Approximate Methods For Otherwise Intractable Problems

KIERAN FRASER RICHARDS

Approximate Methods For Otherwise Intractable Problems

Kieran Richards

A thesis presented for the degree of
Doctor of Philosophy at Durham University

Statistics
Department of Mathematical Sciences
Durham University
United Kingdom

March 2023

Approximate Methods For Otherwise Intractable Problems

Kieran Richards

Submitted for the degree of Doctor of Philosophy

March 2023

Abstract

Recent Monte Carlo methods have expanded the scope of the Bayesian statistical approach. In some situations however, computational methods are often impractically burdensome. We present new methods which reduce this burden and aim to extend the Bayesian toolkit further. This thesis is partitioned into three parts. The first part builds on the Approximate Bayesian Computation (ABC) method. Existing ABC methods often suffer from a local trapping problem which causes inefficient sampling. We present a new ABC framework which overcomes this problem and additionally allows for model selection as a by-product. We demonstrate that this framework conducts ABC inference with an adaptive ABC kernel and extend the framework to specify this kernel in a completely automated way. Furthermore, the ABC part of the thesis also presents a novel methodology for multifidelity ABC. This method constructs a computationally efficient sampler that minimises the approximation error induced by performing early acceptance with a low fidelity model. The second part of the thesis extends the Reversible Jump Monte Carlo method. Reversible Jump methods often suffer from poor mixing. It is possible to construct a "bridge" of intermediate models to facilitate the model transition. However, this scales poorly to big datasets because it requires many evaluations of the model likelihoods. Here we present a new method which greatly improves the scalability at the cost of some approximation error. However, we show that under weak conditions this error is well controlled and convergence is still achieved. The third part of the thesis introduces a multifidelity spatially clustered Gaussian process model. This model enables cheap modelling of nonstationary spatial statistical problems. The model outperforms existing methodology which perform poorly when predicting output at new spatial locations.

Declaration

The work in this thesis is based on research carried out under the supervision of Dr Georgios Karagiannis within the Department of Mathematical Sciences at Durham University. No part of this thesis has been submitted elsewhere for any degree or qualification. This studentship was funded by the EPSRC Doctoral Training Partnership.

Acknowledgements

Firstly I would like to thank my supervisor, Dr Georgios Karagiannis for his support and guidance throughout the course of my doctoral studies.The academic freedom that Georgios has extended to me has enabled me to grow as a statistician and his advice has helped shape my work.

I would also like to thank the department staff who help ensure that the department runs smoothly and remains a welcoming environment. Particularly over the last two years, their efforts have been invaluable during this global pandemic. I would particularly like to thank Pam, who has now retired but was especially welcoming when I started my PhD.

Next, I would thank my family and friends. Their encouragement and support has helped me to remain motivated during these years. Here I would also particularly thank Louise, Caitlin and Eilish, for their support was particularly important during the pandemic. Moreover I would especially thank my Mum, whose support has been especially crucial throughout.

Dedicated to
My family, friends and teachers.

Contents

Part I

Adaptive Approximate Bayesian Computation

Chapter 1

Overview and context

1.1 Accept-reject ABC

Given data $y \in \mathcal{Y}$ and a model of the data generating process $\mathcal{M} = \{f(\cdot|\theta); \theta \in \Theta\}$ we aim to make inference about the parameters of the model $\theta \in \Theta$. This can be done by updating a prior $\pi(\theta)$ with the data through the likelihood function of the model $f(y|\theta)$. This leads to the posterior distribution

$$\pi(\theta|y) = \frac{\pi(\theta) f(y|\theta)}{\pi(y)} \tag{1.1}$$

where $\pi(y) = \int \pi(\theta) f(y|\theta) \, \mathrm{d}\theta$ is the marginal likelihood for continuous θ and is typically intractable. Monte Carlo methods use sampling methods such as rejection sampling, importance sampling and Markov chain Monte Carlo (MCMC) to facilitate this inference whilst avoiding the need to calculate the marginal likelihood. In the last couple of decades interest has risen in likelihood free methods that enable inference even when the likelihood function is also intractable. Likelihood free inference can be performed when the likelihood function is intractable or computationally challenging to evaluate but a generative model $f(x|\theta)$ is available, from which we can simulate pseudodata $x \in \mathcal{Y}$ from the model. Synthetic likelihood methods (Wood 2010) use these pseudodata to construct an approximation of the likelihood by assuming that some vector of summary statistics $S\left(x\right)$ follows a multivariate normal distribution and substitute this approximation in (1.1). Alternatively, Approximate Bayesian Computation (ABC) methods (M. A. Beaumont, W. Zhang and Balding 2002) compare pseudodata x to the observed data y allowing inference to be made with the parameters used to simulate the data based on how close the pseudodata are to the real data. In this thesis we focus on ABC methods that fascilitate likelihood free inference.

One of the first ABC methods is the ABC rejection sampler. We define a discrepancy $u = |y - x|$. Consider first the case where the data are generated from a discrete model. Observe that the probability of generating data given θ that exactly match the observed data is precisely the likelihood function at θ. Then the event that the generated data match the observed data can be used to accept θ without ever evaluating the likelihood function. It follows that the algorithm shown in Algorithm 1.1 samples exactly from the posterior distribution (1.1).

Algorithm 1.1 The exact ABC rejection sampler

```
Inputs: observed data $y$, number of iterations N
Loop over $i = 1, ..., N$
    1. Generate $\theta_i$ from $\pi\left(\theta\right)$
    2. Generate $x_i$ from $f\left(x|\theta_i\right)$
    3. Calculate $u_i = |y - x_i|$
    4. Accept $\theta_i$ if $u_i = 0$
End Loop
Outputs: accepted parameter values $\theta'$
```

In the continuous case the exact ABC rejection sampler is unusable because the acceptance probability is determined by the probability of generating data that exactly match the observed data. The probability of an exact match in the continuous case is zero and hence the exact ABC algorithm will have an acceptance rate of zero. To address this we relax the condition that the observed and simulated data match exactly. Instead we only require that the simulated data are ‘close enough’ to the observed data. To determine if the simulated data are close enough we introduce a kernel function $g_{\varepsilon}(u)$ on the discrepancies with kernel parameter $\varepsilon > 0$ which enables a non-zero acceptance probability at small values of u. This kernel function is specified such that $g_{\varepsilon}(u) \geq 0$ for all u and $\int g_{\varepsilon}(u)\,\mathrm{d}u = 1$ so that g_{ε} is a valid probability density function. Furthermore, we specify g_{ε} such that u has mean 0 and finite variance under the distribution $g_{\varepsilon}(u)$. A common choice for this kernel is the uniform kernel

$$g_{\varepsilon}(u) = \begin{cases} \frac{1}{\varepsilon} & u < \varepsilon \\ 0 & u \geq \varepsilon \end{cases} \tag{1.2}$$

for $\varepsilon > 0$. Using a kernel function allows the acceptance probability to become non-zero, enabling the algorithm to be used in the continuous case. However, we are no longer sampling exactly from the true posterior but from an approximation to the posterior

$$\pi_{\varepsilon}(\theta, u|y) = \frac{\pi(\theta)\, p(u|\theta)\, g_{\varepsilon}(u)}{\pi_{\varepsilon}(y)} \tag{1.3}$$

which we call the ABC posterior. The denominator, $\pi_{\varepsilon}(y)$, is the ABC marginal likelihood and is defined as

$$\pi_{\varepsilon}(y) = \int \pi(\theta)\, p(u|\theta)\, g_{\varepsilon}(u)\, \mathrm{d}u\mathrm{d}\theta \tag{1.4}$$

Algorithm 1.2 The ABC rejection algorithm

```
Inputs: observed data $y$, kernel parameter $\varepsilon$, number of iterations N
Loop over $i = 1, ..., \mathbf{N}$
    1. Generate $\theta_i$ from $\pi(\theta)$
    2. Generate $x_i$ from $f(x|\theta_i)$
    3. Calculate $u_i = |y - x_i|$
    4. Accept $\theta_i$ with probability $g_\varepsilon(u_i)$
End Loop
Outputs: accepted parameter values$\theta_i$
```

For large datasets Algorithm 1.2 is still impractical. The high dimensionality of x means that the discrepancy $u = |y - x|$ is seldom small and it can be difficult to make useful inference. Instead we can use summary statistics $S(x)$ and define the discrepancy as $u = |S(y) - S(x)|$ to describe the data as shown in Algorithm 1.3. Compared to the full dataset, summary statistics can have much smaller dimensionality and allow for useful inference even on large datasets.

Algorithm 1.3 The ABC rejection algorithm with summary statistics and a general kernel

```
Inputs: observed data $y$, kernel parameter $\varepsilon$, number of iterations N
Loop over $i = 1, ..., \mathbf{N}$
    1. Generate $\theta_i$ from $\pi(\theta)$
    2. Generate $x_i$ from $f(x|\theta_i)$
    3. Calculate $u_i = |S(y) - S(x_i)|$
    4. Accept $\theta_i$ with probability $g_\varepsilon(u_i)$
Outputs: accepted parameter values $\theta_i$
```

If these summary statistics $S(x)$ are also sufficient statistics for the model then there is no further approximation error introduced. However, for more realistic problems where likelihood free inference is typically applied, sufficient statistics are not usually available. Instead we must resort to summary statistics which are not necessarily sufficient and introduce an additional source of approximation error to the approximate model. Choosing appropriate summary statistics is one of the major difficulties of ABC but is outside the scope of this thesis. The interested reader can find suitable methods for choosing them in (Blum 2010; Nunes and Balding 2010). When the prior $\pi(\theta)$ offers poor support for the data, rejection sampling can be very inefficient, requiring many simulations to obtain even a small sample from the posterior and wasting considerable computational resources.

1.2 MCMC-ABC

A more efficient ABC sampler than ABC rejection is the Markov Chain Monte Carlo ABC algorithm (Marjoram et al. 2003). We incorporate ABC into the Metropolis-Hastings algorithm by introducing a proposal distribution $q(\theta'|\theta_t)$ which proposes a new set of parameters θ' given a current set of parameters θ_t. The full ABC MCMC algorithm is shown in Algorithm 1.4. By making proposals locally based on the current state of the Markov chain we aim to improve the acceptance rate of ABC compared to ABC rejection sampling (Sisson, Fan and M. Beaumont 2018; Sisson, Fan and Tanaka 2007). For this reason MCMC-ABC has become a popular method in several areas of research where likelihood free sampling is important including in ecology (M. A. Beaumont 2010; Butler et al. 2006) and epidemiology (Tanaka et al. 2006).

