Bayesian Computation for Stochastical Dynamical Systems

Scott Phillips

(Supervisor: Andrew Golightly)

Durham University

May 2, 2025

Plagiarism Declaration

This piece of work is a result of my own work, and I have complied with the Department's guidance on multiple submission and on use of AI tools. Material from the work of others not involved in the project has been acknowledged, quotations and paraphrases suitably indicated, and all uses of AI tools have been declared.

Code and Reproducibility Statement

All code used to produce the figures and tables in this report are available in the following GitHub repository:

`https://github.com/ScottPhillips23uni/Project-IV-code`

This repository contains the full implementation of the PMMH and Gibbs sampling methods, as well as all relevant plotting scripts and supporting files. These allow for full reproduction of the results presented in this report.

Contents

Chapter 1

Introduction

Markov chain Monte Carlo (MCMC) algorithms [1] are extremely successful when investigating posterior distributions in Bayesian statistics. Many methods of MCMC can be used when undertaking such investigations, typically depending on the context of the situation to which they are applied. The goal of MCMC is to estimate a target distribution by generating a Markov chain whose stationary distribution is the target. An example that we will investigate throughout this report is the pseudo-marginal Metropolis-Hastings (PMMH) algorithm [2]. This method is best used when trying to investigate a posterior that has an intractable data likelihood.

PMMH provides a way to bypass the intractable data likelihood. It does this by unbiasedly estimating the target density. The target densities in the numerator and denominator of the resulting acceptance probability are replaced by the unbiased estimates of the intractable data likelihood. Fortunately, this results in an algorithm that still produces estimates from the original target distribution.

One such example of a system with an intractable data likelihood is a Stochastic Dynamical System. Stochastic Differential Equations (SDE) are a way of modelling these random processes that evolve through time. Consequently, they have been applied in areas such as finance [3], biology [4] and epidemiology [5]. As such, the PMMH algorithm is one of the few desirable methods that simulate draws from a target derived from an SDE.

This report addresses the derivation and theoretical validation of the pseudo-marginal Metropolis-Hastings (PMMH) algorithm, followed by its application to Stochastic Kinetic Models (SKMs), which are commonly used in biological modelling [6]. The SKM of interest is the Lotka-Volterra (LV) model [7]. The way this model is investigated in its application to the LV model is through the introduction of a partially observed data set with unknown parameters. Due to this data set being partially observed, a partition is implemented between the observed data points to aid with the estimation of the unbiased estimator. The PMMH algorithm will be applied with the use of an importance sampler to unbiasedly estimate the intractable data likelihood. The proposal mechanism for this estimator will be the Durham and Gallant bridge construct [8] applied over the partitioned interval. Following this, it then remains to evaluate and compare the efficiency of the algorithm to other algorithms, in particular the Gibbs sampler.

This report will assess the efficiency of the PMMH algorithm through the analysis of the Effective Sample Size (ESS), the Expected Squared Jump Distance (ESJD), and the optimal acceptance probability. The optimal acceptance probability will be taken to be values for the general Metropolis-Hastings (MH) algorithm initially and then further enhanced through the use of the ESJD to generate optimal acceptance probabilities for the PMMH algorithm in particular. Ultimately, this work contributes a practical evaluation of PMMH for partially observed SDEs, giving insights into its robustness, computational demands, and limitations when applied to real world models.

Chapter 2

Brownian Motion

To commence our discussion of the application of Bayesian inference to stochastic dynamical systems, we set about explaining the necessary prerequisite material.

2.1 Brief Introduction to Stochastic Differential Equations

We begin by defining a Stochastic Differential Equation (SDE) [9], which can be best understood by first recalling the form of an Ordinary Differential Equation (ODE):

$$dX_t = \alpha(X_t, \theta)dt \tag{2.1}$$

with some initial condition on X_0. Equation 2.1 can then be furthered by introducing the concept of two key components that comprise an SDE: the 'drift' and the 'noise'. The 'drift' corresponds to the deterministic aspect, the ODE. The 'noise' corresponds to the stochastic aspect, which results in the continuation of the ODE to an SDE. This noise can be implemented in the ODE as follows:

$$dX_T = \alpha(X_t, \theta)dt + \beta(X_t, \theta)dW_t. \tag{2.2}$$

Equation 2.2 is the standard form for SDEs where X_t is a stochastic process satisfying the Markov property and represents the state of the SDE at time t. α and β represent functions that depend on the stochastic process X_t and the parameters of the model θ. Finally, W_t represents the Brownian motion, or more specifically, the one-dimensional Brownian process referred to as the Wiener process. It is also worth noting that in the absence of any noise, i.e. $\beta(X_t, \theta) = 0$, the SDE reverts to being the ODE in Equation 2.1.

To showcase the nature of how ODEs and SDEs relate, it is beneficial to consider an ODE and SDE simulation with the same 'drift' components. In Figure 2.1, it is clear that the SDE closely follows the trajectory of the ODE with some randomness incorporated. This illustration demonstrates the need to incorporate SDEs into models of real-world systems due to the randomness that cannot be captured by an ODE.

For more information on ODEs, refer to [10]. It is again worth noting that not all SDEs are analytically tractable. It is a standard procedure to incorporate numerical methods to obtain meaningful inference from SDEs. For further details on this, please refer to Chapter 4, Stochastic Kinetic Models.

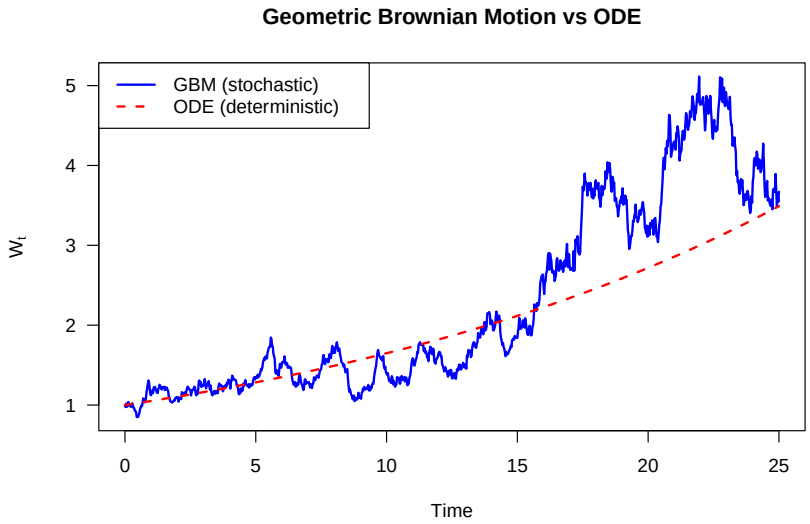

Figure 2.1: Geometric Brownian motion with parameters $\mu = 0.05, \sigma = 0.2$ and its ODE.

2.2 Brownian Motion and Stochastic Differential Equations

2.2.1 Definition and Properties

We now continue our discussion of Brownian motion by writing out the fundamental form of the SDE of interest:

$$dX_t = \mu X_t dt + \sigma X_t dW_t. \tag{2.3}$$

This particular model is called Geometric Brownian motion (GBM) [11] and it is a useful example to consider when discussing analytic solutions of SDEs. μ is the 'drift' of the process, σ is the 'noise' and as before W_t is the Wiener process. To understand why not all SDEs have tractable solutions, it is important to discuss the properties of the Wiener process, which are as follows [12]:

- **Initialization**: $\mathbb{P}(W_0 = 0) = 1$.
- **Normality**: $W_{t_1} - W_{t_0} \sim N(0, t_1 - t_0)$ for all $0 < t_0 < t_1 < \infty$.
- **Independence**: $W_{t_2} - W_{t_1} \perp W_{t_1} - W_{t_0}$ for all $0 < t_0 < t_1 < t_2 < \infty$.

From the normality property, it is possible to infer some intuitive results to aid in understanding Brownian motion:

- $\mathbb{E}[W_{t+h} - W_t] = 0$, this shows that there is no drift, giving an unbiased production of random values.
- The distribution of W_t has stationary increments given by the variance dependent only on the time step.
- Combining the last two inferences gives the result $W_t \sim N(0, t)$. Implying that the Wiener process has Gaussian innovations.

Given these properties, one might assume that Equation 2.3 would be a solvable system in the limit if Brownian motion paths were differentiable in time but, unfortunately, there is no stochastic process that has both differentiability and the previously stated properties. However, there does exist a stochastic process that does have continuous sample paths in time — Brownian motion. This property allows for Equation 2.3 to be solved via an integral equation:

$$X(t) = x_0 + \int_0^t \mu X_s ds + \int_0^t \sigma X_s dW_s, \tag{2.4}$$

or more generally to solve Equation 2.2 via an integral equation:

$$X(t) = x_0 + \int_0^t \alpha(X_s, \theta) ds + \int_0^t \beta(X_s, \theta) dW_s. \tag{2.5}$$

This integral requires the use of an Itô stochastic integral to solve it due to the integration with respect to a stochastic process. Now there are two ways to solve this with varying levels of success, analytically or numerically. Analytic solutions unfortunately are not tractable for all forms of α and β so this is where numerical methods would be employed, typically via the Euler-Maruyama (EM) approximation. Fortunately, Equation 2.4 is analytically tractable and the solution to this will be found in the next section.

2.2.2 Practical Implementation

Before moving on to the next section, it is worth considering several simulations of Brownian motion to give an idea of its empirical behaviour. This can be done via the following algorithm:

Brownian Motion with N Iterations

1. Let s_0 be the starting point. Initialise $W_0 = 0$ and introduce a partition $0 = t_0 < t_1 < .. < t_N = T$.
2. For each iteration i, simulate $Z_i \sim N(0, 1)$ and calculate $W_{t_i} = W_{t_{i-1}} + \sqrt{\frac{T}{N}} Z_i$.

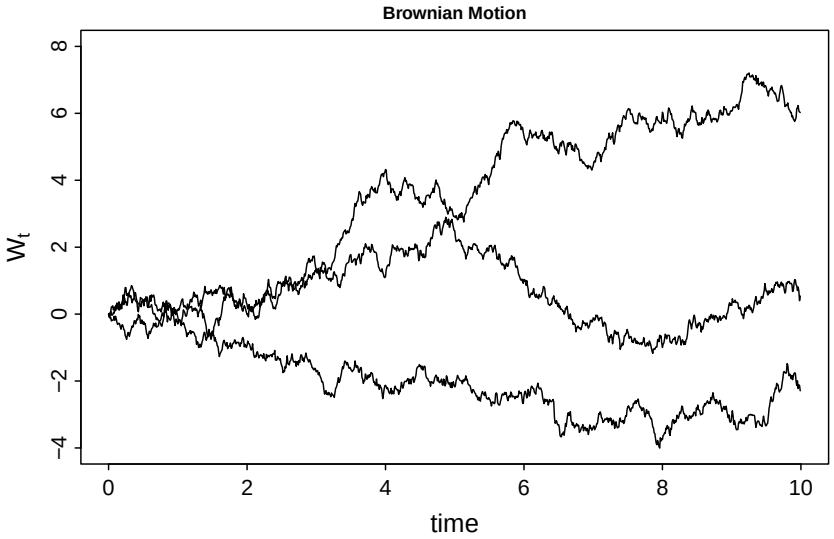

Figure 2.2: 3 simulations of Brownian motion with 1000 iterations over a time period equal to 10.

The implementation of Brownian Motion in R is illustrated in Figure 2.2. It can be seen that as the number of simulations goes to infinity, the mean of these simulations will be a horizontal line along the x-axis, corroborating the properties mentioned earlier.

2.3 Analytic Solution

Solving Equation 2.4 analytically requires first discussing an Itô integral, which has the standard formulation:

$$\int_0^t f(X_s, \theta) dW_s. \tag{2.6}$$

The Itô integral requires extensive definitions and results to fully enable its use and ensure its validity. We will mention some of these to aid in understanding the integral and providing context to some previously mentioned terms:

Definition 1.1 σ-algebra [13]

A σ-algebra $\mathcal{F}$ of subsets of X is a collection $\mathcal{F}$ of subsets X satisfying the following conditions:

1. $\emptyset \in \mathcal{F}$.
2. If B $\in \mathcal{F}$ then its complement b^c is also in $\mathcal{F}$.
3. If $B_1, B_2, ...$ is a countable collection of sets in $\mathcal{F}$ then their Union $\bigcup_{n=1}^{\infty} B_n$ is also in $\mathcal{F}$.

Definition 1.2 Stochastic Process [9]

A stochastic process is a parametrised collection of random variables $(X_t)_{t \in T}$ defined on a probability space $(\Omega, \mathcal{F}, P)$ and assuming real values in $\mathbb{R}$.

Definition 1.3 Markov Process [14]

Consider a stochastic process $X_i \forall i \in (0, T)$, this process is Markov if it satisfies the following:

$$\mathbb{P}(X_i | X_{i-1}, ..., Z_0) = \mathbb{P}(X_i | X_{i-1}).$$

Definition 1.4 Itô process [9]

An Itô process is a stochastic process X_t of the form:

$$X_t = x_0 + \int_0^t \alpha(X_s, \theta) ds + \int_0^t \beta(X_s, \theta) dW_s,$$

where β is such that

$$\mathbb{P}(\int_0^t \beta(X_s, \theta)^2 ds < \infty \text{ for all } t \geq 0) = 1.$$

The above mentioned definitions can now be utilised in deriving the key result for solving Itô integrals.

