in G \setminus \{id\}$. We say that g has order k if $g^k = id$, and if $g^j \neq id$ for $0 < j < k$. If there is no such k, we say that g has infinite order. We also say that the identity element id has order* 1.

In particular, if a group has low-order generators (e.g. $\gamma^2 = id$), then there is little benefit in repeatedly applying the same generator, as one swiftly ends up back in a previously-visited state. As such, we might believe that non-backtracking behaviour is best approached by discouraging the repeated use of generators.

In contrast, when the generators have high, or even infinite order, repeatedly applying a generator is taking the process to new states, which allows for persistent motion and exploration. With this in mind, on such spaces, we will try to construct processes which encourage the re-use of generators, when fruitful.

A useful notion in the high-order setting is that of a *reduced generating set*.

Definition 3.2.7. *Let Γ be a generating set for a group G. We say that $\Gamma_0 \subset \Gamma$ is a reduced generating set if, for all $\gamma \in \Gamma_0$, either $\gamma = \gamma^{-1}$, or $\gamma^{-1} \notin \Gamma_0$.*

Note that a generating set can only be both symmetric and reduced if every element is an involution, i.e. for all $\gamma \in \Gamma_0, \gamma^2 = 1$. The significance of this condition is that it makes it easier to construct processes which are strongly non-reversible, i.e. when it is possible to move in the direction γ, we can stipulate that moving in direction $-\gamma$ is forbidden. This is a particularly direct way of avoiding back-tracking behaviour.

In the algorithms which follow, we construct non-reversible Markov processes with a particularly tractable form of non-reversibility, known as *skew-reversibility*.

Definition 3.2.8. *Let $\mathcal{X}$ be a discrete state space equipped with an involution $S : \mathcal{X} \to \mathcal{X}$. Let π be a probability measure on $\mathcal{X}$ such that for all $x \in \mathcal{X}$, $\pi(x) = \pi(S(x))$. Let $Q : L^2(\pi) \to L^2(\pi)$ be the operator given by $\mathcal{Q}f(x) = f(S(x))$.*

A Markov Jump Process with generator $\mathcal{L}$ is skew-reversible with respect to the measure π and the involution S if, for all $f, g \in L^2(\pi)$, it holds that

$$\mathbf{E}_\pi \left[(\mathcal{Q}\mathcal{L}f)(x) g(x) \right] = \mathbf{E}_\pi \left[f(x) (\mathcal{Q}\mathcal{L}g)(x) \right].$$

By considering $f(x) = \mathbf{I}[x = a], g(x) = \mathbf{I}[x = b]$*, one can show that this is equivalent to*

$$\begin{aligned} \pi(x)\lambda(x \to y) &= \pi(S(y))\lambda(S(y) \to S(x)) && \text{for all } x, y \in \mathcal{X} \\ \Lambda(x) &= \Lambda(S(x)) && \text{for all } x \in \mathcal{X} \end{aligned}$$

A useful feature of skew-reversibility is that, as with standard reversibility, it provides a checkable, local condition, which ensures that the MJP in question leaves π invariant. We will repeatedly use this fact to construct algorithms in the remainder of this section.

In our examples, we will construct processes which are not skew-reversible on the original space $\mathcal{X}$, but on an augmented space[§] of the form $(x, u, \tau) \in \mathcal{X} \times \mathcal{U} \times \{\pm 1\}$. Roughly speaking, when the binary variable τ is equal to 1, the process will use a certain set of dynamics to move around, and when $\tau = -1$, the process will run those dynamics backwards in time, in a suitable sense. The generality of the construction will mean that we can easily construct skew-reversible Markov processes which admit the correct invariant measure, without requiring additional symmetry assumptions on the state space or target distribution.

3.2.3 Tabu Sampler: Self-Avoiding Walks on Spaces With Low-Order Generators

The first such process is adapted to the scenario in which the generating set consists of *low-order* elements, that is, for $\gamma \in \Gamma$, we have that $\gamma^k = \mathrm{id}$ for $k \geqslant 2$ a relatively small integer. In many cases (e.g. Bayesian variable selection, binary spin systems, and permutation problems), we can in fact take $k = 2$, and we focus on this case. The heuristic reasoning we apply in this setting is that one should prefer to avoid re-using generators, as if one applies the generator k times, the process has effectively backtracked. As such, we construct an MJP which, over short-to-medium timescales, is able to avoid such behaviour. Due to this property, we term the process the *Tabu sampler*, by analogy with the Tabu search meta-heuristic [GL98], which is commonly used in combinatorial optimisation.

The process operates on an extended state space, obtained by augmenting the original space $\mathcal{X}$ with two types of variables. The first is, for each generator γ, to append an indicator variable $\alpha(\gamma) \in \{\pm 1\}$. The second is a global indicator variable $\tau \in \{\pm 1\}$. All of these variables are equipped with the uniform distribution over $\{\pm 1\}$.

The behaviour of these variables is as follows: when $\tau = 1$, one is only able to move by using generators such that $\alpha(\gamma) = 1$, and when $\tau = -1$, one is only able to move by using generators such that $\alpha(\gamma) = -1$. Upon making a jump using the generator γ, one flips the variable $\alpha(\gamma)$, thus preventing it from being re-used. When the set of available

[§]Though note that the discrete Zig-Zag sampler (as described below) suppresses the dependence on τ.

moves becomes too small/unfavourable, the τ variable flips, and the previously-unavailable generators become available once more. We refer to the path of the process between these τ-flipping events as an 'excursion'.

The simplicity of the stationary uniform distribution of the $\alpha(\gamma)$ variables obscures their effective behaviour. In practice, the α variables function as a simple binary memory bank, encoding the successful trajectories of the past: as the algorithm is entering regions of low probability, the typically more desirable backward moves are stored for later use instead, giving the sampler the ability to escape potential wells. The Markov process resulting from running the Tabu sampler is therefore in practice memory-augmented, which is most clearly seen if one resets the α variable mid-run (while still leaving the target invariant). In this case, the sampler loses its sense of direction despite having mixed, and typically spends a long time rebuilding the memory bank before it resumes efficient exploration. Thus the α-variable provide a transparent mechanism for avoiding backtracking behaviour, which is necessary for navigating rough energy landscapes.

Algorithm 31 Tabu Sampler for Sampling $\pi(x), x \in \mathcal{X}$, when $\gamma^2 = \mathrm{id}$ for all $\gamma \in \Gamma$

1. At $x \in \mathcal{X}, \{\alpha(\gamma)\}_{\gamma\in\Gamma} \in \{\pm 1\}^{\Gamma}, \tau \in \{\pm 1\}$,
   (a) For $\gamma \in \Gamma$ such that $\alpha(\gamma) = \tau$, compute $\lambda(\gamma; x, \alpha, \tau) = g\left(\frac{\pi(\gamma\cdot x)}{\pi(x)}\right)$.
   (b) For $\gamma \in \Gamma$ such that $\alpha(\gamma) = -\tau$, compute $\lambda(\gamma; x, \alpha, -\tau) = g\left(\frac{\pi(\gamma\cdot x)}{\pi(x)}\right)$.
   (c) Compute

$$\Lambda(x;\alpha,\tau) = \sum_{\gamma\in\Gamma} \lambda(\gamma; x, \alpha, \tau)\cdot \mathbf{I}[\alpha(\gamma) = \tau]$$
$$\Lambda(x;\alpha,-\tau) = \sum_{\gamma\in\Gamma} \lambda(\gamma; x, \alpha, -\tau)\cdot \mathbf{I}[\alpha(\gamma) = -\tau]$$
$$\Lambda(x;\alpha) = \max\left(\Lambda(x;\alpha,\tau), \Lambda(x;\alpha,-\tau)\right).$$

   (d) Sample a waiting time $T \sim$ Exponential(rate $= \Lambda(x;\alpha)$), and advance time by T.
      i. With probability $\frac{\Lambda(x;\alpha,\tau)}{\Lambda(x;\alpha)}$,
         A. Sample a new direction $\gamma \in \Gamma$ with probability $\frac{\lambda(\gamma;x,\alpha,\tau)\cdot\mathbf{I}[\alpha(\gamma)=\tau]}{\Lambda(x;\alpha,\tau)}$.
         B. Flip the value of $\alpha(\gamma)$ to $-\alpha(\gamma)$.
         C. Jump to $y = \gamma \cdot x$.
      ii. With probability $\frac{\Lambda(x;\alpha)-\Lambda(x;\alpha,\tau)}{\Lambda(x;\alpha)}$, flip the value of τ to $-\tau$.

From an implementation perspective, the Tabu sampler is essentially the same complexity as the LBJP; the main cost is still the repeated access to quantities of the form $\frac{\pi(\gamma\cdot x)}{\pi(x)}$. One now has to also maintain the (α, τ) variables, but this cost is negligible.

An interpretation of the Tabu sampler that can make clear its behaviour is the following. Consider the $|\Gamma|$-dimensional hypercube. For each jump, an entire dimension of

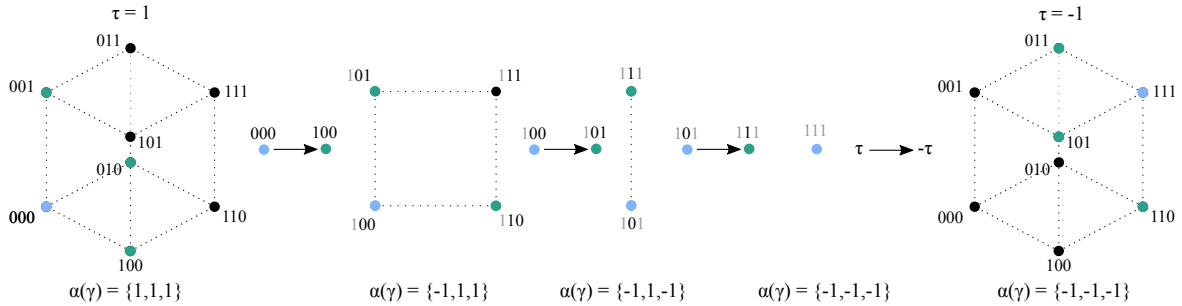

Figure 3.2: An illustration of the Tabu sampler on the $2 \times 2 \times 2$ hypercube. The initial position is illustrated with a light blue dot, the available jumps after applying a generator are coloured light green. For each position, indices available via applications of available generators are coloured black, unavailable locations are light grey. Initially, $\tau = 1$, and all 3 generators are available to use. As the sampler jumps by successively applying generators, the cardinality of the available state-space is halved at each iteration. Furthermore, after 3 jumps the sampler has run out of options, forcing a flip of time (by setting $\tau = -\tau$). This subsequently makes the previously used generators available, and, in this case, all positions become available to the sampler again.

the hypercube is removed from the set of accessible points, i.e., after a jump, the side of the cube associated with γ is inaccessible until τ changes direction. In this sense the Tabu sampler performs a dimension reduction at each jump until a reversal of time occurs, after which all previously inaccessible dimensions are made available again. In figure 3.2 we illustrate this behaviour in the simple case of the $2 \times 2 \times 2$ hypercube.

We remark quickly on an interesting 'self-tuning' property of the Tabu sampler, namely, the mechanism which allows for τ to flip. If the total event rate out of the current state, $\Lambda(x; \alpha, \tau)$ is heavily-dominated by the total event rate out of its mirror state, $\Lambda(x; \alpha, -\tau)$, then the process is highly likely to flip τ in order to access the mirror state. In effect, the process is able to discern that flipping τ would provide a wider range of desirable neighbours to jump to, and is thus somewhat able to adapt to being out of equilibrium in this respect. This also implies that the Tabu sampler will be more effective if the posterior density is multi-modal, as this will encourage diversity in the neighbour set. In numerical experiments, we will estimate the mean excursion as the average number of events of type (d).i which occur before each flip of τ. The realized mean excursion can be interpreted as a proxy for the complexity of the target distribution, as more accepted jumps indicate that the distance between regions of high probability is larger. This situation is one where the Tabu sampler can be expected perform better relative to the LBJP; see the examples in Section 3.3.2 for numerical evidence of this.

The most closely-related existing work of which we are aware is the SARDONICS algorithm presented in [HWDF13]. This is a discrete-time algorithm in which a guided, self-avoiding path of length k is constructed sequentially, the final state of which is then used as a Metropolis-Hastings proposal. While appealing in principle, the algorithm admits

some complications; in particular:

- Despite constructing a path of length k, the proposal ultimately only involves the final state. As such, the self-avoiding path could pass through good potential states which it ultimately has to ignore. As such, the proposal mechanism can be wasteful.
- Tuning of k, or tuning of the randomisation procedure for k is nontrivial.
- As k grows, the complexity of computing the Metropolis-Hastings ratio also grows.

In contrast, the Tabu sampler which is presented in this work softens the constraint of being fully self-avoiding, but retains the propensity of being approximately self-avoiding over short-to-medium timescales. As such, from a practical point of view, it appears to present a tractable alternative which retains most of the desirable behaviours of the fully self-avoiding construction.

3.2.4 Persistent Piecewise Deterministic Markov Processes for Spaces With High Order Generators

Our second class of algorithms is instead adapted to the setting of *high-* or *infinite-order* generators. A reasonable picture to have in mind for this setting is taking $\mathcal{X}$ to be the lattice $\mathbb{Z}^d$, with generators given by the axis-aligned unit vectors. Here, it is less clear that self-avoidance would be beneficial, and it may instead be preferable to *encourage* the re-use of generators, in order to allow for persistent motion across the space. This heuristic motivates the design of our two *Discrete PDMP* algorithms.

3.2.4.1 Discrete Zig-Zag Process

While it is possible to construct a version of the Tabu sampler which can handle low-order generators of order greater than two, we found that designing such an algorithm was slightly less natural, and required more tuning choices to be made. We thus opted instead to seek an algorithm which would retain the high-level behaviour of the Tabu sampler, while being more straightforward to tune and implement. This led us to the *discrete Zig-Zag Process* (dZZ).

The dZZ process also operates on an extended state space, though uses a different augmentation. Throughout, we work with a reduced generating set Γ_0, and for each $\gamma \in \Gamma_0$, we augment the state space with a binary variable $\theta(\gamma)$, equipped with the uniform distribution over $\{\pm 1\}$. The role of $\theta(\gamma)$ is as follows: at any given time, the process may only use the generator γ either forwards (moving from x to $\gamma \cdot x$, when $\theta(\gamma) = 1$), or backwards (moving from x to $\gamma^{-1} \cdot x$, when $\theta(\gamma) = -1$). Meanwhile, if it were sufficiently beneficial to use the generator γ in the opposite direction, then with high probability, the process will flip $\theta(\gamma)$. In this sense, the process is 'self-tuning' in the same fashion as the Tabu sampler.

Algorithm 32 Discrete Zig-Zag Process for Sampling $\pi(x), x \in \mathcal{X}$

1. At $x \in \mathcal{X}, \theta \in \{\pm 1\}^{\Gamma_0}$,
   (a) For $\gamma \in \Gamma_0$, compute

$$\lambda(x, \gamma; \theta) = g\left(\frac{\pi(\gamma^{\theta(\gamma)} \cdot x)}{\pi(x)}\right)$$
$$\lambda(x, \gamma; -\theta) = g\left(\frac{\pi(\gamma^{-\theta(\gamma)} \cdot x)}{\pi(x)}\right)$$
$$\Lambda(x, \gamma) = \max\left(\lambda(x, \gamma; \theta), \lambda(x, \gamma; -\theta)\right)$$

   (b) Compute $\Lambda(x) = \sum_{\gamma \in \Gamma_0} \Lambda(x, \gamma)$.
   (c) Sample a waiting time $T \sim$ Exponential(rate $= \Lambda(x)$), and advance time by T.
   (d) Sample a generator γ with probability $\frac{\Lambda(x,\gamma)}{\Lambda(x)}$, and
      i. With probability $\frac{\lambda(x,\gamma;\theta)}{\Lambda(x,\gamma)}$, jump to $y = \gamma^{\theta(\gamma)} \cdot x$.
      ii. Otherwise, flip the value of $\theta(\gamma)$ to $-\theta(\gamma)$.

A favourable aspect of this process is that it is relatively robust to poorly-scaled targets; if *any* of the available directions are good, then the process is able to sniff them out. The per-iteration cost is comparable to the LBJP, but the non-reversibility allows for a desirable persistent behaviour.

An added by-product of the dZZ process is that the output of this algorithm can suggest new directions which one could add to the generating set; if one often sees moves where an application of γ_1 is followed by an application of γ_2, then one can reasonably augment the generating set to include the move $\gamma_2 \star \gamma_1$. This presents one opportunity for adaptation of these algorithms; we leave exploration of this idea to future work.

3.2.4.2 Discrete Coordinate Sampler

For our third algorithm, we explicitly aim to solve problems in which the group is generated by high-order elements. In this setting, we seek to exhibit persistent behaviour across the space, i.e. if moving from x to $\gamma \cdot x$ is successful, then we will attempt to make additional moves to $\gamma^2 \cdot x, \gamma^3 \cdot x$, and so on.

As in the previous two cases, we operate on an extended state space, though the interpretation is now somewhat different. We first add in a variable v, taking values in our symmetric generating set Γ, which is drawn according to some symmetric distribution ψ (i.e. such that $\psi(v) = \psi(v^{-1})$. We then include a 'direction of time' variable τ, which is equipped with the uniform distribution on $\{\pm 1\}$. In effect, v behaves as a velocity, and τ dictates whether to follow the velocity forwards or backwards in time.

Broadly, the walk attempts to follow the velocity in the direction of time τ. When the walk is heading towards regions of higher probability, $\delta(x, v, -\tau)$ will be equal to 0, and

Algorithm 33 Discrete Coordinate Sampler for Sampling $\pi(x), x \in \mathcal{X}$, $v \sim \psi(v)$

1. At $x \in \mathcal{X}, v \in \Gamma, \tau \in \{\pm 1\}$,
   (a) Compute

$$\begin{aligned}\delta(x, v, \tau) &= g\left(\frac{\pi(v^{\tau} \cdot x)}{\pi(x)}\right) \\ \delta(x, v, -\tau) &= g\left(\frac{\pi(v^{-\tau} \cdot x)}{\pi(x)}\right) \\ \Delta(x, v) &= \max\left(\delta(x, v, \tau), \delta(x, v, -\tau)\right).\end{aligned}$$

   (b) Sample a waiting time $T \sim$ Exponential(rate $= \Delta(x, v)$), and advance time by T.
      i. With probability $\frac{\delta(x,v,\tau)}{\Delta(x,v)}$, jump to $y = v^{\tau} \cdot x$.
      ii. Otherwise, with probability $\frac{\Delta(x,v)-\delta(x,v,\tau)}{\Delta(x,v,\tau)}$, sample a new velocity w according to

$$\begin{aligned}Q(w|x, \tau) &= \frac{\psi(w)\rho(x, w, \tau)}{Z(x)} \\ Z(x) &= \frac{1}{2} \sum_{\tau \in \{\pm 1\}} \sum_{\gamma \in \Gamma} \psi(\gamma)\rho(x, \gamma, \tau),\end{aligned}$$

      where

$$\rho(x, v, \tau) = \left[\delta(x, v, -\tau) - \delta(x, v, \tau)\right]_{+}$$

      set the value of v to w, and flip the value of τ to $-\tau$.

thus the walk will continue make moves in the direction v^τ. Once the walk has started to head towards regions of lower probability, the $\delta(x, v, -\tau)$ term will begin to dominate, and the walk will instead try to modify its own velocity, and pursue a new direction.

We note that relative to the discrete Zig-Zag Process, the computational complexity of making a single forwards jump with the discrete Coordinate Sampler is much cheaper; in particular, it is independent of the size of the generating set. This represents a substantial benefit in certain high-dimensional scenarios. The cost is then that some robustness is lost; as the discrete Zig-Zag process is always able to look in multiple directions, it should be able to adapt to changing 'curvature' more gracefully. In contrast, for 'anisotropic' target distributions, the discrete Coordinate Sampler will likely have to resample its velocity variable quite frequently, which could offset the reduced per-iteration cost.

We caution quickly that there is an odd pathology which can cause the sampler to become reducible when sampling from highly-symmetric target distributions. In particular, if there is a subset $S \subset \mathcal{X}$ and a generator v such that $\delta(x, v, \tau) = \delta(x, v, -\tau)$ for all $x \in S$, then whenever a velocity jump event is experienced in S, the newly-resampled velocity cannot be equal to v, and this can lead to reducibility under certain circumstances. This is often not a problem - in particular, if the other velocities can allow the process to exit the set S, then the problem typically vanishes - but for certain initialisations, this can cause undesirable behaviour. As a safeguard, we generally recommend adding in a small refreshment mechanism, i.e. at some constant rate, the velocity v is resampled from its marginal distribution.

3.2.5 Related Work

The most closely-related work we are aware of comes from the literature on non-reversible MCMC. A good starting reference is [DHN00], which presents an in-depth study of a sampler of this form on the state space $\mathcal{X} = \{1, 2, \cdots, N\}$. A number of related constructions are also presented; in particular, the 'fiber algorithm' described therein directly motivated the construction of our discrete Zig-Zag Process, which of course bears many similarities to the more recent Zig-Zag Process of [BFR19]. [CLP99, TCV11, Vuc16] are all also closely related in spirit, each focusing on how non-reversibility can be harnessed in order to improve convergence to equilibrium. In the continuous setting, a number of sampling algorithms based around *Piecewise Deterministic Markov Processes* (PDMPs) have also been presented with great success (we recommend [VBCDD17] for a recent technical overview); these behave in much the same way (hence the name) and served as a large part of the motivation.

3.3 Numerical Examples

In this section we first discuss implementation choices for practitioners, and subsequently provide numerical evidence of the performance of the Tabu, dC and dZZ samplers. Code in Python 3 for all examples are available online at `https://github.com/jvorstrupgoldman/tabu_dc_dzz`.

3.3.1 Implementation of Continuous-time Samplers

As discussed previously, a MJP on a discrete space can be implemented exactly without discretization error, as no numerical integration is necessary. To estimate statistical quantities $\mathbf{E}_\pi[f(x)]$ for $f : \mathcal{X} \to \mathbb{R}$ from a realisation of a process $(X)_{t\in[0,T]}$, two approaches are available. By ergodicity, the empirical time-average approaches the true expectation in the limit:

$$\lim_{T\to\infty} \frac{1}{T} \int_{[0,T]} f(X_s)ds = \mathbf{E}_\pi \left[f(x)\right],$$

and the integral on the left-hand side may be calculated explicitly as

$$\frac{1}{T} \int_{[0,T]} f(X_s)ds = \sum_{k\geq 1} \frac{\tau_k - \tau_{k-1}}{T} f(X_{\tau_k}),$$

where the sum ranges over each event-time and X_{τ_k} is the value of the process just after the k'th event. Alternatively, a thinning procedure can be applied to the process. In this case, a thinning interval $0 < \vartheta < T$ is chosen such that $\frac{T}{\vartheta}$ is an integer, and at each of these thinning times the process state is stored. Expectations are then calculated simply via the empirical average

$$\mathbf{E}_\pi \left[f(x)\right] \approxeq \frac{\vartheta}{T} \sum_{i=0}^{\frac{T}{\vartheta}} f(X_{i\cdot\vartheta}).$$

The resulting difference in the estimates from using the thinned samples is in practice completely negligible, however, the thinning procedure allows one to explicitly define the process run-time T when pre-allocating storage for each thinned sample, while the discrete time integral-approach requires pre-allocating storage for each event-time, which implies that the final time T is random, or that unused excess storage is necessary. For these reasons, we apply the thinning method to generate samples while the algorithm is running. After an initial trial run, we set the thinning rate ϑ to approximately be equal the mean event-time after the process has reached stationarity. In other words, in this case we on average expect a single event to have occurred for each thinned sample.

3.3.1.1 Choice of Balancing Function

In [Zan19] it is shown that balancing functions in general are Peskun-optimal [Pes73] weighting functions. Furthermore, while the author establishes that in the particular case of independent Bernoulli variables the Barker balancing function is the optimal choice, there is still far from a complete theory concerning the optimal choice of g for general targets $\pi(x)$.

To explore these issues, we consider the *embedded jump chain* $(\hat{x}_k)_{k\geq 1}$ of the LBJP, which is the discrete-time Markov chain consisting of values of the process evaluated after each event-time τ_k, i.e. $\hat{x}_k = X_{\tau_k}$, where $\tau_0 = 0$. The invariant distribution of the jump chain (hereafter, the 'jump measure') is determined by the choice of balancing function as

$$\pi_g^J(x) \propto \pi(x)\Lambda(x) = \pi(x)\left(\sum_{y\in\partial x}\lambda(x\to y)\right) = \pi(x)\left(\sum_{y\in\partial x} g\left(\frac{\pi(y)}{\pi(x)}\right)\right).$$

By studying the effects of the balancing function on the jump measure, we can seek to understand its effects on the mixing of the underlying Markov process.

