Monte Carlo Methods based on Novel Classes of Regeneration-enriched Markov processes

Hector McKimm

Thesis submitted for the degree of:

Doctor of Philosophy in Statistics (Oxford-Warwick Programme)

University of Warwick

Department of Statistics

September 2022

Contents

List of Algorithms

List of Figures

Acknowledgements

I am immensely grateful to a number of people who have helped me over the course of my PhD. My partner, Ellen, and my parents, Dan and Sarah, have been extremely supportive. Especially during the worst months of the pandemic, which was a difficult time for everyone, I really appreciated their help. Likewise my sisters, Connie and Alice, have been very supportive and I wish Alice good luck in her own PhD. Ellen's parents and brother have been very encouraging and I thank them for their kindness. I am grateful for old friends and for those made along the way – other students on the Oxford-Warwick Statistics Programme, at the University of Warwick and those met at conferences and workshops.

I would like to thank my supervisors Gareth Roberts and Murray Pollock for their time, effort and insight in guiding my research, as well as my collaborators Andi Wang and Christian Robert, with whom I greatly enjoyed working on Adaptive Restore. It was an honour to work with the University of Warwick's COVID-19 Modelling Group for a period of a few months. I thank the University of Warwick, the Oxford-Warwick Statistics Programme and the Engineering and Physical Sciences Research Council for the opportunity to work towards completing a PhD.

Declaration

The thesis is my own work, except where it contains work based on collaborative research. I confirm that the thesis has not been submitted for a degree at another university.

The material of Chapter 5 on Adaptive Restore features in McKimm et al. (2022). I lead the development of the methodology for this paper in collaboration with the other authors, wrote the entire paper except for Section 4 (which was written by Andi Wang), wrote all the relevant code and performed all computer experiments.

The methodology of Chapter 6 was developed under the supervision of my supervisors Gareth Roberts and Murray Pollock.

Abstract

Enriching some underlying continuous-time Markov process with regenerations from a fixed regeneration distribution μ at a particular regeneration rate κ results in a Markov process that has a target distribution π as its invariant distribution. Firstly, we introduce a method for adapting the regeneration distribution, which allows a significantly smaller regeneration rate to be used, which makes simulation feasible for a wider range of target distributions. The regeneration distribution is adapted on-the-fly, by adding point masses to it. Secondly, we show that a class of non-π-invariant jump processes, which are defined on an augmented state-space and have a jump chain transition kernel corresponding to a deterministic, invertible mapping, may be enriched with regenerations so that the resulting process is π-invariant. Since the underlying jump process does not need to be π-invariant, its dynamics may be chosen to use gradient information to guide the process to areas of high probability mass, which makes the sampler a promising algorithm for multi-modal target distributions.

Chapter 1

Introduction

In Bayesian statistics, the unknown parameters of a statistical model are regarded as random variables. The statistician specifies a prior distribution for these variables according to their pre-existing beliefs, then updates their beliefs by using Bayes' rule to derive a posterior distribution that takes into account observations. The expectation of functions of the model's parameters, with respect to the posterior distribution, provides useful information. The process of fitting the model to the data and learning about the posterior distribution is called Bayesian inference, see for example page 1 of Gelman et al. (2013).

Statistical models have become more complicated, involving more parameters, more intricate structures and more data. To do inference on these models, a number of algorithms have been developed, which are grouped together as techniques for *Bayesian computation* and includes Monte Carlo methods (see for example Robert and Casella (2004)), the subject of this thesis.

For π the posterior distribution of interest (referred to as the *target* distribution) and f some function, the Monte Carlo method (Metropolis and Ulam, 1949) approximates $\mathbb{E}_\pi[f(X)]$ as $n^{-1}\sum_i f(X_i)$ for samples $X_1, X_2, \ldots, X_n$ from π. The method is asymptotically exact, in the sense that as n tends to infinity, the approximation becomes exact – a property that is not shared by all techniques for Bayesian computation (Minka, 2001; Blei et al., 2017).

When samples from π can't be generated directly, one of the mostly widely used methods is Markov chain Monte Carlo (MCMC), in particular the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970). MCMC generates dependent samples by simulating a Markov chain with invariant distribution π. The fundamental building-block of a Markov chain is a Markov transition kernel, which specifies how the chain moves from one state to the next. Multiple transition kernels can be combined, which allows kernels with different properties to be used, such as the next state tending to be in a small vicinity of the current state (a *local* move) or for the next state to be independent of the current state and anywhere in the state space (a *global* move). However, in the framework of discrete-time MCMC, each transition kernel must be π-invariant. There is no way for transition kernels which individually are not π-invariant to somehow *compensate* for each other so that the resulting Markov chain is π-invariant. The Metropolis-Hastings algorithm uses a *reversible* Markov transition kernel, meaning the *detailed balance* condition is satisfied, which ensures π-invariance. There is evidence that non-reversible Markov chains are superior (Neal, 2004; Suwa and Todo, 2010; Turitsyn et al., 2011; Chen and Hwang, 2013) and hence interest in designing non-reversible MCMC methods. Another potential way to improve MCMC methods is to use *regeneration*, times at which the Markov chain effectively starts again, see for example Chapter 6 of (Asmussen, 2003). Unfortunately, for the current methodology for regenerative MCMC (Mykland et al., 1995; Gilks et al., 1998; Brockwell and Kadane, 2005), which is based on Nummelin's splitting technique (Nummelin, 1978), regenerations tend to recede exponentially as the dimension increases.

Wang et al. (2021) designed a class of Markov process that is suitable for Monte Carlo, combines different dynamics that individually are not π-invariant but together compensate for each other so that the process is π-invariant, is non-reversible and regenerative. The class is called the *Restore process* and is defined by enriching some underlying continuous-time Markov process with regenerations from a *regeneration distribution* μ at rate κ, called the *regeneration rate.* Given the underlying process and regeneration distribution, the regeneration rate may be chosen so that the invariant distribution of the enriched process is π. Hence the Restore process may be used for Monte Carlo, in which context it is referred to as the *Restore sampler.*

Wang et al. (2021) made an excellent contribution in specifying a flexible framework for enriching a Markov process with regeneration so that the resulting process is π-invariant. However, an issue with the Restore sampler is that when μ is a poor approximation of π, the rate κ can become extremely large. This results in the sampler being inefficient, since the process frequently regenerates into areas with very low probability mass. Secondly, when the underlying process is a jump process, the expression for the regeneration rate involves an integral with respect to π, which can't always be evaluated. When the underlying jump process is already π-invariant, it's possible to evaluate the regeneration rate, but then regeneration is in no way compensating for the dynamics of the underlying process.

There are two main contributions in this thesis. Firstly, it is shown how μ may be *adapted* so that a regeneration rate which is as small as possible may be used throughout simulation of the process. The distribution μ is not adapted so that it converges to π, but to a closely related distribution called the *minimal regeneration distribution.* Multiple examples are used to investigate the performance of the corresponding algorithm, referred to as *Adaptive Restore* (McKimm et al., 2022). We find that Adaptive Restore is most suitable for unimodal distributions, since the nature of the adaptive mechanism results in slow convergence of the process when the target is multimodal.

Secondly, we show how a non-π-invariant jump process may be enriched with regenerations so that the resulting jump process is π-invariant. Such a process is called a *Jump Process Adjusted with Regenerations* (Jumpar) because regeneration adjusts the invariant distribution of the jump process so that it is π. A jump process has a holding rate, which determines how long the process tends to remain in each state for, as well as a Markov transition kernel, which describes how the state of the jump process changes at jump times. Two choices for the holding rate are suggested, one being constant and the other depending on μ and π. Three choices for the Markov transition kernel are considered, which relate to random-direction, Hamiltonian and conformal Hamiltonian dynamics. The random-direction dynamics are considered to help to visualise the proposed method. Hamiltonian dynamics are experimented with, because they are apt at finding a next state that is far from the current state, yet has similar probability density; this property helps to explain the effectiveness of Hamiltonian Monte Carlo (Duane et al., 1987; Neal et al., 2011). Lastly conformal Hamiltonian dynamics are considered, since these use gradient information to guide the sampler towards areas of high probability mass, which may improve sampling efficiency. A Jumpar with conformal Hamiltonian dynamics may be particularly well suited to sampling multimodal distributions, because the dynamics use gradient information to guide the process towards the modes.

The thesis is structured as follows. Chapter 2 introduces basic Monte Carlo methods and gives more detail on the issues with current techniques. The Restore process is a continuous-time Markov process; Chapter 3 presents some basic properties of continuous-time Markov processes and describes some specific processes including jump processes, Poisson processes (see for example Kingman (1992)) and Brownian motion (see for instance Mörters and Peres (2010)), as well as simulation methods (Devroye (1986) is a good resource). The Restore process Wang et al. (2021) is covered in Chapter 4, which also makes a minor novel contribution by showing that a Restore process may be used to estimate the normalising constant of an unnormalised target distribution. The main contributions of the thesis are made in Chapters 5 and 6 on Adaptive Restore (McKimm et al., 2022) and Jumpar respectively. Chapter 7 concludes.

Chapter 2

Monte Carlo and Related Methods

Monte Carlo methods, see for example Robert and Casella (2004), are used to compute high dimensional integrals exactly, see Section 1.1 of Liu (2008) for instance. Let X be a random variable on a state space $\mathcal{X}$ with density π. Many problems in Bayesian statistics involve computing the expectation of some function f of random variable X. This expectation may be expressed as an integral:

$$\mathbb{E}_\pi[f(X)] = \int_{\mathcal{X}} f(x)\pi(x)dx. \tag{2.1}$$

We will sometimes use notation:

$$\pi[f] := \mathbb{E}_\pi[f(X)].$$

When an analytic solution is not available, numerical methods can instead be used to compute the expectation. Quadrature methods (see Chapter 4 of Press et al. (2007) for example) approximate the integral as a sum of the integrand evaluated at a number of points. For example, when $\mathcal{X} = [0, 1]$, the Riemann approximation of (2.1) is

$$\int_0^1 f(x)\pi(x)dx \approx \sum_{i=1}^{n} f(x_i)\pi(x_i)$$

for $x_i = i/n; i = 1, \ldots, n$. The Riemann approximation has an error rate of $\mathcal{O}(1/n)$: there exists a constant M such that the error is less than M/n. However, in d dimensions $\mathcal{O}(n^d)$ points are required to achieve an $\mathcal{O}(1/n)$ error rate, thus deterministic numerical integration doesn't scale well to problems in high dimension.

The Monte Carlo method (Metropolis and Ulam, 1949) is to simulate n samples $X_0, \ldots, X_{n-1}$ from π then approximate $\pi[f]$ as:

$$\hat{\pi}_n[f] := \frac{1}{n}\sum_{i=0}^{n-1} f(X_i). \tag{2.2}$$

When the samples are independent, the Strong Law of Large Numbers guarantees that the approximation becomes exact as $n \to \infty$ (see Section 3.2 of Robert and Casella (2004) for example). The Central Limit Theorem states that, when $\sigma^2 := \text{Var}_\pi[f(X)] < \infty$,

$$\sqrt{n}\big(\hat{\pi}_n[f] - \mathbb{E}_\pi[f(X)]\big) \to \mathcal{N}(0, \sigma^2) \tag{2.3}$$

in distribution. The estimator is unbiased and has standard deviation $\sigma/\sqrt{n}$. Thus the rate of convergence of the estimator is $\mathcal{O}(1/\sqrt{n})$, which is independent of dimension, making Monte Carlo more suitable to high-dimensional integration than deterministic methods. A caveat is that σ^2 can be very large for high-dimensional problems.

The question of how to compute integral (2.1) then becomes how to sample from π, the target distribution. This chapter will cover some foundational sampling methods relevant to the novel contributions in later chapters. As a starting point, we assume we are able to simulate uniformly on $[0, 1]$. Section 2.1 details methods for obtaining independent and identically distributed samples from π via direct sampling. Importance Sampling, which generates independent weighted samples, is covered in Section 2.2. A method for generating a sequence of dependent samples, Markov chain Monte Carlo (MCMC), is introduced in Section 2.3. Estimating normalising constants is useful for Bayesian model comparison and is presented in Section 2.4. Finally, Section 2.5 covers some methods related to Monte Carlo methods that are particularly pertinent to this thesis.

2.1 Direct Sampling

When π is relatively simple, it is possible to generate samples from it directly. That is, there is no need to use weighted samples, as in Importance sampling (Section 2.2), or to generate a Markov chain, as in MCMC (Section 2.3).

2.1.1 Inversion Sampling

The most fundamental method for sampling one-dimensional distributions is to use the probability integral transform. Suppose $X \sim \pi$ has cumulative distribution function Π. If $U \sim \mathcal{U}[0, 1]$, then random variable $\Pi^{-1}(U) \sim \pi$.

2.1.2 Sampling via Transformations

When X may be expressed in terms of some *base* random variables, then X may be simulated by first sampling the base random variables then making a transformation. This technique is crucial for simulation. For example, the Box-Muller transformation (Box and Muller, 1958) may be used to obtain two independent standard normal random variables using two independent uniform random variables. For $X \sim \mathcal{N}(m, \sigma^2)$, transformation $X = m + \sigma Z$ may be used to generate X, were $Z \sim \mathcal{N}(0, 1)$. For $X \sim \mathcal{N}(m, \Sigma)$ a multivariate Gaussian random variable, transformation $X = m + LZ$ may be used, for Z a vector of standard normal random variables and L the Cholesky decomposition of Σ (a left triangular matrix satisfying $LL^T = \Sigma$).

Transformations aren't just used for sampling relatively straightforward distributions. Recent work on *Normalizing Flows* (Dinh et al., 2017; Papamakarios et al., 2021) has looked at using a sequence of invertible and differentiable transformations to define very expressive probability distributions. One application of Normalizing Flows is to construct a proposal distribution for use in Importance Sampling (Müller et al., 2019; Prangle and Viscardi, 2019) or the Independence Sampler (Gabrié et al., 2022), see Sections 2.2 and 2.3.2.

Sampling directly from a distribution π is not always possible. However, it is often useful to transform π before using a method such as MCMC (introduced in Section 2.3) to sample the transformed distribution. Section 2.5.2 will explain how to make a pre-transformation of a target distribution.

2.1.3 Rejection Sampling

When sampling via inversion or transformation isn't possible, rejection sampling (von Neumann, 1951) can usually be used instead. The algorithm is simple; density π need only be evaluated pointwise up to a multiplicative constant. A proposal distribution q is used to generate a state x which is accepted with probability $\pi(x)/Mq(x)$ else rejected. Constant M is chosen so that $\pi(x)/Mq(x) \leq 1 \forall x \in \mathcal{X}$. Algorithm 1 describes how to obtain n samples from π using rejection sampling. The instruction "Record X" should be interpreted as recording the sample X in some container, such as a vector, so that the i-th sample is recorded in position

i in the container.

Algorithm 1: Rejection Sampling

```
$i \leftarrow 0$.
while $i < n$ do
    $X \sim q, U \sim \mathcal{U}(0,1)$
    if $U < \pi(X)/Mq(X)$ then
        Record $X$
        $i \leftarrow i + 1$
    end
end
```

Rejection sampling tends to be impractical for high-dimensional problems because it is difficult to find a good proposal distribution. If q is not a good approximation of π then many proposed states will be rejected and thus a lot of computation wasted, before a sample is obtained. Adaptive Rejection Sampling (Gilks, 1992; Gilks and Wild, 1992) may be used to find a better proposal distribution.

2.2 Importance Sampling

One of the first uses of Importance Sampling (IS) was in statistical physics for estimating the probability of rare events (Kahn, 1950). Rare event simulation using standard Monte Carlo is inefficient, because estimates are based on a very small number of effective samples, for example see Section 1.1 of Rubino and Tuffin (2009). IS can be used as a method for variance reduction, for example in computing quantiles of the loss distribution (Value-at-Risk) of a portfolio of financial assets (Glasserman et al., 2000). IS may also be used in Bayesian inference.

For q some distribution, equation (2.1) may be rewritten as:

$$\int_{\mathcal{X}} f(x)\pi(x)dx = \int_{\mathcal{X}} f(x)\frac{\pi(x)}{q(x)}q(x)dx = \int_{\mathcal{X}} f(x)w(x)q(x)dx,$$

where $w(x) := \pi(x)/q(x)$ is called the *weight* of state x. Thus for $X_0, \ldots, X_{n-1} \sim q$, assuming π is a normalized density we are able to approximate the expectation of interest as:

$$\mathbb{E}_\pi[f(X)] \approx \frac{1}{n}\sum_{i=0}^{n-1} f(X_i)w(X_i). \tag{2.4}$$

The right hand side of (2.4) has finite variance when

$$\mathbb{E}_q\left[f^2(X)\frac{\pi^2(X)}{q^2(X)}\right] = \mathbb{E}_\pi\left[f^2(X)\frac{\pi(X)}{q(X)}\right] = \int_{\mathcal{X}} f^2(X)\frac{\pi^2(X)}{q(X)}dx < \infty.$$

So that samples may be used with arbitrary f, the importance distribution q should have heavier tails than π. Algorithm 2 describes how to draw n importance samples.

```
Algorithm 2: Importance Sampling
for $i$ in $0$ to $n-1$ do
    $X \sim q$
    $W \leftarrow \pi(X)/q(X)$
    Record $X, W$
end
```

2.2.1 Auto-normalized Importance Sampling

When π is unnormalized the auto-normalized Importance Sampling estimate must be used instead, see for example Chapter 8 of Chopin and Papaspiliopoulos (2020). Suppose

$$\pi(x) = \frac{\tilde{\pi}(x)}{Z}.$$

The relevant identity becomes:

$$\int_{\mathcal{X}} f(x)\pi(x)dx = \frac{\int_{\mathcal{X}} f(x)\frac{\pi(x)}{q(x)}q(x)dx}{\int_{\mathcal{X}} \frac{\pi(x)}{q(x)}q(x)dx} = \frac{\int_{\mathcal{X}} f(x)\frac{\tilde{\pi}(x)}{Zq(x)}q(x)dx}{\int_{\mathcal{X}} \frac{\tilde{\pi}(x)}{Zq(x)}q(x)dx} = \frac{\int_{\mathcal{X}} f(x)\frac{\tilde{\pi}(x)}{q(x)}q(x)dx}{\int_{\mathcal{X}} \frac{\tilde{\pi}(x)}{q(x)}q(x)dx}.$$

Writing $\tilde{w}(x) := \tilde{\pi}(x)/q(x)$, the auto-normalized Importance Sampling estimator of $\mathbb{E}_\pi[f(X)]$ is:

$$\frac{\sum_{i=0}^{n-1} f(X_i)\tilde{w}(X_i)}{\sum_{i=0}^{n-1} \tilde{w}(X_i)}.$$

This estimator converges almost surely to $\mathbb{E}_\pi[f(X)]$ (see Theorem 9.2 for instance of Owen (2013)). The estimator is the ratio of two unbiased estimators, but is biased itself.

2.2.2 Adaptive Importance Sampling

For a given function f, the proposal distribution q that is optimal in the sense of minimizing the Mean Square Error (MSE) of the Importance Sampling estimator is

$$q(x) = \frac{|f(x)|\pi(x)}{\int_{\mathcal{X}} |f(x')|\pi(x')dx'}. \tag{2.5}$$

However, since this optimal proposal distribution depends on π itself, when π is the posterior distribution of a complicated model, it often won't be possible to sample from q. Furthermore, when there are multiple functions $f_1, f_2, \ldots, f_n$ of interest, rather than designing an optimal proposal for computing the expectation of each function, it may be more efficient to have a single proposal to generate a single set of weighted samples, then reuse these weighted samples to compute each expectation.

Instead of basing q on (2.5), it is recommended to choose q to be as close as possible to π, since using π as the proposal distribution would minimize the variance of the importance weights (Doucet et al., 2009, Section 3.2). Recent work has considered approximating π using a Neural Network (Müller et al., 2019). Adaptive Importance Sampling methods (Bugallo et al., 2017; Martino et al., 2017), iteratively modify the proposal distribution using samples obtained from previous iterations, so that the quality of samples improves. For example, successive proposal distributions may be defined as mixture approximations of the target distribution, using kernel density estimation techniques (West, 1993). Population Monte Carlo (Cappé et al., 2004) represents the main framework for adaptive strategies: N proposal distributions $q_0, q_1, \ldots, q_{N-1}$ are used to generate samples and are adapted at each iteration. One of $q_0, q_1, \ldots, q_{N-1}$ could have heavy tails (Hesterberg, 1995), so that the variance of the estimator is finite.

2.3 Markov Chain Monte Carlo

Rejection and Importance Sampling do not scale well to high-dimensional problems because it becomes more and more difficult to find a proposal distribution that is close to π. For these high-dimensional problems it is better to use Markov Chain Monte Carlo (MCMC). This method simulates a Markov chain with limiting distribution π.

The basic building block of MCMC is a *transition kernel* (see Definition 6.2 for example of Robert and Casella (2004)), a function P on $\mathcal{X} \times \mathcal{X}$ such that

1. $P(x, \cdot)$ is a probability measure $\forall x \in \mathcal{X}$,

2. $P(\cdot, B)$ is a measurable function for all sets $B \subset \mathcal{X}$.

For a set $B \subset \mathcal{X}$, let $\pi(B) := \int_B \pi(x)dx$. A distribution π is invariant for P if (see for example Definition 6.35 of Robert and Casella (2004)):

$$\pi(B) = \int_{\mathcal{X}} \pi(x)P(x, B)dx, \forall B \subset \mathcal{X}.$$

When the chain is *φ-irreducible* and *aperiodic* (for definitions, see pages 82 and 114 of Meyn et al. (2009)), the limiting distribution of the chain is the same as the stationary distribution. For π almost all $x \in \mathcal{X}$, as stated in Section 2 of

Roberts and Tweedie (1996b):

$$||P^n(x, \cdot) - \pi||_{TV} \to 0,$$

where the *total variation norm* of a signed measure ν is $||\nu||_{TV} := 2 \sup_B |\nu(B)|$. A *burn-in* period is required, during which the chain approximately converges to π. Assessing the convergence of MCMC samplers is challenging (Gelman et al., 1992). The computational scientist must usually inspect traceplots of the chain to check that the state-space has been fully explored.

Since the samples are no longer independent, the Strong Law of Large Numbers may no longer be used to justify that $\hat{\pi}_n[f]$ converges to $\pi[f]$. Instead, the *Ergodic Theorem* is used (see for example Theorem 6.63 of Robert and Casella (2004)). This states that if $\{X_i\}_{i \geq 0}$ is a Markov chain with invariant distribution π, then the following are equivalent:

1. if $\mathbb{E}_\pi[|f(X)|] < \infty$ then $\lim_{n\to\infty} \hat{\pi}_n[f] = \mathbb{E}_\pi[f(X)]$,
2. $\{X_i\}_{i \geq 0}$ is *Harris recurrent.*

Informally, a Markov chain is Harris recurrent if for an arbitrary set B, the probability of there being an infinite number of returns to B is 1 (see for example Section 6.1 of Robert and Casella (2004), or the formal definition given by Definition 6.32).

The two most widely used MCMC methods are the Gibbs sampler (Geman and Geman, 1984; Tanner and Wong, 1987; Gelfand and Smith, 1990) and the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970); both simulate a Markov chain with stationary distribution π by repeatedly applying π-invariant transition kernels.

Several transition kernels may be used. Suppose $P_1, P_2, \dots, P_n$ are all π-invariant transition kernels. A cycle kernel applies these kernels one after the other:

$$P = P_1 P_2 \cdots P_n.$$

A mixture kernel applies the kernels in a random order:

$$P = \frac{1}{n}(P_1 + P_2 + \cdots + P_n).$$

Importantly, each transition kernel in the cycle or mixture must be π-invariant. This point is worth emphasizing; by contrast the Restore process (introduced in Chapter 4) combines different dynamics, which by themselves are not π-invariant, in such a way that the the dynamics *compensate* for each other so that together they are π-invariant.

The advantage of using several kernels is that it allows the strengths of different kernels to be used together. For example, a kernel that makes large changes to the state of the Markov chain could be used in combination with a kernel that makes small changes. The rest of this section introduces the Gibbs sampler and the Metropolis-Hastings algorithm then regenerative and non-reversible MCMC.

2.3.1 The Gibbs Sampler

The Gibbs sampler generates the next state in the Markov chain by simulating each component of X from its conditional distribution. Let $X = (X_1, X_2, \dots, X_d)^T$ be a multivariate random variable. For $x_{-j} = x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_d$, let $\pi_{X_j|X_{-j}}(x_j|x_{-j})$ be the probability density function of X_j given $X_1 = x_1, \dots,$ $X_{j-1} = x_{j-1}$, $X_{j+1} = x_{j+1}, \dots X_d = x_d$. Suppose it is possible to simulate from each conditional distribution $\pi_{X_j|X_{-j}}$, for $j = 1, \dots, d$. The Gibbs sampler updates each component in turn from its conditional distribution, as shown in Algorithm 3.

Algorithm 3: The Gibbs Sampler

```
$x = (x_1, x_2, \dots, x_d)$
for $i$ in $0$ to $n-1$ do
    for $j$ in 1 to $d$ do
        $x_j \sim \pi_{X_j|X_{-j}}(x_j|x_{-j})$
    end
    Record $x$
end
```

The Gibbs sampler is frequently used in applied Bayesian statistics, in part because of robust implementations in software such as BUGS (Gilks et al., 1994; Lunn et al., 2009) and JAGS (Plummer, 2003). Though it is viable for a large number of modeling problems, not all models admit conditional distributions that may be easily sampled.

In addition, the Gibbs sampler can be slow to converge, especially when components of the target distribution are strongly correlated. Take as example the distribution with the following density, which features in Park and Lee (2022):

$$\pi(x) \propto \exp\left\{-\frac{1}{2}[x_1^2x_3^2 + 2x_1^2 + x_2^2 + 2x_3^2 - 2x_1x_2 + 2x_1x_3 - 2x_2x_3 - 8x_1 - 8x_3]\right\}. \tag{2.6}$$

All conditional distributions are Gaussian:

$$\begin{aligned}
\pi_{X_1|X_2,X_3} &\equiv \mathcal{N}\left(\frac{x_2 - x_3 + 4}{x_3^2 + 2}, \frac{1}{x_3^2 + 2}\right), \\
\pi_{X_2|X_1,X_3} &\equiv \mathcal{N}\left(x_1 + x_3, 1\right), \\
\pi_{X_3|X_1,X_2} &\equiv \mathcal{N}\left(\frac{x_2 - x_1 + 4}{x_1^2 + 2}, \frac{1}{x_1^2 + 2}\right).
\end{aligned}$$

Figure 2.1 shows, for a Markov chain generated using Gibbs sampling, the autocorrelation of each component and density estimates of the 2-dimensional marginal distributions. Consecutive samples are highly correlated in dimensions 1 and 3, whilst the correlation is far smaller in dimension 2. The 2-dimensional marginals help to explain this behaviour. The correlation between X_1 and X_3 is approximately -0.84, which is relatively strong and results in the Gibbs sampler moving less efficiently over π.

2.3.2 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm offers a more general recipe for simulating a π-invariant Markov chain. It does this by designing a *reversible* Markov chain $\{X_i\}_{i\geq 0}$, one for which the distribution of X_{i+1} conditioned on $X_{i+1} = x$ is the same as the distribution of X_{i+1} conditioned on $X_i = x$ (see for example Definition 6.44 of Robert and Casella (2004)). This is achieved by using a transition kernel P that satisfies the *detailed balance* condition (see Definition 6.45 of Robert and Casella (2004)) for π a probability density function:

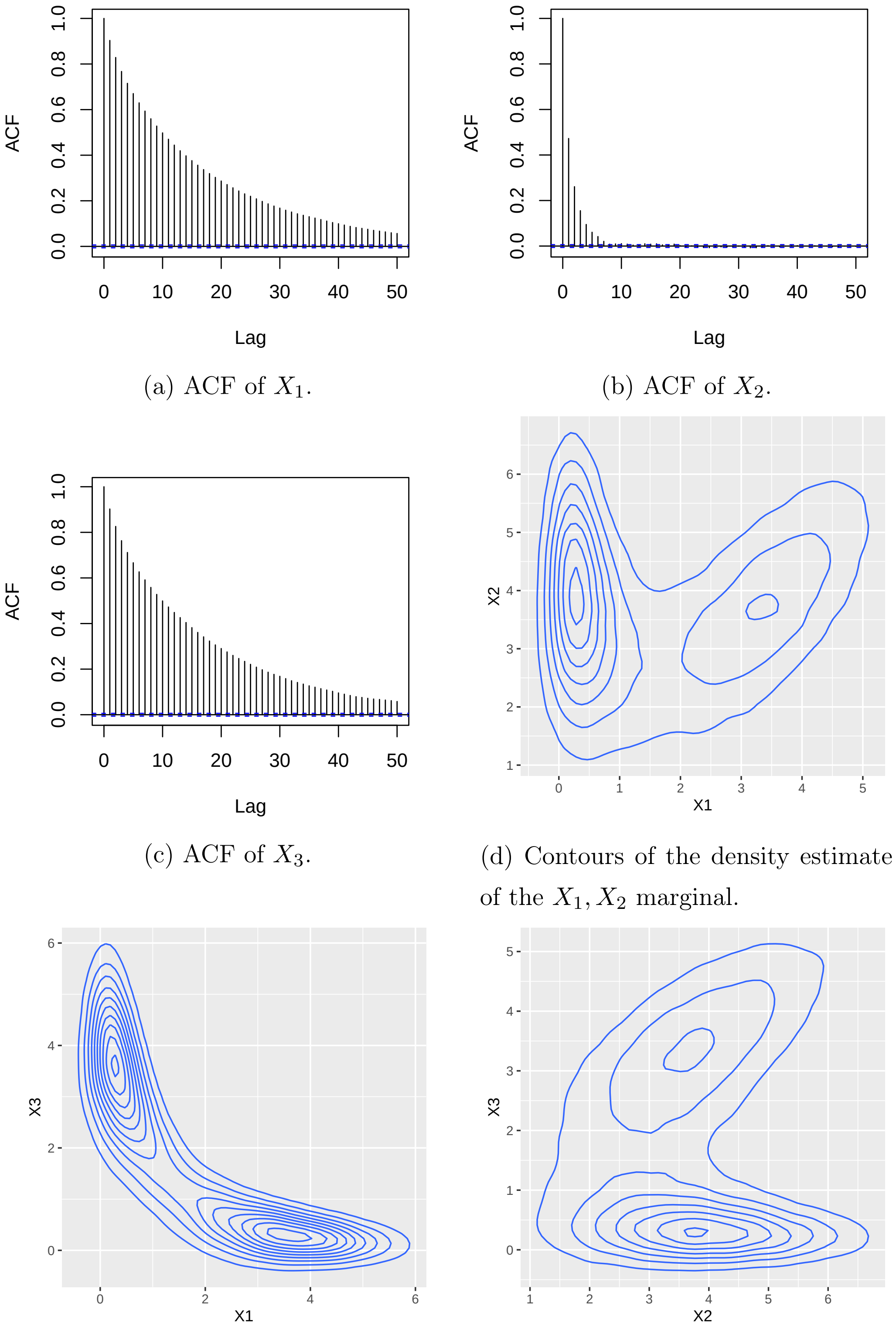
$$\pi(dx)P(x, dy) = \pi(dy)P(y, dx), \forall x, y \in \mathcal{X}. \tag{2.7}$$

From the detailed balance condition, it follows that $\{X_i\}_{i\geq 0}$ is π-invariant and reversible (see for example Theorem 6.46 of Robert and Casella (2004)).

Starting at state x, the Metropolis-Hastings algorithm generates a new state by simulating a proposal state x' from a proposal distribution $q(x, x')$. With probability

$$\alpha(x, x') = \frac{\pi(x')q(x', x)}{\pi(x)q(x, x')}, \tag{2.8}$$

state x' is accepted so that the new state is x'. Otherwise, x' is rejected and there is no change in state – the new state is x. Algorithm 4 provides a summary of how to generate a chain of length n started from state x_0. The instruction “Record X” should be interpreted as adding the sample X to the back of some container that represents the Markov chain. For example, in the C++ programming language, a Standard Template Library vector may be used to store the states of the Markov



(a) ACF of X_1.

(b) ACF of X_2.

(c) ACF of X_3.

(d) Contours of the density estimate of the X_1, X_2 marginal.

(e) Contours of the density estimate of the X_1, X_3 marginal..

(f) Contours of the density estimate of the X_2, X_2 marginal.

Figure 2.1: Autocorrelation functions (ACFs) of the marginals of a Markov chain generated using Gibbs sampling and density estimates of the 2-dimensional marginals for π given by (2.6).

chain and the member function “push_back” may be used to record new states after the current last element of the vector. With states recorded in this way, index i of the container will contain the i-th element of the Markov chain.

Algorithm 4: The Metropolis-Hastings Algorithm

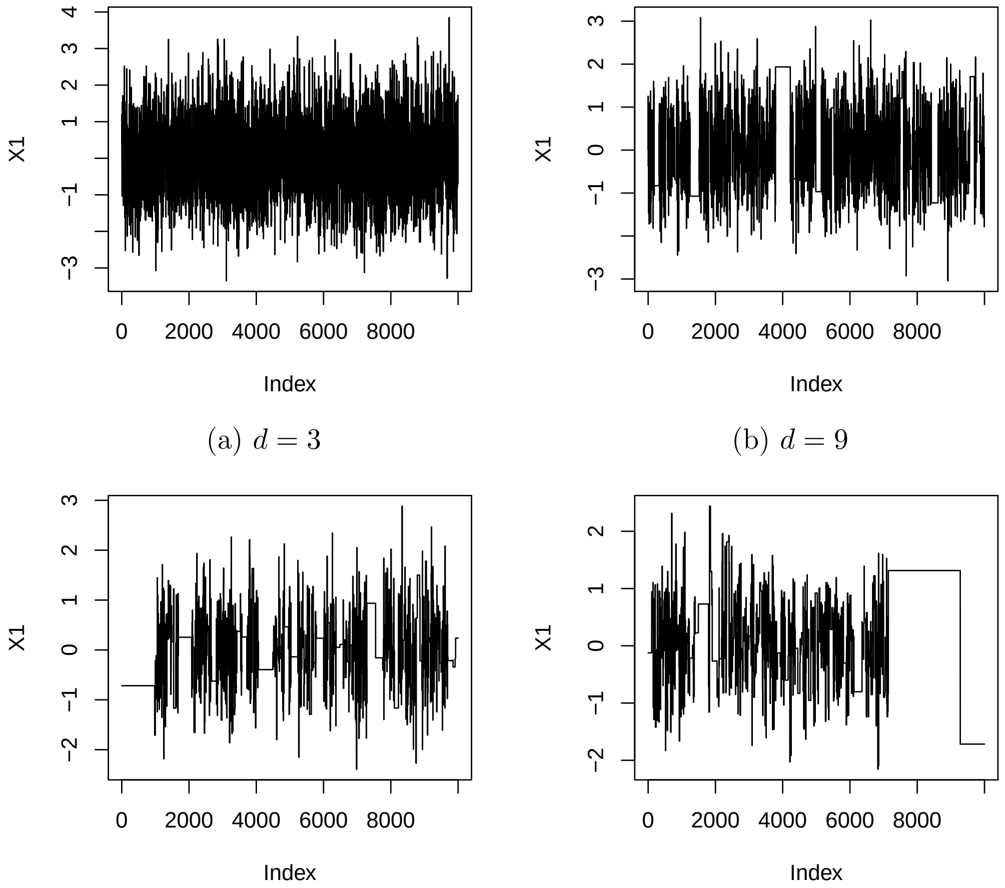
```
$X \leftarrow x_0$
for $i$ in $0$ to $n-1$ do
    $X' \sim q(X, X')$
    $U \sim \mathcal{U}(0,1)$
    if $U < \alpha(X, X')$ then
        $X \leftarrow X'$
    end
    Record $X$
end
```

The proposal distribution is a key part of the algorithm. Two popular families of proposals are now reviewed, which define subcategories of the Metropolis-Hastings algorithm: the Independence Sampler and the Random Walk Metropolis (RWM) algorithm. A third Metropolis-Hastings algorithm, Hamiltonian Monte Carlo, is covered in Chapter 6 since it is most relevant to the methods developed in that chapter. It is worth noting that the Gibbs sampler is in fact a special case of the Metropolis-Hastings algorithm, in which the acceptance probability is 1 for all moves (see Section 3.4 of Andrieu et al. (2003) for example). In addition, there are plenty of other choices of families of proposal distribution, such as that corresponding to the Metropolis-Adjusted Langevin Algorithm (MALA) (Roberts and Tweedie, 1996a; Roberts and Rosenthal, 1998), though the RWM and Independence samplers are most relevant to this thesis.

The Independence Sampler

The Independence Sampler uses a proposal distribution that does not depend on the current state of the chain (Tierney, 1994, Section 2.3.3.). That is, $q(x, x') = q(x'), \forall x \in \mathcal{X}$. In high-dimensional spaces the sampler suffers from the same drawback as Rejection Sampling and Importance Sampling in that it becomes increasingly difficult to find a good proposal distribution. As a result, the Independence Sampler can get “stuck” in a particular state for a long time. Figure 2.2 attempts to illustrate this by showing traceplots for the Independence Sampler in dimensions $d = 3, 9, 27, 81$. The proposal is $q \equiv t_4(0, I_d)$ and the target is $\pi \equiv \mathcal{N}(0.1\mathbb{1}_d, 0.5I_d + 0.5\mathbb{1}_{d\times d})$. As the dimension increases, the number of unique states visited by the Markov chain, which has length 10^4, is $5205, 2391, 1091, 665$.

When q is a good approximation of π, the Independence Sampler is very



(a) $d = 3$

(b) $d = 9$

(c) $d = 27$

(d) $d = 81$

Figure 2.2: Traceplots of Markov chains generated using the Independence Sampler for dimensions $d = 3, 9, 27, 81$. In each dimension, the proposal is $t_4(0, I)$; the target is Gaussian with mean 0.1 and standard deviation 1 for each component and pairwise correlations are 0.5.

effective. Furthermore, it is amenable to regenerative simulation, see Section 2.3.3, and can be used with other kernels to combine moves on local and global scales.

The Random Walk Metropolis Algorithm

The Random-Walk Metropolis (RWM) algorithm uses a Gaussian proposal distribution centred at the current state. Since the proposal is symmetric, equation (2.8) simplifies to $\alpha(x, x') = \pi(x')/\pi(x)$.

The efficiency of the RWM algorithm heavily depends on the proposal distribution and the choice of covariance matrix in particular. To tune the proposal distribution, one must first define a criterion to optimize. One could attempt to directly optimize the *asymptotic variance* of $\hat{\pi}_n[f]$, defined as (see for example equation 3 of Neal (2004)):

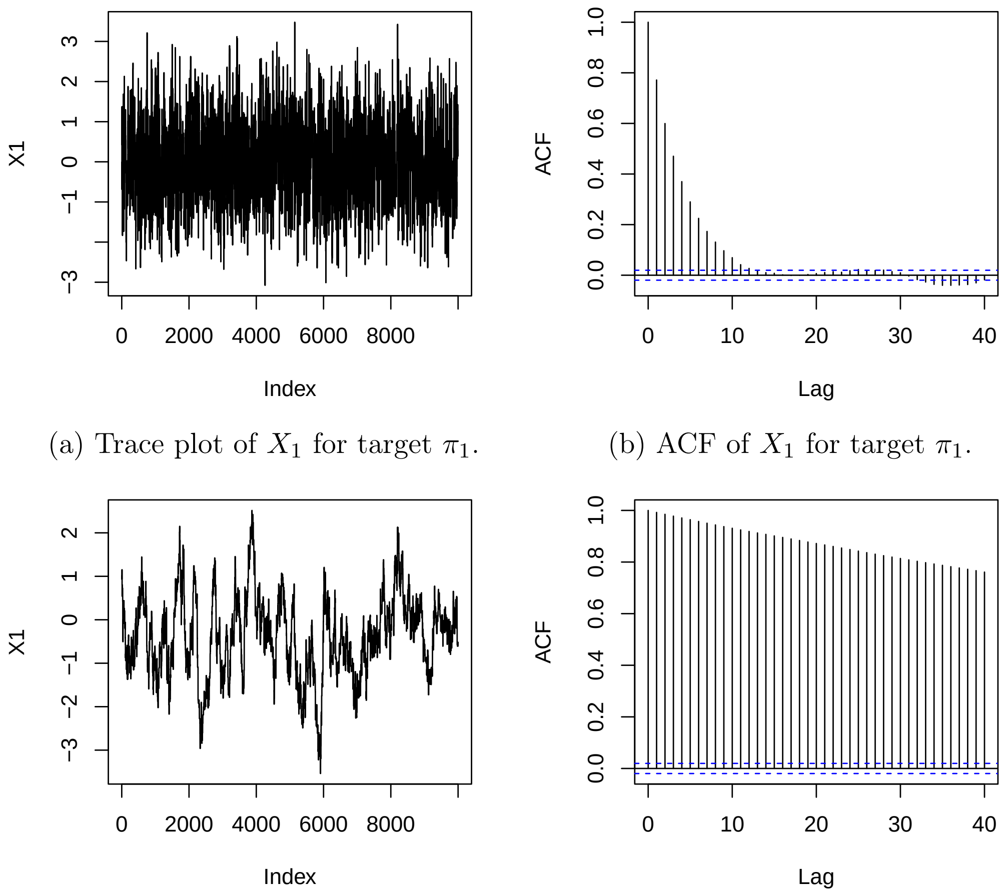
$$\lim_{n \to \infty} n \mathrm{Var}[\hat{\pi}_n[f]].$$

Optimizing the asymptotic variance is typically infeasible because the expression for it is highly non-trivial.

Instead, one could maximise the *expected squared jumped distance* (ESJD), equivalent to minimizing the first-order autocorrelation of the Markov chain, see for example Pasarica and Gelman (2010). The ESJD is the expected squared euclidean distance between consecutive states in the Markov chain. However, both these metrics are specific to the function f (ESJD takes $f(x) = x$), and optimal parameters for f may not generalising to some other function g. Moreover, Markov chains with small asymptotic variance can be slow to converge (Besag and Green, 1993, Section 2).

By contrast, the *expected acceptance probability* is easy to estimate, doesn't depend on f and is supported by theory in particular cases (Roberts et al., 1997). When the target density is a spherical Gaussian, the optimal acceptance rate for the RWM algorithm is approximately 0.44 when $d = 1$ and 0.234 as $d \to \infty$ (Gelman et al., 1996; Roberts et al., 1997). The scale of the proposal distribution can therefore be tuned so that this optimal acceptance rate is reached. When the target distribution is non-Gaussian, these optimal acceptance rates do not necessarily hold. For example, a Gamma-Gamma hierarchical model may have an asymptotically optimal acceptance rate of 0.16 (Bédard, 2006, Section 4.4.3).

When the target distribution is Gaussian and has covariance matrix Σ, the optimal covariance matrix of the proposal distribution for the RWM algorithm is $(2.38^2/d)\Sigma$ (Gelman et al., 1996). This result helps to explain the challenge in sampling high-dimensional distributions. As the dimension of the target increases,



(a) Trace plot of X_1 for target π_1.

(b) ACF of X_1 for target π_1.

(c) Trace plot of X_1 for target π_2.

(d) ACF of X_1 for target π_2.

Figure 2.3: Trace plots and autocorrelation functions for the first marginal of two Markov chains with isotropic Gaussian invariant distributions π_1 and π_2 where π_1 has dimension $d = 2$ and π_2 has dimension $d = 100$.

to maintain an acceptance rate that is close to optimal, the scale of the proposal must decrease at rate $1/d$. This results in the Markov chain making smaller moves, so consecutive states have larger correlation. Figure 2.3 demonstrates this with an empirical example. Two Markov chains are generated with isotropic Gaussian invariant distributions π_1 and π_2 with dimensions $d = 2$ and $d = 100$. Each Markov chain is generated using the RWM algorithm, with the proposal distribution having the optimal covariance matrix. The samples in the Markov chain with invariant distribution π_2 are far more strongly correlated and the chain takes longer to explore the state space.

When MCMC is tuned automatically and during simulation of the Markov chain (as opposed to during a burn-in period, after which tuning parameters are fixed), it is called Adaptive MCMC. There are many strategies (Roberts and Rosenthal, 2009), but adaption must be done carefully, so that the invariant distribution of the Markov chain is still π (Andrieu and Moulines, 2006, Example of Section 1). Adaption may be done by making Robbins-Monro updates (Robbins and Monro, 1951; Atchadé and Rosenthal, 2005; Andrieu and Robert, 2001). These change the scale of the proposal distribution, based on estimates of the expected acceptance probability (Andrieu and Thoms, 2008, Section 4.2). Alternatively, Haario et al. (2001) proposes a method for adapting the entire covariance matrix of the proposal. This method assumes that the support of the target distribution is compact, which is necessary to prove that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} f(X_i) = \pi[f],$$

for $\{X_i\}_{i\geq 0}$ the generated Markov chain. However, the authors remark that in their tests, the method works for (unrestricted) Gaussian distributions and that they expect that their method works for non-compactly supported target distributions with densities satisfying a certain tail condition. When a Metropolis-Hastings algorithm is used in this thesis, its parameters are tuned during a burn-in period, then fixed whilst output is recorded.

Chapter 5 looks at adapting the Restore algorithm, introduced in Chapter 4. Restore algorithms do not have an accept/reject step in the same way that Metropolis-Hastings algorithms do. Instead of tuning scalar parameters, Adaptive Restore adjusts the regeneration distribution used by the algorithm.

Slow convergence

We end this section by commenting on a major issue for the Metropolis-Hastings algorithm: slow convergence. Through examples, it has been demonstrated that the Gibbs Sampler converges slowly when π has strong correlations, the Independence Sampler makes fewer and fewer moves as the proposal distribution and target become less close to each other and the RWM algorithm makes smaller moves as the dimension of π increases. A motivation for combining moves on a local and global scale is to benefit from both frequent and large moves. However, a cycle or mixture kernel inherits the properties of its constituent kernels: as the dimension increases, moves on a local scale become smaller and those on a global scale are made less frequently. A motivation for Restore (introduced in Chapter 4) is to overcome this: dynamics are combined in such a way that local moves do not become smaller and global moves to do not become less frequent.

As a further remark, the Metropolis-Hastings algorithm can be even slower to convergence when π is multimodal. The RWM algorithm struggles to cross regions of low probability density between modes and it is difficult to design a good independent proposal. Technqiues to improve mixing including tempering (Geyer, 1991; Marinari and Parisi, 1992; Neal, 1996b; Kou et al., 2006) and mode-hopping moves (Tjelmeland and Hegstad, 2001; Andricioaei et al., 2001; Sminchisescu and Welling, 2011; Ahn et al., 2013). Restore is well suited to sampling multi-modal distributions, since its regenerative structure provides opportunities for jumping between modes.

2.3.3 Regeneration in Markov Chain Monte Carlo

A Markov chain $\{X_i\}_{i\geq 0}$ is regenerative if there exists a sequence of random times $T_0 < T_1 < \cdots$ such that for every T_i, the future of the process is independent of the past and identically distributed. See Chapter 6 of Asmussen (2003) for an introduction to regenerative processes. In effect, the process restarts itself at each T_i, called regeneration times. Suppose that the chain begins with a regeneration, so that $T_0 = 0$.

Tours are the sets of states between regeneration times. For $i = 1, 2, \dots$ the i-th tour is $\{X_{T_{i-1}}, X_{T_{i-1}+1}, \dots, X_{T_i - 1}\}$. The length of the i^{th} tour is $\tau_i := T_i - T_{i-1}$ for $i = 1, 2, \dots$. For a test function f let

$$H_i := \sum_{j=T_{i-1}}^{T_i - 1} f(X_j); \quad i = 1, 2, \dots.$$

Pairs (τ_i, H_i) are independent and identically distributed, which makes the tours suitable for estimating $\pi[f]$. If $\mathbb{E}[| \ H_i \ |] < \infty$ and $\mathbb{E}[\tau_i] < \infty$ then by the Strong Law of Large Numbers (see Section 2 of Mykland et al. (1995) for example):

$$\hat{\pi}_{T_n}[f] = \frac{\sum_{i=1}^n H_i}{\sum_{i=1}^n \tau_i} = \frac{1}{T_n} \sum_{i=0}^{T_n - 1} f(X_i) \to \pi[f].$$

One can attain an estimate for the variance of the estimator itself (Brockwell and Kadane, 2005). Let the average tour length be:

$$\bar{\tau} := \frac{1}{n} \sum_{i=1}^n \tau_i.$$

Then in distribution (see for example equation 10 of Hobert et al. (2002)):

$$\sqrt{n}(\hat{\pi}_{T_n}[f] - \pi[f]) \to \mathcal{N}(0, \sigma^2). \tag{2.9}$$

A consistent estimator for σ^2 is (see for example equation 11 of Hobert et al. (2002)):

$$\hat{\sigma}_n^2 = \frac{\frac{1}{n} \sum_{i=1}^n (H_i - \hat{\pi}_{T_n}[f]\tau_i)^2}{\bar{\tau}^2}. \tag{2.10}$$

Two further benefits of regeneration in MCMC is that tours may be simulated in parallel and there is no need to discard a burn-in period (assuming the chain starts with a regeneration and is simulated for a fixed number of tours). The latter property follows from the fact that tours are independent and identically distributed.

Nummelin's splitting technique (Nummelin, 1978) is a method for determining when regenerations have happened in a Markov chain. The method has been used for MCMC (Mykland et al., 1995; Gilks et al., 1998; Brockwell and Kadane, 2005). The Independence Sampler (defined in Section 2.3.2) is easy to split and the proposal distribution q may be adapted at regeneration times without disturbing the limiting distribution of the chain. However, regenerations tend to recede exponentially as the dimension increases.

2.3.4 Non-reversible Markov Chain Monte Carlo

For the Markov chain to be π-invariant, it is sufficient but not necessary for the transition kernel to satisfy 2.7, the detailed balance condition. The condition is very useful because if one can construct a Markov transition kernel P that satisfies it, then the Markov chain generated with P will be π-invariant. However, requiring that P satisfies detailed balance is restrictive: there are π-invariant kernels that do not satisfy detailed balance. Furthermore, there is both empirical (Suwa and Todo, 2010; Turitsyn et al., 2011) and theoretical (Neal, 2004; Chen and Hwang, 2013) evidence that non-reversible Markov chains are better in terms of asymptotic variance and speed of convergence.

2.4 Estimating Normalizing Constants

Estimating normalizing constants, or more precisely, ratios of normalizing constants, is useful in Bayesian inference for model comparison (Gelman and Meng, 1998). Normalizing constants tend to be very difficult to estimate. Suppose $\pi(x) = \tilde{\pi}(x)/Z$, where Z is unknown. Estimating Z is difficult because it is a scalar quantity but depends on a measure defined on a d dimensional space: typically $Z = \int_{\mathbb{R}^d} \tilde{\pi}(x)dx$. Section 4.5 notes that the Restore process may be used to estimate normalizing constants. In subsections 2.4.1 and 2.4.2 we briefly present two methods for estimating normalizing constants, *Sequential Monte Carlo* and *Thermodynamic Integration*, without going into much detail. We have already seen one method for estimating normalizing constants: Importance sampling (Section 2.2). There are other techniques, such as *Bridge Sampling* (Meng and Wong,

1996) or *Nested Sampling* (Skilling, 2006), which are not covered.

2.4.1 Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods were first developed for Bayesian filtering problems and dynamics models, see the review of Cappé et al. (2007) and Chapter 1 of Doucet et al. (2001). Here, data $y_1, y_2, \dots, y_n$ are observed sequentially and the aim is to infer the posterior distribution $\pi(x_1, \dots, x_n | y_1, \dots, y_n)$, where $x_1, \dots, x_n$ are hidden states. In the context of such state-space models, SMC methods are also referred to as *Particle Filters.* Gordon et al. (1993) made a key contribution through the proposal of the Bootstrap Filter algorithm, in which particles are propagated forward according to the transition density of the model for the dynamics of the hidden state, assigned a weight proportional to the likelihood of the corresponding observation, then resampled from the discrete distribution proportional to the weights.