Algorithm 1.4 The ABC Metropolis-Hastings (MCMC-ABC) algorithm

```
Inputs: observed data $y$, kernel parameter $\varepsilon$, number of iterations N, initial parameter value $\theta_1$
Loop over $t = 1, ..., \mathbf{N}$
    1. Generate $\theta'$ from $q(\theta|\theta_t)$
    2. Generate $x'$ from $f(x|\theta')$
    3. Calculate $u' = |S(y) - S(x')|$
    4. Accept $\theta_{t+1} = \theta'$ with probability $\alpha = \min(1, R_{ABC})$ with
           $R_{ABC} = \frac{\pi(\theta') g_\varepsilon(u') q(\theta_t|\theta')}{\pi(\theta_t) g_\varepsilon(u) q(\theta'|\theta_t)}$
    5. Otherwise $\theta_{t+1} = \theta_t$
End Loop
Output: parameter values $\theta_1, ..., \theta_{\mathrm{N}}$
```

The MCMC-ABC algorithm achieves an improved acceptance rate compared to ABC rejection particularly if the prior offers poor support for the data (Sisson, Fan and Tanaka 2007). This is because moves can be proposed locally and hence by introducing dependence in the generated parameters we make more informed proposals. Nevertheless, despite the improved acceptance rate over ABC rejection sampling MCMC-ABC can still be very inefficient because it can suffer from a local trapping problem. In particular, the acceptance probability of MCMC-ABC is directly proportional to the likelihood function. This can result in very low acceptance probabilities in regions of low likelihood if the proposal function is also poor (Sisson, Fan and Tanaka 2007). This Markov chain becomes stuck in these regions and takes a long time to escape, producing misleading inference.

1.3 SMC ABC

The Sequential Monte Carlo ABC (SMC-ABC) method proposed in Sisson, Fan and Tanaka (2007) aims to circumvent the local trapping problem of MCMC-ABC. The

SMC ABC method returns to the simple function (1.2) and starts with a very large ABC tolerance. The algorithm then moves the particles through a monotonically decreasing sequence of ABC tolerances to achieve a final sample of particles with a low ABC tolerance. As shown in Algorithm 1.5 this involves weighting the samples. We define the effective sample size (ESS) at step t as

$$\text{ESS}_t = \left(\sum_{i=1}^{N} \left(W_t^{(i)} \right)^2 \right)^{-1}$$

where $W_t^{(i)}$ is the ith particle weight and the population is of size N. We set a threshold E such that when $\text{ESS}_t < E$ we resample the particles with replacement to prevent degeneracy. Thus SMC-ABC can be seen as an extension of importance sampling (Sisson, Fan and Tanaka 2007). This results in a sample of parameters generated independently from the ABC posterior which aims to avoid the local trapping problem that arises because of the dependence in the MCMC-ABC chain.

Algorithm 1.5 The SMC-ABC algorithm

```
Inputs: observed data $y$, kernel parameters $\varepsilon_1, ..., \varepsilon_K$, number of SMC tolerances $K$, population size N, parameters $\theta_1^{(0)}, ..., \theta_N^{(0)}$
Loop over $k = 1, ..., K$
Loop over $i = 1, ..., \mathbf{N}$
    1. Generate $\theta_i^{(k)}$ from $q_k\left(\theta|\theta_i^{(k-1)}\right)$
    2. Generate $x_i^{(k)}$ from $p\left(x|\theta_i^{(k)}\right)$
    3. Calculate $u_i^{(k)} = \left|S\left(y\right) - S\left(x_i^{(k)}\right)\right|$
    4. Assign weight $W_i^{(k)}$ to $\theta_i^{(k)}$ such that
       $$W_i^{(k)} = W^{(k-1)} \frac{\pi\left(\theta_i^{(k-1)}\right) g_{\varepsilon_k}\left(u_i^{(k-1)}\right) q\left(\theta_i^{(k)}|\theta_i^{(k-1)}\right)}{\pi\left(\theta_i^{(k)}\right) g_{\varepsilon_k}\left(u_i^{(k)}\right) q\left(\theta_i^{(k-1)}|\theta_i^{(k)}\right)}$$
End Loop
    5. Renormalise the weights so that they sum to 1 over $i = 1, ..., \mathrm{N}$
    6. If $\mathrm{ESS}_t < E$ then resample with replacement from the particles
End Loop
Output: parameter values $\theta_1^{(K)}, ..., \theta_{\mathrm{N}}^{(K)}$
```

Starting with a large tolerance and reducing the tolerance through a sequence of stricter ABC posteriors allows the algorithm to achieve high acceptance rates whilst avoiding the local trapping problem (Sisson, Fan and Tanaka 2007). However, in Algorithm 1.5 we observe that the SMC-ABC algorithm does not keep every set of parameters that are generated. So whilst the SMC-ABC algorithm is much less wasteful than ABC rejection there are still many moves that are not stored but must be wasted. Moreover, the sequence of ABC tolerances must be chosen carefully. If the ABC tolerances decrease too fast then the sample may become degenerate as many particles are rejected. Alternatively, if the tolerances decrease too slowly then the SMC sampler will be slow to achieve a reasonable approximation to the posterior and waste computational resources. Moreover, a final tolerance must be determined such that the final sample is a reasonable approximation to the posterior.

When made poorly these choices for the tolerance schedule can introduce unnecessary approximation error to the ABC sample. Sophisticated automatic methods to determine the tolerance schedule (Silk, Filippi and Stumpf 2013; Simola et al. 2021) can be used however these methods can reintroduce the local trapping issue to some problems since if the particles are contained within local energy minima then the automatic schedule may finish early.

1.4 Model selection in the ABC setting

The Bayes factor BF is the ratio of the marginal likelihoods from two models $\mathcal{M}_1 = \{f_1(\cdot|\theta); \theta_1 \in \Theta_1\}$ and $\mathcal{M}_2 = \{f_2(\cdot|\theta); \theta_2 \in \Theta_2\}$ under comparison and is often used for model choice. Care must be taken when doing this in the ABC framework since we cannot compute the marginal likelihood $\pi(y|\mathcal{M}_k)$ and must instead use the ABC marginal likelihood $\pi_\varepsilon(y|\mathcal{M}_k)$ as an estimate. When we generate pseudodata from the prior the ABC marginal likelihood can be estimated using the acceptance rate of the ABC rejection sampler since this acceptance rate is an ABC estimate of the predictive prior near the observed data. This estimator is only guaranteed to be unbiased for the true marginal likelihood when $\varepsilon = 0$ and the full data are used (Wilkinson 2013). Even when $\varepsilon = 0$ if summary statistics are used the ABC Bayes factor can be biased (Robert, Cornuet et al. 2011). Even when sufficient summary statistics are chosen for each model this bias can be present since the union of the two sets of sufficient statistics is not necessarily sufficient for model comparison because the summary statistics may not be sufficient for the joint model (Robert, Cornuet et al. 2011). However, Marin et al. (2014) showed that when the expectations of the summary statistics are asymptotically different under the two models, ABC model choice selects the true model asymptotically (Marin et al. 2014).

Chapter 2

Stochastic Approximation Monte Carlo ABC

2.1 Stochastic Approximation Monte Carlo

In Chapter 1 we discussed how several existing ABC methods suffer from a local trapping problem which can result in poor quality ABC samples. In this chapter we aim to address the local trapping problem by introducing ideas from the Stochastic Approximation Monte Carlo (SAMC) algorithm. The core idea of the SAMC algorithm is to partition the sample space into subregions and then construct a Markov chain that samples from the target distribution whilst leading to a random walk through the subregions (Liang 2009; Liang 2014; Liang, Liu and R. Carroll 2011; Liang, Liu and R. J. Carroll 2007). By doing this SAMC aims to ensure that all of the subregions are visited according to desired sampling frequencies ϖ_j, and therefore improves upon the mixing of the MCMC algorithm. Suppose we have a target density $\pi(\theta)$ on a sample space Θ and let $E_1, E_2, ..., E_m$ be m disjoint subregions of Θ such that $E_j = \{\theta \in \Theta; \varepsilon_{j-1} < u(\theta) \leq \varepsilon_j\}$ partitions the sample space by some function $u(\theta)$ often chosen to be the energy function, $U(\theta) = -\log(\pi(\theta))$. Finally we set desired sampling frequencies $\varpi_1, \varpi_2, ..., \varpi_m$ which determine how frequently the sampler aims to visit each subregion. Then the target SAMC density is

$$\pi_{SAMC}\left(\theta\right) = \sum_{j=1}^{m} \frac{\varpi_j}{w_j} \pi\left(\theta\right) I_{E_j}\left(\theta\right)$$

where $w_j = \int_{E_j} \pi(\theta) d\theta$ is the normalising constant of the truncated density $\pi\left(\theta\right)$ in E_j and I_{E_j} is an indicator taking the value one when the sample is in the subregion E_j and is zero otherwise. The constants w_j are typically intractable so we take a working vector ϕ, such that $\exp(\phi_j) \propto w_j/\varpi_j$ which we can estimate up to a normalising constant. The SAMC algorithm as shown in Algorithm 2.1 consists of two stages. The first is the usual sampling step, and the second updates the estimates of ϕ to enable the use of stochastic approximation to solve the equation

$$\int H(\phi, \theta) \pi_{SAMC}\left(\theta | x\right) \mathrm{d}\theta = 0 \tag{2.1}$$

where $H(\phi, \theta) = p - \varpi$, with p a vector indicating which subregions are visited. To ensure convergence the updating step uses a decreasing step size, γ_t, for the update. This step size must satisfy

$$\sum_{t=1}^{\infty} \gamma_t = \infty, \qquad \sum_{t=1}^{\infty} \gamma_t^{\zeta} < \infty$$

for some $\zeta \in (1, 2)$. In the examples throughout this thesis we use a cooling sequence of the form

$$\gamma_t = \frac{t_0}{\max\left(t_0, t^b\right)}$$

which satisfies the requirements.