Lemma 1.1 Itô's Lemma [9]

Consider an Itô process X_t, let $g(X_t, t)$ be a twice differentiable continuous process then let Y_t be such that:

$$Y_t = g(X_t, t).$$

This also turns out to be an Itô process and we now have the final result that

$$dY_t = \frac{\partial g}{\partial t}(X_t, t)dt + \frac{\partial g}{\partial x}(X_t, t)dX_t + \frac{1}{2}\frac{\partial^2 g}{\partial x^2}(X_t, t) \cdot (dX_t)^2$$

where we use the following rules to compute the result

$$dt \cdot dt = dt \cdot dW_t = dW_t \cdot dt = 0, dW_t \cdot dW_t = dt.$$

The key distinction between tractable and intractable SDEs, are the properties of α and β. These functions must typically be linear in their parameters and independent of time. Returning to Geometric Brownian Motion, it is now possible to solve for the analytic solution by first applying Itô's formula to Equation 2.3:

Let $Y_t = \log(X_t)$ where $g(X_t, t) = \log(X_t)$

It is clear to see that:

$$\frac{\partial Y_t}{\partial X_t} = \frac{1}{X_t}, \frac{\partial^2 Y_t}{\partial X_T} = -\frac{1}{2X_T^2}, \frac{\partial Y_T}{\partial t} = 0.$$

Applying this to Ito's lemma results in:

$$dY_t = \frac{1}{X_t}dX_t - \frac{1}{X_t^2}(dX_t)^2.$$

To solve this, it remains that dX_t^2 needs to be calculated and Itô's lemma applied to give:

$$\begin{aligned} dX_t^2 &= dX_t \cdot dX_t \\ &= (\mu X_t dt + \sigma X_t dW_t)^2 \\ &= (\mu^2 X_t^2 dt^2 + 2\mu\sigma X_t^2 dt dW_t + \sigma^2 X_t^2 dW_t^2) \\ &= \sigma^2 X_t^2 dt. \end{aligned}$$

Substituting in dX_t and dX_t^2 gives:

$$dY_T = \frac{1}{X_t}(\mu X_t dt + \sigma X_t dW_t) - \frac{1}{2X_t^2}(\sigma X_T)^2 dt = \mu dt + \sigma dW_t - \frac{\sigma^2}{2}dt.$$

Solving the above by integrating between 0 and t results in:

$$\int_0^t dY_t = \int_0^t (\mu - \frac{\sigma^2}{2})dt + \int_0^t \sigma dW_t.$$

This gives:

$$Y_t - Y_0 = (\mu - \frac{\sigma^2}{2})t + \sigma(W_t - W_0).$$

Utilising the fact that $W_0 = 0$ and the previously calculated $Y_t = log(X_t)$ results in:

$$\log X_T = \log X_0 + (\mu - \frac{\sigma^2}{2})t + \sigma W_t.$$

Now rearranging and taking exponentials gives:

$$X_t = X_0 \exp\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t.$$

Finally applying the fact that $W_t \sim N(0, t)$ finishes the computation to give:

$$X_t|X_0 \sim \log N(\log x_0 + (\mu - \frac{\sigma^2}{2})t, \sigma^2 t)$$

$$X_t|X_s \sim \log N(\log x_s + (\mu - \frac{\sigma^2}{2})(t-s), \sigma^2(t-s)) \tag{2.7}$$

where $\log N(m, v^2)$ denotes the log-normal distribution, and whose logarithm gives a normal distribution with mean m and variance v^2. The result calculated in Equation 2.7 is the solution to GBM and can be used to generate draws for X_t given μ, σ and X_{t-1}. This can be done by applying the following algorithm:

Analytic Geometric Brownian Motion with N Iterations

1. Let x_0 be the starting point. Initialise $X_0 = x_0$ and introduce a partition $0 = t_0 < t_1 < .. < t_N = T$.
2. For iteration i, simulate $Z_i \sim N(0, 1)$, then $X_{t_i} = \log(\log(X_{t_{i-1}}) + (\mu - \frac{\sigma^2}{2})\frac{T}{N} + \sigma\sqrt{\frac{T}{N}}Z_i)$.

This analysis has provided an analytic solution to GBM for which analysis can be carried out. Please refer to [11] for further reading on GBM and the validity of its application to real model systems.

2.4 Numerical Solution

To give some context to the analytic solution and demonstrate the relative strengths and weaknesses of having an analytic compared to a numerical solution, it is prudent to introduce some numerical methods. Numerical approximations can be applied to all SDEs, regardless of the nature of α and β. One such method used throughout this paper is the Euler-Maruyama approximation [15]. There are other equally valid methods, such as the Milstein method [15] and the Itô-Taylor expansion [16], but these are not necessary to investigate now. The reason for utilising the EM approximation is due to its strong convergence rate of $\frac{1}{2}$ [17]. This makes it a very suitable approximation as the limit will converge to the exact function. For further reading on the topic of convergence, please refer to [15]. To derive the EM approximation, first recall an Euler-approximation [18] for a standard ODE:

$$\frac{dX_t}{dt} = \alpha(X_t, t),$$

$$X_t = X_s + \alpha(X_s, s)(t-s) \quad \text{for} \quad s < t.$$

Now the logical extension of this to Equation 2.2 gives:

$$X_t = X_s + \alpha(X_s, s)(t-s) + \beta(X_s, s)(W_t - W_s).$$

Applying what we know about the Wiener process gives:

$$X_t|X_s \sim N(x_s + \alpha(x_s, s)(t-s), \beta(x_s, s)^2(t-s)) \quad \text{for} \quad s < t. \tag{2.8}$$

The resulting EM transition density for GBM can be found to be:

$$X_t|X_s \sim N(x_s + \mu x_s(t-s), \sigma^2 x_s^2(t-s)). \tag{2.9}$$

This is the resulting transition density for the Euler-Maruyama approximation, once again realisations from this algorithm can be simulated through the following algorithm.

Euler-Maruyama Geometric Brownian Motion with N Iterations

1) Let x_0 be the starting point. Initialise $X_0 = x_0$ and partition the interval $0 = t_0 < t_1 < \ldots < t_N = T$.
2) For iteration i, simulate $Z_i \sim N(0,1)$ and calculate $X_{t_i} = x_{t_{i-1}} + \mu x_{t_{i-1}} + \sigma x_{t_{i-1}} \sqrt{\frac{T}{N}} Z_i$.

2.5 Comparison of Analytical and Numerical

Having derived both analytical and numerical methodologies in a general setting and applying the methods to GBM, it is now beneficial to compare the outputs and properties of the methods to showcase that despite analytical solutions being desirable and overall better, approximations are a very valid and efficient alternative.

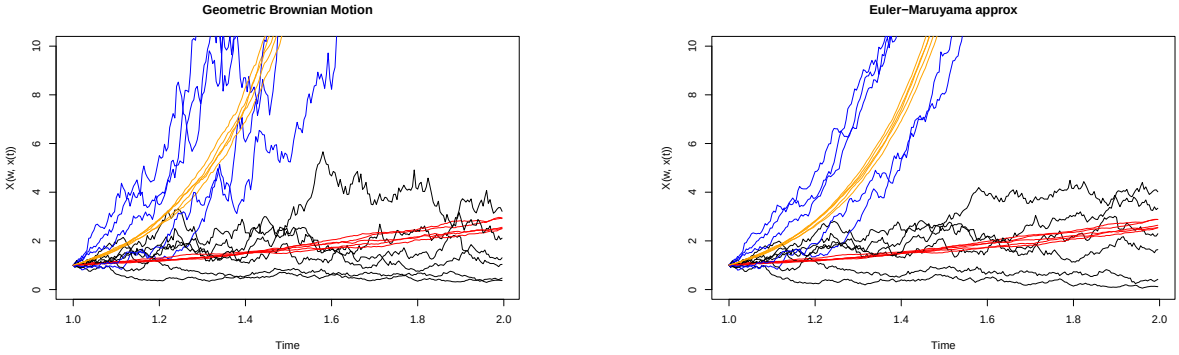

Figure 2.3: The colours above correspond to different conditions, red : $\mu = 1, \sigma = 0.1$, blue : $\mu = 5, \sigma = 1$, black : $\mu = 1, \sigma = 1$, orange : $\mu = 5, \sigma = 0.1$, the solid lines are simulations of each GBM, with the same driving Brownian motion used for EM and GBM simulations.

Both the analytic and numerical algorithms have been applied to produce the simulations shown in Figure 2.3, using the same driving Brownian motion for each parameter regime to showcase how similar the two methods are. It is evident that the EM is excellent for small $t - s$. However, as $t - s$ grows, there are substantial differences in the two schemes. It could be helpful to inspect the kernels of both distributions, these are:

$$\text{Analytic: } p(x_t|x_s) \sim \frac{1}{2x_s\pi\sqrt{(\sigma^2 x_s(t-s)}} \exp - \frac{(\log(x_t) - x_s - \mu x_s(t-s))^2}{2\sigma^2 x_s(t-s)} \tag{2.10}$$

$$\text{Euler-Maruyama: } p(x_t|x_s) \sim \frac{1}{2\pi\sqrt{(\sigma^2 x_s(t-s)}} \exp - \frac{(x_t - x_s - \mu x_s(t-s))^2}{2\sigma^2 x_s(t-s)} \tag{2.11}$$

From the kernel, stark differences become immediately apparent. The EM kernel allows for negative values, something not permitted by GBM. Then, somewhat more subtly, the linear approximation of

the EM scheme affects the variance of our schema. In particular, for large simulations, the error created by the linear approximation cumulates and will result in a poor approximation for GBM. These observations can be summarised as follows:

1. GBM grows exponentially and the EM approximation is based on a linear approximation, as seen in the transition densities. EM will fail to capture the true nature of GBM for large $t - s$ as the errors for each iteration will be cumulatively compounded, particularly in the variance where a huge difference can be seen in the simulations.

2. For large simulations that span large $t - s$, EM can give negative results and this is not permitted by GBM.

3. EM approximation is very sensitive to time-step length — too large and it will not accurately capture the nature of GBM, but too small and all errors are compounded and computation time becomes infeasible.

This comparison concludes our discussion of SDEs, Brownian motion, and numerical/analytical solutions to GBM. In the next section we will lay the groundwork for an inference scheme on the parameters that drive the SDEs and subsequently apply this to the main model of interest, SKMs.

Chapter 3

Pseudo-Marginal Metropolis-Hastings

In this section we will address the pseudo-marginal Metropolis-Hastings algorithm, this scheme is used when the observed data likelihood function, previously referred to as the transition density, is intractable but can be estimated to produce unbiased estimates thereof. We will also build the necessary framework to introduce such an algorithm.

3.1 Markov Chain Monte Carlo

Monte Carlo methods are used when the analytic posterior distributions are intractable, relying on observed values and using the properties of the data to define a distribution. The addition of Markov chains to this process helps it find a distribution of a time dependent process. This defines a MCMC process, the core of the Metropolis-Hastings (MH) algorithm. Before proceeding with the PMMH algorithm, there are some essential results [19] that must be stated:

Definition 3.1 Markov Chain

A Markov Chain is a stochastic process, $X_0, X_1, X_2, ...$, with the property that past states are independent of future states given the present state. Consider a state space $\mathcal{X}$ to be the set of possible states for the stochastic processes. For $A \subset \mathcal{X}$ it can be see that:

$$\begin{aligned}&\mathbb{P}(X_{n+1} \in A | X_n = x, X_{n-1} = x_{n-1}, ..., X_0 = x_0)\\ &= \mathbb{P}(X_{n+1} \in A | X_n = x) \quad \forall x_{n-1}, x_{n-2}, ..., x_0 \in \mathcal{X}.\end{aligned}$$

Definition 3.2 μ-Irreducibility

If there exists a distribution μ such that for a state space $\mathcal{X}$ and for $A \subseteq \mathcal{X}$ with $\mathbb{P}(X \in A) > 0$ for all $x \in \mathcal{X}$ then there exists a positive integer $n = n(x, A)$ such that $\mathbb{P}(X_n \in A | X_0 = x) > 0$.

Definition 3.3 Aperiodicity

A μ-irreducible Markov chain is aperiodic on the state space $\mathcal{X}$ if there do not exist $d > 1$ and disjoint subsets $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_d \subseteq \mathcal{X}$ with $\mathbb{P}(X_{n+1} \in \mathcal{X}_{i+1} | X_n \in \mathcal{X}_)) = 1$ $(i = 1, ..., d-1)$ and $\mathbb{P}(X_{n+1} \in \mathcal{X}_1 | X_n \in \mathcal{X}_\lceil) = 1$, and such that $\mathbb{P}_\mu(X \in \mathcal{X}_i) > 0$.

Definition 3.4 Stationary Distribution

A stationary distribution π is a set of densities that must satisfy for all $x, y \in \mathcal{X}$:

$$\pi(y) = \int_{\mathcal{X}} p(y|x)\pi(x)dx.$$

Definition 3.5 Detailed Balance

A Markov chain satisfies detailed balance if $\forall x, y$:

$$\pi(x)p(y|x) = \pi(y)p(x|y).$$

Theorem 3.1 Limiting Distribution

If a Markov chain is μ-irreducible and aperiodic, and has a proper stationary distribution, π, then for all $x \in \mathcal{X}$:

$$\mathbb{P}(X_n \in A|X_0 \in x) \to \mathbb{P}_\pi(X \in A).$$

Remark:
If a Markov chain with transition density P satisfies detailed balance with distribution ν then $\nu = \pi$ is a stationary distribution of the chain.

3.1.1 Metropolis-Hastings Algorithm

Utilising the definitions stated it is now possible to construct a Markov chain with stationary distribution equal to the posterior distribution. With these foundations set, the MH Algorithm [20] can be stated:

Metropolis-Hastings Algorithm with N Iterations

1. Initialise the chain to $\theta^{(0)} = (\theta_1^{(0)}, ..., \theta_d^{(0)})$ somewhere in the support of $\pi(\theta)$. Set iteration counter to 1.
2. Generate a proposed value θ^* using the proposal kernel $q(\theta^*|\theta^{j-1})$.
3. Evaluate the acceptance probability $\alpha(\theta^*|\theta^{(j-1)})$ of the proposed move, defined by $\alpha(\theta^*|\theta) = \min\left\{1, \frac{\pi(\theta^*)q(\theta|\theta^*)}{\pi(\theta)q(\theta^*|\theta)}\right\}$.
4. Put $\theta^j = \theta^*$ with probability $\alpha(\theta^*|\theta^{(j-1)})$; otherwise put $\theta^{(j)} = \theta^{(j-1)}$.
5. If $j = N$ stop, otherwise put j to j + 1 and go to step 2.

The MH algorithm is a very effective and efficient algorithm, it uses and satisfies the previously stated definitions to ensure that the transition kernel with acceptance probability $\alpha(\theta^*|\theta^{(j-1)})$ converges to the target distribution $\pi(\theta)$. The MH algorithm does this by generating μ-irreducible and aperiodic Markov chains that satisfy detailed balance. A more in-depth analysis of this is better suited for the main algorithm of study, the PMMH algorithm. The MH algorithm opens the door for the simulation of many distributions that otherwise would have been inaccessible.

There is a variation of the MH algorithm called the random-walk metropolis (RWM) algorithm [21]. This takes a proposal of the form $\theta^{(j)} = \theta^{(j-1)} + w^{(j)}$ where $w^{(j)}$ is a random vector generated from a density $g(\cdot)$. The proposal kernel can be found to be symmetric and results in an acceptance probability of the form:

$$\alpha(\theta^*|\theta) = \min\left\{1, \frac{\pi(\theta^*)}{\pi(\theta)}\right\}.$$

This acceptance probability is not dependent on the proposal distribution, a fact that will be essential for the following PMMH schema.

3.2 Derivation

With the framework for the MH algorithm set, it is now possible to extend this to the PMMH algorithm. As mentioned before, this algorithm exists due to the intractability of the likelihood function when trying to simulate draws from a posterior. The PMMH algorithm circumvents this issue by using Monte Carlo methods to construct an unbiased estimator of the likelihood.

3.2.1 Importance Sampling

The Monte Carlo method used here will be importance sampling [22], this is done by using Monte Carlo integration to find the expectation of a function $h(X)$ with respect to a distribution $f(x)$ which is difficult to sample from. This is solved by introducing a proposal distribution $g(x)$, the expectation can now be seen as:

$$\mathbb{E}[h(X)] = \mu_h = \int_{\mathcal{X}} h(x)f(x)dx = \int_{\mathcal{X}} \frac{h(x)f(x)}{g(x)}g(x)dx.$$

Which is equivalent to $\mathbb{E}_g[\frac{h(X)f(X)}{g(X)}] = \mathbb{E}_g[h(X)w(X)]$ where $w(X) = \frac{f(X)}{g(X)}$ represents the weight function. Given iid draws $X_1, ..., X_N$ from $g(\cdot)$, it is then possible to construct an unbiased and consistent importance sampling estimator:

$$\hat{\mu}_h = \frac{1}{N}\sum_{i=1}^{N} h(X_i)w(X_i).$$

3.2.2 Pseudo-Marginal Algorithm

The MH algorithm defined earlier cannot accommodate the intractable likelihood function. However, the PMMH algorithm can. It does this by producing an unbiased estimator for the likelihood and then proceeding as normal with the MH algorithm.