SMC may also be applied to non-dynamic models (Chopin, 2002; Del Moral et al., 2006). It is possible to use SMC to compute the marginal likelihood for use in Bayesian model selection (Zhou et al., 2016). A potential advantage of SMC is that it is well suited to parallel computing (Lee et al., 2010).

2.4.2 Thermodynamic Integration

Suppose $\pi_1(x) = \tilde{\pi}_1(x)/Z_1$ is a distribution known up to some normalization constant. Thermodynamic Integration (Ogata, 1989; Neal, 1993, Section 6.2) may be used to estimate $Z_1 = \int_{\mathcal{X}} \tilde{\pi}_1(x)dx$. Let $\pi_0(x) = \tilde{\pi}_0(x)/Z_0$ be a distribution with known normalizing constant $Z_0 = \int_{\mathcal{X}} \tilde{\pi}_0(x)dx$. If $\tilde{\pi}_0$ and $\tilde{\pi}_1$ have the same support, it is possible to find a path linking them, parameterized by some scalar $\theta \in [0, 1]$. For instance, the geometric path is $\tilde{\pi}_\theta(x) = \tilde{\pi}_0^{1-\theta}(x)\tilde{\pi}_1^{\theta}(x)$. The normalized intermediate distribution is $\pi_\theta(x) = \tilde{\pi}_\theta(x)/Z_\theta$. We have

$$\frac{d}{d\theta} \log Z_\theta = \frac{\frac{d}{d\theta} Z_\theta}{Z_\theta} = \frac{1}{Z_\theta} \frac{d}{d\theta} \int \tilde{\pi}_\theta(x)dx.$$

Assuming it's valid to interchange integration and differentiation and using

$$\frac{d}{d\theta} \log \tilde{\pi}_\theta(x) = \frac{\frac{d}{d\theta}\tilde{\pi}_\theta(x)}{\tilde{\pi}_\theta(x)} \quad \Leftrightarrow \quad \tilde{\pi}_\theta(x) \frac{d}{d\theta} \log \tilde{\pi}_\theta(x) = \frac{d}{d\theta} \tilde{\pi}_\theta(x),$$

we get

$$\frac{d}{d\theta} \log Z_\theta = \int \frac{1}{Z_\theta} \frac{d}{d\theta} \tilde{\pi}_\theta(x)dx = \int \frac{\tilde{\pi}_\theta(x)}{Z_\theta} \frac{d}{d\theta} \log \tilde{\pi}_\theta(x)dx = \mathbb{E}_{\pi_\theta(x)} \left[\frac{d}{d\theta} \log \tilde{\pi}_\theta(x) \right].$$

Writing $U_\theta(x) = \frac{d}{d\theta} \log \tilde{\pi}_\theta(x)$ and integrating from 0 to 1 gives

$$\log \frac{Z_1}{Z_0} = \int_0^1 \mathbb{E}_{\pi_\theta(x)} \left[U_\theta(x)\right] d\theta.$$

Therefore, we are able to estimate Z_1 by using MCMC to estimate $\mathbb{E}_{\pi_\theta(x)}[U_\theta(x)]$ for $\theta = 0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}$; we can then use quadrature to approximate the right-hand side integral. Though it's possible to use n independent Markov chains to estimate each expectation, it's possible to avoid having to discard n burn-in periods by starting chain $i+1$ from the last state of chain i. In fact for small enough increments in θ, it's even possible to estimate each expectation with just one sample, as in the *slow growth* method (Bash et al., 1987).

2.5 Related Methods

The following are not Monte Carlo methods, but are highly relevant for Bayesian computation. The Laplace Approximation of subsection 2.5.1 may be used to approximate a distribution π with a Gaussian. The Laplace Approximation may be used to transform a posterior distribution, as in subsection 2.5.2, so that the transformed distribution is roughly centred at zero and has covariance matrix close to the identity. Methods presented in this thesis rely on the gradient and Laplacian of the log-density of the posterior – subsection 2.5.3 comments on how to compute derivatives. To make a Laplace Approximation one must find the mode of the posterior; subsection 2.5.4 covers the optimization method used to do this.

2.5.1 The Laplace Approximation

The Laplace Approximation provides an analytic approximation of an integral (Tierney and Kadane, 1986). The Laplace approximation has been used in Bayesian experimental design to reduce intractable double-loop integrals to an approximation that can be computed using single-loop numerical integration (Long et al., 2013). Moreover, in the same setting, the Laplace approximation has been used as a suitable proposal distribution for Importance Sampling (Beck et al., 2018). Results exist on the convergence of the Laplace approximation to the posterior distribution as the size of the data increases (Schillings et al., 2020). In future chapters we will use the Laplace Approximation to approximate a distribution with a Gaussian distribution.

Define $U(x) := -\log \pi(x)$ as the *energy* of π. Suppose that π is a univariate distribution. Making a Taylor series expansion of $U(x)$ at its minimum x_0 and using the fact that $U'(x_0) = 0$ gives:

$$\pi(x) \approx \exp\left\{-U(x_0) - \frac{1}{2}U''(x_0)(x - x_0)^2\right\}.$$

Ignoring the constant term $\exp\{-U(x_0)\}$, it can be seen that this approximation of π is Normally distributed with mean x_0 and variance $1/U''(x_0)$. Thus the closer π is to a Normal distribution, the better the Laplace approximation is. In particular, this can make the Laplace approximation very accurate when approximating a posterior distribution dependent on a large quantity of data. By the *Bernstein-von Mises theorem*, for models satisfying certain regularity conditions, the posterior converges to a Gaussian distribution (Vaart, 1998, Section 10.2). Note that one of the regularity conditions is that it is possible to separate the true value of X from the complements of balls centered at the true value of X. Thus the theorem does not apply to all posterior distributions; for example it doesn't apply to multi-modal distributions.

When π is a multidimensional distribution, let x_0 be the mode and $H_U(x_0)$ the Hessian of U at x_0. Then the Laplace approximation is $\mathcal{N}(x_0, H_U(x_0)^{-1})$.

2.5.2 Transformations of Distributions

Applying a transformation to a posterior distribution before drawing samples can improve the efficiency of many methods for Bayesian inference (Hills and Smith, 1992; Rue et al., 2009). In particular, it helps to transform the target so that its covariance structure is closer to the identity, since the scale of each dimension is then roughly the same and hence a sampler can explore any dimension as easily as any other.

Suppose X is a d-dimensional random variable. Typically a linear transformation is used before sampling, so that inference is done on $Y := MX$, for M a $(d \times d)$ matrix. By the change-of-variables formula and the fact that the determinant of the inverse of an invertible matrix is the inverse of the determinant, it follows that:

$$\begin{aligned}\pi_Y(y) = \pi_X(x)\left|\frac{\partial x}{\partial y}\right| = \pi_X(M^{-1}y)\left|\frac{\partial(M^{-1}y)}{\partial y}\right| &= \pi_X(M^{-1}y)|M^{-1}| \\ &= \pi_X(M^{-1}y)|M|^{-1}.\end{aligned}$$

For many methods in Bayesian computation, the probability density function of the target distribution only needs to be known up to a normalizing constant, in which case it isn't necessary to take into account $|M|^{-1}$. Samples $Y_1, Y_2, \ldots$

from π_Y may be transformed back to samples from π_X using $X_i = M^{-1}Y_i$ for $i = 1, 2, \ldots$.

To compute the gradient of the logarithm of the density of a transformed distribution, necessary for methods in this thesis, one simply uses the chain rule:

$$\begin{aligned}
\nabla_y \log \pi_Y(y) &= \nabla_y \log \pi_X(x), \\
&= \left(\frac{\partial x}{\partial y}\right)^T \nabla_x \log \pi_X(x), \\
&= (M^{-1})^T \nabla_x \log \pi_X(x).
\end{aligned}$$

To see this, it may help to differentiate with respect to a single component y_j.

Methods in this thesis also require computing the Laplacian of the target density. Using the product and chain rule for differentiation one may compute that

$$\Delta \log \pi_Y(y) = \sum_{j=1}^{d} \left(M^{-1}\right)_{\cdot,j}^T H_{\log \pi_X}(x) \left(M^{-1}\right)_{\cdot,j} \qquad (2.11)$$

for $(M^{-1})_{\cdot,j}$ column j of matrix M^{-1} and $H_{\log \pi_X}(x)$ the Hessian matrix of $\log \pi_X(x)$ at x. See Appendix A for a derivation of this formula. Equivalently, for Tr the trace of a matrix,

$$\Delta \log \pi_Y(y) = \mathrm{Tr}\left\{\left(M^{-1}\right)^T H_{\log \pi_X}(x) \left(M^{-1}\right)\right\}.$$

A typical choice for M^{-1} is the matrix-equivalent of the square root of Σ, for Σ some estimate of the covariance structure of X. That is, suppose the eigendecomposition of Σ is $\Sigma = V\Lambda V^T$, for V the matrix of (normalized) eigenvectors and Λ a diagonal matrix of corresponding eigenvalues. Let $M^{-1} = V\Lambda^{1/2}$ for $\Lambda^{1/2}$ a diagonal matrix containing the square roots of the eigenvalues. Then $(M^{-1})^T M^{-1} = \Sigma$. Hence, if X was a zero-mean Gaussian with non-identity covariance matrix Σ, then $Y = MX$ would be a Gaussian random variable with identity covariance matrix. The matrix Σ is usually based on the Laplace approximation of π_X.

2.5.3 Computing Derivatives

Methods in this thesis use first and second derivatives of log-density functions. Analytically calculating these quantities is difficult and time-consuming. *Automatic Differentiation* (AD) is a method, now widely used particularly by the Machine Learning community, for computing derivatives (Baydin et al., 2018). AD relies on the fact that any function may be represented as the composition of a number of simpler functions. Computing derivatives amounts to the repeated use of the chain rule of calculus, applied to these simpler functions. AD has been used successfully in a number of software packages, including Stan (Carpenter et al., 2015).

2.5.4 Optimization

To make a Laplace Approximation, one must find the mode of the target distribution. Typically this is done using an optimization method. The “optim” function in R uses the *Nelder-Mead* method (Nelder and Mead, 1965) by default. This method uses neither an analytic expression nor a numerical approximation of the gradient of the function being optimized. The Nelder-Mead algorithm can convergence to a point that is not in fact a stationary point. Whatever method is used for optimization, it is worth checking the gradient at the point found is indeed approximately zero.

In some of our experiments, we found the *BFGS* method (Broydon, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970), available via R’s “optim” function, succeeded where the Nelder-Mead method failed. The BFGS method is a *Quasi-Newton* method: it uses the exact gradient and an approximation of the Hessian to find the optimum.

Chapter 3

Simulation of Continuous-time Markov processes

The novel Monte Carlo methods presented in this thesis rely on continuous-time Markov processes (Asmussen, 2003; Mörters and Peres, 2010; Norris, 1997; Øksendal, 2003). The Restore process, which is fundamental to this thesis and introduced fully in Chapter 4, is a continuous-time Markov process. Chapters 5 and 6 detail continuous-time Monte Carlo methods based on novel classes of Restore processes. This chapter will cover some results on continuous-time Markov processes, which will be useful in constructing and analysing the specific processes to follow.

There is increasing interest in the use of continuous-time Markov processes for Monte Carlo. Piecewise Deterministic Markov Processes (PDMPs) (Davis, 1984) have received a considerable amount of attention. These include the Bouncy Particle Sampler (Bouchard-Côté et al., 2018) and the Zig-Zag process (Bierkens et al., 2019a,b) as well as contributions such as the use of non-constant speed functions (Vasdekis and Roberts, 2021), numerical approximation of switching times (Cotter et al., 2020), use in tandem with MCMC updates of blocks of components (Sachs et al., 2021), the Boomerang Sampler (Bierkens et al., 2020), learning the covariance matrix of the target distribution (Bertazzi and Bierkens, 2021) and alterations designed for high dimensional sparse models (Bierkens et al., 2021).

An entirely different Monte Carlo method, the ScaLE algorithm (Pollock et al., 2020) relies on the concept of a *quasi-stationary distribution* (Collet et al., 2013). Under certain conditions, the *quasi-limiting distribution* of a killed diffusion can be made to be some target distribution (Wang et al., 2019). The ScaLE algorithm is a Monte Carlo method based on simulating a Markov process in such a way that its quasi-stationary distribution is some target distribution.

Divide and conquer methods (Scott et al., 2016; Neiswanger et al., 2014; Wang and Dunson, 2014; Dai et al., 2019) and *subsampling* are major strategies for inference on Big Data, since these methods can reduce computational cost. When using subsampling, instead of needing to compute a function of the entire data, like in the Metropolis-Hastings algorithm (see Section 2.3.2), it is only necessary to compute a function of a subset of the data. PDMPs and the Markov process on which the ScaLE algorithm is based both allow subsampling. It would be useful to be able to use subsampling within simulation of a Restore process, but it is unclear how this could be done.

Section 3.1 introduces a few concepts related to continuous-time Markov processes including their filtration, the Markov property, the Chapman-Kolmogorov equations, generators and adjoint generators. Next, Section 3.2 covers Jump Processes, Poisson processes and the Exponential distribution. Jump Processes are a class of Markov process that may be enriched or adjusted with regenerations, see Section 4.2 and Chapter 6. Poisson processes (see for example Kingman (1992)) and their simulation (see for example Devroye (1986)) are fundamental to the methodology of this thesis, so are covered in Section 3.2.3. Brownian Motion, covered in Section 3.3, serves as the underlying process of a Restore process in Section 4.3 and an Adaptive Restore process in Chapter 5.

3.1 Markov Processes in Continuous-time

A *stochastic process* is a collection or random variables $\{X_t\}_{t\geq 0}$, indexed by time $t \in [0, \infty)$, see Definition 1.8 of Liggett (2010) for example.

A transition function (see Definition 1.2 of Revuz and Yor (2013) for example) is a family of transition kernels $\{P^t\}_{t\geq 0}$ such that for any $B \in \mathcal{X}$:

$$P^{t+s}(x, B) = \int P^s(x, dy)P^t(y, B).$$

The above equation is called the Chapman-Kolmogorov equation. The family $\{P^t\}_{t\geq 0}$ forms a *probability semi-group*, defined next.

Let $\mathcal{X}$ be locally compact and write $C(\mathcal{X})$ for the space of continuous real-valued functions vanishing at infinity. A probability semi-group, see Definition 3.4 of Liggett (2010), is a family of continuous linear operators $\{P^t, t \geq 0\}$ on $C(\mathcal{X})$ satisfying

1. $P^0 f = f, \forall f \in C(\mathcal{X})$.

2. $\lim_{t\downarrow 0} P^t f = f, \forall f \in C(\mathcal{X})$.

3. $P^{t+s} f = P^t P^s f, \forall f \in C(\mathcal{X})$.

4. $P^t f \geq 0$ for every nonnegative $f \in C(\mathcal{X})$.

5. There exists a sequence of function $f_n \in C(\mathcal{X})$ so that $\sup_n ||f_n|| < \infty$ and $P^t f_n$ converges to 1 pointwise for each $t \geq 0$.

A Markov process $\{X_t\}_{t\geq 0}$ (see for instance Definition 1.1 of Rogers and Williams (2000)) is a stochastic process, adapted to a filtration $\{\mathcal{F}_t\}_{t\geq 0}$, such that for any bounded measurable function f, $x \in \mathcal{X}$ and $s \geq 0, t \geq 0$, $\mathbb{P}^x$ almost surely:

$$\mathbb{E}_x[f(X_{s+t})|\mathcal{F}_s] = (P^t f)(X_s).$$

The generator $\mathcal{L}$ of a continuous-time Markov process is an operator, which operates on a subspace $D(\mathcal{L})$ of functions in $\mathcal{X}$, called the domain of the generator. The intuition is that when acting on some function f, the generator gives the rate of expected change in value of the function of the process. We use Definition 4.5 of Eberle (2023) for the generator and the domain:

$$D(\mathcal{L}) = \left\{ f \in \mathcal{X} : \lim_{\epsilon\downarrow 0} \frac{P^\epsilon f - f}{\epsilon} \text{ exists} \right\};$$
$$\mathcal{L}f = \lim_{\epsilon\downarrow 0} \frac{P^\epsilon f - f}{\epsilon} \text{ for } f \in D(\mathcal{L}).$$

A probability distribution π is invariant for $\{P^t\}_{t\geq 0}$ if and only if there exists a dense subset $\mathcal{A} \subseteq D(\mathcal{L})$ such that

$$\int_{\mathcal{X}} \mathcal{L}f(x)\pi(x)dx = 0, \quad \forall f \in \mathcal{A} \tag{3.1}$$

and in this case, the above equation holds for all functions $f \in D(\mathcal{L})$ (see Theorem 4.18 of Eberle (2023)).

For finding the stationary distribution, (3.1) is not particularly useful because all $f \in D(\mathcal{L})$ must be considered. The adjoint of the generator may be used to check that a distribution π is stationary for a process $\{X_t\}_{t\geq 0}$. The adjoint of an operator $\mathcal{L}$ is an operator $\mathcal{L}^\dagger$ such that, for all $f \in D(\mathcal{L})$:

$$\int_{\mathcal{X}} \mathcal{L}f(x)g(x)dx = \int_{\mathcal{X}} f(x)\mathcal{L}^\dagger g(x)dx.$$

Equation (3.1) may be written as

$$\int_{\mathcal{X}} f(x)\mathcal{L}^\dagger \pi(x)dx = 0.$$

Since this holds for all $f \in D(\mathcal{L})$, it follows that $\mathcal{L}^{\dagger}\pi(x) = 0, \forall x \in \mathcal{X}$. Thus the stationary distribution may be found by first finding the adjoint of the generator.

3.2 Jump Processes, Poisson processes and the Exponential distribution

This section presents two related classes of continuous-time Markov process: Jump Proccesses and Poisson processes. Both rely on Exponential random variables, hence the three are covered together in this section.

3.2.1 Properties of the Exponential Distribution

The *holding times* of a *jump process* (see subsection 3.2.2) and the *arrival times* of events of a *Poisson process* (see subsection 3.2.3) are exponentially distributed. Some basic properties of the exponential distribution are recapped here. First, recall that $\tau^{(\lambda)} \sim \text{Exp}(\lambda)$ has distribution function $\mathbb{P}[\tau^{(\lambda)} < t] = 1 - e^{-\lambda t}$. The Exponential distribution is fundamental to continuous-time Markov processes because of the Memoryless property below.

Theorem 1 (The Memoryless property)**.** *A continuous random variable τ has an exponential distribution if and only if* $\mathbb{P}(\tau > s+t|\tau > s) = \mathbb{P}(\tau > t), \forall s, t \geq 0$.

The next property is useful for simulating a Jump process or Poisson process that has two or more components. The property is presented for two independent exponential random variables, but may be extended by induction to countably many exponential random variables.

Theorem 2 (Minimum of Exponential Random Variables)**.** *Let $\tau^{(\lambda_1)} \sim Exp(\lambda_1)$ and $\tau^{(\lambda_2)} \sim Exp(\lambda_2)$ be independent random variables and $\lambda := \lambda_1 + \lambda_2 < \infty$. Then $\tau^{(\lambda)} := \min\{\tau^{(\lambda_1)}, \tau^{(\lambda_2)}\}$ is attained at a unique random value $K \in \{1, 2\}$. Furthermore, $\tau^{(\lambda)}$ and K are independent with $\tau^{(\lambda)} \sim Exp(\lambda)$ and $\mathbb{P}[K = k] = \lambda_k/\lambda$ for $k = 1, 2$ (Norris, 1997, Theorem 2.3.3.).*

3.2.2 Jump Processes

A jump process makes discrete movements (jumps) at random times. The time spent in each state is positive and therefore sample paths are piecewise constant (Asmussen, 2003, II. Markov Jump Processes; 1. Basic Structure.). A jump process may be defined by a Markov transition kernel P, which determines how

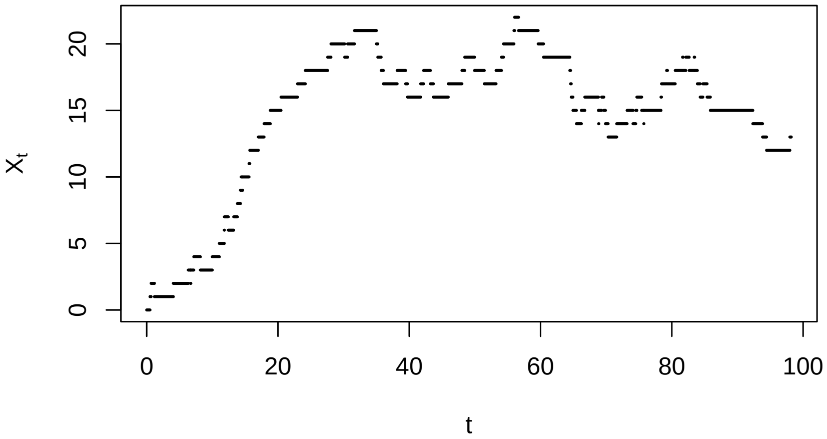


Figure 3.1: Sample path of a jump process $\{X_t\}_{t\geq 0}$ with $\lambda = 1$ and random walk Markov transition kernel.

jumps are made, and a holding rate $\lambda : \mathcal{X} \to \mathbb{R}^+$, which affects the random times for which the jump process occupies states.

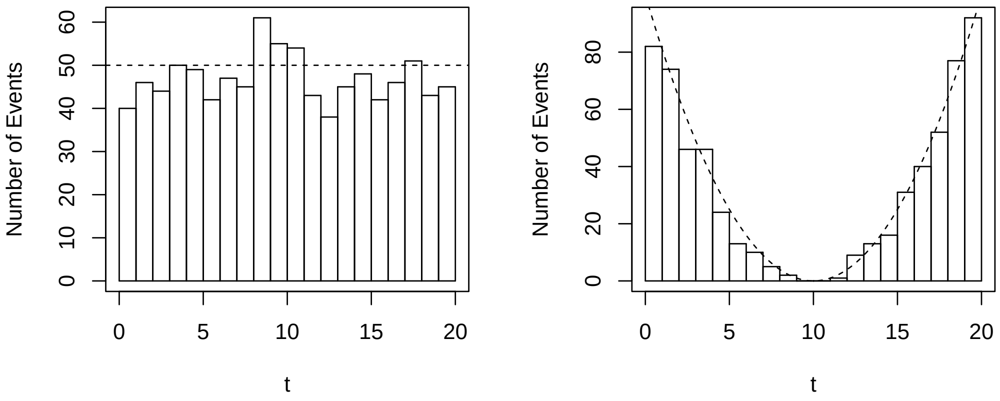
More specifically, suppose the process jumps at times $T_0^{(\lambda)} = 0 < T_1^{(\lambda)} < T_2^{(\lambda)} < \cdots$ and visits states $\tilde{X}_0, \tilde{X}_1, \ldots, \tilde{X}_{n-1}$. The *holding times* are $\tau_i^{(\lambda)} = T_{i+1}^{(\lambda)} - T_i^{(\lambda)}$. The joint distribution of $\{\tilde{X}_i, \tau_i^{(\lambda)}\}_{i=0}^{n-1}$ factorises in the following way. States $\{\tilde{X}_i\}_{i=0}^{n-1}$ form a Markov chain (called the *jump chain*) with Markov transition kernel P. There exists $\lambda : \mathcal{X} \to \mathbb{R}^+$ such that given $\{\tilde{X}_i\}_{i=0}^{n-1}$, random variables $\{\tau_i^{(\lambda)}\}_{i=0}^{n-1}$ are independent with $\tau_i^{(\lambda)} \sim \text{Exp}(\lambda(\tilde{X}_i))$.

The factorisation of the jump chain and holding times makes simulation of a jump process straightforward. Algorithm 5 shows how to simulate a Jump process starting with state x_0, with Markov transition kernel P and holding rate $\lambda : \mathcal{X} \to \mathbb{R}^+$, so that the jump chain has length n. The algorithm records the jump chain and holding times.

Algorithm 5: Jump Process Simulation

```
$X \leftarrow x_0, i \leftarrow 0$
while $i < n$ do
    $\tau^{(\lambda)} \sim \text{Exp}(\lambda(X))$
    Record $X, \tau^{(\lambda)}$
    $i \leftarrow i + 1$
    $X \leftarrow P(X, \cdot)$
end
```

Figure 3.1 shows a sample path of a very simple jump process. Here, $X_0 = 0$ and $\lambda = 1$. Markov transition kernel P is such that $\mathbb{P}[\tilde{X}_{i+1} = \tilde{X}_i + 1] = 0.5$ and $\mathbb{P}[\tilde{X}_{i+1} = \tilde{X}_i - 1] = 0.5$.



(a) $\lambda(t) = 50$

(b) $\lambda(t) = (t - 10)^2$

Figure 3.2: Illustrations of homogeneous and inhomogeneous Poisson processes. Bars show the simulated number of events occurring in each interval of unit length. Dotted lines shows the rate.

3.2.3 Poisson Processes

This thesis uses Poisson processes representing events in time, defined on $\mathbb{R}_+$. Poisson processes have an elegant definition on $\mathbb{R}^d$ (Kingman, 1992), though this is not particularly relevant to the thesis.

A Poisson process on $\mathbb{R}^+$ with rate $\lambda(t)$ satisfying $\lambda(t) \geq 0, \forall t \geq 0$, is defined by the following property (Devroye, 1986, Chapter 6, Section 1.3.). Let $N_1, N_2, \ldots, N_n$ be the number of events occurring on the disjoint intervals (l_1, r_1), (l_2, r_2), $\ldots, (l_n, r_n)$. Then $N_1, N_2, \ldots, N_n$ are independent and for $i = 1, \ldots, n$, random variable N_i has a Poisson distribution with rate $\int_{l_i}^{r_i} \lambda(t)dt$. Thus when the rate is constant ($\lambda(t) \equiv \lambda, \forall t \geq 0$), we have $N_i \sim \text{Poisson}(\lambda \cdot (r_i - l_i))$. Figure 3.2 illustrates this definition of a Poisson process by showing the simulated number of events on each of the unit intervals from 0 to 20 for two Poisson processes with rates $\lambda(t) = 50$ and $\lambda(t) = (t - 10)^2$.

A homogeneous Poisson process on $\mathbb{R}^+$ with a constant rate λ> 0 may equivalently be defined as a process $\{N_t\}_{t\geq 0}$ with $N_0 = 0$, holding times $\tau_0^{(\lambda)}, \tau_1^{(\lambda)}, \ldots$ satisfying $\tau_i^{(\lambda)} \sim \text{Exp}(\lambda)$ and jump chain $N_i = i$ for $i = 0, 1, \ldots$. Figure 3.3 illustrates this definition of a Poisson Process when $\lambda = 1$.

When the rate is an inhomogeneous function of time, a Poisson process may be defined in terms of exponentially distributed holding times as follows. For $\xi \sim \text{Exp}(1)$, holding time $\tau_0^{(\lambda)}$ satisfies:

$$\tau_0^{(\lambda)} = \inf\left\{t \geq 0 : \int_0^t \lambda(u)du \geq \xi\right\}.$$

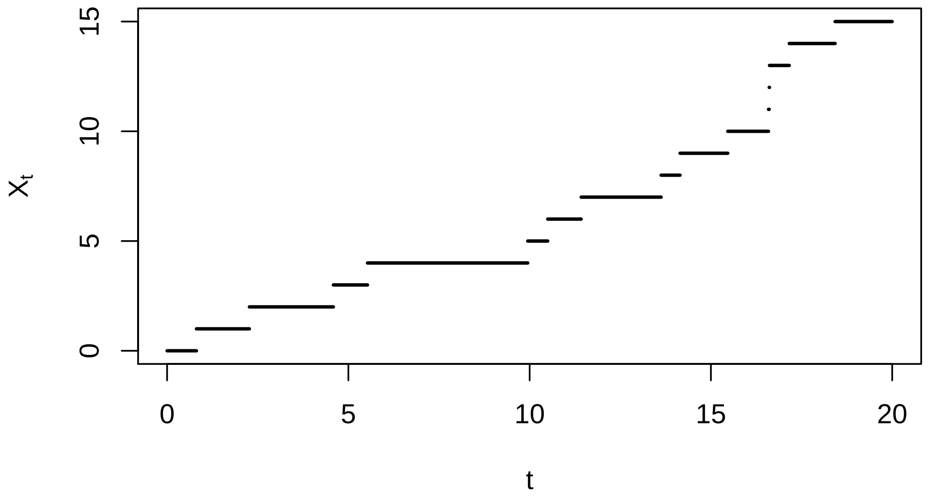


Figure 3.3: A Poisson Process with rate $\lambda = 1$.

Note that under this definition one can show that, as desired:

$$\mathbb{P}\left[\tau_0^{(\lambda)} < s\right] = 1 - \exp\left\{-\int_0^s \lambda(s)ds\right\}.$$

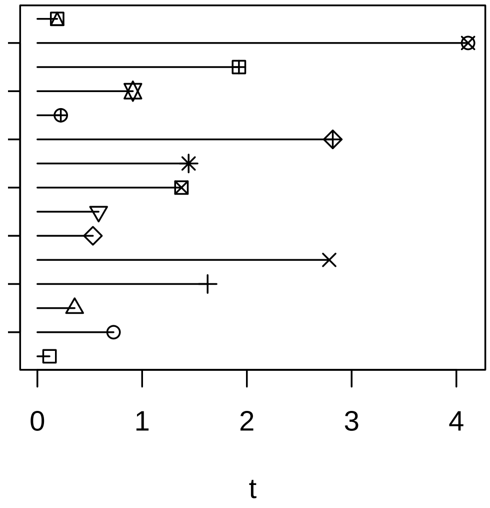
The following theorem concerns the superposition of Poisson processes, see for example Section 2.4 of Norris (1997). It is stated in terms of two Poisson processes but may naturally be extended to more than two, by induction.

Theorem 3 (The Superposition Theorem)**.** *If* $\{N_t\}_{t\geq 0}$ *and* $\{M_t\}_{t\geq 0}$ *are independent Poisson processes of rates* $\lambda_1(t)$ *and* $\lambda_2(t)$ *respectively. Then process* $\{N_t + M_t\}_{t\geq 0}$ *is a Poisson process of rate* $(\lambda_1 + \lambda_2)(t) := \lambda_1(t) + \lambda_2(t)$*.*

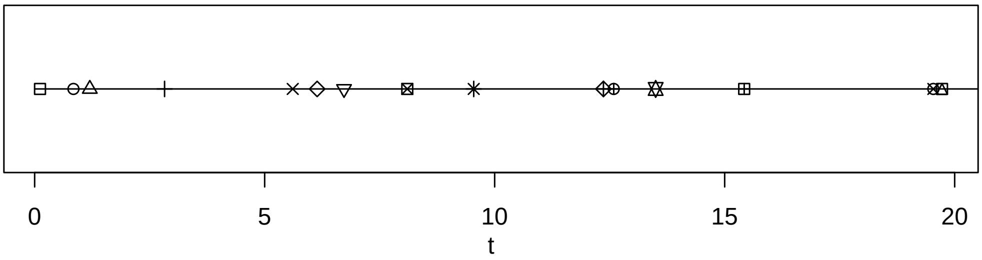
Simulation

A homogeneous Poisson process may be simulated using the *exponential spacings method* (Devroye, 1986, Algorithm proceeding Theorem 1.2. of Chapter 6). That is, use the fact that $\tau_i^{(\lambda)} \sim \text{Exp}(\lambda)$ for $i = 0, 1, \ldots$ to successively simulate holding times. Figure 3.4 aims to illustrate the exponential spacings method when $\lambda = 1$. Here, 15 holding times are simulated and used to compute the first 15 arrival times of an event from the Poisson process. The i-th Exponential random variable and i-th arrival time are plotted with the same unique symbol for $i = 0, 1, \ldots, 14$.

When the rate is inhomogeneous, one can no longer use independent and identically distributed exponential random variables to construct the Poisson process. If it is possible to compute and invert the integrated rate function, then analogous to the inversion sampling method of subsection 2.1.1, this can be used to simulate the inhomogeneous Poisson process. However, one can't always compute and invert the integrated rate function.



(a) Holding time $\tau_0^{(\lambda)}, \tau_1^{(\lambda)}, \ldots \tau_{14}^{(\lambda)}$ (from bottom to top).



(b) Arrival times.

Figure 3.4: The Exponential Spacing Method for holding rate $\lambda = 1$. Exponential random variable i and arrival time i are plotted with the same unique symbol.

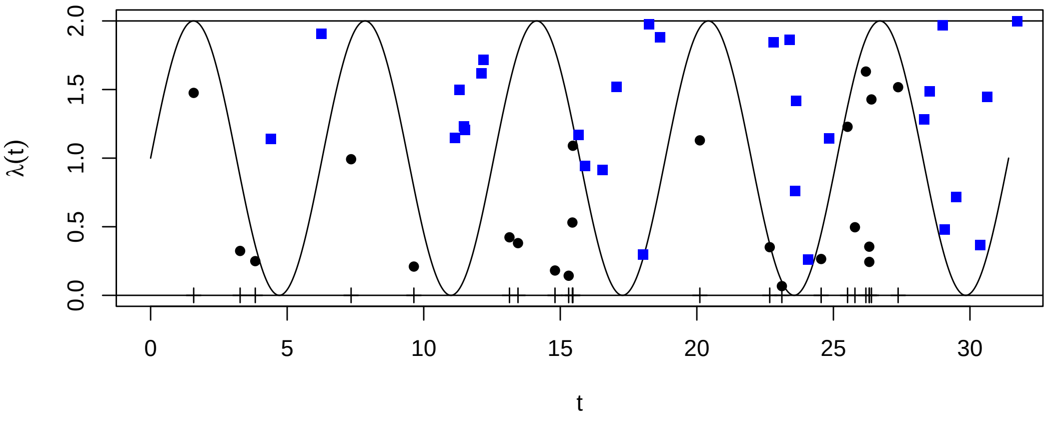


Figure 3.5: Simulation of a Poisson process with rate $\lambda(t) = 1 + \sin(t)$ using Poisson thinning with dominating rate $\bar{\lambda} = 2$. Crosses with y-co-ordinate zero mark the arrival times of events. Black circular dots show the simulated uniform random variables that resulted in events from the dominating process being accepted. Similarly, blue squares show the simulated uniform random variables for rejected events from the dominating process.

Instead, assuming the rate is bounded above by some constant $\bar{\lambda}$, one can instead use *Poisson Thinning* (Lewis and Shedler, 1979). Simulate $\{N_t\}_{t\geq 0}$ a Poisson process with constant rate $\bar{\lambda}$. Then construct $\{M_t\}_{t\geq 0}$ from $\{N_t\}_{t\geq 0}$ by including each event in $\{M_t\}_{t\geq 0}$ with probability $\lambda(t)/\bar{\lambda}$. Figure 3.5 illustrates Poisson thinning for $\lambda(t) = 1 + \sin(t)$ and $\bar{\lambda} = 2$. The arrival times of the Poisson process are shown by crosses with y-coordinate zero. Black circles and blue squares show the uniform random variables that determine whether events from the dominating Poisson process are accepted or rejected.

Suppose, as in Theorem 3, that a Poison process $\{N_t\}_{t\geq 0}$ has rate function $\lambda = \lambda_1 + \lambda_2$, where λ_1 is the rate of a Poisson process $\{N_t^{(1)}\}_{t\geq 0}$ and λ_2 is the rate of a Poisson process $\{N_t^{(2)}\}_{t\geq 0}$. Then the first arrival time of an event from $\{N_t\}_{t\geq 0}$ may be simulated by taking the minimum of the first arrival times of events from processes $\{N_t^{(1)}\}_{t\geq 0}$ and $\{N_t^{(2)}\}_{t\geq 0}$ (Devroye, 1986, Chapter 6, Section 1.3, The composition method). This *Composition method*, which relies on Theorem 2, is crucial to simulating the Restore processes in this thesis, since they involve more than one Poisson process. Figure 3.6 visualises the composition method for two Poisson processes $\{N_t^{(1)}\}_{t\geq 0}$ and $\{N_t^{(2)}\}_{t\geq 0}$ with rates $\lambda_1 = 1$ and $\lambda_2 = 2$ respectively. Black lines ending with a square show the time to the next arrival of an event from $\{N_t^{(1)}\}_{t\geq 0}$. Red lines ending with a round dot show the time to the next arrival of an event from $\{N_t^{(2)}\}_{t\geq 0}$. Poisson process $\{N_t\}_{t\geq 0}$ with rate $\lambda = \lambda_1 + \lambda_2$ is simulated as the composition of these Poisson processes.

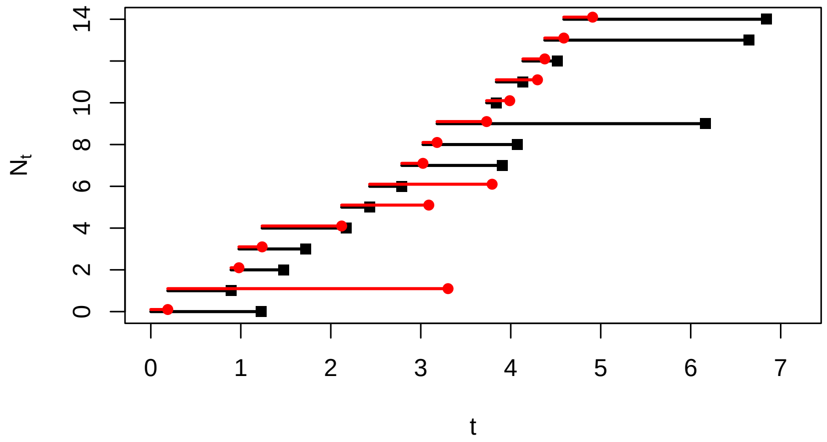


Figure 3.6: The Composition method: simulation of a Poisson process with rate $\lambda = 3$ as the composition of Poisson processes with rates $\lambda_1 = 1$ (holding times shown by black lines ending with squares) and $\lambda_2 = 2$ (holding times shown by red lines ending with circles).

3.3 Brownian Motion

Brownian Motion is a stochastic process of great interest in the field of probability. It is defined as follows (Mörters and Peres, 2010, Definition 1.1):

Definition 3.3.1 (Brownian Motion)**.** *A stochastic process $\{B_t : t \geq 0\}$ is called a Brownian motion starting at $x \in \mathbb{R}$ if*

1. $B_0 = x$;

2. *(Independent increments) $\forall t_1, t_2, \ldots, t_n$ with $0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$, it holds that $B_{t_n} - B_{t_{n-1}}, B_{t_{n-1}} - B_{t_{n-2}}, \ldots, B_{t_2} - B_{t_1}$ are independent random variables;*

3. *(Normally distributed increments) $\forall t \geq 0$ and $h > 0$, increment $B_{t+h} - B_t$ has distribution $\mathcal{N}(0, h)$;*

4. *(Continuous) Almost surely, the function $t \mapsto B_t$ is continuous.*

Brownian Motion exists (Mörters and Peres, 2010, Theorem 1.3). That is, though non-trivial, it is possible to show that there are no contradictions in the conditions imposed by the definition of Brownian Motion.

Since in any finite time-interval, a path of Brownian Motion takes an infinite number of values, it is impossible to simulate and/or store the whole path on a computer. However, due to its independent and normally distributed increments, it's very easy to simulate a finite dimensional distribution of Brownian Motion.

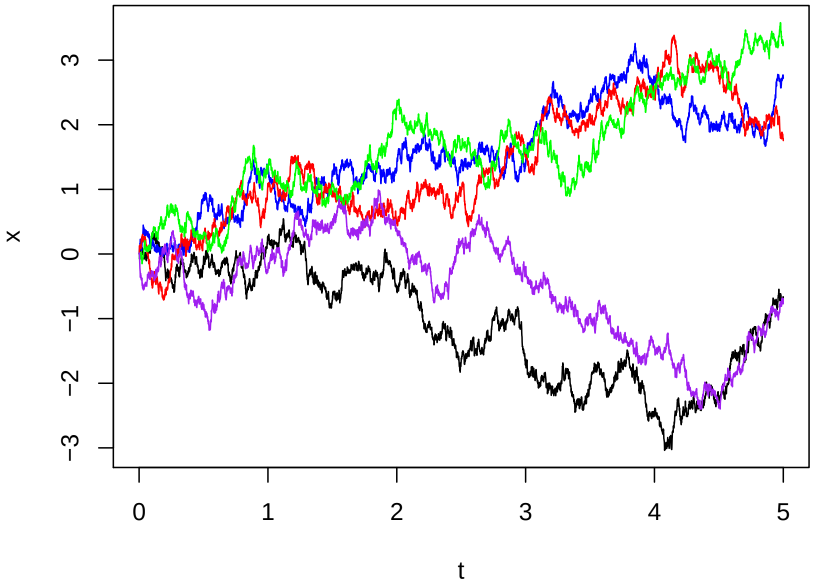


Figure 3.7: Sample paths of Brownian Motion

Starting at time t_0, with B_{t_0} known, for $t_1 > t_0$ we have that $B_{t_1} \sim \mathcal{N}(B_{t_0}, t_1 - t_0)$. Figure 3.7 shows 5 samples path of (finite-dimensional skeletons of) Brownian Motion.

Chapter 4

Regeneration-enriched Monte Carlo methods

As detailed in Chapter 2, there are several weaknesses of standard MCMC methods that one could hope to improve. Firstly, though combinations of transition kernels may be used, each transition kernel in the cycle or mixture must be π-invariant. When used to generate a Markov chain, there is no known way for non-π-invariant transition kernels to somehow *compensate* for each other, so that the chain is π-invariant. A transition kernel P is usually constructed to be π-invariant by ensuring that it is reversible, i.e. satisfies the detailed balance condition, given by equation (2.7). However, there is evidence that non-reversible Markov chains are superior (Neal, 2004; Suwa and Todo, 2010; Turitsyn et al., 2011; Chen and Hwang, 2013). Regeneration in Markov chains is desirable (see subsection 2.3.3), as well as using moves that operate both on a local and global scale. Unfortunately though, methods for regenerative MCMC (Mykland et al., 1995; Gilks et al., 1998; Brockwell and Kadane, 2005), which are based on Nummelin's splitting technique Nummelin (1978), don't scale well to high-dimensional problems. For example, when using a split Independence Sampler to simulate regeneration times, it becomes more difficult to find an independent proposal distribution that is a close fit to the target distribution. For instance, Ahn et al. (2013) found in their experiments that their method, which relies on a split Independence Sampler, was effective only up to 50-dimensional target distributions.

This section introduces the Restore process (Wang et al., 2021), a continuous-time Markov process satisfying the properties listed above, which may be used for Monte Carlo. The Restore process is able to combine dynamics, which by themselves are not π-invariant, in such a way that the dynamics compensate for each other and together are π-invariant. The Restore process is non-reversible, regenerative and makes moves on both local and global scales.

Restore processes are a class of continuous-time Markov processes $\{X_t\}_{t\geq 0}$, defined by enriching an existing Markov process $\{Y_t\}_{t\geq 0}$ with regenerations from some distribution μ at rate κ. The process thus consists of *local* and *global* dynamics. The local dynamics determine how the process changes between regenerations. The underlying process $\{Y_t\}_{t\geq 0}$ might be a jump process or a diffusion. Global dynamics dictate when and how the process regenerates. Global moves occur at the arrival times of a state-dependent Poisson process with rate $\kappa(x)$ referred to as the *regeneration rate*. At these times the process is reinitialised from some *regeneration distribution* μ. Unlike Nummelin's splitting technique (Nummelin, 1978), which retrospectively simulates whether regeneration has occurred after a move by the Markov chain, the Restore process is fundamentally built around regeneration.

This chapter reviews previous work on the Restore process. First, Section 4.1 provides the formal definition of the process. Next, Section 4.2 gives the regeneration rate and algorithm for enriching a jump process with regenerations so that the resulting process in π-invariant. Brownian Motion may be used within the Restore framework — this case is covered in Section 4.3. The concept of the minimal regeneration distribution is central to this thesis and is introduced in Section 4.4.

4.1 The Restore Process and Sampler

The Restore process is defined as follows. Let $\{Y_t\}_{t\geq 0}$ be a jump process or diffusion. The regeneration rate κ must be a locally bounded measurable map from the state space to the non-negative half-line. Define the *tour length* as

$$\tau = \inf\left\{t \geq 0 : \int_0^t \kappa(Y_s)ds \geq \xi\right\}, \tag{4.1}$$

for $\xi \sim \text{Exp}(1)$ independent of $\{Y_t\}_{t\geq 0}$ and $\inf \emptyset = \infty$. Let μ be some distribution and $\left(\{Y_t\}_{t\geq 0}^{(i)}, \tau_i\right)_{i=1}^{\infty}$ be i.i.d realisations of $(\{Y_t\}_{t\geq 0}, \tau)$ with $Y_0 \sim \mu$. The regeneration times are $T_0 = 0$ and $T_j = \sum_{i=1}^{j} \tau_i$ for $j = 1, 2, \ldots$. Then the Restore process $\{X_t\}_{t\geq 0}$ is given by:

$$X_t = \sum_{i=1}^{\infty} \mathbb{1}_{[T_{i-1},T_i)}(t) Y^{(i)}_{t-T_{i-1}}.$$

4.1.1 Invariance and Convergence

For $\mathcal{L}_Y$ the generator of $\{Y_t\}_{t\geq 0}$, the generator of $\{X_t\}_{t\geq 0}$ is (Wang et al., 2021, equation 1):

$$\mathcal{L}_X f(x) = \mathcal{L}_Y f(x) + \kappa(x) \int [f(y) - f(x)]\mu(y)dy. \tag{4.2}$$

To use the Restore process for Monte Carlo integration one chooses κ so that $\{X_t\}_{t\geq 0}$ is π-invariant. Recall that $\mathcal{L}_Y^\dagger$ is the adjoint of $\mathcal{L}_Y$, as defined in Section 3.1. The choice of κ that results in $\{X_t\}_{t\geq 0}$ being π-invariant is (Wang et al., 2021, equation 5)

$$\kappa(x) = \frac{\mathcal{L}_Y^\dagger \pi(x)}{\pi(x)} + C\frac{\mu(x)}{\pi(x)}, \tag{4.3}$$

for $C > 0$ a constant such that $\kappa(x) \geq 0, \forall x \in \mathcal{X}$. Finding an appropriate measure $C\mu$ such that κ is non-negative everywhere is non-trivial and motivates the work in Chapter 5.

As noted by (Wang et al., 2021), proving π-invariance of the Restore process in a general setting by analysing its generator is very difficult; the generator approach runs into highly technical difficulties. Instead, separately considering the specific cases where $\{Y_t\}_{t\geq 0}$ is a jump process or diffusion, one can show that $\{X_t\}_{t\geq 0}$ is π-invariant *without* using the generator approach.

Equation (4.3) may be written as:

$$\kappa(x) = \tilde{\kappa}(x) + C\frac{\mu(x)}{\pi(x)}. \tag{4.4}$$

The form of $\tilde{\kappa}$, the *partial regeneration rate*, depends on whether $\{Y_t\}_{t\geq 0}$ is a jump process or diffusion. We call C the *regeneration constant* and $C\mu$ the *regeneration measure*. Sometimes κ is referred to as the *full regeneration rate*.

Given π-invariance of $\{X_t\}_{t\geq 0}$, due to the regenerative structure of the process, we have (Wang et al., 2021, Theorem 6)

$$\mathbb{E}_\pi[f] = \frac{\mathbb{E}_{X_0\sim\mu}[\int_0^{\tau_1} f(X_s)dx]}{\mathbb{E}_{X_0\sim\mu}[\tau_1]}$$

and almost sure convergence of the ergodic averages: as $t \to \infty$,

$$\frac{1}{t}\int_0^t f(X_s)ds \to \mathbb{E}_\pi[f]. \tag{4.5}$$

The process can thus be used for Monte Carlo and the resulting method is called the *Restore Sampler*. The next subsection gives a Central Limit Theorem for estimators of expectations based on Restore processes.

4.1.2 Central Limit Theorem

A Central Limit Theorem for the Restore process holds, for a function $f : \mathcal{X} \to \mathbb{R}$, under the following assumptions (Wang et al., 2021, Assumption 23): the rate $\kappa : \mathcal{X} \to \mathbb{R}^+$ is locally bounded, the process $\{X_t\}_{t\geq 0}$ is irreducible, $\mathbb{E}_\mu[\tau^2] < \infty$ and f satisfies

$$\mathbb{E}_\mu \left[\left(\int_0^\tau f(X_s)ds \right)^2 \right] < \infty.$$

For $i = 1, 2, \ldots,$ define:

$$H_i := \int_{T_{i-1}}^{T_i} f(X_s)ds. \tag{4.6}$$

The Central Limit Theorem for Restore processes (Wang et al., 2021, Theorem 24) states that

$$\sqrt{n} \left(\frac{\int_0^{T_n} f(X_s)ds}{T_n} - \mathbb{E}_\pi[f] \right) \to \mathcal{N}(0, \sigma_f^2),$$

where convergence is in distribution and

$$\sigma_f^2 := \frac{\mathbb{E}_{X_0\sim\mu}\left[(H_1 - \tau_1 \mathbb{E}_\pi[f])^2 \right]}{\left(\mathbb{E}_{X_0\sim\mu}[\tau_1]\right)^2}. \tag{4.7}$$

Evidently the estimator's variance depends on the expected tour length. This motivates the need to make a good choice of μ, which doesn't result in tours being too short.

Recall from subsection 2.3.3 that for a regenerative Markov chain consisting of n tours, $\sqrt{n}(\hat{\pi}_{T_n}[f] - \pi[f]) \to \mathcal{N}(0, \sigma^2)$ in distribution and a consistent estimator for σ^2 is:

$$\hat{\sigma}_n^2 = \frac{\frac{1}{n}\sum_{i=1}^n \left(H_i - \hat{\pi}_{T_n}[f]\tau_i\right)^2}{\bar{\tau}^2}.$$

The same equation is a consistent estimator for σ_f^2 in (4.7), except that the relevant inputs to the equation are computed from a sample path of a continuous-time Markov process. Namely, H_i is given by (4.6); τ_i are the tour lengths for $i = 1, 2, \ldots, n$; $\bar{\tau}$ is the average tour length and

$$\hat{\pi}_{T_n}[f] = \frac{\int_0^{T_n} f(X_s)ds}{T_n}.$$

4.2 Jump Process Enriched with Regenerations

Recall from subsection 3.2.2 that a jump process has a discrete-time transition kernel P and some holding rate $\lambda : \mathcal{X} \to \mathbb{R}^+$. Transition kernel P determines how the process moves at jump times whilst λ dictates the length of time spent in each state. Let P have *integral kernel* $p(x,y)$ on $\mathcal{X} \times \mathcal{X}$, so that for a measure ν we have $\nu P(dx) = \int_{\mathcal{X}} \nu(dy)p(y,x)dx$. The jump process enriched with regenerations at rate

$$\kappa(x) = \frac{\int_{\mathcal{X}} \lambda(y)\pi(y)p(y,x)dy - \lambda(x)\pi(x)}{\pi(x)} + C\frac{\mu(x)}{\pi(x)}, \tag{4.8}$$

for $C > 0$ a constant such that $\kappa(x) \geq 0, \forall x \in \mathcal{X}$, is π-invariant (Wang et al., 2021, Section 3.2) and called a *Restore jump process.* Wang et al. (2021) prove that the Restore jump process is π-invariant by showing that $\pi Q_t^\mu f = \pi[f]$ for any continuous bounded function f and for all $t \geq 0$, where $\{Q_t^\mu : t \geq 0\}$ is the continuous-time semigroup of $\{X_t\}_{t\geq 0}$. The proof deliberately avoids the generator method for proving π-invariance of the process, since such an approach here runs into technical difficulties.

This regeneration rate may be expressed in the form of equation (4.4), with the partial regeneration rate defined as:

$$\tilde{\kappa}(x) = \frac{\int_{\mathcal{X}} \lambda(y)\pi(y)p(y,x)dy - \lambda(x)\pi(x)}{\pi(x)}.$$

See Section 3.2 of Wang et al. (2021) on how to extend this construction to underlying jump processes with transition kernels that do not have an integral kernel, such as the kernel of the Metropolis-Hastings algorithm.