Algorithm 2.1 The SAMC algorithm

```
Inputs: number of iterations N, initial parameter value $\theta_1$, subregions $E_j$
Loop over $t = 1, ..., \mathrm{N}$
    1. Sampling step
        (a) Generate $\theta'$ from the proposal density $q(\theta|\theta_t)$
        (b) Accept $\theta_{t+1} = \theta'$ with probability $\alpha = \min(1, R)$ with
                $R_{ABC} = \frac{\pi(\theta')q(\theta_t|\theta')\exp\left(\phi_{t,j(\theta_t)}\right)}{\pi(\theta_t)q(\theta'|\theta_t)\exp\left(\phi_{t,j(\theta')}\right)}$
        (c) Otherwise $\theta_{t+1} = \theta_t$
    2. Updating step
        (a) Compute $\boldsymbol{p}_{t+1}$ such that $[p_{t+1}]_j = I_{E_j}(\theta_{t+1})$
        (b) Compute $\phi_{t+1} = \phi_t + \gamma_{t+1}(\boldsymbol{p}_{t+1} - \varpi)$
End Loop
Output: parameter values $\theta_1, ..., \theta_{\mathrm{N}}$, estimate $\phi_{\mathrm{N}}$
```

Importantly the SAMC algorithm weights the subregions by a self-adjusting mechanism. This self adjusting mechanism increases the probability of visiting a different subregion than the current subregion. It is precisely this self adjusting mechanism that essentially provides immunity to the local trapping problem that can occur in MCMC.

2.2 The SAMCABC algorithm

We propose a new algorithm called Stochastic Approximation Monte Carlo ABC (SAMCABC) which introduces ideas from the SAMC method to the ABC framework. Consider that the sample space of the MCMC-ABC is (Θ, U) and we can write our partition as $E_1, E_2, \ldots, E_{m+1}$ where:

$$E_j = \{(\theta, x) \in (\Theta, X)\,; \varepsilon_{j-1} < u(\theta, x) \leq \varepsilon_j\}$$

with $\varepsilon_0 = -\infty$, $\varepsilon_{m+1} = \infty$ and $\{\varepsilon_j; \varepsilon_j \in \mathbb{R}, j = 1 : m\}$ a grid for $m > 0$. Now $w_j = \int_{E_j} \pi_\varepsilon(\theta, u|y) \mathrm{d}\theta \mathrm{d}u$ are the marginal ABC likelihood truncated to the subregions E_j. Consider the algorithm when we set $u\left(\theta', x'\right) = \left|S\left(x\right) - S\left(x'\right)\right|$ to be the discrepancy. By considering only the subregions up to E_m this is MCMC-ABC with a kernel with support truncated to $[0, \varepsilon_m]$ with a partition over the discrepancies. We have the SAMCABC target distribution

$$\pi_{SAMCABC}\left(\theta, u|x\right) = \sum_{j=1}^{m} \frac{\varpi_j}{w_j} \pi\left(\theta\right) p\left(x|\theta\right) g_\varepsilon\left(u\right) I\left(u \in E_j\right) \tag{2.2}$$

achieved using the SAMCABC algorithm as shown in Algorithm 2.2.

By constructing a random walk through the subregions the SAMCABC algorithm aims to mitigate the local trapping problem of the algorithms discussed in Chapter 1 by discouraging the chain from remaining stuck. Furthermore, by choosing ϖ such that the SAMC weights, $\frac{\varpi_j}{w_j}$, are larger for smaller j, the SAMCABC can be encouraged to focus on sampling regions with small discrepancies which are more desirable than areas of higher discrepancy.Then SAMCABC obtains a sample which better approximates the posterior distribution in (1.1) compared to the algorithms discussed in Chapter 1.

Algorithm 2.2 The SAMC-ABC algorithm including the optional step of discarding some of the high discrepancy samples

```
Inputs: observed data $y$, kernel parameter $\varepsilon$, number of iterations N, initial parameter value $\theta_1$, initial discrepancy $u_1$, subregions $E_j$
Loop over $t = 1, ..., \mathrm{N}$
    1. Sampling step
        (a) Generate $\theta'$ from $q(\theta|\theta_t)$
        (b) Generate $x'$ from $p\left(x|\theta'\right)$
        (c) Calculate $u' = \left|S\left(y\right) - S\left(x'\right)\right|$
        (d) Accept $\theta_{t+1} = \theta'$ and $u_{t+1} = u'$ with probability $\alpha = \min\left(1, R_{ABC}\right)$ with
            $R_{ABC} = \frac{\pi(\theta')g_\varepsilon(u')q(\theta_t|\theta')\exp\left(\phi_{t,j(\theta_t,u_t)}\right)}{\pi(\theta_t)g_\varepsilon(u)q(\theta'|\theta_t)\exp\left(\phi_{t,j(\theta',u')}\right)}$
        (e) Otherwise $\theta_{t+1} = \theta_t$ and $u_{t+1} = u_t$
    2. Updating step
        (a) Compute $\boldsymbol{p}_{t+1}$ such that $[p_{t+1}]_j = I_{E_j}\left(\theta_{t+1}, u_{t+1}\right)$
        (b) Compute $\phi_{t+1} = \phi_t + \gamma_{t+1}(\boldsymbol{p}_{t+1} - \varpi)$
End Loop
Output: parameter values $\theta_1, ..., \theta_\mathrm{N}$, discrepancies $u_1, ..., u_\mathrm{N}$,estimate $\phi_\mathrm{N}$
```

2.3 Population SAMCABC

Whilst our proposed algorithm SAMCABC mitigates the local trapping problem found in other ABC methods, it relies upon the convergence of the working vector ϕ to sample from the SAMCABC target distribution (2.2). These estimates of ϕ can be slow to converge when some subregions E_j have a low probability to be visited. Here we demonstrate how convergence can be improved by introducing population methods to the SAMCABC algorithm.

Suppose that at each iteration instead of sampling a single estimate of θ we independently sample a population of κ estimates of θ to obtain the vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_\kappa)$. Then Liang and Wu (2013) showed that we can perform the stochastic approximation step to solve Equation 2.1 by instead solving the equation

$$\int \boldsymbol{H}(\phi, \boldsymbol{\theta}) \boldsymbol{\pi}_{SAMC} \left(\boldsymbol{\theta}|x\right) d\boldsymbol{\theta} = 0$$

where

$$\boldsymbol{H}(\phi, \boldsymbol{\theta}) = \frac{1}{\kappa} \sum_{k=1}^{\kappa} H(\phi, \theta_k)$$

From this we see that $\mathbb{E}\left(\boldsymbol{\theta}\right)$ is the same under the population algorithm as in the single chain algorithm. The population SAMC algorithm provides a more precise estimate of ϕ at each iteration and hence converges faster (Liang and Wu 2013). We can directly apply the same result to SAMCABC and find that population SAMCABC converges more quickly than single chain SAMCABC.

At iteration t of the kth chain let $\left(\theta_t^{(k)}, x_t^{(k)}\right)$ be the state of the Markov chain, with a population size of κ then we have the population SAMC-ABC algorithm as shown in Algorithm 2.3.

Algorithm 2.3 The PSAMC-ABC algorithm for a population size κ

```
Inputs: observed data $y$, kernel parameter $\varepsilon$, number of iterations N, number of parallel chains $\kappa$, initial parameter values $\theta_{1,1}, ..., \theta_{1,\kappa}$, initial discrepancies $u_{1,1}, ..., u_{1,\kappa}$, subregions $E_j$
Loop over $t = 1, ..., \mathrm{N}$
    1. For $k = 1, 2, \ldots, \kappa$ simultaneously draw $\theta_{t+1,k}$ by:
        (a) Generate $\theta'_k$ $q(\theta|\theta_{t,k})$
        (b) Generate $x'_k$ $p\,(x|\theta'_k)$
        (c) Calculate $u'_k = |S\,(y) - S\,(x'_k)|$
        (d) Accept $\theta_{t+1,k} = \theta'_k$ and $u_{t+1,k} = u'_k$ with probability $\alpha = \min\,(1, R_{ABC})$ where
            $$R_{ABC} = \frac{\pi\left(\theta'_k\right) g_\varepsilon\left(u'_k\right) q\left(\theta_{t,k}|\theta'_k\right) \exp\left(\phi_{t,j\left(\theta_{t,k},u_{t,k}\right)}\right)}{\pi\left(\theta_{t,k}\right) g_\varepsilon\left(u_{t,k}\right) q\left(\theta'_k|\theta_{t,k}\right) \exp\left(\phi_{t,j\left(\theta'_k,u'_k\right)}\right)}$$
    2. Otherwise $\theta_{t+1,k} = \theta_{t,k}$ and $u_{t+1,k} = u_{t,k}$
    3. Updating step
        (a) Compute $\boldsymbol{p}_{t+1}$ such that $[p_{t+1}]_j = \frac{1}{\kappa}\sum_{k=1}^{\kappa} I_{E_j}\,(\theta_{t+1,k}, u_{t+1,k})$
        (b) Compute $\phi_{t+1} = \phi_t + \gamma_{t+1}(\boldsymbol{p}_{t+1} - \varpi)$
End Loop
Output: parameter values $\theta_{1,1}, ..., \theta_{\mathrm{N},\kappa}$, discrepancies $u_{1,1}, ..., u_{\mathrm{N},\kappa}$,estimate $\phi_{\mathrm{N}}$
```

2.4 The Adaptive Kernel of SAMCABC

2.4.1 The SAMCABC kernel function

The proposed algorithm SAMCABC can be interpreted as constructing an adaptive ABC kernel to avoid the local trapping problem. The ABC kernel $g_\varepsilon\,(\cdot)$ serves to weight the ABC sampler towards lower discrepancies. The SAMC weights also serve as an adaptive mechanism to weight the sampler towards lower discrepancy subregions. Consider the SAMC-ABC target distribution (2.2) restated here for convenience

$$\pi_{SAMCABC}(\theta, u|x) = \sum_{j=1}^{m} \frac{\varpi_j}{w_j} \pi(\theta) p(x|\theta) g_\varepsilon(u) I(u \in E_j)$$

We define a function $h_\varepsilon(\cdot)$ such that

$$h_\varepsilon(u) = \sum_{j=1}^{m} \frac{\varpi_j}{\hat{w}_j} g_\varepsilon(u) I(u \in E_j) \tag{2.3}$$

and by normalising $\hat{w}_j$ such that $\int h_\varepsilon(u)\, \mathrm{d}u = 1$ we have the following result demonstrating that $h_\varepsilon(\cdot)$ is also a kernel function satisfying our definition of an ABC kernel function given in Section 1.1. The proof of Proposition 2.4.1 can be found in Appendix A.