One approach which may prove insightful here is to consider how the choice of g affects the metastability of the jump chain. In particular, if the jump measure exhibits lower energy barriers between modes, we should expect more desirable mixing behaviour.

A more concrete way to probe this relationship is to study the ratio of the target distribution π to the jump measure π^J, as it provides a sense of where the jump process places emphasis relative to the target. We present here some simple one-dimensional examples on $\mathcal{X} = [50]$, where the neighbourhood structure is defined by setting $\partial x = \{x-1, x+1\}$. In Figure 3.3, we plot three distributions of increasing complexity, and the corresponding jump measures for the different balancing functions. We also test the *non-balanced* (or 'globally-balanced') weighting function $g(t) = t$ to contrast its behaviour.

One heuristic for desirable behaviour is for the jump measure to be remain close to π, as larger discrepancies require more effort from the continuous time process to correct for. This heuristic argument is related to what is put forward in [Zan19, Section 2.1] in favour of balancing functions.

It is clear from Figure 3.3 that an increased complexity of the target distribution emphasises the need for balanced weighting functions, but even within the class of balancing functions, one can observe significant differences in behaviour. For the Metropolis balancing function, modes are over-emphasized in the jump measure, while low-probability regions are visited significantly less often. The square root is generally more robust around modes, but in contrast to the Metropolis balancing function, puts significantly higher probability on extreme regions. The Barker function seems in general to balance the best features of Metropolis and square root, with reasonably stable behaviour around peaks and troughs,

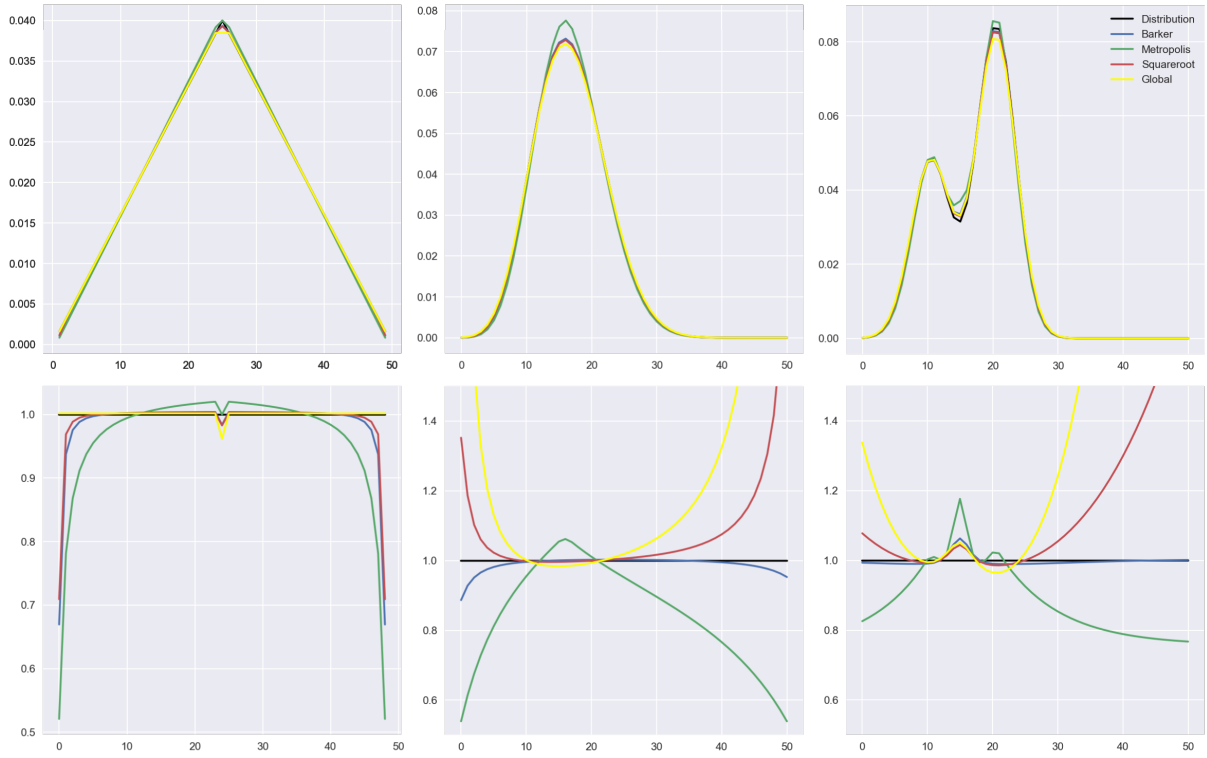

Figure 3.3: Upper row: Three distributions (Triangle, Beta-Binomial(10, 20) and a marginal of a mixture of lattice Gaussians) and their jump-chain invariant distributions for the three admissible balancing functions and the global balancing function. Lower row: Relative ratio of the jump-distribution to the invariant, $\pi_g^J(x)/\pi(x)$.

and significantly less erratic tail behaviour. The non-balanced function performs better than its local counterparts for relatively flat targets, but for more interesting targets performs exactly opposite to what is desired.

3.3.2 Examples with Low Order Generators

We here present five examples that cover a wide swathe of discrete models applied in statistics, machine learning and physics. For each of the examples it is the case that the generators are of order 2. As a benchmark, we compare our non-reversible samplers **only** to the LBJP. We view this as a fair comparison, as i) the LBJP is a generic sampling algorithm (i.e. it is not adapted to a specific model structure) which uses only local information, and ii) the results presented in [Zan19] demonstrate its superior performance relative to other generally-applicable discrete samplers, such as the Hamming Ball sampler of [TY17], random walk proposals, and Gibbs-type schemes. In Table 3.1, we provide a short summary of the performance of the Tabu sampler relative to the LBJP in terms of the ratio of effective sample size per second (ESS/s)[¶], with higher being better for the

[¶]Care has been taken to ensure that the implementation of the samplers is fully comparable in terms of computational resources spent per event.

Tabu sampler. We also note the mean excursion, which we defined above as average length of the self-avoiding walk. As mentioned, the mean excursion provides a good indication of the complexity of the target distribution, and we in general expect the performance of the Tabu sampler relative to processes that allow backtracking to increase as the mean excursion goes up. For each experiment, we ran the samplers 5 times and averaged over each run. Furthermore, for the experiments involving simulated data, we also re-initialized the data to increase the accuracy of the comparison.

	Target Statistic	**MX**	**T/L**
Bayesian Variable Selection	Number of parameters	6.0	1.57
Conditional Permutation Test	Hamming distance from mode	12.8	1.16
Sherrington-Kirkpatrick Model	Energy	83.4	79.89
Log-Submodular Point Process	Sensor count	31.8	24.48
Determinantal Point Process	Number of points	11.0	4.98

Table 3.1: Summary of performance for target distributions with order-2 generators over 5 runs. 'MX' is the mean excursion time, i.e. the average number of events between a time-reversal of τ. 'T/L' is the ratio of ESS/s for the Tabu sampler relative to the LBJP after burn-in, discarding the first 20% of 100, 000 samples. Each ESS calculation is carried out with autocorrelation lags up to order 3000.

3.3.2.1 Bayesian Variable Selection

The probabilistic selection of covariates in problems such as regression has long been an active part of theoretical and applied Bayesian statistical research [MB88, GM97, LSD$^+$03]. While efficient algorithms exist for fitting many of the most popular Bayesian models, the hierarchical framework allows for complex interactions that can make posterior exploration very difficult. We analyze here a hierarchical Bayesian model presented in [SC13], which postulates that the m-dimensional observation vector y in a linear regression framework is observed through

$$y \mid \beta, x, \sigma^2, Z \sim \mathcal{N}(ZI^x\beta, \sigma^2 I_m),$$

with covariates $Z \in \mathbb{R}^{m\times n}$, parameters $\beta \in \mathbb{R}^n$ and the matrix of active covariates $I^x = I_n x$, with the binary inclusion vector $x \in \mathcal{X} = \{0,1\}^n$ our target variable. The group action here is equivalent to the one presented for spin glasses in Example 10, with the generator in this case instead given by picking a single entry in x, x_i, and setting $x_i' = 1 - x_i$. To achieve a closed form marginal posterior that is independent of (β, σ^2)

$$\pi(x|Z, y) = \int \pi(x|Z, y, \beta, \sigma^2) d(\beta, \sigma^2),$$

we pick conjugate priors

$$p(\beta|\sigma^2, x) = \mathcal{N}(0, v^2\sigma^2 I^x), \quad p(\sigma^2) = \text{I}\Gamma\left(\frac{w}{2}, \frac{\lambda w}{2}\right), \quad p(x) = \mathcal{U}(x|\mathcal{X}),$$

with IΓ the inverse-gamma distribution and $\mathcal{U}(\cdot|\mathcal{X})$ the normalised uniform law on $\mathcal{X}$. Setting the hyper-parameter vector (w, v, λ) as in [GM97], we use the Concrete Compressive Strength dataset, originally analysed in [Yeh98], which includes 8 covariates used to explain the compressive strength of concrete, as measured in giga-Pascals. The dataset is augmented with a constant column of ones, five logarithmic variables, and first-order interactions of the log-transformed and initial covariates for a total of $n = 92$ parameters and $m = 1030$ observations in the saturated model. We do not enforce main effects restrictions, which implies that interaction terms can be included independently of whether baseline covariates are included in the model or not. To have manageable rate sizes, the Barker balancing function $g(t) = \frac{t}{1+t}$ is actually needed in this case, as the change in probability from changing the variable subset can be substantial. The goal of sampling $\pi(x)$ is to derive the marginal inclusion probability $\rho_i = \mathbf{E}_\pi[\mathbf{1}_{x_i=1}(x)]$ of each variable in the model, rather than just MAP estimates as is commonly done. The resulting inclusion probabilities from the Tabu sampler or the LBJP are virtually indistinguishable from the ones estimated in [SC13, Figure 5], in comparison with the huge variability observed for the discrete-time adaptive MCMC and standard MCMC samplers there. Nonetheless, the total number of evaluations of the posterior density is higher for the LBJP and the Tabu sampler compared to the tailor-made SMC procedure applied by [SC13], which only evaluates the density once per iteration.

To evaluate the effective sample size, we decide against using the energy, as the posterior is dominated by a single configuration, and this mode is often revisited. Rather, we use the number of active variables as the test statistic. It is noteworthy here that there seems to be limited benefit available from using the Tabu sampler for this dataset, as the posterior appears to be sufficiently well-behaved that the self-avoiding behaviour is largely unnecessary. As such, using locally-balanced proposals allows for the posterior to be explored properly and efficiently. In fact, across some time-scales, the self-avoiding property of the Tabu sampler can actually impede the mixing of the chain, as the sampler can corner itself and be forced to move in a bad direction, recall Figure 3.2 where the sampler ran out of options after 3 jumps. This stands in contrast to the LBJP, which is always able to revert to a previous configuration. Nonetheless, the slightly better performance of the Tabu sampler over repeated runs can reassure us that self-avoidance does not lead to worse performance overall.

3.3.2.2 Conditional Permutation Test

We here consider a version of the matching problem already described in Example 8. Let $S = \{1, 2, \ldots, n\}$ and consider the space $\mathcal{X} = \Sigma_n$ of permutations of S. The goal is to explore likely orderings of S that have high likelihood under our model, which we now describe. For all integer combinations $(i, j) \in S \times S$ we associate a likelihood of that particular pairing given by $\omega_{i,j}$, where $\omega_{i,j} \sim \log \mathcal{N}(0, \sigma^2)$ independently. For any permutation $x \in \mathcal{X}$ we define the target density as

$$\pi(x) = \frac{1}{Z} \prod_{i=1}^{n} \omega_{i,x_i},$$

with $Z = \sum_{x \in \mathcal{X}} \prod_{i=1}^{n} \omega_{i,x_i}$ the normalizing constant. For large σ^2 the log-normal has quite heavy tails, and thus the distribution will be dominated by some pairings being much more likely than others, giving rise to multimodality. However, in comparison with for example the much more narrow distribution of the correlation coefficients J_{ij} in the Sherrington-Kirkpatrick model below, only a few steps are on average needed to travel between close modes. In practice, for the most difficult i.i.d. case considered in [Zan19] where $n = 500$ and $\sigma^2 = 5$, the average excursion length is around 13, which is comparable to what was observed in the determinantal point process example of Section 3.3.2.5. Overall, the Tabu sampler performs similarly to the LBJP, indicating that there is little benefit of avoiding backtracking behaviour when the distribution is very peaked at the modes.

3.3.2.3 Spin Glasses: The Sherrington-Kirkpatrick Model

Spin glasses are a class of models of magnets with competing ferromagnetic and antiferromagnetic interactions, and which are widely studied in statistical physics, in particular condensed matter, but also applied in diverse fields such as protein folding [BW87], neuroscience [FT06] and spatial economics [Kru94]. As in example 10, consider a 2-dimensional lattice $V = L_n^2$ with side-length n, where each vertex is inhabited by a binary spin $x_i \in \{\pm 1\}$. We let $|V| \equiv n^2$ be the overall cardinality of the graph.

We will analyze the Sherrington-Kirkpatrick (SK) model, a lattice-wide model with random interactions on V, implying that the graph (V, E) is complete, i.e. fully connected. The log-probability, or negative Hamiltonian H, of the target is

$$\log \pi(x) \equiv -\beta H(x) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} J_{ij} x_i x_j + h \sum_{l=1}^{n} x_l = \frac{1}{n} \sum_{i=1}^{n} x_i \sum_{j \neq i} J_{ij} x_j + h \sum_{l=1}^{n} x_l + \frac{J_{ll}}{nh},$$

where $J_{ij} \sim \mathcal{N}(0, \beta^2 (2n)^{-1})$ if $i \neq j$ and 0 otherwise, and $h > 0$. In both cases, we simply absorb the inverse temperature β into the likelihood constant h and the correlation

coefficient. It follows that there are $|V|^2$ interaction terms, so the model scales in storage costs at order $O(n^4)$. The SK model generalizes many of the common spin-glass models: If $J_{ij} = c$, a constant, for all $(i,j) \in L_n^2$, the model reduces to the Curie-Weiss model. On the other hand, if the model is local in the sense that two vertices (i,j) and (l,m) only interact when $|i-l| + |j-m| = 1$, the model reduces to the Edwards-Anderson model. Finally, if both simplifications are assumed, we revert to the classical Ising model.

With the generator flipping the i^{th} vertex denoted by γ_i, we can exploit the sum-structure of the probability distribution when calculating the rates. The jump rates can be calculated via

$$\begin{aligned}\lambda(\gamma_i; x) = g\left(\frac{\pi(\gamma_i x)}{\pi(x)}\right) &= g\left(\exp\left\{H(\gamma_i x) - H(x)\right\}\right) \\ &= g\left(\exp\left\{\frac{4}{n} x_i \left(\sum_{j=1}^{n} J_{ij} x_j + \frac{1}{2} h\right)\right\}\right).\end{aligned}$$

More importantly, the update of vertex l given a previous update at i is simply given by $\partial_l H(\gamma_i x) = -\frac{8}{n} J_{il} x_i x_l$, which significantly speeds up computations. We evaluate the model with $n = 100$, $\beta = 1$ and $h = 0.1$. For initialisation, we pick $x_0 = \{1, 1, 1, \ldots, 1, 1\}$. The highly multi-modal nature of the Hamiltonian is a situation where the Tabu sampler can be expected to do well, as proper exploration is contingent upon leaving potential wells, which should be easier as generators leading back towards the well are removed from $\alpha(\gamma)$.

In practice, we observe that for our spin-system with 10,000 spins, the average excursion length of the Tabu sampler is 83, or nearly 1% of the possible spins. In comparison with the four other examples, this is the highest observed excursion length, indicating that modes are distantly spaced for the Sherrington-Kirkpatrick model, in comparison for example with the conditional permutation test of Section 3.3.2.2. We conjecture in general that the Tabu sampler will perform better than other neighbourhood-based samplers the further spaced modes are in terms of group actions required to reach a neighbourhood of a mode; we postpone theoretical analysis to future work. In terms of ESS/s, the Tabu sampler performs close to two order of magnitudes better than the LBJP, which already in the discrete-time experiments of [Zan19] was shown to perform 40-50 times better for the same metric, in the much simpler case of a ferromagnetic 2D Ising model, than random walk proposals, the HB sampler [TY17] and the D-HMC sampler [PP13]. In Figure 3.4, we display the autocorrelation function and the trace plot of the energy for the two samplers.

3.3.2.4 Log-Submodular Distributions I: Facility Location

Modular functions have found broad application within machine learning and combinatorial optimization (see e.g. [Bac19]) in recent years, and provide a cohesive framework for modelling repulsion and regularity of subsets of points in space. For some integer m,

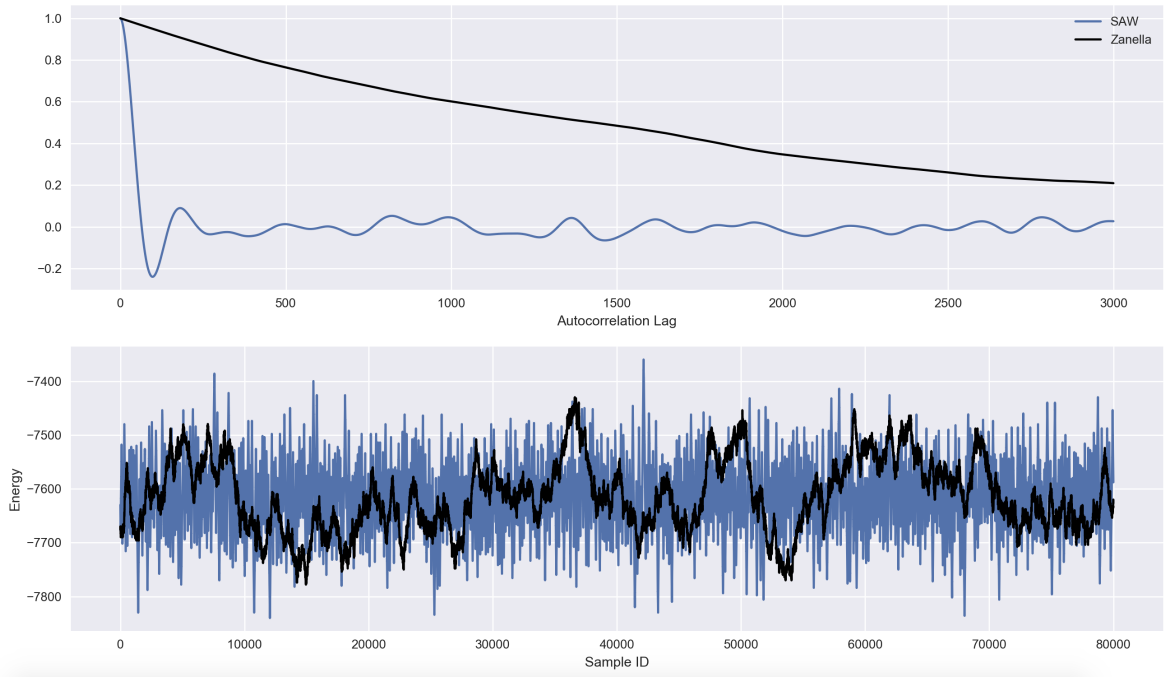

Figure 3.4: Autocorrelation function and energy traceplot for 80, 000 burned-in samples drawn from the 100×100 Sherrington-Kirkpatrick Hamiltonian via the Tabu sampler and LBJP.

consider the set $\mathcal{X} = 2^m$. We say a function $f : \mathcal{X} \to \mathbb{R}$ is submodular if for any sets S, T such that $S \subseteq T$ and any $i \in \mathcal{X} \setminus T$,

$$f(S \cup i) - f(S) \geq f(T \cup i) - f(T),$$

that is, there are diminishing marginal returns from making sets larger. To define a probability distribution, we simply let up to proportionality the probability of a given set S be $\pi(S) \propto \exp\{f(S)\}$. In the following two examples we consider variations on these kinds of distributions.

We first consider a variation on the facility location problem [CG99, JV01] for internet access points. Let $\mathbb{K}$ denote a non-convex polygon in $\mathbb{R}^2$. Inside $\mathbb{K}$ we have a set of m equidistant points consisting of $(v_l)_{1 \leq l \leq m}$, from hereon known as access points. Also in $\mathbb{K}$, there are n individual users $(u_j)_{1 \leq j \leq n}$ placed randomly according to a Gaussian distribution centered at the centroid of $\mathbb{K}$. For each user u_j we calculate the utility available from using the access point v_i as $\Upsilon_{ij} = \exp\{-\kappa \|u_j - v_i\|^2\}$, where κ is a parameter representing the physical signal decay, the distance is contained in the convex hull of $\mathbb{K}$. Consider the value function h

$$h(S) = \sum_{j=1}^{n} \max_{i \in S} \Upsilon_{ij};$$

the function returns the maximum value that can be extracted for all users given they optimally use the closest access point available to them. For installation costs, we have $g_1(S) = -\lambda|S|$ for some fixed cost-parameter $\lambda > 0$, where $|S|$ is the cardinality of the set S. The probability distribution given by

$$\pi(S) \propto \exp\left\{h(S) - g_1(S)\right\}$$

is then a log-submodular probability distribution. To incorporate capacity constraints, we introduce

1. The choice of sensor by user j, given by

$$C(S, j) = \arg\max_{i \in S} \Upsilon_{ij}$$

2. The total number of users of sensor i, given by

$$B(S, i) = \sum_{j=1}^{n} \mathbf{I}[C(S, j) = i]$$

3. The capacity cost function, given by

$$g_2(S) = \psi \sum_{i=1}^{m} \max\{0, B(S, i) - \Psi\}$$

where $\psi > 0$ models the sensitivity of user response to exceeded capacity, and $\Psi \in \mathbb{N}$ is the maximum allowable capacity before bandwidth declines below some acceptable threshold. Although the function $f' = h - g_1 - g_2$ is not submodular, it provides an example of a capacity-constrained facility location problem, and thus an interesting benchmark case, given that it incorporates significant shifts in probability when capacities are reached for access points.

As an illustrative example, we consider the case of providing access to randomly placed users where $\mathbb{K}$ is given as in Figure 3.5, a model football stadium. The access point grid considers a vertical and horizontal spacing of 5 for a total of $m = 704$ access points, and a random number of spectators are drawn inside the stadium boundaries with higher propensity of being along the lateral sides of the field, the expected number of spectators is 8750. Assuming that each access point provides 100 $mbit/s$ download and upload rates, we set the capacity constraint to $\Psi = 25$ users, and $\lambda = \psi = 1$, which corresponds to the complete loss of utility for a single optimally placed user for spectator connected user in excess of access point capacity. For the utility gained, we use an adjustment factor $\kappa = n$. To assess the capacity for the samplers to provide suitably different configurations of sensors, we use the number of sensors as the target statistic. The observed mean excursion

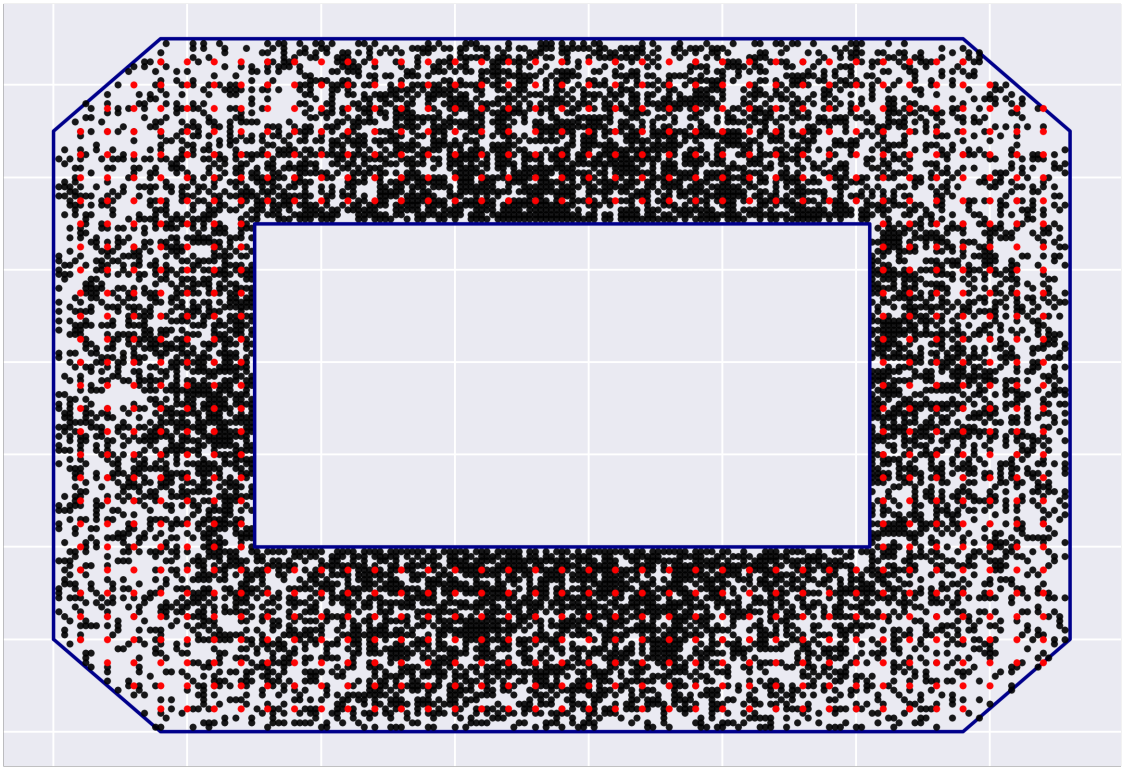

Figure 3.5: Toy-model of a stadium. Each black dot corresponds to a spectator, and each red dot corresponds to a sensor.

in this example is just below 32 for the Tabu sampler, indicating significant complexity in the distribution, and the improvement in effective sample size relative to the LBJP exceeds an order of magnitude. We note that the capacity constraint is rarely breached for this particular set of parameters, and the resulting log sub-modular density without capacity constraints exhibit similar results as the constrained one, with just a slight decline in the relative performance of the Tabu sampler.