Due to the incorporation of an unbiased estimator, it is essential to implement some auxiliary variable $u \sim g(\cdot)$ to account for the randomness incorporated via the estimation. To ensure $\hat{\pi}_u(\theta)$ is an unbiased and non-negative estimator of $\pi(\theta)$, there are several restrictions that must be imposed:

$$\mathbb{E}_g[u] = 1, u > 1.$$

This results in a new target and proposal of the form $\pi(\theta, u)$ and $q(\theta^*, u^*|\theta, u)$ which are as follows:

$$\pi(\theta, u) \propto \hat{\pi}_u(\theta)g(u) = \pi(\theta)ug(u), \quad q(\theta^*, u^*|\theta, u) = q(\theta^*|\theta)g(u^*).$$

The resulting acceptance probability thus takes the form:

$$\begin{aligned}\alpha(\theta^*, u^*|\theta^{(j-1)}, u^{(j-1)}) &= \min\left\{1, \frac{\pi(\theta^*, u^*)q(\theta^{(j-1)}, u^{(j-1)}|\theta^*, u^*)}{\pi(\theta, u)q(\theta^*, u^*|\theta^{(j-1)}, u^{(j-1)})}\right\} \\ &= \min\left\{1, \frac{\pi(\theta^*)u^*q(\theta^{(j-1)}|\theta^*)}{\pi(\theta)uq(\theta^*|\theta^{(j-1)})}\right\}.\end{aligned}$$

Using the previous notation, the PMMH algorithm [2] is as follows:

PMMH Algorithm with N iterations

1. Initialise the chain to $\theta^{(0)} = (\theta_1^{(0)}, ..., \theta_d^{(0)})$ somewhere in the support of $\pi(\theta)$. Draw an initial auxiliary variable $u^{(0)} \sim g(\cdot)$ and compute the noisy target $\hat{\pi}_{u^{(0)}}(\theta^{(0)}) = \pi(\theta^{(0)})u^{(0)}$. Set iteration counter to $j = 1$.
2. Generate a proposed value θ^* from the proposal distribution $q(\theta^*, u^*|\theta, u) = q(\theta^*|\theta)g(u^*)$. Independently draw a new auxiliary variable $u^* \sim g(\cdot)$ and compute a new noisy target $\hat{\pi}_{u^*}(\theta^*) = \pi(\theta^*)u^*$.
3. Evaluate the acceptance probability $\alpha(\theta^*, u^*|\theta^{(j-1)}, u^{(j-1)}) = \min\left\{1, \frac{\pi(\theta^*)u^* q(\theta^{(j-1)}|\theta^*)}{\pi(\theta)u q(\theta^*|\theta^{(j-1)})}\right\}$.
4. Put $\theta^j = \theta^*$ with probability $\alpha(\theta^*, u^*|\theta^{(j-1)}, u^{(j-1)})$; otherwise put $\theta^{(j)} = \theta^{(j-1)}$.
5. If $j = N$ stop, otherwise put j to $j + 1$ and go to step 2.

3.3 Validity

As alluded to in the earlier discussion of the MH algorithm, it is important to show that valid results are produced by the PMMH algorithm. Due to the target being of the form $\pi(\theta, u)$, we will first investigate whether the simulations produced by the algorithm are from the desired target $\pi(\theta)$. After confirming this, it remains to show that $\pi(\theta, u)$ is a stationary distribution of the Markov chains generated by the PMMH algorithm and that these Markov chains converge.

3.3.1 Marginalisation of Posterior

Consider the target of the form $\pi(\theta, u)$ with the properties discussed earlier, the simulations can be found to result from:

$$\begin{aligned}\int \pi(\theta, u)du &\propto \int \pi(\theta)ug(u)du \\ &= \pi(\theta)\int ug(u)du \\ &= \pi(\theta)\mathbb{E}_g(u) \\ &\propto \pi(\theta),\end{aligned}$$

where line 4 is given by the restrictions imposed on the expectation. This shows that when marginalising out the random variable, the desired target is obtained. It is important for the estimator to be unbiased when carrying out the computation previously mentioned. If there were any bias present, it would skew the marginalisation, resulting in simulated draws not from the correct target.

The algorithm naturally carries out this integral as it explores both u and θ together, updating u by generating draws from $g(\cdot)$ for each iteration. This results in an implicit calculation of the integral in the PMMH algorithm.

3.3.2 Detailed Balance

Furthering the proof of the PMMH schema, it is now necessary to show that $\pi(\theta, u)$ is a stationary distribution. This is done by producing a transition kernel for the process and showing detailed balance holds. For PMMH the transition kernel takes the form:

$$p(\theta^*, u^*|\theta, u) = \alpha(\theta^*, u^*|\theta, u)q(\theta^*, u^*|\theta, u).$$

Using previously defined quantities in section 2.2.2. Now checking that this kernel satisfies detailed balance:

$$\pi(\theta, u)p(\theta^*, u^*|\theta, u) = \pi(\theta, u)q(\theta^*|\theta)g(u^*)\min\left\{1, \frac{\pi(\theta^*)u^*q(\theta|\theta^*)}{\pi(\theta)uq(\theta^*|\theta)}\right\}.$$

Finally, rearranging this into the expression:

$$\min\{\pi(\theta, u)q(\theta^*|\theta)g(u^*), \pi(\theta^*, u^*)q(\theta|\theta^*)g(u)\}.$$

which can clearly be seen to be symmetric. It is a common misconception that u breaks symmetry, however, in the schema u is treated as a parameter so it too must be symmetric. Hence $\pi(\theta, u)$ satisfies detailed balance and through the use of Lemma 3.1, conclude that $\pi(\theta, u)$ is the stationary distribution of the algorithm.

3.3.3 Convergence

All that remains to show now, to be able to apply Theorem 3.1, is that the chain converges. This is done by showing that the Markov chain is μ-irreducible and aperiodic. In proving this, we opt for the simple case where we restrict the densities in the sample space $\mathcal{X}$ to satisfy $\pi(\theta) \leq C$ for some $C > 0$.

μ-**irreducibility**: Define the probability of moving from (θ, u) to a measurable set A by:

$$\begin{aligned}
\mathbb{P}((\theta^1, u^1) \in A|(\theta^0, u^0)) &= \int_A q(\theta^*|\theta^0)g(u^*)\min\left\{1, \frac{\pi(\theta^*)u^*q(\theta^0|\theta^*)}{\pi(\theta)uq(\theta^*|\theta^0)}\right\}d\theta^* du^* \\
&= \int_A \min\left\{\frac{q(\theta^*|\theta)g(u^*)}{\pi(\theta^*, u^*)}, \frac{q(\theta|\theta^*)g(u)}{\pi(\theta, u)}\right\}\pi(\theta^*, u^*)d\theta^* du^* \\
&\geq \frac{1}{C}\int_A \min\{q(\theta^*|\theta)g(u^*), q(\theta|\theta^*)g(u)\}\pi(\theta^*, u^*)d\theta^* du^* \\
&= \frac{\mathbb{P}_\mu(A)}{C} \cdot \mathbb{E}_{\theta^*, u^* \sim \pi(\cdot)}[\min\{q(\theta^*|\theta)g(u^*), q(\theta|\theta^*)g(u)\}] > 0,
\end{aligned}$$

hence μ-irreducibility is satisfied.

Aperiodicity: This is considerably easier to show, all that is required is to see that the chain has a non-zero chance of moving to any set A, so it could move from any set in a partition of the sample space to any other, hence it cannot be periodic.

For an in-depth discussion of the theory underpinning the MH algorithm, please refer to [23].

3.4 Practical Application

With the algorithm defined and validity shown, it is prudent to now consider the application of the aforementioned algorithm to a partially observed system. As the target of interest is a posterior distribution given some data, the target now must take the form $\pi(\theta|x)$ for x some partially observed data and utilising the Markov property:

$$\pi(\theta|x) \propto \pi_0(\theta) \prod_{i=0}^{T-1} p(x_{t+1}|x_t, \theta).$$

Here, $\pi_0(\theta)$ represents our prior beliefs about the system and $p(x_{t+1}|x_t, \theta)$ is the observed data likelihood contribution, that for the purpose of the PMMH algorithm, is taken to be unknown. From this point there are two routes to take:

1. Apply the EM approximation to the intractable likelihood over each observed value. However this leads to high variance and poor mixing.
2. Implement a partition $t = t_0 < t_1 < t_2 < ... < t_m = t + 1$ and apply the EM via importance sampling to each point in this partition. This solves the issues of the previous method.

Importance sampling is chosen here as it has the unique property of producing unbiased estimators [24] of the quantities it is finding. When applying importance sampling, it is essential to choose a suitable proposal mechanism $q(x_{t_{i+1}}|x_{t_i}, x_{t_m})$ that can be easily simulated from. One such proposal distribution is the Durham and Gallant bridge construct [25].

3.4.1 Bridge Construct

The Durham and Gallant bridge construct [8] simulates paths between the observed data points. It can be shown to satisfy the following SDE [25]:

$$dX_{t_i} = \frac{X_{t_m} - X_{t_i}}{t_m - t_i}\Delta t + \sqrt{\beta(X_t, \theta)}dW_t, \tag{3.1}$$

where $\beta(x_{t_i}, \theta)$ represents the variance of the intractable process. It is possible to derive a transition density of the SDE, this is essentially done by simply applying the EM approximation to give:

$$q(x_{t_{i+1}}|x_{t_i}, x_{t_m}) \sim N(x_{t_{i+1}}; x_{t_i} + \frac{x_{t_m} - x_{t_i}}{t_m - t_i}\Delta t, \frac{t_m - t_{i+1}}{t_m - t_i}\beta(x_{t_i}, \theta)\Delta t).$$

This Bridge construct is utilised here as it provides a Gaussian approximation of the conditioned process and is sufficient for most models that will be considered. In Figure 3.1, an intuitive plot of the way bridges are generated is shown. The red points represent the observed data points and the coloured lines represent the bridge constructs. The use of bridges is very well illustrated here. Consider the scenario in which only the last and first observed data points is observed in Figure 3.1, applying the EM approximation directly over this would completely miss the true dynamics of the system due to the application effectively applying a shortest distance line between the two points. However, in applying the Durham and Gallant bridge construct, this is not the case and there are certain bridges that would potentially explore this region due to the randomness incorporated. It is worth mentioning that as shown in Equation 3.1, the linear mean of the bridge construct suggests it is possible that it would still fall short, implying that the use of another bridge would be more optimal for these scenarios.

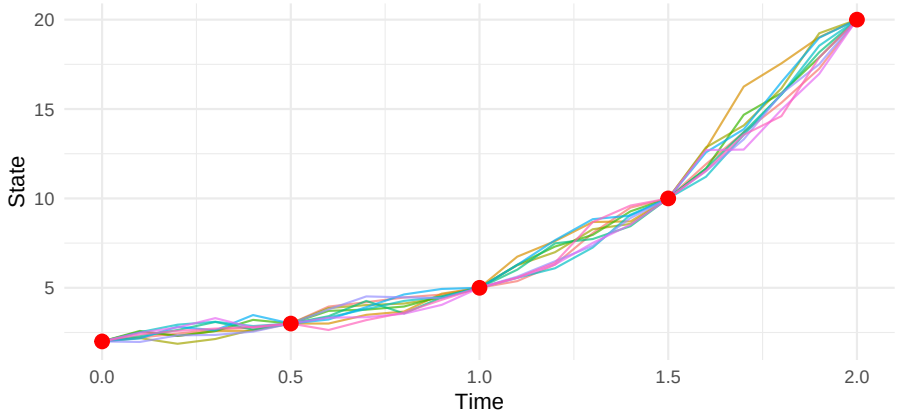

Figure 3.1: Durham and Gallant Bridge Construct utilising 10 simulations between observed data points (red) with $\mu = 3$, $\sigma = 0.1$

3.4.2 Importance Sampling

Equipped with the proposal sampler, the next step is to apply importance sampling to generate an unbiased estimator for the process. Taking θ as known and applying the distributions to the partitioned interval, the intractable likelihood can be found to be the normalising constant of:

$$\begin{aligned} p(x_{t_1}, x_{t_2}, ..., x_{t_{m-1}} | x_{t_0}, x_{t_m}) &= \frac{p(x_{t_1}, , ..., x_{t_{m-1}}, x_{t_m} | x_{t_0})}{p(x_{t_m} | x_{t_0})} \\ &\propto p(x_{t_1}, , ..., x_{t_m} | x_{t_0}) \\ &= \prod_{i=0}^{m-1} p(x_{t_{i+1}} | x_{t_i}) \\ &\approx \prod_{i=0}^{m-1} Pe(x_{t_{i+1}} | x_{t_i}), \end{aligned}$$

where $Pe(x_{t_{i+1}} | x_{t_i})$ is the EM approximation to the intractable likelihood applied over the interval. Furthering this, the joint transition density of the proposal mechanism over the partition to be:

$$q(x_{t_1}, ..., x_{t_{m-1}} | x_{t_0}, x_{t_m}) = \prod_{i=0}^{m-2} q(x_{t_{i+1}} | x_{t_i}, x_{t_m}).$$

Finally, applying importance sampling and generating paths from the proposal distribution, the weights can be calculated as follows:

$$w(x^k_{(t,t+1)}) = \frac{\prod_{i=0}^{m-1} \text{Pe}(x^k_{t_{i+1}} | x^k_{t_i})}{\prod_{i=0}^{m-2} q(x^k_{t_{i+1}} | x^k_{t_i}, x_{t_m})}$$

for $k = 1, ..., N$ representing the number of the path simulated for each time step. Combining all of this, it is possible to show that the transition density can be unbiasedly estimated by the mean of the importance sampled weights applied to the partition:

$$\begin{aligned}
p(x_{t+1}|x_t) =& p(x_{t_m}|x_{t_0}) \\
=& \int p(x_{t_1}, ..., x_{t_m}|x_{t_0}) dx_{t_1}, ..., dx_{t_{m-1}} \\
=& \int \prod_{i=0}^{m-1} \mathrm{Pe}(x_{t_{i+1}}|x_{t_i}) dx_{t_1}, ..., dx_{t_{m-1}} \\
=& \int \frac{\prod_{i=0}^{m-1} \mathrm{Pe}(x_{t_{i+1}}|x_{t_i})}{\prod_{i=0}^{m-2} q(x_{t_{i+1}}|x_{t_i}, x_{t_m})} \prod_{i=0}^{m-2} q(x_{t_{i+1}}|x_{t_i}, x_{t_m}) dx_{t_1} ... dx_{t_{m-1}} \\
=& \mathbb{E}_{X_{(t,t+1)} \sim q(\cdot)}[\hat{\mathrm{w}}(x_{(t,t+1)})] \\
\approx& \frac{1}{N} \sum_{k=1}^{N} \hat{\mathrm{w}}(x^k_{(t,t+1)}),
\end{aligned}$$

where, given u, a collection of Gaussian innovations driving the bridges, it can be seen that:

$$\frac{1}{N} \sum_{k=1}^{N} \hat{\mathrm{w}}(x^k_{(t,t+1)}) \approx \mathbb{E}_{u \sim g(\cdot)}[\hat{p}_u(x_{t+1}|x_t, \theta)] = p(x_{t+1}|x_t, \theta).$$

Hence, we have shown a method for the approximation of an unbiased estimator over partitioned intervals for an intractable likelihood function and can now proceed with the PMMH algorithm.