Proposition 2.4.1. *The function $h_\varepsilon(\cdot)$ given in (2.3) is positive everywhere and satisfies $\int h_\varepsilon(u)\, du = 1$. Furthermore a random variable u from the distribution with probability density function $h_\varepsilon(\cdot)$ has mean 0 and finite variance.*

By rewriting the SAMC-ABC target distribution as

$$\pi_{SAMCABC}(\theta, u|x) \propto \pi(\theta) p(x|\theta) h_\varepsilon(u)$$

we see that the final target distribution of SAMCABC is not the same approximation to the posterior as targeted in the ABC algorithms described in Chapter 1. Instead the SAMCABC algorithm targets an approximate posterior with a different ABC kernel. This new ABC kernel $h_\varepsilon(u)$ is a stricter approximation than the initial fixed ABC kernel by properly choosing $\{E_j\}$ and $\{w_j\}$. Thus we can see the adaptive weights as a way of adapting the kernel function to escape local traps.

2.4.2 Model selection via the adaptive kernel

Consider the interpretation of ABC as discussed by Wilkinson (2013) that ABC methods produce exact results for the wrong model. In other words, instead of sampling from the model $\mathcal{M}$, ABC methods sample implicitly from a different approximate model, $\mathcal{M}_g$. The ABC marginal likelihood estimate is therefore the unbiased estimate of the marginal likelihood of this implicit ABC model, $\pi_\varepsilon^{(g)}(y)$. Furthermore, under this interpretation the ABC Bayes factor is biased because instead of comparing the two generative models we compare the two approximate models. The ABC Bayes factor is therefore an unbiased Bayes factor for the comparison of these implicit approximate models. In this context it is desireable to estimate the marginal likelihood of the ABC model in order to perform model comparison.

The proposed SAMCABC sampler adaptively constructs the ABC kernel and thus targets a different ABC posterior given the ABC model $\mathcal{M}_h$. In particular, this means that the SAMCABC sampler also produces exact results from a wrong model but that the implicit model is different to the implicit model of standard ABC. When using the proposed algorithms we are then faced with a choice. We can weight the sample using the SAMC importance sampling weights to calculate estimates from the original ABC posterior. Alternatively, we can make estimates using the unweighted sample to obtain estimates from the ABC posterior targeted by the SAMCABC method. When the proposed algorithm successfully targets a posterior with lower ABC bias than the original it may be desirable to make estimates using the unweighted SAMCABC samples. To perform model comparison using the new implicit model $\mathcal{M}_h$ we must calculate the marginal likelihood $\pi_\varepsilon^{(h)}(y)$ of the SAMCABC posterior. The marginal likelihood $\pi_\varepsilon^{(h)}(y)$ can be estimated from the proposed methods.

Our proposed algorithm provides estimates of the ABC marginal likelihood up to a normalising constant as a byproduct. We can determine this normalising constant by comparing the estimate to an alternative estimate of the ABC marginal likelihood. One alternative estimate can be obtained from the acceptance rate of an ABC rejection sampler with kernel g_ε (Wilkinson 2013; Sisson, Fan and M. Beaumont 2018). Another alternative estimator of the ABC marginal likelihood is the following importance sampling estimator which sometimes has lower variance than the rejection sampling estimator.

$$\hat{\pi}_\varepsilon\left(y\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\pi\left(\theta_i\right)g_\varepsilon\left(u_i\right)}{q\left(\theta_i|\theta^*\right)} \tag{2.4}$$

where the (θ', u') are sampled from the Metropolis-Hastings proposal function given some fixed θ^*. The derivation of this estimator can be found in Appendix A. To minimise the variance of the estimator we take θ^* to be the maximum a posteriori estimate of θ^*. The following proposition shows how the marginal likelihood can be calculated under the SAMCABC framework and used to determine the normalising constant of the SAMCABC weights.

Proposition 2.4.2. *Consider the subregion weights $\varpi_j/\hat{w}_j$ calculated under the SAMC-ABC framework. We have that the $\hat{w}_j$ are the estimated normalising constants within each subregion E_j and that the subregions E_j form a partition on the parameter space. Then it follows that the ABC marginal likelihood $\pi_\varepsilon^{(g)}\left(y\right)$ can be calculated as:*

$$\pi_\varepsilon^{(g)}\left(y\right) = \sum_{j=1}^{m} C\hat{w}_j$$

Setting $\sum_{j=1}^{m} C\hat{w}_j$ to be equal to an alternative estimate such as (2.4) gives us the constraint required to determine an estimate of C.

Substituting our estimates $w_j = \hat{C}\hat{w}_j$ we can now obtain an estimate of the SAMCABC marginal likelihood $\pi_\varepsilon^{(h)}\left(y\right)$ without any further sampling. For instance using the estimator of the marginal likelihood in (2.4) we obtain

$$\hat{C} = \frac{n\sum_{j=1}^{m}\hat{w}_j}{\sum_{i=1}^{n}\frac{\pi(\theta')g_\varepsilon(u')}{q(\theta'|\theta^*)}}$$

Proposition 2.4.3. *Consider the SAMCABC adaptive kernel*

$$h_\varepsilon(u) \propto \sum_{j=1}^{m}\frac{\varpi_j}{w_j}g_\varepsilon(u)\,I_{E_j}(u)$$

where $\frac{\varpi_j}{w_j}$ *is the SAMC weight for subregion* E_j *and* $g_\varepsilon(u)$ *is the ABC kernel. The SAMCABC marginal likelihood* $\pi_\varepsilon^{(h)}(y)$ *can be calculated as*

$$\pi_\varepsilon^{(h)}(y) = \left(\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\int_{E_j}g_\varepsilon(u)\,du\right)^{-1}$$

Let us consider an example where we use the uniform ABC kernel which is a popular choice in the literature. If $g_\varepsilon(u)$ is the uniform ABC kernel on $u \in [0,\varepsilon]$. Then the ABC marginal likelihood, $\pi_\varepsilon^{(g)}(y)$, and the SAMCABC marginal likelihood, $\pi_\varepsilon^{(h)}(y)$, are

$$\begin{aligned}\pi_\varepsilon^{(g)}(y) &= \sum_{j=1}^{m}w_j\\ \pi_\varepsilon^{(h)}(y) &= \left(\frac{1}{m}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\right)^{-1}\end{aligned}$$

The derivation of this marginal likelihood estimate can be found in Appendix A.

2.4.3 Bias reduction via the adaptive kernel

We discuss the conditions under which the ABC posterior targeted by our proposed adaptive kernel method has lower bias than the original ABC posterior. To do this we first consider the ABC bias. In Sisson, Fan and M. Beaumont (2018), it is shown that working in the univariate case the bias in the ABC likelihood up to second order in ε can be written as

$$\hat{b}_{\varepsilon}^{(g)}\left(y|\theta\right) = \frac{1}{2}\varepsilon^{2}\sigma_{g}^{2}f''\left(y|\theta\right) \tag{2.5}$$

where $\sigma_g^2 = \int u^2 g\left(u\right) \mathrm{d}u$ is the variance of the kernel function $g\left(u\right)$ when $g_{\varepsilon}\left(u\right) = \frac{1}{\varepsilon}g\left(\frac{u}{\varepsilon}\right)$ with $u = |x-y|$ is the ABC kernel. Thus the bias in the likelihood is proportional to the kernel variance and we should choose the desired sampling density ϖ such that the variance of the adaptive ABC kernel $h\left(u\right)$ is decreased from the variance of the fixed ABC kernel $g\left(u\right)$. In this way SAMCABC not only mitigates local trapping in ABC but also reduces the ABC approximation error. We can do this by choosing ϖ such that the sampler is encouraged to sample from subregions close to $u = 0$. In particular, we can use this decrease in bias to offset an increase in bias introduced by using a larger maximum tolerance. Hence we can use the SAMCABC algorithm with a higher maximum tolerance than ABC without increasing the bias. This provides an additional mechanism through which the proposed algorithms can avoid local trapping.

It is shown in Sisson, Fan and M. Beaumont (2018) that the point-wise bias in the ABC posterior, $a_{\varepsilon}^{(g)}\left(\theta|y\right)$, can be written as:

$$a_{\varepsilon}^{(g)}\left(\theta|y\right) = \frac{b_{\varepsilon}^{(g)}\left(y|\theta\right)\pi\left(\theta\right)}{\pi_{\varepsilon}^{(g)}\left(y\right)} + \pi\left(\theta|y\right)\left(\frac{\pi\left(y\right)}{\pi_{\varepsilon}^{(g)}\left(y\right)} - 1\right)$$

where $\pi\left(y\right)$ is the true marginal likelihood. The following proposition shows that under certain conditions the ABC bias of the unweighted SAMCABC sample is lower than the ABC bias of the weighted samples. The proof of Proposition 2.4.4 can be found in Appendix A.

Proposition 2.4.4. *For small tolerance, ε, ABC kernel variance, σ_g^2, and SAMCABC kernel variance, σ_h^2, where $\sigma_h^2 \leq \sigma_g^2$ we have that the point-wise posterior bias* *of the original ABC posterior is greater in magnitude than that of the SAMCABC posterior:*

$$\left|\hat{a}_{\varepsilon}^{(h)}\left(y|\theta\right)\right| \leq \left|\hat{a}_{\varepsilon}^{(g)}\left(y|\theta\right)\right|$$

with equality everywhere if and only if $\sigma_h^2 = \sigma_g^2$.

By considering the ratio of the biases in the unweighted and weighted samples we can determine the bias reduction factor in the ABC likelihood. The following proposition demonstrates this for the widely used uniform ABC kernel. Assume $g_{\varepsilon}\left(u\right)$ is the uniform kernel with maximum tolerance ε. Then clearly the ABC kernel variance is $\sigma_g^2 = \frac{1}{3}$.

Proposition 2.4.5. *For SAMC-ABC, we have that*

$$\sigma_h^2 = \frac{1}{3m^3}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(j^3 - \left(j-1\right)^3\right)$$

and so it follows that the bias reduction factor is:

$$\frac{\hat{b}_{\varepsilon}^{(h)}\left(y|\theta\right)}{\hat{b}_{\varepsilon}^{(g)}\left(y|\theta\right)} = \frac{1}{m^3}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(j^3 - \left(j-1\right)^3\right)$$

Clearly when the bias reduction factor, $\hat{b}_{\varepsilon}^{(h)}\left(y|\theta\right)/\hat{b}_{\varepsilon}^{(g)}\left(y|\theta\right)$, is less than one we have that the SAMCABC kernel has reduced bias compared to the original ABC kernel. The proof of Proposition 2.4.5 can be found in Appendix A.