The normalization constant for π does not need to be known. If $\pi(x) = \tilde{\pi}(x)/Z$ and Z is unknown, then the regeneration rate becomes, for $\tilde{C} = CZ$:

$$\kappa(x) = \frac{\int_{\mathcal{X}} \lambda(y)\tilde{\pi}(y)p(y,x)dy - \lambda(x)\tilde{\pi}(x)}{\tilde{\pi}(x)} + \tilde{C}\frac{\mu(x)}{\tilde{\pi}(x)}.$$

We say that the normalizing constant is *absorbed* into the regeneration constant.

The holding and regeneration rates must satisfy (Wang et al., 2021, Assumption 19) $\int \lambda(x)\pi(x)dx < \infty$ and $\int(\lambda(x)+\kappa(x))^2\pi(x)dx < \infty$. The first condition is necessary for the process to be non-explosive. The latter condition is used to justify an exchange of integration and differentiation in the proof of π-invariance of the jump process. When used for Monte Carlo, such a process is called a *Jump Process Restore Sampler* (JPRS).

Since $\{X_t\}_{t\geq 0}$ is piecewise constant, evaluating the left-hand side of (4.5) is easy. Let $\tilde{X}_i$ denote the i-th element of the jump chain and $\tau_i^{(\lambda+\kappa)}$ the corresponding holding time. Note that $\tilde{X}_i$ is different to X_t as the first quantity is indexed by its position in the jump chain and the second by time. The quantities are related in that

$$X_t = \begin{cases} \tilde{X}_0 \ \forall\ t \in [0, \tau_0^{(\lambda+\kappa)}), \\ \tilde{X}_i \ \forall\ t \in \left[\sum_{j=0}^{i-1} \tau_j^{(\lambda+\kappa)}, \sum_{j=0}^{i} \tau_j^{(\lambda+\kappa)}\right) \ \text{for } i > 0. \end{cases}$$

Say the length of the jump chain is m and that $\sum_{i=0}^{m-1} \tau_i^{(\lambda+\kappa)} = T$. Then the left-hand side of equation (4.5) can be rewritten as a weighted sum:

$$\frac{1}{T}\int_0^T f(X_t)dt = \frac{1}{T}\sum_{i=0}^{m-1} \tau_i^{(\lambda+\kappa)} f(\tilde{X}_i).$$

Pseudocode for simulating n tours of $\{X_t\}_{t\geq 0}$ is given by Algorithm 6. Here, index i counts the number of tours simulated. The composition method of subsection 3.2.3 is used to simulate the holding times of the enriched process, for which the holding rate is the sum of the holding rate λ of the underlying process and the regeneration rate κ. Whenever the process moves to a new state, two exponential random variables are simulated, which are the arrival times of events from rate λ and κ Poisson processes. The sampler makes a local or global move according to whichever event arrives first. The algorithm outputs are the jump chain and holding times of the jump process, as well as the tour number corresponding to each segment of the jump process. The regenerative structure of the process can provide an estimate of the variance of the estimator itself, as explained in subsection 4.1.2, which is why it is useful to record the tour numbers as output.

Algorithm 6: The Jump Process Restore Sampler

```
$\tilde{X} \sim \mu, i \leftarrow 0$
while $i < n$ do
    $\tau^{(\lambda)} \sim \text{Exp}(\lambda(\tilde{X}))$
    $\tau^{(\kappa)} \sim \text{Exp}(\kappa(\tilde{X}))$
    $\tau^{(\lambda+\kappa)} \leftarrow \tau^{(\lambda)} \wedge \tau^{(\kappa)}$
    Record $\tilde{X}, \tau^{(\lambda+\kappa)}, i$
    if $\tau^{(\lambda)} < \tau^{(\kappa)}$ then
        $\tilde{X} \sim P(\tilde{X}, \cdot)$
    else
        $\tilde{X} \sim \mu$
        $i \leftarrow i + 1$
    end
end
```

A special case of the regeneration rate for the JPRS is when P is already a π-invariant Markov transition kernel, such as that of a Metropolis-Hastings sampler (see subsection 2.3.2) with invariant distribution π. Taking $\lambda \equiv 1$, the regeneration rate then reduces to:

$$\kappa(x) = C\frac{\mu(x)}{\pi(x)}. \tag{4.9}$$

Figure 4.1 shows an example of a Restore jump process with invariant distribution a mixture of Gaussians:

$$\pi(x) = \frac{1}{2}\mathcal{N}(x; m_1, \Sigma_1) + \frac{1}{2}\mathcal{N}(x; m_2, \Sigma_2),$$

for $m_1 = (-2, -2), m_2 = (2, 2)$,

$$\Sigma_1 = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \text{ and } \Sigma_2 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}.$$

The process used a π-invariant Random Walk Metropolis Markov transition kernel, with variance of the proposal distribution equal to 0.25^2. The regeneration constant was set as $C = 0.01$. The regeneration distribution was $\mu \equiv \mathcal{N}(0, 3I_2)$. The first 8 tours of the process are shown, each in a different colour, in Figure 4.1. The dynamics of the process are such that local moves are small and tend to result in exploration of one of the distribution's modes. At regeneration instances, the process tends to do one of the following: (i) jump to the other mode; (ii) make a large jump to another location in the same mode; (iii) jump to a state far from either mode, in which case it tends to quickly regenerate again.

Chapter 6 makes a contribution to existing methodology by showing how a non-π-invariant jump process may be adjusted with regenerations, so that the resulting jump process is π-invariant.

4.3 Regeneration-enriched Brownian Motion

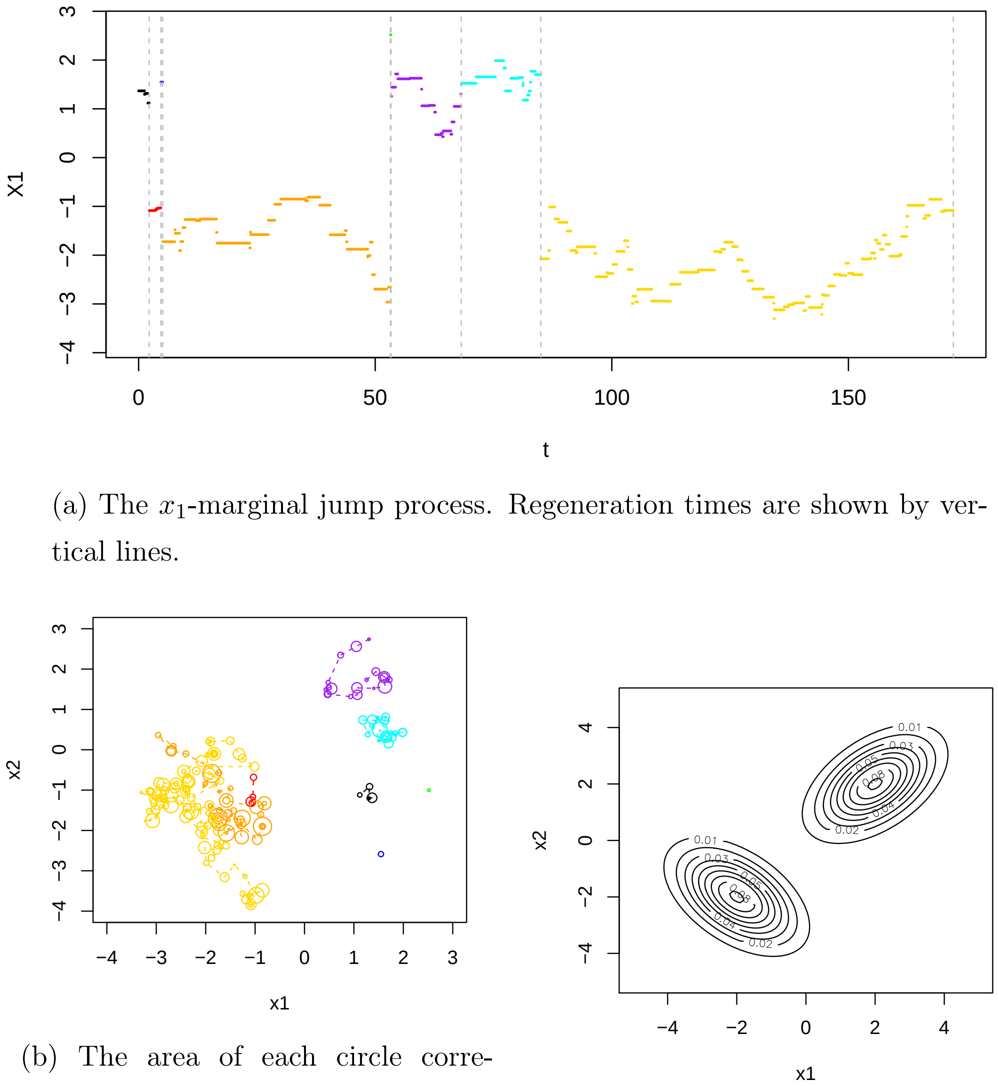
Recall that $U(x) = -\log \pi(x)$ is the energy corresponding to a target distribution π. Let μ be some distribution and rate κ be given by

$$\kappa(x) = \frac{1}{2}\left(||\nabla U(x)||^2 - \Delta U(x)\right) + C\frac{\mu(x)}{\pi(x)}, \tag{4.10}$$

where C is a constant so that $\kappa(x) \geq 0, \forall x \in \mathcal{X}$. Then a Brownian motion enriched with regenerations from μ at rate κ has invariant distribution π (Wang et al., 2021, Section 3.1) and is called a *Brownian Motion Restore process.* The normalizing constant of π doesn't need to be known, since it may be absorbed into the constant C, as in Section 4.2. The partial regeneration rate is

$$\tilde{\kappa}(x) := \frac{1}{2}\left(||\nabla U(x)||^2 - \Delta U(x)\right).$$

Note that equation (4.10) for the full regeneration rate may be expressed in terms of the partial regeneration rate using equation (4.4).



(a) The x_1-marginal jump process. Regeneration times are shown by vertical lines.

(b) The area of each circle corresponds to the holding time for each state. Lines connect consecutive states in the same tour.

(c) Contours of π.

Figure 4.1: Eight tours of a jump process enriched with regenerations. In (a) and (b), each tour is given a different colour. Here, π is a mixture of two Gaussian distributions, $\mu \equiv \mathcal{N}(0, 3I)$ and $C = 0.01$. The underlying jump process moves according to a Random Walk Metropolis Markov transition kernel with variance of the symmetric proposal distribution 0.25^2.

4.3.1 Poisson Thinning of Regeneration Times

Poisson thinning (see subsection 3.2.3) is used to simulate regeneration events. This is because the regeneration rate itself is a stochastic process given by $t \mapsto \kappa_t := \kappa(X_t)$ and hence no closed form for the right hand side of (4.1) is available. Suppose $\kappa(x) < K, \forall x \in \mathcal{X}$. That is,

$$K := \sup_{x \in \mathcal{X}} \kappa(x) < \infty.$$

Then $\kappa_t < K, \forall t \geq 0$ and K may be used as the dominating rate in Poisson thinning.

4.3.2 Truncating the Regeneration Rate

For many target distributions κ will *not* be bounded. Then, to use a global dominating rate, κ must be *truncated* at some level. Alternatively, when κ is bounded but the bound is very large, then for simulation purposes truncation may be desirable. When a truncated regeneration rate is used, we will denote the truncation level as $\mathcal{K}$. In other words, this notation signals the use of rate

$$\kappa(x) = \left(\tilde{\kappa}(x) + C\frac{\mu(x)}{\pi(x)}\right) \wedge \mathcal{K}$$

This truncation introduces error, so that the Monte Carlo approximation is no longer exact, however the error is negligible in practice as long as $\mathcal{K}$ is large enough. Indeed, it is in theory possible to quantify the size of this error (Wang et al., 2021, Theorem 30) and show that as $\mathcal{K}$ goes to infinity, the error tends to zero (Wang et al., 2021, Proposition 32). Bounding the error using Theorem 30 of Wang et al. (2021) may be difficult in challenging problems involving complex posterior distributions.

There are computational considerations for the choice of $\mathcal{K}$. As $\mathcal{K}$ increases, although the error introduced by the truncation decreases, the computational cost of simulating the process increases. This is because the rate κ is evaluated more often, which involves evaluating $\pi, \nabla U$ and ΔU. A major contribution of this thesis, developed in Chapter 5, is a method for adaptive simulation of a Restore process, which greatly decreases how large $\mathcal{K}$ needs to be. Subsection 5.3.1 in particular presents guidance on choosing the truncation level.

Figure 4.2a shows a sample path of a Brownian Motion Restore process and Figure 4.2b shows the corresponding regeneration rate along the sample path. In Figure 4.2b, the black line shows $\kappa(X_t)$. The t-coordinates of the green and red dots are potential regeneration times; the vertical co-ordinate shows sample values of random variables $U \sim \mathcal{U}(0, \mathcal{K})$. For a potential regeneration event occurring at time t, when $U > \kappa(X_t)$, regeneration does not happen and the dot is shown in green. When $U < \kappa(X_t)$, regeneration does happen and the dot is shown in red.

Subsection 5.3.1 explains how to choose $\mathcal{K}$ for a d-dimensional isotropic Gaussian distribution. This class of target distribution can guide the selection of $\mathcal{K}$ for more challenging posterior distributions, as long as the distributions of interest are sufficiently close to Gaussian.

4.3.3 Output

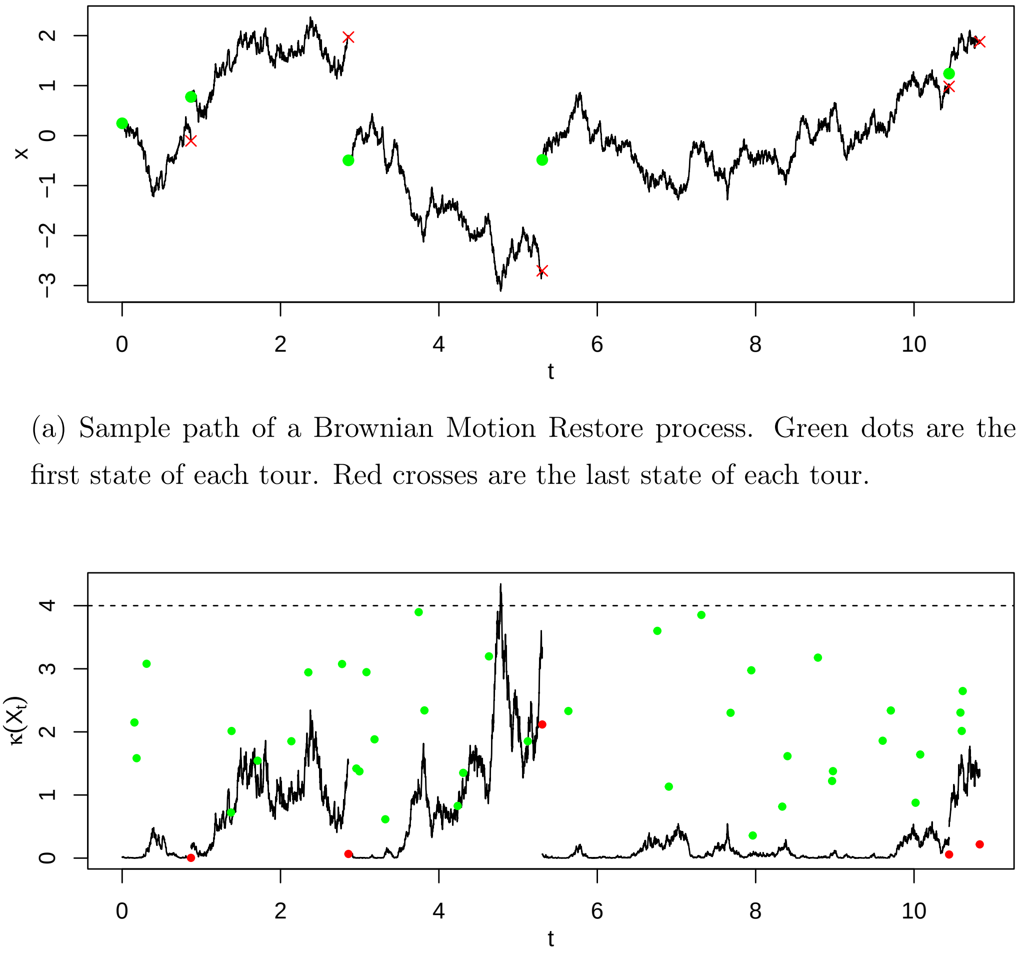
The left-hand side of equation (4.5) can't be evaluated exactly when the underlying process is a Brownian motion. Instead, the output of the sampler is the state of the process either at fixed, evenly-spaced intervals or at the arrival times of an exogenous, homogeneous Poisson process with rate Λ.

When output times are fixed, let $\{t_1, t_2, \dots\}$ be an evenly spaced mesh of times, with $t_i = i\Delta$ for $i = 1, 2, \dots$ and $\Delta > 0$ some constant. When output times are random, let $\{t_1, t_2, \dots\}$ be the events of a homogeneous Poisson process with rate $\Lambda > 0$. In either case, the output of the process is $\{X_{t_1}, X_{t_2}, \dots\}$. Suppose there are n output states, then we estimate expectations using the unbiased approximation:

$$\pi[f] \approx \frac{1}{n}\sum_{i=1}^{n} f(X_{t_i}).$$

4.3.4 Simulation

Algorithmically, there is little difference between using fixed and random output times. We use a homogeneous Poisson process to record output events, since this method is marginally simpler. The composition method of subsection 3.2.3 may be used to simulate the Poisson processes with rates $\mathcal{K}$ and Λ. That is, generate the arrival times $\tau^{(\mathcal{K})}$ and $\tau^{(\Lambda)}$ of events from the two processes. If $\tau^{(\Lambda)}$ arrives before $\tau^{(\mathcal{K})}$, simulate the Restore process forward in time by $\tau^{(\Lambda)}$ and record output. Else, simulate the Restore process forward in time by $\tau^{(\mathcal{K})}$ then use Poisson Thinning (see subsection 3.2.3) to determine whether regeneration happens. When using a fixed mesh of times, the composition method no longer applies, so one must keep track of the times of the next output and potential regeneration events. Algorithm 7 summarises how to simulate n tours of a Brownian Motion Restore process whilst recording output at the rate of an exogenous homogeneous Poisson



(a) Sample path of a Brownian Motion Restore process. Green dots are the first state of each tour. Red crosses are the last state of each tour.

(b) The stochastic process in black gives the value of the regeneration rate for the sample path above of a Brownian Motion Restore process. The horizontal dotted lines shows $\mathcal{K} = 4$. The time of simulation and value of the random variables used for Poisson thinning, $U_i \sim \mathcal{U}(0, \mathcal{K})$ with $i = 0, 1, \dots,$ are shown by the dots. For U simulated at time t, regeneration does not happen and the dot is green if $U > \kappa(X_t)$, else regeneration happens and the dot is red.

Figure 4.2: Sample path of a Brownian Motion Restore process and an illustration of Poisson thinning of regeneration times for $\pi \equiv \mathcal{N}(0, 1), \mu \equiv \mathcal{N}(0, 1/2), \mathcal{K} = 4$.

process.

Algorithm 7: The Brownian Motion Restore Sampler

```
$X \sim \mu, t \leftarrow 0, i \leftarrow 0$
while $i < n$ do
    $\tau^{(\mathcal{K})} \sim \text{Exp}(\mathcal{K}), \tau^{(\Lambda)} \sim \text{Exp}(\Lambda)$
    if $\tau^{(\Lambda)} < \tau^{(\mathcal{K})}$ then
        $t \leftarrow t + \tau^{(\Lambda)}, X \sim \mathcal{N}(X, \tau^{(\Lambda)})$. Record $X, t, i$
    else
        $t \leftarrow t + \tau^{(\mathcal{K})}, X \sim \mathcal{N}(X, \tau^{(\mathcal{K})}), u \sim \mathcal{U}[0, 1]$
        if $u < \kappa(X)/\mathcal{K}$ then
            $X \sim \mu, i \leftarrow i + 1$
        end
    end
end
```

Figure 4.3 shows plots of $\pi, \tilde{\kappa}, \kappa, \mu$ for $\pi \equiv \mathcal{N}(0,1), \mu \equiv \mathcal{N}(0,4)$ and $C = 1$, as well as a sample path consisting of 5 tours simulated using $\mathcal{K} = 100$. In this example, μ has heavier tails than π, so κ becomes becomes very large away from the origin. This results in a large variance in the tour lengths $\tau_1, \tau_2, \tau_3, \tau_4, \tau_5$, as can be seen in the plot of the sample path.

Figure 4.4 shows a sample path of a two-dimensional Brownian Motion Restore process when $\mu \equiv \mathcal{N}(0, I), \pi \equiv \mathcal{N}(0, \Sigma)$ and $\mathcal{K} = 100$, where.

$$\Sigma = \begin{pmatrix} 1.2 & 0.4 \\ 0.4 & 0.8 \end{pmatrix}. \tag{4.11}$$

Each tour has a different colour. The gray ellipses are the contours of π.

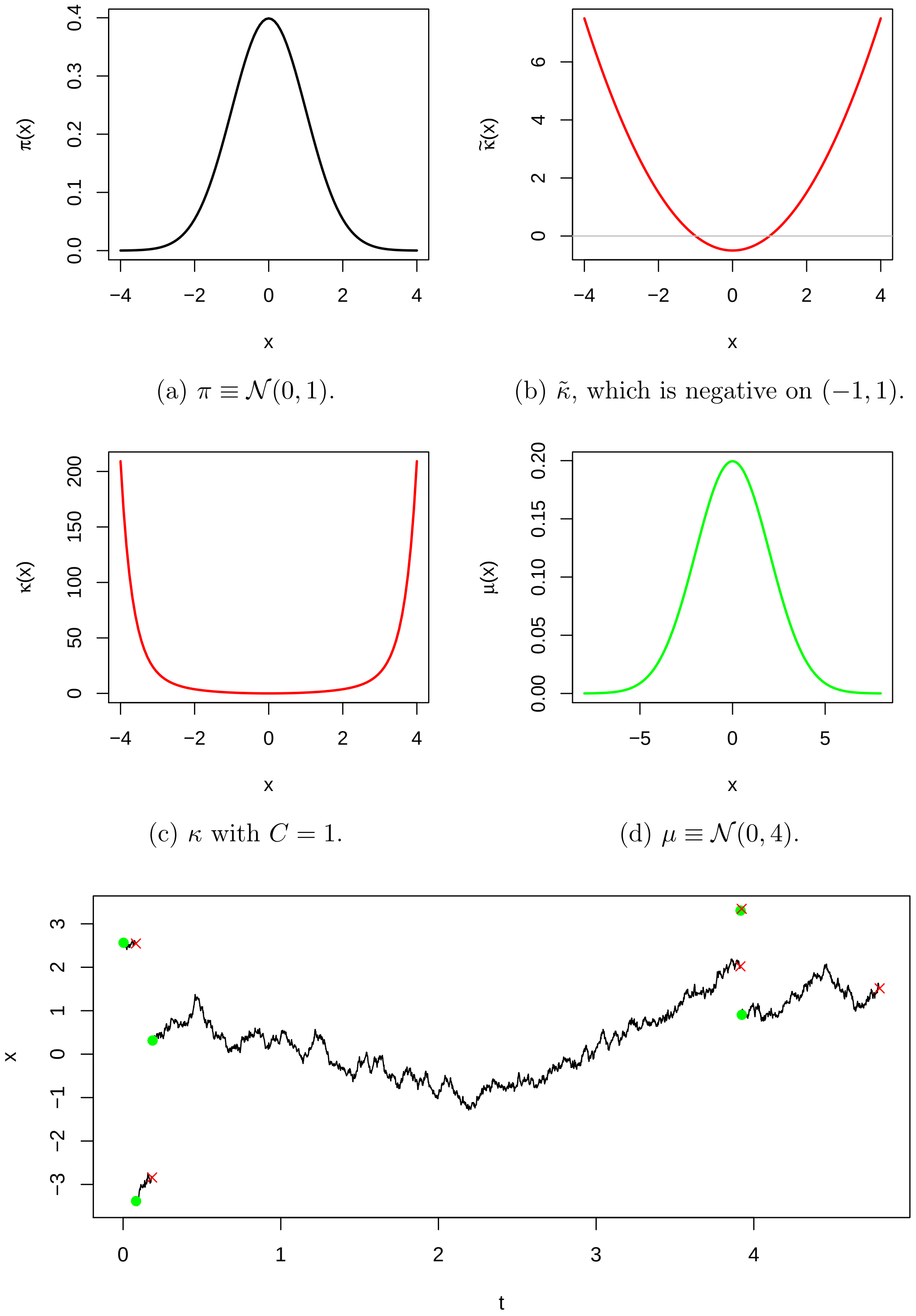
4.4 Minimal Regeneration Distribution

The *minimal regeneration measure* $C^+\mu^+$ (Wang et al., 2021, Section 5.1) is the regeneration measure for which the full regeneration rate is as small as possible. The minimal regeneration measure satisfies:

$$\tilde{\kappa}(x) + C^+ \frac{\mu^+(x)}{\pi(x)} = \tilde{\kappa}(x) \vee 0, \forall x \in \mathcal{X}. \tag{4.12}$$

Note that Wang et al. (2021) use notation C^* and μ^* instead of C^+ and μ^+. We call C^+ the *minimal regeneration constant* and μ^+ the *minimal regeneration distribution.* Rate $\kappa^+ : \mathcal{X} \to \mathbb{R}^+$ defined as

$$\kappa^+(x) := \tilde{\kappa}(x) + C^+ \frac{\mu^+(x)}{\pi(x)},$$



(a) $\pi \equiv \mathcal{N}(0,1)$.

(b) $\tilde{\kappa}$, which is negative on $(-1,1)$.

(c) κ with $C=1$.

(d) $\mu \equiv \mathcal{N}(0,4)$.

(e) Five tours of a restore process simulated with rate truncated at $\mathcal{K}=100$. Each tour's first output state is shown with a green dot and its last output state shown with a red cross.

Figure 4.3: Target and regeneration distributions, partial and full regeneration rates and a sample path of regeneration-enriched Brownian Motion.

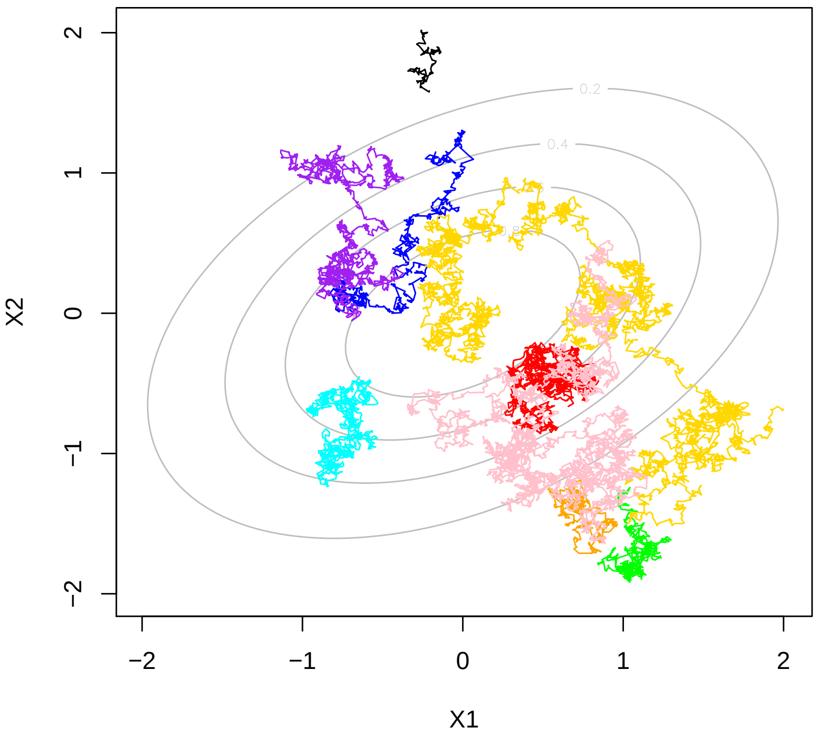


Figure 4.4: A sample path of a two-dimensional Brownian Motion Restore process with $\mu \equiv \mathcal{N}(0, I)$ and $\mathcal{K} = 100$. Contours of $\pi \equiv \mathcal{N}(0, \Sigma)$, for Σ given by equation (4.11), are shown in gray. Each of the 9 tours are plotted in a different colour.

is called the *minimal regeneration rate*, since for all C and μ, we have

$$\kappa^+(x) \leq \tilde{\kappa}(x) + C\frac{\mu(x)}{\pi(x)}, \forall x \in \mathcal{X}.$$

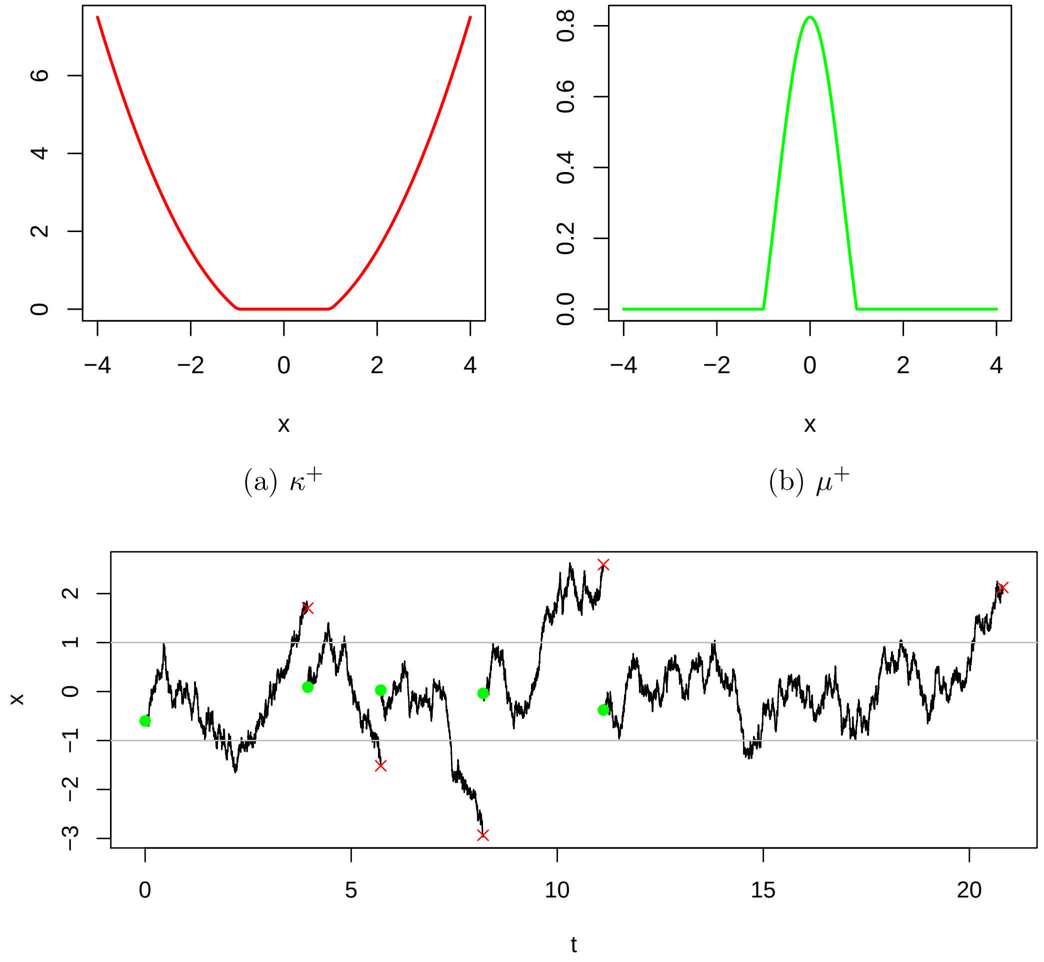
Rearranging (4.12) gives the pointwise expression for μ^+ as:

$$\mu^+(x) = \frac{1}{C^+}[0 \vee -\tilde{\kappa}(x)]\pi(x). \tag{4.13}$$

The Restore process under μ^+, κ^+, C^+ is referred to as *Minimal Restore*. Note that C^+ is the normalization constant of the unnormalized measure $[0 \vee -\tilde{\kappa}(x)]\pi(x)$. Since $\tilde{\kappa}$ is locally bounded, when the support of the function $[0 \vee -\tilde{\kappa}]$ is compact, this measure is normalizable. For all of the examples considered in Chapter 5, the support of $[0 \vee -\tilde{\kappa}]$ is compact. Indeed, this is a weak condition, which holds when π satisfies a suitable sub-exponential tail condition (Wang et al., 2019).

Figure 4.5 shows κ^+, μ^+ and a sample path of $\{X_t\}_{t\geq 0}$ for $\pi \equiv \mathcal{N}(0,1)$. The energy and its derivatives are $U(x) = x^2/2 + \text{const}, \nabla U(x) = x$ and $\Delta U(x) = 1$, so $\tilde{\kappa}(x) = (x^2 - 1)/2$. The density of the minimal regeneration distribution μ^+ is then:

$$\mu^+(x) = \begin{cases} \frac{\sqrt{e}}{2}(1 - x^2)\exp\{-x^2/2\}, & \text{if } x \in (-1, 1); \\ 0, & \text{otherwise.} \end{cases}$$



(a) κ^+

(b) μ^+

(c) Sample path consisting of 5 tours. Each tour's first output state is shown with a green dot and its last output state shown with a red cross. Tours always begin inside and end outside the interval $(-1, 1)$.

Figure 4.5: The minimal regeneration rate and distribution for a Brownian Motion Restore process with $\pi \equiv \mathcal{N}(0, 1)$, as well as a sample path of 5 tours.

Thus tours of the process always start in interval [-1,1] and end outside [-1,1].

More generally, an isotropic Gaussian in d dimensions has $||\nabla U(x)||^2 = x^T x$ and $\Delta U(x) = d$. Hence the partial regeneration rate is

$$\tilde{\kappa}(x) = \frac{1}{2}(x^T x - d),$$

so μ^+ has support on $\{x : x^T x < d\}$, which is a hyper-sphere with radius $\sqrt{d}$.

Recall that when the dominating rate is denoted $\mathcal{K}$, it signals that the regeneration rate is truncated at $\mathcal{K}$. An advantage of using κ^+ is that this dominating rate may be a lot lower. When κ^+ is truncated, let $\mathcal{K}^+$ denote the truncation level. That is, when the dominating rate is denoted $\mathcal{K}^+$, this signals that the rate used is in fact

$$\kappa(x) = \kappa^+(x) \wedge \mathcal{K}^+.$$

4.4.1 Characteristics of the Minimal Regeneration Rate for some Example Target Distributions

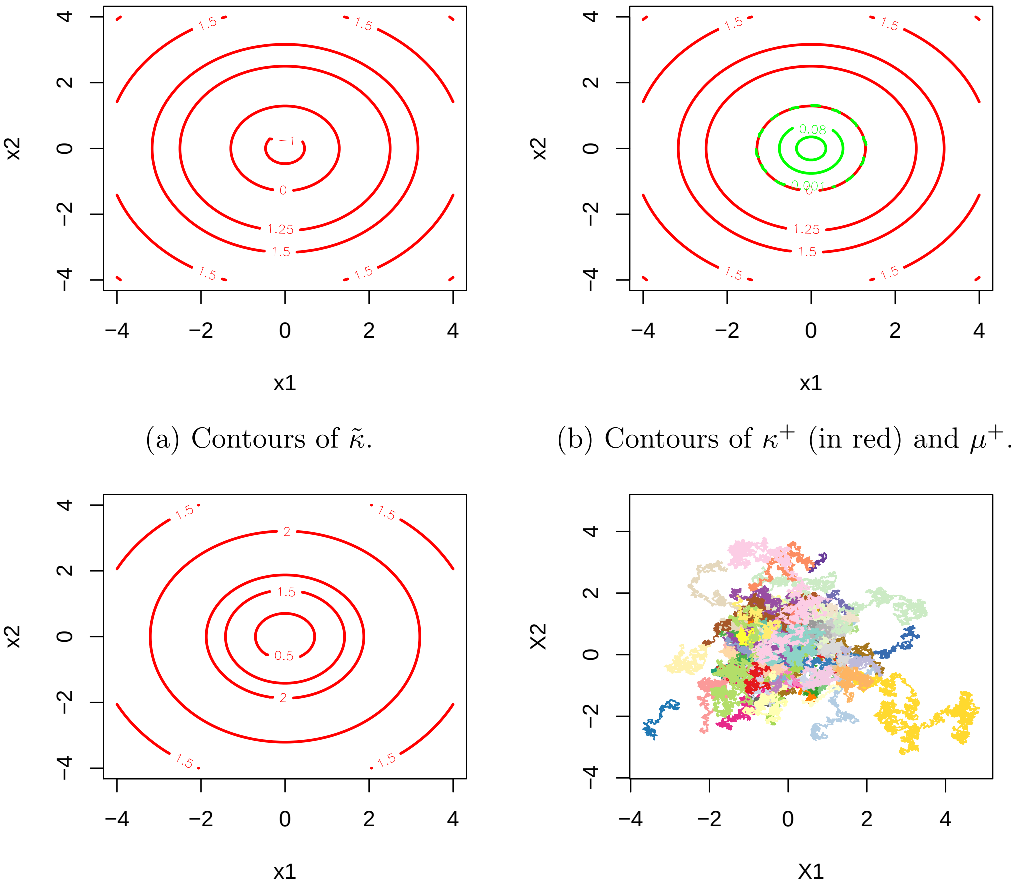
This subsection considers the characteristics of the shape of κ^+ for several choices of π. Later, we will relate the performance of the Adaptive Restore algorithm of Chapter 5 to these characteristics. In addition, Adaptive Restore assumes that μ^+ has compact support. Whilst this is the case for many target distributions, it is worth noting that μ^+ is not always compact, as some of the examples below show.

Heavy-tailed distribution can be difficult to sample from using MCMC methods. Recall that the multivariate t-distribution has density

$$\pi(x) \propto \left[1 + \frac{1}{\nu}(x-m)^T\Sigma^{-1}(x-m)\right]^{-(\nu+d)/2}.$$

and is heavy-tailed. For a multivariate t-distribution, the set $\{x \in \mathbb{R}^d : \tilde{\kappa}(x) < 0\}$ is compact and there exists a bounding rate $\tilde{K}$ such that $\tilde{\kappa}(x) < \tilde{K}, \forall x \in \mathbb{R}^d$. See Appendix B.1 for the expression for $\tilde{\kappa}(x)$. An interesting aspect of the geometry of $\tilde{\kappa}$ is that, starting at the mean and moving outwards to the tails, the rate starts off as negative, increases to a maximum then decays to zero. This geometry enables tours of the process to spend a long time in the tails of π before regenerating. Figure 4.6a shows contours of $\tilde{\kappa}(x)$ for $d = 2$ and $\pi \equiv t_{10}(0, I)$. Figure 4.6b shows contours of κ^+ in red and μ^+ in green. Figure 4.6c is a contour plot of κ when $\mu \equiv \mathcal{N}(0, I)$ and C is chosen so that the minimum of κ is approximately 0.01 (since this value is close to zero). Here, an appropriate bound for κ is $K = 2.23$, so simulation of the process is efficient compared to other target distributions for which the dominating rate used for Poisson thinning is a lot higher. Note that this dominating rate bounds κ everywhere and that no truncation of κ is necessary. Finally, 4.6d shows a sample path of a Brownian Motion Restore process with $\pi \equiv t_{10}(0, I), \mu \equiv \mathcal{N}(0, I)$ and κ as in Figure 4.6c.

Figure 4.7 shows contour plots of π, κ^+, κ (when $\mu \equiv \mathcal{N}(0, I)$) and μ^+ when π is a mixture of Gaussian distributions. See Appendix B.2 for derivations of the gradient and Laplacian of the log-density. Here, the set $\{x \in \mathbb{R}^d : \tilde{\kappa}(x) < 0\}$ is formed of two disjoint regions. This observation will be important when considering the performance of the Adaptive Restore method of Chapter 5 on multi-modal distributions.



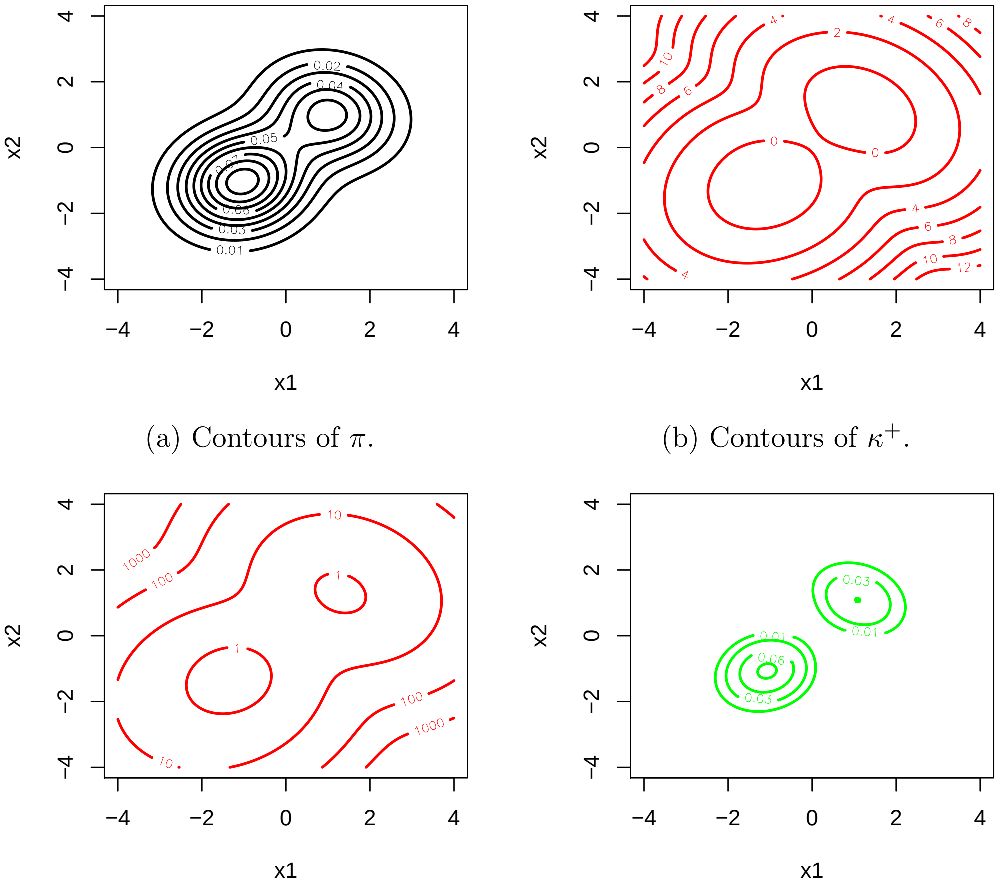
(a) Contours of $\tilde{\kappa}$.

(b) Contours of κ^+ (in red) and μ^+.

(c) Contours of κ for $\mu \equiv \mathcal{N}(0, I)$ and C such that $\min_{x \in \mathcal{X}} \kappa(x) \approx 0.01$.

(d) Sample path for $\mu \equiv \mathcal{N}(0, I)$ and C such that $\min_{x \in \mathcal{X}} \kappa(x) \approx 0.01$.

Figure 4.6: Contours of $\tilde{\kappa}, \kappa^+, \mu^+$ and κ (when $\mu \equiv \mathcal{N}(0, I)$) for $\pi \equiv t_{10}(0, I)$, as well as a sample path of a Brownian motion Restore process with tours plotted using different colours. Starting from the mean of π and moving out towards the tails, κ^+ first increases to its maximum value, then decays towards zero.



(a) Contours of π.

(b) Contours of κ^+.

(c) Contours of κ for $\mu \equiv \mathcal{N}(0, I)$.

(d) Contours of μ^+.

Figure 4.7: Contour plots of $\pi, \tilde{\kappa}, \kappa$ and μ^+ for a Gaussian mixture distribution.

For target distributions that are known to be challenging to sample from using MCMC, μ^+ is supported on small sub-regions of $\mathcal{X}$. We consider two such examples: the *Banana* and *Funnel* distributions. For the Banana distribution, it is difficult to tell from the expression for $\tilde{\kappa}(x)$ whether the region $\{x : \tilde{\kappa}(x) < 0\}$ is compact. For the Funnel distribution, the set $\{x : \tilde{\kappa}(x) < 0\}$ is not bounded. The methodology developed in Chapter 5, on adaptive simulation of Restore processes, makes the assumption that $\{x : \tilde{\kappa}(x) < 0\}$ is compact. Although this is a weak condition, which holds for π satisfying a sub-exponential tail condition (Wang et al., 2019), we present these target distributions here to demonstrate that $\{x : \tilde{\kappa}(x) < 0\}$ is not always compact.

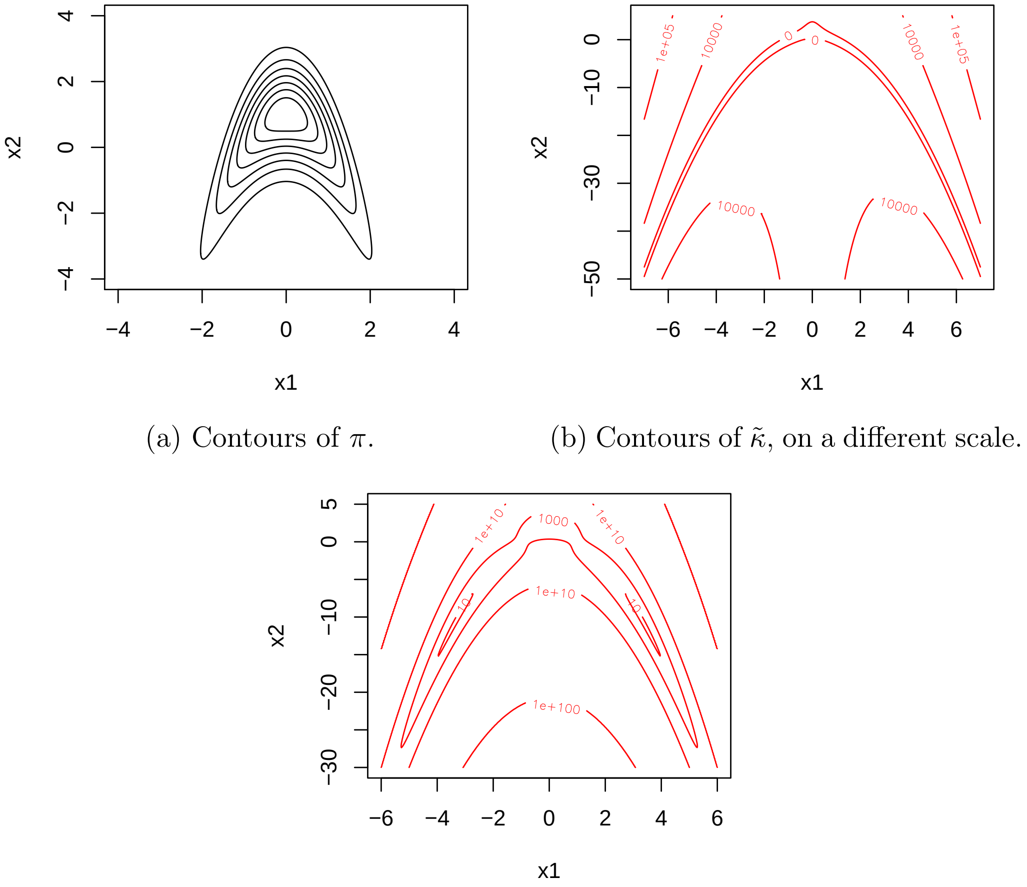
In addition, it is worth noting that Brownian Motion Restore processes work best for sampling random variables for which the covariance matrix is close to the identity matrix. In particular, the sampler does not perform well at sampling distributions with thin ridges of probability mass. This is because the transition density of Brownian motion over a finite time interval has a symmetric Gaussian covariance matrix. When π has mass contained in a thin ridge, the rate κ increases very fast away from that ridge. Consequently, a Brownian Motion Restore Sampler will regenerate very frequently, since the process is likely to make a random walk away from the manifold of space containing high probability mass under π.

First consider the Banana distribution. Let $X_1 \sim \mathcal{N}(0, \sigma_1^2)$ and $X_2, X_3, \ldots, X_d \sim \mathcal{N}(0, \sigma_2^2)$, with $\sigma_1, \sigma_2 > 0$. Define $Y_1, Y_2, \ldots, Y_d$ as the transformed variables $Y_1 = X_1, Y_2 = X_2 - bX_1^2 + ab, Y_3 = X_3, \ldots, Y_d = X_d$, for constants $a, b > 0$. The inverse transformations are thus $X_1 = Y_1, X_2 = Y_2 + bY_1^2 - ab, X_3 = Y_3, \ldots X_d = Y_d$. It is easy to show that $|dx/dy| = 1$. Thus the transformed density is

$$\pi(y) = \frac{1}{(2\pi)^{d/2}\sigma_1\sigma_2^{d-1}} \exp\left\{ -\frac{1}{2}\left[\frac{y_1^2}{\sigma_1^2} + \frac{(y_2 + by_1^2 - ab)^2}{\sigma_2^2} + \frac{y_3^2}{\sigma_2^2} + \cdots + \frac{y_d^2}{\sigma_2^2} \right] \right\}.$$

The expectation and variance of $Y_1, \ldots, Y_d$ may be computed analytically. Clearly, $\mathbb{E}[Y_1] = 0, \mathrm{Var}[Y_1] = \sigma_1^2, \mathbb{E}[Y_3] = \mathbb{E}[Y_4] = \cdots = \mathbb{E}[Y_d] = 0$ and $\mathrm{Var}[Y_3] = \mathrm{Var}[Y_4] = \cdots = \mathrm{Var}[Y_d] = \sigma_2^2$. For Y_2 we have $\mathbb{E}[Y_2] = \mathbb{E}[X_2 - bX_1^2 + ab] = b(a - \sigma_1^2)$ and $\mathrm{Var}[Y_2] = \sigma_2^2 + 2b^2\sigma_1^4$.

For the Banana distribution in dimension $d = 2$ with hyperparameters $a = b = \sigma_1 = \sigma_2 = 1$, contour plots of π and $\tilde{\kappa}$ are shown in Figure 4.8. See Appendix B.3 for the gradient and Laplacian of the Banana distribution. Here, μ^+ is defined on a thin manifold of space. The full rate κ is shown for $\mu \equiv t_3(0, I)$ and C chosen so that $\min_{x \in \mathcal{X}} \kappa(x) \approx 0.01$. The regeneration rate is very large in most of the space and hence the Brownian Motion Restore sampler is not very effective, since



(a) Contours of π.

(b) Contours of $\tilde{\kappa}$, on a different scale.

(c) Contours of κ, on a different scale, for $\mu \equiv t_3(0, I)$ and C chosen so that $\min \kappa \approx 0.01$.

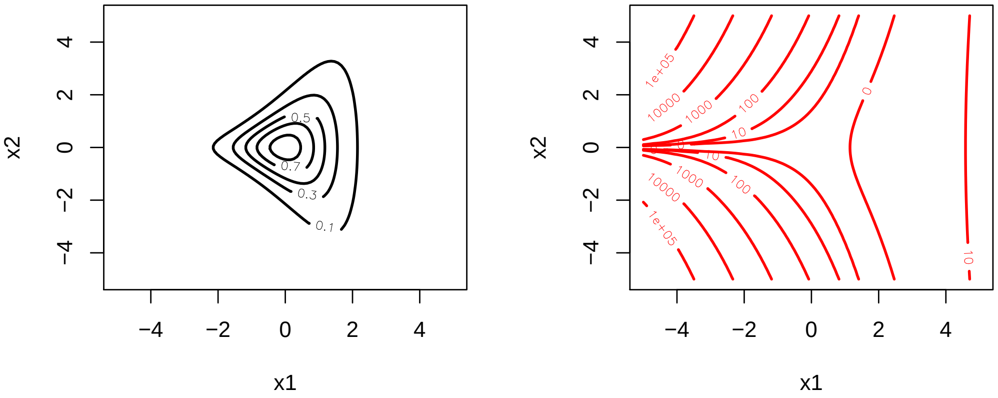
Figure 4.8: Contours of $\pi, \tilde{\kappa}$ and κ for the Banana distribution with $d = 2$ and $a = b = \sigma_1 = \sigma_2 = 1$. Since π has mass on a thin manifold of $\mathcal{X}$, κ increases quickly away from this manifold and hence the Restore process is likely to regenerate very frequently.

the process regenerates very frequently and very little local exploration is done. A sample path is *not* displayed, since the tours would essentially look like dots.