3.4.3 PMMH Implementation

The implementation of partitions between the observed data points leads to a target of the form

$$\pi_u(\theta|x) \propto \pi(\theta) \hat{p}_u(x_1, ...x_n|x_0, \theta) g(u).$$

Marginalising the target as before gives

$$\begin{aligned}
\int \pi_u(\theta|x) du \propto \pi(\theta) \int \hat{p}_u(x_1, ...x_n|x_0, \theta) g(u) du \\
= \pi(\theta) \mathbb{E}_{u \sim g(\cdot)}[\hat{p}_u(x_1, ..., x_n|x_0, \theta)] \\
= \pi(\theta) p(x_1, ..., x_n|x_0, \theta) \propto \pi(\theta|x).
\end{aligned}$$

Recalling the PMMH algorithm, there are several items to note:

- In step 1, the drawing of the auxiliary variable happens automatically upon the calculation of the unbiased estimator so the only necessary steps here are to initialise the chain and compute the target for the initial values.
- In step 2, the proposal distribution used will typically be a random walk proposal and will take the form $\theta^* \sim N(\theta, \lambda \cdot I)$ for some tuning parameter λ to be discussed presently.
- In step 2, the generation of the u^* is not implicitly carried out during the algorithm, they are instead a notational convention to represent the uncertainty implemented and carried out by the calculation of the unbiased estimator $\hat{p}(x_{t+1}|x_t, \theta)$ for the observed data-likelihood $\forall t \in [0, N]$.

- In step 3, the acceptance probability becomes a ratio of the target densities:

$$\alpha(\theta^*, u^*|\theta, u) = \frac{\pi(\theta^*, u^*|x)}{\pi(\theta^{(j-1)}, u^{(j-1)}|x)} = \frac{\pi_0(\theta^*)\prod_{t=0}^{N-1}\hat{p}(x_{t+1}|x_t, \theta^*)}{\pi_0(\theta^{(j-1)})\prod_{t=0}^{N-1}\hat{p}(x_{t+1}|x_t, \theta^{(j-1)})}.$$

- Due to numerical instability, these acceptance probabilities are typically calculated on the log scale.

3.5 Brownian Motion

Having fully defined the PMMH algorithm, we can apply the example discussed earlier, Geometric Brownian Motion, to help explore how the algorithm works in its applications and any practical considerations that need to be made:

- First define $(\mu, \sigma) = (\theta_1, \theta_2)$.
- The target density is taken to be the GBM model:

$$\pi(\theta|x) \propto \pi_0(\theta)\prod_{t=0}^{N-1}\hat{p}(x_{t+1}|x_t, \theta)$$

$$\pi_0(\theta) \sim N\left(\begin{pmatrix}0.5\\0.3\end{pmatrix}, \begin{pmatrix}0.5 & 0\\0 & 0.5\end{pmatrix}\right).$$

- Take the target contributions for the importance sampling to be:

$$p(x_{t_{i+1}}|x_{t_i}) \sim N(x_{t_i}; x_{t_i} + \theta_1 x_{t_i}\Delta t, \theta_2^2 x_{t_i}^2 \Delta t).$$

- The proposal density is taken to be the Durham and Gallant Bridge Construct:

$$q(x_{t_{i+1}}|x_{t_i}, x_{t_m}) \sim N(x_{t_{i+1}}; x_{t_i} + \frac{x_{t_m} - x_{t_i}}{t_m - t_i}\Delta t, \frac{t_m - t_{i+1}}{t_m - t_i}\theta_2^2 x_{t_i}^2 \Delta t).$$

- The initial conditions and parameters are chosen to be:
  - Observed data is synthetically generated from a GBM model with parameters $\theta = (0.5, 0.3)$, take 100 values with an inter-observation time equal to 0.1, implying a time period of length 10 is observed.
  - Initial random walk proposal mechanism variance = $0.01 \cdot I_3$, further tuning run done with variance of the previous run multiplied by $2.38^2/2$.
  - Number of bridges = 30.
  - Inter-observation time = 0.01.
  - Time simulated over per observed value = 0.1.
  - Number of iterations = 10000.

In Figures 3.2 and 3.3, there are various visualisations of the outputs described by the Brownian motion above. The trace plots illustrate the movement of the accepted proposed values, from this movement it is possible to calculate the empirical acceptance probability of the initial and tuning runs to be 0.208 and 0.348, respectively. The density plots show the effects of an increased number of iterations and higher acceptance probability on the smoothing of the density estimate. The Autocorrelation Function (ACF) plots detail very clearly why the tuning run again is far more desirable, high levels of autocorrelation between points suggest that the amount of information gained by each accepted move is small and must be addressed. A more in-depth look at the various practical considerations and metrics with which to assess the PMMH algorithm will follow next.

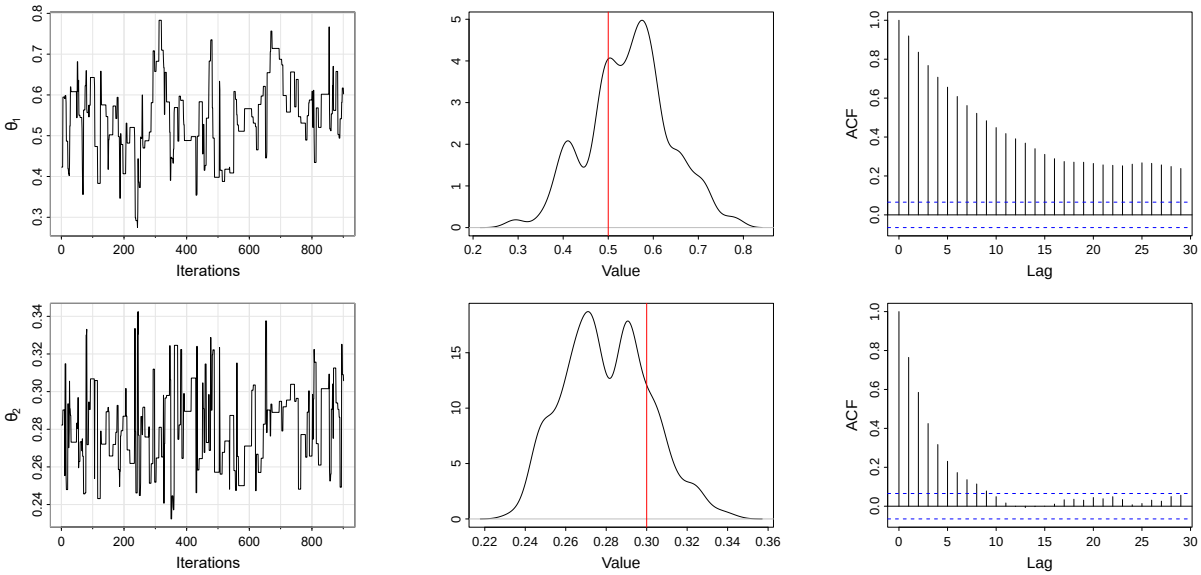

Figure 3.2: Trace, density and autocorrelation plots of the initial run.

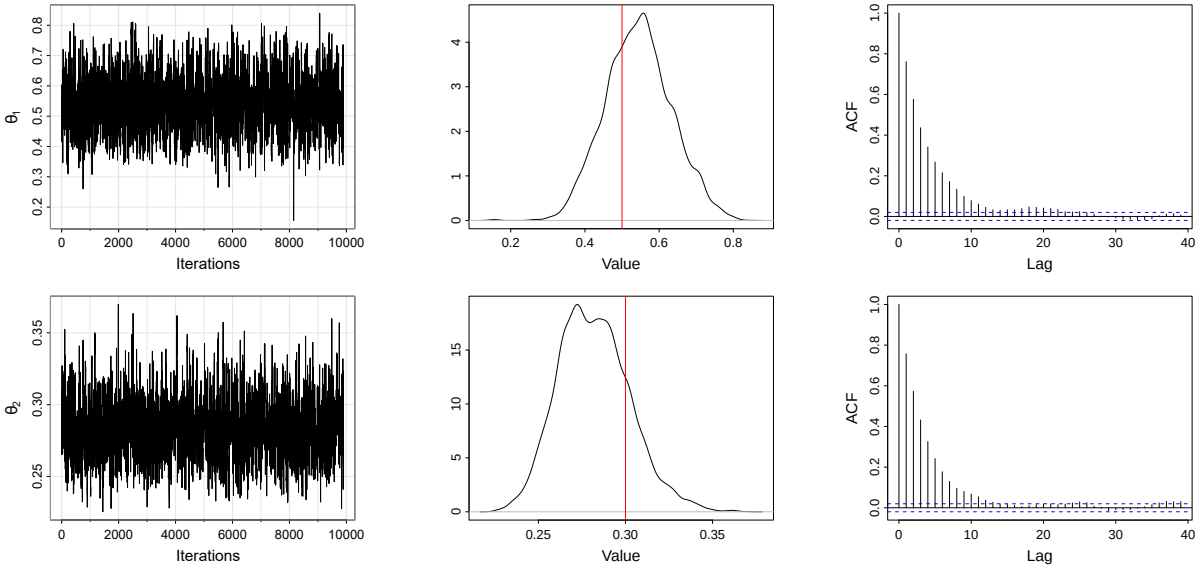

Figure 3.3: Trace, density and autocorrelation plots of the tuning run.

3.6 Practical Considerations

There are numerous aspects of this algorithm that must be considered during its application and when inspecting the output to ensure that the algorithm is performing correctly. This will be done by carrying out the analysis using the previously mentioned Geometric Brownian Motion. There is ample literature on the tuning of the various MH algorithms to ensure the efficiency of the algorithm. We will explore some of these aspects briefly here.

3.6.1 Burn-In

When carrying out PMMH, the desired result is to obtain draws from the target distribution $\pi(\theta)$. However, this is not always guaranteed as the initialisation of the chain may be far from the ground truth. Fortunately, the chain will converge, it is just a matter of discarding the first K iterations when analysing the output to ensure the draws come from the converged draws. This is called 'burn-in' and can be clearly seen in Figure 3.4, where there are trace plots and histograms showcasing the effects of retaining the 'burn-in' on the output used in Figure 3.3.

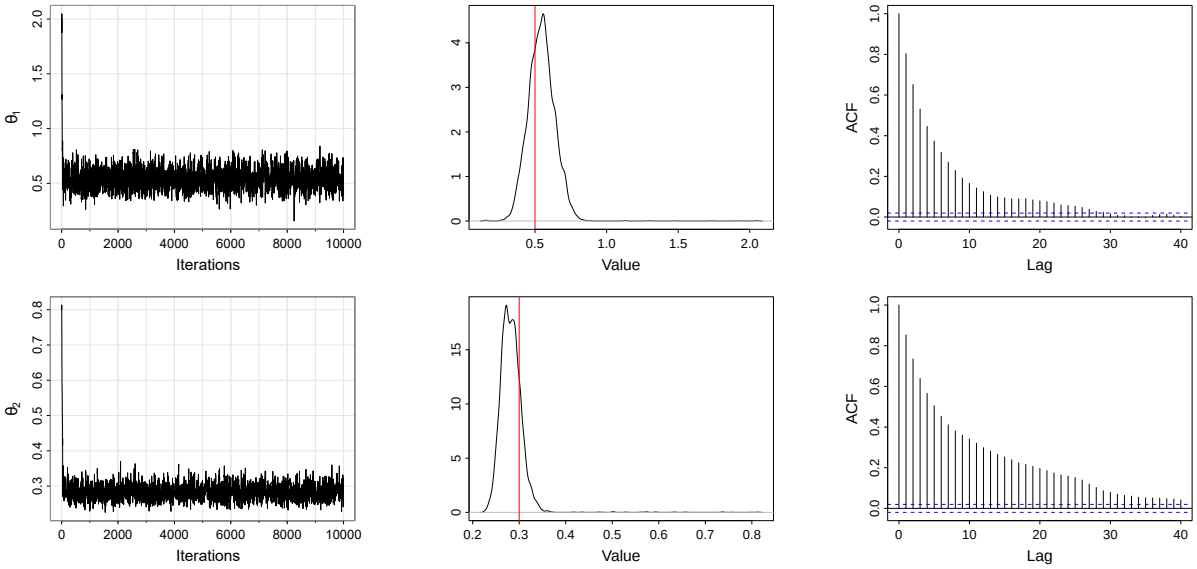

Figure 3.4: Trace, density and autocorrelation plots of the tuning run with burn-in.

The effects become immediately apparent. Burn-in increases the autocorrelation, reduces the acceptance rate, and is essential to remove in order to give the best possible tuning matrix.

3.6.2 Acceptance Rate

The acceptance rate is one of the most important aspects to consider when tuning a PMMH algorithm. The acceptance rate is predominantly affected by the variance matrix of the proposal matrix. In practice, one typically proceeds with an initialisation run, choosing a diagonal matrix with variance that seems sensible relative to the initialisation of the parameters. In Table 3.1, the optimal acceptance probabilities for each dimension are stated as proposed by [26].

Dimension d	1	2	3	4	5	6	7
p_{opt}	0.441	0.352	0.316	0.279	0.275	0.266	0.255

Table 3.1: Optimal acceptance probabilities.

It is possible to use the tuning matrix proposed in [27] to aid in achieving these results, this tuning matrix can be found by scaling the innovation variance λ of the initial run by a constant as follows:

$$\lambda_p = \left[\frac{(2.38)^2}{d}\right]\lambda.$$

With a further limiting value of 0.234 as $d \to \infty$ as shown in [27]. These results hold fairly well in the context of all MCMC algorithms. However, it is possible to specify some more specific

constraints for the PMMH algorithm using values proposed in [28] to find the new scaling value:

$$\lambda_p = \left[\frac{(2.562)^2}{d}\right]\lambda. \tag{3.2}$$

Unlike the MH algorithm, whose optimal values are referenced in Table 3.1, the PMMH algorithm takes far longer to run due to the nature of its construction. It is therefore prudent to consider a different set of optimal acceptance probabilities that take into account the computation time of the PMMH's observed data likelihood. Fortunately, such a range exists, as demonstrated in [27]. This states that the new optimal acceptance probability rate ranges between 0.07 and 0.234. Where 0.07 is applicable to high computational cost and 0.234 is for a cheaper run. These are the asymptotic probabilities that are made under the assumption that dimension $d \to \infty$ so it is worth bearing this in mind when considering the acceptance probabilities for low-dimensional targets. The generalised efficiency criterion for tuning is given by:

$$\text{Efficiency} = \frac{\text{ESJD}(\lambda)}{\text{Expected one-step computing time}},$$

where the scaling value in Equation 3.2 is an asymptotic solution to the maximisation of the Efficiency criterion. In practice, one should aim to achieve an acceptance probability in the range $0.07 - 0.234$ to maximise efficiency relative to the computational burden. Equation 3.2 is a useful scaling parameter that can be applied towards achieving this range.

In reference to the simulations run earlier, the acceptance probabilities obtained for the initial and tuning runs are equal to 0.208 and 0.348. This is not in the optimal range dictated by the literature on PMMH optimal tuning as per [28] but aligns very closely with the standard RWM optimal values. This can be attributed to the low dimensionality of the target and a well-tuned initial proposal. However, as the model dimensionality increases, it becomes increasingly important to rely on the PMMH optimal parameter range. For further reading on the topic of optimal acceptance probabilities, refer to [29], [30].