2.5 Benchmark examples

We present the use of the proposed SAMC-ABC algorithm on synthetic examples and show that the proposed algorithm can both eliminate the local trapping problem and reduce the ABC bias.

2.5.1 Marginal Likelihood Estimation

We consider an exponential model with a conjugate prior distribution. We demonstrate marginal likelihood estimation using the SAMCABC method and compare it to the known marginal likelihood of the conjugate model. We generate 100 random variates from the distribution

$$Y_i \sim \mathrm{Exp}\left(\lambda\right)$$

with $\lambda = 0.1$ and use these as data. We calculate the sufficient statistic $T\left(Y\right) = \sum_{i=1}^{100} Y_i = 1038.35$ and note that

$$T\left(Y\right) \sim \mathrm{Gamma}\left(100, \lambda\right) \tag{2.6}$$

We set a gamma prior

$$\lambda \sim \mathrm{Gamma}\left(\alpha, \beta\right)$$

with hyperparameters $\alpha = 1$ and $\beta = 2$.

Given pseudodata, X, we define a uniform ABC kernel on the discrepancy $u = \left|T\left(Y\right) - T\left(X\right)\right|$ with support on $u \in [0, 80]$. We partition the sample space into 5 equally spaced subregions and set linearly decreasing desired sampling frequencies $\varpi = \left\{\frac{1}{3}, \frac{4}{15}, \frac{1}{5}, \frac{2}{15}, \frac{1}{15}\right\}$. The Metropolis-Hastings step generates random walk proposals, $\lambda_{t+1} \sim N\left(\lambda_t, 0.1^2\right)$, initialised with λ_0 generated from the prior. For the stochastic approximation step we set the stepsize equal to $\gamma_t = \frac{t_0}{\max(t_0,t)}$ with $t_0 = 10$. We run the sampler for 2×10^6 iterations and discard the first half as burnin.

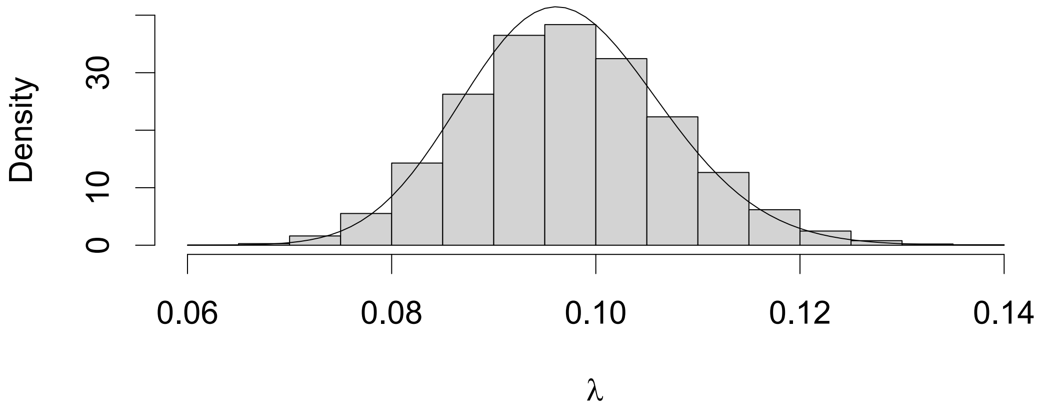


Figure 2.1: Histogram of a posterior sample of λ obtained using SAMCABC. True posterior is overlaid as a straight line.

We observe in Figure 2.1 the posterior sample approximates the true posterior well and is centered around the true parameter value $\lambda = 0.1$. Furthermore, we observed no local trapping. From the SAMCABC weights we find

$$\hat{w} = (1.56, 2.42, 4.33, 9.78, 39.6)$$

which we normalise to

$$w = (0.0007, 0.0011, 0.0019, 0.0043, 0.0174)$$

using the marginal likelihood estimator (2.4). Then we can estimate the ABC marginal likelihood as

$$\pi_{\varepsilon}^{(g)}(Y) = \sum_{i=1}^{5} w_i = 0.0254$$

We note this should not be compared to the true marginal likelihood $\pi(Y) = 3.4 \times 10^{-147}$ since the summary statistic was used. The summary statistic $T(Y)$ is sufficient for λ but is not sufficient for model comparison. However, as discussed in

Section 1.4 ABC model selection can still select the correct model if the expectation of the summary statistic is asymptotically different under the different models. Furthermore, since we know the $T(Y)$ is Gamma distributed (2.6) we can compare the SAMCABC estimate with the appropriate marginal probability:

$$\pi\left(T\left(Y\right)-\varepsilon<T\left(X\right)\leq T\left(Y\right)+\varepsilon\right)=0.0245$$

which is very close to the SAMCABC estimate. Moreover, we can find the ABC marginal likelihood estimate for the adaptive SAMCABC kernel

$$\pi_{\varepsilon}^{(h)}\left(y\right)=\left(\frac{1}{5}\sum_{j=1}^{5}\frac{\varpi_j}{w_j}\right)^{-1}=0.0057$$

which is very close to the appropriate marginal mixture probability

$$\sum_{j=1}^{5}\frac{\varpi_j}{w_j}\pi\left(\varepsilon_{j-1}<\left|T\left(Y\right)-T\left(X\right)\right|\leq\varepsilon_j\right)=0.0051$$

The massive difference between $\pi(y)$ and the ABC marginal likelihoods emphasises why care needs to be taken when performing model comparison with ABC. Nevertheless, ABC model selection can be done using these marginal likelihoods when the summary statistics have asymptotically different expectations under the different models. Moreover, as ε decreases both ABC marginal likelihoods converge towards the marginal likelihood of the summary statistic (2.7).

$$\pi\left(T\left(y\right)\right)=\frac{\Gamma\left(\alpha+n\right)}{\Gamma\left(\alpha\right)\Gamma\left(n\right)}\frac{\beta^{\alpha}T\left(Y\right)^{n-1}}{\left(\beta+T\left(Y\right)\right)^{\alpha+n}}=0.0002 \tag{2.7}$$

We observe that as ε decreases the ABC marginal likelihood decreases. Similarly, the SAMCABC kernel has lower variance than the ABC kernel and a lower marginal likelihood. This happens because the ABC approximation error decreases and higher discrepancies are rejected more often. To ensure fair model comparison the same

ABC kernel should be used for all models.

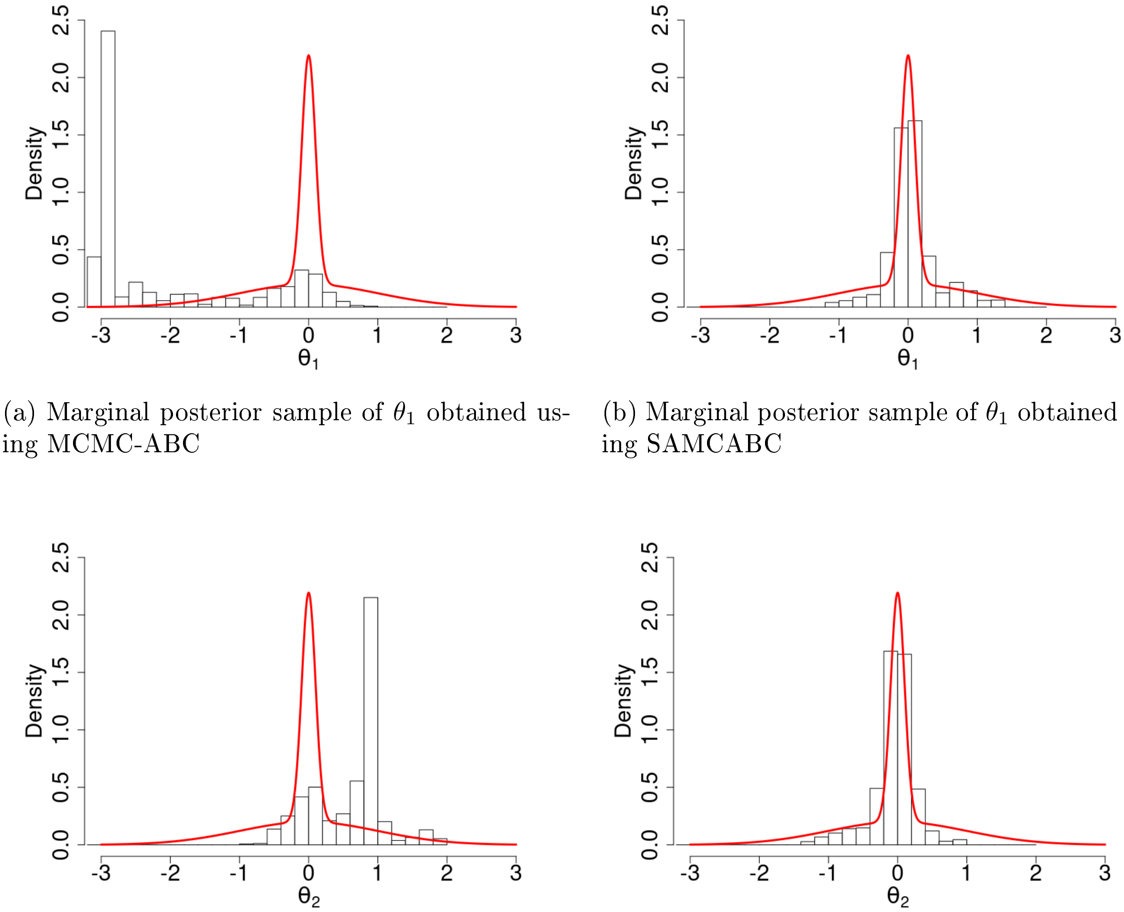
2.5.2 Normal mixture model

We consider a bivariate Normal mixture model based on the model from Sisson, Fan and Tanaka (2007). We demonstrate the local trapping problem of ABC which becomes more severe as the dimension of the parameter space grows. The model which is used generates data from the following sampling distribution:

$$f\left(x|\theta\right)=\frac{1}{2}N\left(x|\theta_1,\Sigma_1\right)+\frac{1}{2}N\left(x|\theta_2,\Sigma_2\right)$$

where $N\left(x|\mu,\Sigma\right)$ is the probability density function of a $N\left(\mu,\Sigma\right)$ distributed random variable and $\Sigma_1=I$ and $\Sigma_2=0.1^2I$ are fixed and known. The long flat tails of the resulting posterior are areas which an MCMC-ABC sampler can struggle to escape; resulting in severe local trapping as shown in Figure 2.2.