3.3.2.5 Log-Submodular Distributions II: Determinantal Point Processes

Similar to the facility location example above, we here consider sampling subsets S of elements of a pre-fixed set of items $[m] = \{1, 2, \ldots, m\}$ for some positive integer m. A determinantal point process (DPP), generically denoted $\mathcal{P}$, is a point process on $[m]$ taking values in $\mathcal{X} = 2^m$, we call each such subset a point configuration, see [KT12] for a very comprehensive overview. The defining feature of DPPs is that there is some real, positive semi-definite matrix K, denoted the marginal kernel, that captures a sense of 'closeness' or similarity of the different items. We then say a point process $\mathcal{P}$ is a DPP if whenever

$X \in \mathcal{X}$ is a random subset drawn from $\mathcal{P}$ and $S \subseteq \{1, 2, \ldots, m\}$,

$$\mathbf{P}(S \subseteq X) = \det K_S,$$

where $\det \cdot$ denotes the determinant, and K_S is the restriction matrix of K with corresponding rows and columns corresponding to the active indices. In this case, the probability of including any single entry $1 \leq j \leq m$ in X is just equal to the j^{th} diagonal entry of K, so in particular for any two indices i, j, we have

$$\mathbf{P}(\{i, j\} \subseteq X) = \det K_{\{i,j\}} = K_{i,i}K_{j,j} - K_{i,j}^2,$$

such that the probability is decreasing the closer the two points are in terms of the marginal kernel. By this property, DPPs explicitly model repulsion of points. A DPP is also an example of a log-submodular probability distribution, as the determinant is always increasing in the number of points. For our purposes, it is easier to work with an explicit form of DPPs known as L-ensembles. Let L be the real $M \times M$ symmetric matrix, such that

$$K = L(L + I)^{-1} \Rightarrow L = K(I - K)^{-1},$$

whenever the left-hand side exists. In this case, letting $\mathbf{X} \sim \mathcal{P}$ be the random variable associated with the DPP, the probability of selecting *exactly* the set $X = \{x_1, \ldots, x_K\}$ is

$$\mathbf{P}(\mathbf{X} = X) \propto \det L_X,$$

where L_X is the $K \times K$ matrix with entries given by $(L_X)_{i,j} = L_{x_i,x_j}$ for $1 \leqslant i, j \leqslant K$. Furthermore, the normalization constant is given by $Z(L) = \det(L + I)$. An interesting property is the following: let $|\mathbf{X}|$ be the cardinality of $\mathbf{X}$, and write $(\lambda_i)_{1 \leqslant i \leqslant M}$ for the eigenvalues of L. The law of $|\mathbf{X}|$ is then that of the number of successes in M independent Bernoulli trials with success probabilities given by $p_i = \lambda_i/(\lambda_i + 1)$. In particular, it follows that $\mathbf{E}[|\mathbf{X}|] = \sum_{i=1}^{M} \lambda_i/(\lambda_i + 1)$.

In this example we will consider $m = 500$, and for each item i we draw a uniformly distributed point s_i in the unit square. To calculate L, we apply the standard squared exponential kernel $L(i, j) = \exp\{-\frac{1}{2}\|s_i - s_j\|^2\}$. The expected number of points drawn from $\mathcal{P}_L$ is very close to 60 for any random selection of points under this model. We are interested in how many points are being drawn, and if a varied selection of points is being chosen, so similarly to the above example, our target statistic is the number of active points at any time, and the samplers are initiated at $S_0 = \{1, 2, \ldots, m\}$, e.g. every point is active. In this case both the Tabu sampler and the LBJP mix well across the posterior, and the

ESS/s of the log-energy are close, with a slight edge to the Tabu sampler, corresponding to an improvement in efficiency of about 50%. However, when it comes to the generation of diverse point clouds, measured by the number of points chosen, the directed movement of the Tabu sampler leads to the sampler being around 6 times more efficient. In practice, the LBJP is primarily modifying outliers with high impact, in probability, on the overall cloud, while the Tabu sampler uses its irreversible behaviour to more effectively change the composition. Neither sampler ever proposes more than 80 or fewer than 40 points, and the mean excursion length is 11.

3.3.3 Examples with High-Order Generators

In the following sections we consider two examples of spaces where the order of the generators are infinite and finite, respectively. We compare our proposed dZZ and dCS with the LBJP.

3.3.3.1 Lattice Gaussians

Extensions of the Gaussian distribution to discrete observations and spaces have received some attention, with some examples including variants on Boltzmann machines in machine learning (as in e.g. [KCHK20, CK20]) and lattice Gaussians. The latter distribution has since the seminal paper [GPV08] been applied within cryptography schemes, in particular because these lattice-based encryption methods appear resilient to brute-force algorithms given even significant quantum computing power (see [Fol14] for extended discussion.) The hardness part of lattice-based encryption schemes is based on the difficulty of finding the shortest possible vector (the 'SVP' in encryption theory) given a lattice [HKR$^+$16]. Furthermore, sampling these distributions require at least 100 bit floating point precision, adding additional constraints to the implementation of the sampling algorithms. Outside of quantum cryptography, the lattice Gaussian distribution has also seen use in coding theory [LB14b] and spatial econometrics [Ans01]. Here we consider a simple toy case to illustrate the dynamics of the discrete Zig-Zag process, discrete coordinate sampler, and the LBJP on spaces with higher-order generators. Let $d > 1$ be a fixed integer, and consider a set of basis vectors $(\mathbf{v}_i)_{1 \leq i \leq d}$ in $\mathbb{R}^d$, which in general will be neither normalized nor mutually orthogonal. We then define a lattice $\mathbf{B}$ as the countably-infinite set of vectors

$$\mathbf{B} = \left\{ \sum_{i=1}^{d} \mathbf{v}_i z_i : \mathbf{z} = (z_1, \cdots, z_d) \in \mathbb{Z}^d \right\}$$

For simplicity, we denote by $B\mathbf{z} \in \mathbf{B}$ the point on the lattice given by the input vector $\mathbf{z}$. Given an arbitrary lattice $\mathbf{B}$, the centered lattice Gaussian has probability proportional to

$$\pi(\mathbf{z}) \propto \exp\left(-\frac{\pi\|B\mathbf{z}\|_2^2}{s^2}\right)$$

for some Gaussian parameter $s > 0$. For the distribution to be useful in cryptography, the variance parameter has to be moderately large; we take $s = 500$ in order to satisfy the requirements posited in the BLISS scheme of [DDLL13]. While the choice of lattice basis is fundamental in cryptographic applications, the extra cost of working on $\mathbf{B}$ is negligible. As such, we focus our attention on working with the natural basis of $\mathbb{Z}^d$.

To compare the methods, we initially let $d = 3$. In this case, it is clear from the algorithmic constructions that the runtime of the LBJP will be twice that of the dZZ, while the dZZ again will at least be thrice as slow as the dCS. To elucidate the difference, we first run the samplers until the algorithm time $T = 100,000$ and thin at intervals of $t = 1$, and subsequently run until the LBJP and dCS have run for an equivalent amount of wall-clock time compared to the LBJP (We re-run the experiments multiple times to verify the consistency in the comparison.) Our initialization point is $z_0 = (1000, 1000, 1000)$, which is far out in the tails of the target.

The results are shown in Figure 3.6 and 3.7. The random-walk like structure of the LBJP highlights the difficulty in designing good proposal mechanisms for the lattice Gaussian, since at any given point the change in probability at each neighbour is miniscule. The consequence is that the locally-balanced weighting function effectively loses its ability to distinguish meaningfully between neighbouring states, and the sampler begins to resemble a random walk Metropolis-Hastings sampler. We suspect that for heavy-tailed target distributions, the LBJP will run into this problem more dramatically.

In contrast to the LBJP, the irreversible dZZ and dCS mix quickly, as their persistent behaviour drags them towards the mode of the distribution. For this particular target, a naive implementation (in the sense of not exploiting known modes, symmetries etc. of the target) of the dCS runs 4.5 times quicker than the dZZ, however the ESS/s is of the same order.

3.3.3.2 Discrete Lattice Gauge Theories

In particle physics, *lattice gauge theory* (LGT) is a model of quantum field theory in which space is discretised onto a lattice. Calculating quantities of interest in lattice gauge theory involves evaluating high-dimensional integrals, and as such, Monte Carlo methods have been widely used in this area; see [Reb83] for an introduction. In the standard formulation, the state of an LGT model is a collection of 'gauge field' variables, indexed by the edges of the lattice, i.e. $x = \{x_{i,j}\}_{(i,j)\in E(L)}$, where the gauge fields $x_{i,j}$ take values in some Lie

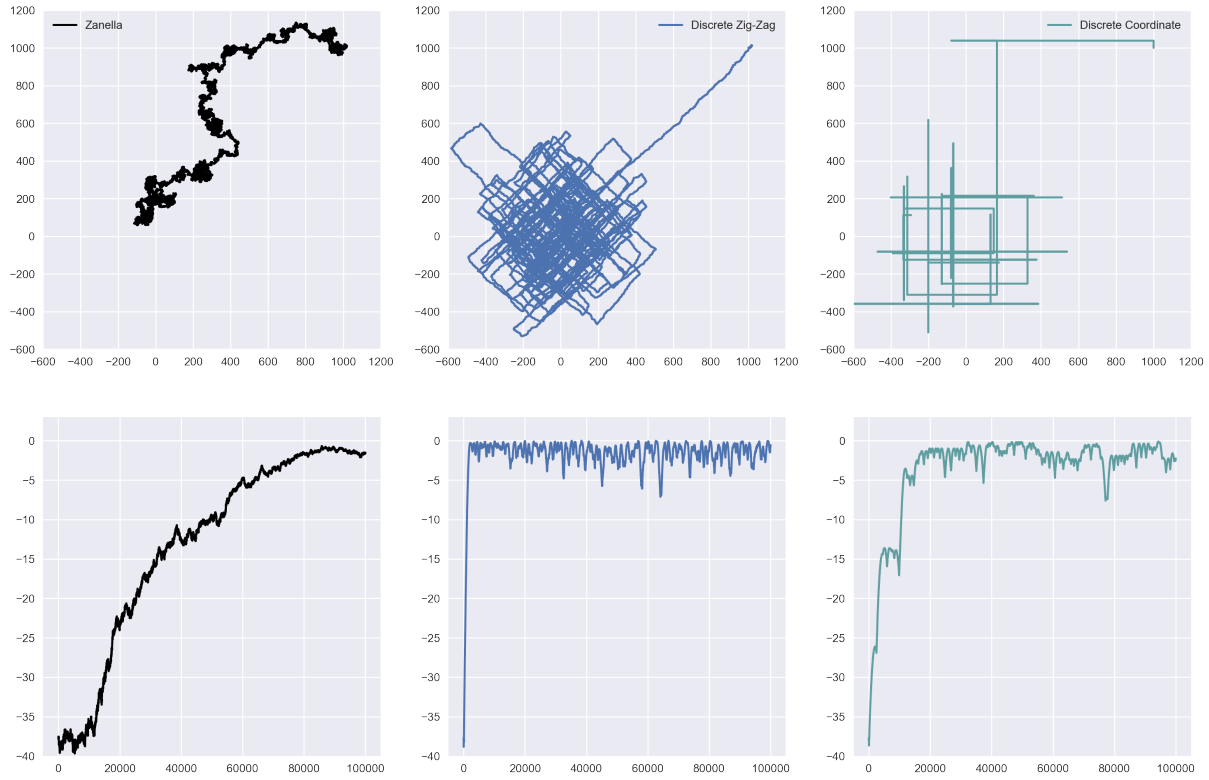

Figure 3.6: Sampling until internal time $T = 100,000$. Upper row: Position of the first and second coordinate of the 3-dimensional lattice Gaussian. Lower row: traceplot of the log-probability for the distribution.

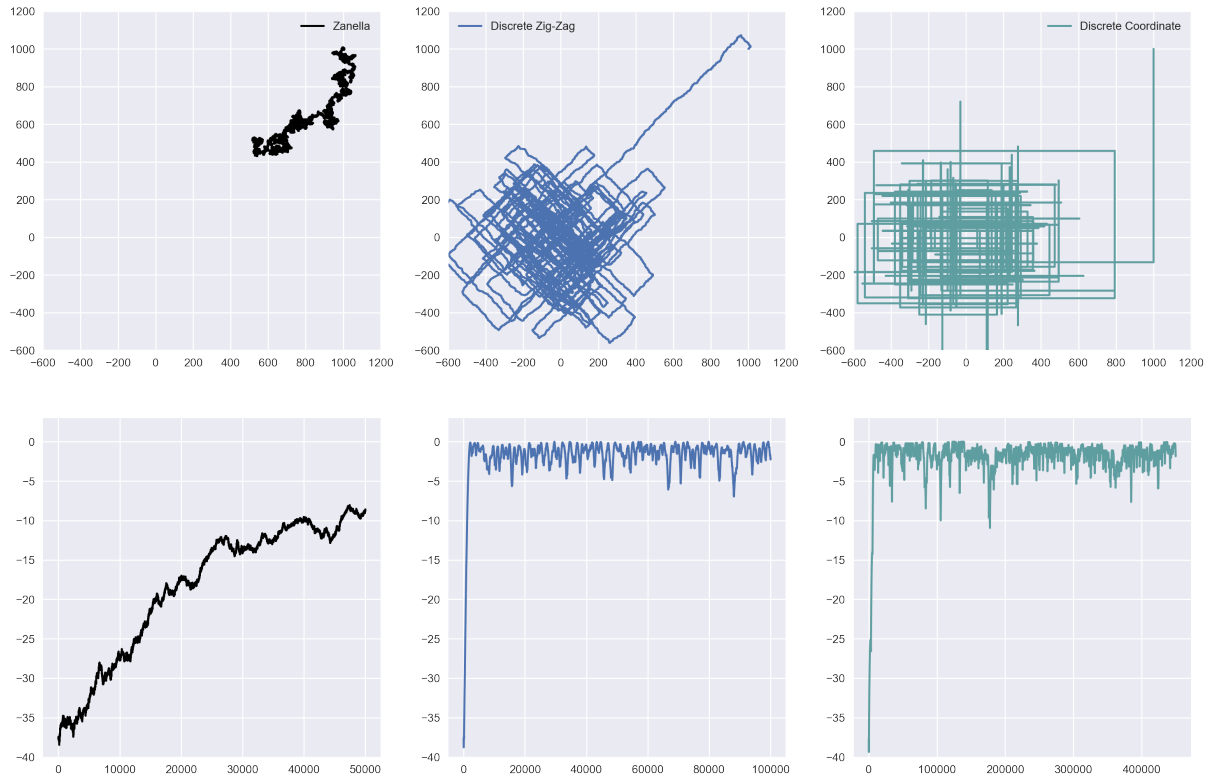

Figure 3.7: Sampling until wall-clock time is equivalent to the runtime to generate 100,000 samples from the discrete zig-zag sampler. Upper row: Position of the first and second coordinate of the 3-dimensional lattice Gaussian. Lower row: traceplot of the log-probability for the distribution.

group, e.g. the group of two-dimensional rotations, $SO(2) \cong \mathbb{S}^1$. It is occasionally of interest to go one step further, and also discretise *the group*; this gives rise to a *discrete lattice gauge theory*; see e.g. [Rom07]. We will apply our algorithms to a specific instance of a discrete LGT.

In particular, we focus on the setting where the lattice is a subset of $\mathbb{Z}^2$, the continuous group is $SO(2)$, and the discrete approximation is given by $\mathbb{Z}_p$, i.e. the integers $\mod p$. Our target distribution is defined as follows: for each 1×1 square P in the lattice, with vertices (i, j, k, l), define

$$V(P) = 1 - \cos\left(\frac{2\pi}{p}\left[x_{i,j} + x_{j,k} + x_{k,l} + x_{l,i}\right]\right).$$

For $\beta > 0$, we then define the target distribution π as

$$\pi(x) \propto \exp\left(-\beta \sum_{P \in L} V(P)\right).$$

Our state space is then $\mathbb{Z}_P^{E(L)}$, with generators given by 'increase $x_{i,j}$ by 1', for each edge (i, j). For large p, these generators have high order, and so our PDMP-type algorithms are appropriate.

For comparison, we let $\beta = 1$, $p = 53$ and consider a 4×4 subset of $\mathbb{Z}^2$. This gives us $P = 9$ squares and 24 vertices, for a total of 53^{24} possible sample combinations, illustrating the challenges the discrete lattice gauge model poses even in very low dimensions. We ran each sampler for an equivalent amount of wall-clock time (relative to generating 10^6 samples with the dCS), thinning approximately at each event of the sampler. Because of the dependencies across factors, the LBJP and dZZ in this case generate an equal amount of samples at the same runtime, while we note that the LBJP is thinning at twice the rate. To compare the resulting dynamics, we consider the following map of time and the first coordinate

$$(t, x_1) \mapsto \left(t, \cos\left(\frac{2\pi}{p}(x_1 \mod p)\right), \sin\left(\frac{2\pi}{p}(x_1 \mod p\right)\right),$$

and plot the resulting movement on the circle in the upper row of Figure 3.8. The dynamics are consistent with what is observed for the lattice Gaussian, with the LBJP experiencing difficulty in circumnavigating the whole circle, but the overall energy still mixes reasonably well. In comparison, the dZZ showcases good persistent behaviour at all times as it applies the same generators recurrently, while the dCS interestingly enough occasionally covers the entire circle more than once whenever the coordinate is active. Comparing our new schemes, the dZZ decorrelates much quicker than the dCS, but also requires access to all states accessible via a generator at each iteration. The resulting ESS/s is therefore still

slightly higher for the dCS compared to the dZZ, with approximately 13 and 10 times improvement on the LBJP, respectively, on average.

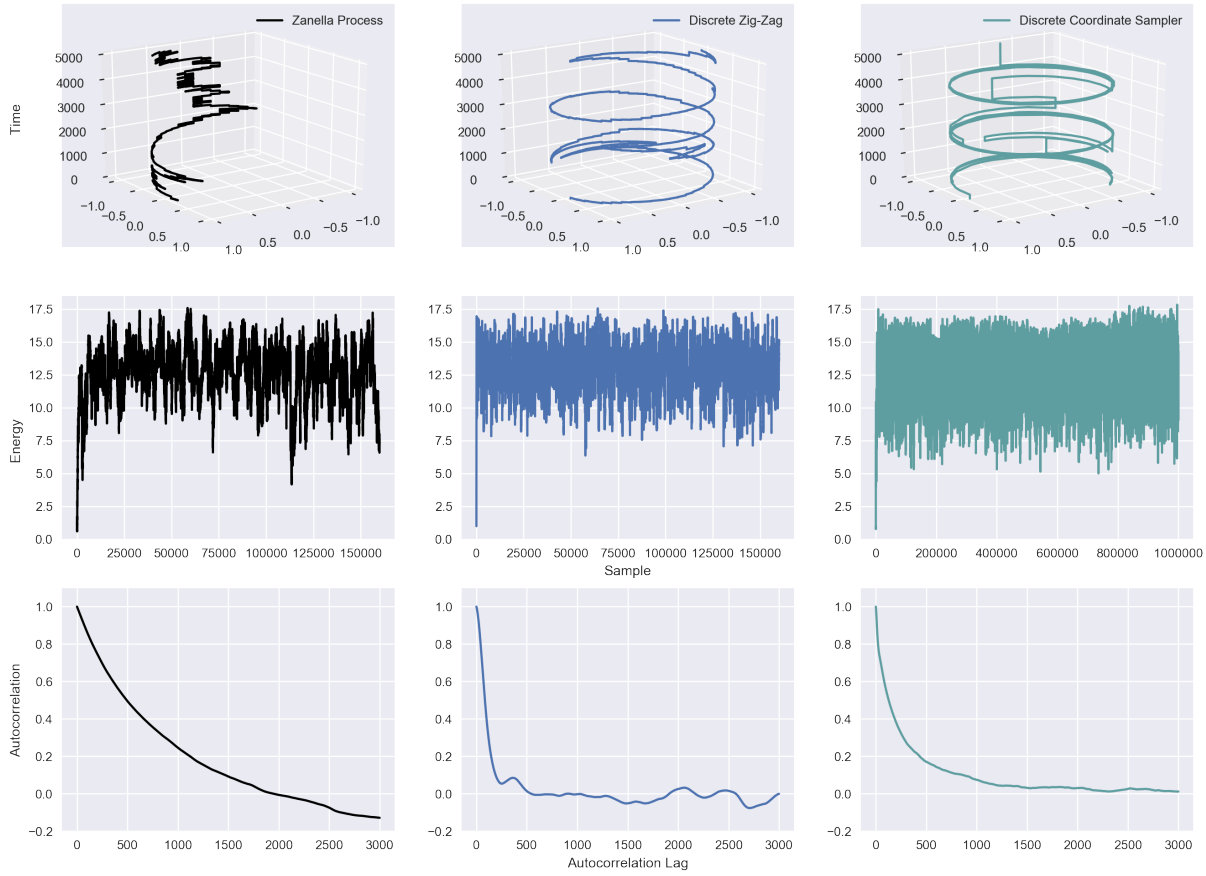

Figure 3.8: Sampling until wall-clock time corresponds to 10^6 samples from the dCS. Upper row: Position on the circle of the first coordinate up until $t = 5,000$. Middle row: traceplot of the log-probability of the distribution. Lower row: Autocorrelation function up to order $3,000$, calculated after discarding the first 20% of samples generated.

3.4 Conclusion and Outlook

In this work, we have presented a collection of algorithms which can be used for continuous-time Monte Carlo sampling on structured, discrete spaces. We have established correctness of these algorithms, and demonstrated their performance on a collection of examples. A particular benefit of these samplers is that they each directly use the structure of the underlying discrete space, and do not rely upon relaxations or embeddings into continuous spaces to achieve their improved performance.

Going forward, we highlight three directions in which this line of work could be advanced.

An important starting point would be to carry out a theoretical study of the quantitative convergence properties of these new samplers, and under which conditions we can expect

them to mix quickly. On continuous state spaces, one generally believes that log-concavity is sufficient for most sensible algorithms to mix quickly; on discrete spaces, it is not yet clear whether there is a similarly easily-verified condition which could ensure fast mixing of 'sensible' Markov chains on general state spaces (though see [Joh17] for some quite specific steps in this direction). On a related note, there is a wealth of existing work which makes use of group theory and representation theory to study the mixing behaviour of Markov chains, typically to uniform distributions. It would be exciting to see whether these techniques could provide useful answers in the setting of sampling from non-uniform distributions over structured spaces.

At a directly practical level, there is work to be done on how these algorithms can fit into the broader Monte Carlo toolbox. For example, we have used a naive thinning scheme to calculate our sample algorithms; one can imagine that a Rao-Blackwellisation procedure along the lines of [CR96, DR11] could provide some cheap variance reduction. One could also try to identify natural couplings for these processes, which could enable their use in the unbiased estimation schemes of [JOA20]. There is also ample opportunity to study how parallelism, adaptation schemes, and convergence diagnostics could play a role in further improving the performance of these algorithms.

Finally, we are excited to see how these techniques could give rise to improved algorithms for target distributions with a combination of discrete and continuous components. This could include statistical models with this flavour (e.g. mixture models, as in [RG97, Ste00, CRR03]), or 'artificially-discrete' sampling methods like Parallel Tempering (e.g. [ED05]). Recent work has shown that continuous-time sampling from continuous distributions can be done tractably and efficiently; it would be rewarding to connect those techniques with the methods in this paper.

3.5 Appendix

3.5.1 Proof Techniques

In this section, consider a Markov Jump Process on a discrete state space $\mathcal{U}$, with jump rates $\lambda(u \to v)$, and generator given by

$$(\mathcal{L}f)(u) = \sum_{v \in \partial u} \lambda(u \to v) \left[f(v) - f(u) \right].$$

For such processes, there are two standard routes to establishing invariance with respect to a given measure Π.