Finally, the Funnel distribution is a product of Gaussians, with the variance of component i for $i = 2, \dots, d$ depending on the first component:

$$\pi(x) = \mathcal{N}(x_1; 0, a^2) \prod_{i=2}^{d} \mathcal{N}(x_i; 0, e^{2bx_1}).$$

Figure 4.9 shows contour plot of π and κ^+ when $a = 1$ and $b = 0.5$. See Appendix B.4 for the gradient and Laplacian of the log-density of this distribution, as well as the corresponding partial regeneration rate. Interestingly, for this distribution, the set $\{x : \tilde{\kappa}(x) < 0\}$ is not bounded.



(a) Contours of π.

(b) Contours of κ^{+}.

Figure 4.9: Contour plots of π and $\tilde{\kappa}$ for the Funnel distribution with $a = 1$ and $b = 0.5$.

4.5 Estimating Normalizing Constants

Section 2.4 briefly covered some existing methods for estimating normalising constants. This section makes a novel contribution in showing that Restore processes can be used to estimate normalising constants. Recall that τ, given by equation (4.1), is the tour length. When μ and π are both normalized, it is known that $C = 1/\mathbb{E}_{\mu}[\tau]$ (Wang et al., 2021, proof of Theorem 16 on π-invariance of diffusion Restore processes), where $\mathbb{E}_{\mu}[\tau]$ denotes expectation with respect to a Restore process with initial value distributed according to μ.

Suppose $\pi(x) = \tilde{\pi}/Z$ and Z is unknown. Then Z is absorbed into C, in that

$$\kappa(x) = \tilde{\kappa}(x) + \tilde{C}\frac{\mu(x)}{\tilde{\pi}(x)}$$

and $\tilde{C} = CZ$. Rearanging, $C = \tilde{C}/Z$ so $Z = \tilde{C}\mathbb{E}_{\mu}[\tau]$. Suppose n tours take simulation time T, then $Z \approx \tilde{C}T/n$.

Chapter 5

Adaptive Simulation of Regeneration-Enriched Brownian Motion

To simulate a Brownian Motion Restore process, a regeneration measure must be chosen so that the corresponding regeneration rate is non-negative everywhere. When μ differs greatly from π, tours of the process frequently start in areas where π has low probability mass and for which the regeneration rate is very large, so regeneration occurs very frequently. This is computationally wasteful, since π and its derivatives must be evaluated in order to determine regeneration events. This chapter considers adapting μ so that the minimal regeneration rate may be used. We call the novel Markov process an *Adaptive Restore process* (McKimm et al., 2022) and the original Restore process the *Standard Restore process.*

The chapter is structured as follows. Section 5.1 explains how, for μ fixed, C may be increased so that κ becomes a valid rate. The section also motivates the need to adapt μ as well as C, via an example. Next, Section 5.2 shows that, if done naively, adaptive strategies may fix the issue of κ taking negative values, but without ensuring κ doesn't become extremely large. The main contribution of this chapter is in Section 5.3, which details the Adaptive Restore algorithm. A discussion concludes the chapter in Section 5.4.

5.1 Regeneration distribution fixed and only its constant adapted

One of the main motivations for adapting the regeneration measure is to guarantee that $\kappa(x) \geq 0, \forall x \in \mathcal{X}$. This section explains how, given a fixed regeneration distribution μ, the regeneration constant C may be adapted upwards to obtain a valid regeneration rate. For many target distributions, it will be possible to find a valid regeneration rate simply by adapting C upwards. A sufficient condition is that the support of μ^+ is compact and contained by the support of μ. Essentially, whenever a state with negative rate is found, increase C so that the rate evaluated at that same state becomes positive. This idea is now formulated more precisely.

Let $\{X_t^{(\mu,\kappa)}\}_{t\geq 0}$ denote a Brownian Motion Restore process with regeneration distribution μ and regeneration rate κ. To express the regeneration constant and distribution used, we'll use notation:

$$\kappa_{C\mu}(x) := \tilde{\kappa}(x) + C\frac{\mu(x)}{\pi(x)}.$$

Suppose that the regeneration constant after i adaptions is C_i, with C_0 the initial constant. After i adaptions, a process $\{X_t^{(\mu,\kappa_{C_i\mu}\vee 0)}\}_{t\geq 0}$ is generated, denoted $\{X_t^{(i)}\}_{t\geq 0}$ for short. Denote $\kappa_{C_i\mu} \vee 0$ by κ_i for short; this rate is used to guarantee non-negativity – the truncation becomes redundant once C is large enough. Let ι be the first time that κ_i is evaluated at a state x and that $\kappa_{C_i\mu}(x) < 0$.

Potential regeneration times $T_0^{(\mathcal{K})}, T_1^{(\mathcal{K})}, \ldots$ are the arrival times of a Poisson process with rate $\mathcal{K}$. These times are thinned, which involves evaluating κ at the current state, to determine whether regeneration occurs. Now,

$$\iota := \inf\left\{t : \kappa_i(X_t) < 0 \text{ and } t \in \left\{T_0^{(\mathcal{K})}, T_1^{(\mathcal{K})}, \ldots\right\}\right\}.$$

Using the convention that $\inf\emptyset = \infty$, a large enough C will result in $\iota = \infty$. When $\iota < \infty$, the process must be restarted with a larger value of C. Suppose that $X_\iota = x$. Then, for $\underline{\kappa} > 0$ a small user-chosen constant, set

$$C_{i+1} = \frac{\pi(x)}{\mu(x)}\left[\underline{\kappa} - \tilde{\kappa}(x)\right], \tag{5.1}$$

discard $\{X_t^{(i)}\}_{t\geq 0}$ (and any output recorded) then begin simulating $\{X_t^{(i+1)}\}_{t\geq 0}$. Since equation (5.1) can be arranged as:

$$\underline{\kappa} = \tilde{\kappa}(x) + C_{i+1}\frac{\mu(x)}{\pi(x)},$$

this guarantees $\kappa_{i+1}(x) > 0$. Larger values of $\underline{\kappa}$ are 'safer', in that they result in larger values of C_{i+1}. The method is described by Algorithm 8, which is a simple modification of Algorithm 7.

Algorithm 8: The Brownian Motion Restore Sampler with Adaption of the Regeneration Constant

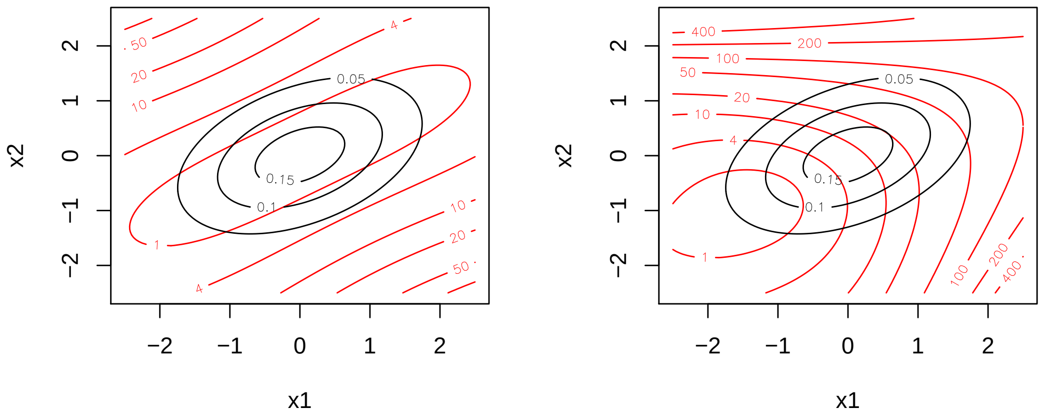
```
$X \sim \mu, t \leftarrow 0, i \leftarrow 0$
while $i < n$ do
    $\tau^{(\mathcal{K})} \sim \text{Exp}(\mathcal{K}), \tau^{(\Lambda)} \sim \text{Exp}(\Lambda)$
    if $\tau^{(\Lambda)} < \tau^{(\mathcal{K})}$ then
        $t \leftarrow t + \tau^{(\Lambda)}, X \sim \mathcal{N}(X, \tau^{(\Lambda)})$. Record $X, t, i$
    else
        $t \leftarrow t + \tau^{(\mathcal{K})}, X \sim \mathcal{N}(X, \tau^{(\mathcal{K})}), u \sim \mathcal{U}[0, 1]$
        if $\kappa(X) < 0$ then
            Discard output, $X \sim \mu, t \leftarrow 0, i \leftarrow 0$
            $C \leftarrow \pi(X)[\underline{\kappa} - \tilde{\kappa}(X)]/\mu(X)$
        else
            if $u < \kappa(X)/\mathcal{K}$ then
                $X \sim \mu, i \leftarrow i + 1$
            end
        end
    end
end
```

We now provide a sketch of a proof for why the above procedure works. Let C^* be such that $\min_{x \in \mathcal{X}} \kappa_{C^*\mu}(x) = 0$ and let $x^* := \arg\min \kappa_{C^*\mu}(x)$, so that $\kappa_{C^*\mu}(x^*) = 0$. Define the set $A \subset \mathcal{X}$ as

$$A := \{x \in \mathcal{X} : \kappa_{C^*\mu}(x) < \underline{\kappa}\}.$$

Note that x^* is in A. Suppose that the current value of the regeneration constant is insufficiently large. That is, suppose that $C_i < C^*$, so that $\kappa_{C_i\mu}(x^*) < 0$. Furthermore, suppose that the rate is not just negative at x^*, but also in a region around x^*. More precisely, assume that that there exists a neighbourhood N of x^* such that $\forall x \in N$, $\kappa_{C_i\mu}(x) < 0$. There is some chance that the process $\{X_t\}_{t \geq 0}$ will enter the non-singular set $N \cap A$ and remain in the set for a period of time greater than zero. Therefore, eventually the rate $\kappa_{C_i\mu}$ will be evaluated for a state $\hat{x}$ with $\hat{x} \in N \cap A$. Suppose that $\kappa_{C^*\mu}(\hat{x}) = \hat{\kappa}$. Since $\hat{x}$ is in A, we have $\hat{\kappa} < \underline{\kappa}$. The adaptive procedure results in the next value of the regeneration constant, C_{i+1}, satisfying $\kappa_{C_{i+1}\mu}(\hat{x}) = \underline{\kappa}$. Therefore $\kappa_{C^*\mu}(\hat{x}) < \kappa_{C_{i+1}\mu}(\hat{x})$ and $C_{i+1} > C^*$.

Note that if $\underline{\kappa} = 0$, the adaptive procedure is no longer guaranteed to find a sufficiently large regeneration constant. Suppose that the rate is evaluated at



(a) $\mu_a = \mathcal{N}_2(0, I_2), C = 7.86$.

(b) $\mu_b = \mathcal{N}_2(1, I_2), C = 147.15$.

Figure 5.1: Adapting the regeneration constant of Brownian Motion Restore. Contours of Gaussian target distribution π and $\kappa_{C\mu}$ in black and red respectively, for different choices of C and μ.

a state $\hat{x} \neq x^*$ at which $\kappa_{C_i\mu}(\hat{x}) < 0$. Then the adaptive procedure with $\underline{\kappa} = 0$ results in $\kappa_{C_{i+1}\mu}(\hat{x}) = 0$. Since $\kappa_{C^*\mu}(\hat{x}) > 0$, it follows that $C_{i+1} < C^*$, so the rate $\kappa_{C_{i+1}\mu}$ is not valid. If the rate is evaluated at x^*, then the adaptive procedure results in $C_{i+1} = C^*$, so that $\kappa_{C_{i+1}\mu}$ is a valid rate. However, since x^* is a point, the rate will never be evaluated exactly at this state and so the adaptive procedure will never find a non-negative rate.

5.1.1 When Adapting the Regeneration Constant only is Insufficient

Here, two illustrations are presented of why the choice of regeneration distribution is so important. First suppose that the target distribution has density $\pi(x) = \exp\{-x^T\Sigma^{-1}x/2\}$, with

$$\Sigma = \begin{pmatrix} 1.2 & 0.4 \\ 0.4 & 0.8 \end{pmatrix}. \tag{5.2}$$

Consider sampling π by simulating a Brownian Motion Restore process with two choices of regeneration distribution, $\mu_a = \mathcal{N}_2(0, I_2)$ and $\mu_b = \mathcal{N}_2(1, I_2)$. Distribution μ_a naturally seems to be a better choice than μ_b, since it has the same mean as π. Algorithm 8 results in C equal to 7.86 and 147.15 (2.d.p) respectively. Figure 5.1 shows contours of π in black, as well as those of $\kappa_{C=7.86,\mu_a}$ and $\kappa_{C=147.15,\mu_b}$ in red.

The example demonstrates that a bad choice of μ, in this case μ_b, causes κ to be very large in some regions. As a result, to ensure the set $\{x \in \mathcal{X} : \kappa(x) > \mathcal{K}\}$

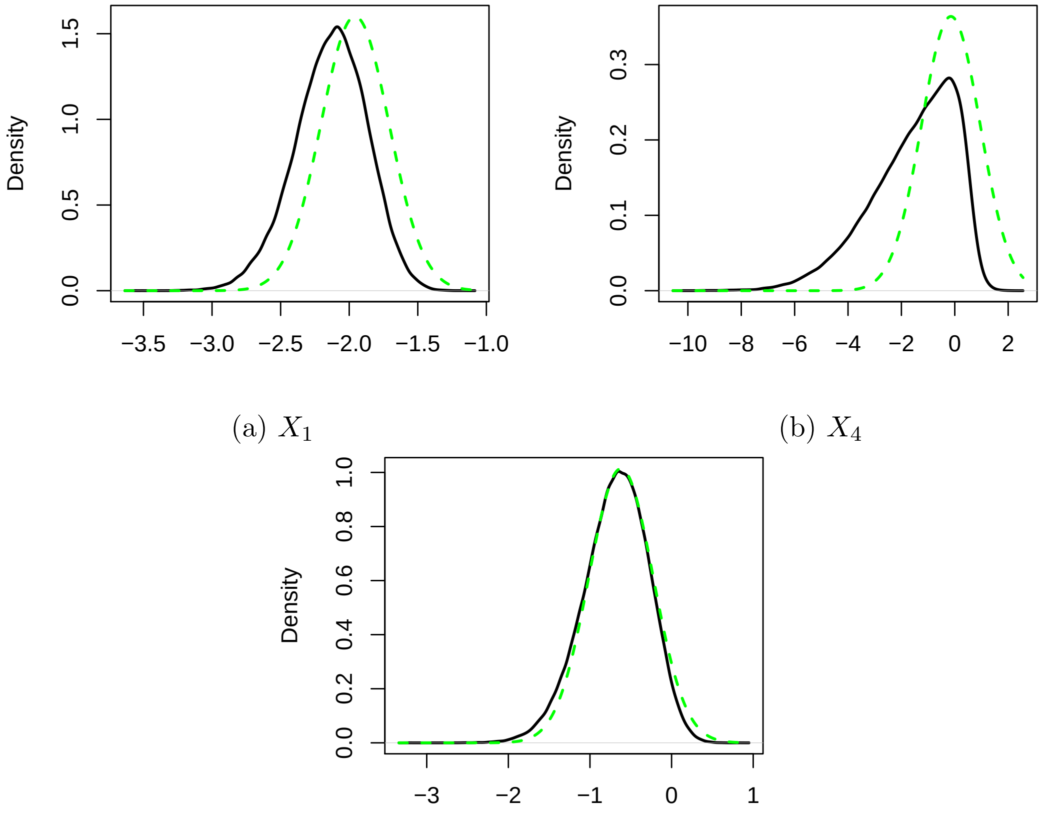
has small mass, it is necessary to set $\mathcal{K}$ to be very high. For this example, based on plots of contours of κ, a conservative estimate for $\mathcal{K}$ (so that the introduction of error is negligible) might be 50 and 400, corresponding to μ_a and μ_b respectively. Since $\mathcal{K}$ is the main driver of computational cost, this means simulating a Restore process using μ_b is roughly eight times as expensive as using μ_a.

Though simulation using a bounding rate of $\mathcal{K} = 400$ will be very slow, it is not too impractical. We now present a target posterior distribution which results in extremely large values of the regeneration rate and is hence infeasible to simulate.

Consider a Logistic Regression model (see Appendix B.5) of hospital patients, where the task is to classify whether a patient is a Liver patient (someone being treated for liver problems) or not. The model has 10 predictors, so including the intercept term, the target posterior distribution has dimension $d = 11$. The size of the data is $n = 579$. There was one binary predictor, which was scaled to have range 1 and mean zero. The other predictors were scaled to have means zero and standard deviations 0.5. A Gaussian product prior was used, with standard deviation 20 for each variable.

The posterior distribution was transformed based on its Laplace approximation (see subsections 2.5.1 and 2.5.2). The aim of transforming π using its Laplace approximation is for the transformed distribution to have roughly zero mean and identity covariance matrix. Since this pre-transformation is used, it may seem reasonable to use as regeneration distribution a Gaussian with zero-mean and identity covariance matrix. If the Laplace approximation is fairly accurate, the transformed target distribution will be close to this choice of regeneration distribution. We will see that for this example the posterior is highly non-Gaussian, the transformed posterior is thus highly non-Gaussian and hence the seemingly reasonable choice of $\mu \equiv \mathcal{N}(0, I_d)$ results in extremely large values of κ.

First, to illustrate that the posterior distribution is non-Gaussian, we use the Random Walk Metropolis (RWM) algorithm (see subsection 2.3.2), with a large thinning period (thin $=$ 100) to obtain samples from the target distribution before it is transformed. We display 3 of the 11 marginals. For 9 out of the 11 dimensions, a plot of the kernel density estimate of the posterior marginal against the Laplace approximation for that marginal shows very little difference between the mean and mode. However, for dimensions 1 (shown in Figure 5.2a) and 5 (not shown in Figure 5.2), the mean and mode are visibly different. The most remarkable marginal is the forth, shown in Figure 5.2b, which is highly skewed. For many marginals, such as the 6th plotted in Figure 5.2c, though KDE plots



(a) X_1

(b) X_4

(c) X_6

Figure 5.2: Kernel density estimates of marginals 1, 4 and 6 of the posterior distribution of a Logistic Regression Model of Liver Patients (pre-transformation) in black. Marginals of the Laplace approximation in dashed green.

make it look like the mean and variance has been accurately approximated, close inspection reveals a small skew.

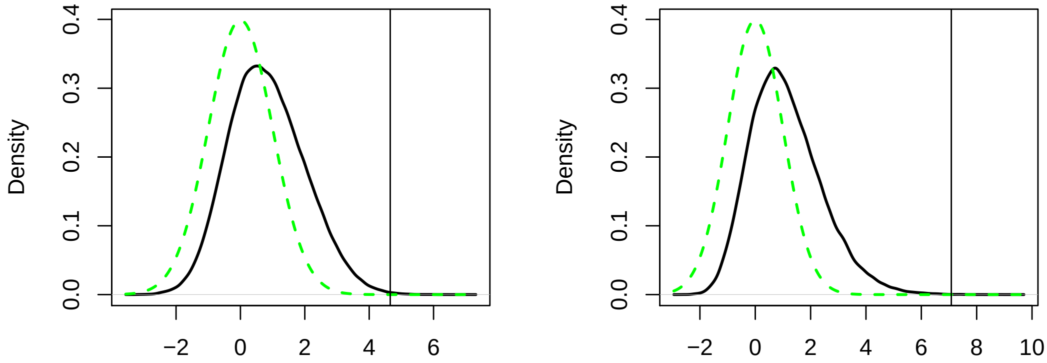
Sampling the *transformed* posterior using the RWM algorithm, a thinning interval of 90 is large enough that the autocorrelation of the chain is less than 0.1 for all variables. The marginals that are least Gaussian in shape, 2 and 5, are plotted in Figure 5.3. The vertical lines are explained in full later, but show a state x_b for which C must be especially large to ensure $\kappa_{C\mu}(x_b) \geq 0$.

For the samples $\{x_1, \dots, x_n\}$ generated using the RWM algorithm, C such that $\kappa(x_i) \geq 0, \forall i \in \{1, \dots, n\}$ was computed as

$$C = \max_{x_i \in \{x_1,\dots,x_n\}} -\tilde{\kappa}(x_i)\pi(x_i)/\mu(x_i).$$

Estimates of the quantile function for $\tilde{\kappa}$ and κ (using this value of C) are shown in Figure 5.4b.

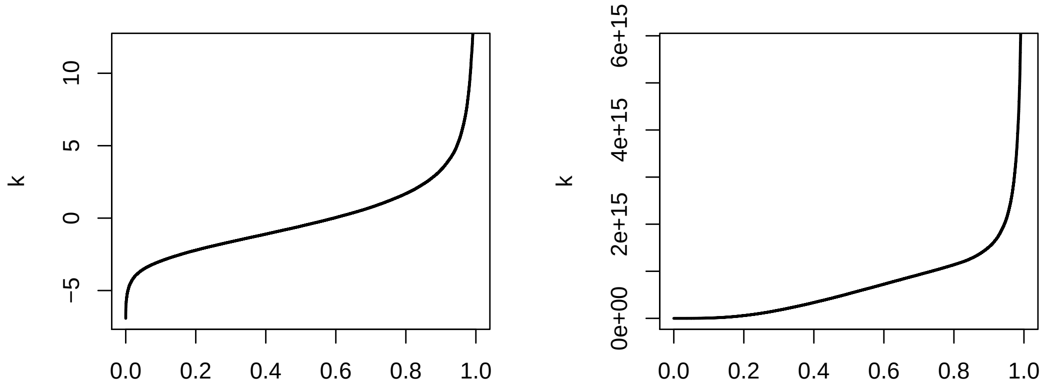
Due to the high discrepancy between μ and π, the regeneration rate is extremely large in most of the state space. Let x_b be the sample so that for the



(a) X_2

(b) X_5

Figure 5.3: Kernel density estimates of marginals 2 and 5 of the transformed posterior distribution of a Logistic Regression Model of Liver Patients in black. Density of the standard normal regeneration distribution in dashed green. The vertical lines show the relevant coordinate of x_b, the state given by equation (5.3), which forces C to be very large. At this state, $\tilde{\kappa}$ is negative, but the ratio μ/π is very small, so C must be very large for κ to be non-negative.



(a) $\mathbb{P}[\tilde{\kappa}(X) < k]$

(b) $\mathbb{P}[\kappa(X) < k]$

Figure 5.4: Estimates of the quantile functions of the partial and full regeneration rates for a Logistic Regression Model of Liver Patients.

value of C computed earlier, $\tilde{\kappa}(x_b) + C\mu(x_b)/\pi(x_b) = 0$. That is,

$$x_b := \underset{x \in \{x_1, \ldots, x_n\}}{\arg\max} \{-\tilde{\kappa}(x)\pi(x)/\mu(x)\}. \tag{5.3}$$

At x_b, the geometry of π and μ are such that C must be very large in order to ensure $\kappa(x_b) \geq 0$. For this example, x_b is:

$$x_b = (0.44, 4.65, -3.30, -3.10, 7.08, 1.38, -0.53, 0.77, -0.25, 0.41, -0.22)^T.$$

The second and forth co-ordinate of x_b are shown in relation to the marginals of π and μ by vertical lines in Figure 5.3. These marginals help to illustrate that $\mu(x_b)/\pi(x_b)$ is very small. Since $\tilde{\kappa}(x_b)$ is negative, C must be very large to compensate for this tiny value. Although the ratio μ/π will be even smaller for some states even further into the tails, the partial regeneration rate is negative only in a compact region near the mode of π, hence outside of this region, there is no need for the term $C\mu/\pi$ to compensate for $\tilde{\kappa}$ being negative.

This example is particularly pathological for Brownian Motion Restore sampling. To better understand which distributions are and are not suitable for Brownian Motion Restore sampling, we considered the posterior distribution of a Logistic Regression model of the following five data-sets, obtained from the UCI Machine Learning Repository (Dua and Graff, 2017):

Data	n	d	Description
Diabetes	768	9	Response variable: presence or absence of diabetes in a population of Pima Indians. Predictors: diagnostic measurements. (Smith et al., 1988).
Heart	270	14	Response: presence or absence of heart disease in patients in a Cleveland hospital. Predictors: health measurements. (Detrano et al., 1989).
Breast	683	10	Data obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. Response: whether the breast mass is benign or malignant. Predictors are features of an image of breast mass. (Mangasarian and Wolberg, 1990).
German	1000	25	Response: whether an applicant was granted a loan or not. Predictors: measurements relating to credit score. Provided by Professor Dr. Hans Hofmann; categorical variables converted to numerical attributed by Strathclyde University.
Liver	579	11	Response: whether a patient is being treated for liver problems or not. Predictors: diagnostic measurements. Data collected from the north east of Andhra Pradesh, India (Ramana et al., 2012).

In the table below, Monte Carlo estimates of $\mathcal{K}^+$ s.t. $\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] \approx 0.01$ and $\mathcal{K}$ s.t. $\mathbb{P}[\kappa(X) > \mathcal{K}] \approx 0.01$ are given for the different data sets.

Data	$\mathcal{K}^+$ s.t. $\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] \approx 0.01$	$\mathcal{K}$ s.t. $\mathbb{P}[\kappa(X) > \mathcal{K}] \approx 0.01$
Diabetes	6.37	12.34
Heart	7.58	78
Breast	9.75	2356
German	9.28	2.07×10^5
Liver	12.08	5.8×10^{15}

The following table presents Monte Carlo estimates of $\mathbb{E}[\kappa(X)]$. To provide further evidence that x_b is often found in the tails of μ, the last column gives values of $||x_b||^2$.

Data	$\mathbb{E}[\kappa(X)]$	$\lvert\lvert x_b \rvert\rvert^2$
Diabetes	4	0.58
Heart	16	22.41
Breast	368	32.73
German	63707	56.54
Liver	8.6×10^{14}	95.58

Since x_b resides in the tails of a Gaussian regeneration distribution μ, why not use as regeneration distribution a heavier-tailed distribution, such as a multivariate t-distribution? One could do this, but at a price. As well as regenerating more often into regions for which it is beneficial for μ to have more mass because π has a heavy tail there, the process will also regenerate more often into tail regions in which π has a light tail.

5.2 Naive Adaptive Restore

Subsection 5.1.1 shows that adapting C only and keeping μ fixed can be insufficient. This motivates the need for an algorithm that adapts μ – the question is how? Section 5.3 covers Adaptive Restore, the central contribution of this chapter. Before introducing Adaptive Restore fully, this section covers two "naive" algorithms for adapting μ. The first method, *Gaussian Mixture Brownian Motion Restore* (GM-BMR) of subsection 5.2.1, adds lumps of mass to measure $C\mu$ in regions where negative regeneration rates are detected, thus ensuring that the regeneration rate is eventually non-negative everywhere. The second method (subsection 5.2.2) provides a technique for sampling μ^+, which may then be approximated, possibly leading to more efficient sampling.

GM-BMR has been suggested by several colleagues in casual conversations. We argue that the method is ineffective – it does not solve the issue of κ being very large. Nonetheless, the technique is presented because it illustrates the strength of Adaptive Restore: the use of κ^+ throughout simulation. Likewise, the method of subsection 5.2.2 does not guard against κ being very large, but helps in understanding Adaptive Restore.

5.2.1 Gaussian Mixture Regeneration Distribution

A central aim for adapting $C\mu$ is to ensure $\kappa_{C\mu}(x) \geq 0, \forall x \in \mathcal{X}$. One method is to represent μ as a mixture of Gaussians. When $x \in \mathcal{X}$ such that $\kappa_{C\mu}(x) < 0$ is identified, the idea is to add a lump of mass to $C\mu$ in such a way that after adaption $\kappa_{C\mu}(x) > 0$. A sequence of regeneration measures $C_0\mu_0, C_1\mu_1, \ldots$ is used. Care is take to ensure mass in only ever *added* to the regeneration measure, so that:

$$C_{i+1}\mu_{i+1}(x) \geq C_i\mu_i(x), \forall x \in \mathcal{X}, i = 0, 1, \ldots.$$

Whenever a state $x \in \mathcal{X}$ is found for which $\kappa_{C\mu}(x) < 0$, a component is added to μ with mean x and C adjusted so that the weight of all pre-existing components remains the same. That is, for $\sigma > 0$ some constant, let μ_i be:

$$\mu_i \equiv \frac{1}{i+1} \sum_{l=0}^{i} \mathcal{N}(\nu_l, \sigma^2 I_d).$$

Let $\{X_t^{(i)}\}_{t \geq 0}$ again denote the process simulated after i adaptions, which is now $\{X_t^{(\mu_i, \kappa_{C_i\mu_i} \vee 0)}\}_{t \geq 0}$. Suppose $\{X_t^{(i)}\}_{t \geq 0}$ is evaluated at time $t \geq 0$, that $X_t^{(i)} = x^*$ and $\kappa_{C_i\mu_i}(x^*) < 0$. Letting $\underline{\kappa} > 0$ be some small constant (0.01 for example) and $\nu_{i+1} := x^*$, define C_{i+1} and μ_{i+1} as:

$$\begin{aligned}
\mu_{i+1} &\equiv \frac{1}{i+2} \sum_{l=0}^{i+1} \mathcal{N}(\nu_l, \sigma^2 I_d), \\
C_{i+1} &= \left\{ \frac{i+2}{i+1} C_i \right\} \vee [\underline{\kappa} - \tilde{\kappa}(x^*)] \frac{\pi(x^*)}{\mu_{i+1}(x^*)}.
\end{aligned}$$

For all $x \in \mathcal{X}$, the regeneration measure $C_{i+1}\mu_{i+1}$ satisfies:

$$\begin{aligned}
C_{i+1}\mu_{i+1}(x) &= \left(\left\{ \frac{i+2}{i+1} C_i \right\} \vee [\underline{\kappa} - \tilde{\kappa}(x^*)] \frac{\pi(x^*)}{\mu_{i+1}(x^*)} \right) \frac{1}{i+2} \sum_{l=0}^{i+1} \mathcal{N}(\nu_l, \sigma^2 I_d), \\
&\geq \frac{i+2}{i+1} C_i \frac{1}{i+2} \sum_{l=0}^{i+1} \mathcal{N}(x; \nu_l, \sigma^2 I_d), \\
&= \frac{1}{i+1} C_i \left[\sum_{l=0}^{i} \mathcal{N}(x; \nu_l, \sigma^2 I_d) + \mathcal{N}(x; \nu_{i+1}, \sigma^2 I_d) \right], \\
&= C_i \left[\mu_i(x) + \frac{1}{i+1} \mathcal{N}(x; \nu_{i+1}, \sigma^2 I_d) \right], \\
&> C_i\mu_i(x).
\end{aligned}$$

Thus each adaption causes the regeneration rate to increase everywhere. Furthermore, the regeneration rate becomes positive at x^* since:

$$\begin{aligned}
\tilde{\kappa}(x^*) + C_{i+1} \frac{\mu_{i+1}(x^*)}{\pi(x^*)} &= \tilde{\kappa}(x^*) + \left(\left\{ \frac{i+2}{i+1} C_i \right\} \vee [\underline{\kappa} - \tilde{\kappa}(x^*)] \frac{\pi(x^*)}{\mu_{i+1}(x^*)} \right) \frac{\mu_{i+1}(x^*)}{\pi(x^*)}, \\
&\geq \tilde{\kappa}(x^*) + [\underline{\kappa} - \tilde{\kappa}(x^*)] \frac{\pi(x^*)}{\mu_{i+1}(x^*)} \frac{\mu_{i+1}(x^*)}{\pi(x^*)}, \\
&\geq \underline{\kappa}.
\end{aligned}$$

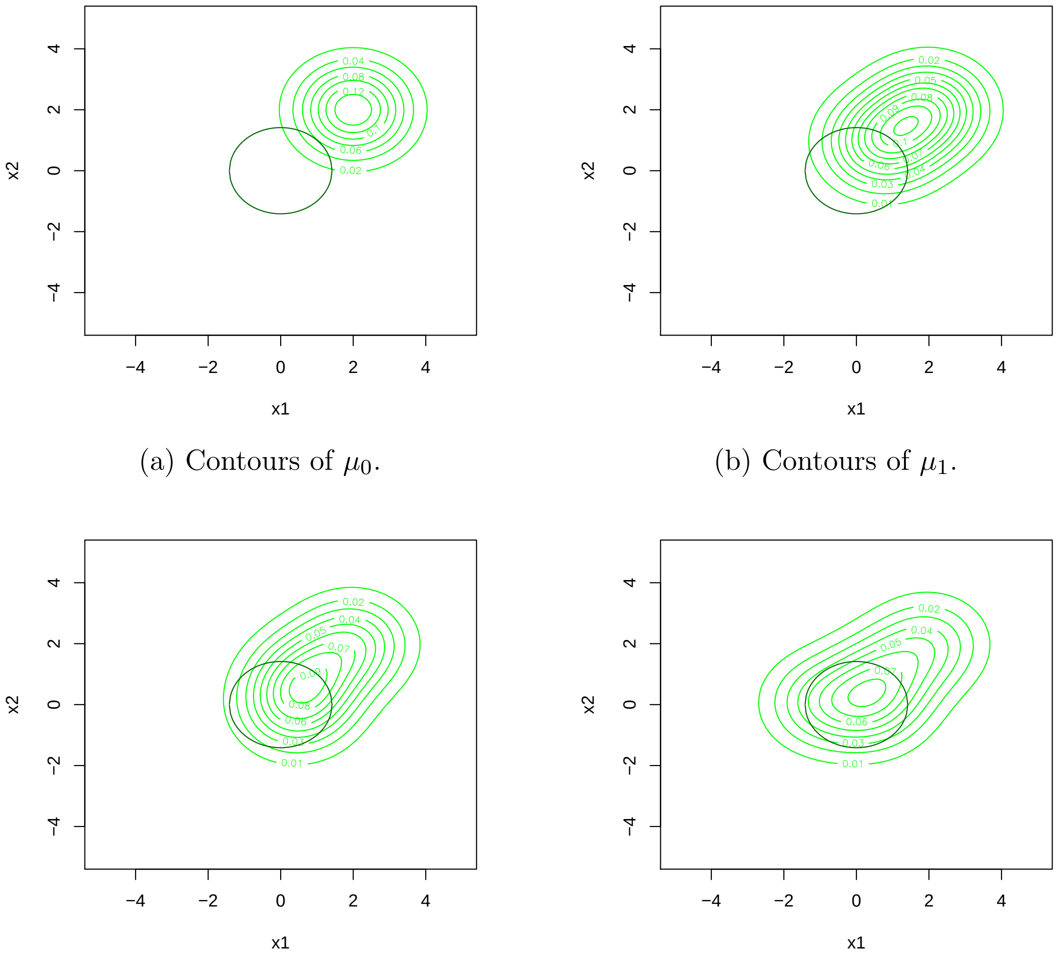
Algorithm 9 shows how to simulate n tours of the GM-BMR process.

Algorithm 9: The Gaussian Mixture Brownian Motion Restore Sampler

```
$i \leftarrow 0, j \leftarrow 0, \mu \leftarrow \mathcal{N}(\nu_0, \sigma^2 I_d), X \sim \mu, t \leftarrow 0, N \leftarrow \{\nu_0\}$
while $i < n$ do
    $\tau^{(\Lambda)} \sim \mathrm{Exp}(\Lambda), \tau^{(\mathcal{K})} \sim \mathrm{Exp}(\mathcal{K})$
    if $\tau^{(\Lambda)} < \tau^{(\mathcal{K})}$ then
        $t \leftarrow t + \tau^{(\Lambda)}, X \sim \mathcal{N}(X, \tau^{(\Lambda)} I_d)$. Record $X, t, i$.
    else
        $t \leftarrow t + \tau^{(\mathcal{K})}, X \sim \mathcal{N}(X, \tau^{(\mathcal{K})} I_d)$
        if $\kappa(X) < 0$ then
            $N \leftarrow N \cup \{X\}$
            $\mu \leftarrow (j+2)^{-1} \sum_{\nu \in N} \mathcal{N}(\nu, \sigma^2 I_d)$
            $C \leftarrow (j+2)C/(j+1) \vee [\underline{\kappa} - \tilde{\kappa}(X)]\pi(X)/\mu(X)$
            $j \leftarrow j+1$. Discard output. $X \sim \mu, t \leftarrow 0, i \leftarrow 0$
        else
            $u \leftarrow \mathcal{U}(0,1)$
            if $u < \kappa(X)/\mathcal{K}$ then
                $X \leftarrow \mu, i \leftarrow i+1$
            end
        end
    end
end
```

As an example, we consider sampling from a two-dimensional distribution, $\pi = \mathcal{N}(0, I)$, taking $\sigma = 1, \underline{\kappa} = 0.01, C_0 = 0$ and $\mu_0 = \mathcal{N}((2,2)^T, I)$. For the sample path simulated, it took three adaptions to achieve non-negativity of the rate. For this simple example, it was possible to confirm that the adapted regeneration rate was non-negative everywhere, by insepcting contour plots of the final expression for κ. Figure 5.5 illustrates the sequence of regeneration distributions used. The sequence of means in the Gaussian mixture was $\nu_0 = (2,2)^T, \nu_1 = (0.73, 0.93)^T, \nu_2 = (0.40, -0.10)^T, \nu_3 = (-1.01, 0.06)^T$ (2.d.p). The sequence of C's was $C_0 = 0, C_1, = 1.52, C_2 = 10.10, C_3 = 13.47$ (2.d.p). Figure 5.5 shows how μ 'moves' towards the support of μ^+.

Though the algorithm works well for this toy example, there are several issues. Firstly, although a measure $C\mu$ is eventually found such that $\kappa_{C\mu}(x) \geq 0, \forall x \in \mathcal{X}$, the method does not solve the issue of requiring $\mathcal{K}$ to be extremely large. Suppose $\exists x_1, x_2 \in \mathcal{X}$ such that $\kappa(x_1) < 0$ and $\kappa(x_2)$ is extremely large. Adding mass concentrated at x_1 to $C\mu$ will also add mass (albeit possibly a small amount) to $C\mu$ at x_2, which will make $\kappa(x_2)$ even larger. Second, choosing a small value for



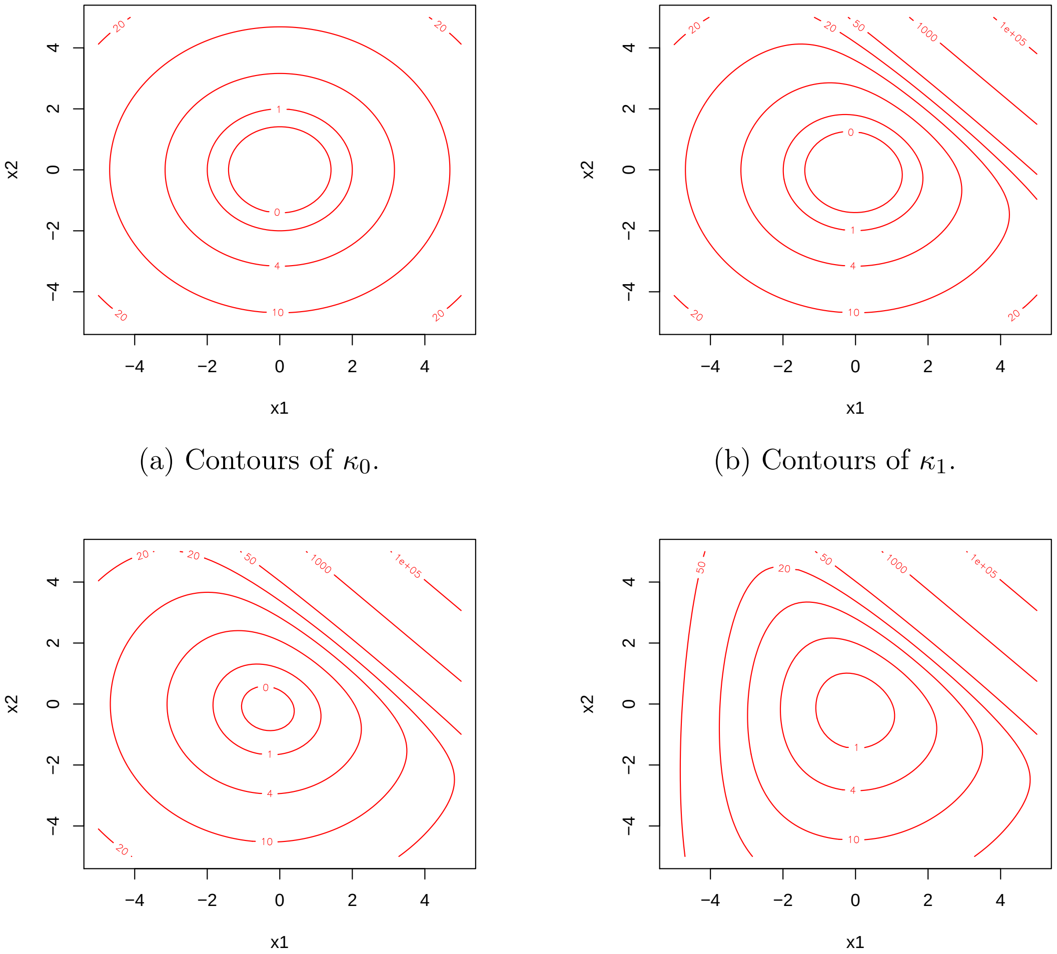
(a) Contours of μ_0.

(b) Contours of μ_1.

(c) Contours of μ_2.

(d) Contours of μ_3.

Figure 5.5: Regeneration distribution a mixture of Gaussians with an increasing number of components. $\pi \equiv \mathcal{N}(0, I)$. The boundary of the support of μ^+, $x_1^2 + x_2^2 < 2$, is shown by the dark green circle. Contours of $\mu_i, i = 0, 1, 2, 3$ shown in green.



(a) Contours of κ_0.

(b) Contours of κ_1.

(c) Contours of κ_2.

(d) Contours of κ_3.

Figure 5.6: Regeneration rate adapted by adding components to the Gaussian mixture regeneration distribution. Contours of κ_i for $i = 0, 1, 2, 3$ are shown in red. $\pi \equiv \mathcal{N}(0, I)$.

σ could lead to μ consisting of a very large number of components. In this case, the computational cost of evaluating μ will become very expensive.

5.2.2 Minimal Regeneration Distribution Approximation

It is necessary to find $C\mu$ such that $\kappa_{C\mu}(x) \geq 0, \forall x \in \mathcal{X}$. Yet at the same time, for $\epsilon > 0$ small, we would like to ensure that $\mathbb{P}[\kappa_{C\mu}(X) > \mathcal{K}] < \epsilon$ for $\mathcal{K}$ not too large. Using the minimal regeneration constant, distribution and rate (C^+, μ^+ and κ^+) would achieve these aims. However, since μ^+ is related to π, it is not easy to sample. The key to Adaptive Restore, as will be described in Section 5.3, is a novel method for simulating a Restore process under κ^+, whilst concurrently sampling distributions π_t and μ_t that simultaneously converge to π, μ^+. In building up to this, we now consider how one might use an approximation of μ^+.

Let $\{X_t\}_{t\geq 0}$ be a Restore process with fixed regeneration measure $C\mu$ and regeneration rate $\kappa_{C\mu}$. Suppose $C\mu$ is such that $\kappa_{C\mu}$ is non-negative everywhere, but is nonetheless expensive to simulate because μ is a poor choice, compensated for by C being large, so that $\mathbb{E}[\kappa_{C\mu}(X)]$ is very large. Let $T_1^{(-)}, T_2^{(-)}, \ldots$ be the arrival times of a Poisson process with rate

$$\kappa^-(x) := [0 \vee -\tilde{\kappa}(x)]. \tag{5.4}$$

Suppose that the process is simulated for some time T. Then, although we do not prove it, we conjecture that

$$\mathbb{E}_{\mu^+}[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

for $0 < T_1^{(-)} < \cdots < T_n^{(-)} < T$ and $x_i = X_{T_i^{(-)}}$ for $i = 1, \ldots, n$. Recall that when the state of the process is recorded as output at the rate of a constant Poisson process, the mean of a function of these states is $\mathbb{E}_\pi[f(X)]$. The intuition behind the conjecture above is that when the rate of the output Poisson process is instead κ^-, the density of the invariant distribution in the expectation changes to $\kappa^-(x)\pi(x) = \mu^+(x)$.

One could redefine μ based on these states, then redefine C so that the new rate $\kappa_{C\mu}$ is again non-negative everywhere. For example, the simplest scheme might redefine μ by fitting a Gaussian to the samples in set

$$\{X_t : t = T_1^{(-)}, T_2^{(-)}, \ldots, T_n^{(-)}\}.$$

The new rate may be a lot smaller on average, since μ is now an approximation of μ^+.

The algorithm described above has a serious issue. To obtain samples from μ^+, for the initial regeneration distribution μ, one must already have found a constant C such that $\kappa_{C\mu}$ is a valid rate. As has been stressed, such a rate may be impractically large. A great strength of the Adaptive Restore algorithm, presented next, is that it uses rate κ^+ throughout, which makes simulation feasible.

5.3 Adaptive Restore

An Adaptive Restore process (McKimm et al., 2022) is defined by enriching some underlying continuous-time Markov process with regenerations at rate κ^+ from, at time t, a distribution μ_t. Initially, the regeneration distribution is some fixed distribution μ_0. The regeneration distribution is updated by the addition of point masses at certain times. Let π_t be the stationary distribution of the Restore process with fixed regeneration distribution μ_t. We have simultaneous convergence of (μ_t, π_t) to (μ^+, π). Let $N(t)$ be the number of events from a Poisson process with rate κ^-, as given by equation (5.4), that arrive before time t. When $N(t) = 0$ let $\mu_t \equiv \mu_0$ for μ_0 some fixed initial distribution. Let $X_{T_i^{(-)}} = x_i$ for $i = 1, 2, \ldots$. Then for $a > 0$ some constant, for $N(t) > 0$ the density of μ_t is given by

$$\mu_t(x) = \frac{t}{a+t}\frac{1}{N(t)}\sum_{i=1}^{N(t)} \delta_{x_i}(x) + \frac{a}{a+t}\mu_0(x). \tag{5.5}$$

The rate κ^- Poisson process is simulated using Poisson thinning, so it is assumed that there exists a constant

$$K^- := \sup_{x\in\mathcal{X}} \kappa^-(x),$$

such that $K^- > 0$. The distribution μ_t is therefore a mixture of a fixed distribution μ_0 and a discrete measure $N(t)^{-1}\sum_{i=1}^{N(t)} \delta_{X_{x_i}}(x)$. The constant a is called the *discrete measure dominance time*, since it is the time at which regeneration is more likely to be from the discrete measure of the mixture distribution.

Section 4 of McKimm et al. (2022) shows that the measure μ_t in (5.5) converges weakly almost surely to μ^+. The validity of the Adaptive Restore algorithm follows, since the invariant distribution of a Restore process with fixed regeneration distribution μ^+ and regeneration rate κ^+ is π. Proving that μ_t converges weakly almost surely to μ^+ is highly non-trivial; the proof builds on previous work by Benaim et al. (2018) and Wang et al. (2020). This thesis does not cover the theory of Adaptive Restore because the focus of this thesis is on methodological and computational aspects of Restore algorithms.

A key assumption is that μ^+ has compact support. There are distributions, such as the Funnel or Banana distributions, shown in subsection 4.4.1, for which μ^+ does not have compact support. However, we have found that μ^+ has compact support for most practical examples, such as those presented in subsection 5.3.3.

Algorithm 10 describes the method. Three Poisson processes, one homogenous and two inhomogeneous, are simulated in parallel. Here, E denotes the set of point masses that make up the discrete component of μ_t. The process is generated for a fixed number of tours, though another condition such as the number of samples or simulation time could equally be used.

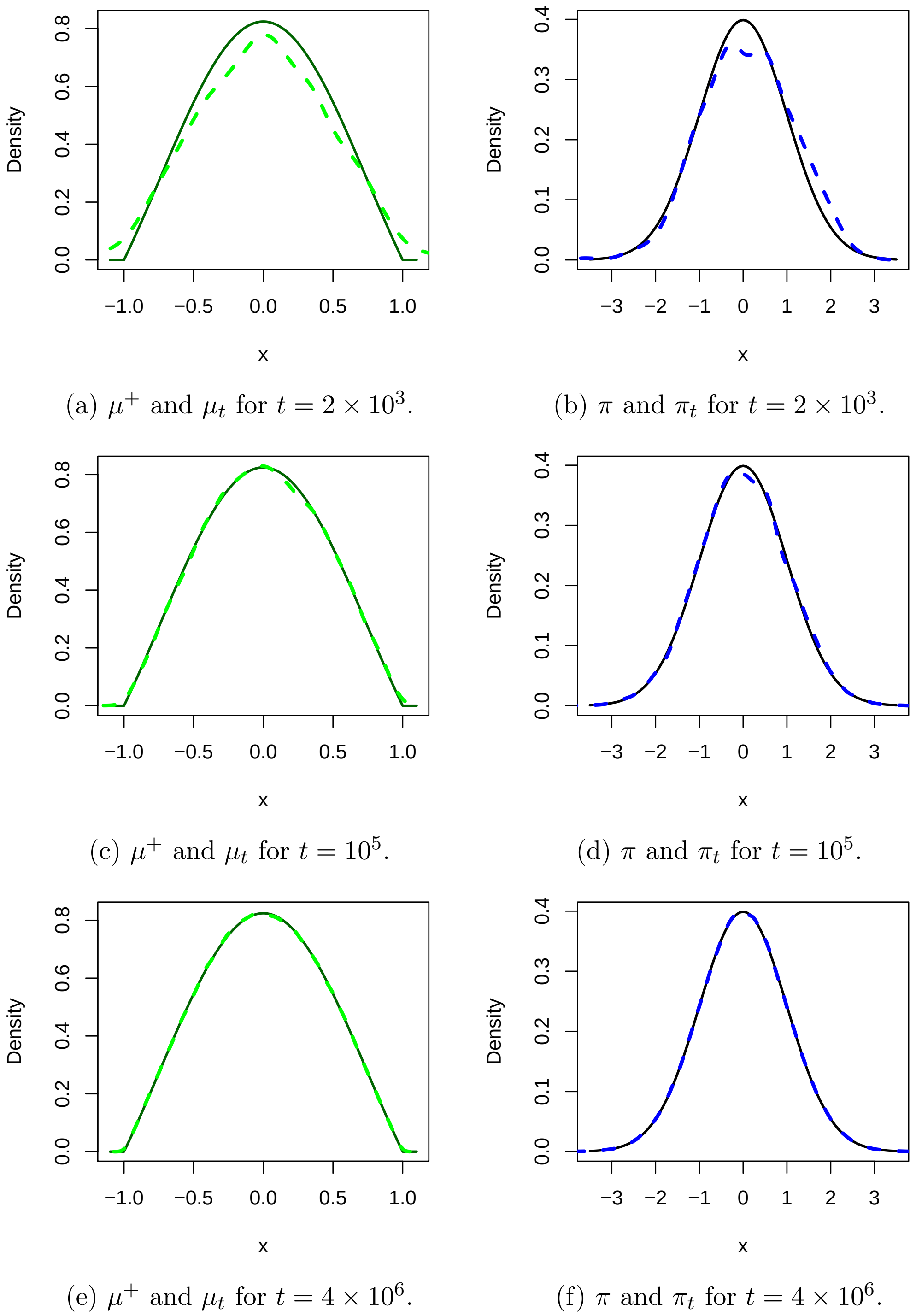
Algorithm 10: The Adaptive Brownian Motion Restore Sampler

```
$t \leftarrow 0, E \leftarrow \emptyset, i \leftarrow 0, X \sim \mu_0$.
while $i < n$ do
    $\tau^{(+)} \sim \text{Exp}(\mathcal{K}^+), \tau^{(\Lambda)} \sim \text{Exp}(\Lambda), \tau^{(-)} \sim \text{Exp}(K^-)$.
    if $\tau^{(+)} < \tau^{(\Lambda)}$ and $\tau^{(+)} < \tau^{(-)}$ then
        $X \sim \mathcal{N}(X, \tau^{(+)}), t \leftarrow t + \tau^{(+)}, u \sim \mathcal{U}[0,1]$.
        if $u < \kappa^+(X)/\mathcal{K}^+$ then
            if $|E| = 0$ then
                $X \sim \mu_0$.
            else
                $X \sim \mathcal{U}(E)$ with probability $t/(a+t)$, else $X \sim \mu_0$.
            end
            $i \leftarrow i + 1$.
        end
    else if $\tau^{(\Lambda)} < \tau^{(+)}$ and $\tau^{(\Lambda)} < \tau^{(-)}$ then
        $X \sim \mathcal{N}(X, \tau^{(\Lambda)}), t \leftarrow t + \tau^{(\Lambda)}$. Record $X, t, i$.
    else
        $X \sim \mathcal{N}(X, \tau^{(-)}), t \leftarrow t + \tau^{(-)}, u \sim \mathcal{U}[0,1]$.
        if $u < \kappa^-(X)/K^-$ then $E \leftarrow E \cup \{X\}$.
    end
end
```

Figure 5.7 illustrates the method with a simple example: $\pi \equiv \mathcal{N}(0,1), \mu_0 \equiv \mathcal{N}(0,1/3)$ and $a = 1000$. Plots of μ_t and π_t for $t \in \{2000, 10^5, 4 \times 10^6\}$ show the distributions converge. Note, a kernel density estimate of π_t is shown, whilst the plot of μ_t is a mixture of μ_0 and a kernel density estimate of the discrete measure.