3.6.3 Effective Sample Size

A metric to measure the PMMH algorithms efficiency is the Effective Sample Size (ESS) [31]. This is the amount by which autocorrelation within chains produced by the PMMH algorithms increases the uncertainty in estimates and it can be calculated as follows:

$$N_{eff} = \frac{N}{1 + \sum_{t=1}^{\infty} \rho_t}, \tag{3.3}$$

where ρ_t represents the autocorrelation of the process at lag t. Intuitively, ESS calculates the number of independent samples N_{eff} that would lead to an estimator with the same variance as with the N autocorrelated samples. It is now possible to utilise this metric to investigate the optimality of the parameter regimes. A large value for ESS is desirable when running these simulations, at least 100 to ensure reliability from the output.

3.6.4 Expected Squared Jump Distance

The ESS is sufficient to evaluate the efficiency of the sampler as discussed above. However, another metric can be introduced to provide some context to the optimal acceptance probabilities for a PMMH algorithm: the Expected Squared Jump Distance (ESJD) [32], [28]. For the purposes of

this report it will be used alongside the ESS to assess the overall efficiency of the parameter chains. It can be defined as follows:

$$ESJD(\lambda) = \mathbb{E}_{J_\lambda}[(\theta_{t+1} - \theta_t)^2] \tag{3.4}$$

The intuition behind this criterion is that its maximisation is equivalent to the minimisation of the lag-1 autocorrelation of the chain. This directly relates to the concept of the amount of information obtained at each accepted iteration. The lower the autocorrelation, the more effectively the chain explores the target distribution. As such, the ESJD serves as a useful metric to consider when applying PMMH. Similarly to the ESS, the larger the value, the better the simulation in comparison to other runs. There is no recommended value for ESJD, it should be used predominantly as a comparative metric.

3.6.5 Number of Bridges

We now consider aspects related to the specific examples and methodologies utilised in this report. Due to the use of the bridge construct, it is worth investigating the optimal number of bridges to use in the proposal mechanism. The aim is to strike a balance between ESJD, ESS, optimal acceptance probability and crucially the computation time for the simulations.

In Table 3.2, there are a selection of runs utilising the same initial parameters as stated in the Brownian Motion subsection, except that the number of bridges are varied. Using this output it becomes clear that when looking at the metrics applied to the parameters $\theta = (\theta_1, \theta_2)$, the number of bridges that is most effective appears to be around 30. However, it has a large increase in computation time with little increase to its ESS and ESJD, suggesting that, with limited computational power, five bridges would be more optimal. It is worth mentioning that, as the maximal value of the ESS and ESJD indicated the best results, the minimum of the ESS and ESJD has been taken across both parameter chains to give a more complete overview of the best chain.

3.6.6 Partition Size

As mentioned in the previous section, it is now necessary to investigate the effect of partition size on the same parameters previously stated. In Table 3.2, the partition size is varied and reveals that the optimal size is about 2. This is not what one would expect and suggests the use of a different bridge construct that better explores the dynamics of the GBM model. However, it is still possible, due to the large ESS, ESJD and optimal acceptance probabilities, that this bridge construct is a good fit and functions better under a smaller partition. Further investigation of bridge constructs is required to investigate this, but this is not necessary for the purposes of this report.

Metric	**1**	**5**	**30**	**90**	**2**	**10**	**50**	**100**
Computation Time (sec)	71	244	1190	3390	366	1190	4170	9850
Minimum ESS	978	1260	1350	1330	1270	1230	1280	1220
Minimum ESJD ($\times 10^{-4}$)	1.67	1.99	2.03	1.92	2.19	2.01	1.91	1.82
Acceptance Probability	0.302	0.333	0.348	0.369	0.326	0.348	0.363	0.312
	Number of Bridges				**Partition Size**			

Table 3.2: Comparison of performance metrics for θ

This concludes the discussion on practical considerations and ultimately the PMMH algorithm itself. For further information on these topics please refer to [33].

Chapter 4

Stochastic Kinetic Models

Having defined the PMMH algorithm in full and had a brief look at its applications to the GBM model to showcase how the PMMH algorithm works, we now move onto the main model of study; the Stochastic Kinetic Models (SKM) [34]. In this chapter we will outline the derivation of SKM models and briefly discuss the specific model which we will be investigating — the Lotka-Volterra model.

4.1 Reaction Network

To commence the derivation of an SKM, a method for visualising the set of all possible reactions occurring in a system must be constructed first. This can be done via a reaction network. Consider a quantity of each species $X_1, ..., X_U$ and reaction rates $\theta_1, ..., \theta_V$. The reaction network can be defined as follows:

$$\begin{aligned} p_{1,1}X_1 + p_{1,2}X_2 + \cdots + p_{1,U}X_U &\xrightarrow{\theta_1} q_{1,1}X_1 + q_{1,2}X_2 + \cdots + q_{1,U}X_U \\ &\vdots \\ p_{V,1}X_1 + p_{V,2}X_2 + \cdots + p_{V,U}X_U &\xrightarrow{\theta_V} q_{V,1}X_1 + q_{V,2}X_2 + \cdots + q_{V,U}X_U \end{aligned}$$

Where P and Q are $V \times U$ matrices representing the amount of each species required or produced by each species at each reaction. For ease later on, it is also worth defining the stoichiometric matrix, which is a $U \times V$ matrix S such that $S = (Q - P)'$.

4.2 Markov Jump Process

With a method for visualising the SKM and its various components, the next step is to attempt to put this reaction network in a more mathematically rigorous format.

To carry this out, it is essential to consider how these reactions occur. They occur when a collection of species collide under the influence of some stochastic process. The probability of these reactions occurring depends on the current state of the system, this is due to the Markovian property. In addition to this, when a reaction occurs, the system evolves in discrete jumps. Combining these two factors gives the resulting Markov Jump Process (MJP).

Definition 4.1 Markov Jump Process: A stochastic process that evolves in continuous time in a discrete state space with transitions occurring at random times governed by the Markov property.

Using the Markov Jump Process definition, the probability of a reaction or a transition occurring can be defined. Consider a reaction i taking place in the time interval $[t, t + dt]$, the probability of this reaction is given by:

$$h_i(X_t, \theta_i)dt + \mathcal{O}(dt),$$

where the instantaneous rate/hazard function $h_i(\cdot)$ takes the form:

$$h_i(X_t, \theta_i) = \theta_i \prod_{j=1}^{U} \binom{X_{j,t}}{p_{i,j}}.$$

This hazard function results from the assumption of mass action kinetics which states that the rate of a chemical reaction is proportional to the product of the concentrations of the reacting species.

From this point the subsequent analysis can proceed through two paths: numerical and exact simulation. Proceeding with exact simulation is possible via the Gillespie algorithm [35] but is very inefficient for large systems. The numerical solution circumvents this by sacrificing some accuracy for speed and efficiency; this can be constructed via the diffusion approximation. Both of these methods will be discussed in detail now.

4.3 Gillespie Algorithm

The algorithm through exact simulation for a general stochastic reaction with V reactions and hazard functions corresponding to the reactions can be stated as follows [35]:

Gillespie Algorithm

1. Set t = 0 and initialise the rate constants and species quantities.
2. Generate random numbers $r_1, r_2 \sim U([0, 1])$.
3. Compute the propensity function of the system:

$$h(x, \theta) = \sum_{i=1}^{V} h_i(x, \theta_i).$$

4. Calculate the time to the next event:

$$\tau = \frac{1}{h(x, \theta)} log\left(\frac{1}{r_1}\right),$$

and set $t = t + \tau$.
5. Figure out which of the V reactions took place by finding the integer j such that:

$$r_2 \geq \frac{1}{h(x, \theta)} \sum_{i=1}^{j-1} h_i(x, \theta_i), \qquad r_2 \leq \frac{1}{h(x, \theta)} \sum_{i=1}^{j} h_i(x, \theta_i).$$

Carry out reaction j, update the species states and go back to step 2 unless $t = T_{max}$.

Despite its exactness, clearly it is very computationally intensive. For large numbers of species, there will be an prohibitively large quantity of reactions occurring and the simulation will progress very slowly. It is worth mentioning a couple of points:

- Step 3 is assuming that the next reaction time is distributed by an exponential distribution with parameter $h(x, \theta)$. This can be seen as the equation takes the form of the CDF of the exponential distribution.
- Step 4 ensures the probability that event i occurs given any reaction occurs is consistent across multiple iterations.

These two steps together are what make the Gillespie algorithm follow the behaviour of a Markov Jump Process. It works by sampling the time until the next reaction using an exponential distribution, and then picking which reaction happens based on how likely it is compared to the others. Despite it being very computationally heavy, it provides an exact simulation of how the system evolves over time, capturing the randomness in stochastic systems.

4.4 Diffusion Approximation

One possible numerical solution is the diffusion approximation, which makes two key assumptions that are worth mentioning. There must be a large number of species present, and the discrete state jumps stated in the MJP are to be approximated as continuous perturbations. Using these assumptions, the derivation can now be carried out.

First, the probability of a single reaction i occurring is governed by an exponential distribution $X_i \sim \exp(h(x, \theta_i))$ with rate function equal to the hazard function i from the Gillespie algorithm. This can then be generalised to find the number of events, A, occurring in an interval of length dt. This is distributed by $X_i(t) \sim Po(h_i(X_t, \theta_i) \cdot dt)$. This generalisation is calculated using the Kolmogorov forward equations which will not be shown here, rather the result will be taken as fact since it will provide little benefit to carry out this derivation. If the reader is interested please refer to [36].

Using the assumptions stated earlier, it is possible to now apply the central limit theorem to the Poisson distribution, $X_i(t)$, and find that the number of reactions occurring in a time interval dt is given by:

$$dR_t = h(X_t, \theta) \cdot dt + \text{diag}\{\sqrt{h(X_t, \theta)}\} \cdot dW_t.$$

Using the stoichiometric matrix defined in Section 4.1 and the fact that $dX_t = S \cdot dS_t$, the state of the species after a time interval dt can be found. The resulting equation is given by:

$$dX_t = S \cdot h(X_t, \theta) \cdot dt + \sqrt{S\text{diag}\{h(X_t, \theta)\}S'}dW_t. \tag{4.1}$$

Equation 4.1 is the general form of the SDE for the diffusion approximation. It can be used with all of the preceding work in this report to undergo parameter inference on SKMs. It is also worth noting that to return to the deterministic state one just needs to ignore the components relating to the Wiener process.

4.5 Lotka-Volterra Model

One such SKM of interest is the Lotka-Volterra model [7], this model evaluates the effects of predator and prey through their natural birth and death rates with an interaction term to encapsulate the predator's consumption of the prey. The following reaction network with X_1 as the prey and X_2 as the predator surmises this:

$$X_1 \xrightarrow{\theta_1} 2X_1,\ X_2 \xrightarrow{\theta_2} \emptyset,$$
$$X_1 + X_2 \xrightarrow{\theta_3} 2X_2.$$

Using the previously-derived diffusion approximation for SKMs, the approximation to the reaction network via an SDE can be seen as follows:

$$\begin{bmatrix} dX_1 \\ dX_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \theta_1 X_1 - \theta_3 X_1 X_2 \\ \theta_3 X_1 X_2 - \theta_2 X_2 \end{bmatrix}}_{\alpha(x)} \text{dt} + \underbrace{\begin{bmatrix} \theta_1 X_1 + \theta_3 X_1 X_2 & \theta_3 X_1 X_2 \\ \theta_3 X_1 X_2 & \theta_3 X_1 X_2 + \theta_2 X_2 \end{bmatrix}^{\frac{1}{2}}}_{\beta(x)} dW_t$$

With the SDE defined, an investigation can be conducted into the dynamics of the Lotka-Volterra model, and the relationship between the ODE and the SDE. In Figure 4.1 the dynamics for the specific parameter regime $\theta = (0.5, 0.0025, 0.3)$ shows periodic behaviour as demonstrated by the black lines representing the ODE. The simulations produced by the SDE in blue and red showcase the advantages of using SDEs, in this case it allows for far more exotic dynamics between the systems such as extinctions and population booms, this is only possible due to the stochasticity present.

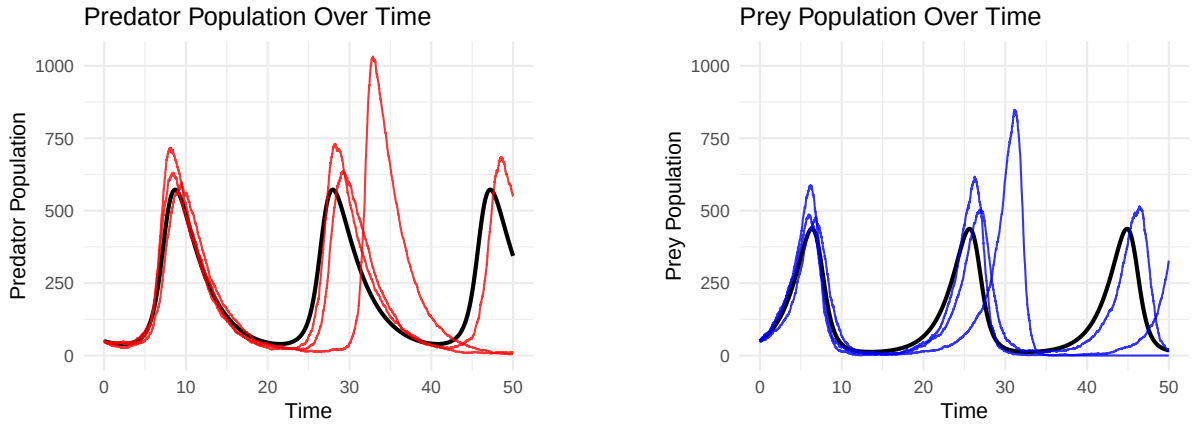

Figure 4.1: Comparison of Predator and Prey Populations using ODE (black) vs SDE (blue and red)

In addition to the dynamics of the system it is beneficial to compare the analytical and numerical methods discussed. This is illustrated in Figure 4.2, using 500 iterations over a time period of 12 with parameter regime $\theta = (0.5, 0.0025, 0.3)$ and initial starting point $X = (50, 50)$. The red-dashed lines correspond to confidence intervals around the mean of the simulated draws. The means for both methods are roughly the same. However, the confidence intervals for the Gillespie algorithm increase substantially with time. This suggests that the diffusion approximation does not fully capture the randomness of the system. However, this property likely drives the reduction in the computation time.