3.5.1.1 Detailed Balance

The first of these is *detailed balance*. An MJP is said to be in detailed balance with respect to Π if

$$\forall f, g \in L^2(\Pi), \quad \mathbf{E}_\Pi[(\mathcal{L}f)(u)g(u)] = \mathbf{E}_\Pi[f(u)(\mathcal{L}g)(u)].$$

It is standard to show that this condition is sufficient for Π to be an invariant measure for the MJP. By considering $f(x) = \mathbf{I}[x = u], g(x) = \mathbf{I}[x = v]$, one can show that it is necessary and sufficient that,

$$\forall u, v \in \mathcal{U}, \quad \Pi(u)\lambda(u \to v) = \Pi(v)\lambda(v \to u),$$

which can typically be verified by inspection.

3.5.1.2 Skew-Detailed Balance

The second such route is known as *skew-detailed balance*. In this case, the space $\mathcal{U}$ is equipped with a map $S : \mathcal{U} \to \mathcal{U}$, satisfying $S(S(u)) = u$ for all u. Defining the operator $\mathcal{Q}$ by

$$Qf(u) = f(S(u)),$$

the MJP is said to be in skew-detailed balance with respect to $(\Pi, \mathcal{Q})$ if

$$\forall f, g, \quad \mathbf{E}_\Pi[(\mathcal{Q}\mathcal{L}f)(u)g(u)] = \mathbf{E}_\Pi[f(u)(\mathcal{Q}\mathcal{L}g)(u)].$$

In what follows, we derive some assumptions under which skew-detailed balance will hold. In particular, we require following 'local conditions'

$$\forall u \in \mathcal{U}, \quad \Pi(S(u)) = \Pi(u),$$
$$\forall u, v \in \mathcal{U}, \quad \Pi(u)\lambda(u \to v) = \Pi(S(v))\lambda(S(v) \to S(u)).$$

Moreover, defining

$$\Lambda(u) = \sum_{v \in \partial u} \lambda(u \to v),$$

we require the additional 'semi-local condition'

$$\forall u \in \mathcal{U}, \quad \Lambda(u) = \Lambda(S(u)).$$

The local conditions are relatively standard, and crop up in e.g. the literature on non-reversible discrete-time Markov chains. However, the semi-local condition appears to be unique to the continuous-time setting, and may be of independent interest.

In any case, under these conditions, the MJP can be demonstrated to be skew-reversible, and thus leave Π invariant. Indeed, in the proofs which follow, skew-reversibility, rather than graph or group structure, will be the key ingredient.

We note that if an MJP satisfies the local conditions, but **not** the semi-local condition, it is possible to induce skew-detailed balance by including an extra jump type, as follows: at rate $[\Lambda(S(u)) - \Lambda(u)]_+$, jump from u to $S(u)$. Indeed, this is how we derived the algorithms presented in this paper.

3.5.2 Proofs of Invariance for Individual Algorithms

3.5.2.1 Proof for the LBJP

For the LBJP, one has that

$$\pi(x)\lambda(x \to y) = \pi(y)\lambda(y \to x),$$

by construction, as the jump rates are defined using balancing functions. As such, by detailed balance, the algorithm leaves π invariant.

3.5.2.2 Proof for the Tabu Sampler with Order-2 Generators

For the Tabu Sampler, we will show invariance by establishing that the process is skew-reversible with respect to the joint invariant measure $\Pi(x, \alpha, \tau)$, and the involution $S(x, \alpha, \tau) = (x, \alpha, -\tau)$. In particular, we claim that

$$\Pi(x, \alpha, \tau)\lambda((x, \alpha, \tau) \to (\gamma \cdot x, \alpha^\gamma, \tau)) = \Pi(\gamma \cdot x, \alpha^\gamma, -\tau)\lambda((\gamma \cdot x, \alpha^\gamma, -\tau) \to (x, \alpha, -\tau)).$$

As this transition leaves the probability mass functions of α and τ unchanged, we need only check that

$$\pi(x)\lambda((x, \alpha, \tau) \to (\gamma \cdot x, \alpha^\gamma, \tau)) = \pi(\gamma \cdot x)\lambda((\gamma \cdot x, \alpha^\gamma, -\tau) \to (x, \alpha, -\tau)).$$

Expanding the jump probabilities, this is equivalent to

$$\pi(x)g\left(\frac{\pi(\gamma \cdot x)}{\pi(x)}\right) \cdot \mathbf{I}[\alpha(\gamma) = \tau] = \pi(\gamma \cdot x)g\left(\frac{\pi(x)}{\pi(\gamma \cdot x)}\right) \cdot \mathbf{I}[\alpha^\gamma(\gamma) = -\tau].$$

The indicator functions in this expression are equal by construction, i.e. the move from (x, α, τ) to $(\gamma \cdot x, \alpha^\gamma, \tau)$ is allowed exactly when the move from $(\gamma \cdot x, \alpha^\gamma, -\tau)$ to $(x, \alpha, -\tau)$ is allowed. Moreover, because the jump rates are given in terms of balancing functions, one has that

$$\pi(x)g\left(\frac{\pi(\gamma \cdot x)}{\pi(x)}\right) = \pi(\gamma \cdot x)g\left(\frac{\pi(x)}{\pi(\gamma \cdot x)}\right).$$

As such, the two sides are genuinely equal. The additional jump rates for transitioning between (x, α, τ) and $S(x, \alpha, \tau) = (x, \alpha, -\tau)$ ensure that $\Lambda((x, \alpha, \tau)) = \Lambda(S(x, \alpha, \tau))$, and thus by skew-reversibility, the process leaves Π invariant.

3.5.2.3 Tabu Sampler on General Graphs

One can also define a Tabu Sampler on general graphs, using a similar construction to enforce that, at any given time, the process can only traverse each edge in one of the two directions.

More precisely, define an orientation ω to be any function $E \to V \times V$ which sends the edge (x, y) to either (x, y) or (y, x), i.e.

$$\text{for all } (x, y) \in E, \quad \omega(x, y) = (\omega_{-1}(x, y), \omega_{+1}(x, y)) \in \{(x, y), (y, x)\},$$

and write Ω for the space of all orientations. Now, construct a Markov process with state $(x, \omega, \tau) \in \mathcal{X} \times \Omega \times \{\pm 1\}$, such that at (x, ω, τ), the process can only jump to neighbours y such that $\omega_\tau(x, y) = x$. Upon making this jump, the values of $(\omega_{-1}(x, y), \omega_{+1}(x, y))$ are swapped. Moreover, at an appropriate rate, the value of τ is flipped to the negative of its previous value.

Specifically, one defines a process with local jump rates

$$\lambda\left((x, \omega, \tau) \to (y, \omega^{x \to y}, \tau)\right) = g\left(\frac{\pi(y)}{\pi(x)}\right) \cdot \mathbf{I}\left[\omega_\tau(x, y) = x\right]$$

and semi-local flipping rate

$$\lambda\left((x, \omega, \tau) \to (x, \omega, -\tau)\right) = \left[\sum_{y \in \partial x} g\left(\frac{\pi(y)}{\pi(x)}\right) \cdot \left\{\mathbf{I}\left[\omega_\tau(x, y) = x\right] - \mathbf{I}\left[\omega_{-\tau}(x, y) = x\right]\right\}\right]_+ .$$

An entirely analogous proof establishes that this process will have an invariant measure which is the product of $\pi(x)$, the uniform measure over orientations, and the uniform measure over τ.

In the process described above, there is an orientation assigned to each edge in the graph. In the original Tabu sampler, there is an orientation which is shared across all edges of the form $x \to \gamma \cdot x$. One can generalise this notion of shared edges further, to obtain various generalisations. This may be useful for models where there is not a group structure per se, but there is still a notion of edges being similar. Some similar ideas have been explored recently in [GM20].

3.5.2.4 Proof for the discrete Zig-Zag Process

For the discrete Zig-Zag process, we will show invariance by first establishing that when only one generator is available, the process leaves $\Pi(x, \theta)$ invariant. As the full discrete Zig-Zag process can be viewed as a superposition of the single-generator processes, the infinitesimal generator of the full process is simply the sum of the infinitesimal generators

of the single-generator processes, and one can thus deduce that the full process leaves Π invariant.

To estabish that the single-generator process leaves Π invariant, we will show that the process is skew-reversible with respect to the joint invariant measure $\Pi(x, \theta(\gamma))$ and the involution $S(x, \theta(\gamma)) = (x, -\theta(\gamma))$. This claim is equivalent to the equality

$$\Pi(x, \theta) \cdot \lambda((x, \theta) \to (y, \theta)) = \Pi(y, -\theta) \cdot \lambda((y, -\theta) \to (x, -\theta))$$

where $y = \gamma^{\theta(\gamma)} \cdot x, \theta = \theta(\gamma)$. As in the case of the Tabu sampler, the transition leaves the probability mass function of θ unchanged, and so we need only check that

$$\pi(x) \cdot g\left(\frac{\pi(y)}{\pi(x)}\right) \cdot \mathbf{I}[x \to y \text{ allowed by } \theta] = \pi(y) \cdot g\left(\frac{\pi(x)}{\pi(y)}\right) \cdot \mathbf{I}[y \to x \text{ allowed by } -\theta].$$

Again, the indicator functions are equal by construction, and the use of balancing functions guarantees that

$$\pi(x) \cdot g\left(\frac{\pi(y)}{\pi(x)}\right) = \pi(y) \cdot g\left(\frac{\pi(x)}{\pi(y)}\right).$$

As such, the two sides are genuinely equal. The additional jump rates for transitioning between (x, θ) and $S(x, \theta) = (x, -\theta)$ ensure that $\Lambda((x, \theta)) = \Lambda(S(x, \theta))$, and thus by skew-reversibility, the process leaves Π invariant.

Now, let $\mathcal{L}^\gamma$ be the generator of the discrete Zig-Zag process which only uses the generator γ. The generator of the full process is then given by $\mathcal{L} = \sum_{\gamma \in \Gamma_0} \mathcal{L}^\gamma$. Since each of these generators is associated to a process which leaves Π invariant, we see that for any $f \in L^2(\Pi)$,

$$\begin{aligned}
&\mathbf{E}_\Pi\left[(\mathcal{L}^\gamma f)(x)\right] = 0 \quad \text{for all } \gamma \in \Gamma_0 \\
\implies &\sum_{\gamma \in \Gamma_0} \mathbf{E}_\Pi\left[(\mathcal{L}^\gamma f)(x)\right] = 0 \\
\implies &\mathbf{E}_\Pi\left[\sum_{\gamma \in \Gamma_0} (\mathcal{L}^\gamma f)(x)\right] = 0 \\
\implies &\mathbf{E}_\Pi\left[(\mathcal{L} f)(x)\right] = 0
\end{aligned}$$

and thus that the full process also leaves Π invariant.

Note that a simple adaptation of this proof allows for the design a correct variant of the discrete Zig-Zag process which prefers to use some generators more than others, by constructing a process with generator $\mathcal{L}_w = \sum_{\gamma \in \Gamma_0} w(\gamma) \mathcal{L}^\gamma$ for some set of positive weights w. There may be scope for using such a process with an adaptive choice of w to accelerate sampling.

3.5.2.5 Proof for the discrete Coordinate Sampler

For the discrete Coordinate Sampler, we will show invariance by a direct computation, as the mechanism for changing the velocity is somewhat more elaborate than in the other algorithms. We begin by writing down the generator of the process:

$$\begin{aligned}(\mathcal{L}f)(x) = \delta(x,v,\tau)\cdot[f(v^\tau\cdot x,v,\tau)-f(x,v,\tau)]\\ +\rho(x,v,\tau)\cdot\sum_{w\in\Gamma}\frac{\psi(w)\rho(x,w,\tau)}{Z(x)}\left[f(x,w,-\tau)-f(x,v,\tau)\right].\end{aligned}$$

We begin by computing the expectation of the first part of the first term:

$$\begin{aligned}&\mathbf{E}_\Pi\left[\delta(x,v,\tau)\cdot f(v^\tau\cdot x,v,\tau)\right]\\ &=\sum_{x,v,\tau}\pi(x)\psi(v)R(\tau)\delta(x,v,\tau)\cdot f(v^\tau\cdot x,v,\tau)\\ &=\sum_{x,v,\tau}\pi(x)\psi(v)R(\tau)\delta(x,v,\tau)\cdot f(v^\tau\cdot x,v,\tau)\\ &=\sum_{x,v,\tau}\pi(x)\psi(v)R(\tau)g\left(\frac{\pi(v^\tau\cdot x)}{\pi(x)}\right)\cdot f(v^\tau\cdot x,v,\tau)\\ &=\sum_{x,v,\tau}\pi(v^\tau\cdot x)\psi(v)R(\tau)g\left(\frac{\pi(x)}{\pi(v^\tau\cdot x)}\right)\cdot f(v^\tau\cdot x,v,\tau)\\ &=\sum_{y,v,\tau}\pi(y)\psi(v)R(\tau)g\left(\frac{\pi(v^{-\tau}\cdot y)}{\pi(y)}\right)\cdot f(y,v,\tau)\\ &=\sum_{y,v,\tau}\pi(y)\psi(v)R(\tau)\delta(y,v,-\tau)\cdot f(y,v,\tau)\\ &=\mathbf{E}_\Pi\left[\delta(x,v,-\tau)\cdot f(x,v,\tau)\right]\end{aligned}$$

We can thus deduce that

$$\mathbf{E}_\Pi\left[\delta(x,v,\tau)\cdot[f(v^\tau\cdot x,v,\tau)-f(x,v,\tau)]\right]=\mathbf{E}_\Pi\left[\{\delta(x,v,-\tau)-\delta(x,v,\tau)\}\cdot f(x,v,\tau)\right]$$

We now focus on computing the expectation of the first part of the second term:

$$\begin{aligned}
&\mathbf{E}_\Pi\left[\rho(x,v,\tau)\cdot\sum_{w\in\Gamma}\frac{\psi(w)\rho(x,w,\tau)}{Z(x)}f(x,w,-\tau)\right]\\
&=\sum_{x,v,\tau,w}\pi(x)\psi(v)R(\tau)\cdot\rho(x,v,\tau)\cdot\frac{\psi(w)\rho(x,w,\tau)}{Z(x)}f(x,w,-\tau)\\
&=\sum_{x,v,\sigma,w}\pi(x)\psi(w)R(\sigma)\cdot\rho(x,w,-\sigma)\cdot\frac{\psi(v)\rho(x,v,-\sigma)}{Z(x)}f(x,v,\sigma)\\
&=\sum_{x,v,\sigma,}\pi(x)R(\sigma)\psi(v)\rho(x,v,-\sigma)f(x,v,\sigma)\sum_{w}\frac{\psi(w)\rho(x,w,-\sigma)}{Z(x)}\\
&=\sum_{x,v,\sigma,}\pi(x)R(\sigma)\psi(v)\rho(x,v,-\sigma)f(x,v,\sigma)\\
&=\mathbf{E}_\Pi\left[\rho(x,v,-\tau)f(x,v,\tau)\right]
\end{aligned}$$

and thus that the second term has expectation

$$=\mathbf{E}_\Pi\left[\left\{\rho(x,v,-\tau)-\rho(x,v,\tau)\right\}f(x,v,\tau)\right].$$

Now, by noting that

$$\begin{aligned}
\rho(x,v,-\tau)-\rho(x,v,\tau)&=\left[\delta(x,v,\tau)-\delta(x,v,-\tau)\right]_+-\left[\delta(x,v,-\tau)-\delta(x,v,\tau)\right]_+\\
&=\delta(x,v,\tau)-\delta(x,v,-\tau),
\end{aligned}$$

we can add together the two expectation terms, and see that $\mathbf{E}_\Pi\left[(\mathcal{L}f)(x,v,\tau)\right]=0$. Thus, Π is an invariant measure for the chain.

Chapter 4

An Optimal Scaling Theory for a Multi-Core Metropolis-Hastings Algorithm

Abstract

In this work, we present an implementation of the Metropolis-Hastings algorithm which makes use of parallel computing resources. The implementation applies naturally to any family of proposals, including those based on Random Walks, Langevin Diffusions, and Hamiltonian Dynamics. An attractive aspect of this use of parallelism is that it permits larger step-sizes to be used in these proposals, while maintaining the same real-time acceptance rate. Using this perspective, we formulate an optimal scaling problem in the vein of [RGG97], which, for a given family of proposals and a given number of processors P, provides a value to which the acceptance rate of the resulting Markov chain should be tuned. This problem can be efficiently solved for any fixed P and one can numerically estimate the expected gain in efficiency over the corresponding single-processor algorithm. Moreover, we carry out an asymptotic analysis, which establishes how this optimal efficiency gain scales with the number of processors P.

4.1 Introduction

Markov Chain Monte Carlo (MCMC) is a class of algorithms which are used for approximate sampling from a desired probability distribution π. The Metropolis-Hastings Algorithm is a general, widely-applied framework within MCMC, which allows one to

devise Markov Chains with a desired invariant measure π. In a Metropolis-Hastings chain, one stochastically *proposes* moves according to some 'proposal kernel' $q(x \to y)$, and then stochastically *accepts or rejects* these moves, according to a criterion for how suitable the transition from x to y is, under the desired invariant measure π. The proposal kernel q typically depends explicitly on some step-size parameter h, which informs the typical length-scale of the proposal moves. With such algorithms, there is a fundamental tension. Proposing moves which are too large will typically lead to low acceptance rates, and a chain which rarely moves. On the other hand, proposing moves which are too small will lead to a chain which requires many steps to move a given distance. As such, it is of clear practical interest to devise optimality criteria which allow users to navigate this tradeoff and automatically tune their proposal mechanisms to optimise the efficiency of the resulting algorithm.

One relatively successful criterion has come from the theory of *optimal scaling*. In this framework, one first considers deploying a Metropolis-Hastings chain to sample from a target measure of product form, $\Pi^N(x^1, \cdots, x^N) = \pi(x^1) \cdots \pi(x^N)$. One then identifies a scaling of $h = h(N)$ such that the acceptance probability of the N-dimensional algorithm remains of constant order as N grows. Typically, upon an appropriate rescaling of time, one can show that the path of the first component x^1 converges weakly to some stochastic process in the large-N limit. By analysing the behaviour of the limit process, one can devise a protocol for tuning the parameters of the chain, such that the speed of convergence to equilibrium is optimised. This form of analysis has been highly influential, and has been a significant driver in the development of *adaptive* MCMC methods, where the parameters of a proposal are learned on-the-fly and tuned to their optimal setting automatically as the chain runs.

A separate consideration which has been of recurrent interest in the study of MCMC algorithms is the role of parallel computing. MCMC is naturally viewed as a sequential process, and as such, it has not always been clear how parallel computing resources could be most usefully deployed in improving sampling efficiency. One simple way of employing parallel resources in a Metropolis-Hastings algorithm would be to use multiple cores to propose and evaluate several possible moves in parallel, ultimately moving to the first accepted proposal. This approach could be used with a view towards either i) boosting the real-time acceptance rate of the algorithm, or ii) making more ambitious moves and increasing the upper limit of feasible step-sizes.

In this work, we will study this 'Multi-Core Metropolis-Hastings' algorithm and verify this behaviour, i.e. that given $P > 1$ cores, one should generally increase the proposal step-size h, thus aiming for a lower acceptance rate for each proposal, while achieving a higher *real-time* acceptance rate. Moreover, given a class of proposal kernels, we will derive explicit numerical expressions for the optimal acceptance rate of the resulting algorithm for

any finite P. Finally, we characterise the asymptotic behaviour of the optimised algorithm as P tends to infinity.

The layout of the paper is as follows. In Section 4.2, we describe the classical theory of optimal scaling for Metropolis-Hastings algorithms. In Section 4.3, we give a precise formulation of our simple Multi-Core Metropolis-Hastings algorithm. In Section 4.4, we describe how the efficiency measures used in classical optimal scaling should be adapted to the multi-core setting and explain how to optimise them. In Section 4.5, we carry out an asymptotic analysis, giving indications of what range of benefits can be expected as the number of processors P tends to infinity. Finally, in Section 4.6 we comment on the outlook for parallelism in MCMC, and discuss other ways in which parallel resources could be deployed practically for improved MCMC sampling.

4.2 Optimal Scaling Theory

The majority of the theory of optimal scaling for MCMC is carried out in the Metropolis-Hastings framework, and as such, we present it from this perspective.

4.2.1 Metropolis-Hastings Algorithms

We begin by describing the construction of the Metropolis-Hastings algorithm. We focus on the task of sampling from a distribution on $\mathbf{R}^d$, with full support and a smooth, positive density with respect to Lebesgue measure. We assume also that all proposal distributions have full support and a smooth, positive density. We will also routinely abuse notation and identify distributions with their densities. Additional settings can certainly be handled by the framework, but these assumptions simplify the presentation considerably, without particularly obscuring the main concepts.

Suppose that we are interested in sampling from a distribution $\pi(x)$, whose density we can evaluate up to a constant factor. Suppose also that for each $x \in \mathbf{R}^d$, there is a *proposal distribution* $q(x \to y)$, from which we are able to draw samples, and whose density we can evaluate exactly. The Metropolis-Hastings algorithm for sampling from π, with proposal q, proceeds as follows:

The key fact about this algorithm is that it generates sequence of points x which form a Markov chain with π as an invariant measure. Moreover, under reasonable assumptions on π and q, as the number of iterations tends to infinity, the distribution of x will converge to π, and under further assumptions, one can also prove that this convergence is sufficiently fast that certain Law of Large Numbers- and Central Limit Theorem-type results hold, i.e. that the samples generated by the algorithm can be used to efficiently estimate expectations of functions under π.

Algorithm 34 Metropolis-Hastings algorithm for sampling from π, with proposal q

1. At location x, propose a move to y, where $y \sim q(x \to y)$.
2. Evaluate the *acceptance ratio*, given by

$$\alpha(x \to y) = g\left(\frac{\pi(y)q(y \to x)}{\pi(x)q(x \to y)}\right)$$

   where $g(t) = \min(1, t)$.
3. Simulate $u \sim \text{Unif}[0, 1]$, and set $b = \mathbf{I}[u < \alpha]$.
4. If $b = 1$, move to y. Otherwise, remain at x.

In this work, we will focus our attention on three of the most commonly studied families of proposal distributions. They are the *Random Walk* proposal,

$$q^{\text{RW}}(x \to y; h) = \mathcal{N}(y|x, hI),$$

the *Langevin diffusion* proposal,

$$q^{\text{LD}}(x \to y; h) = \mathcal{N}\left(y|x + \frac{1}{2}h\nabla \log \pi(x), hI\right),$$

and the *Hamiltonian dynamics* proposal,

$$q^{\text{HD}}((x, p) \to (y, q); h, T) = \delta(\mathcal{F} \circ \mathcal{L}((x, p); h, T), (dy, dq)).$$

In the definition of q^{HD}, we define $\mathcal{L}((x, p); h, T)$ as the output of running the leapfrog integrator to solve Hamilton's equations of dynamics for T units of time, using time-steps of size h, starting at the location (x, p). T is typically referred to as the *integration time*. Note that Th^{-1} is implicitly an integer in this formulation. We will later use the notation $\mathcal{L}((x, p); T)$ to denote the *exact* solution to Hamilton's equations, such that for any (x, p, T), as $h \to 0^+$, we have that $\mathcal{L}((x, p); h, T) \to \mathcal{L}((x, p); T)$. Finally, we define $\mathcal{F}$ by $\mathcal{F}(x, p) = (x, -p)$.

In all cases, $h > 0$ is a step-size parameter, which dictates the scale of the proposal moves which are being made. When put into the Metropolis-Hastings framework, the resulting algorithms are known as *Random Walk Metropolis-Hastings* (RWMH), the *Metropolis-Adjusted Langevin Algorithm* (MALA), and *Hamiltonian Monte Carlo* (HMC) respectively. We note that in the case of Hamiltonian Monte Carlo, there are some subtle differences. For one, instead of trying to draw samples from $\pi(x)$, one tries to draw samples from an *augmented target distribution*, given as $\mu(x, p) = \pi(x) \cdot \mathcal{N}(p|0, I)$, which admits π as a

marginal distribution. Moreover, the algorithm in fact alternates between two steps:

1. Propose a move from (x, p) to (y, q) using q^{HD}, accept or reject that move using the Metropolis-Hastings test.

2. Jump from (x, p) to (x, p'), where p' is drawn independently from $\mathcal{N}(p|0, I)$.

While significant at the level of implementation, these differences will not impede the mathematical claims which follow. We will not dwell much further on the more esoteric details of how HMC differs from the other proposals; for the interested reader, [Bet17] provides a thorough review.

A natural question for each of these proposals concerns how this parameter h ought to be set. On one hand, for small h, the acceptance ratio α will tend to be close to 1, and more moves will be accepted. On the other hand, for larger h, the moves which are accepted will generally be larger in magnitude, relative to their cost of simulation. Striking a balance between accepting a reasonable proportion of moves, and having those moves be of a reasonable size, is at the core of defining efficient Metropolis-Hastings schemes.