Similarly to the univariate case of Sisson, Fan and Tanaka (2007) broad uniform $U\left(-10,10\right)$ priors were used for the unknown means which both have a true value of zero. The MCMC-ABC sampler generates independent proposals for θ_1 and θ_2 from $\theta_i^{(t+1)}\sim N\left(\theta_i^{(t)},0.15^2\right)$ random walks initialized at $\theta^{(0)}=(0,0)$. The data set consists of a single observation at $y=(0,0)$ and the sampler uses a uniform ABC kernel on $u=||x-y||$ with support on $u\in[0,0.3]$. We ran the sampler for 2×10^5 iterations of which the first half were discarded as burn in. Our SAMCABC algorithm divides the sample space into 10 equally spaced subregions which partition the interval $u\in[0,0.3]$. Further for the cooling sequence required for the stochastic approximation step we take $\gamma_t=\frac{t_0}{\max\left(t_0,t^b\right)}$ with $t_0=100$ and $b=0.7$.



(a) Marginal posterior sample of θ_1 obtained using MCMC-ABC

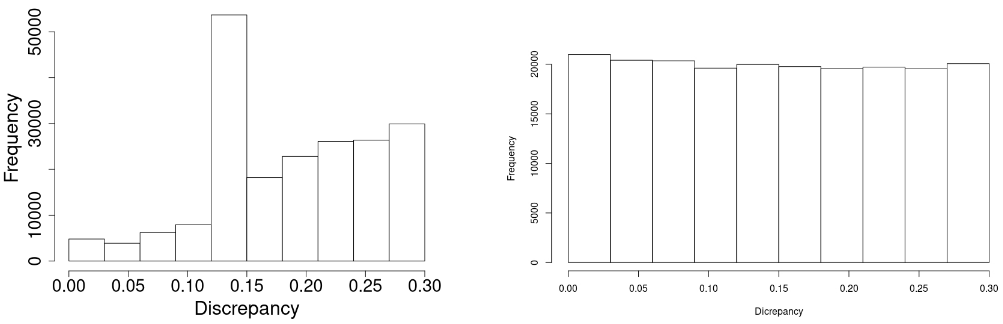
(b) Marginal posterior sample of θ_1 obtained using SAMCABC

(c) Marginal posterior sample of θ_2 obtained using MCMC-ABC

(d) Marginal posterior sample of θ_2 obtained using SAMCABC

Figure 2.2: In (a) and (b) we present the marginal posterior samples of θ_1 obtained by MCMC-ABC and SAMCABC respectively. Figures (c) and (d) show the corresponding marginal posterior samples for θ_2. The true marginal posteriors are overlaid in red and it is clear that MCMC-ABC suffers from local trapping whilst the proposed SAMCABC algorithm avoids this.

In Figure 2.2 we observe clear local trapping in the tails of the MCMC-ABC posterior whilst the SAMCABC algorithm successfully prevents this local trapping. We can also see that the SAMCABC algorithm successfully pushed the sampler towards low discrepancies in Figure 2.3.



(a) Histogram showing the number of visits to each subregion by the MCMC-ABC sampler

(b) Histogram showing the number of visits to each subregion by the SAMCABC sampler

Figure 2.3: Here we present the sampled discrepancies between the simulated pseudodata and the observed data. The histogram for MCMC-ABC in (a) shows indications of local trapping around 0.15 and overall samples mostly from high discrepancy regions. The SAMC-ABC sample shown in (b) shows the flattened discrepancy distribution that the algorithm targets to prevent local trapping and sample more often from low discrepancy regions.

In Figure 2.3 we observe clear indications of local trapping for the MCMC-ABC sampler somewhere in the region $(0.12, 0.15)$. Furthermore the histograms show that the SAMCABC sampler visited regions of low discrepancy with much higher frequency than the MCMC-ABC sampler, reducing the bias of the ABC posterior.

2.5.3 g-and-k distribution benchmark

We consider a synthetic data set simulated from a g-and-k quantile distribution. ABC methods are appropriate for this problem because quantile distributions have no explicit likelihood function available but can be sampled from using the cumulative distribution function (Allingham, King and Mengersen 2009; Drovandi and Pettitt 2011). This makes ABC methods particularly well suited to performing inference on these problems. The g-and-k distribution can be parameterised by the quantile function

$$Q_{gk}\left(p; A, B, g, k\right) = A + B\left(1 + c\frac{1 - \exp\left(-gz\left(p\right)\right)}{1 + \exp\left(-gz\left(p\right)\right)}\right)\left(1 + z\left(p\right)^2\right)^k z\left(p\right)$$

where the parameters A, B, g, and k describe the first four moments, $z(p)$ is the pth quantile of the standard normal distribution. Finally c is a further parameter for which the standard practice is to set to $c = 0.8$.

For the purposes of comparing MCMC-ABC to the proposed algorithms we generate synthetic data from a g-and-k distribution with a very long and heavy right tail. In particular we set $A = 3$, $B = 1$, $g = 2$, and $k = 2$ and simulate 10^4 data points from the distribution with these parameters.

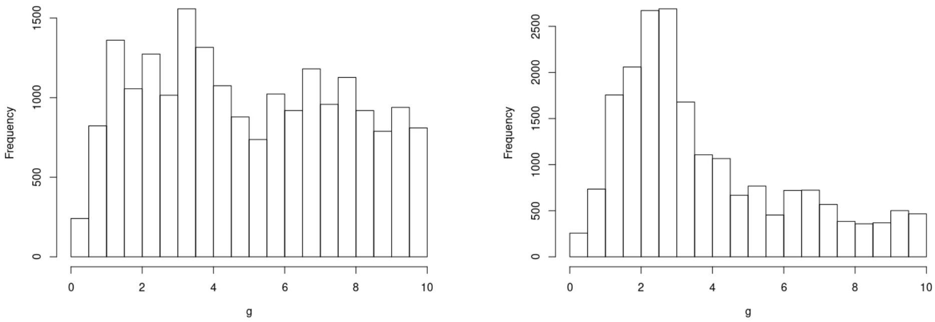
We then attempt to fit the g-and-k model to these data by using the ABC methods and the four robust summary statistics suggested in Drovandi and Pettitt (2011):

$$\begin{aligned} S_a =& L_2; & S_g =& \left(L_3 + L_1 - 2L_2\right)/S_b; \\ S_b =& L_3 - L_1; & S_k =& \left(E_7 - E_5 + E_3 - E_1\right)/S_b \end{aligned}$$

where L_i is the ith quartile and E_j is the jth octile. These four summary statistics are estimators of the first four moments of the data respectively. Then for a vector of the summary statistics, $S(x)$, we set $u = ||S(x) - S(y)||$. Finally, following Allingham, King and Mengersen (2009) we set wide independent uniform priors on the interval $[0, 10]$ for each of the four parameters.

We first apply MCMC-ABC to the synthetic data using a uniform ABC kernel with maximum tolerance $\varepsilon = 0.3$. This was run for $5 \times 10^3 + 2 \times 10^4$ iterations of which the first 5×10^3 were discarded as burn in. The SAMCABC sampler then partitions the sample space on u into 30 evenly spaced subregions. Similarly to the normal mixture model we take $a_t = \frac{t_0}{\max\left(t_0, t^b\right)}$ for the cooling sequence required for the stochastic approximation step with $t_0 = 100$ and $b = 0.7$.

For the parameters A,B, and k both of the algorithms perform similarly with the SAMC-ABC sample being slightly more concentrated around the mean than the MCMC-ABC samples. This reflects the slight reduction in ABC bias achieved by the adaptive kernel of the proposed algorithm. However for the third parameter of the model, g, we observe in Figure 2.4 that the MCMC-ABC sampler suffers from local trapping whilst the SAMCABC sampler largely avoids this.



(a) The MCMC-ABC marginal estimates of g

(b) The SAMCABC marginal estimates of g

Figure 2.4: Here we present the marginal posterior samples of the parameter g obtained using MCMC-ABC and SAMCABC. The MCMC-ABC algorithm displays trapping in the long heavy tail and massively oversamples this region of the parameter space. The proposed SAMCABC algorithm avoids this and finds the posterior mode near the true value of the parameter at $g = 2$.

For g the MCMC-ABC sampler becomes trapped in the long right tail of the distribution. The MCMC-ABC sampler also fails to identify the true value of the parameter which is $g = 2$ whereas the SAMCABC algorithm produces a large peak around $g = 2$.

We also demonstrate population SAMCABC on the g-and-k model. For population SAMCABC we retain the same partition and desired sampling frequency that was used for the single chain SAMCABC sampler. The population method is run with 10 parallel sampling chains and only for $5 \times 10^2 + 2 \times 10^3$ iterations of which 5×10^2 were discarded as burn in so that the sample size is the same as the single chain runs.

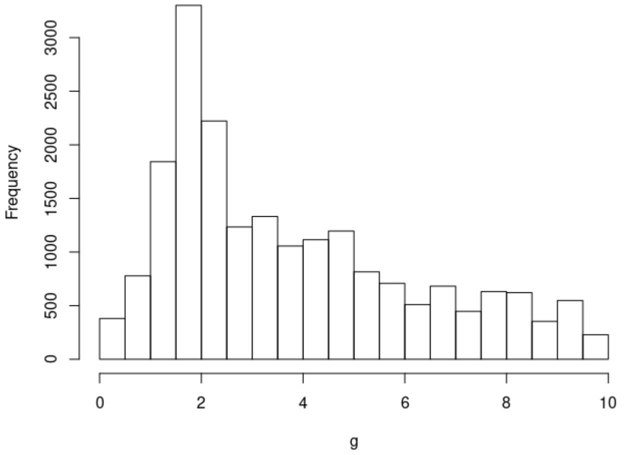


Figure 2.5: We present the marginal posterior samples of the parameter g obtained using the population SAMCABC method. Using fewer iterations with several chains in parallel we find that the methods still avoid oversampling the tail and hence identify the mode near the true value of the parameter at $g = 2$.