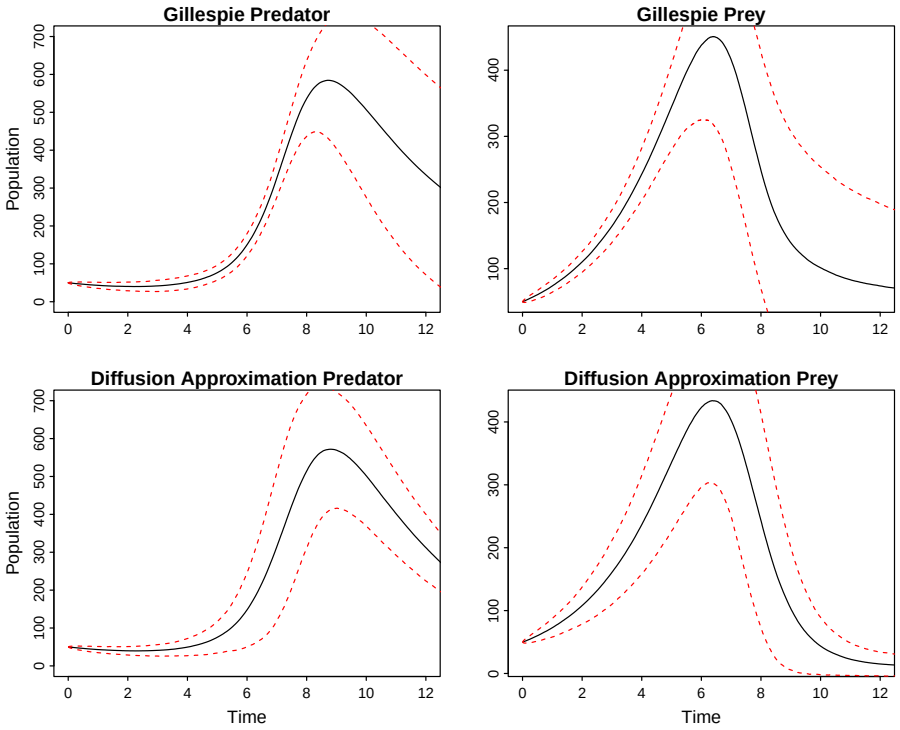

Figure 4.2: Lotka-Volterra simulation through the Diffusion Approximation and Gillespie algorithms with the mean (black line) and a 95% confidence interval (dashed line).

In reference to the computational efficiency of the simulations, the diffusion approximation simulation ran for 0.2 seconds. Comparatively, the Gillespie algorithm took ten seconds to achieve the same point in time. This is considerably longer and further solidifies the decision to use numerical methods for further analysis of the model. It is worth mentioning that initially, a run was carried out over a time period of 50. However, the Gillespie algorithm did not finish running after two days of simulation, despite the diffusion approximation finishing in five seconds. This considerable computational efficiency and accuracy in capturing the mean of the process motivates the diffusion approximation as the best process to use to model the system. This concludes our discussion of SKMs and the Lotka-Volterra model.

Chapter 5

Inference for SKMs using Bridge Constructs

Up to this point, we have discussed the pseudo-marginal Metropolis-Hastings algorithm, its application to a simple GBM model and used this to showcase various properties of the PMMH algorithm. We now move onto the main area of study, the application of the PMMH algorithm to the Lotka-Volterra SKM.

5.1 Aims and Initial Setup

The aim of this chapter is to demonstrate the effectiveness of the PMMH algorithm compared to other MCMC schemes, in particular the Gibbs sampler. This will be done by generating a synthetic dataset using the parameters $\theta = (0.5, 0.0025, 0.3)$ as in [7] and initial starting values for both species equal to $(50, 50)$ spanning a time period of length 50. Further, we will only use a partially-observed subset of this data, taking points from the data at consistent intervals. The aim, using this data, is now to investigate the effectiveness of the various algorithms and their constituent parts on the output.

5.2 Pseudo-Marginal Metropolis-Hastings

Applying the PMMH algorithm to the Lotka-Volterra model requires the following:

- First define some key quantities:

$$\beta(X_t, \theta) = \begin{bmatrix} \theta_1 X_{t,1} + \theta_3 X_{t,1} X_{t,2} & \theta_3 X_{t,1} X_{t,2} \\ \theta_3 X_{t,1} X_{t,2} & \theta_3 X_{t,1} X_{t,2} + \theta_2 X_{t,2} \end{bmatrix}^{\frac{1}{2}}$$

$$\alpha(X_t, \theta) = \begin{bmatrix} \theta_1 X_{t,1} - \theta_3 X_{t,1} X_{t,2} \\ \theta_3 X_{t,1} X_{t,2} - \theta_2 X_{t,2} \end{bmatrix}.$$

- The prior density is taken to be:

$$\pi_0(\theta) \sim N\left(\begin{pmatrix} 0.5 \\ 0.0025 \\ 0.3 \end{pmatrix}, \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix} \right).$$

- Take the target for the importance sampling to be:

$$p(x_{t_{i+1}}|x_{t_i}) \sim N(x_{t_i}; x_{t_i} + \alpha(x_{t_i}, \theta)\Delta t, \beta(x_{t_i}, \theta)\Delta t).$$

- The proposal density for the the importance sampling is taken to be the Durham and Gallant Bridge Construct:

$$q(x_{t_{i+1}}|x_{t_i}, x_{t_m}) \sim N(x_{t_{i+1}}; x_{t_i} + \frac{x_{t_m} - x_{t_i}}{t_m - t_i}\Delta t, \frac{t_m - t_{i+1}}{t_m - t_i}\beta(x_{t_i}, \theta)\Delta t).$$

- The initial conditions and parameters are chosen to be:
  - Random walk proposal mechanism takes initial run with variance mechanism = $0.001 \cdot I_3$, further tuning run done with variance of the previous runs multiplied by $(2.562^2)/3$.
  - Number of bridges = 30.
  - Inter-observation time = 0.05.
  - Time simulated over per observed value = 1.
  - Number of simulations = 10000.

In Figure 5.1, trace plots have been produced, showcasing the mixing of the sampler, which has an acceptance probability of 0.1549. The computation time came out to be approximately 61 600 seconds for the initial run and 68 800 seconds for the tuning run. This will be further investigated below.

As mentioned previously, this PMMH schema will be compared to a Gibbs sampler, which will be described in Section 5.3. However, it is worth mentioning that a variety of runs will also be carried out to examine the effect of certain parameter changes on PMMH efficiency and potentially find areas where it far exceeds the efficiency of the Gibbs sampler. This will be discussed in Section 5.4.

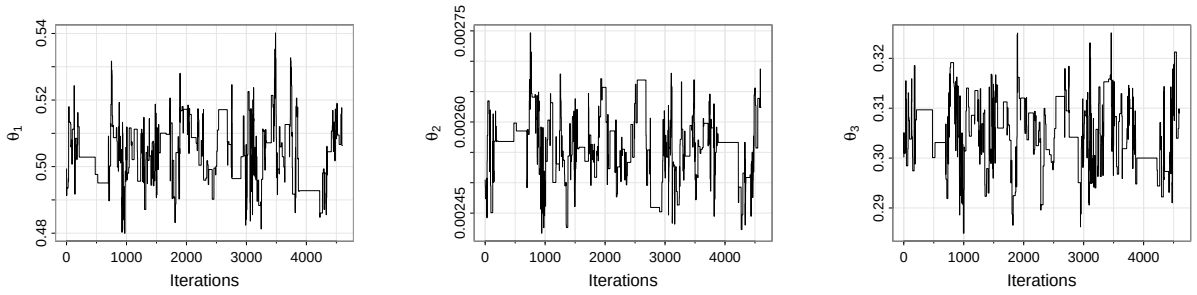

Figure 5.1: Trace plots of the tuning run after burn-in.

5.3 Gibbs Sampler

This algorithm is effectively a block-wise transition version of the Metropolis-Hastings algorithm, it is very efficient for higher dimensions and only requires the conditional probabilities to compute. The algorithm [37] can be seen as follows for a generic target $\pi(\theta_1, ..., \theta_d)$.

Gibbs Sampling Algorithm

1. Initialise the chain to $\theta^{(0)} = (\theta_1^{(0)}, ..., \theta_d^{(0)})$ somewhere in the support of $\pi(\theta)$. Set iteration counter to 0.
2. Obtain a new value $\theta^{(j)}$ from $\theta^{(j-1)}$ by iterative generation of values:
   - $\theta_1^{(j)} \sim \pi(\theta_1|\theta_2^{(j-1)}, ..., \theta_d^{(j-1)})$ using a Metropolis-Hastings step with proposal distribution $q_1(\theta_1^*|\theta_1^{(j-1)})$.
   - $\theta_2^{(j)} \sim \pi(\theta_2|\theta_1^{(j)}, \theta_3^{(j-1)}, ..., \theta_d^{(j-1)})$ using a Metropolis-Hastings step with proposal distribution $q_2(\theta_2^*|\theta_2^{(j-1)})$.
   - $\theta_d^{(j)} \sim \pi(\theta_d|\theta_1^{(j)}, ..., \theta_{d-1}^{(j)})$ using a Metropolis-Hastings step with proposal distribution $q_d(\theta_d^*|\theta_d^{(j-1)})$.
3. If $j = N$, stop, otherwise put j to $j + 1$, and return to step 2.

This is the general form of the Gibbs sampler given in [23] to which we refer the reader to for further details.

5.3.1 Practical Application

The Gibbs sampling algorithm has now been defined and briefly discussed. Its application to the Lotka-Volterra model with the parameter regime is the next component of assessing the efficiency of the PMMH. Firstly, the data provided is partially-observed so consider the data to be of the form:

$$\begin{aligned} x^0 &= (x_{t_0}, ..., x_{t_N}) \\ x^u &= (x_{\tau_{0,1}}, x_{\tau_{0,2}}, ..., x_{\tau_{0,m-1}}, x_{\tau_{1,1}}, ..., x_{\tau_{n-1,m-1}}), \end{aligned}$$

where x^0 represents the N observed data values and x^u represents the proposed values over the partition induced between observed data values. With this in mind the target of the Gibbs sampler can be seen to be:

$$\pi(x^u, \theta|x^0) \propto \pi_0(\theta) \prod_{i=0}^{n-1} \prod_{j=0}^{m-1} Pe(x_{\tau_{i,j+1}}|x_{\tau_{i,j}}, \theta).$$

When applying the Gibbs sampler, block-wise transitions are used. The nuance in this is the utilisation of one block to be the set of unobserved data values x^u and the other block to be the parameters of interest θ as opposed to the general way where typically only parameters are updated. This can be seen as follows;

$$\begin{aligned} &\theta|x^0, x^u && \text{[MH step using Random Walk Proposal]} \\ &x^u|\theta, x^0 && \text{[MH step using Bridge Proposal].} \end{aligned}$$

This is the general setup for applying the Gibbs sampler in this context, but in order to carry out the process, more detail is required. In addition, if the conditional distributions are not available for simulation, then one way to circumvent this is to simulate the conditionals through the use of the Metropolis-Hastings algorithm. This will all be carried out now, starting with the θ block.

- For iteration k, start with the block transition for θ:
  - Propose $\theta^* \sim N(\theta^{(k-1)}, \Sigma)$.
  - Accept proposed block with probability:

$$\alpha(\theta^*|\theta^{(i-1)}) = \min(1, R_\theta).$$

  - Define R_θ to be:

$$\begin{aligned} R_\theta =& \frac{\pi(x^{u(k)}, \theta^*|x^0)}{\pi(x^{u(k)}, \theta^{(k-1)}|x^0)} \\ \propto& \frac{\pi_0(\theta^*)\prod_{i=0}^{n-1}\prod_{j=0}^{m-1} Pe(x^{(k-1)}_{\tau_{i,j+1}}|x^{(k-1)}_{\tau_{i,j}}, \theta^*)}{\pi_0(\theta^{(k-1)})\prod_{i=0}^{n-1}\prod_{j=0}^{m-1} Pe(x^{(k-1)}_{\tau_{i,j+1}}|x^{(k-1)}_{\tau_{i,j}}, \theta^{(k-1)})}. \end{aligned}$$

- Having updated the θ block, proceed with the updating of the unobserved values.
- It is important to note that these are updated in N chunks, representing the unobserved data points between each observed value $(t_0, t_1), (t_1, t_2), ..., (t_{n-1}, t_n)$.
  - For each block of the form (t_i, t_{i+1}), propose using the Durham and Gallant Bridge Sampler:

$$\begin{aligned} x^{u^*}_{(i,i+1)} =& (x^{u^*}_{\tau_{i,1}}, x^{u^*}_{\tau_{i,2}}, ..., x^{u^*}_{\tau_{i,m-1}}) \\ \sim& q(x^{u^*}_{(i,i+1)}|x^0_{t_i}, x^0_{t_{i+1}}, \theta^{(k)}) \\ =& \prod_{j=0}^{m-2} q(x_{\tau_{i,j+1}}|x_{\tau_{i,j}}, x^0_{t_{i+1}}). \end{aligned}$$

  - Accept each individual block update with probability:

$$\alpha(x^{u^*}_{(i,i+1)}|x^{u^{(k-1)}}_{(i,i+1)}) = \min(1, R_i).$$

  - Where R_i is defined as:

$$R_i = \frac{\prod_{j=0}^{m-1} Pe(x^*_{\tau_{i,j+1}}|x^*_{\tau_{i,j}}, \theta^{(k)})}{\prod_{j=0}^{m-1} Pe(x^{(k-1)}_{\tau_{i,j+1}}|x^{(k-1)}_{\tau_{i,j}}, \theta^{(k)})} \times \frac{\prod_{j=0}^{m-2} q(x^{(k-1)}_{\tau_{i,j+1}}|x^{(k-1)}_{\tau_{i,j}}, x^u_{t_{i+1}}, \theta^{(k)})}{\prod_{j=0}^{m-2} q(x^*_{\tau_{i,j+1}}|x^*_{\tau_{i,j}}, x^u_{t_{i+1}}, \theta^{(k)})}.$$

  - This is a standard Metropolis-Hastings step with target density equal to the EM approximation to the LV model and the proposal density is the DG bridge construct.
  - Now, set $x^*_{\tau_{i,0}} = x^0_{t_i}$ and $x^*_{\tau_{i,m}} = x^0_{t_{i+1}}$ for each block and then repeat this process for $k = 1, ..., N$.
- This concludes both block updates. All that remains is to repeat this iterative process M times.

5.4 Application to Lotka-Volterra

Having detailed how to carry out the Gibbs sampler in the context of this problem, we can now produce results from the Gibbs sampler detailed above with the following parameters and conditions as such:

- Utilise partially observed data set described previously in section 5.1.
- Initialise parameters to be $\theta = (0.9, 0.05, 0.7)$ to showcase the convergence of the chain to the ground truth.
- Initialise random walk proposal mechanism for θ proposals to have variance parameter $= I_3 \cdot 0.001$, then do further tuning run using the variance of the previous run multiplied by the constant $2.38^2/3$.
- Size of partition = 0.05.
- Size of Inter-observation time = 1.

In Figure 5.2, the trace plots for the preceding sampler can be seen. It is apparent that the Gibbs sampler does not perform nearly as well by examining the trace plots. The computation time for this algorithm took 5 818 seconds, significantly faster than the PMMH algorithm. However, the mixing is far worse even if it does converge to the correct values. This will be addressed now.

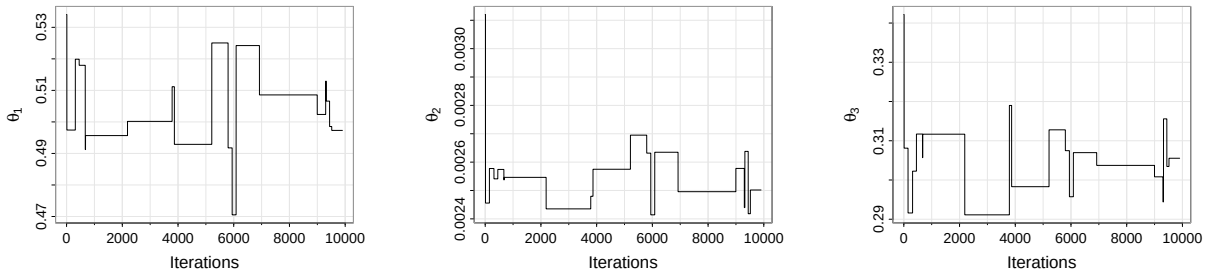

Figure 5.2: Trace plots of the tuning run after burn-in.