4.2.2 Overview on Optimal Scaling

The optimal scaling approach to tuning step-sizes can roughly be split into three phases of calculations.

1. The 'critical scaling' calculations, in which one determines the rate at which the step-size should scale for high-dimensional problems.

2. The 'continuum limit' calculations, in which one demonstrates that, when using the critical scaling to set step-size, an appropriately-rescaled version of the high-dimensional Metropolis-Hastings chain converges to a certain continuous-time stochastic process.

3. The 'optimal acceptance' calculations, in which one uses the continuum limit of the chain to recommend a target average acceptance rate for the Metropolis-Hastings chain.

For the critical scaling phase, one first fixes a family of proposals $\{q(x \to y; h)\}_{x \in \mathcal{X}, h>0}$, a target distribution π, and an integer N. One then writes down both the target measure

$$\Pi^N(x^1, \cdots, x^N) = \pi(x^1) \cdots \pi(x^N),$$

and the Metropolis-Hastings algorithm for sampling from Π^N, using the proposals $\{q(x \to y; h)\}_{x \in \mathcal{X}}$, for some $h > 0$.

Next, assuming that the Metropolis-Hastings chain is initialised at stationarity (though note that some works relax this assumption at the cost of a more involved analysis, see

e.g. [CRR05, JLM14, JLM15]), one writes down an expression for the acceptance rate $\alpha^N = \alpha(x \to y)$. Given this expression, one can typically identify a constant $\gamma > 0$ such that, by scaling $h = h_0 \cdot N^{-\gamma}$, and sending $N \to \infty$, the acceptance rate α^N tends to a deterministic limit of constant order, i.e.

$$\alpha^N(x \to y) \to A(h_0; \pi) \in (0, 1).$$

For example,

1. For the Random Walk proposal, it was shown in [RGG97] that one can take $\gamma = 1$, and the limiting acceptance probability is then given by

   $$A^{\mathrm{RW}}(h_0; \pi) = 2\Phi\left(-c_\pi^{\mathrm{RW}} \cdot h_0^{1/2}\right)$$

   where Φ is the CDF of a standard Gaussian random variable, and c_π^{RW} is a constant which depends only on the target distribution π.

2. For the Langevin Diffusion proposal, it was shown in [RR98] one can take $\gamma = 1/3$, and the limiting acceptance probability is then given by

   $$A^{\mathrm{LD}}(h_0; \pi) = 2\Phi\left(-c_\pi^{\mathrm{LD}} \cdot h_0^{3/2}\right)$$

   where c_π^{LD} is a constant which depends only on the target distribution π.

3. For the Hamiltonian Dynamics proposal, one nominally has two parameters to select, h and T. The assumption used in [BPR$^+$13] is to treat T as fixed, and set $h = T \cdot \lfloor N^{1/4} \rfloor^{-1}$. This is asymptotically equivalent to taking $\gamma = 1/4$, and we will drop the floor symbols from here onwards. With these choices, the authors then show that the limiting acceptance probability is given by

   $$A^{\mathrm{HD}}(h_0; \pi) = 2\Phi\left(-c_{\pi,T}^{\mathrm{HD}} \cdot h_0^2\right)$$

   where $c_{\pi,T}^{\mathrm{HD}}$ is a constant which depends only on the target distribution π and the integration time T.

One can also derive expressions of this form for a variety of other proposal distributions, see e.g. [ARR09, BDM12, BDM14] for some other examples.

For the continuum limit calculations, one then proceeds by studying the N-dimensional process, with step-size scaling as above, and aims to show that, when rescaled appropriately, the *marginal* path law of the **first** coordinate converges to an appropriate stochastic process.

For example,

1. For the Random Walk proposal, upon rescaling time by a factor of N, one obtains the continuous-time Langevin diffusion

$$dX = \frac{1}{2} S^{\mathrm{RW}}(h_0; \pi) \nabla \log \pi(X) dt + \sqrt{S^{\mathrm{RW}}(h_0; \pi)} dW$$

where the *speed measure* S^{RW} is defined as

$$S^{\mathrm{RW}}(h_0; \pi) = h_0 \cdot A^{\mathrm{RW}}(h_0; \pi).$$

This limiting process is equivalent to taking the diffusion $dX = \frac{1}{2} \nabla \log \pi(X) dt + dW$ and then rescaling time by a factor of $S^{\mathrm{RW}}(h_0; \pi)$.

2. For the Langevin Diffusion proposal, one instead rescales time by a factor of $N^{1/3}$ and obtains the same diffusion

$$dX = \frac{1}{2} S^{\mathrm{LD}}(h_0; \pi) \nabla \log \pi(X) dt + \sqrt{S^{\mathrm{LD}}(h_0; \pi)} dW,$$

albeit with time rescaled by a different factor, given by

$$S^{\mathrm{LD}}(h_0; \pi) = h_0 \cdot A^{\mathrm{LD}}(h_0; \pi).$$

3. For the Hamiltonian Dynamics proposal, no time rescaling occurs. Instead, the limiting process is a Markov chain, defined by iterating the following two steps:

   (a) Resample $p \sim \mathcal{N}(0, 1)$.

   (b) Compute $(y, q) = \mathcal{L}((x, p); T)$, and with probability $A^{\mathrm{HD}}(h_0; \pi)$, move to (y, q), otherwise, remain at (x, p).

Given such a limiting process, one proceeds to the optimal acceptance calculations. In this phase, one reasons about how h_0 can be tuned, such that the corresponding limiting stochastic process is mixing as quickly as possible, per unit of work.

For both the Random Walk and Langevin Diffusion proposals, the limiting stochastic process is a time-rescaled version of the continuous-time Langevin diffusion. Varying h_0 only affects the speed at which the diffusion travels, and does not affect the paths it traces out. As such, we are justified in trying to directly optimise the factor by which time is rescaled, i.e. the speed measure $S(h_0; \pi)$.

By contrast, for the Hamiltonian dynamics proposal, the variability in the limiting process comes from the cost of simulating each step of the algorithm. With step-size $h \sim h_0 N^{-1/4}$, each step of the algorithm costs $\sim h^{-1}$ to simulate, and is then accepted with probability $A^{\mathrm{HD}}(h_0; \pi)$. With this in mind, the authors propose maximising the ratio

of the acceptance probability to the cost per step, i.e. they define a speed measure as $S^{\mathrm{HD}}(h_0;\pi) = h_0 \cdot A^{\mathrm{HD}}(h_0;\pi)$.

We summarise the key examples as follows:

Proposal	**Scaling (γ)**	**Acceptance Rate**	**Speed Measure**
Random Walk	1	$2\Phi\left(-c_\pi^{\mathrm{RW}} \cdot h_0^{1/2}\right)$	$h_0 \cdot A^{\mathrm{RW}}(h_0;\pi)$
Langevin Diffusion	1/3	$2\Phi\left(-c_\pi^{\mathrm{LD}} \cdot h_0^{3/2}\right)$	$h_0 \cdot A^{\mathrm{LD}}(h_0;\pi)$
Hamiltonian Dynamics	1/4	$2\Phi\left(-c_\pi^{\mathrm{HD}} \cdot h_0^{2}\right)$	$h_0 \cdot A^{\mathrm{HD}}(h_0;\pi)$

We note that it is common to reparametrise $h_0 = L^\delta$ for some 'dynamical exponent' $\delta > 0$ depending on the family of distributions, where L is defined such that a typical proposal move is at a distance $\sim L$ from the previous point. For example, for diffusive proposals like the Random Walk and Langevin Diffusion, one can take $\delta = 2$, and for ballistic proposals like Hamiltonian Dynamics, one instead takes $\delta = 1$. We will adopt this convention going forward, and abbreviate $\alpha(L) = A(L^\delta;\pi), s(L) = S(L^\delta;\pi)$, leaving the dependence on π implicit.

In all of our examples, it is possible to write

$$\begin{aligned}\alpha(L) &= 2\Phi\left(-c_\pi \cdot L^{\delta/2\gamma}\right)\\ s(L) &= L^\delta \cdot \alpha(L).\end{aligned}$$

When this is the case, one can then rewrite the speed measure in terms of α as

$$s(\alpha) = c_\pi^{-2\gamma} \cdot \left(\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right)^{2\gamma} \cdot \alpha.$$

A consequence of this expression for the speed measure in terms of α is that although the optimal value of L will generally depend on the properties of π, the optimal acceptance rate α_* is *independent* of the target measure. As a result, one can tune the step-size of a Metropolis-Hastings algorithm purely by monitoring the empirical acceptance rate, and adjusting the step-size accordingly. For more concrete guidance on algorithms with which to carry out this adaptation, see e.g. [AT08].

4.3 Multi-Core Metropolis-Hastings

When running a Metropolis-Hastings algorithm in practice, one effectively only needs to store i) the distinct locations which the chain has visited, and ii) for how many units of time the chain stayed at each of these locations. As such, with other things being equal, it is desirable to accelerate the procedure of finding new, distinct locations to move to, provided that the mixing behaviour of the chain is not compromised.

One scenario under which this can be achieved is when one has access to parallel computing resources. While on a single-core machine, one has to repeat the propose-accept-reject procedure *sequentially* until a new accepted location is found, on multi-core machines, one can propose and evaluate many potential new states at once *in parallel*. In real time, this allows for potential speed-ups. We present a simple version of such an algorithm below, noting that [SWJ93] present a similar algorithm in the context of simulated annealing.

Algorithm 35 Multi-Core Metropolis-Hastings, with P processors

1. At location x, for $p = 1, \ldots, P$, in parallel,
   (a) Simulate $y^p \sim q(x \to y)$.
   (b) Evaluate the *acceptance ratio*, given by
   $$\alpha^p = \alpha(x \to y^p) = g\left(\frac{\pi(y^p)q(y^p \to x)}{\pi(x)q(x \to y^p)}\right)$$
   where $g(t) = \min(1, t)$.
   (c) Simulate $u^p \sim \text{Unif}[0, 1]$, and set $b^p = \mathbf{I}[u^p < \alpha^p]$.
2. If $b^p = 0$ for all p, advance time by P, and stay at x
3. Otherwise, let $P^* = \min\{p : b^p = 1\}$, advance time by P^*, and move to y^{P^*}.

It is important to note that this algorithm generates *the same Markov chain* as the standard Metropolis-Hastings algorithm, and as such, will also converge in law to π. The differences arise through i) implementation, and ii) the *real-time* convergence of the algorithm.

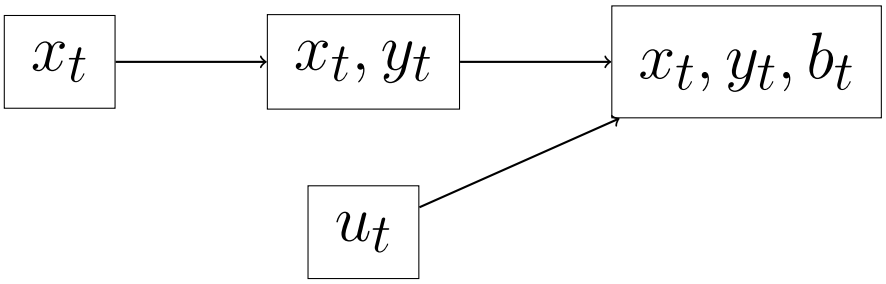

Figure 4.1: Schematic for Standard Metropolis-Hastings Implementation

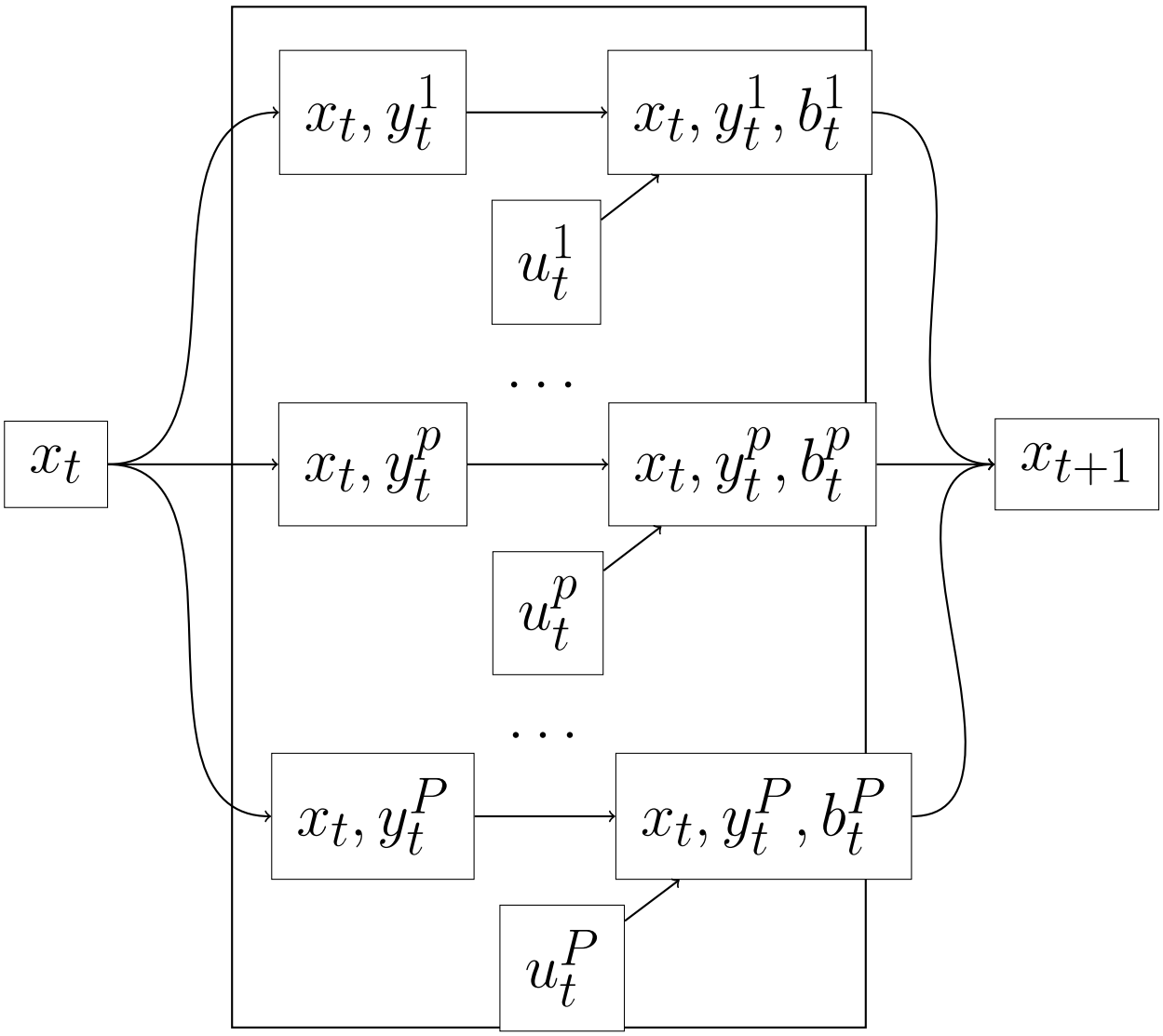

Figure 4.2: Schematic for Multi-Core Metropolis-Hastings Implementation

A key observation about this new scenario is that the standard optimality criteria may no longer be appropriate. In particular, if one is interested in how quickly the chain decorrelates *in real time*, it seems reasonable that for an efficient scheme, one only really needs 1 of the P proposals to be accepted at each step. This suggests that one ought to make more ambitious proposal moves when there are multiple cores available.

4.3.1 Other Parallelisations of MCMC Algorithms

We pause briefly to review other work which uses parallelisation to accelerate MCMC procedures. We restrict ourselves to procedures which ultimately generate a *single* Markov chain; procedures involving multiple chains have a qualitatively different nature, and we omit them from our discussion.

One approach which has been studied in some depth is 'prefetching' as introduced in

[Bro06]; note also related work on 'speculative evaluation' of proposals in [BJB10]. In prefetching, the goal is to simulate the Markov chain forward k steps in time and use parallelisation to enable this. The key observation underlying prefetching is that over a k-step horizon, there are 2^k possible sequences of accept-reject decisions which can unfold, which form a tree structure, and one can assess which of these decisions should ultimately take place in parallel. Subsequent work [Str10, AKW$^+$14] has focused on efficient ways of pruning this tree structure, allowing for attention to be focused on more promising sequences of proposals. These methods can be highly efficient, but tend to assume that the proposal family is sufficiently simple that multi-step proposals can be generated easily; this impedes the use of more advanced proposals like discretisations of Langevin diffusions and Hamiltonian dynamics.

To elaborate, the common assumptions underlying prefetching methods tend to be that i) the key cost of any MCMC step is likelihood evaluation, ii) proposing moves is considered to be relatively cheap, and iii) proposing moves several steps ahead is not much more expensive than proposing one. The first of these is quite reasonable, and the latter two are holds for random walk proposals. However, for gradient-based proposals, the cost of evaluating the gradient of the log-likelihood is typically comparable to the cost of evaluating the likelihood, and so ii) is somewhat less true. The third point is even less true; understanding where the chain could end up after k steps requires considering at least k gradient evaluations. As such, these prefetching schemes are not as well-adapted to scenarios in which gradient-based proposals are available.

One can also use parallelisation to accelerate MCMC algorithms in which individual steps have random runtimes, e.g. [NMA14] employ parallelism to speed up Elliptical Slice Sampling, by accelerating the search for a valid point in the 'slice'. It is also common practice to speed up individual steps of a blocked Gibbs sampler by updating disjoint blocks in parallel, as in e.g. the Splash sampler of [GLGG11].

4.4 Optimising Multi-Core Metropolis-Hastings

In what follows, we make the standing assumption that the cost of evaluating π and its derivatives is much greater than the cost of communication between processors. Suppose we are using proposals where step-sizes scale as $h = L^\delta \cdot N^{-\gamma}$, the asymptotic speed measure is given by $s(L) = L^\delta \cdot \alpha(L)$, and the asymptotic acceptance rate is given by $\alpha(L) = 2 \cdot \Phi(-cL^{\delta/2\gamma})$. We argue that, when P processors are available, the correct extension of the speed measure is given by

$$\begin{aligned} s_P(L) &= L^\delta \cdot \alpha_P^{\text{Eff}}(L) \\ \alpha_P^{\text{Eff}}(L) &= 1 - (1 - \alpha(L))^P . \end{aligned}$$

The justification for this choice is that with each accepted move, one is travelling a distance of the same order as the single-processor algorithm, but for each real unit of time, the probability of *at least one* of the P proposals being accepted is $\alpha_P^{\text{Eff}}(L)$. We note that this is compatible with each of the speed measures proposed in the $P = 1$ setting.

As before, one can re-write the speed measure purely in terms of the limiting acceptance probability α as

$$s_P(\alpha) \propto \left(\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right)^{2\gamma} \cdot \left(1 - (1-\alpha)^P\right),$$

up to a multiplicative factor which depends only on π.

We will now maximise this objective function for three commonly used families of proposals which are used in Metropolis-Hastings schemes. We use α_P^* to denote the optimal acceptance rate for a given family of proposals when P cores are available, and we define the relative efficiency of the P-processor scheme by

$$\text{RE}_P = s_P(\alpha_P^*)/s_1(\alpha_1^*),$$

i.e. the ratio of efficiencies of the best P-core algorithm to the best single-core algorithm. All numerical quantities are reported to 3 significant figures.

4.4.1 Application to RWMH

For random walk proposals, it was shown in [RGG97] that the dimension-robustness exponent is given by $\gamma = 1$. As a result, one can write the speed measure as

$$s_P(\alpha) \propto \left(\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right)^{2} \cdot \left(1 - (1-\alpha)^P\right).$$

Maximising this expression numerically for $P \in \{2^k : 0 \leqslant k \leqslant 10\}$, we obtain that

P	$\alpha^*_{\mathbf{P}}$	$\mathbf{RE_P}$
1	0.234	1.00
2	0.200	1.78
4	0.158	2.99
8	0.114	4.67
16	0.0759	6.82
32	0.0473	9.35
64	0.0280	12.2
128	0.0160	15.3
256	0.00887	18.6
512	0.00482	22.0
1024	0.00259	25.5

Table 4.1: Optimal Acceptance Rates and Relative Efficiency for Multi-Core Random-Walk Metropolis-Hastings with $P = 2^k$ processors for $1 \leqslant k \leqslant 10$.

Figure 4.3 shows i) the optimal acceptance rate, ii) the effective acceptance rate at optimality, and iii) the relative efficiency at optimality vary as P grows. We see that the optimal acceptance rate decreases, the effective acceptance rate at optimality increases towards 1, and the relative efficiency at optimality grows steadily. Note that the relative efficiency grows superlinearly with respect to $\log_2 P$, rather than with respect to P.

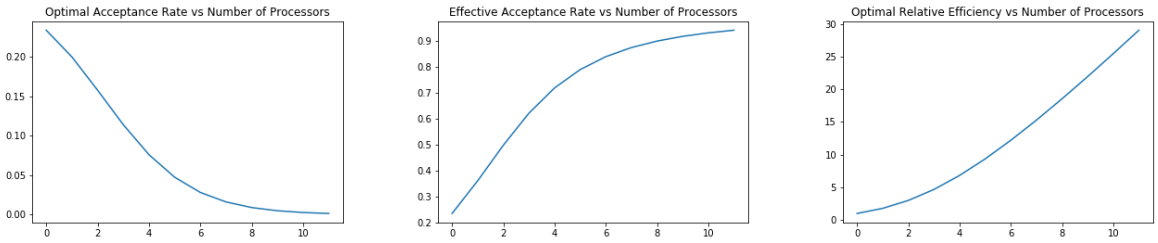

Figure 4.3: Asymptotics for Random Walk Proposals (horizontal axis is $\log_2 P$)

We also superimpose plots of efficiency against acceptance rate for different values of P. The key points to note are that as P grows, i) the efficiency curves move strictly upwards, ii) the values of α which maximise the curves move to the left, and iii) all curves lie below the 'envelope curve' $s_\infty(\alpha) \propto \left(\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right)^2$. Similar comments will apply in both of the following subsections; we will not repeat them explicitly.

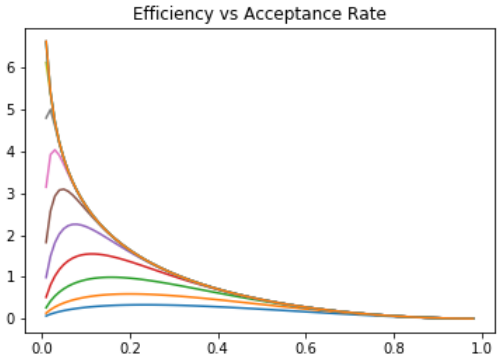

Figure 4.4: Efficiency-Acceptance Plots for RWMH, $p = 2^1, \ldots, 2^6$ (horizontal axis is $\log_2 P$)

4.4.2 Application to MALA

For Langevin diffusion proposals, it was shown in [RR98] that the dimension-robustness exponent is given by $\gamma = 1/3$. As a result, one can write the speed measure as

$$s_P(\alpha) \propto \left(\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right)^{2/3} \cdot \left(1 - (1-\alpha)^P\right).$$

Maximising this expression numerically for $P \in \{2^k : 0 \leqslant k \leqslant 10\}$, we obtain that

P	$\alpha^*_{\mathbf{P}}$	$\mathbf{RE_P}$
1	0.574	1.00
2	0.466	1.48
4	0.344	2.01
8	0.232	2.53
16	0.145	3.02
32	0.0854	3.46
64	0.0482	3.86
128	0.0264	4.21
256	0.0142	4.53
512	0.00750	4.82
1024	0.00393	5.09

Table 4.2: Optimal Acceptance Rates and Relative Efficiency for Multi-Core Metropolis-Adjusted Langevin Algorithm with $P = 2^k$ processors for $1 \leqslant k \leqslant 10$.