5.3.1 Algorithm Inputs

Though Adaptive Restore simplifies and improves Restore simulation by allowing the use of κ^+, there are a number of tuning parameters. In this section, we



(a) μ^+ and μ_t for $t = 2 \times 10^3$.

(b) π and π_t for $t = 2 \times 10^3$.

(c) μ^+ and μ_t for $t = 10^5$.

(d) π and π_t for $t = 10^5$.

(e) μ^+ and μ_t for $t = 4 \times 10^6$.

(f) π and π_t for $t = 4 \times 10^6$.

Figure 5.7: Adaptive Restore for $\pi \equiv \mathcal{N}(0, 1), \mu_0 \equiv \mathcal{N}(0, 1/3)$ and $a = 1000$. In figures (a), (c) and (e), the solid green line shows μ^+ and the dashed green line shows μ_t, where a density estimate of the discrete component is used. In Figures (b), (d) and (f) the black line shows π and the dashed blue line shows a density estimate of π.

provide guidance on selecting these tuning parameters.

Regeneration Rate Truncation Level

Recall from subsection 5.3.1 that for many target distributions, the regeneration rate is *not* bounded. Even when the ratio μ/π is bounded, the partial regeneration rate corresponding to many target distributions is unbounded. It is then necessary to truncate the regeneration rate, so that Poisson thinning may be used to simulation regeneration events, even when the minimal regeneration rate is used. The truncation level should be as small as possible, in order to control the computational expense of simulating the process, whilst being large enough that the error introduced is negligible. It seems natural to set $\mathcal{K}^+$ so that

$$\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] < \epsilon,$$

for $\epsilon > 0$ a small constant, such as $\epsilon = 0.01, 0.001, 0.0001$. This is equivalent to ensuring that κ^+ is only truncated in a region with small mass under π, thus only a small error is introduced.

Note that rate κ^+ tends to be highest in areas where density π is lowest: in the tails of π. Simulating $\{X_t\}_{t\geq 0}$ under $\kappa^+ \wedge \mathcal{K}^+$ instead of κ^+ means the rate is lower in the tails of π and hence $\{X_t\}_{t\geq 0}$ spends more time in the tails.

When π is an isotropic Gaussian, it is possible to compute this $\mathcal{K}^+$ since,

$$\begin{aligned}\mathbb{P}[\kappa^+(X) < \mathcal{K}^+] &= \mathbb{P}[0.5(x^T x - d) < \mathcal{K}^+], \\ &= \mathbb{P}\left[\sum_{i=1}^{d} x_i^2 < 2\mathcal{K}^+ + d\right], \\ &= \mathbb{P}[Q < 2\mathcal{K}^+ + d],\end{aligned}$$

for $Q \sim \chi^2_d$. In other words, the quantile function of a chi-squared variable with d degrees of freedom can be used to choose $\mathcal{K}^+$. Figure 5.8 shows plots of $\mathcal{K}^+$ satisfying $\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] = \epsilon$ for $\epsilon = 0.01, 0.001, 0.0001$. Since $\mathcal{K}^+$ dominates the computational cost of the algorithm, the plot gives an insight into the algorithm's complexity. For example, when $d = 100$, choosing $\mathcal{K}^+ = 30$ would be prudent, in which case κ^+ is evaluated on average 30 times per unit of simulation time.

The choice of ϵ should balance computational cost with accuracy. Smaller ϵ necessitate large $\mathcal{K}^+$ which in turn increases computational cost. However, larger ϵ (smaller $\mathcal{K}^+$) induces larger error.

As an illustration, we used Brownian Motion Restore to sample a bivariate Gaussian distribution. To avoid any burn-in issues encountered with Adap-

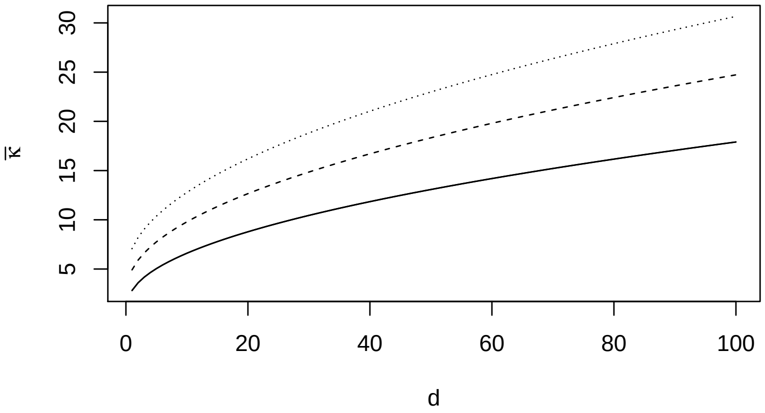


Figure 5.8: Guidance on choosing $\mathcal{K}^+$ so that $\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] = \epsilon$ for π a d-dimensional isotropic Gaussian distribution. Lines for $\epsilon = 0.01, 0.001, 0.0001$ are respectively solid, dashed and dotted.

tive Restore, we used rejection sampling to obtain samples from μ^+. We compared estimates of $\mathbb{E}[X^T X]$ for $\mathcal{K}^+ \in \{3.61, 5.91, 8.21\}$, which correspond to $\epsilon = 0.01, 0.001, 0.0001$. For each truncation level, 100 processes were generated, each consisting of 4×10^5 tours. Samples were recorded at rate 1, resulting in just over a million samples per process. Since $X^T X \sim \chi_2^2$, we have $\mathbb{E}[X^T X] = 2$. Estimates of $\mathbb{E}[X^T X]$ for $\mathcal{K}^+$ corresponding to a given ϵ are shown in Figure 5.9. Note that the functional is a difficult one to estimate, one reason being that it compresses a vector to a scalar and another reason being that taking the square of components of the vector amplifies inaccuracies. In this example, $\epsilon = 0.0001$ results in negligle bias (46 estimates were less than 2), $\epsilon = 0.001$ in small but noticable bias (14 estimates were less than 2) and $\epsilon = 0.01$ in significant bias (0 estimates were less than 2; the smallest estimate was 2.02644). Note that the length of tours was on average smallest for $\epsilon = 0.0001$ and largest for $\epsilon = 0.01$, owing to the truncation of κ. Thus for a fixed number of tours and output rate, more samples are produced for larger ϵ.

An appropriate value of $\mathcal{K}^+$ won't be known in advance. An approach one could take is to monitor the quantile function of $\kappa^+(X)$, then increase $\mathcal{K}^+$ if it is smaller than a proportion $1-\epsilon$ of these. This method is summarized by Algorithm

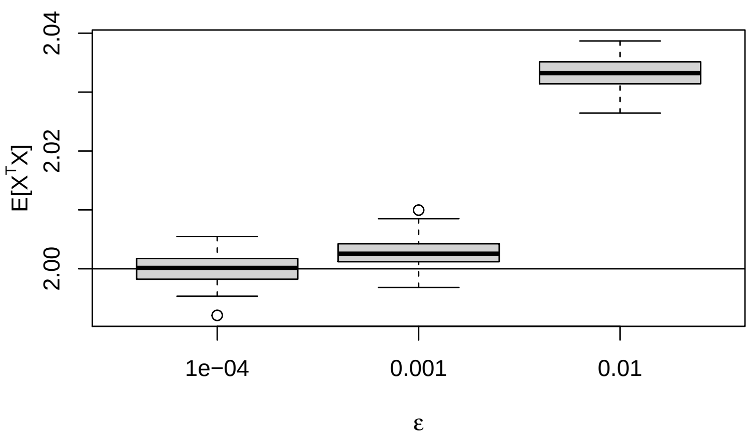


Figure 5.9: Bias introduced by truncating the regeneration rate. Boxplot of 100 estimates of $\mathbb{E}[X^T X]$ for a bivariate Gaussian distribution, computed using Brownian Motion Restore with $\mathcal{K}^+$ such that $\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] = \epsilon; \epsilon \in \{0.01, 0.001, 0.0001\}$, so that $\mathcal{K}^+ \in \{3.61, 5.91, 8.21\}$.

11. Here, χ is the set of output states.

Algorithm 11: Monotonically Increasing Adaption of the truncation of the Regeneration Rate

```
$\chi \leftarrow \emptyset, j \leftarrow \lfloor (1-\epsilon)n \rfloor$
while $|\chi| < n$ do
    Simulate $\{X_t\}_{t \geq 0}$ until $n$ samples are recorded in $\chi$
    Relabel $x_i \in \chi; i = 1, \dots, n$; so that $\kappa^+(x_1) < \kappa^+(x_2) < \cdots < \kappa^+(x_n)$
    if $\mathcal{K}^+ < \kappa^+(x_j)$ then
        $\mathcal{K}^+ \leftarrow \kappa^+(x_j)$
        $\chi \leftarrow \emptyset$
    end
end
```

An issue with this method is that if the initial value of $\mathcal{K}^+$ is too low, then the invariant distribution of the process will have heavier tails than the target distribution, so the estimate of the quantile function of $\kappa^+(X)$ will result in $\mathcal{K}^+$ being overestimated. For example, consider π as an unnormalized bivariate zero-mean Gaussian distribution with covariance matrix Σ given by equation (4.11). Then for $n = 10^4$ and $\epsilon = 0.01$, initial values $\mathcal{K}^+ \in \{4, 8, 12\}$ result in final estimates of $\mathcal{K}^+ \in \{21.07, 12.59, 12\}$. This suggests that it may help to relax the restriction that adaption of $\mathcal{K}^+$ be monotonically increasing. Instead, allow $\mathcal{K}^+$ to be adapted upwards or downwards and monitor $\mathcal{K}^+$ for convergence.

Algorithm 12 summarises the method. Now, the process is simulated until n output states are produced, then κ^+ is adapted. This procedure is repeated m

times. Alternatively, the procedure could be repeated until the change in κ^+ is less than Δ, for $\Delta > 0$ some small constant.

Algorithm 12: Adaption of the truncation of the Regeneration Rate

```
$\chi \leftarrow \emptyset, j \leftarrow \lfloor (1-\epsilon)n \rfloor$
for $i$ in 1 to $m$ do
    Simulate $\{X_t\}_{t\geq 0}$ until $n$ samples are recorded in $\chi$
    Relabel $x_i \in \chi; i = 1, \ldots, n;$ so that $\kappa^+(x_1) < \kappa^+(x_2) < \cdots < \kappa^+(x_n)$
    $\mathcal{K}^+ \leftarrow \kappa^+(x_j)$
    $\chi \leftarrow \emptyset$
end
Simulate $\{X_t\}_{t\geq 0}$ until $n$ samples are recorded in $\chi$
```

We continue with the same example, allowing $m = 5$ adaptions of $\mathcal{K}^+$ either up or down. Equilibrium is reached at about $\mathcal{K}^+ \approx 11$.

Output Rate

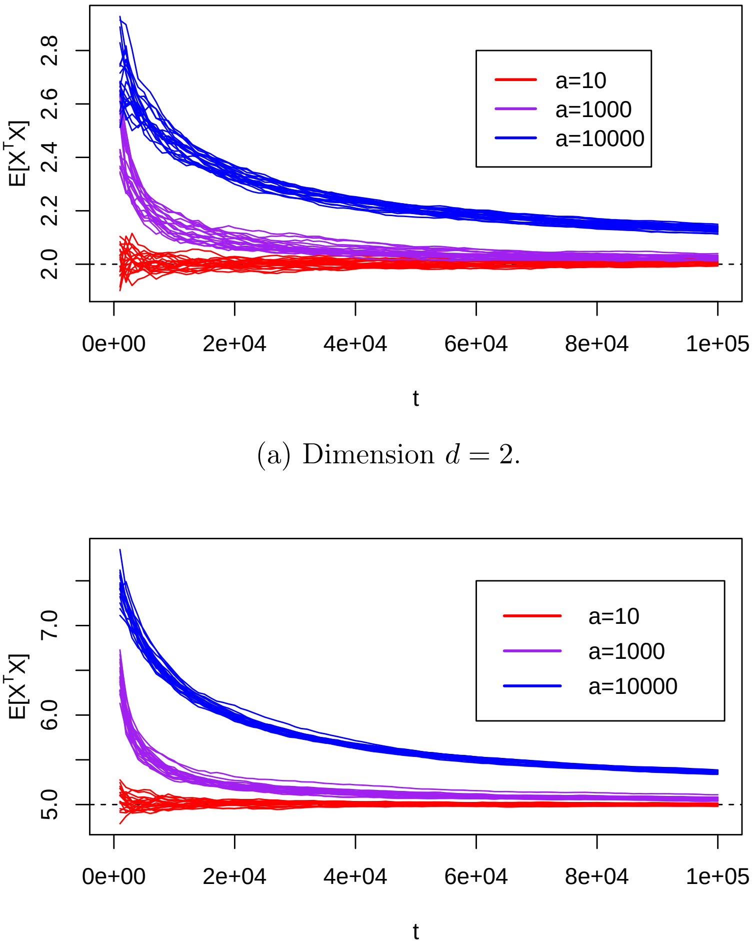
The rate Λ at which the state of the process is stored as output should be adapted so that, on average, each tour of the process produces at least one output state. Recall $\mathbb{E}_\mu[\tau]$ is the expected tour length under regeneration distribution μ. If $\Lambda < \mathbb{E}_\mu[\tau]^{-1}$ then on average, each tour produces less than one output state. This results in tours, which on convergence of μ_t to μ^+ are independent and identically distributed, effectively being wasted. If Λ is much greater than $\mathbb{E}_\mu[\tau]^{-1}$, then output states will be highly correlated. A sensible choice might be $\Lambda = a\mathbb{E}_\mu[\tau]^{-1}$, with say $1 < a < 10$. Suppose that in a burn-in period of length b, during which no output is recorded, m tours are simulated. Then Λ may be set as am/b.

Regeneration Rate Truncation Level and Output Rate

When suitable values for both $\mathcal{K}^+$ and Λ are unknown in advance, they should be tuned in tandem, since each parameter affects the other. When $\mathcal{K}^+$ increases, $\mathbb{E}_\mu[\tau]$ decreases, which affects the tuning of Λ. Conversely, Λ affects the tuning procedure for $\mathcal{K}^+$, which relies on the quantile function of $\kappa(X)$, estimated from samples $x_1, \ldots, x_n$ recorded at rate Λ. During the burn-in period, we suggest iterating between updating κ^+ given fixed Λ then updating Λ given fixed κ^+.

Discrete Measure Dominance Time

We investigate the effect of a, the discrete measure dominance time, on convergence of the process when $d \in \{2, 5\}, \pi \equiv \mathcal{N}(0, I_d)$ and $\mu_0 \equiv \mathcal{N}(1, I_d)$. We do not



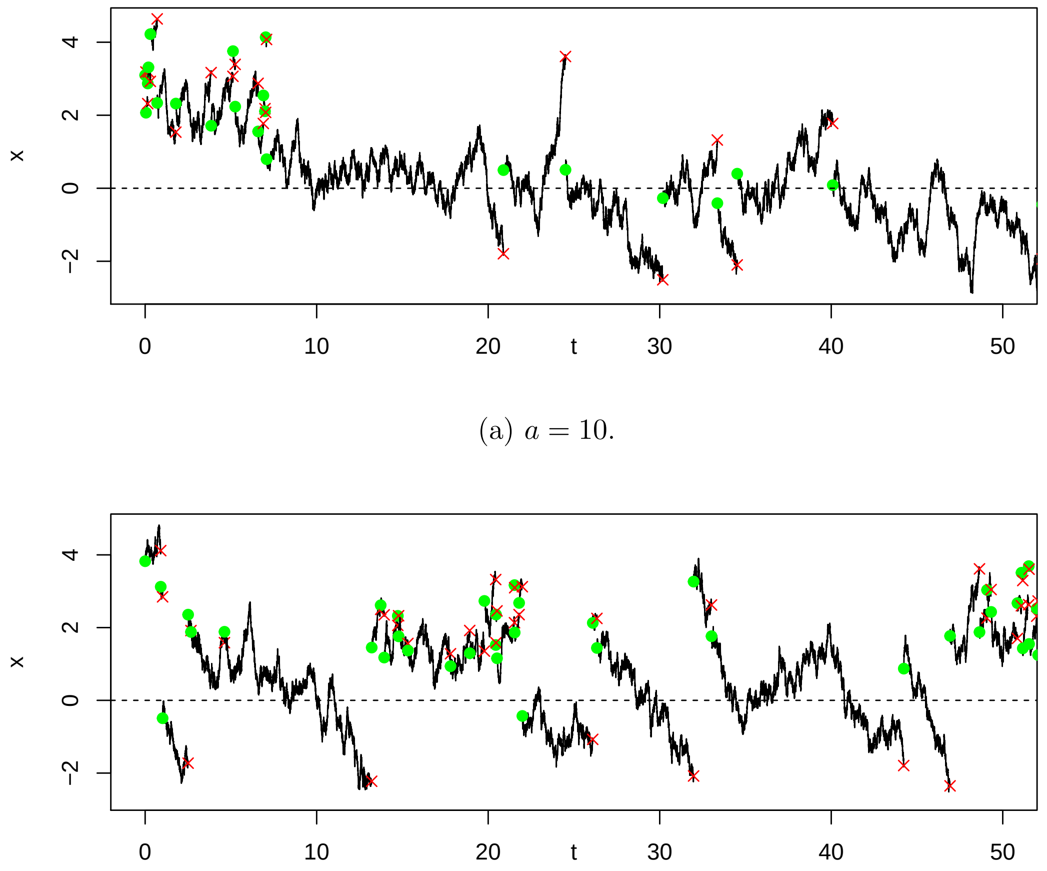
(a) Dimension $d = 2$.

(b) Dimension $d = 5$.

Figure 5.10: Convergence of an Adaptive Restore process for different values of a, the discrete measure dominance time, when $d = 2$ and $d = 5$, for $\pi \equiv \mathcal{N}(0, I_d)$ and $\mu_0 \equiv \mathcal{N}(1, I_d)$.

aim to find the empirically optimal value of a for this example, since it's likely that such a value would be specific to this problem; instead we aim to get a sense of the order of magnitude of a good choice of a. For each $a \in \{10, 1000, 10^4\}$, Figure 5.10 shows rolling estimates of $\mathbb{E}[X^T X]$ for 20 independent processes. For this toy example and for the values of a considered, it appears $a = 10$ is best. However, it remains unclear how to choose optimal a for arbitrary target distributions.

When μ_0 is not centred at μ^+, having a large means that μ_t takes longer to converge in distribution to μ^+. By contrast, small a allows for quicker adaption. Figure 5.11 attempts to illustrate this via an example. The figures show sample paths of two processes, each with $\pi \equiv \mathcal{N}(0, 1)$ and $\mu_0 \equiv \mathcal{N}(2, 1)$. Output states immediately proceeding regeneration times are shown by green dots, while output



(a) $a = 10$.

(b) $a = 10^4$.

Figure 5.11: Sample paths of Adaptive Restore processes with $\pi \equiv \mathcal{N}(0,1), \mu_0 \equiv \mathcal{N}(2,1)$ and different values of a, the discrete measure dominance time. Green dots and red crosses show respectively output states immediately after and before regeneration times.

states immediately preceeding regeneration times are shown with red crosses. The top figure has $a = 10$, while the bottom figure has $a = 10^4$. For $a = 10$, regenerations quickly begin to occur closer to zero. By contrast, for $a = 10^4$, regenerations continue to be centred at two, which hinders convergence.

However, a larger value of a encourages more regenerations from μ_0, which makes it more likely for $\{X_t\}_{t\geq 0}$ to explore regions it hasn't previously visited. This may be beneficial for some target distributions.

Estimating the Infimum of the Partial Regeneration Rate

The dominating rate K^- is not initially known. For the examples presented in this thesis, a suitable upper bound on κ^- was computed by evaluating this rate at the states of a Markov chain with invariant distribution π, generated using the RWM algorithm. Here we consider how a suitable upper bound on κ^- could be computed, without resorting to using an extensive pre-run of a Metropolis-Hastings algorithm. A sensible strategy may be to simulate a sequence of Adaptive Restore processes, with process i using a value K_i^- as an estimate of a bound on κ^- and $K_0^- < K_1^- < \cdots$. During the simulation of process i, if a state x is identified at which $\kappa^-(x) > K_i^-$, then process i is terminated and process $i+1$ begun with $K_{i+1}^- = \kappa^-(x) + \underline{\kappa}$ for $\underline{\kappa} > 0$ a small constant such as $\underline{\kappa} = 0.01$. The first estimate of a bound, K_0^-, may be approximated using gradient descent on $\tilde{\kappa}$. Alternatively, K_0^- may be based on the value of K^- when π is an isotropic Gaussian distribution, which is $d/2$ for d the dimension.

To describe the adaptive mechanism more thoroughly, let $\{X_t^{(i)}\}_{t\geq 0}$ be an Adaptive Restore process, simulated with K_i^- serving as an approximate upper bound on κ^-. In addition, let $T_0^{(K_i^-)}, T_1^{(K_i^-)}, \dots$ be the arrival times of the rate K_i^- Poisson process. At these times, κ^- is evaluated. Let

$$\iota^{(i)} := \inf\left\{t : \kappa^-(X_t^{(i)}) > K_i^- \text{ and } t \in \{T_0^{(K_i^-)}, T_1^{(K_i^-)}, \dots\}\right\}.$$

Suppose that $\iota^{(i)} < \infty$ and that $X_{\iota^{(i)}} = x$. Then define $K_{i+1}^- = \kappa^-(x) + \underline{\kappa}$ and simulate process $\{X_t^{(i+1)}\}_{t\geq 0}$. The rate κ^- will eventually be evaluated at a state close enough to the point at which κ^- has its maximum value to trigger K_{i+1}^- to satisfy $K_{i+1}^- > K^-$.

Burn-in and Convergence

A nice aspect of Standard Restore is that there is no burn-in period. Since tours are independent and identically distributed, with invariant distribution π, there is no reason to discard a portion of the process. Unfortunately this quality is lost when using Adaptive Restore, since a burn-in period is needed during which μ_t and π_t converge approximately to μ^+ and π.

Empirical studies show that convergence can be slow. For example, let $X \sim \pi \equiv \mathcal{N}(0, I_5)$ and $\mu_0 \equiv \mathcal{N}(5^{-1/2}, I_5)$. Then $X^T X \sim \chi_5^2$ so $\mathbb{E}[X^T X] = 5$. The rolling average for the estimate of $\mathbb{E}[X^T X]$ is shown in Figure 5.12, for an Adaptive Restore process simulated for a total time of 8×10^6, with $a = 1000$. Wall-clock time for the simulation was 7 minutes. The process appears to have approximately converged at around $t = 3 \times 10^6$. The regeneration rate was truncated at $\mathcal{K}^+ = 15.44$, which is such that $\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] = 10^{-6}$. Note that choosing $\mathcal{K}^+ = 12.93$, which is such that $\mathbb{P}[\kappa^+(X) > \mathcal{K}^+] = 10^{-5}$, resulted in obvious bias; this again shows the importance of choosing a sufficiently high truncation level.

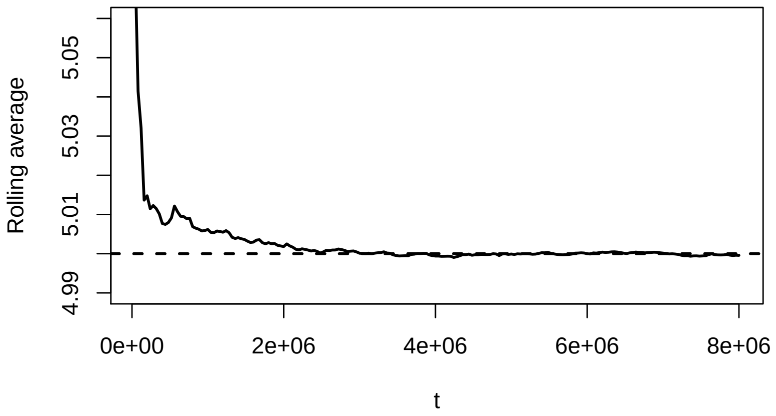


Figure 5.12: Rolling estimate of $\mathbb{E}[X^T X]$ for $d = 5, X \sim \pi \equiv \mathcal{N}(0, I_d)$ and $\mu_0 \equiv \mathcal{N}(d^{-1/2}, I_d)$, computed via the simulation of an Adaptive Restore process.

5.3.2 Initializing from a Point Mass

An interesting special case is when the initial regeneration distribution is a point mass. Setting $a = 0$ and $\mu_0(x) = \delta_{x_0}(x)$, defining $T_0^{(-)} = 0$ and $X_{T_0^{(-)}} = x_0$ gives a simpler expression for μ_t:

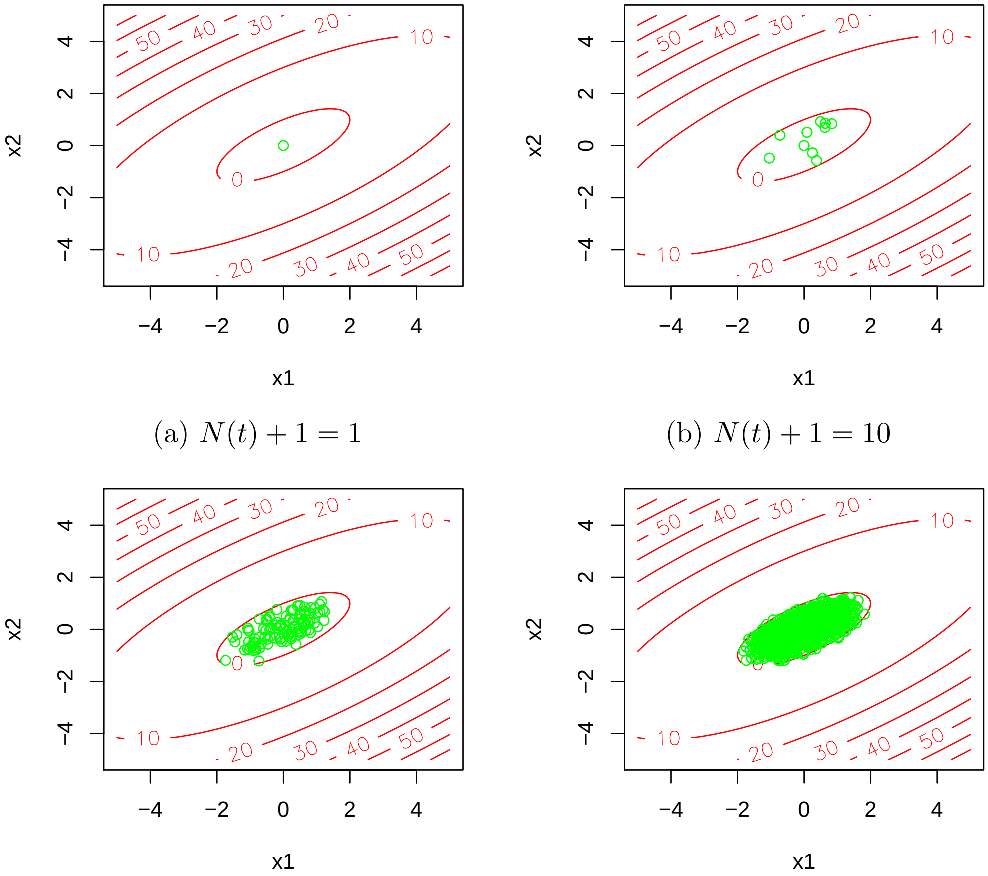
$$\mu_t(x) = \frac{1}{N(t)+1} \sum_{i=0}^{N(t)} \delta_{x_i}(x).$$

Though it is possible to choose μ_0 to be a point mass, it is not recommended. Since μ_t starts as μ_0 but is adapted to become μ^+, it seems natural that μ_0 should be a good approximation of μ^+. A point approximation of μ^+ is unlikely to be a good approximation.

Figure 5.13 demonstrates this special case, when $\pi \equiv \mathcal{N}(0, \Sigma)$, with Σ given by (5.2). Contours of κ^+ are shown in red. The point masses forming μ_t, when $N(t) + 1 = 1, 10, 100, 1000$, are shown by green dots. Since μ^+ has support on region $\{x \in \mathcal{X} : \tilde{\kappa}(x) < 0\}$, the green dots are all within the ellipse defined by $\tilde{\kappa}(x) = 0$.

5.3.3 Examples

The examples presented show that Adaptive Restore can significantly decrease the truncation level used for simulation of the algorithm. This is especially the case when π has skewed tails.



(a) $N(t) + 1 = 1$

(b) $N(t) + 1 = 10$

(c) $N(t) + 1 = 100$

(d) $N(t) + 1 = 1000$

Figure 5.13: Adaptive Restore with initial point-mass regeneration distribution and Gaussian target distribution. Contours of κ^+ in red. Point masses making up μ_t are green dots.

Transformed Beta Distribution

Consider $X' \sim \text{Beta}(2,2)$, so that $\pi_{X'}(x') \propto x'(1-x')$ for $x' \in [0,1]$. Let X be defined by the logit transformation of X', that is $X = \log\left(\frac{X'}{1-X'}\right)$, so that X has support on the real line. The inverse of this transformation is $X' = \frac{e^X}{e^X+1}$. The Jacobian is $\frac{dx'}{dx} = \frac{e^x}{(e^x+1)^2}$. Thus X has density:

$$\pi(x) \propto \frac{e^x}{e^x+1}\left(1 - \frac{e^x}{e^x+1}\right)\frac{e^x}{(e^x+1)^2} \propto \frac{e^{2x}}{(e^x+1)^4}.$$

We call the corresponding probability distribution the *transformed beta distribution.* Its energy and the derivatives of the energy are given by:

$$U(x) = 4\log(e^x+1) - 2x + \text{const},$$

$$\nabla U(x) = \frac{2(e^x-1)}{e^x+1},$$

$$\Delta U(x) = \frac{4e^x}{(e^x+1)^2}.$$

The partial regeneration rate is

$$\tilde{\kappa}(x) = \frac{4e^{2x} - 12e^x + 4}{2(e^x+1)^2}.$$

Note that $\min_{x\in\mathbb{R}} \tilde{\kappa}(x) = -0.5$. Since $\tilde{\kappa}(x) < 2, \forall x \in \mathbb{R}$, this distribution is a useful test case, since an Adaptive Restore process may be efficiently simulated without any truncation of the regeneration rate. In addition, the first and second moments, 0 and $(\pi^2-6)/3$, may be computed analytically.

Taking $\mu_0 \equiv \mathcal{N}(0.5, 1)$ we simulated 200 Adaptive Restore processes each with a burn-in period of 5×10^6 followed by a period of length 10^6 during which output was recorded at rate 10. We deliberately chose μ_0 to be centred away from the mean of π, in order to test that the process still converges. The discrete measure dominance time was 1000. 106 estimates of the first moment were greater than the exact first moment. 98 estimates of the second moment were greater than the exact second moment. This indicates the processes have (approximately) converged to the correct invariant distribution.

Logistic Regression Model of Breast Cancer

We used Adaptive Restore to simulate from the (transformed) posterior of a Logistic Regression model of breast cancer ($d = 10$). We used a Gaussian product prior with variance $\sigma^2 = 400$. Following Gelman et al. (2008), we scaled the data so that response variables were defined on $\{-1, 1\}$, non-binary predictors

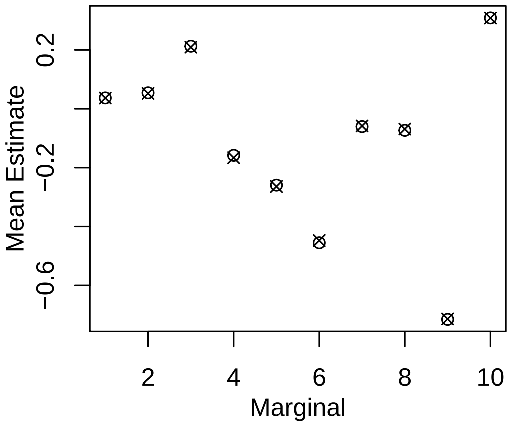


Figure 5.14: Estimates of the mean of each marginal of the (transformed) posterior distribution of a Logistic Regression model of Breast Cancer. Circles show a RWM estimate, crosses an Adaptive Restore estimate.

had mean 0 and standard deviation 0.5, while binary predictors had mean 0 and range 1. The posterior distribution was transformed based on its Laplace approximation.

We simulated an Adaptive Restore process with $\mu_0 \equiv \mathcal{N}(0, I_d)$ and parameters $K^- = 5.2, \mathcal{K}^+ = 19.64, \Lambda = 10.0, a = 1000$, $b = 6 \times 10^6, T = 10^6$. We chose a long burn-in time because other experiments have shown that this is necessary for convergence of (μ_t, π_t). To check the estimate, we generated 10^6 samples using the RWM algorithm, first tuning the scale of the symmetric proposal distribution and using a thinning interval of 50 so that the samples had small autocorrelation. Figure 5.14 plots the estimate of the mean of each marginal for the RWM estimate (circles) and the Adaptive Restore estimate (crosses). The Euclidean distance between these estimates was 0.014 (2.s.f).

Although it is possible to use Adaptive Restore to correctly compute the first moment of this target distribution, the example also demonstrates that the process is expensive to simulate, both in terms of time and memory. It took 5 hours and 20 minutes to simulate the Adaptive Restore process, whereas it took only 26 minutes to generate the RWM Markov chain. Furthermore, the Adaptive Restore process requires a lot of memory, since all the states in the discrete component of the measure μ_t must be recorded. The sample path presented for this example was simulated on a laptop, which had enough memory for the process to be successfully simulated. Ideally, marginal credible regions for the estimates would be displayed, created by simulating multiple independent Adaptive Restore processes. Unfortunately, we were unable to simulate these multiple processes. Doing so on a laptop would have taken too long (serially simulating 100 independent processes would have taken roughly a month). We attempted to use a compute server to simulate the processes in parallel, but found that the programs crashed because they used too much memory. The high memory requirement of the algorithm is thus a serious problem. In the discussion of this chapter (section 5.4), we give an idea for a strategy for reducing the memory requirement of the algorithm, which we call Adaptive Restore with short-term memory.

Hierarchical Model of Pump Failure

Consider the following hierarchical model of pump failure (Carlin and Gelfand, 1991):

$$\begin{aligned} R_i &\sim \text{Poisson}(X_i' t_i); i = 1, 2, \ldots, 10; \\ X_i' &\sim \text{Gamma}(c_1, X_{11}'); i = 1, 2 \ldots, 10; \\ X_{11}' &\sim \text{InverseGamma}(c_2, c_3); \end{aligned}$$

with constants $c_1 = 1.802, c_2 = 2.01, c_3 = 1.01$. Observation R_i ($i = 1, 2, \ldots, 10$) is the number of recorded failures of pump i, which is observed for a period of time t_i ($i = 1, 2, \ldots, 10$). The failure rate of pump i is X_i' ($i = 1, \ldots, 10$). Before sampling, we transformed the posterior to be defined on $\mathbb{R}^d$ by making a change-of-variables, defining $X_i = \log X_i'$ ($i = 1, \ldots, 10$). We then transformed the posterior again, based on its Laplace approximation, as described in subsections 2.5.1 and 2.5.2.

The posterior exhibits heavy and skewed tails. Under Standard Restore with an isotropic Gaussian regeneration distribution, this results in $\mathbb{E}_\pi[\kappa(X)] \approx 1.9 \times 10^7$, which is far too large for simulation to be practical. By contrast, $\mathbb{E}_\pi[\kappa^+(X)] \approx 1$. We are able to accurately compute the first moment of the posterior in less than an hour of simulation time, by setting $b = 5 \times 10^6$ and $T = 5 \times 10^6$.

Log-Gaussian Cox Point Process Model

The Log-Gaussian Cox Point Process model has been used to test HMC samplers (Hirt et al., 2021; Girolami and Calderhead, 2011). A $[0, 1]^2$ area is divided into a $n \times n$ grid. The number of points $Y_{i,j}$ in cell i, j is conditionally independent of the number of points in other cells given the cell's latent intensity $\Lambda_{i,j}$ and has Poisson distribution with rate $n^2 \Lambda_{i,j}$. Here, $\Lambda_{i,j} = \exp\{X_{i,j}\}$ and $X = \{X_{i,j}\}$ is a latent field. Let x and y be vectors representing X and Y. Assume X is a

Gaussian process with mean vector zero and covariance function

$$\Sigma_{(i,j),(i',j')} = \exp\{-\delta(i,i';j,j')/n\},$$

where $\delta(i,i';j,j') = ((i-i')^2 + (j-j')^2)^{1/2}$. We have:

$$\log \pi(x|y) = \sum_{i,j} y_{i,j}x_{i,j} - n^2 \exp\{x_{i,j}\} - \frac{1}{2}x^T\Sigma^{-1}x + \text{const.}$$

Writing $\exp\{x\}$ for the vector with $(\exp\{x\})_{i,j} = \exp\{x_{i,j}\}$, we have

$$\begin{aligned}\nabla_x \log \pi(x|y) &= y - m\exp\{x\} - \Sigma^{-1}x,\\ H_{\log \pi}(x) &= \operatorname{diag}(-m\exp\{x\}) - \Sigma^{-1},\\ \Delta_x \log \pi(x|y) &= -m\sum_{i,j}\exp\{x_{i,j}\} - \operatorname{trace}(\Sigma^{-1}).\end{aligned}$$

We present results for simulated data on a 5 by 5 grid, so $d = 25$. After a pre-transformation based on the Laplace approximation (see subsections 2.5.1 and 2.5.2), the posterior distribution of this model is close to an Isotropic Gaussian distribution. For Standard Restore, setting $\mu \equiv \mathcal{N}(0, I)$ results in $\mathbb{P}[\kappa(X) < 181] \approx 0.9999$ and $\mathbb{E}[\kappa(X)] \approx 19.5$. Thus $\mathcal{K} = 181$ would be appropriate.

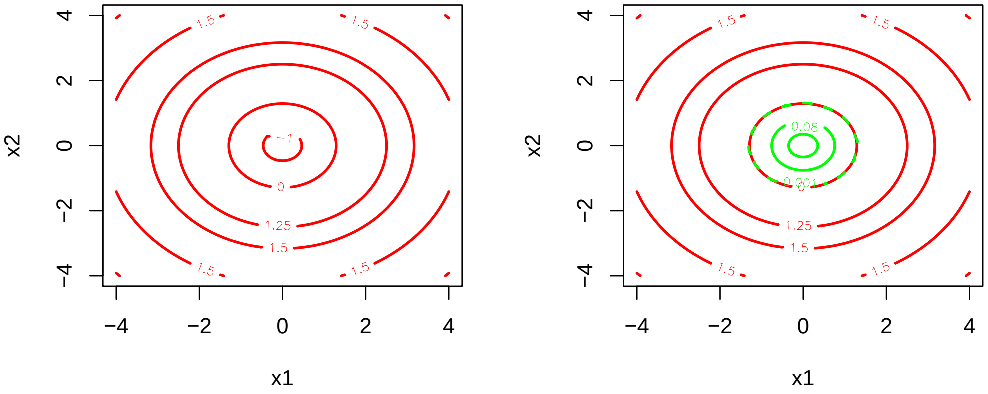
For Adaptive Restore we have $\mathbb{P}[\kappa^+ < 18.5] \approx 0.9999$ and $\mathbb{E}[\kappa^+(X)] \approx 1.4$. Thus Adaptive Restore reduces both the necessary truncation level and average regeneration rate by a factor of 10. However, simulation runs indicate that convergence of the Adaptive process for this $d = 25$ posterior is slow. We used a long burn-in time of $b = 5 \times 10^6$ and a long simulation time of $T = 5 \times 10^6$ to allow the process to converge. Though μ_t does not need to adapt to account for skew so much, it still needs to change significantly so that it is centred correctly — this is harder in higher dimensions.

Multivariate t-distribution

Recall that a d-dimensional multivariate t-distribution with mean m, scale matrix Σ and ν degrees of freedom has density:

$$\pi(x) \propto \left[1 + \frac{1}{\nu}(x-m)^T\Sigma^{-1}(x-m)\right]^{-(\nu+d)/2}.$$

We consider sampling from a bivariate t-distribution with $\nu = 10$, zero mean and identity scale matrix. We then have $\kappa^+(x) < 1.55, \forall x \in \mathbb{R}^d$ (this bound is not tight), so can take $K^+ = 1.55$ (no need to truncate κ). In general, Restore processes are particularly well suited to simulating from t-distributions, since the regeneration rate is naturally bounded. For this example, we can take $K^- = 1.2$.



(a) Contours of $\tilde{\kappa}$.

(b) Contours of κ^+ (red) and μ^+.

Figure 5.15: Contours of $\tilde{\kappa}, \kappa^+$ and μ^+ for π a bivariate t-distribution with $\nu = 10$.

The process is quickly able to recover the true variance of each marginal of the target distribution, which is $\nu/(\nu - 2)$. Figure 5.15 shows contours of $\tilde{\kappa}, \kappa^+$ and μ^+. A notable feature is that, moving outwards from the origin, κ^+ rises to its maximum value then asymptotically tends to zero.

Mixture of Gaussian distributions

We explore the use of an Adaptive Restore process for simulating from the Gaussian mixture distribution

$$\pi(x) = w_1 \mathcal{N}(x; m_1, \Sigma_1) + w_2 \mathcal{N}(x; m_2, \Sigma_2),$$

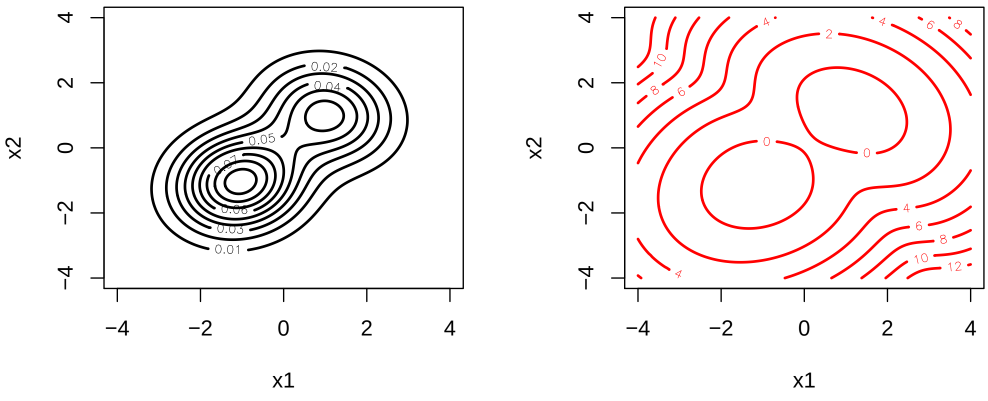
for $w_1 = 0.4, w_2 = 0.6, m_1 = (1.05, 1.05), m_2 = (-1.05, -1.05)$,

$$\Sigma_1 = \begin{pmatrix} 1 & -0.1 \\ -0.1 & 1 \end{pmatrix} \text{ and } \Sigma_2 = \begin{pmatrix} 1 & 0.1 \\ 0.1 & 1 \end{pmatrix}.$$

Figure 5.16 shows contour plots of the density of π and κ^+. In particular, figure 5.16b shows that the region for which κ^+ is zero, which corresponds to the support of μ^+, consists of two separate non-connected areas. For Standard Restore and Adaptive Restore, we set μ and μ_0 respectively to $\mathcal{N}(0, 3I)$.

Standard Restore was able to sample from this distribution well, even though we had to set $\mathcal{K} = 1000$ so that the truncation wouldn't overly affect κ. Setting $\Lambda = 10$, a simulation time of $T = 10^4$ generated samples which produced an estimate of the mean that had Root Mean Square Error (RMSE) 0.00358 (3.s.f). Simulation took 3.5 minutes.

By comparison, Adaptive Restore allows the truncation level to be much reduced, to $\mathcal{K}^+ = 20$. We set $a = 10^4$ to allow both modes to be explored before



(a) Contours of π.

(b) Contours of κ^+.

Figure 5.16: Contours of π and κ^+ for π a mixture of bivariate Gaussian distributions. Note that there are two disjoint compact regions on which κ^+ is zero.

the discrete measure became dominant. A burn-in time of $b = 9 \times 10^5$ and simulation time of $T = 10^5$ took 6 minutes. Setting $\Lambda = 1$, so that in expectation the number of samples produced equals that for Standard Restore, the RMSE was 0.0555 (3.s.f). In this experiment $b+T = 10^6$, which is 100 times greater than the simulation time used for generating a Standard Restore process, yet the RMSE is worse. For this multi-modal example, the Adaptive Restore process converges slowly due to its urn-like behaviour. Though the process converges asymptotically, in the short-run the process tends to visit areas it has visited before, which makes convergence slow.

5.4 Discussion

This chapter has introduced the Adaptive Restore process (McKimm et al., 2022), which adapts the regeneration distribution on the fly. Like Standard Restore, Adaptive Restore benefits from global moves. For target distributions that are hard to approximate with a parametric distribution, Adaptive Restore is more suitable than Standard Restore, because its use of the minimal regeneration rate makes simulation computationally feasible. In comparison to simpler algorithms such as Random Walk Metropolis, the process can still be slow to simulate and convergence appears to be slow when the target is multimodal. However, the algorithm shows promise in sampling distributions with skewed tails, for which Standard Restore can be computationally intractable.

Global dynamics allow the Adaptive Restore process to make large moves across the space, a property shared by Standard Restore. This feature is desirable for MCMC samplers (since it results in a Markov chain with smaller autocorrelation) and has motivated the development of algorithms such as Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal et al., 2011) or its extension, the No-U-Turns Sampler (NUTS) (Hoffman and Gelman, 2014). Crucially, as the dimension d of π increases, μ^+ remains close to π. This means that, unlike other methods making use of global regenerative moves via the independence sampler and Nummelin splitting (Nummelin, 1978; Mykland et al., 1995), global moves are more likely to be to areas of the space where π has significant mass.

Experiments on a number of target distributions have indicated that the Restore process is particularly effective at simulating from heavy-tailed distributions, a class of distributions that other samplers can struggle with (Mengersen and Tweedie, 1996; Roberts and Tweedie, 1996a,b). A heuristic explanation for this behaviour is that regeneration is a useful mechanism for allowing the sampler to escape the tails of the distribution and move back to the centre of the space.

A large benefit of Adaptive Restore over Standard Restore is its use of the minimal regeneration rate. We have shown via an example that even for a sensible choice of fixed μ, the corresponding rate κ can be extremely large in parts of the space. While frequent regeneration is not in itself a bad thing, frequent regeneration into regions of low probability mass is computationally wasteful. Using κ^+ results in π and its derivatives being evaluated far less.

Some properties of Standard Restore that are unfortunately not inherited by Adaptive Restore are independent and identically distributed tours, an absence of burn-in period and the ability to estimate normalizing constants. Moreover, convergence appears to be slow for multi-modal distributions. Since tours begin with distribution μ_t and this distributed changes over time, tours are no longer independent and identically distributed. A burn-in period is required, during which μ_t converges to μ^+ and π_t, the stationary distribution of the process at time t, converges to π. For Standard Restore, $\tilde{C}$, the regeneration constant with the normalizing constant Z absorbed, is defined explicitly and hence can be used to recover Z. On the other hand, for Adaptive Restore this constant is defined implicitly and thus can't be used to recover Z.

Despite these downsides, Adaptive Restore represents a significant improvement on Standard Restore by making simulation tractable for a wider range of target distributions. We have shown that simulation of mid-dimensional target distributions is practical with Adaptive Restore. There is plenty of scope for further research – we briefly lay out some directions.

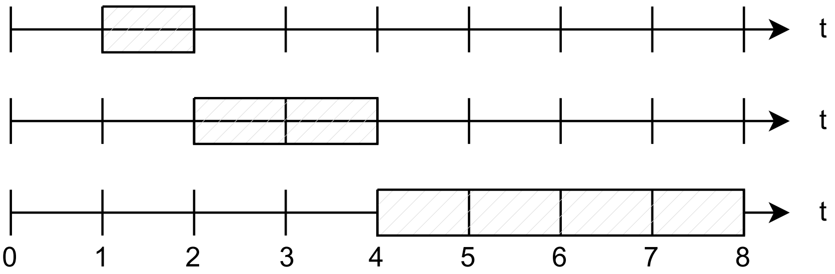


Figure 5.17: For $t = 2, 4, 8$, the filled rectangle indicates $[t/2, t]$. For the short-term memory version of Adaptive Restore, the events of a Poisson process with rate $\kappa^-(X_t)$, which arrive within this interval, contribute to distribution μ_t.

5.4.1 Short-term memory

An Adaptive Restore process with *short-term memory* might converge faster. This would use a modified version of the regeneration distribution at time t. Let

$$E_t := \{X_t : t = T_i^{(-)} \text{ for } i \in 1, \dots, N(t) \text{ and } t/2 < T_i^{(-)} < t\}.$$

If $|E_t| = 0$ then let $\mu_t \equiv \mu_0$, else

$$\mu_t(x) = \frac{t}{a+t} \frac{1}{|E_t|} \sum_{y \in E_t} \delta_y(x) + \frac{a}{a+t} \mu_0(x).$$

Figure 5.17 serves to illustrate how interval $[t/2, t]$ grows with time, yet also leaves interval $[0, t/2]$ forgotten. Point masses added to the regeneration distribution at an early stage may be far less accurate in approximating μ^+ than those added at a later stage, thus removing these point masses from the regeneration distribution may improve convergence.

A computer program for simulating an Adaptive Restore process uses a lot of memory. In this chapter, we saw that this becomes a problem when the total simulation time is large. An Adaptive Restore process with short-term memory would reduce the memory requirements of a computer program implementing the algorithm. Another interesting option to explore for reducing the algorithm's memory requirement would be to thin the samples making up the discrete component of μ_t. It would be interesting to see whether an *Optimal Thinning* method could be used, such as the method proposed by Riabiz et al. (2020), which selects a subset of samples with empirical distribution close to optimal in approximating the sampled distribution.