5.5 Comparison of Methodologies

The aim of this section is to compare the overall efficiency of PMMH and the Gibbs sampler. This will be investigated by conducting multiple runs using the previously-stated baselines for both algorithms and running them under the same conditions to achieve a fair comparison. The variations can be surmised as:

- Vary the partition size between values 5, 20 and 100.
- Vary the observed data set to have inter-observation time be equal to 5, 1, 0.1.

These changes are selected to explore the key differences in the algorithms.

5.5.1 Partition Variations

In Figure 5.3, the ACF plots for each of the simulations are shown. It becomes apparent that the autocorrelation in the simulations produced by the Gibbs sampler is much higher when compared to the PMMH schema, showcasing the PMMH strength over the Gibbs sampler. However, there is a trend between increasing partition size and increasing autocorrelation. This is as expected due to the method in which bridges are constructed. The further the bridge has to stretch, the more dependent the proposed values are on each other. Unlike the GBM model, the DG bridge construct clearly captures the dynamics of the LV model very well.

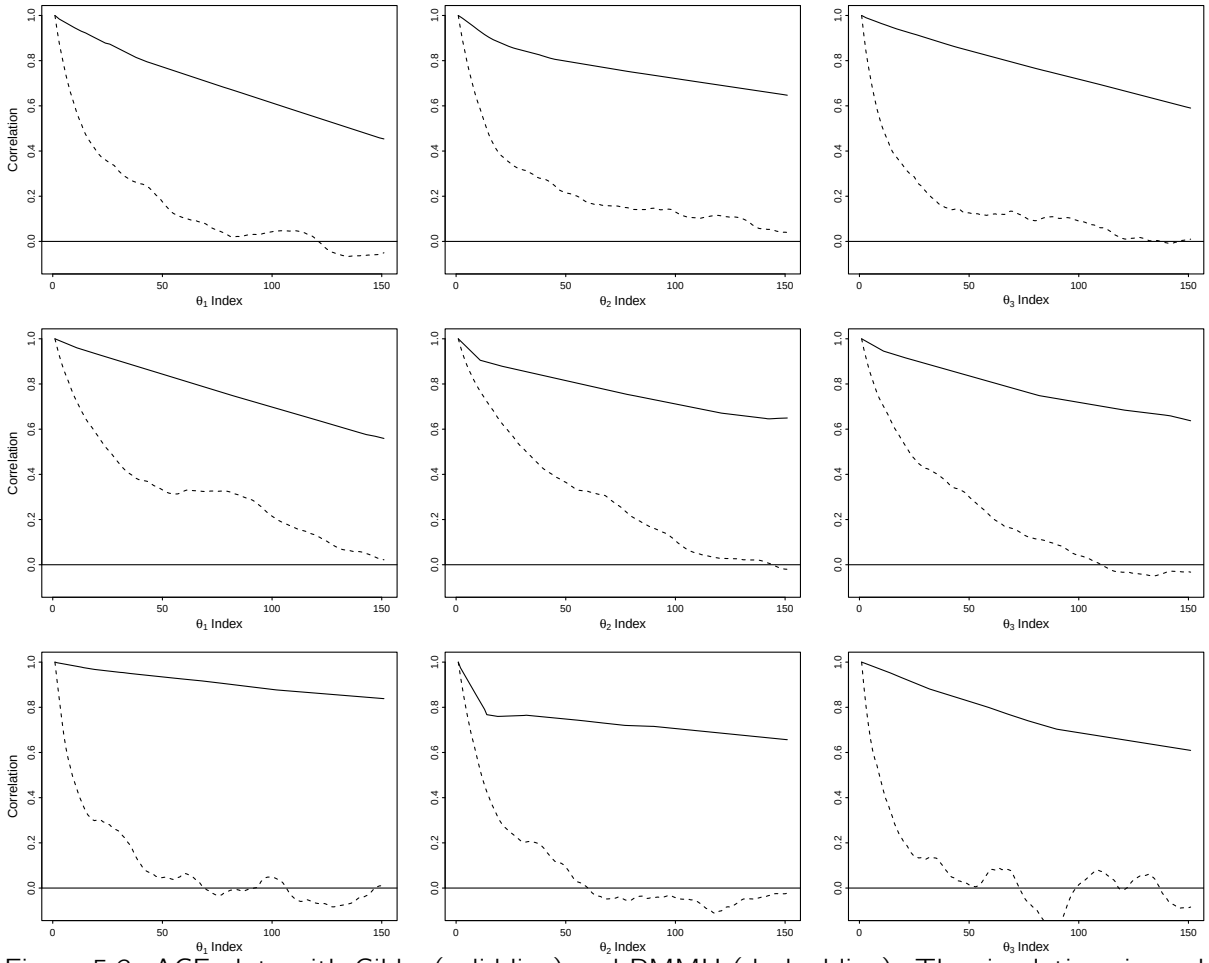

Figure 5.3: ACF plots with Gibbs (solid line) and PMMH (dashed line). The simulations in each row from the top correspond to partition size 5, 20 and 100 respectively.

Table 5.1 shows the various metrics discussed in Section 3, further illustrating the effectiveness and efficiency of the algorithm. Clearly, the Gibbs sampler is outperformed by the PMMH. However, within this table, there is a row indicating the number of burn-in iterations discarded. This is essential to take into consideration because, when looking back at chapters 3.6.3 and 3.6.4, the ESS and ESJD metrics are dependent on the number of iterations. This suggests that despite the low ESS and ESJD of the partition size equal to 100, it does outperform the other partition sizes relative to the number of iterations the metrics are calculated with. As the chain has achieved convergence after discarding the burn-in, the autocorrelation profile of the data can be assumed to be roughly constant. Therefore, it is possible to approximate the growth in ESS with iterations to

be directly proportional to the number of iterations it is increased by. For further computation, it would be worth doubling the iterations used to give a more standardised view of the results, but the computation cost is a huge factor when calculating this and thus is not always feasible. A rule of thumb for a valid ESS is at least 100, any less and the posterior draws cannot be trusted.

Partition	**PMMH**			**Gibbs**		
Size	5	20	100	5	20	100
Computation Time (sec)	20000	68800	328000	1410	5820	29400
Minimum ESS	169	83.5	68.1	19.8	18	17.7
Minimum ESJD ($\times 10^{-11}$)	46.0	21.3	40.7	3.78	7.46	8.32
Burn-in	4500	5400	8100	100	100	100
Acceptance Probability	0.1466	0.1549	0.1172	–	–	–

Table 5.1: Comparison of PMMH and Gibbs performance for parameter θ across different partition sizes.

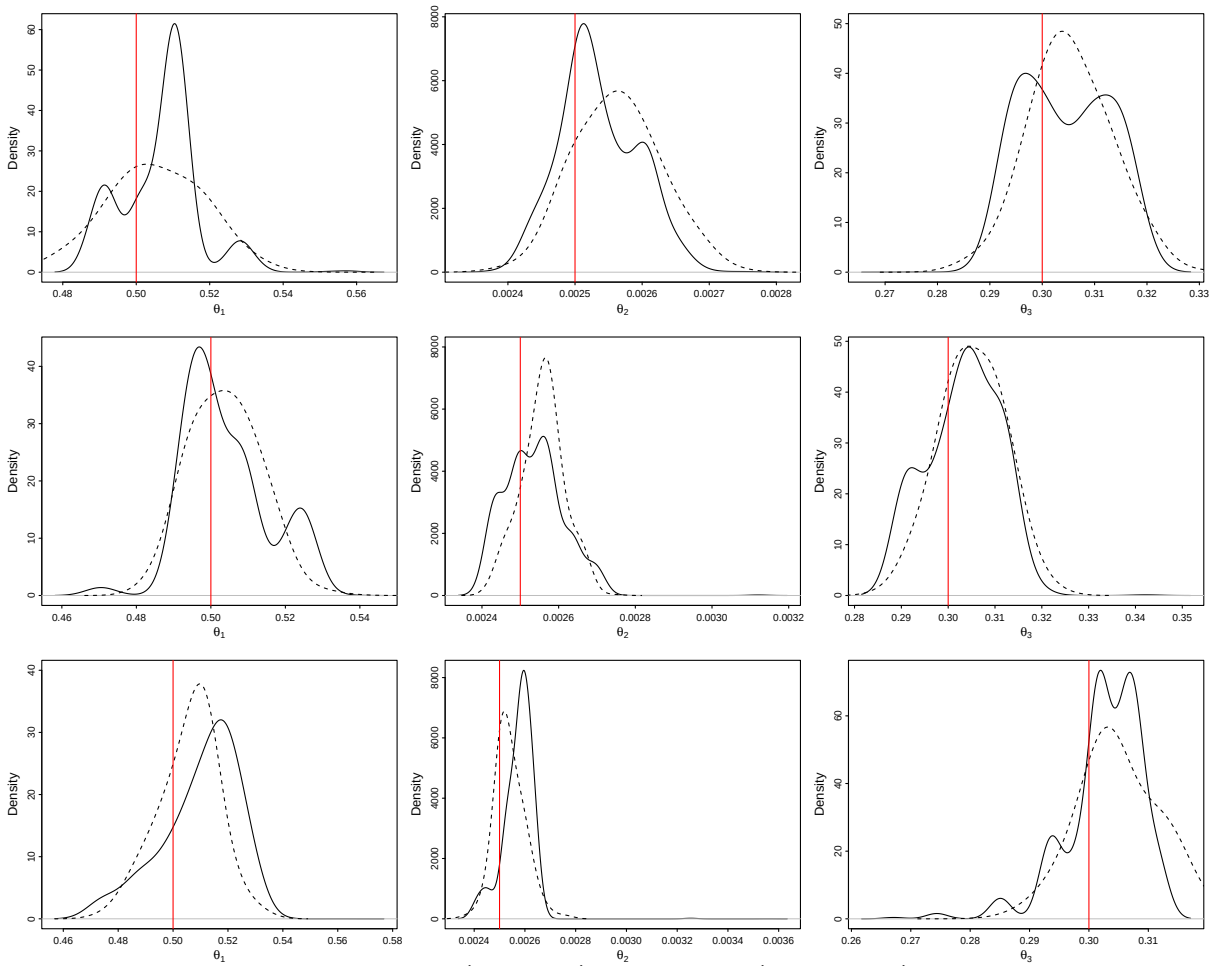

Figure 5.4: Density plots with Gibbs (solid line) and PMMH (dashed line), the simulations in each row from the top correspond to partition size 5, 20 and 100 respectively

In Figure 5.4, the density plots for each of the simulation parameters are shown. When calculating the densities of each of the output parameters, a smoothing has been applied. This smoothing has been done by widening the bandwidth over which the density has been calculated. The new

value for the bandwidth is two times the original, giving more stability in the density estimates. It can be seen that in the PMMH algorithm, the densities are single-peaked and look very similar to normal distributions with means similar to the true ground truth of the dataset. The Gibbs sampler has a much larger variance in its density estimates. However, its peaks largely coincide with the ground-truth estimates.

Overall, the PMMH algorithm outperforms the Gibbs sampler in every metric, apart from computation time and the Gibbs sampler seems largely unaffected by the parameter changes explored here. However, within the analysis of the PMMH algorithm, it can be seen that the larger the partition size, the better the algorithm performs relative to the number of iterations that are used for metric calculation.

5.5.2 Inter-observation Variations

It is now necessary to continue the discussion and investigate the effect of the inter-observations on the algorithms. In Figure 5.5, the ACF plots for each of the simulated outputs are shown. It is clear to see that the autocorrelation between simulated values decreases massively as more data is provided to the PMMH sampler. However, the Gibbs sampler is consistently poor in its relative autocorrelations. Despite the PMMH being greatly affected by the amount of data observed, it still performs better than the Gibbs sampler at each level.

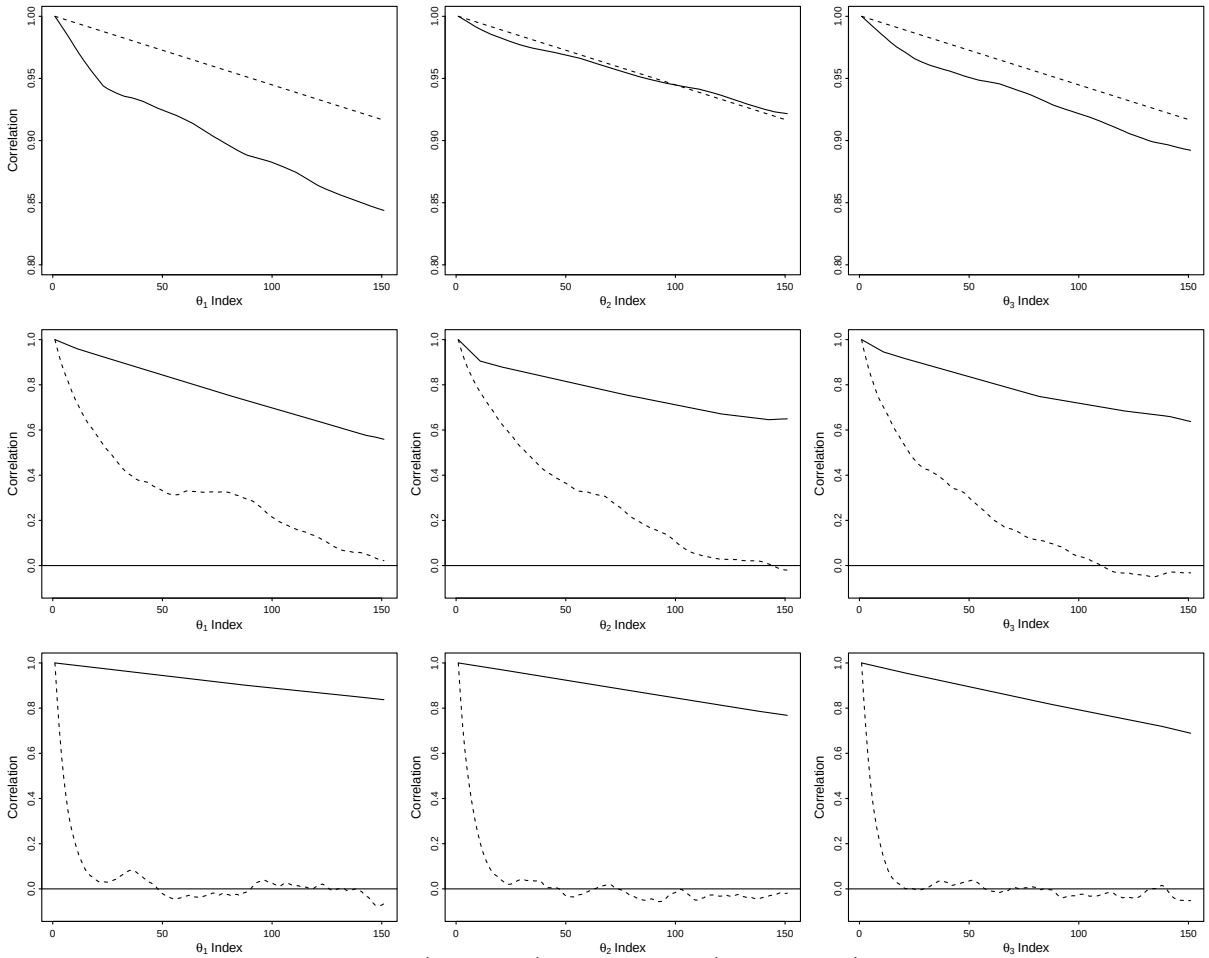

Figure 5.5: ACF plots with Gibbs (solid line) and PMMH (dashed line), the simulations in each row from the top correspond to inter-observation size 5, 1 and 0.1 respectively

Inter-observation	**PMMH**			**Gibbs**		
	5	1	0.1	5	1	0.1
Computation Time (sec)	1520	68840	129000	1190	5820	13700
Minimum ESS	16.6	83.5	660	24.1	18.0	12.8
Minimum ESJD ($\times 10^{-11}$)	0.372	21.3	90.9	23.2	7.46	1.39
Burn-in	4500	5400	1800	100	100	100
Acceptance Probability	0.0018	0.1549	0.2618	–	–	–

Table 5.2: Comparison of PMMH and Gibbs methods across inter-observation intervals.