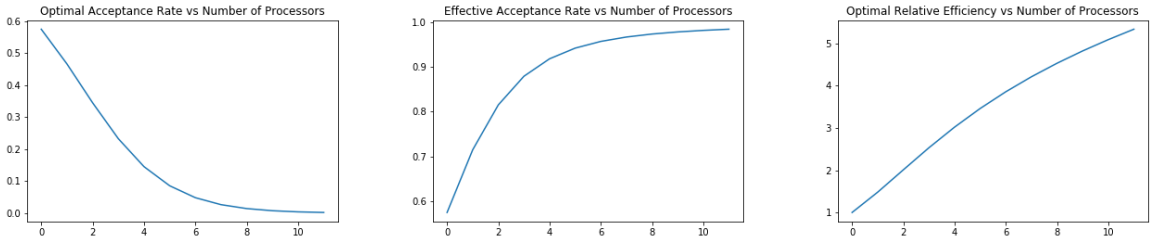

Figure 4.5: Asymptotics for Langevin Diffusion Proposals

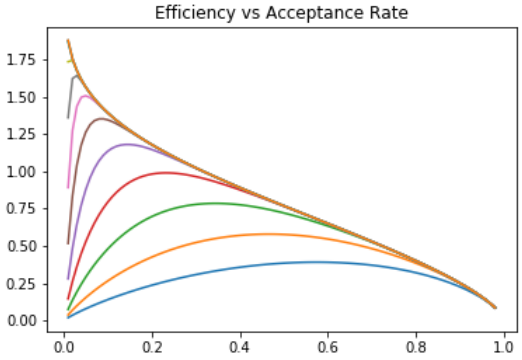

Figure 4.6: Efficiency-Acceptance Plots for MALA, $p = 2^1, \ldots, 2^6$ (horizontal axis is $\log_2 P$)

4.4.3 Application to HMC

For Hamiltonian dynamics proposals,it was shown in [BPR+13] that the dimension-robustness exponent is given by $\gamma = 1/4$. As a result, one can write the speed measure as

$$s_P(\alpha) \propto \left(\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right)^{1/2} \cdot \left(1 - (1-\alpha)^P\right).$$

Maximising this expression numerically for $P \in \{2^k : 0 \leqslant k \leqslant 10\}$, we obtain that

$\mathbf{P}$	$\alpha^*_{\mathbf{P}}$	$\mathbf{RE_P}$
1	0.651	1.00
2	0.528	1.41
4	0.390	1.82
8	0.262	2.21
16	0.162	2.54
32	0.0950	2.83
64	0.0533	3.08
128	0.0290	3.30
256	0.0155	3.49
512	0.00818	3.66
1024	0.00428	3.81

Table 4.3: Optimal Acceptance Rates and Relative Efficiency for Multi-Core Hamiltonian Monte Carlo with $P = 2^k$ processors for $1 \leqslant k \leqslant 10$.

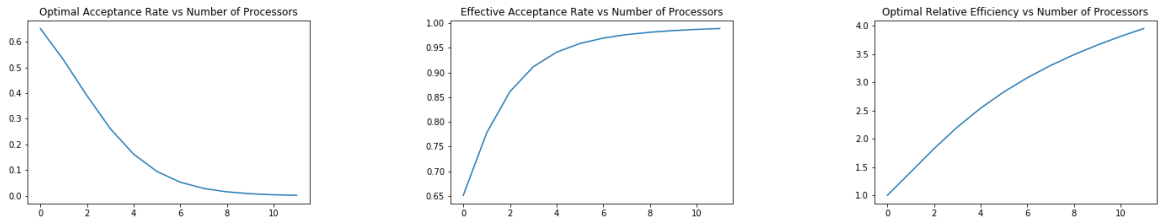

Figure 4.7: Asymptotics for Hamiltonian Dynamics Proposals

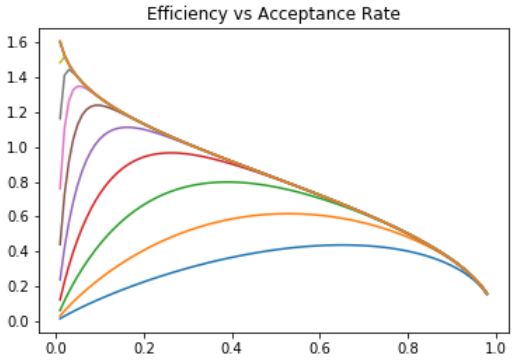

Figure 4.8: Efficiency-Acceptance Plots for HMC, $p = 2^1, \ldots, 2^6$ (horizontal axis is $\log_2 P$)

4.4.4 Empirical Verification

In order to assess the extent to which these theoretical predictions are borne out in a practical setting, we carried out exploratory experiments. If these predictions are to be

taken seriously, the natural application is to i) run the Multi-Core Metropolis-Hastings algorithm and ii) tune the step-size such that the acceptance rate matches the theoretical recommendation. As such, we are chiefly concerned with verifying whether the *theoretically* most-efficient acceptance rate is consistent with the *empirically* most-efficient acceptance rate setting. If this is borne out, then we are prepared to believe that the theoretical recommendations are valid. In our experiments, we measure efficiency by dimension-normalised expected squared jumping distance (ESJD), that is,

$$\mathrm{ESJD}_d = \frac{1}{d}\mathbf{E}_{x\sim\pi}\left[\mathbf{E}_{y\sim q^{\mathrm{MH}}(x\to y)}\left[|y-x|^2\right]\right],$$

where $q^{\mathrm{MH}}(x \to y)$ is the transition kernel of the Metropolised chain.

To this end, for each of the proposals {RW, LD, HD}, we ran the Multi-Core Metropolis-Hastings algorithm with $P \in \{1, 4, 16, 64\}$ cores, on a target formed as an iid product of D standard Gaussian measures. For the proposals {RW, LD}, we took $D = 100$, and ran the chain for 25000 steps. For HD, we took $D = 250$, and ran the chain for 10000 steps, at each step using $s \sim \mathrm{Unif}(1, 2, \ldots, 10)$ steps in the leapfrog integrator. Each chain was initialised at stationarity, and there were no indications that any of the chains had difficulty in converging to equilibrium. Each experimental setting of (h, P) was repeated between 5 and 10 times, always resulting in well-concentrated estimates of the average acceptance rate and ESJD. For the plots below, at each setting of (h, P), we took the median of the average acceptance rate and ESJD to de-clutter and smoothen the plots.

The key takeaways from the experiments are that the theoretically-optimal acceptance rates are in reasonable agreement with the empirically-optimal acceptance rates. For both RW and LD, the recommended α^* is routinely within $\pm 5\%$ of the empirically-optimal setting, indicating that the recommendations can be trusted reasonably well. The results for HD are somewhat less reliable, as even for large d, the optimal step-size decays quite slowly, and the asymptotic regime is less visible.

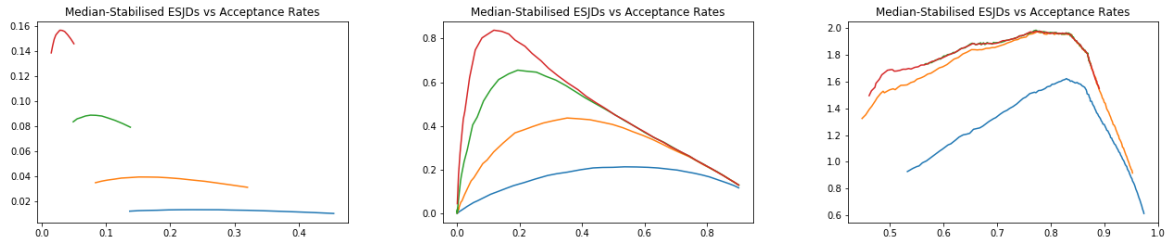

Figure 4.9: Plots of ESJD against Average Acceptance Rate for RWMH (left), MALA (middle), and HMC (right). { Blue, Orange, Green, Red } lines correspond to $P = \{1, 4, 16, 64\}$ respectively. In the rightmost plot, the Green line is hidden behind the Red line.

4.5 Asymptotic Behaviour of Multi-Core Metropolis-Hastings in the Large-P Regime

While the finite-P results obtained in the previous section are already directly applicable, it is also of interest to understand what can be expected in the large-P limit, i.e. to understand the asymptotic benefits of having many cores available. In this section, we attempt to answer this question by studying how the relative efficiency of the multi-core algorithm scales with P.

Theorem 12. *Consider a family of proposal distributions $\{q(x \to y; h)\}_{x \in \mathcal{X}, h>0}$ with dimension-robustness constant γ, such that the P-processor efficiency measure is proportional to*

$$s_P(\alpha) = \left(\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right)^{2\gamma} \cdot \left(1 - (1-\alpha)^P\right).$$

There exists a positive constant $C(\gamma) < \infty$ such that for $P > 1$, the optimal efficiency $s_P(\alpha_P^*)$ is bounded from above as

$$s_P(\alpha_P^*) \leqslant C(\gamma) \cdot (\log P)^\gamma .$$

Moreover, let $c > 0$, and define $\alpha(P) = c \cdot P^{-1}$. There then exists a positive constant $D(c, \gamma) \leqslant C(\gamma)$ such that for $P > 1$, the efficiency measure is bounded from below as

$$s_P(\alpha(P)) \geqslant D(c, \gamma) \cdot (\log P)^\gamma .$$

The proof of this theorem can be found in the appendix.

The content of this theorem is twofold. Firstly, it asserts that the gain in relative efficiency which can be achieved by using the Multi-Core Metropolis-Hastings algorithm is at most of order $(\log P)^\gamma$, where γ is a constant depending only on the family of proposals. One immediate implication of this is that parallelism is *relatively* more beneficial for families of proposal with a large value of γ, i.e. proposals which are *less robust* to dimensionality. We highlight that this does not mean that such proposals should be preferred in general; the assertion is that their performance will be improved more *in relative terms*. It is worth emphasising that the benefits of using smarter proposal distributions will typically overwhelm the benefits of parallelism.

The second part of the theorem states that, for large P, one can get within a constant factor of the optimal efficiency simply by scaling the target acceptance rate like $\alpha(P) \propto P^{-1}$. Unfortunately, we were unable to derive a tight characterisation of the optimal acceptance rate; one can also get within a constant factor of optimality by scaling $\alpha(P) \propto P^{-1-\varepsilon}$ for $\varepsilon > 0$. It would be of theoretical interest to obtain a tight characterisation of how α_P^*

scales with P. From a practical standpoint, however, this is not a concern, as obtaining α_P^* numerically only requires solution of a 1-dimensional, unimodal maximisation.

A closely-related open problem which remains is to obtain a finer characterisation of the limiting behaviour of the effective acceptance rate. Empirically, it is clear that α_P^{Eff} increases with P, and we conjecture that it will in fact converge to 1. Note that resolving the asymptotic behaviour of α_P^* would essentially settle this question; in particular, verifying that $1/\alpha_P^* = o(P)$ (as we observe numerically) would imply that $\alpha_P^{\text{Eff}} = 1 - o(1)$ as $P \to \infty$. It would again be of theoretical interest to confirm this conjecture, as well as to quantify the rate at which convergence takes place.

4.6 Conclusion and Outlook

In this work, we have put forward an MCMC procedure which makes use of parallel computing resources, and derived results which indicate what one can expect to gain from using this procedure. To conclude this work, we address three natural follow-up questions, namely: i) where should this procedure be used?, ii) to which other MCMC schemes could this framework be extended?, and iii) when using parallel computing resources to enhance MCMC sampling, is this the right way to go about it? We proceed to address these sequentially.

One natural use-case for the Multi-Core Metropolis-Hastings algorithm would be in scenarios where the acceptance rate achievable by existing proposals is already very low. While it is perhaps trivially true a priori that parallelism would help here, the theory derived here implies that it would also help more in a *relative* sense: worse proposals (lower γ) benefit relatively more from the presence of additional processors (higher P). A key candidate in this category would be Reversible-Jump MCMC [Gre95], where deriving efficient between-model moves remains a difficult task (though see the work of [BGR03, Gag19] for some guidance on this task). Another candidate would be MCMC implementations of Approximate Bayesian Computation [MMPT03], where acceptance rates can also be unstable due to the additional randomness involved with estimating the intractable likelihood. We note that the r-Hit Kernel of Lee [Lee12] is one approach which can provably [LŁ14] stabilise these acceptance rates, though requires additional computation; reducing the real-time cost of this algorithm through parallelisation is a natural strategy, quite related to the algorithm presented here.

There are of course a number of MCMC algorithms which are *not* covered by this framework, perhaps the most obvious being Gibbs sampling [GG84, GS90]. Without pressing into the details, it seems clear that the use of parallelism in Gibbs-type algorithms ought to be qualitatively different than for Metropolis-Hastings algorithms; see [GLGG11, JSW13, TSD20] for some examples in this direction.

A class of Metropolis-Hastings schemes which are *not* immediately covered by this work, but to which the analysis could reasonably be extended, are those which involve multiple algorithmic parameters. One key example is Pseudo-Marginal Metropolis-Hastings [AR09], in which one needs to tune both i) the proposal step-size, and ii) the computational effort deployed to estimate the likelihood*. There are thus multiple scalings at play. To give a concrete example, given $P = P_1 \cdot P_2$ processors, one could make P_1 proposal moves, and then for each of them, form an importance sampling estimator of the likelihood using P_2 samples. Given any such factorisation of P, there will be an optimal step-size h; one must then *also* optimise over these factorisations. Extending the analysis of these

*Note that [Dro14] consider using multi-CPU architectures in this way, but only to estimate the likelihood more accurately.

methods to the multi-core setting would likely involve some synthesis of the techniques of [DPDK15, She16] with the analysis presented herein. It would be particularly interesting to see how the tradeoff between P_1 and P_2 is navigated as P grows.

Note that these comments also apply to both ABC-MCMC and Particle Metropolis-Hastings, which can be viewed as instances of the Pseudo-Marginal framework. Other related families of algorithms with a multi-stage structure in this vein include Multiple-Try Metropolis [LLW00, BDM12], and MHAAR Algorithms [ADYC18], and would likely require similar techniques.

A more recent pair of algorithm families which might be more amenable to direct analysis are Sequential Proposal MCMC [PA20] and Discrete PDMPs [VBCDD17, ST17, Mon19]. The extent to which these algorithms will benefit from parallelism depends heavily on the proposal mechanism at hand. In particular, for the HMC-like algorithms proposed in [PA20, VBCDD17], generating T steps' worth of proposals must be done sequentially , and thus will generally take $O(T)$, even in real time. In contrast, for the Discrete Bouncy Particle Sampler of [ST17] and its elliptical variant, under the assumption that no bounce takes place, step generation can be carried out in constant time, using parallel resources. Given a suitable optimal scaling result for the dBPS, it would be straightforward to extend the framework outlined in this work to devise a variant of the dBPS which leverages parallel structure, and compute the analogous finite-P recommendations and large-P asymptotics.

In closing, we offer some remarks on the outlook for parallelism in MCMC. Perhaps the most important question in this domain is precisely how one should employ parallel computing resources. In this work, we have focused on a scheme which maintains a single Markov chain, and sets multiple cores to work on trying to get this chain to move around the space. This setting allows for a transparent analysis, but is far from the only way of doing things. We present here three particularly promising alternatives, which could each be used in place of, or in conjunction with, the methods presented in this work.

Firstly, one could consider running multiple independent chains, distributed across cores. This would be relatively simple to implement and analyse. For example, once all of the chains have converged to stationarity, this would lead to an Effective Sample Size approximately P times bigger than the single-core algorithm. However, in real-time, the chains would converge to equilibrium no quicker than a single chain, and as such, one has to wait until stationarity is reached to reap the benefits. A practical benefit of this approach is that by sharing proposal parameters across chains, adaptation can be carried out more robustly, as the increased number of chains serves to reduce the variability of the associated stochastic approximation schemes which are typical of adaptive MCMC (see e.g. [AT08, RR09, AFMP09] for details). A sensible hybridisation of this approach with the approach of this paper would be to first use the Multi-Core algorithm with a

large step-size to reach equilibrium, and then once convergence is detected, switch to using independent chains to collect samples.

Secondly, one could consider running multiple *interacting* chains, as in [GRG94, GW10, LMW18, GINR19]. Algorithms of this form are not as well-understood as their non-interacting counterparts, but progress is being made (see e.g. [NP19, DNS19, GIHLS20]). One key question in this area which remains unclear is precisely how the chains best ought to interact in order to improve convergence to equilibrium. Given a suitable interaction scheme, one can reasonably imagine such chains would be able to outperform non-interacting approaches.

Thirdly, one could consider running multiple chains, each carrying out *distinct* tasks, as in e.g. Umbrella Sampling [TVKWD16] or Parallel Tempering [SW86, ARR11, SBCDD19]. This represents a contrasting 'Divide-and-Conquer' approach to sampling, and is perhaps more typical of how parallelism is employed in other domains. A key practical challenge here is to ascertain precisely *how* one ought to divide the task, e.g. *which* distinct tasks each core should be solving, but the potential benefits are quite clear: given a suitable stratification of the task at hand, one can dramatically reduce the complexity of the problem.

4.7 Appendix

In this section, we prove Theorem 12 from the main text. We do so by bounding the growth in efficiency for any possible sequence of acceptance rates $\alpha(P)$.

Lemma 13. *For $\alpha \in (0, a)$, with $a = 2\left(1 - \Phi\left(\frac{1}{\sqrt{2\pi}}\right)\right) \approx 0.689936$, it holds that*

$$\Phi^{-1}\left(1 - \frac{\alpha}{2}\right) \leqslant \sqrt{2 \log \frac{2}{\alpha}}$$

Proof. Let $z = \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)$, then

$$\frac{\alpha}{2} = \int_z^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) dx \leqslant \int_z^\infty \frac{1}{\sqrt{2\pi}} \frac{x}{z} \exp\left(-\frac{1}{2}x^2\right) dx = \frac{1}{\sqrt{2\pi}} \frac{1}{z} \exp\left(-\frac{1}{2}z^2\right)$$

and hence $\log \frac{2}{\alpha} \geqslant \frac{1}{2}z^2 + \log\left(\sqrt{2\pi}z\right)$. Because $\alpha \in (0, a)$ implies that $z > \frac{1}{\sqrt{2\pi}}$, we can deduce that $\log \frac{2}{\alpha} \geqslant \frac{1}{2}z^2$, from which the result follows. □

Note that for the three proposal families we consider, the optimal single-processor acceptance probability is less than a, and thus we can restrict ourselves to sequences of acceptance rates $\alpha(P)$ which stay in $[0, a)$ for all P. By Lemma 13, we can bound s_P from above by $t_P(\alpha) = \left(\log \frac{2}{\alpha}\right)^\gamma \cdot \left(1 - (1 - \alpha)^P\right)$, which is simpler to compute with. In what follows, we will bound the growth of t_P from above, and use these bounds to control the growth of s_P.

Proposition 14. *If $\liminf_{P\to\infty} \alpha(P) > 0$, then it holds that $\sup_{P\geqslant 1} t_P(\alpha) < \infty$.*

Proof. Let $\liminf_{P\to\infty} \alpha(P) = a_0$; then $\alpha(P) \geqslant a_0/2 > 0$ eventually. One then has that uniformly in P,

$$\begin{aligned} t_P(\alpha) &\leqslant \left(\log \frac{4}{a}\right)^\gamma \cdot \left(1 - (1 - \alpha)^P\right) \\ &\leqslant \left(\log \frac{4}{a}\right)^\gamma \end{aligned}$$

from which one deduces that $\sup_{P\geqslant 1} t_P(\alpha) < \infty$. □

Proposition 15. *If $\liminf_{P\to\infty} P \cdot \alpha(P) = c \in (0, \infty) \cup \{\infty\}$, then it holds that*

$$\sup_{P\geqslant 2} \frac{t_P(\alpha)}{(\log P)^\gamma} < \infty$$

Proof. By assumption, we have that $P \cdot \alpha(P) > \tilde{c} = \min(c/2, 2)$ eventually. Hence

eventually,

$$\begin{aligned} t_P(\alpha) &\leqslant \left(\log \frac{2P}{\tilde{c}}\right)^{\gamma} \cdot \left(1-(1-\alpha)^P\right) \\ &\leqslant \left(\log \frac{2P}{\tilde{c}}\right)^{\gamma}. \end{aligned}$$

Noting that for $P \geqslant 2$, it holds that

$$\begin{aligned} \left(\log \frac{2P}{\tilde{c}}\right)^{\gamma} &= \left(\log P + \log \frac{2}{\tilde{c}}\right)^{\gamma} \\ &= (\log P)^{\gamma} \cdot \left(1 + \frac{\log \frac{2}{\tilde{c}}}{\log P}\right)^{\gamma} \\ &\leqslant (\log P)^{\gamma} \cdot \left(1 + \frac{\log \frac{2}{\tilde{c}}}{\log 2}\right)^{\gamma}, \end{aligned}$$

we deduce that $\sup_{P \geqslant 2} \frac{t_P(\alpha)}{(\log P)^{\gamma}} < \infty$. □

Proposition 16. *Under the assumption that*

$$\begin{aligned} \liminf_{P\to\infty} P \cdot \alpha(P) &= 0 \\ \limsup_{P\to\infty} \frac{\log\left(\frac{1}{P\cdot\alpha(P)}\right)}{\log P} &= c < \infty, \end{aligned}$$

it holds that $\sup_{P \geqslant 2} \frac{t_P(\alpha)}{(\log P)^{\gamma}} < \infty$.

Proof. As $\limsup_{P\to\infty} \frac{\log\left(\frac{1}{P\cdot\alpha(P)}\right)}{\log P} = c$, then for sufficiently large P, we have that

$$\frac{\log\left(\frac{1}{P\cdot\alpha(P)}\right)}{\log P} \leqslant 2c.$$

We can then see that $\log \frac{2}{\alpha} \leqslant \log 2 + (2c+1)\log P$. Moreover, using Bernoulli's inequality, one can bound $\left(1-(1-\alpha)^P\right) \leqslant P\alpha$. Hence, for $P \geqslant 2$,

$$\begin{aligned} \frac{t_P(\alpha)}{(\log P)^{\gamma}} &\leqslant \frac{(\log 2 + (2c+1)\log P)^{\gamma} \cdot P\alpha}{(\log P)^{\gamma}} \\ &= \left(2c+1+\frac{\log 2}{\log P}\right)^{\gamma} \cdot P\alpha \\ &\leqslant (2c+1+1)^{\gamma} \cdot P\alpha. \end{aligned}$$

The first term in the product is constant with respect to P, and $P\alpha \to 0$ by assumption, hence the result follows. □

Lemma 17. *Let $x_0 \in (0,1)$. There exists a constant $C > 1$ such that for all $x \in [0, x_0]$ and for all $P \geqslant 1$, the following inequality holds:*

$$\left(1 - \frac{x}{P}\right)^P \geqslant \exp(-Cx).$$

Proof. Consider $g(x) = \log\left(\frac{1}{1-x}\right)$. g is convex on $[0,1)$, hence for any $x \in [0, x_0]$, once can bound g from above by its secant $S(x) = Cx$, where $C = \frac{g(x_0)}{x_0}$. Thus, for all $x \in [0, x_0]$, we have

$$g(x) = \log\left(\frac{1}{1-x}\right) \leqslant Cx \implies 1 - x \geqslant \exp(-Cx),$$

and by noting that $x \in [0, x_0], P \geqslant 1$ implies that $\frac{x}{P} \in [0, x_0]$ also, we can deduce that

$$1 - \frac{x}{P} \geqslant \exp\left(-C\frac{x}{P}\right) \implies \left(1 - \frac{x}{P}\right)^P \geqslant \exp(-Cx)$$

as required. □

Proposition 18. *If $\liminf_{P\to\infty} P \cdot \alpha(P) = 0$ and*

$$\limsup_{P\to\infty} \frac{\log\left(\frac{1}{P\cdot\alpha(P)}\right)}{\log P} = \infty$$

along any subsequence such that $P \cdot \alpha(P) \to 0$, then it holds that

$$\sup_{P\geqslant 2} \frac{t_P(\alpha)}{(\log P)^\gamma} < \infty$$

Proof. By Lemma 17, along any subsequence of P such that $P \cdot \alpha(P) \to 0$, we can eventually bound

$$(1-\alpha)^P = \left(1 - \frac{P\alpha}{P}\right)^P \geqslant \exp(-CP\alpha)$$

for some constant $C > 1$. Then, for sufficiently large P,

$$1 - (1-\alpha)^P \leqslant 1 - \exp(-CP\alpha) \leqslant CP\alpha.$$

Write $\rho(P) = \log\left(\frac{1}{P\cdot\alpha(P)}\right)/\log P$; one can then write

$$\begin{aligned}\frac{t_P(\alpha)}{(\log P)^\gamma} &\leqslant C\exp(-P\rho(P))\cdot\left(\frac{\log 2+(1+\rho(P))\cdot\log P}{\log P}\right)^\gamma\\ &= C\exp(-P\rho(P))\cdot\left(\frac{\log 2}{\log P}+(1+\rho(P))\right)^\gamma\\ &\leqslant C\exp(-2\rho(P))\cdot(1+(1+\rho(P)))^\gamma\,.\end{aligned}$$

Now, it is straightforward to show that

$$\begin{aligned}\sup_{\rho>0}\left[\exp(-2\rho)\cdot(2+\rho)^\gamma\right] &\leqslant \sup_{\rho+2>0}\left[\exp(-2\rho)\cdot(2+\rho)^\gamma\right]\\ &= \exp(4)\sup_{\mu>0}\left[\mu^\gamma\exp(-2\mu)\right]\\ &= \exp(4)\left(\frac{\gamma}{2\exp(1)}\right)^\gamma<\infty\end{aligned}$$

from which we obtain the desired result. □

Theorem 19. *There exists a positive constant $C(\gamma)<\infty$ such that for $P>1$, the optimal efficiency $s_P(\alpha_P^*)$ is bounded from above as*

$$s_P(\alpha_P^*)\leqslant C(\gamma)\cdot(\log P)^\gamma\,.$$

Proof. Propositions 14, 15, 16 and 18 establish this to be true for all possible behaviours of acceptance sequences, so in particular, it is true for the optimal sequence $\alpha(P)=\alpha_P^*$. The result follows. □

Proposition 20. *By taking $\alpha(P)=\frac{c}{P}$, one can achieve an efficiency of order $(\log P)^\gamma$.*

Proof. For small α, $s_P(\alpha)$ and $t_P(\alpha)$ are asymptotically equivalent. Scaling $\alpha(P)=\frac{c}{P}$, one has that $\left(\log\frac{2}{\alpha}\right)^\gamma=\left(\log P+\log\frac{2}{c}\right)^\gamma\sim(\log P)^\gamma$. Furthermore, $\left(1-\frac{c}{P}\right)^P\to\exp(-c)$ as $P\to\infty$. Hence, $t_P(\alpha(P))\sim(1-\exp(-c))\cdot(\log P)^\gamma$, which is of the desired order. □

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[ACD11] Artin Armagan, Merlise Clyde, and David B Dunson. Generalized beta mixtures of gaussians. In *Advances in neural information processing systems*, pages 523–531, 2011.