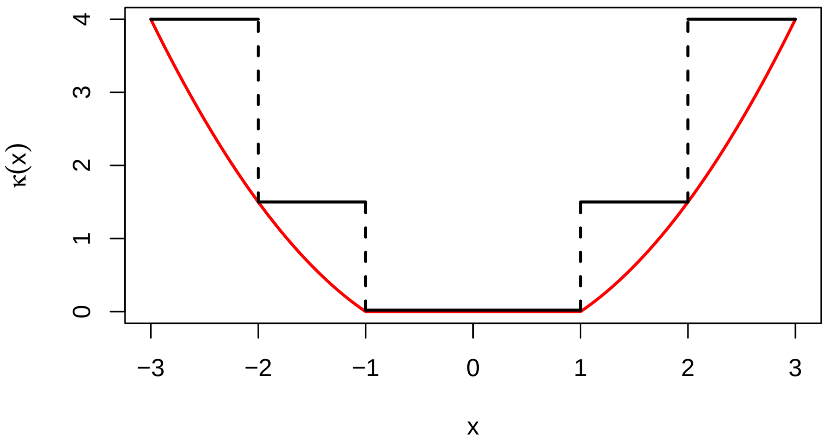


Figure 5.18: An illustration of using local bounds on the regeneration rate. In red, κ^+ for $\pi \equiv \mathcal{N}(0,1)$. In black, the piece-wise constant dominating rate given by equation (5.6).

5.4.2 Local bounds on the regeneration rate

Instead of having a single global bound on the regeneration rate, one could partition the state-space into subsets, each with its own local bound. For example, when π is the standard normal distribution, $\kappa^+(x) = (x^2-1)/2 \vee 0$ and an appropriate global truncation level is $\mathcal{K}^+ = 4$. One could define $\mathcal{K}^+$ as a piecewise-constant function such as:

$$\mathcal{K}^+(x) = \begin{cases} 0, & |x| \leq 1; \\ 1.5, & 1 < |x| \leq 2; \\ 4, & |x| \geq 2. \end{cases} \tag{5.6}$$

Figure 5.18 plots this dominating rate against κ^+. The advantage would be that κ^+ is evaluated less frequently. For this example, κ^+ wouldn't be evaluated as long as X_t remains on the interval $[-1,1]$.

To simulate a Brownian Motion Restore process with piecewise-constant dominating rate would require using conditioned Brownian Motion. That is, one would need to simulate the length of time that the process stays in an interval corresponding to a particular value of the dominating rate. Doing so would not significantly increase the computational cost of the method, but would be a programming challenge.

Chapter 6

Using Regeneration to Correct the Invariant Distribution of a Jump Process

Recall from Section 4.2 that a Restore jump process is defined by enriching some underlying jump process $\{Y_t\}_{t\geq 0}$, with holding rate λ and Markov transition kernel P, with regenerations from distribution μ at a rate κ. When the regeneration rate is given by (4.8), that is

$$\kappa(x) = \frac{\int \lambda(y)\pi(y)p(y,x)dy - \lambda(x)\pi(x)}{\pi(x)} + C\frac{\mu(x)}{\pi(x)},$$

with $C > 0$ such that $\kappa(x) \geq 0, \forall x \in \mathcal{X}$, then the enriched process$\{X_t\}_{t\geq 0}$ is π-invariant.

For an arbitrary choice of Markov transition kernel, it may not be possible to evaluate the integral term in the numerator of (4.8) analytically. Indeed, the fundamental motivation for using Monte Carlo is to evaluate integrals with respect to π. Wang et al. (2021) point out that taking $\lambda \equiv 1$ and P to be a π-invariant Markov transition kernel, the regeneration rate reduces to (4.9):

$$\kappa(x) = C\frac{\mu(x)}{\pi(x)}.$$

Thus Wang et al. (2021) provide an algorithmically simple way to enrich a π-invariant jump process with regenerations. The enriched process enjoys the benefits of regenerative simulation, such as parallel simulation and absence of burn-in, see subsection 2.3.3. However, two of the original motivations for Restore, as set out in the introduction of Chapter 4, were to design algorithms that (i) combine dynamics that by themselves are not π-invariant, in such a way that they compensate for each other and (ii) break free of the requirement on Markov transition kernels to be reversible, which acts as a constraint.

This chapter shows that a class of jump processes which are not already π-invariant may be enriched with regenerations so that the resulting Restore processes are π-invariant. An instance of the class is called a *Jump Process Adjusted with Regenerations* (Jumpar). The word “adjusted” is used to signify that the invariant distribution of the process is altered so that it corresponds to a given target distribution. Thus, the meaning of “adjusted” in this context is the same as that of the same word in “Metropolis Adjusted Langevin Algorithm” (Roberts and Tweedie, 1996a). By contrast, enriching with regenerations an underlying jump process that is already π-invariant does not change the invariant distribution.

The chapter is laid out as follows. Section 6.1 sets out the main framework for enriching non-π-invariant jump processes so that the enriched jump process is π-invariant. Two options for the holding rate are examined in Section 6.2: constant and non-constant. Three choices of the dynamics used in the underlying process are then inspected in Sections 6.3, 6.4 and 6.5: Random Direction, Hamiltonian and Conformal Hamiltonian. Section 6.6 concludes the chapter with a discussion.

6.1 Class of Jump Process Adjusted with Regenerations

Let V be an auxiliary variable defined on $\mathcal{V}$ having distribution $\pi_V(v)$ with known normalizing constant. The augmented variable $\theta := (X, V)$ takes values $\vartheta := (x, v)$ on *phase* space $\Theta := \mathcal{X} \times \mathcal{V}$. A state $\vartheta = (x, v)$ in the augmented space may be thought of as a particle with *position* x and a *velocity* v. Hence the variable V is referred to as the *velocity*. The augmented variable has a distribution with density:

$$\pi_\theta(\vartheta) := \pi_{X\times V}(x, v) := \pi_X(x) \times \pi_V(v).$$

The augmented regeneration distribution has density:

$$\mu_\theta(\vartheta) := \mu_{X\times V}(x, v).$$

The central theorem of this chapter is as follows.

Theorem 4. *Let $\{\Xi_t\}_{t\geq 0}$ be a jump process on Θ with holding rate $\lambda(\vartheta)$ and jump chain transition kernel P corresponding to a bijective transformation $A(\vartheta)$,* *so that* $\forall\vartheta \in \Theta$ *and for all sets* $B \subset \Theta$

$$P(\vartheta, B) = \begin{cases} 1, & \textit{if } A(\vartheta) \in B, \\ 0, & \textit{else.} \end{cases} \tag{6.1}$$

Suppose A^{-1} *is the inverse of* A *and has Jacobian* $J_{A^{-1}}$. *Let* $\{\theta_t\}_{t\geq 0}$ *be the jump process resulting from enriching* $\{\Xi_t\}_{t\geq 0}$ *with regenerations from distribution* μ_θ *at rate* κ. *Then* π_θ *is the invariant distribution of* $\{\theta_t\}_{t\geq 0}$ *for*

$$\kappa(\vartheta) = \frac{\lambda(A^{-1}(\vartheta))\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| - \lambda(\vartheta)\pi_\theta(\vartheta) + C\mu_\theta(\vartheta)}{\pi_\theta(\vartheta)}, \tag{6.2}$$

where the density $C\mu_\theta(\vartheta)$ *is large enough to guarantee that* $\kappa(\vartheta) \geq 0, \forall\vartheta \in \Theta$ *and it is assumed that* $\pi[(\lambda + \kappa)^2] < \infty$.

The jump process $\{\theta\}_{t\geq 0}$ constructed in Theorem 4 is called a Jump Process Adjusted with Regenerations (Jumpar). Augmenting the state space and using a jump chain transition kernel corresponding to a bijective transformation is the mechanism for getting around the potentially difficult task of evaluating the integral in the numerator of (4.8).

Note that the normalizing constant for $\pi_\theta(\vartheta)$ doesn't need to be known, since it can be absorbed into the constant C. To see this write (6.2) in terms of $\tilde{\pi}_\theta(\vartheta)/Z$ and multiply through by Z. Then writing $\tilde{C} = CZ$ we get

$$\kappa(\vartheta) = \frac{\lambda(A^{-1}(\vartheta))\tilde{\pi}_\theta(A^{-1}(\vartheta)) - \lambda(\vartheta)\tilde{\pi}_\theta(\vartheta) + \tilde{C}\mu_\theta(\vartheta)}{\tilde{\pi}_\theta(\vartheta)}. \tag{6.3}$$

We now prove Theorem 4. Wang et al. (2021) noted that trying to prove π-invariance of a Restore process via analysis of its generator results in highly technical difficulties. Instead, they give proofs of invariance which avoid using the generator approach. The following proof is based on that of Theorem 22 of (Wang et al., 2021) and thus likewise avoids complications arising from analysing the generator.

Proof. We drop subscripts θ. The transition kernel of the jump chain of $\{\theta\}_{t\geq 0}$ is:

$$P^\mu(\vartheta, d\vartheta') = \frac{\lambda(\vartheta)}{\lambda(\vartheta) + \kappa(\vartheta)} P(\vartheta, d\vartheta') + \frac{\kappa(\vartheta)}{\lambda(\vartheta) + \kappa(\vartheta)} \mu(d\vartheta'). \tag{6.4}$$

The overall holding rate is:

$$\bar{\lambda}(\vartheta) = \lambda(\vartheta) + \kappa(\vartheta).$$

Let $\{Q_t^\mu : t \geq 0\}$ be the continuous-time semigroup for the Restore process θ. We want to show $\pi Q_t^\mu f = \pi[f]$ for any continuous bounded function f, for each

$t \geq 0$. It suffices to show the time derivative of the mapping $t \to \pi Q_t^\mu f$ is 0 at $t = 0$. In the proof of Theorem 22, Wang et al. (2021) show that

$$\begin{aligned}\frac{d}{dt}Q_t f(\vartheta) = -\bar{\lambda}(\vartheta)\exp\big\{-\bar{\lambda}(\vartheta)t\big\}f(\vartheta) + \bar{\lambda}(\vartheta)P^\mu[Q_t f](\vartheta) \\ - \int_0^t \bar{\lambda}(\vartheta)^2 \exp\Big\{-\bar{\lambda}(\vartheta)(t-s)P^\mu[Q_s f](\vartheta)\Big\}ds.\end{aligned}$$

Since it is assumed that $\pi\left[\bar{\lambda}^2\right] < \infty$, we have:

$$\begin{aligned}\frac{d}{dt}\pi Q_t f\bigg|_{t=0} &= \frac{d}{dt}\int \pi(\vartheta)Q_t f(\vartheta)d\vartheta\bigg|_{t=0}, \\ &= \int \pi(\vartheta)\, \frac{d}{dt}Q_t f(\vartheta)\bigg|_{t=0} d\vartheta, \\ &= \int \pi(\vartheta)\bar{\lambda}(\vartheta)[P^\mu f(\vartheta) - f(\vartheta)]d\vartheta.\end{aligned}$$

Using (6.2), we can show that (6.4) may be written:

$$\begin{aligned}P^\mu(\vartheta, d\vartheta') =& \Big[\lambda(\vartheta)\pi(\vartheta)P(\vartheta, d\vartheta') \\ &+ \big[\lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| \; - \lambda(\vartheta)\pi(\vartheta) + C\mu(\vartheta)\big]\mu(d\vartheta')\Big] \\ &\cdot\Big[\lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| + C\mu(\vartheta)\Big]^{-1}.\end{aligned} \tag{6.5}$$

Thus

$$\begin{aligned}P^\mu f(\vartheta) = \; & \Big[\lambda(\vartheta)\pi(\vartheta)f(A(\vartheta)) \\ &+ \big[\lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| \; - \lambda(\vartheta)\pi(\vartheta) + C\mu(\vartheta)\big] \\ &\cdot \int f(\vartheta')\mu(\vartheta')d\vartheta'\Big] \\ &\cdot\Big[\lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| + C\mu(\vartheta)\Big]^{-1}.\end{aligned}$$

We have

$$\pi(\vartheta)\bar{\lambda}(\vartheta) = \lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| + C\mu(\vartheta).$$

Thus

$$\begin{aligned}\pi(\vartheta)\bar{\lambda}(\vartheta)P^\mu f(\vartheta) = \; & \lambda(\vartheta)\pi(\vartheta)f(A(\vartheta)) \\ &+ \big[\lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| - \lambda(\vartheta)\pi(\vartheta) + C\mu(\vartheta)\big] \\ &\cdot \int f(\vartheta')\mu(\vartheta')d\vartheta'.\end{aligned}$$

and

$$\pi(\vartheta)\bar{\lambda}(\vartheta)f(\vartheta) = \lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|f(\vartheta) + C\mu(\vartheta)f(\vartheta).$$

Now,

$$
\begin{aligned}
\left.\frac{d}{dt}\pi Q_t f\right|_{t=0} = & \int \lambda(\vartheta)\pi(\vartheta)f(A(\vartheta))d\vartheta \\
& + \int \left[\lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| - \lambda(\vartheta)\pi(\vartheta)\right] d\vartheta \\
& \cdot \int f(\vartheta')\mu(\vartheta')d\vartheta' \\
& + \int C\mu(\vartheta)d\vartheta \int f(\vartheta')\mu(\vartheta')d\vartheta' \\
& - \int \lambda(A^{-1}(\vartheta))\pi(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|f(\vartheta)d\vartheta \\
& - \int C\mu(\vartheta)f(\vartheta)d\vartheta.
\end{aligned}
$$

The terms on the first and fifth lines cancel. To see this, use a change of variables. Similarly, the integral on the second line is zero. The fourth and sixth lines cancel, because the first integral on the third line is equal to C. This completes the proof, since we have shown that

$$\left.\frac{d}{dt}\pi Q_t f\right|_{t=0} = 0.$$

□

6.1.1 Velocity Refreshment

The framework developed above may be extended by introducing the option to *refresh* the velocity before each local move by resampling the velocity from the V-marginal of $\lambda\pi_\theta$. For example, if $\lambda \equiv 1$ so that $\lambda\pi_\theta \equiv \pi_\theta \equiv \pi_X\pi_V$, this would amount to resampling V from π_V. Alternatively, if $\lambda \equiv \mu_\theta/\pi_\theta$ and $\mu_\theta \equiv \mu_X\mu_V$ then $\lambda\pi_\theta \equiv \mu_\theta \equiv \mu_X\mu_V$ and refreshing the velocity consists of resampling V from μ_V.

Refreshing the velocity does not change the regeneration rate. We do not prove this, but present an informal derivation for why this is the case. Equation (4.8) gives the expression for the regeneration rate under which a Restore jump process on $\mathcal{X}$ is π-invariant. Rewritten in terms of ϑ, the augmented variable, for Restore jump processes on Θ, this expression is

$$\kappa(\vartheta) = \frac{\lambda\pi P(\vartheta) - \lambda\pi(\vartheta)}{\pi(\vartheta)} + C\frac{\mu(\vartheta)}{\pi(\vartheta)}.$$

Write the Markov transition kernel of the underlying jump process as $P = P_1P_2$, where P_1 is invariant for the V-marginal of $\lambda\pi_\theta$ (and corresponds to refreshing the velocity) and P_2 corresponds to a bijective transformation $A(\vartheta)$, so may be expressed by (6.1). It then follows that

$$\begin{aligned}\kappa(\vartheta) &= \frac{\lambda\pi P_1 P_2(\vartheta) - \lambda\pi(\vartheta)}{\pi(\vartheta)} + C\frac{\mu(\vartheta)}{\pi(\vartheta)},\\ &= \frac{\lambda\pi P_2(\vartheta) - \lambda\pi(\vartheta)}{\pi(\vartheta)} + C\frac{\mu(\vartheta)}{\pi(\vartheta)},\end{aligned}$$

so κ is given by equation (6.2) of Theorem 4.

6.1.2 Simulation

Algorithm 13 describes how to simulate n tours of a Jumpar. Here, i counts the number of tours; $\tau^{(\lambda)}$ and $\tau^{(\kappa)}$ denote the times to the next local and global moves respectively, conditioned on the jump process remaining in its current state; REFRESH is a user-specified boolean variable indicating whether to resample the velocity before making a local move.

Algorithm 13: The Jump Process Adjusted with Regenerations Sampler

```
$i \leftarrow 0, (X, V) \sim \mu_\theta$
while $i < n$ do
    $\tau^{(\lambda)} \sim \mathrm{Exp}(\lambda(X, V))$
    $\tau^{(\kappa)} \sim \mathrm{Exp}(\kappa(X, V))$
    $\tau^{(\lambda+\kappa)} \leftarrow \tau^{(\lambda)} \wedge \tau^{(\kappa)}$
    Record $X, V, \tau^{(\lambda+\kappa)}$
    if $\tau^{(\lambda)} < \tau^{(\kappa)}$ then
        If REFRESH: $V \sim [\lambda\pi_\theta]_V$ (the $V$-marginal of $\lambda\pi_\theta$)
        $(X, V) \leftarrow A(X, V)$
    else
        $X, V \leftarrow \mu_\theta$
        $i \leftarrow i + 1$
    end
end
```

Algorithm 13 is pseudo-code and has been written with simplicity in mind – an efficient implementation would save certain values in memory before each local move, in order to reduce computation. In particular, suppose the process has state ϑ_i. The next state visited is $\vartheta_{i+1} = A(\vartheta_i)$, which has regeneration rate:

$$\kappa(\vartheta_{i+1}) = \frac{\lambda(\vartheta_i)\pi_\theta(\vartheta_i)|J_{A^{-1}}(\vartheta_{i+1})| - \lambda(\vartheta_{i+1})\pi_\theta(\vartheta_{i+1}) + C\mu_\theta(\vartheta_{i+1})}{\pi_\theta(\vartheta_{i+1})}.$$

By saving the value $z_i := \lambda(\vartheta_i)\pi_\theta(\vartheta_i)$ before making a local move, it is possible to decrease the computational cost of the algorithm, since

$$\kappa(\vartheta_{i+1}) = \frac{z_i|J_{A^{-1}}(\vartheta_{i+1})| - \lambda(\vartheta_{i+1})\pi_\theta(\vartheta_{i+1}) + C\mu_\theta(\vartheta_{i+1})}{\pi_\theta(\vartheta_{i+1})}.$$

Thus we only need to compute A^{-1} once, at the beginning of each tour.

6.2 Choice of Holding Rate

To be valid, the regeneration rate must be non-negative everywhere. With this criteria in mind, two choices of holding rate are used. Subsection 6.2.1 uses a constant holding rate and a heuristic method for finding a suitably large regeneration constant. Alternatively, subsection 6.2.2 uses a weighted holding rate, which guarantees that the regeneration rate is non-negative everywhere.

6.2.1 Constant Holding Rate

Suppose the holding rate is constant: $\lambda(\vartheta) \equiv \lambda_0, \forall \vartheta \in \Theta, \lambda_0 > 0$. Then the regeneration rate may be written as

$$\kappa(\vartheta) = \frac{\lambda_0\big[\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| - \pi_\theta(\vartheta)\big] + C\mu_\theta(\vartheta)}{\pi_\theta(\vartheta)}.$$

When $\lambda_0 = 1$:

$$\kappa(\vartheta) = \left[\frac{\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|}{\pi_\theta(\vartheta)} - 1 + C\frac{\mu_\theta(\vartheta)}{\pi_\theta(\vartheta)}\right].$$

Suppose $\pi_X(x) = \tilde{\pi}_X(x)/Z$ and it is possible to evaluate $\tilde{\pi}_X(x)$ but Z is unknown. The augmented target distribution is then $\pi_\theta(\vartheta) = \pi_X(x)\pi_V(v) = \tilde{\pi}_X(x)\pi_V(v)/Z$ and we are able to evaluate $\tilde{\pi}_\theta(\vartheta) = \tilde{\pi}_X(x)\pi_V(v)$. The regeneration rate is then

$$\kappa(\vartheta) = \left[\frac{\tilde{\pi}_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|}{\tilde{\pi}_\theta(\vartheta)} - 1 + \tilde{C}\frac{\mu_\theta(\vartheta)}{\tilde{\pi}_\theta(\vartheta)}\right],$$

where $\tilde{C} = CZ$. As explained in Section 4.5, we may estimate Z as $\tilde{C}T/n$, for T the total simulation time and n the number of tours simulated.

Complexity

It's assumed that the cost of evaluating μ_θ is negligible in comparison to that of evaluating π_θ, so can be ignored. Suppose ϑ_i for $i = 0, 1, 2, \ldots$ has position i in the jump chain of a given tour. Then the computational cost associated with ϑ_0 is the cost of evaluating π_θ at ϑ_0, evaluating $|J_{A^{-1}}(\vartheta_0)|$, computing $A^{-1}(\vartheta_0)$

and evaluating π_θ at $A^{-1}(\vartheta_0)$. For $i = 1, 2, \ldots$ the computational cost associated with ϑ_i is the cost of computing $A(\vartheta_{i-1})$ (in order for the process to move to ϑ_i), evaluating $|J_{A^{-1}}(\vartheta_i)|$ and evaluating π_θ at ϑ_i. Sections 6.3, 6.4 and 6.5 consider transformations for which $|J_{A^{-1}}(\vartheta)|$ equals a particular constant for all $\vartheta \in \Theta$. Thus in these special cases, evaluating $|J_{A^{-1}}(\vartheta)|$ has negligible cost and can be ignored.

Constraints on the regeneration measure

Measure $C\mu_\theta$ must be chosen so that $\kappa(\vartheta) > 0, \forall \vartheta \in \Theta$. This is equivalent to the condition:

$$\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| - \pi_\theta(\vartheta) + C\mu_\theta(\vartheta) \geq 0, \forall \vartheta \in \Theta. \tag{6.6}$$

Since in general there are no constraints on A, the term $\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|$ may be arbitrarily small so is of no help in lessening the constraints on measure $C\mu_\theta$. The best we can do is to find $C\mu_\theta$ such that

$$C\mu_\theta(\vartheta) - \pi_\theta(\vartheta) \geq 0, \forall \vartheta \in \Theta.$$

This condition is equivalent to $\pi_\theta(\vartheta)/C\mu_\theta(\vartheta) \leq 1$, which is the condition that applies to rejection sampling when using some proposal distribution μ_θ to simulate from π_θ.

Increasing the regeneration constant on-the-fly

Recall that $\kappa_{C\mu}$ is used to denote the full regeneration rate when the regeneration measure is $C\mu$. Namely, $\forall x \in \mathcal{X}, \kappa_{C\mu}(x) = \tilde{\kappa}(x) + C\mu(x)/\pi(x)$. When simulating the enriched process under measure $C\mu_\theta$, upon finding a state ϑ such that $\kappa_{C\mu}(\vartheta) < 0$, the method of Section 5.1 is used. That is, the process is restarted (any existing output is discarded) using a regeneration constant $C' > C$ such that $\kappa_{C'\mu}(\vartheta) = \underline{\kappa}$, for $\underline{\kappa} > 0$ a small constant such as $\underline{\kappa} = 0.01$. As long as μ_θ has heavier tails than μ_θ^+ (the minimal regeneration distribution, see Section 4.4), this method will eventually find a constant C such that $\kappa_{C\mu}(\vartheta) \geq 0, \forall \vartheta \in \Theta$. It is possible to choose $\mu_\theta \equiv \mu_X \mu_V$ and $\mu_V \equiv \pi_V$. Indeed, it may at first seem like a good idea to set $\mu_V \equiv \pi_V$ since then V regenerates from the target distribution. Condition $\pi_\theta(\vartheta)/C\mu_\theta(\vartheta) \leq 1$ simplifies to $\pi_X(x)/C\mu_X(x) \leq 1$. However, this can cause slow adaption of constant C. It may be that $\kappa(\vartheta) < 0$ for values of $\vartheta = (x, v)$ for which v is in the tails of π_V. Since the tails of π_V are reached infrequently, adaption of C takes a long time. See subsection 6.4.4 for an example of this.

6.2.2 Weighted Holding Rate

Assuming π is normalised, setting $C = 1$ and defining

$$\lambda(\vartheta) = \frac{\mu_\theta(\vartheta)}{\pi_\theta(\vartheta)}, \tag{6.7}$$

equation (6.2) reduces to:

$$\kappa(\vartheta) = \frac{\mu_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|}{\pi_\theta(\vartheta)}. \tag{6.8}$$

If we only have access to the unnormalised density $\tilde{\pi}_\theta$, setting

$$\lambda(\vartheta) = \frac{\mu_\theta(\vartheta)}{\tilde{\pi}_\theta(\vartheta)}$$

and $\tilde{C} = 1$ (equivalent to implicitly setting $C = 1/Z$) gives:

$$\kappa(\vartheta) = \frac{\mu_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|}{\tilde{\pi}_\theta(\vartheta)}.$$

Using a weighted holding rate has the advantage of guaranteeing that the regeneration rate is non-negative everywhere for *any* choice of regeneration distribution. That is, there are no restrictions on μ_θ.

When $\mu_\theta \equiv \mu_X \mu_V$ and $\mu_V \equiv \pi_V$, these rates may be written as

$$\lambda(x, v) = \frac{\mu_X(x)}{\tilde{\pi}_X(x)},$$

$$\kappa(x, v) = \frac{\mu_X([A^{-1}(x, v)]_x)\mu_V([A^{-1}(x, v)]_v)|J_{A^{-1}}(x, v)|}{\tilde{\pi}_X(x)\pi_V(v)}.$$

Since $\tilde{C} = 1$, the normalizing constant Z may be estimated as T/n (see Section 4.5).

When the holding rate is weighted, there is no minimal regeneration distribution and hence no minimal regeneration rate. This is a consequence of the underlying process itself relying on μ, as well as the enriched process regenerating from μ. To see this, first notice that the partial regeneration rate is

$$\tilde{\kappa}(\vartheta) = \frac{\mu_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| - \mu_\theta(\vartheta)}{\pi_\theta(\vartheta)}.$$

Recall equation (4.13) for the density function of the minimal regeneration distribution. In terms of the random variable θ, this is:

$$\mu_\theta^+(\vartheta) = \frac{1}{C^+}[0 \vee -\tilde{\kappa}(\vartheta)]\pi_\theta(\vartheta).$$

The minimal regeneration measure would thus satisfy

$$C^+\mu_\theta^+(\vartheta) = \left[\mu_\theta^+(\vartheta) - \mu_\theta^+(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| \vee 0\right],$$

which may alternatively be written as

$$0 = \left[(1 - C^+)\mu_\theta^+(\vartheta) - \mu_\theta^+(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| \vee -C^+\mu_\theta^+(\vartheta)\right].$$

Consider a state ϑ_0 for which $\mu_\theta^+(\vartheta_0) > 0$. Then since $-C^+\mu_\theta^+(\vartheta_0) < 0$ it must hold that

$$0 = (1 - C^+)\mu_\theta^+(\vartheta_0) - \mu_\theta^+(A^{-1}(\vartheta_0))|J_{A^{-1}}(\vartheta_0)|.$$

We now show that a distribution μ_θ^+ satisfying the above equation does not exist, for the case where $|J_{A^{-1}}(\vartheta)| = J_0$ for all $\vartheta \in \Theta$, for $J_0 \geq 1$ some constant. This case is considered, because the specific transformations considered in the next sections all have constant Jacobian matrices greater than or equal to one. We do not show that a distribution μ_θ^+ satisfying the above equation does not exists for transformations A in general. Rearranging the equation, we find that

$$\frac{\mu_\theta^+(\vartheta_0)}{\mu_\theta^+(A^{-1}(\vartheta_0))} = \frac{|J_{A^{-1}}(\vartheta_0)|}{1 - C^+}.$$

This means that the constant C^+ must satisfy $C^+ < 1$, otherwise the ratio of densities on the left-hand side would be negative. This constant must also satisfy $C^+ > 0$, by definition of the Restore process. The ratio $\mu_\theta^+(\vartheta_0)/\mu_\theta^+(A^{-1}(\vartheta_0))$ is therefore positive, so the density μ_θ^+ increases without bound (along the trajectory defined by starting at ϑ_0 and repeatedly applying the transformation A) so μ_θ^+ is not a proper distribution.

Under a weighted holding rate, in comparison to a constant holding rate, the variance of the holding times of the underlying process increases. In addition, there can be a numerical issue if μ has heavier tails than π. When the process is moving out into the tails of π, sometimes λ and κ have similar large values. This results in inefficient computation, since sample paths will visit a sequence of states for a very short amount of time each.

Complexity

Again assume that the cost of evaluating μ_θ is negligible in comparison to that of evaluating π_θ. The computational cost associated with ϑ_0, the first state in the jump chain of a given tour, is the cost of evaluating $\pi_\theta(\vartheta_0)$, $|J_{A^{-1}}(\vartheta_0)|$ and computing $A^{-1}(\vartheta_0)$. For a subsequent state ϑ_i in the jump chain of the given tour, $i \in \{1, 2, \ldots\}$, the additional computational cost is that associated with computing the transformation $A(\vartheta_{i-1})$ in order to calculate ϑ_i, then the cost of evaluating $\pi_\theta(\vartheta_i)$ and $|J_{A^{-1}}(\vartheta_i)|$.

6.2.3 Summary

This section has considered two choices for the holding rate, both motivated by the requirement that κ is non-negative everywhere. A comparison of the two is given in the table below.

	Constant Holding Rate	Weighted Holding Rate
$\lambda(\vartheta)$	λ_0	$\mu(\vartheta)/\pi(\vartheta)$
Computational cost associated with ϑ_0, the first state in the jump chain of a given tour.	Evaluating $\pi_\theta(\vartheta_0)$ and $\lvert J_{A^{-1}}(\vartheta_0)\rvert$, computing $A^{-1}(\vartheta_0)$ then evaluating $\pi_\theta(A^{-1}(\vartheta_0))$.	Evaluating $\pi_\theta(\vartheta)$ and computing $A^{-1}(\vartheta_0)$.
Computational cost associated with ϑ_i for $i = 1, 2, \ldots$	Computing $\vartheta_i = A(\vartheta_{i-1})$ then evaluating $\lvert J_{A^{-1}}(\vartheta_i)\rvert$ and $\pi_\theta(\vartheta)$.	Computing $\vartheta_i = A(\vartheta_{i-1})$ then evaluating $\lvert J_{A^{-1}}(\vartheta_i)\rvert$ and $\pi_\theta(\vartheta)$.
Restriction on μ_θ.	Given by equation (6.6), which in practice reduces to $\pi_\theta(\vartheta)/C\mu_\theta(\vartheta) \leq 1$.	None.
μ_θ^+	Exists.	Does not exist.
Potential numerical issues.	None found.	The jump process may visit a sequence of states for a very short amount of time each.

It is not clear from this summary alone which holding rate should be preferred. We now consider three choices of local dynamics: Random-Direction, Hamiltonian and Conformal Hamiltonian. The experiments conducted for these choices of transformation will lead us to conclude that a Jumpar with constant holding rate in fact has the most potential to be an efficient sampler.

6.3 Random Direction Dynamics

Theorem 4 set out how to enrich a class of jump processes with regenerations so that the resulting jump process is π-invariant. The class of jump process that the theorem applies to have a jump chain transition kernel P corresponding to a bijective transformation A. In this section we consider a particular choice of transformation, denoted A_1, which corresponds to the velocity of the process remaining fixed and the position moving according to the velocity. In Sections 6.4 and 6.5 we consider other transformations, denoted A_2 and A_3 respectively, corresponding to Hamiltonian and Conformal Hamiltonian dynamics.

Consider enriching with regenerations a jump process $\{\Xi_t\}_{t\geq 0} = \{Y_t, W_t\}_{t\geq 0}$ with jump chain transition kernel corresponding to the mapping:

$$A_1(x, v) = (x + v, v). \tag{6.9}$$

Such a process is clearly not π-invariant: for $\Xi_0 = (Y_0, W_0) = (y_0, w_0)$, $Y_t \to \infty$ as $t \to \infty$ whilst $W_t = w_0$ $\forall t$. Mapping A_1 has inverse $A_1^{-1}(x, v) = (x - v, v)$ and $|J_{A^{-1}}(\vartheta)| = 1, \forall \vartheta \in \Theta$. We are therefore able to define a jump process $\{\theta_t\}_{t\geq 0}$ by enriching $\{\Xi_t\}_{t\geq 0}$ with regenerations so that $\{\theta_t\}_{t\geq 0}$ has invariant distribution corresponding to a target distribution π_θ. We call such a process a Random Direction Jumpar (RD-Jumpar).

Complexity

The computational cost of the transformation and its inverse is negligible compared to that of evaluating π_θ and can be ignored. For a constant holding rate, the cost of the first state in the jump chain of a tour is two evaluations of π_θ. Subsequent states cost one evaluation of π_θ. For a weighted holding rate, the cost of generating a jump chain of length n is n evaluations of π_θ.

Special Case: Simplified Regeneration Rate

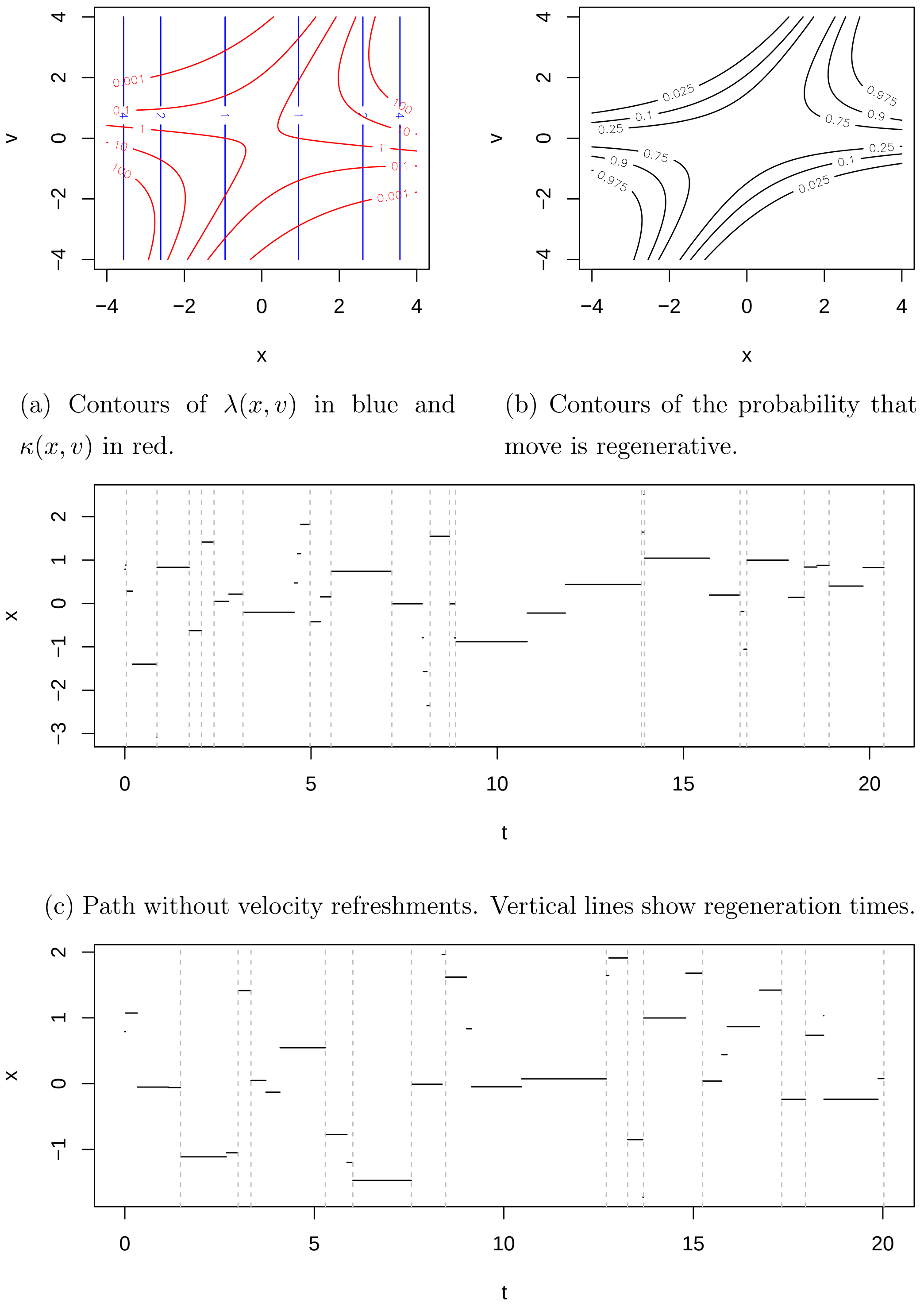
Suppose the regeneration distribution is chosen to be separable: $\mu_{X,V}(x, v) = \mu_X(x)\mu_V(v)$. When a weighted holding rate is used and we choose $\mu_V \equiv \pi_V$, then $\lambda(x, v) = \mu_X(x)/\pi_X(x)$ and

$$\kappa(x, v) = \frac{\mu_X(x - v)}{\pi_X(x)}.$$

When the holding rate is non-constant, generating a component of the jump chain requires evaluating the target density just once.

Figure 6.1 shows an example of when $\pi_X \equiv \mathcal{N}(0, 0.81)$ and μ_X, μ_V, π_V are standard Normal distributions. The figures show contours of the holding rate, regeneration rate and the probability that a move is regenerative, as well as sample paths of processes with and without velocity refreshment.

Random-Direction dynamics are useful for demonstrating the combination of local and global moves. However, for high-dimensional target distributions, these dynamics will be ineffective, since it is likely that the process will move away from the region where most of the mass of π is concentrated.



(a) Contours of $\lambda(x, v)$ in blue and $\kappa(x, v)$ in red.

(b) Contours of the probability that a move is regenerative.

(c) Path without velocity refreshments. Vertical lines show regeneration times.

(d) Path with velocity refreshments. Vertical lines show regeneration times.

Figure 6.1: Plots associated with a Random-Direction Jump Process Adjusted with Regenerations with weighted holding rate, $\pi_X \equiv \mathcal{N}(0, 0.81)$ and $\mu_X \equiv \mu_V \equiv \pi_V \equiv \mathcal{N}(0, 1)$.

6.4 Hamiltonian Dynamics

It may be possible to improve on the performance of RD-Jumpar by using a more sophisticated transformation. This section introduces the *Hamiltonian Jump Process Adjusted with Regenerations* (H-Jumpar). This jump process is defined by enriching an underlying jump process, which moves according to unadjusted *Hamiltonian dynamics*, with regenerations so that the process is π-invariant. This section first presents Hamiltonian dynamics, including Hamilton's equations of motion and their analytic integration. Next, it is shown how approximate Hamiltonian dynamics may be simulated by numerically integrating the equations of motion using the *Leapfrog method*, which may be represented as a transformation. Next Hamiltonian Monte Carlo is covered (Duane et al., 1987; Neal et al., 2011), an algorithm that uses Hamiltonian dynamics. Finally, the Hamiltonian Jumpar is introduced in full and experiments presented.

6.4.1 Hamilton's Equations of Motion

Hamilton's equations of motion describe how a particle moves so that its total energy is conserved. Let the particle be $\vartheta = (x, v)$, with position $x \in \mathcal{X}$, velocity $v \in \mathcal{V}$, potential energy $U(x)$ and kinetic energy $K(v)$. The Hamiltonian is the total energy:

$$H(\vartheta) = H(x, v) = U(x) + K(v).$$

The Hamiltonian is conserved when the particle moves according to Hamilton's equations of motion:

$$\begin{aligned}\frac{dx}{dt} &= \nabla_v H(x, v),\\ \frac{dv}{dt} &= -\nabla_x H(x, v).\end{aligned}$$

A particle moving with Hamiltonian dynamics in one-dimensional position space may be visualised as a frictionless puck moving over a smooth, curved surface. When the puck goes uphill it gains potential energy and loses kinetic energy (hence it slows down) and vice versa when going downhill.

Given a particle's position and velocity at time 0, their values at time I can be found by *jointly* integrating the equations of motion:

$$\begin{aligned}x(I) &= \int_0^I \nabla_v H(x(s), v(s))ds + x(0),\\ v(I) &= \int_0^I -\nabla_x H(x(s), v(s))ds + v(0).\end{aligned}$$

Hamiltonian dynamics are used in Molecular Dynamics simulation (Alder and Wainwright, 1959). Although it is sometimes possible to integrate Hamilton's equations analytically (Pakman and Paninski, 2014), in most cases it is necessary to use a numerical integrator (Leimkuhler and Reich, 2004).

Numerical Integration of Hamilton's Equations of Motion

A numerical integrator approximates the trajectory of a particle moving according to Hamiltonian dynamics by iteratively computing its position and velocity at a discrete mesh of times. The leapfrog method is most widely used because it is computationally inexpensive whilst remaining adequately accurate in most applications. The technique splits the *integration time* I into a sequence of L equally spaced points, with spacing ϵ, so $I = L\epsilon$. Integer L is called the *number of leapfrog steps* and ϵ is called the *leapfrog step-size.* The leapfrog method alternates between updates of v and x. Starting at $s = 0$, first the velocity is update with a half-step and the position by a full step:

$$v\left(\frac{\epsilon}{2}\right) = v(0) - \frac{\epsilon}{2}\nabla_x U\left(x(t)\right), \tag{6.10}$$

$$x(\epsilon) = x(0) + \epsilon\nabla_v K\left(v\left(\frac{\epsilon}{2}\right)\right). \tag{6.11}$$

Next, for $s = \epsilon, 2\epsilon, \ldots, (L-1)\epsilon$, full-step updates of v and x are made:

$$v\left(s + \frac{\epsilon}{2}\right) = v\left(s - \frac{\epsilon}{2}\right) - \epsilon\nabla_x U(x(s)), \tag{6.12}$$

$$x(s+\epsilon) = x(s) + \epsilon\nabla_v K\left(v\left(s + \frac{\epsilon}{2}\right)\right). \tag{6.13}$$

Finally, to complete the numerical integration, a half-step update of v is performed

$$v(L\epsilon) = v\left(\left(L - \frac{1}{2}\right)\epsilon\right) - \frac{\epsilon}{2}\nabla_x U(x(L\epsilon)). \tag{6.14}$$

The method gets its name from the way the updates of the velocity and position "leap" over each other. Integrating Hamilton's equations of motion this way, the Hamiltonian is only conserved approximately.

Representation as a Transformation

The leapfrog method may be represented as an invertible transformation with unit Jacobian. To see this, note that for two invertible and differentiable transformations A_i and A_j, their composition $A_j \circ A_i$ is also invertible and differentiable, for example see Section 2.1 of Papamakarios et al. (2021). In particular:

$$\begin{aligned}(A_j \circ A_i)^{-1} &= A_i^{-1} \circ A_j^{-1}, \\ |J_{A_j \circ A_i}(\theta)| &= |J_{A_j}(\theta)| \cdot |J_{A_i}(\theta)|.\end{aligned}$$

Each of equations (6.10)-(6.14) are invertible and have unit Jacobian, therefore their composition has unit Jacobian. Recalling that A_1 represents the transformation (6.9), let the leapfrog transformation be

$$A_2 : \Theta \to \Theta.$$

6.4.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) uses Hamiltonian dynamics as a mechanism for proposing moves in a Metropolis-Hastings algorithm (Duane et al., 1987; Neal, 1996a; Neal et al., 2011). HMC operates on an augmented space, simulating a Markov chain with limiting distribution $\pi_\Theta \equiv \pi_X \pi_V$, with X the random variable of interest and V an auxilliary velocity variable. An isotropic Gaussian is usually chosen for π_V.

HMC alternates between making a Gibbs update of the velocity by resampling from π_V, then making a Metropolis-Hastings update of $\theta = (X, V)$. Hamiltonian dynamics are used to propose a new state. Let $F : \Theta \to \Theta$ be the transformation that negates the velocity:

$$F(X, V) = (X, -V).$$

The proposal distribution q used for the Metropolis-Hastings step is equivalent to applying $F \circ A_2$. To apply the transformation A_2, the Hamiltonian is defined in terms of the energies U and K corresponding to distributions π_X and π_V. Once the leapfrog transformation has been made, the velocity is "flipped" using F so that the proposal distribution is symmetrical: $q(\vartheta, \vartheta') = q(\vartheta', \vartheta); \forall \vartheta, \vartheta' \in \Theta$. For ϑ the current state and ϑ' the proposed state, the acceptance probability given by equation (2.8) then simplifies to

$$\alpha(\vartheta, \vartheta') = \frac{\pi_\theta(\vartheta')}{\pi_\theta(\vartheta)}.$$

If the flip transformation F is not applied, then the proposed state would be $A_2(\vartheta)$ and the acceptance probability would be, for all ϑ:

$$\alpha(\vartheta, A_2(\vartheta)) = \frac{\pi_\Theta(A_2(\vartheta))q(A_2(\vartheta), \vartheta)}{\pi_\Theta(\vartheta)q(\vartheta, A_2(\vartheta))} = \frac{\pi_\Theta(A_2(\vartheta)) \times 0}{\pi_\Theta(\vartheta)q(\vartheta, A_2(\vartheta))} = 0.$$

If the Jacobian of the transformation were not unit, it would need to be accounted for in the acceptance probability. HMC is described in full in Algorithm 14.

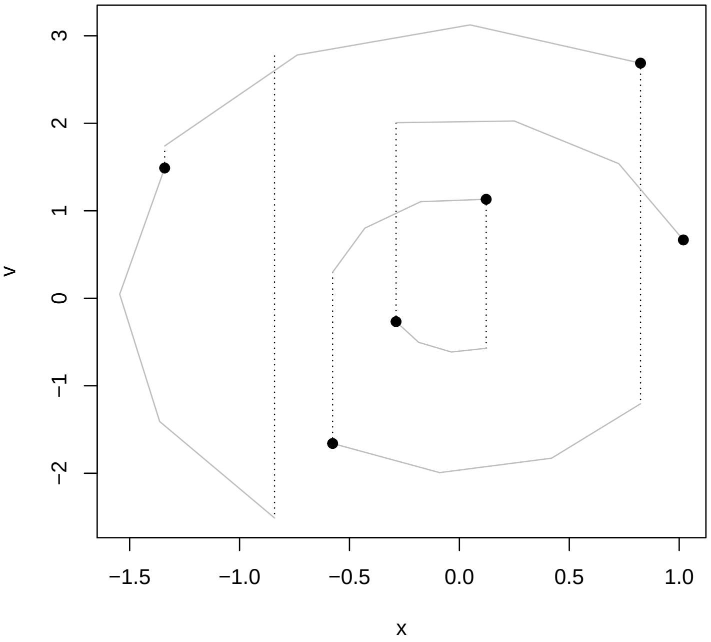
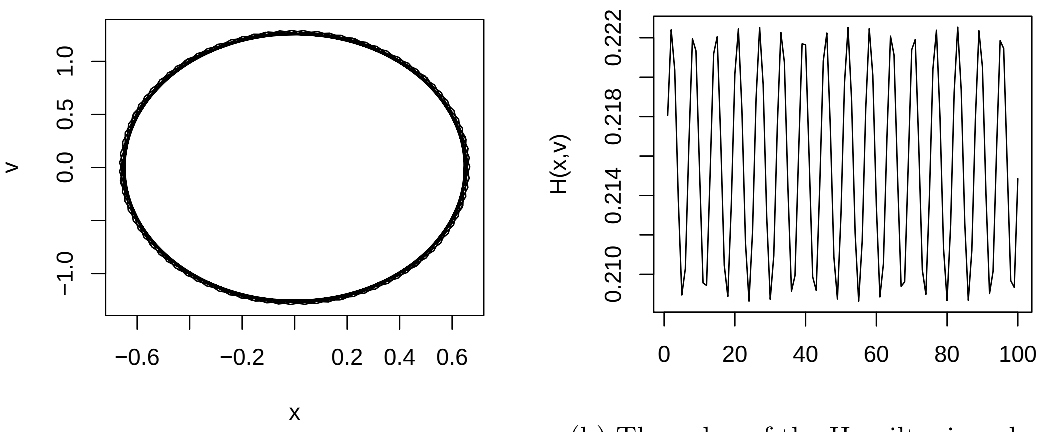


Figure 6.2: Dots show the first 6 samples from $\pi_X \equiv \mathcal{N}(0,1)$ obtained with HMC sampling. Dotted lines show the resampling of the velocity v at the start of each trajectory from $\pi_V \equiv \mathcal{N}(0,4)$. Gray lines trace the $L = 3$ leapfrog steps of size $\epsilon = 1$ taken to propose each new state of the chain. The approximate Hamiltonian dynamics result in roughly elliptical paths that remain in level sets of the augmented distribution.

Algorithm 14: Hamiltonian Monte Carlo

```
for $i$ in $0$ to $n-1$ do
    $V \sim \pi_V$
    $(X^*, V^*) \leftarrow A_2(X, V)$
    $V^* \leftarrow -V^*$
    $\alpha \leftarrow 1 \wedge \exp\{H(X,V) - H(X^*,V^*)\}$
    $U \sim \mathcal{U}(0,1)$
    if $U < \alpha$ then
        $(X,V) \leftarrow (X^*,V^*)$
    end
    Record $X, V$
end
```

Figure 6.2 illustrates the dynamics of a Markov chain generated using HMC when $\pi_X \equiv \mathcal{N}(0,1), \pi_V \equiv \mathcal{N}(0,4), \epsilon = 1$ and $L = 3$. Since the sampler approximately conserves the Hamiltonian, the leapfrog steps approximately remain in the same level sets of the joint distribution of (X, V).



(a) Trajectory consisting of 100 leapfrog steps.

(b) The value of the Hamiltonian along the trajectory. The number of Leapfrog steps is given by the x-axis.

Figure 6.3: A trajectory with stable Hamiltonian, consisting of 100 leapfrog updates with $U(x) = x^2/2$ and $K(v) = v^2/8$ The Hamiltonian is not conserved exactly but oscillates around its starting value.

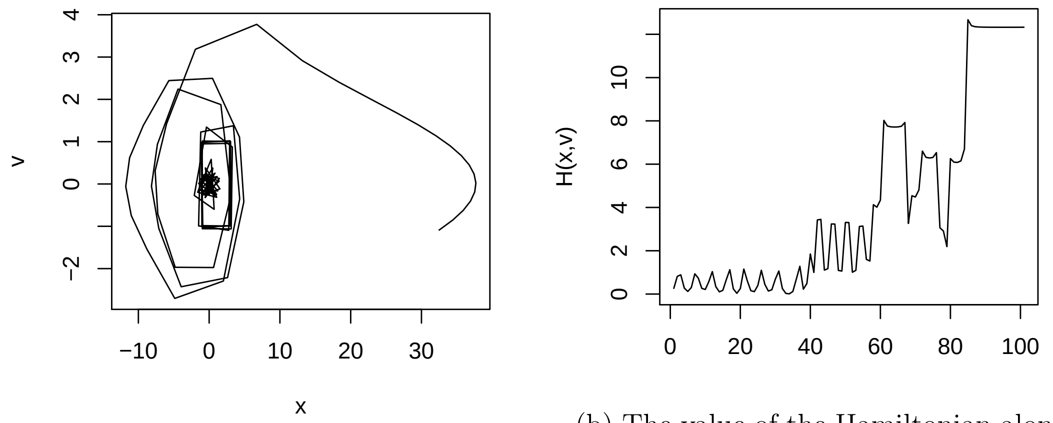
Tuning parameters

The leapfrog method does not conserve the Hamiltonian exactly. However, when ϵ is sufficiently small, the Hamiltonian will oscillate around its initial value, as in Figure 6.3. If the Hamiltonian doesn't deviate too far from its initial value, proposal moves are likely to be accepted.

An analytic study of the scaling of HMC (Beskos et al., 2013) finds that, in order for the acceptance probability not to decay as the dimension d goes to infinity, ϵ must be $\mathcal{O}(d^{-1/4})$ and thus $\mathcal{O}(d^{1/4})$ steps are needed to traverse the space. The study finds that the asymptotically optimal acceptance rate, to three decimal places, is 0.651. This result agrees with empirical findings, that an acceptance rate of 0.7 is roughly optimal (Chen et al., 2000).

Despite this result, tuning ϵ and L is still difficult. The step-size should be large, though the accuracy of the approximation should be retained. Figure 6.4 illustrates an instance where the step-size is too large ($\epsilon = 2$ in this case), which results in a very bad approximation of Hamiltonian dynamics – instead of remaining close to the level set corresponding to the initial value of the Hamiltonian, the path spirals outwards and the Hamiltonian changes significantly.

Once a stable step-size has been found, an appropriate L may then be chosen. A useful heuristic is that the number of leapfrog steps should be chosen so that the Markov chain has small lag-1 autocorrelation. Strong correlations between consecutive states usually indicate that the particle has not moved far away



(a) Hamiltonian trajectory consisting of 100 leapfrog steps.