In Figure 2.5 we observe that the results are very similar to the results of the single chain SAMCABC sampler. Hence in this example by using parallel computing architecture we can speed up the convergence of the algorithm and use less CPU time without compromising the quality of the sample.

2.6 A data analysis example using ebola data

After the Ebola outbreak of 2014-15 there were several attempts to model the dynamics of Ebola transmission. We consider the model of Ponce et al. (2019) which aims to estimate the transmission rate of Ebola using data from the outbreak of Ebola in Liberia. These data were obtained from the Centre for Disease Control's (CDC) website (CDC 2014) where they were collated from World Health Organization (WHO) reports (WHO 2014). These data describe the number of cases and deaths recorded on 25 different occasions between the 5th June and the 14th September 2014. These dates were chosen because the epidemic curve had not yet been altered by the interventions which occurred after the 14th September 2014 (Ebola Response Team 2014). Hence these data can be used to estimate the pre-intervention transmission rate of Ebola.

The model of Ponce et al. (2019) is a compartmental model that divides the population into epidemiological classes which describe the disease progression. Transitions between these classes occur at exponential rates and many of these rates have been investigated elsewhere already. We focus primarily on investigating the three infection rates(β_I,β_H,β_D) which describe how quickly infected, hospitalized and dead Ebola victims infect new people respectively. Additionally we attempt to estimate the proportion of asymptomatic Ebola victims δ.

To perform the uncertainty quantification Ponce et al. (2019) use a deterministic specification of the model which is defined by a series of differential equations. They assume the cases and deaths output of the model are Poisson distributed and perform maximum likelihood estimation to determine the parameters. For δ they select the value by trying only a few different values and using AIC to select the preferred model.

We demonstrate the advantages of using ABC for the uncertainty quantification of such a complex model. Unlike Ponce et al. (2019) we can use the stochastic model which can be simulated from using the Gillespie algorithm (Gillespie 1977) to obtain a closer approximation to the exact data generating process assumed by the model. We use a slightly modified Gillespie algorithm which uses tau-leaps (Gillespie 2001; Cao, Gillespie and Petzold 2006) to greatly speed up the simulation at the cost of some small approximation error which can be controlled by the size of the tau-leaps. We then place priors on the parameters as follows, $\beta_I \sim \text{ U}(0, 0.4)$,$\beta_H/\beta_I \sim \text{ U}(0, 0.7)$,$\beta_D/\beta_I \sim \text{ U}(0.8, 1.5)$,$\delta \sim \text{ U}(0, 0.5)$, where each prior includes the estimated value from Ponce et al. (2019) and uses the infection rate in the general population β_I as a baseline. For the summary statistics we simply used the eighth, sixteenth and twenty-fourth data points from each of the time series. These had to be scaled and centered as they differed greatly in magnitude. This was done by simulating 10^6 times from the prior and estimating the mean and standard deviation of each statistic.

For the ABC samplers we use a uniform kernel with maximum tolerance $\varepsilon = 1$. We considered Metropolis updates with independent normal proposals with standard deviations of 0.03 for β_I and δ and standard deviations of 0.1 for β_H/β_I and β_D/β_I. The MCMC-ABC and SAMCABC algorithms were run for 4×10^4 iterations of which the first half were discarded as burn in.

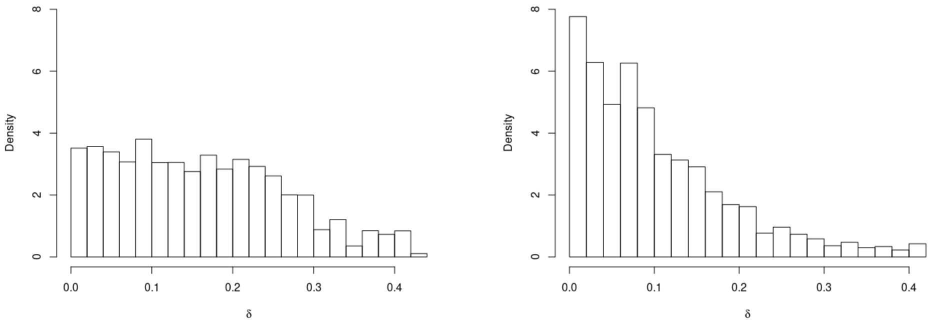
The SAMC-ABC sampler partitions the sample space on u into 10 evenly spaced subregions. Similarly to the normal mixture model we take the cooling sequence required for the stochastic approximation step with $t_0 = 100$ and $b = 0.7$.

The mean estimates for each of the parameters are listed in Table 2.1.

	MCMC-ABC	SAMCABC	Ponce et al.
β_I	$0.338(0.255, 0.397)$	$0.323(0.248, 0.398)$	$0.319(0.309, 0.330)$
β_H	$0.122(0.009, 0.227)$	$0.109(0.009, 0.213)$	$0.191(0.185, 0.198)$
β_D	$0.394(0.257, 0.542)$	$0.369(0.254, 0.525)$	$0.383(0.371, 0.396)$
δ	$0.160(0.006, 0.394)$	$0.103(0.003, 0.347)$	0.3

Table 2.1: Mean estimates from the MCMC-ABC and SAMCABC algorithms and their 95% credible intervals. The estimates of Ponce et al. (2019) are provided along with their 95% confidence intervals for comparison. There is no confidence interval for δ because Ponce et al. (2019) determined this parameter by trying four different values and choosing the one with the lowest AIC for the model.

The main differences are for β_H where the ABC estimates are smaller than those of Ponce et al. (2019) and for δ where the SAMCABC estimate is also much smaller. Note that Ponce et al. (2019) only compared $\delta = 0$, $\delta = 0.15$, $\delta = 0.3$ and $\delta = 0.5$.



(a) The MCMC-ABC marginal estimates of δ

(b) The SAMCABC marginal estimates of δ

Figure 2.6: Marginal posterior samples for δ, the proportion of cases which were asymptomatic. Results obtained using MCMC-ABC sample heavily from the tail and do not present a clear mode. Conversely, results obtained using the proposed SAMC-ABC algorithm show a posterior marginal with high density near $\delta = 0$ and decreasing density as δ grows.

In Table 2.1 we observe that the biggest difference between the MCMC-ABC and SAMCABC algorithms is found in the estimates of δ. We observe in Figure 2.6 that this is because of local trapping in the MCMC-ABC sample. It is clear that the MCMC-ABC sampler gets trapped at higher values of δ far more often than SAMCABC.

Chapter 3

Continuous Contour ABC

3.1 Optimising the acceptance rate of MCMC-ABC

In MCMC-ABC many proposals may be made which are outright rejected by the ABC kernel. In this way the acceptance ratio can be dominated by the ABC kernel rather than the Metropolis Hastings proposals. As a result Adaptive MCMC methods as in Andrieu and Thoms (2008) may not be effective in achieving optimal Metropolis Hastings acceptance rates. In particular we mostly do not have control over the simulator which we use to make proposals in the pseudomarginal space of $u = ||x - y||$. Since we also then use the ABC kernel to place an upper bound on this pseudomarginal density we can produce many proposals which have an acceptance probability of zero.

One solution is to adapt the maximum tolerance of the ABC kernel to achieve the desired acceptance rate as is done in the adaptive tolerance algorithm of Vihola and Franks (2020). To generalise the algorithm introduced by Vihola and Franks (2020) consider the Metropolis-Hastings acceptance ratio in Algorithm 1.4 restated in (3.1)

$$R = \frac{\pi\left(\theta'\right) q\left(\theta^{(t)}|\theta'\right) g_{\varepsilon}\left(u'\right)}{\pi\left(\theta^{(t)}\right) q\left(\theta'|\theta^{(t)}\right) g_{\varepsilon}\left(u^{(t)}\right)} \tag{3.1}$$

The algorithm of Vihola and Franks (2020) adapts the kernel function $g_{\varepsilon}(\cdot)$ by adjusting ε. However, this is not the only way we can adjust the kernel function. In general we can adjust the variance of the kernel function to achieve an optimal acceptance rate. To see how we might do this consider the proposed discrepancy, u', and the previously accepted discrepancy, $u^{(t)}$. The proposed discrepancy has been generated from the simulator $f(u|\theta)$. In contrast the previously accepted discrepancy, $u^{(t)}$, was generated from the simulator weighted towards $u = 0$ by the kernel. Hence proposals where the new discrepancy is greater than the previous by a given margin are more likely than proposals where the new discrepancy is less than the old by the same margin. Consequently the acceptance rate is reduced by a factor of $\lambda = \frac{g_{\varepsilon}(u')}{g_{\varepsilon}\left(u^{(t)}\right)}$ more often than it is increased by the same factor.

Now as we increase the variance of the ABC kernel it begins to flatten and the factor $\lambda \to \lambda^*$ for all discrepancies. Here λ^* denotes the ratio closest to 1 for the kernel family. In particular $\lambda = \lambda^*$ for all discrepancies when the ABC kernel is closest to the uniform kernel. This suggests that we can achieve the optimal acceptance rate in a similar manner as Andrieu and Thoms (2008) where we adapt the kernel variance instead of the proposal density, and that the algorithm in Vihola and Franks (2020) is a special case of this adaptation where the kernel variance is adapted by adapting the maximum tolerance parameter. This algorithm is detailed in Algorithm 3.1 where s_t is some parameter of the kernel function that is directly proportional to the kernel variance σ_g^2.