[ADL13] Artin Armagan, David B Dunson, and Jaeyong Lee. Generalized double pareto shrinkage. *Statistica Sinica*, 23(1):119, 2013.

[ADL16] Johan Alenlöv, Arnaud Doucet, and Fredrik Lindsten. Pseudo-marginal hamiltonian monte carlo. *arXiv preprint arXiv:1607.02516*, 2016.

[ADNR18] Christophe Andrieu, Alain Durmus, Nikolas Nüsken, and Julien Roussel. Hypocoercivity of piecewise deterministic markov process-monte carlo. *arXiv preprint arXiv:1808.08592*, 2018.

[ADYC18] Christophe Andrieu, Arnaud Doucet, Sinan Yıldırım, and Nicolas Chopin. On the utility of metropolis-hastings with asymmetric acceptance ratio. *arXiv preprint arXiv:1803.09527*, 2018.

[AFMP09] Yves Atchade, Gersende Fort, Eric Moulines, and Pierre Priouret. Adaptive markov chain monte carlo: theory and methods. *Preprint*, 2009.

[AG07] Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*, volume 57. Springer Science & Business Media, 2007.

[AGH+19] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[AH12] David F Anderson and Desmond J Higham. Multilevel monte carlo for continuous time markov chains, with applications in biochemical kinetics. *Multiscale Modeling & Simulation*, 10(1):146–179, 2012.

[AKW+14] Elaine Angelino, Eddie Kohler, Amos Waterland, Margo Seltzer, and Ryan P. Adams. Accelerating mcmc via parallel predictive prefetching, 2014.

[AL19] Christophe Andrieu and Samuel Livingstone. Peskun-tierney ordering for markov chain and process monte carlo: beyond the reversible scenario. *arXiv preprint arXiv:1906.06197*, 2019.

[Ans01] Luc Anselin. Spatial econometrics. *A companion to theoretical econometrics*, 310330, 2001.

[App09] David Applebaum. *Lévy processes and stochastic calculus*. Cambridge university press, 2009.

[APSAS17] S Agapiou, Omiros Papaspiliopoulos, D Sanz-Alonso, and AM Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3):405–431, 2017.

[AR09] Christophe Andrieu and Gareth O Roberts. The pseudo-marginal approach for efficient monte carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

[ARR09] YF Atchadé, Gareth O Roberts, and Jeffrey S Rosenthal. Optimal scaling of metropolis-coupled markov chain monte carlo. *Preprint*, 2009.

[ARR11] Yves F Atchadé, Gareth O Roberts, and Jeffrey S Rosenthal. Towards optimal scaling of metropolis-coupled markov chain monte carlo. *Statistics and Computing*, 21(4):555–568, 2011.

[AS04] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[Asm08] Søren Asmussen. *Applied probability and queues*, volume 51. Springer Science & Business Media, 2008.

[AT08] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and computing*, 18(4):343–373, December 2008.

[Bac19] Francis Bach. Submodular functions: from discrete to continuous domains. *Mathematical Programming*, 175(1-2):419–459, 2019.

[Bal17] Carlo Baldassi. A method to reduce the rejection rate in monte carlo markov chains. *Journal of Statistical Mechanics: Theory and Experiment*, 2017(3):033301, 2017.

[BCH+12] Kurt Binder, David M Ceperley, J-P Hansen, MH Kalos, DP Landau, D Levesque, H Mueller-Krumbhaar, D Stauffer, and J-J Weis. *Monte Carlo methods in statistical physics*, volume 7. Springer Science & Business Media, 2012.

[BCSJ12] Alexandre Bouchard-Côté, Sriram Sankararaman, and Michael I Jordan. Phylogenetic inference via sequential monte carlo. *Systematic biology*, 61(4):579–593, 2012.

[BCVD18] Alexandre Bouchard-Côté, Sebastian J Vollmer, and Arnaud Doucet. The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method. *Journal of the American Statistical Association*, 113(522):1–13, 2018.

[BDM12] Mylène Bédard, Randal Douc, and Eric Moulines. Scaling analysis of multiple-try mcmc methods. *Stochastic Processes and their Applications*, 122(3):758–786, 2012.

[BDM14] Mylène Bédard, Randal Douc, and Eric Moulines. Scaling analysis of delayed rejection mcmc methods. *Methodology and Computing in Applied Probability*, 16(4):811–838, 2014.

[Béd17] Mylène Bédard. Hierarchical models: Local proposal variances for rwm-within-gibbs and mala-within-gibbs. *Computational Statistics & Data Analysis*, 109:231–246, 2017.

[Bet17] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[BFH+20] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. Jax: composable transformations of python+ numpy programs, 2018. *URL http://github.com/google/jax*, page 18, 2020.

[BFR19] Joris Bierkens, Paul Fearnhead, and Gareth Roberts. The zig-zag process and super-efficient sampling for bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

[BGL+17] Alexandros Beskos, Mark Girolami, Shiwei Lan, Patrick E Farrell, and Andrew M Stuart. Geometric mcmc for infinite-dimensional inverse problems. *Journal of Computational Physics*, 335:327–351, 2017.

[BGR03] Stephen P Brooks, Paolo Giudici, and Gareth O Roberts. Efficient construction of reversible jump markov chain monte carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):3–39, 2003.

[BGvdMS20] Joris Bierkens, Sebastiano Grazzi, Frank van der Meulen, and Moritz Schauer. A piecewise deterministic monte carlo method for diffusion bridges. *arXiv preprint arXiv:2001.05889*, 2020.

[Bie16] Joris Bierkens. Non-reversible metropolis-hastings. *Statistics and Computing*, 26(6):1213–1228, 2016.

[Bil08] Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2008.

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[BJB10] Jonathan MR Byrd, Stephen A Jarvis, and Abhir H Bhalerao. On the parallelisation of mcmc by speculative chain execution. In *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010.

[BKL75] Alfred B Bortz, Malvin H Kalos, and Joel L Lebowitz. A new algorithm for monte carlo simulation of ising spin systems. *Journal of Computational Physics*, 17(1):10–18, 1975.

[BMP12] Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.

[BPR06] Alexandros Beskos, Omiros Papaspiliopoulos, and Gareth O Roberts. Retrospective exact simulation of diffusion sample paths with applications. *Bernoulli*, 12(6):1077–1098, 2006.

[BPR+13] Alexandros Beskos, Natesh Pillai, Gareth Roberts, Jesus-Maria Sanz-Serna, and Andrew Stuart. Optimal tuning of the hybrid monte carlo algorithm. *Bernoulli*, 19(5A):1501–1534, 2013.

[BPRF06] Alexandros Beskos, Omiros Papaspiliopoulos, Gareth O Roberts, and Paul Fearnhead. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):333–382, 2006.

[BPSSS11] Alexandros Beskos, Frank J Pinski, Jesús Maria Sanz-Serna, and Andrew M Stuart. Hybrid monte carlo on hilbert spaces. *Stochastic Processes and their Applications*, 121(10):2201–2230, 2011.

[BR05] Alexandros Beskos and Gareth O Roberts. Exact simulation of diffusions. *The Annals of Applied Probability*, 15(4):2422–2444, 2005.

[Bré13] Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

[BREZ18] Nawaf Bou-Rabee, Andreas Eberle, and Raphael Zimmer. Coupling and convergence for hamiltonian monte carlo. *arXiv preprint arXiv:1805.00452*, 2018.

[Bro06] Anthony Brockwell. Parallel markov chain monte carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 15(1):246–261, March 2006.

[BW87] Joseph D Bryngelson and Peter G Wolynes. Spin glasses and the statistical mechanics of protein folding. *Proceedings of the National Academy of Sciences*, 84(21):7524–7528, 1987.

[BWBS19] Thomas B Berrett, Yi Wang, Rina Foygel Barber, and Richard J Samworth. The conditional permutation test for independence while controlling for confounders. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2019.

[BZB02] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[CCBJ18] Xiang Cheng, Niladri S Chatterji, Peter L Bartlett, and Michael I Jordan. Underdamped langevin mcmc: A non-asymptotic analysis. In *Conference on Learning Theory*, pages 300–323, 2018.

[CD18] Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

[CF05] J. Andrés Christen and Colin Fox. Markov chain monte carlo using an approximation. *Journal of Computational and Graphical Statistics*, 14(4):795–810, December 2005.

[CG92] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[CG99] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 378–388. IEEE, 1999.

[CH13] Ting-Li Chen and Chii-Ruey Hwang. Accelerating reversible markov chains. *Statistics & Probability Letters*, 83(9):1956–1962, 2013.

[CK20] Stefano Carrazza and Daniel Krefl. Sampling the riemann-theta boltzmann machine. *Computer Physics Communications*, page 107464, 2020.

[CKP20] Martin Chak, Nikolas Kantas, and Grigorios A Pavliotis. On the generalised langevin equation for simulated annealing. *arXiv preprint arXiv:2003.06448*, 2020.

[CLP99] Fang Chen, László Lovász, and Igor Pak. Lifting markov chains to speed up mixing. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 275–281. ACM, 1999.

[CLW19] Yu Cao, Jianfeng Lu, and Lihan Wang. On explicit l^2-convergence rate estimate for underdamped langevin dynamics. *arXiv preprint arXiv:1908.04746*, 2019.

[CMG+18] Wilson Ye Chen, Lester Mackey, Jackson Gorham, Francois-Xavier Briol, and Chris Oates. Stein points. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 844–853, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[CPS10] Carlos M Carvalho, Nicholas G Polson, and James G Scott. The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480, 2010.

[CR96] George Casella and Christian P Robert. Rao-blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[CR11a] Bruno Casella and Gareth O Roberts. Exact simulation of jump-diffusion processes with monte carlo applications. *Methodology and Computing in Applied Probability*, 13(3):449–473, 2011.

[CR11b] Dan Crisan and Boris Rozovskii. *The Oxford handbook of nonlinear filtering*. Oxford University Press, 2011.

[Cre81] Michael Creutz. Monte carlo study of renormalization in lattice gauge theory. *Physical Review D*, 23(8):1815, 1981.

[Cre12] Drew Creal. A survey of sequential monte carlo methods for economics and finance. *Econometric reviews*, 31(3):245–296, 2012.

[CRR03] Olivier Cappé, Christian P Robert, and Tobias Rydén. Reversible jump, birth-and-death and more general continuous time markov chain monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):679–700, 2003.

[CRR05] Ole F Christensen, Gareth O Roberts, and Jeffrey S Rosenthal. Scaling limits for the transient phase of local metropolis–hastings algorithms. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):253–268, 2005.

[CSASS19] MP Calvo, D Sanz-Alonso, and JM Sanz-Serna. Hmc: avoiding rejections by not using leapfrog and an analysis of the acceptance rate. *arXiv preprint arXiv:1912.03253*, 2019.

[Dal17] Arnak S Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):651–676, 2017.

[Dav84] Mark HA Davis. Piecewise-deterministic markov processes: A general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 353–388, 1984.

[DBCD19] George Deligiannidis, Alexandre Bouchard-Côté, and Arnaud Doucet. Exponential ergodicity of the bouncy particle sampler. *The Annals of Statistics*, 47(3):1268–1287, 2019.

[DCWY19] Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-hastings algorithms are fast. *Journal of Machine Learning Research*, 20(183):1–42, 2019.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Annual Cryptology Conference*, pages 40–56. Springer, 2013.

[Dev06] Luc Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006.

[DGA00] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[DGM18a] Alain Durmus, Arnaud Guillin, and Pierre Monmarché. Geometric ergodicity of the bouncy particle sampler. *arXiv preprint arXiv:1807.05401*, 2018.

[DGM18b] Alain Durmus, Arnaud Guillin, and Pierre Monmarché. Piecewise deterministic markov processes and their invariant measure. *arXiv preprint arXiv:1807.05421*, 2018.

[DHN00] Persi Diaconis, Susan Holmes, and Radford M Neal. Analysis of a nonreversible markov chain sampler. *Annals of Applied Probability*, pages 726–752, 2000.

[DKPR87] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[DLP16] Andrew B Duncan, Tony Lelievre, and GA Pavliotis. Variance reduction using nonreversible langevin samplers. *Journal of statistical physics*, 163(3):457–491, 2016.

[DM97] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 325(6):653–658, 1997.

[DM13] Persi Diaconis and Laurent Miclo. On the spectral analysis of second-order markov chains. *Annales de la Faculté des Sciences de Toulouse. Mathématiques. Série 6*, 22(3):573–621, 2013.

[DM17] Alain Durmus and Éric Moulines. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, June 2017.

[DMDJ06] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[DMS15] Jean Dolbeault, Clément Mouhot, and Christian Schmeiser. Hypocoercivity for linear kinetic equations conserving mass. *Transactions of the American Mathematical Society*, 367(6):3807–3828, 2015.

[DNP17] AB Duncan, N Nüsken, and GA Pavliotis. Using perturbed underdamped langevin dynamics to efficiently sample from probability distributions. *Journal of Statistical Physics*, 169(6):1098–1131, 2017.

[DNS19] A Duncan, N Nüsken, and L Szpruch. On the geometry of stein variational gradient descent. *arXiv preprint arXiv:1912.00894*, 2019.

[DPBCD18] George Deligiannidis, Daniel Paulin, Alexandre Bouchard-Côté, and Arnaud Doucet. Randomized hamiltonian monte carlo as scaling limit of the bouncy particle sampler and dimension-free convergence rates. *arXiv preprint arXiv:1808.04299*, 2018.

[DPDK15] Arnaud Doucet, Michael K Pitt, George Deligiannidis, and Robert Kohn. Efficient implementation of markov chain monte carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2):295–313, 2015.

[DR11] Randal Douc and Christian P Robert. A vanilla rao–blackwellization of metropolis–hastings algorithms. *The Annals of Statistics*, 39(1):261–277, 2011.

[Dro14] Christopher C Drovandi. Pseudo-marginal algorithms with multiple cpus. , Queensland University of Technology, 2014.

[DS01] Jesper Dall and Paolo Sibani. Faster monte carlo simulations at low temperatures. the waiting time method. *Computer Physics Communications*, 141(2):260–267, 2001.

[DS17] Masoumeh Dashti and Andrew M. Stuart. The bayesian approach to inverse problems. In *Handbook of Uncertainty Quantification*, pages 311–428. Springer International Publishing, 2017.

[DSC98] Persi Diaconis and Laurent Saloff-Coste. What do we know about the metropolis algorithm? *Journal of Computer and System Sciences*, 57(1):20–36, 1998.

[DT12] Arnak S Dalalyan and Alexandre B Tsybakov. Sparse regression learning by aggregation and langevin monte-carlo. *Journal of Computer and System Sciences*, 78(5):1423–1443, 2012.

[ED05] David J Earl and Michael W Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[EGZ19] Andreas Eberle, Arnaud Guillin, and Raphael Zimmer. Couplings and quantitative contraction rates for langevin dynamics. *The Annals of Probability*, 47(4):1982–2010, 2019.

[EHL06] Y. Efendiev, T. Hou, and W. Luo. Preconditioning markov chain monte carlo simulations using coarse-scale models. *SIAM Journal on Scientific Computing*, 28(2):776–803, January 2006.

[EK09] Stewart N Ethier and Thomas G Kurtz. *Markov processes: characterization and convergence*, volume 282. John Wiley & Sons, 2009.

[ELVE19] Weinan E, Tiejun Li, and Eric Vanden-Eijnden. *Applied Stochastic Analysis*, volume 199. American Mathematical Soc., 2019.

[Eve09] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[Fea98] Paul Fearnhead. *Sequential Monte Carlo methods in filter theory*. PhD thesis, University of Oxford, 1998.

[Fin15] Axel Finke. *On extended state-space constructions for Monte Carlo methods*. PhD thesis, University of Warwick, 2015.

[FKLW97] Brendan J Frey, Frank R Kschischang, Hans-Andrea Loeliger, and Niclas Wiberg. Factor graphs and algorithms. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 35, pages 666–680. University of Illinois, 1997.

[Fol14] János Folláth. Gaussian sampling in lattice based cryptography. *Tatra Mountains Mathematical Publications*, 60(1):1–23, 2014.

[FS01] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier, 2001.

[FT06] Mark C Fuhs and David S Touretzky. A spin glass model of path integration in rat medial entorhinal cortex. *Journal of Neuroscience*, 26(16):4266–4276, 2006.

[Gag19] Philippe Gagnon. A step further towards automatic and efficient reversible jump algorithms. *arXiv preprint arXiv:1911.02089*, 2019.

[Gal16] Nikhil Galagali. *Bayesian inference of chemical reaction networks*. PhD thesis, Massachusetts Institute of Technology, 2016.

[GC11] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[GCS+13] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[GDM+14] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1242–1250, Bejing, China, 22–24 Jun 2014. PMLR.

[Gew89] John Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica: Journal of the Econometric Society*, pages 1317–1339, 1989.

[Gey98] Charles J Geyer. Markov chain monte carlo lecture notes. *Course notes, Spring Quarter*, 1998.

[GG84] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[GG06] Mike Giles and Paul Glasserman. Smoking adjoints: Fast monte carlo greeks. *Risk*, 19(1):88–92, 2006.

[GGML15] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

[GGZ18] Xuefeng Gao, Mert Gurbuzbalaban, and Lingjiong Zhu. Breaking reversibility accelerates langevin dynamics for global non-convex optimization. *arXiv preprint arXiv:1812.07725*, 2018.

[GIHLS20] Alfredo Garbuno-Inigo, Franca Hoffmann, Wuchen Li, and Andrew M Stuart. Interacting langevin diffusions: Gradient structure and ensemble kalman sampler. *SIAM Journal on Applied Dynamical Systems*, 19(1):412–441, 2020.

[Gil15] Michael B. Giles. Multilevel monte carlo methods. *Acta Numerica*, 24:259–328, April 2015.

[GINR19] Alfredo Garbuno-Inigo, Nikolas Nüsken, and Sebastian Reich. Affine invariant interacting langevin dynamics for bayesian inference. *arXiv preprint arXiv:1912.02859*, 2019.

[GL98] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[Gla13] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

[GLGG11] Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel gibbs sampling: From colored fields to thin junction trees. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 324–332, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[GŁR17] Flávio B Gonçalves, Krzysztof Łatuszyński, and Gareth O Roberts. Barker's algorithm for bayesian inference with intractable likelihoods. *Brazilian Journal of Probability and Statistics*, 31(4):732–745, 2017.

[GM97] Edward I George and Robert E McCulloch. Approaches for bayesian variable selection. *Statistica sinica*, pages 339–373, 1997.

[GM20] Philippe Gagnon and Florian Maire. Lifted samplers for partially ordered discrete state-spaces. *arXiv preprint arXiv:2003.05492*, 2020.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.

[Gre95] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[GRG94] Walter R Gilks, Gareth O Roberts, and Edward I George. Adaptive direction sampling. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 43(1):179–189, 1994.

[GS90] Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

[GS17] Matthew M. Graham and Amos J. Storkey. Asymptotically exact inference in differentiable generative models. *Electronic Journal of Statistics*, 11(2):5105–5164, 2017.

[GSC95] Alan E Gelfand, Sujit K Sahu, and Bradley P Carlin. Efficient parametrisations for normal linear mixed models. *Biometrika*, 82(3):479–488, 1995.

[GSS93] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, 140(2):107–113, 1993.

[GT94] Robert C Griffiths and Simon Tavaré. Ancestral inference in population genetics. *Statistical science*, pages 307–319, 1994.

[Gue19] Benjamin Guedj. A primer on pac-bayesian learning. *arXiv preprint arXiv:1901.05353*, 2019.

[Gus98] Paul Gustafson. A guided walk metropolis algorithm. *Statistics and computing*, 8(4):357–364, 1998.

[GW92] Walter R Gilks and Pascal Wild. Adaptive rejection sampling for gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(2):337–348, 1992.

[GW10] Jonathan Goodman and Jonathan Weare. Ensemble samplers with affine invariance. *Communications in applied mathematics and computational science*, 5(1):65–80, 2010.

[Ham13] John Hammersley. *Monte carlo methods*. Springer Science & Business Media, 2013.

[Has70] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 04 1970.

[HDSMR16] 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*, pages 1–9, 2016.

[HG14] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[HHMS93] Chii-Ruey Hwang, Shu-Yin Hwang-Ma, and Shuenn-Jyi Sheu. Accelerating gaussian diffusions. *The Annals of Applied Probability*, pages 897–913, 1993.

[HHMS05] Chii-Ruey Hwang, Shu-Yin Hwang-Ma, and Shuenn-Jyi Sheu. Accelerating diffusions. *The Annals of Applied Probability*, 15(2):1433–1444, 2005.

[Hil00] Martin Hildebrand. Rates of convergence of the diaconis-holmes-neal markov chain sampler. *preprint*, 2000.

[HKLC18] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2078–2087, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[HKR+16] James Howe, Ayesha Khalid, Ciara Rafferty, Francesco Regazzoni, and Máire O'Neill. On practical discrete gaussian samplers for lattice-based cryptography. *IEEE Transactions on Computers*, 67(3):322–334, 2016.

[HKRC18] Ya-Ping Hsieh, Ali Kavis, Paul Rolland, and Volkan Cevher. Mirrored langevin dynamics. In *Advances in Neural Information Processing Systems*, pages 2878–2887, 2018.

[HLCP02] Carine Hue, J-P Le Cadre, and Patrick Pérez. Sequential monte carlo methods for multiple target tracking and data fusion. *IEEE Transactions on signal processing*, 50(2):309–325, 2002.

[HLR94] Brian L Hammond, William A Lester, and Peter James Reynolds. *Monte Carlo methods in ab initio quantum chemistry*, volume 1. World Scientific, 1994.

[HR19] Jonathan H Huggins and Daniel M Roy. Sequential monte carlo as approximate sampling: bounds, adaptive resampling via infinity-ess, and an application to particle gibbs. *Bernoulli*, 25(1):584–622, 2019.

[HSR20] Liam Hodgkinson, Robert Salomone, and Fred Roosta. The reproducing stein kernel approach for post-hoc corrected sampling, 2020.

[HST99] Heikki Haario, Eero Saksman, and Johanna Tamminen. Adaptive proposal distribution for random walk metropolis algorithm. *Computational Statistics*, 14(3):375–396, 1999.

[HST01] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

[HST05] Heikki Haario, Eero Saksman, and Johanna Tamminen. Componentwise adaptation for high dimensional mcmc. *Computational Statistics*, 20(2):265–273, 2005.

[HWDF13] Firas Hamze, Ziyu Wang, and Nando De Freitas. Self-avoiding random dynamics on integer complex systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(1):9, 2013.

[Ise09] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Cambridge university press, 2009.

[Jäc02] Peter Jäckel. *Monte Carlo methods in finance*, volume 71. J. Wiley, 2002.

[Jay96] Edwin T Jaynes. *Probability theory: the logic of science*. Washington University St. Louis, MO, 1996.

[JD08] Adam M Johansen and Arnaud Doucet. A note on auxiliary particle filters. *Statistics & Probability Letters*, 78(12):1498–1504, 2008.

[JH00] Søren Fiig Jarner and Ernst Hansen. Geometric ergodicity of metropolis algorithms. *Stochastic processes and their applications*, 85(2):341–361, 2000.

[JL11] Andrew Jones and Ben Leimkuhler. Adaptive stochastic methods for sampling driven molecular systems. *The Journal of chemical physics*, 135(8):084125, 2011.

[JLM14] Benjamin Jourdain, Tony Lelièvre, and Błażej Miasojedow. Optimal scaling for the transient phase of metropolis hastings algorithms: the longtime behavior. *Bernoulli*, 20(4):1930–1978, 2014.

[JLM15] Benjamin Jourdain, Tony Lelièvre, and Błażej Miasojedow. Optimal scaling for the transient phase of the random walk metropolis algorithm: the mean-field limit. *The Annals of Applied Probability*, 25(4):2263–2300, 2015.