(b) The value of the Hamiltonian along the trajectory. The number of Leapfrog steps is given by the x-axis.

Figure 6.4: Unstable Hamiltonian dynamics resulting from a stepsize $\epsilon = 2$ which is too large. Here, the target is a t-distribution with 3 degrees of freedom, augmented with a standard normal distribution for the velocity variable. The trajectory spirals outwards and the Hamiltonian increases.

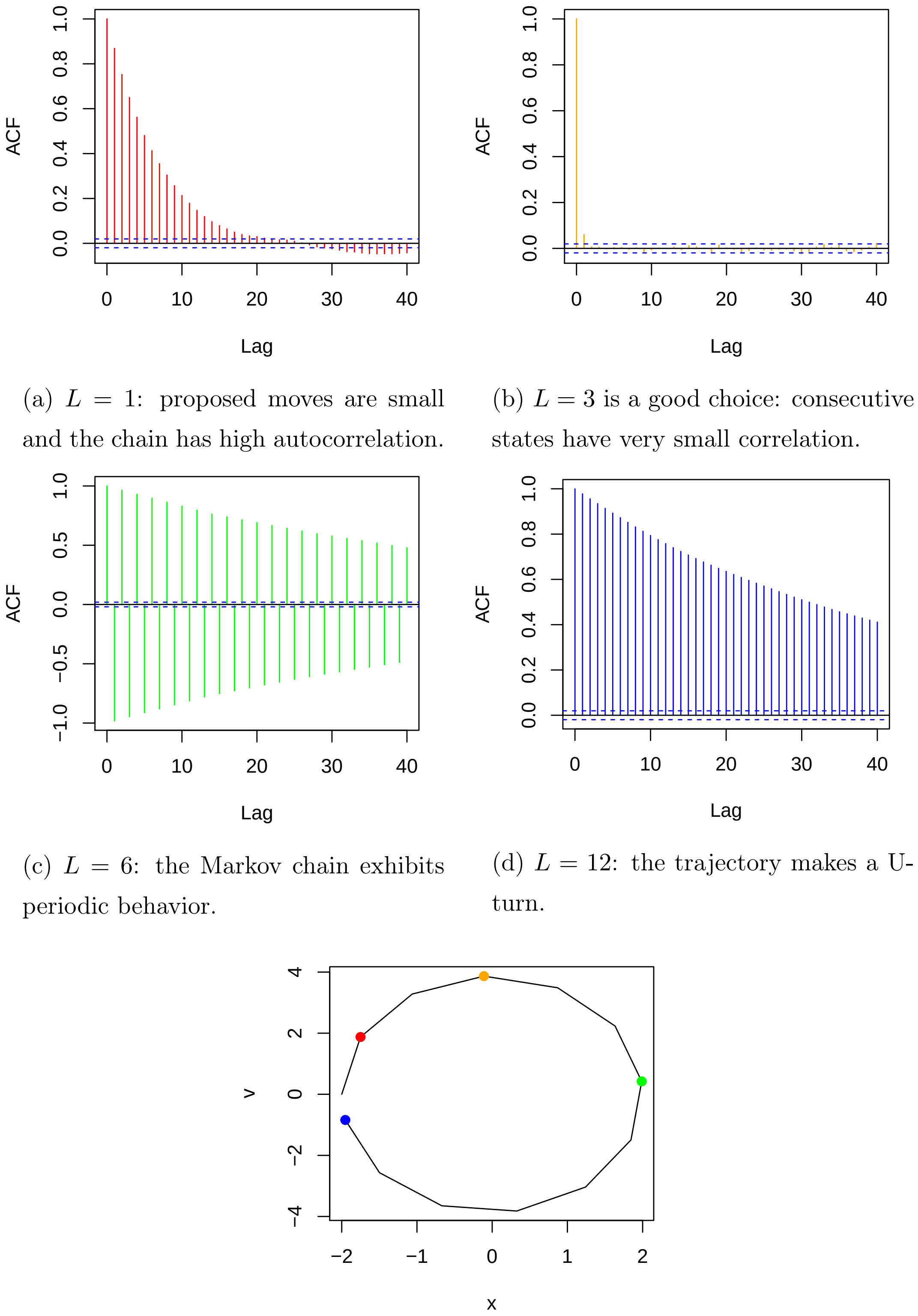
enough from its start point. However, large autocorrelation may indicate that the integration time is in fact too large and that a long Hamiltonian trajectory has ended close to where it began. Negative autocorrelation may indicate periodic behavior: the chain may be moving back and forth from one side of phase space to the other.

Figure 6.5 illustrates how small changes in L can result in vastly different autocorrelations in the chain. Tuning HMC only becomes more difficult in multiple dimensions, where a good choice of ϵ and L can vary between dimensions.

Difficulties in tuning HMC inspired the development of the No-U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014) and its efficient implementation in the STAN programming language (Carpenter et al., 2017). NUTS handles the tuning problem by automating the choice of ϵ and adaptively setting L in order to avoid the sampler turning back on itself, all whilst retaining reversibility.

Scaling and Efficiency

The *typical set* of a probability distribution is the region of state space containing a significant amount of probability mass. The mode has high probability density, but in high dimensions has very small volume (a feature of high-dimensional spaces). Thus, counter-intuitively, in high-dimensions the mode does not contain a large amount of probability mass and is not in the typical set. By contrast,



(a) $L = 1$: proposed moves are small and the chain has high autocorrelation.

(b) $L = 3$ is a good choice: consecutive states have very small correlation.

(c) $L = 6$: the Markov chain exhibits periodic behavior.

(d) $L = 12$: the trajectory makes a U-turn.

(e) A trajectory started at $\vartheta = (-2, 0)$, with the state after $L = 1, 3, 6, 12$ steps shown in red, orange, green and blue respectively.

Figure 6.5: Autocorrelation plots for Markov chains simulated with HMC using $L = 1, 3, 6, 12$ and a trajectory with approximate Hamiltonian dynamics.

regions far from the mode may have a large volume, but have very small probability density, so also don't contain significant probability mass. The typical set lies between these extremes and tends to concentrate as the dimension increases. Generally put, the performance of a MCMC algorithm depends on how well the sampler explores the typical set. HMC performs well because proposed moves, driven by Hamiltonian dynamics, tend to remain within the level sets of the augmented target distribution. Thus the sampler will efficiently explore the typical set Betancourt (2017).

6.4.3 Hamiltonian Jump Process Adjusted with Regenerations

Let $\{\Xi_t\}_{t\geq 0} = \{Y_t, W_t\}_{t\geq 0}$ be a jump process with holding rate $\lambda : \Theta \to \Theta$ and jump chain defined by a Markov transition kernel corresponding to the leapfrog transformation A_2 as defined in subsection 6.4.1. Then $\{\Xi_t\}_{t\geq 0}$ may be enriched with regenerations, as in Theorem 4 so that $\{\theta_t\}_{t\geq 0}$, the enriched process, has invariant distribution π_θ. Such a process is called a Hamiltonian Jump Process Adjusted with Regenerations (H-Jumpar).

Complexity

For a tour beginning with state ϑ_0, one must compute $A^{-1}(\vartheta_0)$. Recall that Hamiltonian dynamics are simulated by numerically integrating Hamilton's equations of motion over an auxiliary time parameter s. To compute $A^{-1}(\vartheta_0)$, we integrate these equations backwards in time using L leapfrog steps of size ϵ. This requires making $L+1$ evaluations of ∇U at times $s = 0, -\epsilon, \dots, -L\epsilon$. Thus ϑ_0 has an associated computational cost of $L+1$ evaluations of ∇U plus two evaluations of U if λ is constant or one evaluation of U if λ is weighted.

For ϑ_i a subsequent state in the tour ($i = 1, 2, \dots$) the associated cost is that involved with computing $A(\vartheta_{i-1})$ and in evaluating $\pi(\vartheta_i)$. The gradient ∇U will already have been evaluated at ϑ_{i-1}, so only L additional evaluations of ∇U are needed. Thus ϑ_i for $i = 1, 2, \dots$ has an associated cost of L evaluations of ∇U plus one evaluation of U (whether the rate is constant or weighted).

6.4.4 Illustration of Dynamics: Normal distribution

As an example of using Hamiltonian dynamics, we consider using a Hamiltonian Jumpar to sample a normal distribution. Recall from subsection 6.2.1 that when

$\lambda(\vartheta) \equiv 1$, the regeneration measure is constrained, in that it must satisfy equation (6.6):

$$\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)| - \pi_\theta(\vartheta) + C\mu_\theta(\vartheta) \geq 0, \forall \vartheta \in \Theta.$$

In the general case, when A is any class of bijection, the term $\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|$ may be arbitrarily small and thus the condition effectively reduces to

$$C\mu_\theta(\vartheta) - \pi_\theta(\vartheta) \geq 0, \forall \vartheta \in \Theta.$$

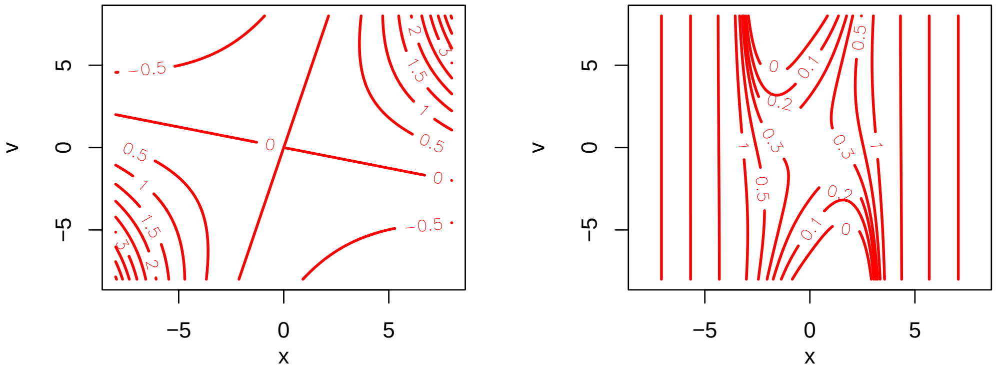
Approximate Hamiltonian dynamics in particular are liable to "explode", meaning that the particle represented by the dynamics spirals out towards infinity. This may happen when the target distribution has very steep gradients in parts of the state space, or if the leapfrog step-size is too large. An example of unstable approximate Hamiltonian dynamics are illustrated in Figure 6.4. Therefore when A corresponds to approximate Hamiltonian dynamics, it is particularly true that in practice the term $\pi_\theta(A^{-1}(\vartheta))|J_{A^{-1}}(\vartheta)|$ in equation (6.6) does nothing to lessen the constraint on μ_θ.

For this reason, for a constant holding rate we recommend setting μ_θ as a t-distribution, since this distribution has heavy tails. In the example below, we consider sampling from a standard normal target distribution when the regeneration distribution is a t-distribution.

By contrast, when the holding rate is weighted, there are no constraints on the regeneration distribution. We then recommend setting the regeneration distribution to be an isotropic Gaussian distribution (assuming the target distribution has undergone a pre-transformation so that it is roughly zero-centred with identity covariance matrix). In our experiment, we therefore consider a Gaussian target distribution, with variance close but not equal to 1.

Unit Holding Rate

First we demonstrate that setting $\mu_V \equiv \pi_V$ and using the strategy of adapting C of subsection 6.2.1, it is difficult to find $C > 0$ such that $\kappa(\vartheta) \geq 0, \forall \vartheta \in \Theta$. Let $\mu_X \equiv t_\nu$ with $\nu = 10$. Under regeneration constant C_i, we simulate $\{\Xi_t\}_{t \geq 0}$ using $\epsilon = 0.5$ and $L = 1$ until either 10^5 tours are completed or a state $\vartheta' \in \Theta$ such that $\kappa_{C_i \mu_\theta}(\vartheta') < 0$ is found. In the latter case, we discard all output and restart simulation of the process using C_{i+1} such that $\kappa_{C_{i+1}\mu_\theta}(\vartheta') = \underline{\kappa} = 0.01$. Starting with $C_0 = 0$, the value of C that resulted in 10^5 tours of the process being completed, without encountering a state with negative regeneration rate, was $C = 0.281$ (3.s.f). However, this value of C is not in fact large enough.



(a) Contours of $\tilde{\kappa}(x, v)$.

(b) Contours of $\kappa(x, v)$.

Figure 6.6: Contours of $\tilde{\kappa}$ and κ for $\pi_X \equiv \pi_V \equiv \mu_V \equiv \mathcal{N}(0, 1)$ and $\mu_X \equiv t_\nu$ with $\nu = 10$. The parameters used were $\epsilon = 0.5$ and $L = 1$. The regeneration measure does not result in κ being non-negative everywhere.

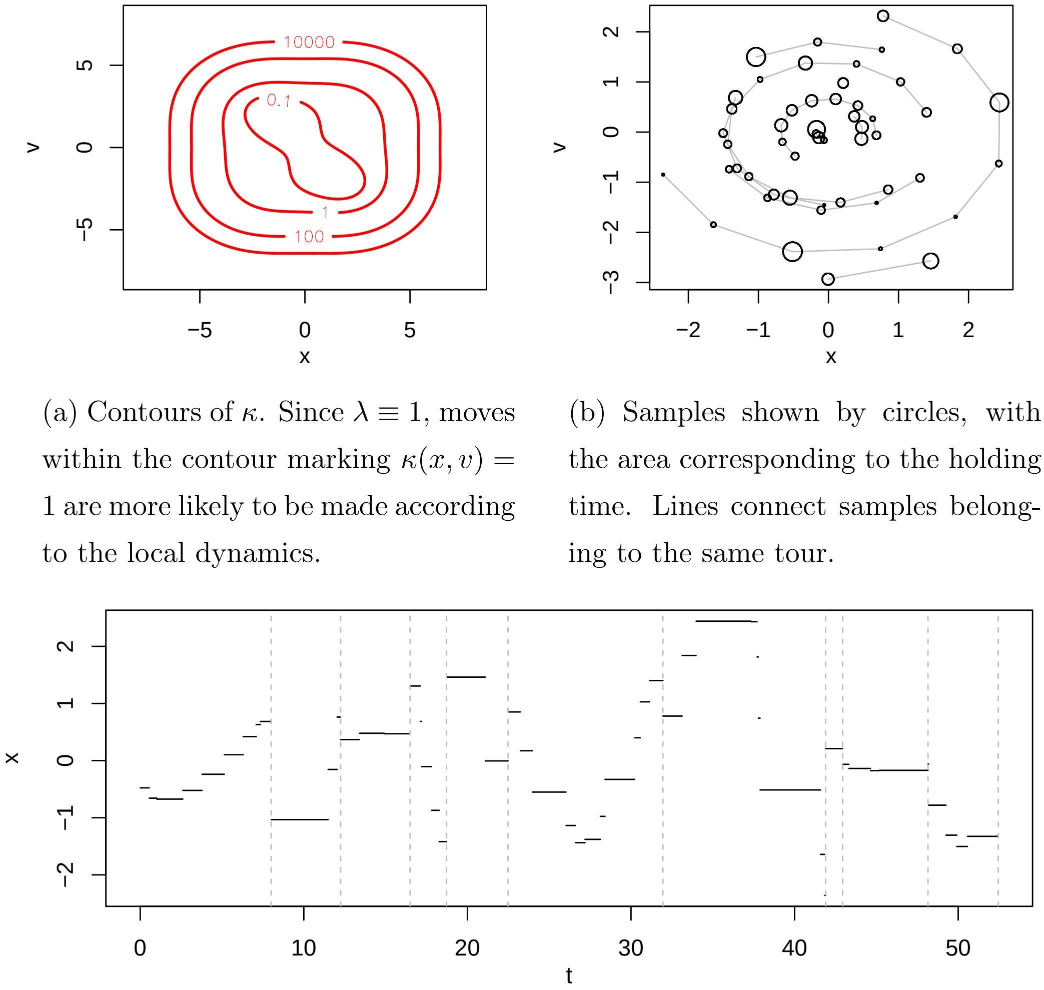
Figures 6.6a and 6.6b show contour plots of $\tilde{\kappa}$ and κ for this example. For the value of C computed, κ is still negative in parts of the space.

When C is not large enough for the regeneration rate to be non-negative everywhere, the invariant distribution of the process is biased. For example, suppose we generate 200 estimates of the second moment of π_X and that random variable N is the number of estimates greater than the exact second moment, which is 1. For an unbiased process, $\mathbb{E}[N] = 0.5 \times 200 = 100$ and $P[N > 117] = 0.00657$ (3.s.f). Using the procedure above, the experiment produced $N = 118$. This is because in many trials, C was not large enough for the regeneration rate to be non-negative everywhere. However, the bias introduced was small: the average estimate of the second moment was 1.000931.

By contrast, taking $\mu_V \equiv t_\nu$ with $\nu = 10$, a constant C is found for which the regeneration rate is valid. The process is more likely to make a local move when $\kappa(\vartheta) < 1$; in this example, this occurs near the centre of the space. Contours of $\kappa_{C\mu_\theta}$ are shown in 6.7a. The first 10 tours of a sample path are shown in Figures 6.7b and 6.7c. The first shows the weighted jump chain, with samples in the same tour linked by gray lines. The second shows the x-marginal of the process, with dashed vertical lines showing regeneration times.

Weighted Holding Rate

Figures 6.8 and 6.9 relate to an example where $\pi_X \equiv \mathcal{N}(0, 0.81)$, $\mu_X \equiv \mu_V \equiv \pi_V \equiv \mathcal{N}(0, 1)$ and a non-constant holding rate is used to guarantee non-negativity of κ. Parameters $\epsilon = 0.8$ and $L = 1$ were used in the Leapfrog method. Figure 6.8a shows the holding rate in blue and the regeneration rate in red. The holding

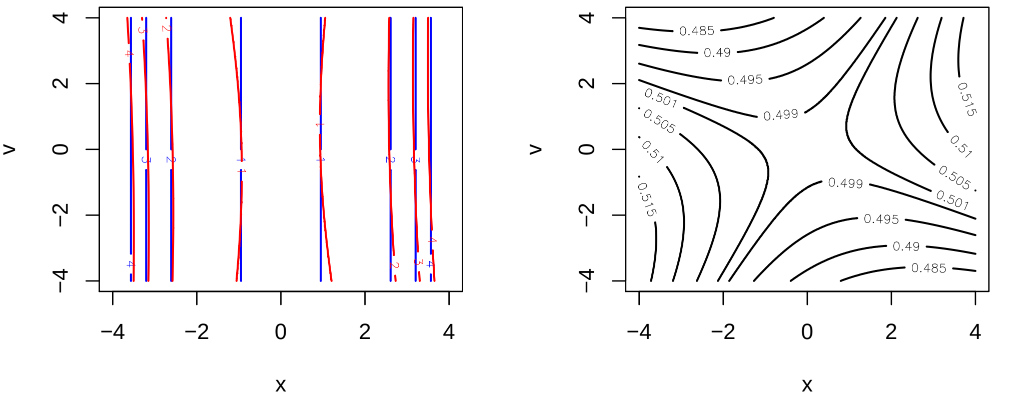


(a) Contours of κ. Since $\lambda \equiv 1$, moves within the contour marking $\kappa(x, v) = 1$ are more likely to be made according to the local dynamics.

(b) Samples shown by circles, with the area corresponding to the holding time. Lines connect samples belonging to the same tour.

(c) The x-marginal of the jump process. Vertical lines show regeneration times.

Figure 6.7: Hamiltonian Jump Process Adjusted with Regenerations with unit holding rate, $\pi_X \equiv \pi_V \equiv \mathcal{N}(0, 1)$ and $\mu_X \equiv \mu_V \equiv t_\nu$.



(a) Contours of $\lambda(x, v)$ in blue and $\kappa(x, v)$ in red.

(b) Contours of the probability of that a move is regenerative.

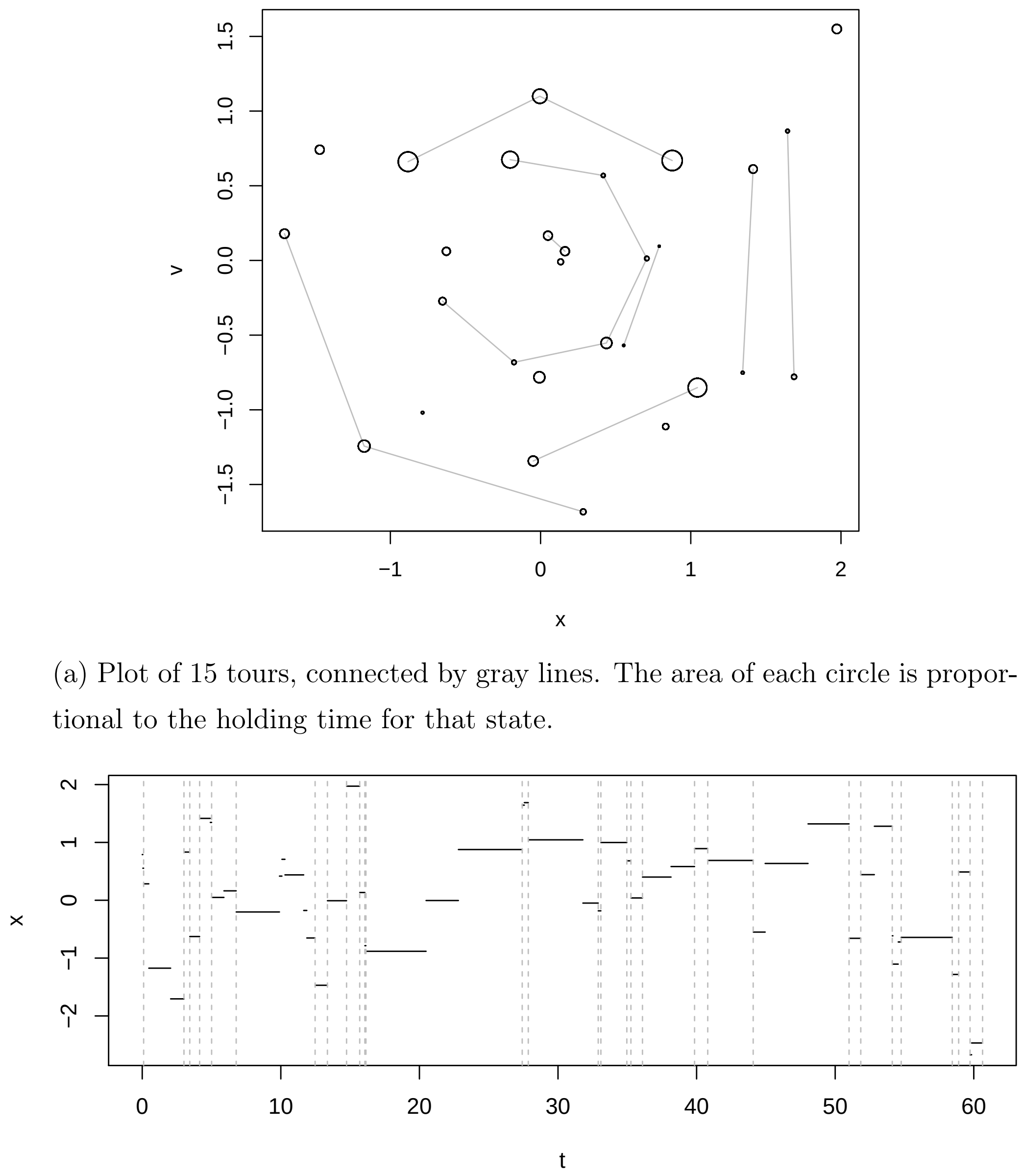
Figure 6.8: Contours of the holding and regeneration rates for a Hamiltonian Jumpar with weighted holding rate, as well as contours of the probability that a move is regenerative. Here, $\pi_X \equiv \mathcal{N}(0, 0.81)$ and $\mu_X \equiv \mu_V \equiv \pi_V \equiv \mathcal{N}(0, 1)$.

and regeneration rates are similar because the Hamiltonian corresponding to π_θ is approximately conserved by the leapfrog integrator and μ_X is close to π_X. Figure 6.8b shows contours of the probability that a move is regenerative. The closeness of the regeneration and holding rates means the probability that a move is regenerative is close to half in most of the space where π_θ has significant mass.

Figure 6.9 shows the dynamics of the process for the simple example. In Figure 6.9a, states belonging to the same tour are connected by gray lines and the area of each circle is proportional to the length of the holding time for that state. The figure illustrates how sometimes multiple local moves occur successively. On the other hand, sometimes the process regenerates into a state, then regenerates again before making a local move. Figure 6.9b shows the x-marginal of the jump process.

6.4.5 Examples

A difficult aspect of HMC is tuning the leapfrog step-size and the number of leapfrog steps. Indeed, a significant research effort has been dedicated to this challenge. An interesting aspect of Hamiltonian Jumpar is that it allows for the automatic and random selection of the number of steps. When $L = 1$ and there are no velocity refreshments, at each state entered into by the jump process, either a local move is made so that the process continues along the same Hamiltonian trajectory (increasing the integration time), or the process regenerates and the



(a) Plot of 15 tours, connected by gray lines. The area of each circle is proportional to the holding time for that state.

(b) The x-marginal of a sample path of a Hamiltonian Jumpar. Vertical lines mark regeneration times.

Figure 6.9: Hamiltonian Jump Process Adjusted with Regenerations, with weighted holding rate, $\pi_X \equiv \mathcal{N}(0, 0.81), \mu_X \equiv \mu_V \equiv \pi_V \equiv \mathcal{N}(0, 1)$ and $\epsilon = 0.8, L = 1$.

Hamiltonian trajectory comes to an end.

This section investigates the use of a Hamiltonian Jumpar for sampling from the posterior distributions of models with various characteristics. First, a model of Dugongs (sea cows) is considered, which has $d = 3$ and is close to Gaussian in shape. Second, a bivariate Banana distribution (defined in subsection 4.4.1) displaying strongly correlated variables is sampled. Lastly a bimodal distribution is considered.

Dugong Model

We investigated the use of a Hamiltonian Jumpar for sampling from the posterior distribution for a model of Dugongs (sea cows) (Gilks et al., 1998; Carlin and Gelfand, 1991; Ratkowsky and Marcel Dekker, 1983). For this model, we will denote the unknown parameters as α, β, γ; the predictors $\{x_i\}_{i=1}^n$ and the response $\{y_i\}_{i=1}^n$. The model states that the length y_i of Dugong i given its age x_i has likelihood:

$$y_i \sim \mathcal{N}(\alpha - \beta\gamma^{x_i}, \sigma^2).$$

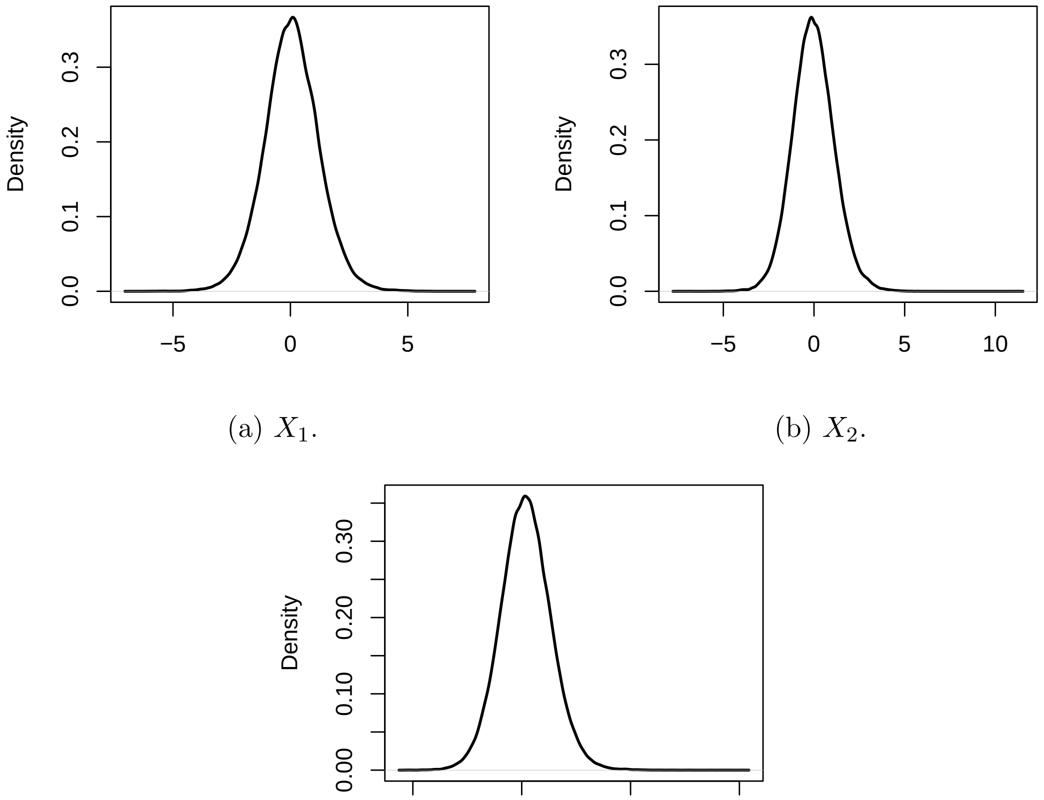
The unknown parameters satisfy $\alpha > 0, \beta > 0, 0 < \gamma < 1$ and are given a flat prior. Parameter $\tau = \sigma^{-2}$ is given a gamma prior, with density proportional to $\tau^{a-1}e^{-a\tau}$ for $a = 0.001$ and is integrated out analytically. The posterior is then:

$$\pi\left(\alpha, \beta, \gamma | \{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n\right) \propto \left[2a + \sum_{i=1}^{n} (y_i - \alpha + \beta\gamma^{x_i})^2\right]^{-a-\frac{n}{2}}.$$

We first make log and logit transformations so that the transformed posterior is defined on $\mathbb{R}^3$. We then make another transformation based on the Laplace approximation, as described in subsection 2.5.1. Figure 6.10 shows the marginals of this distribution, generated using the RWM algorithm. The distribution has heavy tails.

For eight different version of the Hamiltonian Jumpar algorithm, we generated 100 sample paths, each consisting of a jump chain of length 10^5. Before running the Hamiltonian Jumpar algorithms, we tuned HMC for the same problem, finding that $\epsilon = 0.86$ and $L = 2$ were good parameters. This motivated using $\epsilon = 0.86$ in the following versions of a Hamiltonian Jumpar:

- Unit holding rate, $L = 1$, no velocity refreshment (uL1)
- Unit holding rate, $L = 1$, velocity refreshment (uL1r)
- Unit holding rate, $L = 2$, no velocity refreshment (uL2)



(a) X_1.

(b) X_2.

(c) X_3.

Figure 6.10: Kernel density estimates of the marginals of the transformed posterior distribution of a model of Dugongs (sea cows), computed using samples generated by the Random Walk Metropolis algorithm.

- Unit holding rate, $L = 2$, velocity refreshment (uL2r)
- Weighted holding rate, $L = 1$, no velocity refreshment (wL1)
- Weighted holding rate, $L = 1$, velocity refreshment (wL1r)
- Weighted holding rate, $L = 2$, no velocity refreshment (wL2)
- Weighted holding rate, $L = 2$, velocity refreshment (wL2r)

When using a weighted holding rate we set $\mu_V \equiv \pi_V \equiv \mathcal{N}(0, I_3)$. When using a unit holding rate we set $\pi_V \equiv \mathcal{N}(0, I_3)$ and $\mu_V \equiv t_{20}(0, I_3)$. In the latter case, a very long pre-run was used to ensure C was large enough. The motivation for not setting μ_V as equivalent to π_V was, as explained in 6.4.4, to speed-up the adaption of C. In all cases we used $\mu_X \equiv t_4(0, I_3)$. Boxplots of the results are given in Figure 6.11. The right-most boxplot shows 100 estimates computed using 10^5 weighted samples generated using Importance Sampling with proposal distribution $t_4(0, I_3)$. The estimators computed using Importance Sampling had smallest MSE. A reason for this may be that π is very close to μ_X, so when μ_X is used as the proposal for Importance Sampling, the Importance Sampler is able to generate very high quality samples. Indeed, for simple models in low to mid dimensional space, Importance Sampling is very effective Chopin and Ridgway (2017).

Banana distribution

For the last example, a Hamiltonian-Jumpar performed worse than Importance Sampling at estimating the mean of a component of a simple 3-dimensional posterior. However, since Importance Sampling was particularly well suited to the last example, this section explores whether Hamiltonian Jumpar offers advantages when there is a larger discrepancy between the target and regeneration / importance distribution.

Let π_X be the Banana distribution, defined in Section 4.4.1, with dimension $d = 2$ and parameters $a = 100, b = 0.04, \sigma_1 = 10$ and $\sigma_2 = 1$. The Laplace approximation of this target distribution has mean $m = (0, 4)^T$ and covariance matrix:

$$\Sigma = \begin{pmatrix} 100 & 0 \\ 0 & 1 \end{pmatrix}.$$

No transformation of π_X is made. Set $\pi_V \equiv \mathcal{N}(0, I_2)$ and $\mu_X \equiv t_3(m, \Sigma)$. When using a unit holding rate, set $\mu_V \equiv t_{20}(0, I_2)$ but when using a weighted holding rate, set $\mu_V \equiv \pi_V$. Tuning HMC we find that $\epsilon = 0.85$ and $L = 16$ are good

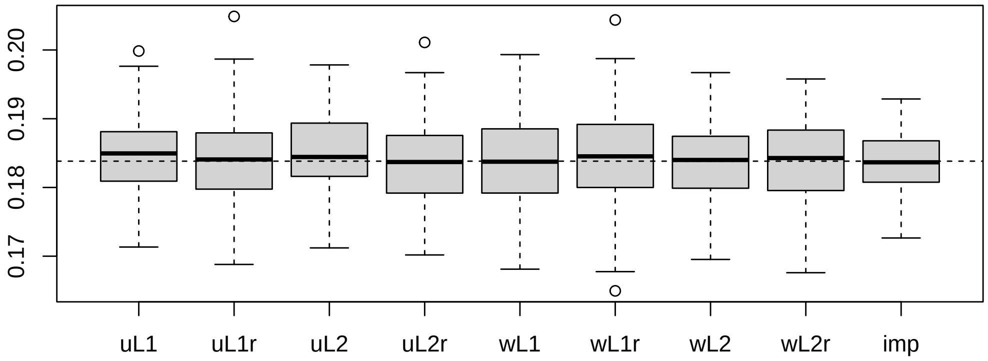


Figure 6.11: Boxplots of estimates of the mean of the third variable in a model of Dugongs. Each boxplot consists of 100 estimates. The first 8 boxplots contain estimates computed using a Hamiltonian Jumpar. For these the labels have first letter either “u” for unit holding rate or “w” for weighted holding rate. The second and third characters are either “L1” for $L = 1$ leapfrog step or “L2” for $L = 2$ leapfrog steps. A fourth character “r” indicates that velocity refreshment was used, else there was no velocity refreshement. The ninth boxplot, labelled imp, shows the Importance Sampling estimates. The horizontal line shows a very good approximation of the mean, computed using a long run of HMC.

choices of the tuning parameters. Thus ϵ is fixed as 0.85 for all simulations. The versions of Hamiltonian Jumpar considered were the same as those for the Dugong model, except with $L \in \{1, 16\}$: unit holding rate, $L = 1$, no velocity refreshment (uL1); unit holding rate, $L = 1$, velocity refreshment (uL1r); unit holding rate, $L = 16$, no velocity refreshment (uL16); unit holding rate, $L = 16$, velocity refreshment (uL16r); weighted holding rate, $L = 1$, no velocity refreshment (wL1); weighted holding rate, $L = 1$, velocity refreshment (wL1r); weighted holding rate, $L = 16$, no velocity refreshment (wL16); weighted holding rate, $L = 16$, velocity refreshment (wL16r). For each type of process, 100 sample paths are estimated and used to compute an estimate of the mean of the second component, which has exact value 0. Each sample path is a jump process with jump chain of length 4×10^5. Importance Sampling is used to compute 100 estimates of the mean for comparison, each estimate being auto-normalized and based on 4×10^5 weighted samples. Results are shown in Figure 6.12.

Hamiltonian Jumpar sampling is again less effective than Importance Sampling, which has the smallest MSE of the estimates produced. Here, the versions of H-Jumpar with the smallest MSE had a weighted holding rate and $L = 16$ leapfrog steps. Since making such a transformation is very expensive, in a com-

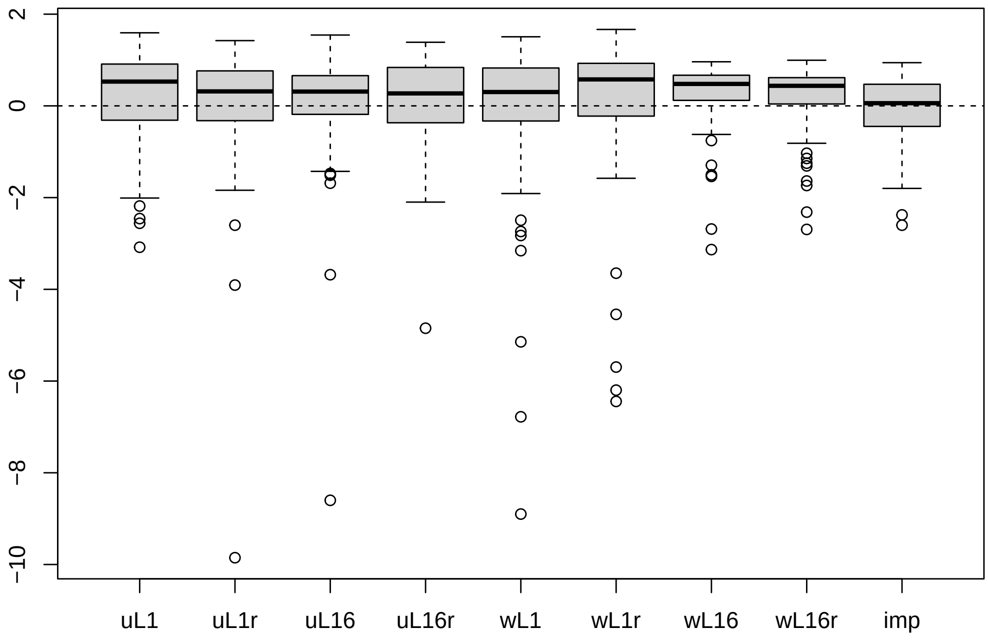


Figure 6.12: Boxplots of estimates of the mean of the second variable of a Banana distribution. Each boxplot consists of 100 estimates. The first 8 boxplots contain estimates computing using a Hamiltonian Jumpar. For these the labels have first letter either "u" for unit holding rate or "w" for weighted holding rate. The second character is "L" which is followed b a number, either 1 for $L = 1$ leapfrog steps of 16 for $L = 16$ leapfrog steps. If there is a final character "r", it indicates that velocity refreshment was used, else there was no velocity refreshment. The ninth boxplot, labelled imp, shows the Importance Sampling estimates. The horizontal line shows the exact mean.

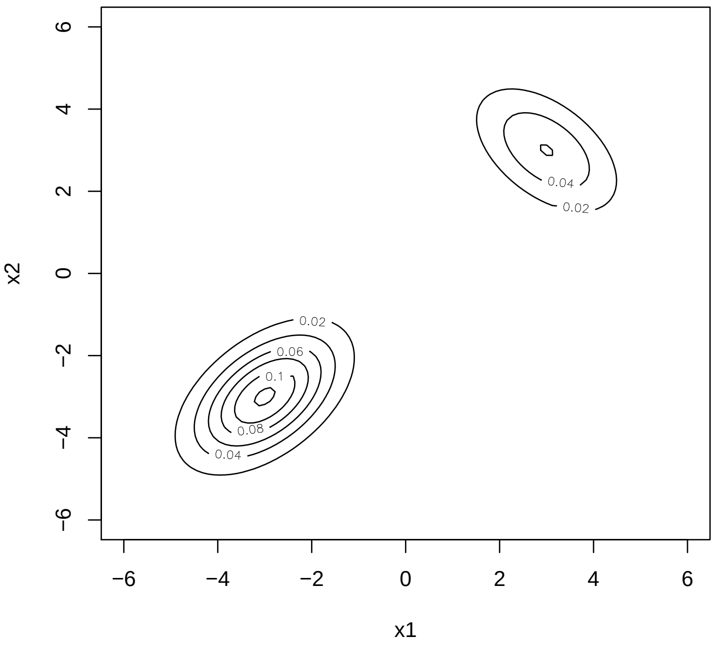


Figure 6.13: Contours of the density of a mixture of bivariate Gaussian distributions. There is a large region of low probability density between the modes of the distribution.

parison of H-Jumpar to Importance Sampling based on a fixed computational budget instead of a fixed number of weighed samples, H-Jumpar would do even worse.

Gaussian Mixture

Consider sampling from a mixture of two bivariate Gaussian distributions, with density

$$\pi(x) = w_1\mathcal{N}(x; m_1, \Sigma_1) + w_2\mathcal{N}(x; m_2, \Sigma_2), \tag{6.15}$$

where $w_1 = 0.33, w_2 = 0.67, m_1 = (3, 3)^T, m_2 = (-3, -3)^T$,

$$\Sigma_1 = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \quad , \quad \Sigma_2 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}.$$

As shown by the plot of the contours of the density in Figure 6.13, there is a large region of low probability density between the modes. This region of low probability density acts as a barrier, making it difficult for Metropolis-Hastings samplers to cross from one mode to the other.

Figure 6.14 shows the traceplot of the first marginal of a Markov chain started at $(3, 3)^T$ and generated using HMC so that the stationary distribuiton of the chain is π. Parameters $\epsilon = 0.78$ and $L = 2$ were selected by tuning HMC on the unimodal distribution $\mathcal{N}(0, \Sigma_2)$. After 10^6 iterations, though the chain has

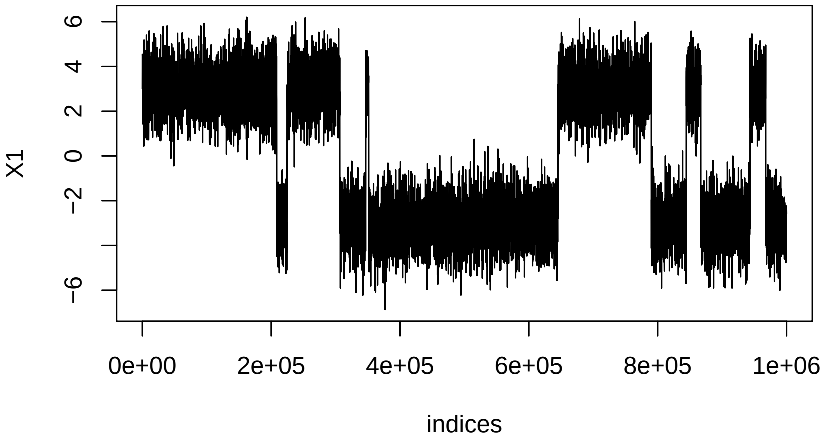


Figure 6.14: Trace plot of a Markov chain generated using HMC, with target distribution a mixture of Gaussians. The x-axis shows the indices of the states in the Markov chain.

moved between the modes, the estimate of the mean of the first component is very misleading (MSE=0.89), because the chain has not yet accurately sampled the modes in proportion to their weights.

In comparison, a Hamiltonian Jumpar allows better switching between modes, which results in a far better estimate of the mean of the first component of X. However, we find that Importance Sampling again outperforms Hamiltonian Jumpar. Let $\pi_V \equiv \mathcal{N}(0, I_2)$ and $\mu_X \equiv \mathcal{N}(0, 9I_2)$, the latter also serving as the importance distribution. For a unit holding rate, let $\mu_V \equiv t_{20}(0, I_2)$ and for a weighted holding rate let $\mu_V \equiv \pi_V \equiv \mathcal{N}(0, I_2)$. We generated 100 estimates of the mean of the first component of X, each based on 10^5 weighted samples, for each of 8 versions of Hamiltonian Jumpar as well as Importance Sampling. The versions of Hamiltonian Jumpar considered were the same as for the Dugong model: unit holding rate, $L = 1$, no velocity refreshment (uL1); unit holding rate, $L = 1$, velocity refreshment (uL1r); unit holding rate, $L = 2$, no velocity refreshment (uL2); unit holding rate, $L = 2$, velocity refreshment (uL2r); weighted holding rate, $L = 1$, no velocity refreshment (wL1); weighted holding rate, $L = 1$, velocity refreshment (wL1r); weighted holding rate, $L = 2$, no velocity refreshment (wL2); weighted holding rate, $L = 2$, velocity refreshment (wL2r). Figure 6.15 shows boxplots of the estimators. The estimators generated using Importance Sampling had the smallest MSE.

This section has looked at enriching a Jump process, which moves according to unadjusted Hamiltonian dynamics, with regenerations so that the invariant distribution of the enriched process is π_θ. However, examples have shown the process does not make for a particularly efficient algorithm for generating sam-

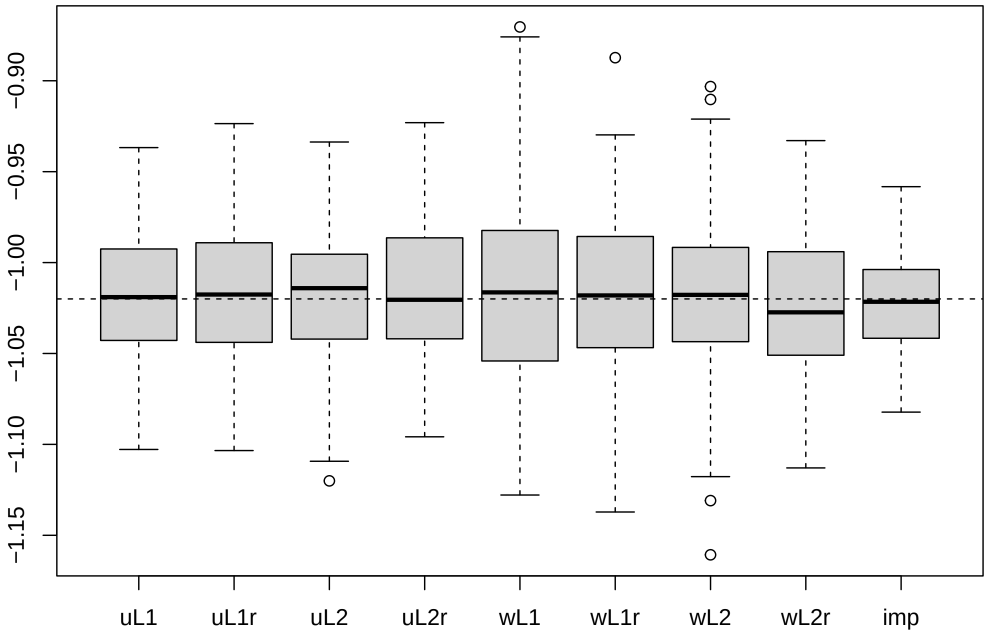


Figure 6.15: Boxplots of estimates of the mean of the first component of a Gaussian Mixture model. Each boxplot consists of 100 estimates. The first 8 boxplots contain estimates computed using a Hamiltonian Jumpar. For these the labels have first letter either "u" for unit holding rate or "w" for weighed holding rate. The second character is "L" which is followed by a number, either 1 for $L = 1$ leapfrog steps or 2 for $L = 2$ leapfrog steps. If there is a final character "r", it indicates that velocity refreshment was used, else there was no velocity refreshment. The ninth boxplot, labeled imp, shows the Importance Sampling estimates. The horizontal line shows the exact mean.

ples for Monte Carlo. The next section considers a modification to Hamiltonian dynamics, which results in a promising sampling algorithm.

6.5 Conformal Hamiltonian Dynamics

This section introduces the *Conformal Hamiltonian Jump Process Adjusted with Regenerations* (Conformal Hamiltonian Jumpar or CH-Jumpar). The transformation for this class of Jumpar corresponds to conformal Hamiltonian dynamics, which introduce a friction term into Hamiltonian dynamics. Instead of the Hamiltonian being preserved, the particle represented by the dynamics gradually loses energy and moves towards the minimizer of the potential energy as it does. Thus the dynamics may be used for optimisation – under certain conditions, linear convergence to the global minimizer of the potential energy is achieved (Maddison et al., 2018).

To visualize conformal Hamiltonian dynamics, one might think of a skateboard moving on a half-pipe under friction. The skateboard starts at the top of one side of the half-pipe, accelerates as is moves down, then decelerates as it moves up the other side until it reaches a point below where it first started, at which time it is momentarily stationary again. The skateboard keeps moving from side to side like this, gradually losing energy due to friction, until it is stationary at the bottom of the half-pipe.

The motivation for using conformal Hamiltonian dynamics in a sampler is as a means of finding areas of high probability mass automatically, using gradient information alone. As briefly covered in subsection 2.3.2, when sampling from a multi-modal distribution, the standard Metropolis-Hastings algorithm switches between modes very infrequently because the areas of low probability density between the modes act as barriers. MCMC algorithmns designed for multimodal distributions generally encourage mode-jumping either through tempering strategies (Geyer, 1991; Marinari and Parisi, 1992; Neal, 1996b; Kou et al., 2006) or by identifying all the modes of π before the Markov chain is generated (see Pompe et al. (2020) for example). Here, we experiment with using a CH-Jumpar to sample a multimodal distribution π *without* already knowing the locations of the modes. We are interested in whether a dispersed regeneration distribution may be used, which is compensated for by the conformal Hamiltonian dynamics leading the sampler towards the regions of π with high probability mass. Conformal Hamiltonian dynamics have been used in Importance Samplers (Rotskoff and Vanden-Eijnden, 2019; Thin et al., 2021) as a way to sample more often from the modes of the distribution.

In this section we define the transformation used to approximate conformal Hamiltonian dynamics, define the Conformal Hamiltonian Jumpar and show that this process is effective in sampling a target distribution where most of the mass is contained in small spikes of apriori unknown locations.

6.5.1 Conformal Hamiltonian Jump Process Adjusted with Regenerations

Let $\pi_V \equiv \mathcal{N}(0, I)$. The transformation corresponding to approximate conformal Hamiltonian dynamics, with step-size $\epsilon > 0$ and friction $\gamma > 0$ is $A_3 : \Theta \to \Theta$ given by

$$A_3(\vartheta) = A_3(x, v) = \big(x + \epsilon[e^{-\epsilon\gamma}v - \epsilon\nabla U(x)], e^{-\epsilon\gamma}v - \epsilon\nabla U(x)\big)^T .$$

The transformation is computed using steps $v \leftarrow e^{-\epsilon\gamma}v - \epsilon\nabla U(x)$ followed by $x \leftarrow x + \epsilon v$. The inverse is:

$$A_3^{-1}(x, v) = (x - \epsilon v, e^{\gamma\epsilon}[v + \epsilon\nabla U(x - \epsilon v)])^T ,$$

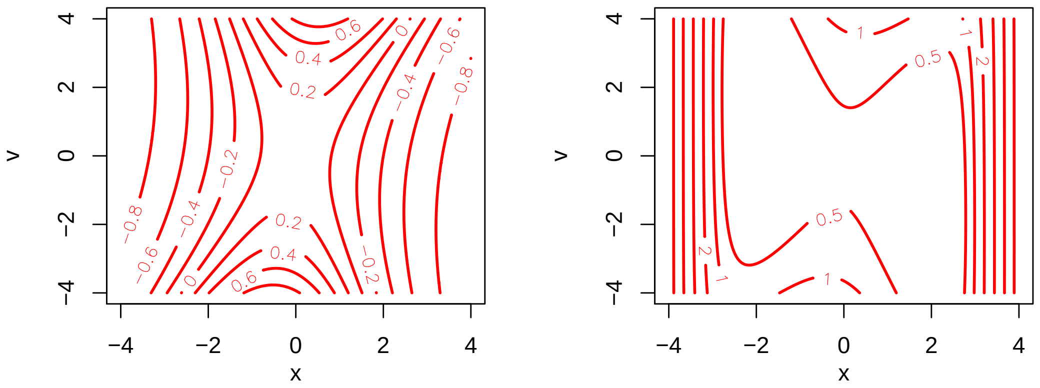
which is computed using steps $x \leftarrow x - \epsilon v$ followed by $v \leftarrow e^{\gamma\epsilon}[v + \epsilon\nabla U(x)]$. For d the dimension of x-co-ordinate, $|J_{A_3^{-1}}| = e^{\gamma\epsilon d}$.