Table 5.2 shows the metrics against which the algorithms are measured. As mentioned in the previous sub-section, these metrics are dependent on the size of the burn-in. However, here it is much more apparent that as the amount of data increases, the performance of the PMMH algorithm increases. Comparatively, the Gibbs sampler performs poorly irrespective of the amount of data it has. This further demonstrates that the PMMH algorithm is the superior algorithm, despite its vulnerability to the amount of data it is run through.

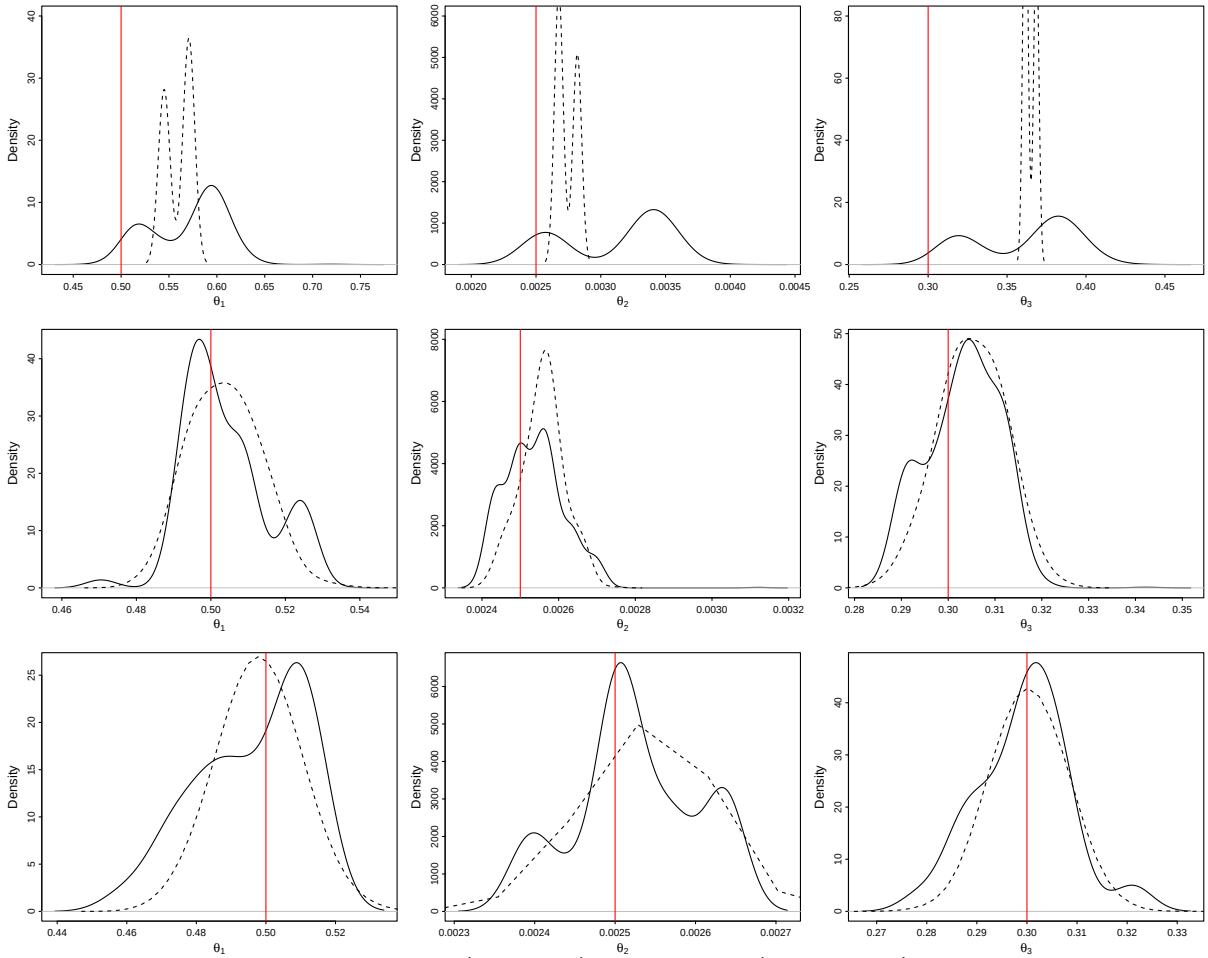

Figure 5.6: Density plots with Gibbs (solid line) and PMMH (dashed line), the simulations in each row from the top correspond to partition size 5, 1 and 0.1 respectively

In Figure 5.6, the density plots for each of the simulated outputs are shown, with smoothing applied again. The same bandwidths as stated previously are used and one can see that for less data the

density plots are considerably less localised to the ground truth, improving as more data becomes available.

Overall, the PMMH algorithm outperforms the Gibbs sampler again when subject to varying amounts of data. The Gibbs sampler is not responsive to these fluctuations. However, the PMMH algorithm is vulnerable to the amount of data it has to use, but despite this, it still performs well relative to the number of burn-in iterations.

In summary, this section has overseen the comparison of two different MCMC schemes, the Gibbs sampler and the PMMH algorithm. It is abundantly clear through the analysis of these algorithms that the PMMH schema exceeds the Gibbs sampler in every metric. The autocorrelation between points in the PMMH schema is far lower than the Gibbs sampler. The density estimates of the PMMH scheme are well-peaked around the mean with low variance and an overall smoother density compared to the Gibbs sampler. The performance metrics for the PMMH schema, ESS and ESJD are within the desired range and far exceed the Gibbs sampler. The only downside of the PMMH scheme is that the computation time is orders of magnitude higher than that of the Gibbs sampler. However, this does not devalue the PMMH schema. This is due to the fact that the amount of information obtained through the Gibbs sampler is a similar order of magnitude smaller. Suggesting an iteration increase would be required, resulting in similar computation times for both schemes. However, the Gibbs sampler would likely still produce unreliable parameter chains. Fortunately, for the Gibbs sampler, in spite of its shortcomings, it appears to be relatively unaffected by the parameter variations investigated. Whether this is as a result of the Gibbs sampler performing poorly to begin with or it is truly unaffected by this requires further investigation. This concludes our discussion of the PMMH algorithms application to the LV model and comparison to the Gibbs sampler.

Chapter 6

Conclusion

In this report, we have conducted a comprehensive investigation into the pseudo-marginal Metropolis-Hastings (PMMH) algorithm, examining both its theoretical foundation and practical application in stochastic dynamical systems. To simulate realistic scenarios, we used a partially observed dataset, representing the kind of periodically observed data often encountered in real-world processes. To address the issue of the partially observed data, we employed the Durham and Gallant (DG) bridge construct within an importance sampling framework, enabling simulation of latent paths over a partition of our choosing between the observed data points to attempt to capture the dynamics of the system.

The PMMH algorithm was first applied to Geometric Brownian Motion (GBM) to evaluate its behaviour and convergence properties in an exploratory setting. Following this, we applied the method to the Lotka-Volterra (LV) model, allowing us to assess and compare its performance to a Gibbs sampler in this partially-observed context.

The application of PMMH to the Lotka-Volterra model demonstrated its strong performance in estimating the parameters of the underlying process in a partially observed dataset, with efficiency metrics such as effective sample size and acceptance rate falling within the optimal range. While the algorithm is computationally expensive compared to the Gibbs sampler, it consistently produced better-mixed parameter chains and more accurate parameter estimates. These results demonstrate the algorithm's suitability for inference problems where observational noise or partial observation limits the performance of more standard methods.

Our empirical findings support the well-documented mixing problems of the Gibbs sampler: as the number of intermediate time points decreases, so does the dependence between the parameters and the latent path. In addition to this our empirical findings show that the PMMH is particularly well-suited to models in the contexts investigated. We have also found that performance is highly sensitive to the choice of proposal mechanism for the importance sampler, reinforcing the importance of tuning and design decisions in practical implementations. Both schemes struggle with observations, due to the linearity of the bridge construct, motivating a different construct from the one considered here. Overall, the results highlight PMMH as a reliable, but computationally demanding approach for partially observed data.

This work highlights the PMMH algorithm as a powerful inference tool for stochastic systems with intractable likelihoods, particularly in the context of partially observed data. In the future, this work can be extended in a number of directions. One such direction is the application of PMMH to scenarios where only a subset of the SDE components are observed. This would require modification of the bridge construct and incorporation of a particle filter to estimate the observed data likelihood. An example of this setting is a predator-prey system where only the predator can be observed.

Bibliography

[1] C.J. Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

[2] A. Golightly and C. Sherlock. Augmented Pseudo-Marginal Metropolis–Hastings for Partially Observed Diffusion Processes. *Statistics and Computing*, 32, 2022.

[3] B. Eraker. MCMC Analysis of Diffusion Models With Application to Finance. *Journal of Business Economic Statistics*, 19(2):177–191, 2001.

[4] M. Laine. *Adaptive MCMC Methods with Applications in Environmental and Geophysical Models*. Doctoral dissertation, University of Helsinki, 2008.

[5] G. Hamra, R. MacLehose, and D. Richardson. Markov Chain Monte Carlo: An Introduction for Epidemiologists. *International Journal of Epidemiology*, 42(2):627–634, 2013.

[6] J.W. Haefner. *Modeling Biological Systems: Principles and Applications*. Springer US, 2 edition, 2005.

[7] P.J. Wangersky. Lotka-Volterra Population Models. *Annual Review of Ecology and Systematics*, 9:189–218, 1978.

[8] G.A. Whitaker, A. Golightly, R.J. Boys, and C. Sherlock. Improved Bridge Constructs for Stochastic Differential Equations. *Statistics and Computing*, 27:885–900, 2017.

[9] Oksendal B. *Stochastic Differential Equations: An Introduction with Applications*, volume 6. Springer Publishing.

[10] E.L. Ince. *Ordinary Differential Equations*. Dover Publications, Inc., 1956.

[11] R.R. Marathe and S.M. Ryan. On the Validity of the Geometric Brownian Motion Assumption. *The Engineering Economist*, 50(2):159–192, 2005.

[12] M. Csörgő. Brownian Motion—Wiener Process. *Canadian Mathematical Bulletin*, 22(3):257–279, 1979.

[13] C. Villegas. On Qualitative Probability /sigma-Algebras. *The Annals of Mathematical Statistics*, 35(4):1787 – 1796, 1964.

[14] D.W. Stroock. *An Introduction to Markov Processes*. Graduate Texts in Mathematics. Springer Berlin Heidelberg, 2013.

[15] D.J. Higham. An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations. *SIAM Review*, 43(3):525–546, 2001.

[16] H. Öz. *Advances and Applications of Stochastic Ito-Taylor Approximation and Change of Time Method in the Financial Sector*. PhD thesis, Middle East Technical University, 2013.

[17] G. Leobacher and M. Szölgyenyi. A strong order 1/2 method for multidimensional SDEs with discontinuous drift. *The Annals of Applied Probability*, 27(4), August 2017.

[18] H. Lamba, J.C. Mattingly, and A.M. Stuart. An adaptive Euler–Maruyama scheme for SDEs: convergence and stability. *IMA Journal of Numerical Analysis*, 27(3):479–506, 2006.

[19] J.R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, UK, 1997.

[20] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335, 1995.

[21] C.P. Robert. The Metropolis–Hastings Algorithm. In *Monte Carlo Statistical Methods*, Springer Texts in Statistics, pages 231–283. Springer New York, New York, NY, 1999.

[22] S. Tokdar and R. Kass. Importance Sampling: A Review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

[23] D. Gamerman and H.F. Lopes. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2 edition, 2006.

[24] P.R. Halmos. The Theory of Unbiased Estimation. *The Annals of Mathematical Statistics*, 17(1):34–43, 1946.

[25] G.B. Durham and A.R. Gallant. Numerical Techniques for Maximum Likelihood Estimation of Continuous-Time Diffusion Processes. *Journal of Business Economic Statistics*, 20(3):297–338, 2002.

[26] A. Gelman, G.O. Roberts, and W.R. Gilks. Efficient Metropolis Jumping Rules. In *Bayesian Statistics 5: Proceedings of the Fifth Valencia International Meeting*, pages 599–608. Oxford University Press, 1996.

[27] J.S. Rosenthal. Optimal Proposal Distributions and Adaptive MCMC. In Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng, editors, *Handbook of*, pages 93–112. Chapman Hall/CRC, 2011.

[28] C. Sherlock, A.H. Thiery, G.O. Roberts, and J.S. Rosenthal. On the Efficiency of Pseudo-Marginal Random Walk Metropolis Algorithms. *The Annals of Statistics*, 43(1):238–275, 2015.

[29] A. Li, L. Wang, T. Dou, and J.S. Rosenthal. Exploring the Generalizability of the Optimal 0.234 Acceptance Rate in Random-Walk Metropolis and Parallel Tempering Algorithms, 2024.

[30] G.O. Roberts, A. Gelman, and W.R. Gilks. Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.

[31] R.V. Lenth. Some Practical Guidelines for Effective Sample Size Determination. *The American Statistician*, 55(3):187–193, 2001.

[32] A. Gelman and C. Pasarica. Adaptively Scaling the Metropolis Algorithm Using Expected Squared Jumped Distance. *Statistica Sinica*, 20(1):345–367, 2010.

[33] C. Sherlock. Variance Bounds and Robust Tuning for Pseudo-Marginal Metropolis–Hastings Algorithms, 2024.

[34] A. Golightly and C.S. Gillespie. *Simulation of Stochastic Kinetic Models*, pages 169–187. Humana Press, Totowa, NJ, 2013.

[35] C.V. Rao and A.P. Arkin. Stochastic Chemical Kinetics and the Quasi-Steady-State Assumption: Application to the Gillespie Algorithm. *Journal of Chemical Physics*, 118(11):4999–5010, 2003.

[36] F.M. Spieksma. Kolmogorov Forward Equation and Explosiveness in Countable State Markov Processes. *Annals of Operations Research*, 241(1):3–22, 2016.

[37] A.E. Gelfand and A.F.M. Smith. Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.