[JOA20] Pierre E. Jacob, John O'Leary, and Yves F. Atchadé. Unbiased markov chain monte carlo methods with couplings. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(3):543–600, May 2020.

[Joh17] Oliver Johnson. A discrete log-sobolev inequality under a bakry–émery type condition. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, volume 53, pages 1952–1970. Institut Henri Poincaré, 2017.

[JR07] Søren F Jarner and Gareth O Roberts. Convergence of heavy-tailed monte carlo markov chain algorithms. *Scandinavian Journal of Statistics*, 34(4):781–815, 2007.

[JS89] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM journal on computing*, 18(6):1149–1178, 1989.

[JS96] Mark Jerrum and Alistair Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996.

[JSW13] Matthew Johnson, James Saunderson, and Alan Willsky. Analyzing hogwild parallel gaussian gibbs sampling. In *Advances in Neural Information Processing Systems*, pages 2715–2723, 2013.

[JT03] Søren F Jarner and Richard L Tweedie. Necessary conditions for geometric and polynomial ergodicity of random-walk-type. *Bernoulli*, 9(4):559–578, 2003.

[JV01] Kamal Jain and Vijay V Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM (JACM)*, 48(2):274–296, 2001.

[JW95] Steven L Jacques and Lihong Wang. Monte carlo modeling of light transport in tissues. In *Optical-thermal response of laser-irradiated tissue*, pages 73–100. Springer, 1995.

[KCHK20] Daniel Krefl, Stefano Carrazza, Babak Haghighat, and Jens Kahlen. Riemann-theta boltzmann machine. *Neurocomputing*, 2020.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[Kit96] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[KK17] Bo'az Klartag and Alexander V Kolesnikov. Remarks on curvature in the transportation metric. *Analysis Mathematica*, 43(1):67–88, 2017.

[KK19] Bo'az Klartag and Alexander V Kolesnikov. Extremal kähler–einstein metric for two-dimensional convex bodies. *The Journal of Geometric Analysis*, 29(3):2347–2373, 2019.

[Kla14] Bo'az Klartag. Logarithmically-concave moment measures i. In *Geometric Aspects of Functional Analysis*, pages 231–260. Springer, 2014.

[Kle20] Tore Selland Kleppe. Connecting the dots: Towards continuous time hamiltonian monte carlo, 2020.

[KM16] Alexander V Kolesnikov and Emanuel Milman. Riemannian metrics on convex sets with applications to poincaré and log-sobolev inequalities. *Calculus of Variations and Partial Differential Equations*, 55(4):77, 2016.

[Kol14] Alexander V Kolesnikov. Hessian metrics, cd (k, n)-spaces, and optimal transportation of log-concave measures. *Discrete & Continuous Dynamical Systems - A*, 34(4):1511–1532, 2014.

[KOS18] Juan Kuntz, Michela Ottobre, and Andrew M Stuart. Non-stationary phase of the mala algorithm. *Stochastics and Partial Differential Equations: Analysis and Computations*, 6:446–499, 2018.

[KOS19] Juan Kuntz, Michela Ottobre, and Andrew M Stuart. Diffusion limit for the random walk metropolis algorithm out of stationarity. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, volume 55, pages 1599–1648. Institut Henri Poincaré, 2019.

[Kra06] Werner Krauth. *Statistical mechanics: algorithms and computations*, volume 13. OUP Oxford, 2006.

[Kru94] Paul Krugman. Complex landscapes in economic geography. *The American Economic Review*, 84(2):412–416, 1994.

[KSJ+16] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[KT12] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[KW09] Malvin H Kalos and Paula A Whitlock. *Monte Carlo methods*. John Wiley & Sons, 2009.

[KY03] Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

[Lau96] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[LB14a] David P Landau and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2014.

[LB14b] Cong Ling and Jean-Claude Belfiore. Achieving awgn channel capacity with lattice gaussian coding. *IEEE Transactions on Information Theory*, 60(10):5918–5929, 2014.

[LBGY16] Christopher Lester, Ruth E Baker, Michael B Giles, and Christian A Yates. Extending the multi-level method for the simulation of stochastic biological systems. *Bulletin of mathematical biology*, 78(8):1640–1677, 2016.

[LC06] Richard A Levine and George Casella. Optimizing random scan gibbs samplers. *Journal of Multivariate Analysis*, 97(10):2071–2100, 2006.

[LEc09] Pierre LEcuyer. Quasi-monte carlo methods with applications in finance. *Finance and Stochastics*, 13(3):307–349, 2009.

[Lee12] Anthony Lee. On the choice of mcmc kernels for approximate bayesian computation with smc samplers. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12. IEEE, 2012.

[Liu08] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[Liv15] Samuel Livingstone. Geometric ergodicity of the random walk metropolis with position-dependent proposal covariance. *arXiv preprint arXiv:1507.05780*, 2015.

[LJN+17] Fredrik Lindsten, Adam M Johansen, Christian A Naesseth, Bonnie Kirkpatrick, Thomas B Schön, JAD Aston, and Alexandre Bouchard-Côté. Divide-and-conquer with sequential monte carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

[LL10] Qing Li and Nan Lin. The bayesian elastic net. *Bayesian analysis*, 5(1):151–170, 2010.

[LŁ14] Anthony Lee and Krzysztof Łatuszyński. Variance bounding and geometric ergodicity of markov chain monte carlo kernels for approximate bayesian computation. *Biometrika*, 101(3):655–671, 2014.

[LL16] Qiang Liu and Jason D. Lee. Black-box importance sampling, 2016.

[LLW00] Jun S Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[LM08] Fabrizio Leisen and Antonietta Mira. An extension of peskun and tierney orderings to continuous time markov chains. *Statistica Sinica*, pages 1641–1651, 2008.

[LM16] Ben Leimkuhler and Charles Matthews. *Molecular Dynamics.* Springer, 2016.

[LMW18] Benedict Leimkuhler, Charles Matthews, and Jonathan Weare. Ensemble preconditioning for markov chain monte carlo simulation. *Statistics and Computing*, 28(2):277–290, 2018.

[LNP13] Tony Lelièvre, Francis Nier, and Grigorios A Pavliotis. Optimal non-reversible linear drift for the convergence to equilibrium of a diffusion. *Journal of Statistical Physics*, 152(2):237–274, 2013.

[LNT09] Ben Leimkuhler, Emad Noorizadeh, and Florian Theil. A gentle stochastic thermostat for molecular dynamics. *Journal of Statistical Physics*, 135(2):261–277, 2009.

[LP13] Andreas Löpker and Zbigniew Palmowski. On time reversal of piecewise deterministic markov processes. *Electronic Journal of Probability*, 18, 2013.

[LR09] Benedict Leimkuhler and Sebastian Reich. A metropolis adjusted nosé-hoover thermostat. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43(4):743–755, 2009.

[LSD+03] Kyeong Eun Lee, Naijun Sha, Edward R Dougherty, Marina Vannucci, and Bani K Mallick. Gene selection: a bayesian variable selection approach. *Bioinformatics*, 19(1):90–97, 2003.

[LSZ15] Kody Law, Andrew Stuart, and Kostas Zygalakis. Data assimilation. *Cham, Switzerland: Springer*, 2015.

[Lux18] Ivan Lux. *Monte Carlo particle transport methods.* CRC press, 2018.

[LWK95] Jun S Liu, Wing H Wong, and Augustine Kong. Covariance structure and convergence rate of the gibbs sampler with various scans. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):157–169, 1995.

[LWME19] Xuechen Li, Yi Wu, Lester Mackey, and Murat A Erdogdu. Stochastic runge-kutta accelerates langevin monte carlo and beyond. In H. Wallach, H. Larochelle, A. Beygelzimer, F. D Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7748–7760. Curran Associates, Inc., 2019.

[LZ19] Samuel Livingstone and Giacomo Zanella. On the robustness of gradient-based mcmc algorithms. *arXiv preprint arXiv:1908.11812*, 2019.

[Mai18] Florian Maire. Dereversibilizing metropolis-hastings: simple implementation of non-reversible mcmc methods. , University of Montreal, 2018.

[MB88] Toby J Mitchell and John J Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

[MCF15] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

[McL11] Don L McLeish. *Monte Carlo simulation and finance*, volume 276. John Wiley & Sons, 2011.

[MDS17] Manon Michel, Alain Durmus, and Stéphane Sénécal. Forward event-chain monte carlo: Fast sampling by randomness control in irreversible markov chains, 2017.

[MFCW19] Yi-An Ma, Emily B Fox, Tianqi Chen, and Lei Wu. Irreversible samplers from jump and continuous markov processes. *Statistics and Computing*, 29(1):177–202, 2019.

[Mic16] Manon Michel. *Irreversible Markov chains by the factorized Metropolis filter: algorithms and applications in particle systems and spin models*. PhD thesis, Ecole Normale Superieure de Paris-ENS Paris, 2016.

[MKK14] Manon Michel, Sebastian C Kapfer, and Werner Krauth. Generalized event-chain monte carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps. *The Journal of chemical physics*, 140(5):054116, 2014.

[MM17] Ioannis Mitliagkas and Lester Mackey. Improving gibbs sampler scan quality with dogs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2469–2477. JMLR. org, 2017.

[MMN+13] Guri I Marchuk, Gennadi A Mikhailov, MA Nazareliev, Radzmik A Darbinjan, Boris A Kargin, and Boris S Elepov. *The Monte Carlo methods in atmospheric optics*, volume 12. Springer, 2013.

[MMPT03] Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

[MMW+19] Wenlong Mou, Yi-An Ma, Martin J Wainwright, Peter L Bartlett, and Michael I Jordan. High-order langevin diffusion yields an accelerated mcmc algorithm. *arXiv preprint arXiv:1908.10859*, 2019.

[Mon19] Pierre Monmarché. Kinetic walks for sampling. *arXiv preprint arXiv:1903.00550*, 2019.

[MRR+53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[MS17] Oren Mangoubi and Aaron Smith. Rapid mixing of hamiltonian monte carlo on strongly log-concave distributions. *arXiv preprint arXiv:1708.07114*, 2017.

[MT12] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[NB99] M Newman and G Barkema. *Monte carlo methods in statistical physics chapter 1-4*. Oxford University Press: New York, USA, 1999.

[Nea04] Radford M Neal. Improving asymptotic variance of mcmc estimators: Non-reversible chains are better. *arXiv preprint math/0407281*, 2004.

[Nea11] Radford Neal. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[NLS14] Christian Andersson Naesseth, Fredrik Lindsten, and Thomas B Schön. Sequential monte carlo for graphical models. In *Advances in Neural Information Processing Systems*, pages 1862–1870, 2014.

[NLS19] Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Elements of sequential monte carlo. *Foundations and Trends® in Machine Learning*, 12(3):187–306, 2019.

[NMA14] Robert Nishihara, Iain Murray, and Ryan P Adams. Parallel mcmc with generalized elliptical slice sampling. *The Journal of Machine Learning Research*, 15(1):2087–2112, 2014.

[NP19] Nikolas Nüsken and Grigorios Pavliotis. Constructing sampling schemes via coupling: Markov semigroups and optimal transport. *SIAM/ASA Journal on Uncertainty Quantification*, 7(1):324–382, 2019.

[NR19] Nikolas Nüsken and Sebastian Reich. Note on interacting langevin diffusions: Gradient structure and ensemble kalman sampler by garbuno-inigo, hoffmann, li and stuart. *arXiv preprint arXiv:1908.10890*, 2019.

[Ora19] Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

[PA20] Joonha Park and Yves Atchadé. Markov chain monte carlo algorithms with sequential proposals. *Statistics and Computing*, pages 1–21, 2020.

[Pap03] Omiros Papaspiliopoulos. *Non-centered parameterisations for data augmentation and hierarchical models*. PhD thesis, Lancaster University, 2003.

[Pav14] Grigorios A Pavliotis. *Stochastic processes and applications: diffusion processes, the Fokker-Planck and Langevin equations*, volume 60. Springer, 2014.

[PB20] Thomas P. Prescott and Ruth E. Baker. Multifidelity approximate bayesian computation. *SIAM/ASA Journal on Uncertainty Quantification*, 8(1):114–138, January 2020.

[Per15] Marcelo Pereyra. Proximal markov chain monte carlo algorithms. *Statistics and Computing*, 26(4):745–760, May 2015.

[Pes73] Peter H Peskun. Optimum monte-carlo sampling using markov chains. *Biometrika*, 60(3):607–612, 1973.

[PFJR16] Murray Pollock, Paul Fearnhead, Adam M Johansen, and Gareth O Roberts. The scalable langevin exact algorithm: Bayesian inference for big data. *arXiv preprint arXiv:1609.03436*, 2016.

[PGCP17] Ari Pakman, Dar Gilboa, David Carlson, and Liam Paninski. Stochastic bouncy particle sampler. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2741–2750, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[PGT94] Vijay S Pande, A Yu Grosberg, and Toyoichi Tanaka. Folding thermodynamics and kinetics of imprinted renaturable heteropolymers. *The Journal of chemical physics*, 101(9):8246–8257, 1994.

[PJR16] Murray Pollock, Adam M. Johansen, and Gareth O. Roberts. On the exact and ε-strong simulation of (jump) diffusions. *Bernoulli*, 22(2):794–856, May 2016.

[Pol15] Murray Pollock. On the exact simulation of (jump) diffusion bridges. In *2015 Winter Simulation Conference (WSC)*, pages 348–359. IEEE, 2015.

[PP13] Ari Pakman and Liam Paninski. Auxiliary-variable exact hamiltonian monte carlo samplers for binary distributions. In *Advances in neural information processing systems*, pages 2490–2498, 2013.

[PR09] Ilya Prigogine and Stuart A Rice. *Monte Carlo Methods in Chemical Physics*, volume 228. John Wiley & Sons, 2009.

[Pra16] Dennis Prangle. Lazy abc. *Statistics and Computing*, 26(1–2):171–185, January 2016.

[PRS07] Omiros Papaspiliopoulos, Gareth O Roberts, and Martin Sköld. A general framework for the parametrization of hierarchical models. *Statistical Science*, pages 59–73, 2007.

[PRZ20] Omiros Papaspiliopoulos, Gareth O Roberts, and Giacomo Zanella. Scalable inference for crossed random effects models. *Biometrika*, 107(1):25–40, 2020.

[PS10] Nicholas G Polson and James G Scott. Shrink globally, act locally: Sparse bayesian regularization and prediction. *Bayesian statistics*, 9(501-538):105, 2010.

[PV17] Juho Piironen and Aki Vehtari. Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics*, 11(2):5018–5051, 2017.

[RBC15] Christian Robert and Alexandre Bouchard-Cote. Bouncy particle sampler, November 2015.

[RBS15] Luc Rey-Bellet and Konstantinos Spiliopoulos. Irreversible langevin samplers and variance reduction: a large deviations approach. *Nonlinearity*, 28(7):2081, 2015.

[RC13] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[RC15] Sebastian Reich and Colin Cotter. *Probabilistic forecasting and Bayesian data assimilation*. Cambridge University Press, 2015.

[RCALJ82] Peter J Reynolds, David M Ceperley, Berni J Alder, and William A Lester Jr. Fixed-node quantum monte carlo for molecules. *The Journal of Chemical Physics*, 77(11):5593–5603, 1982.

[RCC+20] Marina Riabiz, Wilson Chen, Jon Cockayne, Pawel Swietach, Steven A. Niederer, Lester Mackey, and Chris. J. Oates. Optimal thinning of mcmc output, 2020.

[Reb83] Claudio Rebbi. *Lattice gauge theories and Monte Carlo simulations*. World scientific, 1983.

[RF13] Paul K Romano and Benoit Forget. The openmc monte carlo particle transport code. *Annals of Nuclear Energy*, 51:274–281, 2013.

[RG97] Sylvia Richardson and Peter J Green. On bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society: series B (statistical methodology)*, 59(4):731–792, 1997.

[RGG97] Gareth O Roberts, Andrew Gelman, and Walter R Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[RK13] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[Rob07] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[Rom07] JC Romers. Discrete gauge theories in two spatial dimensions. Master's thesis, Universiteit van Amsterdam, 2007.

[RR98] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

[RR01] Gareth O Roberts and Jeffrey Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

[RR09] Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, January 2009.

[RR15] Gareth O Roberts and Jeffrey S Rosenthal. Surprising convergence properties of some simple gibbs samplers under various scans. *International Journal of Statistics and Probability*, 5(1):51–60, 2015.

[RT96] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341, December 1996.

[SBCDD19] Saifuddin Syed, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Non-reversible parallel tempering: an embarassingly parallel mcmc scheme. *arXiv preprint arXiv:1905.02939*, 2019.

[SBF18] Julien Stoehr, Alan Benson, and Nial Friel. Noisy hamiltonian monte carlo for doubly intractable distributions. *Journal of Computational and Graphical Statistics*, 28(1):220–232, October 2018.

[SC13] Christian Schäfer and Nicolas Chopin. Sequential monte carlo on large binary sampling spaces. *Statistics and Computing*, 23(2):163–184, 2013.

[SDMD14] Jascha Sohl-Dickstein, Mayur Mudigonda, and Michael R DeWeese. Hamiltonian monte carlo without detailed balance. *arXiv preprint arXiv:1409.5191*, 2014.

[She16] Chris Sherlock. Optimal scaling for the pseudo-marginal random walk metropolis: insensitivity to the noise generating mechanism. *Methodology and Computing in Applied Probability*, 18(3):869–884, 2016.

[Sim17] Umut Simsekli. Fractional langevin monte carlo: Exploring lévy driven stochastic differential equations for markov chain monte carlo. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3200–3209. JMLR. org, 2017.

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.

[SL19] Ruoqi Shen and Yin Tat Lee. The randomized midpoint method for log-concave sampling. In *Advances in Neural Information Processing Systems*, pages 2098–2109, 2019.

[Smi13] Adrian Smith. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[Sok97] Alan Sokal. Monte carlo methods in statistical mechanics: foundations and new algorithms. In *Functional integration*, pages 131–192. Springer, 1997.

[SR93] Adrian FM Smith and Gareth O Roberts. Bayesian computation via the gibbs sampler and related markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(1):3–23, 1993.

[SRL10] Gabriel Stoltz, Mathias Rousset, , and Tony Lelievre. *Free energy computations: A mathematical perspective*. World Scientific, 2010.

[ST17] Chris Sherlock and Alexandre H Thiery. A discrete bouncy particle sampler. *arXiv preprint arXiv:1707.05200*, 2017.

[Ste00] Matthew Stephens. Bayesian analysis of mixture models with an unknown number of components—an alternative to reversible jump methods. *the Annals of Statistics*, 28(1):40–74, 2000.

[Str10] Ingvar Strid. Efficient parallelisation of metropolis–hastings algorithms using a prefetching approach. *Computational Statistics & Data Analysis*, 54(11):2814–2835, November 2010.

[Stu10] Andrew M Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

[SW86] Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

[SWJ93] Andrew Sohn, Zhihong Wu, and Xue Jin. Parallel simulated annealing by generalized speculative computation. In *Proceedings of 1993 5th IEEE Symposium on Parallel and Distributed Processing*, pages 416–419. IEEE, 1993.

[SZTG20] Umut Simsekli, Lingjiong Zhu, Yee Whye Teh, and Mert Gürbüzbalaban. Fractional underdamped langevin dynamics: Retargeting sgd with momentum under heavy-tailed gradient noise. *arXiv preprint arXiv:2002.05685*, 2020.

[TCV11] Konstantin S Turitsyn, Michael Chertkov, and Marija Vucelja. Irreversible monte carlo algorithms for efficient sampling. *Physica D: Nonlinear Phenomena*, 240(4-5):410–414, 2011.

[Tie98] Luke Tierney. A note on metropolis-hastings kernels for general state spaces. *Annals of applied probability*, pages 1–9, 1998.

[TMM20] Xin T Tong, Mathias Morzfeld, and Youssef M Marzouk. Mala-within-gibbs samplers for high-dimensional distributions with sparse conditional structure. *SIAM Journal on Scientific Computing*, 42(3):A1765–A1788, 2020.

[TSD20] Alexander Terenin, Daniel Simpson, and David Draper. Asynchronous gibbs sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 144–154, 2020.

[TT18] Alexander Terenin and Daniel Thorngren. A piecewise deterministic markov process via (r, θ) swaps in hyperspherical coordinates. *arXiv preprint arXiv:1807.00420*, 2018.

[TVKWD16] Erik H Thiede, Brian Van Koten, Jonathan Weare, and Aaron R Dinner. Eigenvector method for umbrella sampling enables error analysis. *The Journal of chemical physics*, 145(8):084115, 2016.

[TY17] Michalis K Titsias and Christopher Yau. The hamming ball sampler. *Journal of the American Statistical Association*, 112(520):1598–1611, 2017.

[UML14] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pages 467–475, 2014.

[VBCDD17] Paul Vanetti, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Piecewise deterministic markov chain monte carlo. *arXiv preprint arXiv:1707.05296*, 2017.

[Vea97] Eric Veach. *Robust Monte Carlo methods for light transport simulation*, volume 1610. Stanford University PhD thesis, 1997.

[VGŁR20] Dootika Vats, Flávio Gonçalves, Krzysztof Łatuszyński, and Gareth O Roberts. Efficient bernoulli factory mcmc for intractable likelihoods. *arXiv preprint arXiv:2004.07471*, 2020.

[VGS20] Jacob Vorstrup Goldman and Sumeetpal Singh. Blocking the bouncy particle sampler, with applications to state-space models. in preparation, 2020.

[Vih12] Matti Vihola. Robust adaptive metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, 22(5):997–1008, 2012.

[VLCR15] Peter Jan Van Leeuwen, Yuan Cheng, and Sebastian Reich. *Nonlinear data assimilation*. Springer, 2015.

[VM20] Marie Vialaret and Florian Maire. On the convergence time of some non-reversible markov chain monte carlo methods. *Methodology and Computing in Applied Probability*, pages 1–39, 2020.

[VPD19] Maxime Vono, Daniel Paulin, and Arnaud Doucet. Efficient mcmc sampling with dimension-free convergence rate using admm-type splitting. *arXiv preprint arXiv:1905.11937*, 2019.

[VPZ19] Luis Vargas, Marcelo Pereyra, and Konstantinos C. Zygalakis. Accelerating proximal markov chain monte carlo by using an explicit stabilised method, 2019.

[VSD03] Ba-Ngu Vo, Sumeetpal Singh, and Arnaud Doucet. Sequential monte carlo implementation of the phd filter for multi-target tracking. In *Proc. Int'l Conf. on Information Fusion*, pages 792–799, 2003.

[Vuc16] Marija Vucelja. Lifting—a nonreversible markov chain monte carlo algorithm. *American Journal of Physics*, 84(12):958–968, 2016.

[VW77] JP Valleau and SG Whittington. A guide to monte carlo for statistical mechanics: 1. highways. In *Statistical mechanics*, pages 137–168. Springer, 1977.

[Wil07] Darren J Wilkinson. Bayesian methods in bioinformatics and computational systems biology. *Briefings in bioinformatics*, 8(2):109–116, 2007.

[Wil18] Darren J Wilkinson. *Stochastic modelling for systems biology*. CRC press, 2018.

[WL20a] Antoine Wehenkel and Gilles Louppe. Graphical normalizing flows. *arXiv preprint arXiv:2006.02548*, 2020.

[WL20b] Antoine Wehenkel and Gilles Louppe. You say normalizing flows i see bayesian networks. *arXiv preprint arXiv:2006.00866*, 2020.

[Woo10] Simon N Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 2010.

[WPRS19] Andi Q Wang, Murray Pollock, Gareth O Roberts, and David Steinsaltz. Regeneration-enriched markov processes with application to monte carlo. *arXiv preprint arXiv:1910.05037*, 2019.

[WR17] Changye Wu and Christian P Robert. Generalized bouncy particle sampler. *arXiv preprint arXiv:1706.04781*, 2017.

[WR20] Changye Wu and Christian P Robert. Coordinate sampler: a non-reversible gibbs-like mcmc sampler. *Statistics and Computing*, 30(3):721–730, 2020.

[WRS18] Changye Wu, Christian P Robert, and Julien Stoehr. Discrete piecewise-deterministic markov processes using hamiltonian dynamics. *In Preparation*, 2018.

[WRS19] Andi Q Wang, Gareth O Roberts, and David Steinsaltz. An approximation scheme for quasi-stationary distributions of killed diffusions. *Stochastic Processes and their Applications*, 2019.

[WT11] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress.

[Yeh98] I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

[YM11] Yaming Yu and Xiao-Li Meng. To center or not to center: That is not the question—an ancillarity–sufficiency interweaving strategy (asis) for boosting mcmc efficiency. *Journal of Computational and Graphical Statistics*, 20(3):531–570, 2011.

[Zan19] Giacomo Zanella. Informed proposals for local mcmc in discrete spaces. *Journal of the American Statistical Association*, pages 1–35, 2019.

[ZR19] Giacomo Zanella and Gareth Roberts. Scalable importance tempering and bayesian variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(3):489–517, 2019.