Let $\{\Xi_t\}_{t\geq 0} = \{Y_t, W_t\}_{t\geq 0}$ be a jump process with holding rate $\lambda : \Theta \to \Theta$ and jump chain defined by a Markov transition kernel corresponding to the transformation A_3 as defined above. Then $\{\Xi_t\}_{t\geq 0}$ may be enriched with regenerations, as in Theorem 4 so that $\{\theta_t\}_{t\geq 0}$, the enriched process, has invariant distribution π_θ. Such a process is called a Conformal Hamiltonian Jump Process Adjusted with Regenerations.

Complexity

Suppose that a tour consists of states $\vartheta_0, \vartheta_1, \ldots, \vartheta_n$. When the process regenerates into state $\vartheta_0 = (x_0, v_0)$, whether the holding rate is constant or weighted, one must compute $\pi_\theta(\vartheta_0)$ and $A_3^{-1}(\vartheta_0)$. The latter involves computing $\nabla U(x_0 - \epsilon v_0)$, which we count as equivalent in cost to evaluating π_X. If the holding rate is constant, $\pi_\theta(A_3^{-1}(\vartheta_0))$ must be evaluated, which is again equivalent in cost to evaluating π_X because the cost of evaluating π_V may be ignored. Thus each time a global move is made, there is a cost equivalent to two or three evaluations of π_X, depending on whether the holding rate is weighted or constant.

The cost associated with each state $\vartheta_i = (x_i, v_i)$ for $i \in \{1, \ldots, n\}$ is that of making the transformation to get to ϑ_i and of evaluating $\pi_\theta(\vartheta_i)$. The former



(a) Contours of $\tilde{\kappa}(x, v)$.

(b) Contours of $\kappa(x, v)$.

Figure 6.16: Contour plots of $\tilde{\kappa}$ and κ for a Conformal Hamiltonian Jump Process Adjusted with Regenerations with $\lambda \equiv 1, \pi_X \equiv \mathcal{N}(0,1), \mu_X \equiv t_\nu(0,1), \epsilon = 0.5$ and $\gamma = 0.2$.

involves computing $A_3(\vartheta_{i-1})$, which requires computing $\nabla U(x_{i-1})$, equivalent in cost to evaluating $\pi_X(x_{i-1})$. The latter is equivalent in cost to evaluating $\pi_X(x_i)$. Thus each local moves has an associated cost equivalent to two evaluation of π_X.

6.5.2 Examples

Standard Normal Target

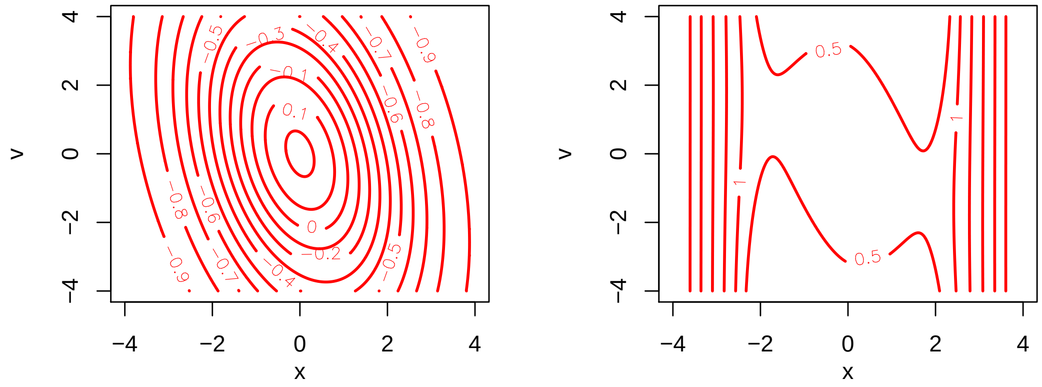
Let $\pi_X \equiv \mathcal{N}(0,1)$ and $\mu_X \equiv t_\nu(0,1)$, with $\nu = 3$. For a unit holding rate and $\epsilon = 0.5$, plots of $\tilde{\kappa}$ and κ for $\gamma = 0.2$ and $\gamma = 0.4$ are given in Figures 6.16 and 6.17 respectively. Figure 6.18 shows the weighted samples generated by 50 tours of the process. The area of each circle represents the weight of that sample. Tours are connected by gray lines.

Gaussian Mixture Distribution

Consider a d-dimensional Gaussian mixture distribution

$$\pi(x) = w_1 \mathcal{N}(x; m_1, \Sigma_1) + w_2 \mathcal{N}(x; m_2, \Sigma_2),$$

with $d = 4, w_1 = 0.33, w_2 = 0.67, m_1 = 3 \times 1_d, m_2 = -3 \times 1_d, \Sigma_1 = \Sigma_2 = 0.1 I_d$. Suppose that we have very little idea of where the modes of π are – therefore when comparing Importance Sampling and Jumpar sampling, we take as proposal and regeneration distribution $\mu_X \equiv \mathcal{N}(0, 9I_d)$. Though $d = 4$ is not particularly large



(a) Contours of $\tilde{\kappa}(x, v)$.

(b) Contours of $\kappa(x, v)$.

Figure 6.17: Contour plots of $\tilde{\kappa}$ and κ for a Conformal Hamiltonian Jump Process Adjusted with Regenerations with $\lambda \equiv 1, \pi_X \equiv \mathcal{N}(0, 1), \mu_X \equiv t_\nu(0, 1), \epsilon = 0.5$ and $\gamma = 0.4$.

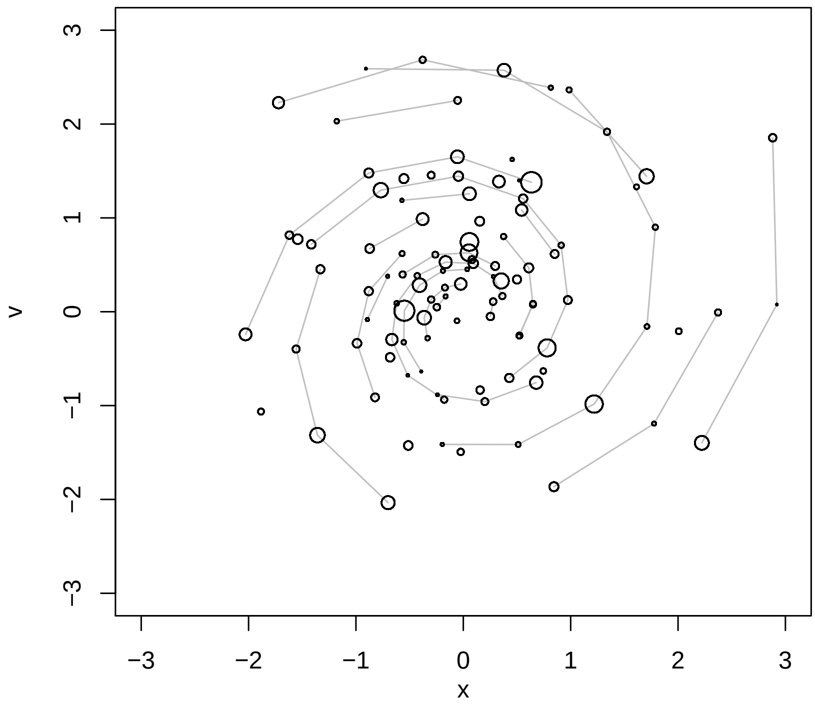


Figure 6.18: Fifty tours of a Conformal Hamiltonian Jump Process Adjusted with Regenerations with $\lambda \equiv 1, \pi_X \equiv \mathcal{N}(0, 1), \mu_X \equiv t_\nu(0, 1), \epsilon = 0.5$ and $\gamma = 0.4$. Circles represented weighted samples, with area representing the weight. Tours are connected by gray lines.

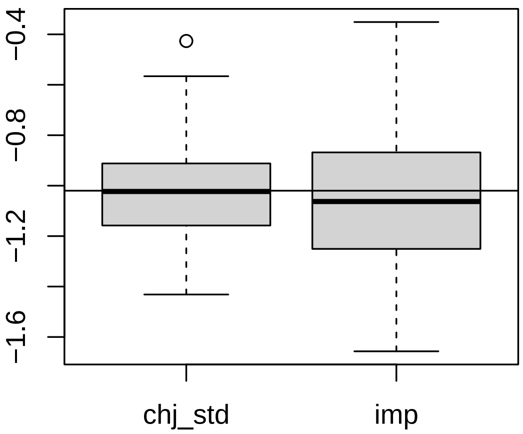


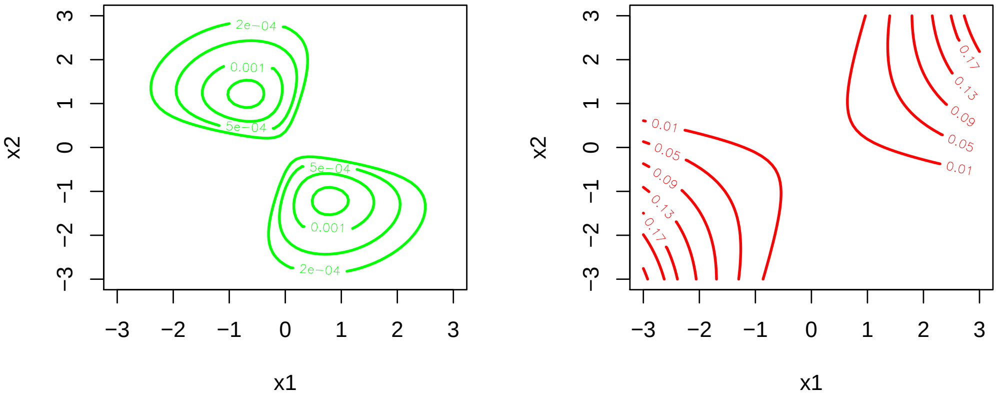
Figure 6.19: Estimating the mean of a Gaussian Mixture Distribution: a comparison of Conformal Hamiltonian Jumpar sampling and Importance Sampling. Boxplots of 100 estimates of the mean of the first component of a Gaussian mixture distribution, each computed using 10^6 weighted samples. chj_std: Conformal Hamiltonian Jumpar with unit holding rate. imp: Importance Sampling. Horizontal line shows the true value of the mean.

by the standards of modern Bayesian inference techniques, since we don't know apriori the locations of the small regions where the mass of π is concentrated, this is still a difficult sampling problem. Another way to understand the difficulty of this sampling problem is to consider the computational cost of performing integration via quadrature; using a grid with mesh-size 0.1 on region $[-6, 6]^4$ would require over 2×10^8 evaluations of the integrand.

We computed 100 estimates of the mean of the first component of X by generating 100 CH-Jumpars with unit holding rate and jump chain length 10^6. Figure 6.19 shows a Boxplot comparing the estimated moments to those generated using Importance Sampling. The horizontal line shows the true value of the mean. Using the Jumpar lead to a 52% reduction in MSE. Estimates computed using a weighted holding rate were worse than Importance Sampling estimates and are not shown.

6.6 Discussion

This chapter has introduced a class of Jump processes defined by enriching a non-π-invariant jump process with regenerations from μ at rate κ so that the resulting



(a) Contours of μ^+.

(b) Contours of κ^+.

Figure 6.20: Contour plots of μ^+ and κ^+ for a Hamiltonian Jump Process Adjusted with Regenerations with $\lambda \equiv 1$ and $\pi_X \equiv \pi_V \equiv \mathcal{N}(0, 1)$.

process is π-invariant. Three choices of local dynamics have been proposed: Random Direction, Hamiltonian and Conformal Hamiltonian. The holding rate may either be constant or weighted. The former necessitates finding an appropriate regeneration constant, such that the regeneration rate is non-negative everywhere. The latter can result in inefficient computation: sample paths exhibit a sequence of very short staying times, corresponding to a sequence of states for which both the holding and regeneration rates are very large and near equal.

Combining the adaptive mechanism of Chapter 5 with the methodology presented in this chapter may improve performance. When the holding rate is constant, this would allow the minimal regeneration distribution and rate, μ^+ and κ^+, to be used. This would remove the need to find an appropriate value of C. Figure 6.20 shows μ^+ and κ^+ for Hamiltonian dynamics and $\pi_X \equiv \mathcal{N}(0, 1)$. Note that when the holding rate is non-constant, there is no minimal regeneration distribution. This is a consequence of the underlying process relying on μ, as well as the enriched process regenerating from μ.

The most promising dynamics are Conformal Hamiltonian. The friction term introduced into Hamiltonian dynamics makes the process move towards areas of high probability mass. This enables efficient sampling of posterior distributions for which the mass is concentrated in small spikes with apriori unknown locations.

Chapter 7

Conclusion

Statistical models are becoming more challenging. There is demand for Bayesian computation methods that are able to deal with high-dimensional posterior distributions, displaying complicated inter-dependencies between covariates and relying on large quantities of data. MCMC remains a highly useful tool and has the key strength of producing asymptotically exact estimates of expectations. However, MCMC has limitations that make it not the most suitable tool for some applications. In Chapter 2, four desiderata for MCMC methods were set out. These were:

1. Compensation.
   The process consists of different dynamics, which by themselves are non π-invariant, but together compensate for each other, so that in tandem they are π-invariant.

2. Local and Global dynamics.
   The process consists of local dynamics (the next state of the process tends to be in a small vicinity of the current state) and global dynamics (the next state is independent of the current state and is anywhere in the state space).

3. Regeneration.
   The process is regenerative: there exists a sequence of random times at which the future of the process is independent of the past and identically distributed. One can then simulate the process in parallel, without a burn-in period and attain an estimate for the variance of the estimator itself from a single sample path.

4. Non-reversible.
   The process is non-reversible; it does not satisfy the detailed balance condition.

Wang et al. (2021) proposed the Restore process, which satisfies all of the above criteria. Although it was a fantastic contribution, the paper did not set out how a non-π-invariant jump process may be enriched with regenerations so that the resulting jump process is π-invariant, nor did it explore the use of Restore for sampling mid-dimensional distributions.

Wang et al. (2021) motivated the two major contributions of this thesis, which are as follows:

1. Adaptive Restore (Chapter 5) (McKimm et al., 2022): showing that a Brownian motion may be enriched with regeneration from a distribution that is adapted over time, so that the resulting process is π-invariant.

2. Jumpar (Chapter 6): defining a class of π-invariant jump process by enriching with regeneration an underlying non-π-invariant jump process on an augmented state space, for which the jump chain is defined by a Markov transition kernel corresponding to a deterministic, invertible mapping.

For both contributions, a number of experiments have been used to investigate the properties of the proposed algorithms. In addition, Chapter 4 made a minor contribution in Section 4.5 by showing that Restore processes may be used to estimate normalising constants.

The main practical issue with Standard Restore is that the regeneration rate can become extremely large when μ is not a good approximation of π. As the dimension of the problem increases, it is inevitable that any fixed choice of μ will not approximate π well and hence κ will become very large. In previous work on simulating regenerative Markov chains (Mykland et al., 1995) using Nummelin's splitting technique (Nummelin, 1978), regenerations recede exponentially with the dimension of π. By contrast, the pitfall of Standard Restore is that as the dimension increases the process regenerates repeatedly, but mostly to areas of very low probability mass, which wastes computation.

Adaptive Restore can drastically reduce the average regeneration rate and thus make simulation feasible. As an example, for the hierarchical model of pump failure of subsection 5.3.3, using Adaptive Restore instead of Standard Restore reduces the regeneration rate by a factor 10^7. Regarding the four criteria for MCMC simulation set out above, Adaptive Restore uses compensating dynamics, local and global dynamics and is non-reversible. Adaptive Restore is not in fact regenerative: the adaption of μ_t means that the cycles are no longer independently distributed. As a consequence, a burn-in period is necessary for Adaptive Restore and simulating the process in parallel is less trivial. It is worth noting that if μ_t were fixed at any time $t \geq 0$, the process would again be regenerative.

There remains plenty of scope for further work. Chapter 5 concerns adaptively enriched Brownian motion, but the adaptive framework could equally be applied to diffusions. It would be interesting to see whether a drift coefficient could help to improve the sampler. As covered in Section 5.4, using an Adaptive Restore process with short-term memory, or using local bounds for the regeneration rate, could likewise improve the sampler. An Adaptive Restore process with short-term memory would make the process less expensive to simulate in terms of computer memory, since for any fixed time interval fewer states would need to be stored. Using local bounds for the regeneration rate could result in κ^+ (and thus ∇U and ΔU) being evaluated less.

Previous work on Jump Processes Enriched with Regenerations (Wang et al., 2021) showed how a π-invariant jump process could be enriched with regenerations. Although such a process satisfies all of the four criteria listed above, the process isn't fully able to take advantage of non-reversible dynamics, because the underlying jump process must be specified by a reversible Markov transition kernel for the jump chain. A Jumpar also satisfies the four criteria above, but by contrast takes as underlying process a non-π-invariant jump process. This allows the process to fully break free of the requirement that the Markov transition kernel be reversible. The benefits of this are best evidenced by the potential of the Conformal Hamiltonian Jumpar, which uses a friction term to move towards areas of high probability mass. Two other dynamics for the underlying jump process were considered, but other transformations could also be used. Furthermore, it may be worth investigating whether the adaptive mechanism of Chapter 5 could be used for the jump processes of Chapter 6.

Instead of making minor adjustments to state-of-the-art methods with the aim of making maginal improvements, this thesis has experimented with largely unknown and untested methods. In doing so, the aim has been to take a completely different approach to Monte Carlo, by looking at the use of continuous-time Markov processes. The methods presented are not yet ready for practical use in sampling the most challenging target distributions, but progress has been made in exploring how regeneration-enriched Markov processes may be used for Monte Carlo.

Bibliography

Ahn, S., Chen, Y., and Welling, M. (2013). Distributed and Adaptive Darting Monte Carlo through Regenerations. In Carvalho, C. M. and Ravikumar, P., editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 108–116, Scottsdale, Arizona, USA. PMLR.

Alder, B. J. and Wainwright, T. E. (1959). Studies in Molecular Dynamics. I. General Method. *The Journal of Chemical Physics*, 31(2):459–466.

Andricioaei, I., Straub, J. E., and Voter, A. F. (2001). Smart Darting Monte Carlo. *The Journal of Chemical Physics*, 114(16):6994–7000.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43.

Andrieu, C. and Moulines, E. (2006). On the ergodicity properties of some adaptive MCMC algorithms. *The Annals of Applied Probability*, 16(3):1462 – 1505.

Andrieu, C. and Robert, C. P. (2001). Controlled MCMC for Optimal Sampling. Technical report, Cahiers de Mathématiques du Ceremade, Université Paris-Dauphine.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373.

Asmussen, S. (2003). *Applied Probability and Queues*, volume 51. Springer-Verlag New York.

Atchadé, Y. F. and Rosenthal, J. S. (2005). On adaptive Markov chain Monte Carlo algorithms. *Bernoulli*, 11(5):815 – 828.

Bash, P. A., Singh, U. C., Langridge, R., and Kollman, P. A. (1987). Free Energy Calculations by Computer Simulation. *Science*, 236(4801):564–568.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research*, 18.

Beck, J., Dia, B. M., Espath, L. F., Long, Q., and Tempone, R. (2018). Fast Bayesian experimental design: Laplace-based importance sampling for the expected information gain. *Computer Methods in Applied Mechanics and Engineering*, 334:523–553.

Bédard, M. (2006). *On the robustness of optimal scaling for Random Walk Metropolis algorithms*. PhD thesis, University of Toronto.

Benaim, M., Cloez, B., and Panloup, F. (2018). Stochastic Approximation of Quasi-stationary Distributions on Compact Spaces and Applications. *The Annals of Applied Probability*, 28(4):2370–2416.

Bertazzi, A. and Bierkens, J. (2021). Adaptive schemes for piecewise deterministic Monte Carlo algorithms.

Besag, J. and Green, P. J. (1993). Spatial statistics and bayesian computation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(1):25–37.

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

Betancourt, M. (2017). A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.

Bierkens, J., Fearnhead, P., and Roberts, G. (2019a). The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data. *The Annals of Statistics*, 47(3):1288–1320.

Bierkens, J., Grazzi, S., Kamatani, K., and Roberts, G. (2020). The Boomerang Sampler.

Bierkens, J., Grazzi, S., van der Meulen, F., and Schauer, M. (2021). Sticky PDMP samplers for sparse and local inference problems.

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

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Bouchard-Côté, A., Vollmer, S. J., and Doucet, A. (2018). The Bouncy Particle Sampler: A Nonreversible Rejection-Free Markov Chain Monte Carlo Method. *Journal of the American Statistical Association*, 113(522):855–867.

Box, G. E. P. and Muller, M. E. (1958). A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2):610 – 611.

Brockwell, A. E. and Kadane, J. B. (2005). Identification of Regeneration Times in MCMC Simulation, With Application to Adaptive Schemes. *Journal of Computational and Graphical Statistics*, 14(2):436–458.

Broydon, C. G. (1970). The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90.

Bugallo, M. F., Elvira, V., Martino, L., Luengo, D., Miguez, J., and Djuric, P. M. (2017). Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79.

Cappé, O., Godsill, S. J., and Moulines, E. (2007). An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924.

Cappé, O., Guillin, A., Marin, J.-M., and Robert, C. P. (2004). Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929.

Carlin, B. P. and Gelfand, A. E. (1991). An iterative Monte Carlo method for nonconjugate Bayesian analysis. *Statistics and Computing*, 1(2):119–128.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A Probabilistic Programming Language. *Journal of statistical software*, 76(1).

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P., and Betancourt, M. (2015). The Stan Math Library: Reverse-Mode Automatic Differentiation in C++. *arXiv preprint arXiv:1509.07164*.

Chen, L., Qin, Z., and Liu, J. S. (2000). Exploring Hybridg Monte Carlo in Bayesian Computation. *ISBA Proceedings*.

Chen, T.-L. and Hwang, C.-R. (2013). Accelerating reversible Markov chains. *Statistics & Probability Letters*, 83(9):1956–1962.

Chopin, N. (2002). A Sequential Particle Filter Method for Static Models. *Biometrika*, 89(3):539–551.

Chopin, N. and Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo*. Springer International Publishing.

Chopin, N. and Ridgway, J. (2017). Leave Pima Indians Alone: Binary Regression as a Benchmark for Bayesian Computation. *Statistical Science*, 32(1):64–87.

Collet, P., Martínez, S., and San Martín, J. (2013). *Quasi-Stationary Distributions: Markov Chains, Diffusions and Dynamical Systems*. Springer, Berlin, Heidelberg.

Cotter, S., House, T., and Pagani, F. (2020). The NuZZ: Numerical ZigZag Sampling for General Models. *arXiv preprint arXiv:2003.03636*.

Dai, H., Pollock, M., and Roberts, G. (2019). Monte Carlo fusion. *Journal of Applied Probability*, 56(1):174–191.

Davis, M. H. A. (1984). Piecewise-Deterministic Markov Processes: A General Class of Non-Diffusion Stochastic Models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(3):353–388.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436.

Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J.-J., Sandhu, S., Guppy, K. H., Lee, S., and Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American Journal of Cardiology*, 64(5):304–310.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer, New York, NY.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP. In *International Conference on Learning Representations*.

Doucet, A., De Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo methods in Practice*. Springer.

Doucet, A., Johansen, A. M., et al. (2009). A Tutorial on Particle Filtering and Smoothing: Fifteen years later. *The Oxford Handbook of Nonlinear Filtering*, pages 656–704.

Dua, D. and Graff, C. (2017). UCI Machine Learning Repository.

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

Eberle, A. (2023). Markov Processes. University of Bonn. Lecture Notes.

Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322.

Gabrié, M., Rotskoff, G. M., and Vanden-Eijnden, E. (2022). Adaptive Monte Carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10):e2109420119.

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, 85(410):398–409.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Chapman and Hall/CRC.

Gelman, A., Jakulin, A., Pittau, M. G., and Su, Y.-S. (2008). A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2(4):1360 – 1383.

Gelman, A. and Meng, X.-L. (1998). Simulating Normalizing Constants: From Importance Sampling to Bridge Sampling to Path Sampling. *Statistical science*, pages 163–185.

Gelman, A., Roberts, G. O., Gilks, W. R., et al. (1996). Efficient Metropolis jumping rules. *Bayesian statistics*, 5(599-608):42.

Gelman, A., Rubin, D. B., et al. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical science*, 7(4):457–472.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

Geyer, C. J. (1991). Markov Chain Monte Carlo Maximum Likelihood. *Interface Foundation of North America.*

Gilks, W. R. (1992). Derivative-free adaptive rejection sampling for Gibbs sampling. *Bayesian Statistics*, 4(2):641–649.

Gilks, W. R., Roberts, G. O., and Sahu, S. K. (1998). Adaptive Markov Chain Monte Carlo through Regeneration. *Journal of the American statistical association*, 93(443):1045–1054.

Gilks, W. R., Thomas, A., and Spiegelhalter, D. J. (1994). A Language and Program for Complex Bayesian Modelling. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 43(1):169–177.

Gilks, W. R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(2):337–348.

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

Glasserman, P., Heidelberger, P., and Shahabuddin, P. (2000). Variance reduction techniques for estimating value-at-risk. *Management Science*, 46(10):1349–1364.

Goldfarb, D. (1970). A Family of Variable-Metric Methods Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113(6).

Haario, H., Saksman, E., Tamminen, J., et al. (2001). An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242.

Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrica*, 57:97–109.

Hesterberg, T. (1995). Weighted Average Importance Sampling and Defensive Mixture Distributions. *Technometrics*, 37(2):185–194.

Hills, S. E. and Smith, A. F. M. (1992). Parameterization Issues in Bayesian Inference. In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., editors, *Bayesian Statistics 4*, pages 227–246. Oxford University Press, Oxford, U.K.

Hirt, M., Titsias, M., and Dellaportas, P. (2021). Entropy-based adaptive Hamiltonian Monte Carlo. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Hobert, J. P., Jones, G. L., Presnell, B., and Rosenthal, J. S. (2002). On the Applicability of Regenerative Simulation in Markov Chain Monte Carlo. *Biometrika*, 89(4):731–743.

Hoffman, M. D. and Gelman, A. (2014). The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

Kahn, H. (1950). Random Sampling (Monte Carlo) Techniques in Neutron Attenuation Problems. II. *Nucleonics*, 6(6).

Kingman, J. F. C. (1992). *Poisson Processes*, volume 3. Clarendon Press.

Kou, S., Zhou, Q., Wong, W. H., et al. (2006). Equi-energy sampler with applications in statistical inference and statistical mechanics. *The annals of Statistics*, 34(4):1581–1619.

Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2010). On the utility of graphics cards to perform massively parallel simulation of advanced monte carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789. PMID: 22003276.

Leimkuhler, B. and Reich, S. (2004). *Simulating Hamiltonian Dynamics*, volume 14. Cambridge University Press.

Lewis, P. A. W. and Shedler, G. S. (1979). Simulation of nonhomogeneous poisson processes by thinning. *Naval Research Logistics Quarterly*, 26(3):403–413.

Liggett, T. M. (2010). *Continuous Time Markov Processes: An Introduction*, volume 113. American Mathematical Soc.

Liu, J. S. (2008). *Monte Carlo Strategies in Scientific Computing*. Springer Science & Business Media.

Long, Q., Scavino, M., Tempone, R., and Wang, S. (2013). Fast estimation of expected information gains for Bayesian experimental designs based on Laplace approximations. *Computer Methods in Applied Mechanics and Engineering*, 259:24–39.

Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, 28(25):3049–3067.

Maddison, C. J., Paulin, D., Teh, Y. W., O'Donoghue, B., and Doucet, A. (2018). Hamiltonian Descent Methods.

Mangasarian, O. L. and Wolberg, W. H. (1990). Cancer diagnosis via linear programming. *SIAM News*, 23(5).

Marinari, E. and Parisi, G. (1992). Simulated Tempering: A New Monte Carlo Scheme. *EPL (Europhysics Letters)*, 19(6):451.

Martino, L., Elvira, V., Luengo, D., and Corander, J. (2017). Layered adaptive importance sampling. *Statistics and Computing*, 27(3):599–623.

McKimm, H., Wang, A. Q., Pollock, M., Robert, C. P., and Roberts, G. O. (2022). Sampling using Adaptive Regenerative Processes.

Meng, X.-L. and Wong, W. H. (1996). Simulating Ratios of Normalizing Constants Via A Simple Identity: A Theoretical Exploration. *Statistica Sinica*, 6(4):831–860.

Mengersen, K. L. and Tweedie, R. L. (1996). Rates of convergence of the Hastings and Metropolis algorithms. *The Annals of Statistics*, 24(1):101 – 121.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The journal of chemical physics*, 21(6):1087–1092.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341.

Meyn, S., Tweedie, R. L., and Glynn, P. W. (2009). *Markov Chains and Stochastic Stability*. Cambridge Mathematical Library. Cambridge University Press, 2 edition.

Minka, T. P. (2001). Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Müller, T., Mcwilliams, B., Rousselle, F., Gross, M., and Novák, J. (2019). Neural Importance Sampling. *ACM Trans. Graph.*, 38(5).

Mykland, P., Tierney, L., and Yu, B. (1995). Regeneration in Markov Chain Samplers. *Journal of the American Statistical Association*, 90(429):233–241.

Mörters, P. and Peres, Y. (2010). *Brownian Motion*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Neal, R. (1996a). *Bayesian Learning for Neural Networks*. 118. Springer-Verlag New York. Lecture Notes in Statistics.

Neal, R. M. (1993). Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical report, University of Toronto.

Neal, R. M. (1996b). Sampling from Multimodal Distributions Using Tempered Transitions. *Statistics and computing*, 6(4):353–366.

Neal, R. M. (2004). Improving Asymptotic Variance of MCMC Estimators: Non-reversible Chains are Better.

Neal, R. M. et al. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2.

Neiswanger, W., Wang, C., and Xing, E. P. (2014). Asymptotically Exact, Embarrassingly Parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 623–632, Arlington, Virginia, USA. AUAI Press.

Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313.

Norris, J. R. (1997). *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Nummelin, E. (1978). A splitting technique for Harris recurrent Markov chains. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 43(4):309–318.

Ogata, Y. (1989). A Monte Carlo method for high dimensional integration. *Numerische Mathematik*, 55(2):137–157.

Øksendal, B. (2003). *Stochastic Differential Equations.* Springer.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples.* In Progress.

Pakman, A. and Paninski, L. (2014). Exact Hamiltonian Monte Carlo for Truncated Multivariate Gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64.

Park, T. and Lee, S. (2022). Improving the Gibbs sampler. *WIREs Computational Statistics*, 14(2):e1546.

Pasarica, C. and Gelman, A. (2010). Adaptively scaling the Metropolis algorithm using the average squared jumped distance. *Statistica Sinica*, 20(1):343–364.

Plummer, M. (2003). JAGS: A Program for Analysis of Bayesian Graphical Models using Gibbs Sampling. *3rd International Workshop on Distributed Statistical Computing (DSC 2003); Vienna, Austria*, 124.

Pollock, M., Fearnhead, P., Johansen, A. M., and Roberts, G. O. (2020). Quasi-stationary Monte Carlo and the ScaLE algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(5):1167–1221.

Pompe, E., Holmes, C., and Łatuszyński, K. (2020). A framework for adaptive MCMC targeting multimodal distributions. *The Annals of Statistics*, 48(5):2930 – 2952.

Prangle, D. and Viscardi, C. (2019). Distilling importance sampling.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing, Third Edition.* Cambridge University Press.

Ramana, B. V., Babu, M. S. P., and Venkateswarlu, N. (2012). A Critical Comparative Study of Liver Patients from USA and INDIA: an Exploratory Analysis. *International Journal of Computer Science Issues (IJCSI)*, 9(3):506–516.

Ratkowsky, D. and Marcel Dekker, I. (1983). *Nonlinear Regression Modeling: A Unified Practical Approach*. Statistics Series. M. Dekker.

Revuz, D. and Yor, M. (2013). *Continuous Martingales and Brownian Motion*. Springer Berlin, Heidelberg.

Riabiz, M., Chen, W., Cockayne, J., Swietach, P., Niederer, S. A., Mackey, L., and Oates, C. J. (2020). Optimal thinning of mcmc output.

Robbins, H. and Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Robert, C. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer-Verlag New York.

Roberts, G. O., Gelman, A., Gilks, W. R., et al. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268.

Roberts, G. O. and Rosenthal, J. S. (2009). Examples of Adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367.

Roberts, G. O. and Tweedie, R. L. (1996a). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363.

Roberts, G. O. and Tweedie, R. L. (1996b). Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83(1):95–110.

Rogers, L. C. G. and Williams, D. (2000). *Diffusions, Markov Processes and Martingales*, volume 2 of *Cambridge Mathematical Library*. Cambridge University Press, 2 edition.

Rotskoff, G. M. and Vanden-Eijnden, E. (2019). Dynamical Computation of the Density of States and Bayes Factors Using Nonequilibrium Importance Sampling. *Phys. Rev. Lett.*, 122:150602.

Rubino, G. and Tuffin, B. (2009). *Rare Event Simulation using Monte Carlo Methods*. John Wiley & Sons.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392.

Sachs, M., Sen, D., Lu, J., and Dunson, D. (2021). Posterior computation with the Gibbs zig-zag sampler.

Schillings, C., Sprungk, B., and Wacker, P. (2020). On the convergence of the Laplace approximation and noise-level-robustness of Laplace-based Monte Carlo methods for Bayesian inverse problems. *Numerische Mathematik*, 145(4):915–971.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. (2016). Bayes and Big Data: The Consensus Monte Carlo Algorithm. *International Journal of Management Science and Engineering Management*, 11:78–88.

Shanno, D. F. (1970). Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656.

Skilling, J. (2006). Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833 – 859.

Sminchisescu, C. and Welling, M. (2011). Generalized darting monte carlo. *Pattern Recognition*, 44(10):2738 – 2748. Semi-Supervised Learning for Visual Content Analysis and Understanding.

Smith, J. W., Everhart, J. E., Dickson, W., Knowler, W. C., and Johannes, R. S. (1988). Using the ADAP learning Algorithm to Forecast the Onset of Diabetes Mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 261. American Medical Informatics Association.

Suwa, H. and Todo, S. (2010). Markov Chain Monte Carlo Method without Detailed Balance. *Phys. Rev. Lett.*, 105:120603.

Tanner, M. A. and Wong, W. H. (1987). The Calculation of Posterior Distributions by Data Augmentation. *Journal of the American Statistical Association*, 82(398):528–540.

Thin, A., Idrissi, Y. J. E., Corff, S. L., Ollion, C., Moulines, E., Doucet, A., Durmus, A., and Robert, C. P. (2021). NEO: Non Equilibrium Sampling on the

Orbits of a Deterministic Transform. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems.*

Tierney, L. (1994). Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics*, 22(4):1701 – 1728.

Tierney, L. and Kadane, J. B. (1986). Accurate Approximations for Posterior Moments and Marginal Densities. *Journal of the American Statistical Association*, 81(393):82–86.

Tjelmeland, H. and Hegstad, B. K. (2001). Mode Jumping Proposals in MCMC. *Scandinavian Journal of Statistics*, 28(1):205–223.

Turitsyn, K. S., Chertkov, M., and Vucelja, M. (2011). Irreversible Monte Carlo algorithms for efficient sampling. *Physica D: Nonlinear Phenomena*, 240(4):410–414.

Vaart, A. W. v. d. (1998). *Asymptotic Statistics.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Vasdekis, G. and Roberts, G. O. (2021). Speed Up Zig-Zag.

von Neumann, J. (1951). Various Techniques Used in Connection with Random Digits. In Householder, A. S., Forsythe, G. E., and Germond, H. H., editors, *Monte Carlo Method*, volume 12 of *National Bureau of Standards Applied Mathematics Series*, chapter 13, pages 36–38. US Government Printing Office, Washington, DC.

Wang, A. Q., Kolb, M., Roberts, G. O., and Steinsaltz, D. (2019). Theoretical properties of quasi-stationary Monte Carlo methods. *The Annals of Applied Probability*, 29(1).

Wang, A. Q., Pollock, M., Roberts, G. O., and Steinsaltz, D. (2021). Regeneration-enriched Markov processes with application to Monte Carlo. *The Annals of Applied Probability*, 31(2):703 – 735.

Wang, A. Q., Roberts, G. O., and Steinsaltz, D. (2020). An approximation scheme for quasi-stationary distributions of killed diffusions. *Stochastic Processes and their Applications*, 130(5):3193–3219.

Wang, X. and Dunson, D. B. (2014). Parallelizing MCMC via Weierstrass Sampler.

West, M. (1993). Approximating Posterior Distributions by Mixtures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(2):409–422.

Zhou, Y., Johansen, A. M., and Aston, J. A. (2016). Toward Automatic Model Comparison: An Adaptive Sequential Monte Carlo Approach. *Journal of Computational and Graphical Statistics*, 25(3):701–726.

Appendix A

Derivations

Here we derive (2.11) of subsection 2.5.2. Recall that the vector of random variables Y is defined via the transformation $Y = MX$, for M a $(d \times d)$ matrix with inverse M^{-1}. Row i and column j are denoted respectively as $M_{i,\cdot}$ and $M_{\cdot,j}$. By the chain rule,

$$\frac{\partial}{\partial y_j} \log \pi_Y(y) = \frac{\partial}{\partial y_j} x \cdot \nabla_x \log \pi_X(x).$$

By the product rule:

$$\frac{\partial^2}{\partial y_j^2} \log \pi_Y(y) = \frac{\partial^2}{\partial y_j^2} x \cdot \nabla_x \log \pi_X(x) + \frac{\partial}{\partial y_j} x \cdot \frac{\partial}{\partial y_j} \nabla_x \log \pi_X(x).$$

Since X is a linear transformation of Y, $\frac{\partial^2}{\partial y_j^2} x$ is zero so the first term may be ignored. The second term is

$$\begin{aligned}
\frac{\partial}{\partial y_j} x \cdot \frac{\partial}{\partial y_j} \nabla_x \log \pi_X(x) &= \sum_{i=1}^{d} \frac{\partial x_i}{\partial y_j} \frac{\partial}{\partial y_j} \left\{ \frac{\partial}{\partial x_i} \log \pi_X(x) \right\}, \\
&= \sum_{i=1}^{d} \frac{\partial x_i}{\partial y_j} \left[\frac{\partial^2}{\partial x_1 \partial x_i} \log \pi_X(x) \frac{\partial x_1}{\partial y_j} \right. \\
&\qquad + \frac{\partial^2}{\partial x_2 \partial x_i} \log \pi_X(x) \frac{\partial x_2}{\partial y_j} \\
&\qquad + \cdots \\
&\qquad \left. + \frac{\partial^2}{\partial x_d \partial x_i} \log \pi_X(x) \frac{\partial x_d}{\partial y_j} \right], \\
&= \sum_{i=1}^{d} \frac{\partial x_i}{\partial y_j} \left[H_{\log \pi_X}(x) \right]_{i,\cdot} \frac{\partial}{\partial y_j} x, \\
&= \left(\frac{\partial}{\partial y_j} x \right)^T H_{\log \pi_X}(x) \frac{\partial}{\partial y_j} x, \\
&= (M^{-1})_{\cdot,j}^T H_{\log \pi_X}(x) (M^{-1})_{\cdot,j}.
\end{aligned}$$

It then follows that $\Delta \log \pi_Y(y) = \sum_{j=1}^{d} \left(M^{-1}\right)_{\cdot,j}^T H_{\log \pi_X}(x) \left(M^{-1}\right)_{\cdot,j}$.

Appendix B

Distributions

B.1 Multivariate t-distribution

A multivariate t-distribution with ν degrees of freedom, mean m, scale matrix Σ and dimension d has probability density function:

$$\pi(x) \propto \left[1 + \frac{1}{\nu}(x-m)^T \Sigma^{-1}(x-m)\right]^{-(\nu+d)/2}.$$

Let $Q = \Sigma^{-1}$ be the precision matrix. For c a constant (the logarithm of the normalising constant) and writing $z := (x-m)^T Q(x-m)$, the energy, its gradient and Laplacian are:

$$U(x) = \frac{\nu+d}{2}\log\left\{1+\frac{z}{\nu}\right\} + c,$$

$$\nabla U(x) = \frac{\nu+d}{\nu+z} Q(x-m),$$

$$\Delta U(x) = \frac{\nu+d}{\nu+z}\left\{\operatorname{tr}(Q) - 2\frac{(x-m)^T Q^T Q(x-m)}{\nu+z}\right\}.$$

The partial rate for Brownian Motion Restore is then:

$$\tilde{\kappa}(x) = \frac{1}{2}\left(\frac{\nu+d}{\nu+z}\right)\left\{\left(\frac{\nu+d+2}{\nu+z}\right)(x-m)^T Q^T Q(x-m) - \operatorname{tr}(Q)\right\}.$$

B.2 Mixture of Gaussians

Let the target distribution be a mixture of n Gaussian distributions with weights $w_1, \dots w_n$, means $m_1, \dots, m_n$ and covariance matrices $\Sigma_1, \dots, \Sigma_n$. Each component of the mixture is denoted $\pi_i(x)$, so we have:

$$\pi(x) = \sum_{i=1}^{n} w_i \pi_i(x) = \sum_{i=1}^{n} w_i \mathcal{N}(x; m_i, \Sigma_i). \tag{B.1}$$

We then have:

$$\nabla \log \pi(x) = \frac{1}{\pi(x)} \nabla \pi(x) = \frac{\sum_{i=1}^n w_i \nabla \pi_i(x)}{\sum_{i=1}^n w_i \pi_i(x)} = \frac{\sum_{i=1}^n -w_i \pi_i(x) \Sigma_i^{-1}(x - \mu_i)}{\sum_{i=1}^n w_i \pi_i(x)}.$$

Let $z_i = -\Sigma_i^{-1}(x - \mu_i) = \Sigma_i^{-1}(\mu_i - x)$. Note that z_i is computed not by matrix multiplication but by solving the system of linear equations given by $\Sigma_i z_i = \mu_i - x$. Then

$$\nabla \log \pi(x) = \frac{\sum_{i=1}^n w_i \pi_i(x) z_i}{\sum_{i=1}^n w_i \pi_i(x)}.$$

The Laplacian of the log-density may be computed using the following equations. Here, the dot product of two vectors u and v is denoted $u \cdot v$ and column j of a matrix A is denoted $A_{\cdot,j}$.

$$\begin{aligned}
\frac{\partial}{\partial x_i} \log \pi(x) &= \frac{1}{\pi(x)} \frac{\partial}{\partial x_i} \pi(x), \\
\frac{\partial^2}{\partial x_i^2} \log \pi(x) &= \frac{\left(\frac{\partial^2}{\partial x_i^2}\pi(x)\right)\pi(x) - \left(\frac{\partial}{\partial x_i}\pi(x)\right)^2}{\pi(x)^2}, \\
\frac{\partial}{\partial x_i}\pi(x) &= -\sum_{j=1}^n w_j \pi_j(x) (\Sigma_j^{-1})_{\cdot,i} \cdot (x - \mu_j), \\
\frac{\partial^2}{\partial x_i^2}\pi(x) &= \sum_{j=1}^n w_j \pi_j(x) \left\{ \left[(\Sigma_j^{-1})_{\cdot,i} \cdot (x - \mu_j)\right]^2 - (\Sigma_j^{-1})_{ii} \right\}, \\
\Delta \log \pi(x) &= \sum_{i=1}^d \frac{\partial^2}{\partial x_i^2} \log \pi(x), \\
&= \frac{1}{\pi(x)^2} \sum_{i=1}^d \left(\frac{\partial^2}{\partial x_i^2}\pi(x)\right)\pi(x) - \left(\frac{\partial}{\partial x_i}\pi(x)\right)^2
\end{aligned}$$

B.3 Banana

Derivations related to the Banana distribution, defined in Section 4.4.1.

$$\begin{aligned}
\mathrm{Var}[Y_2] &= \mathrm{Var}[X_2 - bX_1^2 + ab], \\
&= \mathrm{Var}[X_2] + b^2 \mathrm{Var}[X_1^2], \\
&= \mathrm{Var}[X_2] + b^2\left(\mathbb{E}[X_1^4] - \mathbb{E}[X_1^2]^2\right), \\
&= \sigma_2^2 + b^2(3\sigma_1^4 - \sigma_1^4), \\
&= \sigma_2^2 + 2b^2\sigma_1^4.
\end{aligned}$$

The log-density of the transformed target is

$$\log \pi(y) = -\frac{1}{2}\left[\frac{y_1^2}{\sigma_1^2} + \frac{(y_2 + by_1^2 - ab)^2}{\sigma_2^2} + \frac{y_3^2}{\sigma_2^2} + \cdots + \frac{y_d^2}{\sigma_2^2}\right] + \text{const},$$

which has gradient:

$$\nabla \log \pi(y) = -\begin{pmatrix} y_1/\sigma_1^2 + 2by_1(y_2 + by_1^2 - ab)/\sigma_2^2 \\ (y_2 + by_1^2 - ab)/\sigma_2^2 \\ y_3/\sigma_2^2 \\ \vdots \\ y_d/\sigma_2^2 \end{pmatrix}.$$

Setting the gradient to zero, it can be seen that the mode is $y = (0, ab, 0, \ldots, 0)$. The Hessian of the energy is:

$$H_U(y) = \frac{1}{\sigma_2^2}\begin{pmatrix} \sigma_2^2/\sigma_1^2 + 6b^2y_1^2 + 2by_2 - 2ab^2 & 2by_1 & 0 & \cdots & 0 \\ 2by_1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & & 1 \end{pmatrix}.$$

At the mode, the Hessian matrix is:

$$H_U(y_{\text{mode}}) = \begin{pmatrix} \sigma_1^{-2} & 0 & \cdots & 0 \\ 0 & \sigma_2^{-2} & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & \sigma_2^{-2} \end{pmatrix}.$$

Therefore its inverse is:

$$H_U^{-1}(y_{\text{mode}}) = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & \sigma_2^2 \end{pmatrix}.$$

B.4 Funnel

The Funnel distribution is a product of Gaussians, with the variance of component i for $i = 2, \ldots, d$ depending on the first component:

$$\pi(x) = \mathcal{N}(x_1; 0, a^2) \prod_{i=2}^{d} \mathcal{N}(x_i; 0, e^{2bx_1}).$$

The energy, its first and second partial derivatives, gradient and Laplacian are then:

$$U(x) = \frac{x_1^2}{2a^2} + \frac{1}{2}\sum_{i=2}^{d} x_i^2 e^{-2bx_1} + \text{const};$$

$$\frac{\partial U}{\partial x_1} = x_1 a^{-2} - b e^{-2bx_1} \sum_{i=2}^{d} x_i^2;$$

$$\frac{\partial U}{\partial x_i} = x_i e^{-2bx_1}; i = 2, \ldots, d;$$

$$\frac{\partial^2 U}{\partial x_1^2} = a^{-2} + 2b^2 e^{-2bx_1} \sum_{i=2}^{d} x_i^2;$$

$$\frac{\partial^2 U}{\partial x_i^2} = e^{-2bx_1}; i = 2, \ldots, d;$$

$$\nabla U(x) = \left(x_1 a^{-2} - b e^{-2bx_1} \sum_{i=2}^{d} x_i^2, x_2 e^{-2bx_1}, \ldots, x_j e^{-2bx_1}, \ldots, x_d e^{-2bx_1} \right)^T;$$

$$\Delta U(x) = a^{-2} + e^{-2bx_1} \left[d - 1 + 2b^2 \sum_{i=2}^{d} x_i^2 \right].$$

Therefore the partial regeneration rate for Brownian Motion Restore may be written as:

$$\tilde{\kappa}(x) = \frac{1}{2} \Bigg\{ \left(x_1 a^{-2} - b e^{-2bx_1} \sum_{i=2}^{d} x_i^2 \right)^2 - a^{-2} + e^{-2bx_1} \left(1 - d + \left[e^{-2bx_1} - 2b^2 \right] \sum_{i=2}^{d} x_i^2 \right) \Bigg\}.$$

Fixing $x_2 = x_3 = \cdots = x_d = 0$ the partial rate is:

$$\begin{aligned} \tilde{\kappa}(x_1, 0, \ldots, 0) &= \frac{1}{2} \left\{ x_1^2 a^{-4} - a^{-2} + (1 - d) e^{-2bx_1} \right\}, \\ &= \frac{1 - d}{2} e^{-2bx_1} + O(x_1^2). \end{aligned}$$

Since $d > 1$, clearly $\tilde{\kappa}(x_1, 0, \ldots, 0) \to -\infty$ as $x_1 \to -\infty$.

B.5 Logistic Regression Model

The standard notation for a Logistic Regression model is that the data, consisting of predictor-response pairs, is denoted $\{(x_i, y_i)\}_{i=1}^n$ and the random variables of interest (the regression coefficients) are $\beta = (\beta_1, \beta_2, \ldots, \beta_d)^T$. We formulate the model here in terms of this standard notation, but in the main text the random variables $\beta = (\beta_1, \beta_2, \ldots, \beta_d)^T$ are denoted as $X = (X_1, X_2, \ldots, X_d)^T$, for reasons of consistency. The likelihood for the model is:

$$l(\{(x_i, y_i)\}_{i=1}^n | \beta) = \left[\prod_{i=1}^{n} \frac{1}{1 + \exp\{-y_i \beta^T x_i\}} \right].$$

Suppose a Gaussian prior with variance σ^2 is used for each component of β. Then the posterior distribution and its first and second derivatives are as follows. Here, $\mathbb{1}$ is the indicator function and c is a constant.

$$\begin{aligned}
\pi(\beta|x,y) &\propto \left[\prod_{i=1}^{n} \frac{1}{1+\exp\{-y_i\beta^T x_i\}}\right]\left[\prod_{j=1}^{d} \exp\left\{-\frac{\beta_j}{2\sigma^2}\right\}\right], \\
\log \pi(\beta|x,y) &= -\sum_{i=1}^{n} \log\left(1+\exp\{-y_i\beta^T x_i\}\right) - \frac{1}{2\sigma^2}\sum_{j=1}^{d} \beta_j^2 + c, \\
\frac{\partial \log \pi(\beta|x,y)}{\partial \beta_k} &= -\frac{\beta_k}{\sigma^2} + \sum_{i=1}^{n} \frac{y_i x_{i,k} \exp\{-y_i\beta^T x_i\}}{1+\exp\{-y_i\beta^T x_i\}}, \\
\frac{\partial^2 \log \pi(\beta|x,y)}{\partial \beta_j \beta_k} &= -\frac{\mathbb{1}\{j=k\}}{\sigma^2} - \sum_{i=1}^{n} \frac{y_i^2 x_{i,j} x_{i,k} \exp\{-y_i\beta^T x_i\}}{[1+\exp\{-y_i\beta^T x_i\}]^2}.
\end{aligned}$$

Following Gelman et al. (2008) and Chopin and Ridgway (2017), the response variables were defined on $\{-1,1\}$ and the predictors were standardized so that non-binary predictors have mean 0 and standard deviation 0.5, while binary predictors have mean 0 and range 1.

Appendix C

Abbreviations

Abbreviation	**Terms**
ACF	Autocorrelation function
AD	Automatic Differentiation
CH-Jumpar	Conformal Hamiltonian Jump Process Adjusted with Regenerations
ESJD	Expected Squared Jump Distance
GM-BMR	Gaussian Mixture Brownian Motion Restore
H-Jumpar	Hamiltonian Jump Process Adjusted with Regenerations
HMC	Hamiltonian Monte Carlo
IS	Importance Sampling
JPRS	Jump Process Restore Sampler
Jumpar	Jump Process Adjusted with Regenerations
MALA	Metropolis-Adjusted Langevin Algorithm
MCMC	Markov Chain Monte Carlo
MSE	Mean Square Error
NUTS	No-U-Turns Sampler
PDMP	Piecewise Deterministic Markov Process
RD-Jumpar	Random-Direction Jump Process Adjusted with Regenerations
RMSE	Root Mean Square Error
RWM	Random Walk Metropolis
SMC	Sequential Monte Carlo