Algorithm 3.1 Adaptive MCMC-ABC

```
Inputs: observed data $y$, initial kernel parameter $s_1$, number of iterations N, initial parameter value $\theta_1$, initial discrepancy $u_1$, desired acceptance rate $\alpha^*$
Loop over $t = 1, ..., \mathrm{N}$
    1. Sampling step
        (a) Sample $\theta' \sim q\left(\theta' | \theta_t\right)$
        (b) Sample $x' \sim f\left(x' | \theta'\right)$
        (c) Calculate $u' = \left|S\left(y\right) - S\left(x'\right)\right|$
        (d) Accept $\left(\theta_t, u_t\right) = \left(\theta', u'\right)$ with probability $\alpha_{t+1} = \min\left(R, 1\right)$ with:
                $$R = \frac{\pi\left(\theta'\right) q\left(\theta_t | \theta'\right) g_{s_t}\left(u'\right)}{\pi\left(\theta_t\right) q\left(\theta' | \theta_t\right) g_{s_t}\left(u_t\right)}$$
        (e) Otherwise $\left(\theta_{t+1}, u_{t+1}\right) = \left(\theta_t, u_t\right)$
    2. Stochastic approximation step
        (a) Set $\log\left(s_{t+1}\right) = \log\left(s_t\right) + \gamma_t\left(\alpha^* - \alpha_{t+1}\right)$
End Loop
Output: parameter values $\theta_1, ..., \theta_{\mathrm{N}}$, discrepancies $u_1, ..., u_N$, kernel parameters $s_1, ..., s_{\mathrm{N}}$
```

The algorithm uses a stochastic approximation updating step to adjust the kernel parameter s_t so that the desired acceptance rate α^* is achieved. The adaptive tolerance algorithm of Vihola and Franks (2020) is a special case of Algorithm 3.1 when the kernel family is chosen to be uniform and the parameter is chosen to be the maximum tolerance. In the following sections we will extend this adaptation to the SAMC weights of Algorithm 2.1 to couple the local trapping protections of the adaptive kernel with the sampling benefits of the optimised acceptance rate.

3.2 Automatic construction of the SAMCABC partition weights

The SAMC-ABC framework introduced in Chapter 2 provides other benefits that we might wish to combine with the adaptation here. In particular the SAMCABC algorithm mitigates the local trapping problem of ABC, decreases the ABC bias compared to the base ABC kernel and provides an estimate of the marginal likelihood. The SAMCABC algorithm works by weighting the samples by a ratio of the desired sampling distribution and a histogram estimate of the standard ABC marginal. This is done by using stochastic approximation to update the partition weights ϕ at each iteration as discussed in Chapter 2. By doing this we adaptively construct an ABC kernel which pushes the sampler to sample from a desired sampling distribution in the discrepancies. This enables the sampler to escape local traps and also to push itself into low discrepancy regions which are otherwise rarely visited.

Directly embedding Algorithm 3.1 into the SAMCABC framework results in an algorithm which does not converge since both algorithms attempt to adapt the kernel function with different objectives. Furthermore, the SAMCABC framework requires that the maximum tolerance ε remains fixed. Hence we should consider kernel families which can be adapted without changing ε. For kernel functions where this is not the case a fixed ε can be imposed by truncating the kernel density at ε.

In the SAMCABC framework we estimate importance sampling weights of the form $\frac{\varpi}{w}$. The desired sampling distribution ϖ is fixed and we simply update our estimates of the marginal density w. We cannot update the ABC kernel without destroying the SAMC estimate of this marginal since changes to the fixed ABC kernel will require new SAMC estimates. However we can update the desired sampling density $\varpi_s(u)$ using some parameter s which can determine the desired sampling probability parametrically without interacting with the SAMC updates of w. This parameter s is proportional to the SAMCABC kernel variance. In this way we can still update the ABC kernel to achieve an optimal Metropolis-Hastings acceptance rate. We do this by an additional stochastic approximation updating step for the desired sampling distribution and correct the weights accordingly. For this new stochastic approximation step we suggest a cooling sequence of $\eta_t = \frac{a_t}{\lambda}$. For the examples in this chapter we have used $\lambda = 50$. The proposed algorithm is detailed in Algorithm 3.2

Algorithm 3.2 Adaptive SAMCABC

```
Inputs: observed data $y$, initial kernel parameter $s_1$, number of iterations N, initial parameter value $\theta_1$, initial discrepancy $u_1$, desired acceptance rate $\alpha^*$, subregions $E_j$
Loop over $t = 1, ..., \mathrm{N}$
    1. Sampling Step
        (a) Sample $\theta' \sim q\left(\theta'|\theta_t\right)$
        (b) Sample $x' \sim f\left(x'|\theta'\right)$
        (c) Calculate $u' = \left|S\left(y\right) - S\left(x'\right)\right|$
        (d) Accept $\left(\theta_{t+1}, u_{t+1}\right) = \left(\theta', u'\right)$ with probability $\alpha_{t+1} = \min\left(R, 1\right)$ with:
            $$R = \frac{\pi\left(\theta'\right) q\left(\theta_t|\theta'\right) g_{s_t}\left(u'\right) \exp\left(-\phi_{t,j(u')}\right)}{\pi\left(\theta_t\right) q\left(\theta'|\theta_t\right) g_{s_t}\left(u_t\right) \exp\left(-\phi_{t,j(u_t)}\right)}$$
        (e) Otherwise $\left(\theta_{t+1}, u_{t+1}\right) = \left(\theta_t, u_t\right)$
    2. Updating Step I
        (a) Compute $\boldsymbol{p}_{t+1} = \boldsymbol{p}_{t+1}\left(\theta_{t+1}, u_{t+1}\right)$ where $\left[p_{t+1}\right]_j = \begin{cases} 1 & \text{, if } \left(\theta_{t+1}, u_{t+1}\right) \in E_j \\ 0 & \text{, if } \left(\theta_{t+1}, u_{t+1}\right) \notin E_j \end{cases}$
        (b) Compute $\phi'_{t+1} = \phi_t + \gamma_{t+1}\left(\boldsymbol{p}_{t+1} - \varpi_t\right)$
    3. Updating Step II
        (a) Compute $\log\left(s_{t+1}\right) = \log\left(s_t\right) + \eta_{t+1}\left(\alpha^* - \alpha_{t+1}\right)$
        (b) Evaluate $\varpi'_{t+1}\left(s_{t+1}\right)$ and normalise to $\varpi_{t+1} = \frac{\varpi'_{t+1}}{\sum \varpi'_{t+1}}$
        (c) Compute $\phi_{t+1,j} = \phi'_{t+1,j} + \log\left(\frac{\varpi_{t,j}}{\varpi_{t+1,j}}\right)$ for $j = 1, ..., m$
End Loop
Output: parameter values $\theta_1, ..., \theta_{\mathrm{N}}$, estimate $\phi_{\mathrm{N}}$, kernel parameters $s_1, ..., s_{\mathrm{N}}$
```

Algorithm 3.2 thus chooses the desired sampling distribution $\varpi_s\left(u\right)$ such that a desired acceptance rate is achieved. Simultaneously by correcting the SAMC weights after each iteration we retain the immunity to ABC local trapping provided by the SAMCABC framework. Note that it is important that the desired sampling probabilities are non-zero everywhere, otherwise step $3\left(c\right)$ of the algorithm fails and the correction cannot be made. We find that with sensible choices for the maximum tolerance and with slower updating steps for s relative to the SAMC weights the algorithm can avoid the small values of s that lead to zeros in $\varpi_s(u)$.

3.3 The Continuous Contour ABC algorithm

The adaptive SAMCABC algorithm presented in Section 3.2 combines the optimal acceptance rate of adaptive MCMC-ABC with the local trapping mitigations of SAMCABC. We can improve the convergence of this algorithm by introducing ideas from Continuous Contour Monte Carlo (CCMC) (Liang 2007). To incorporate these ideas we introduce a new algorithm Continuous Contour ABC (CCABC) which replaces the histogram estimate of the marginal density used in SAMC with a kernel density estimate. This enables smoother adaptive kernel updates and hence more efficient convergence.

Let $\xi(u)$ be the marginal ABC density of $u = ||S(x)-S(y)||$ and let $\{z_i : i = 1, ..., W\}$ be grid points forming a rectangle U in the space of u. Let $\xi(z_i)$ be the true marginal density value at z_i. We consider the desired sampling distribution, $\varpi(u)$, similar to the desired sampling frequencies in SAMCABC which can be specified as any density on $u \in U$ and thus determines how much the sampler will "push" towards $u = 0$. For any ABC discrepancy u, we write $\hat{\xi}(u) = \frac{\varpi(u)}{\xi(u)}$ and evaluate $\hat{\xi}(u)$ by linear interpolation of the nearest grid points.

The CCABC algorithm consists of a sampling step and a stochastic approximation updating step. The sampling step draws M samples from a distribution that admits a density such that

$$\pi^{(t)}_{\text{CCABC}}(\theta, x|y) \propto \pi_{\text{ABC}}(\theta, x|y)\ \hat{\xi}^{(t)}(u)$$

via MCMC-ABC. Then the updating step updates $\hat{\xi}(u)$ at each of the grid points $\{z_i : i = 1, ..., W\}$ such that visits near the same lattice points as the new sample are discouraged and the chain becomes more likely to sample near lattice points that are further away. We follow Liang (2007) and suggest that the following sequences be used to update the marginal density and bandwidth of the kernel density estimation respectively:

$$a_t = \frac{t_0}{\max\left(t_0, t\right)}, \qquad h_t = \min\left(a_t^{\gamma}, \frac{\mathrm{range}\left(u\right)}{2\left(1 + \log_2\left(M\right)\right)}\right) \tag{3.2}$$

Similarly to SAMCABC the sampler constructs weights using stochastic approximation such that as $t \to \infty$ the working marginal density converges almost surely to the ratio of the true marginal density and the desired marginal density at the grid points up to some arbitrary multiplicative constant. For the stochastic approximation step, we have two cooling sequences. In addition to the cooling sequence γ_t used in SAMC-ABC let h_t be a positive, non-increasing sequence such that $h_t \to 0$ when $t \to \infty$ be the sequence of bandwidths used for kernel density estimation at iteration t.

Algorithm 3.3 CCABC

```
Inputs: observed data $y$, kernel parameter $\varepsilon$, number of iterations N, initial parameter value $\theta_1$, initial discrepancy $u_1$
Loop over $t = 1, ..., \mathbf{N}$, gridpoints $z_j$
    1. Sampling step
        (a) Set $\theta_t^{(0)} = \theta_{t-1}^{(M)}$
    Loop over $r = 1, ..., M$
        (b) Generate $\theta'_r$ from the proposal distribution $q\left(\theta|\theta_t^{(r-1)}\right)$
        (c) Generate $x'_r$ from the distribution $p\left(x|\theta'_r\right)$
        (a) Calculate $u'_r = |S(y) - S(x'_r)|$
        (b) Accept $\left(\theta_t^{(r)}, u_t^{(r)}\right) = (\theta'_r, u'_r)$ with probability $\alpha = \min(R, 1)$ where R is given by:
            $$R = \frac{\pi(\theta'_r)\, q\left(\theta_t^{(r-1)}|\theta'_r\right) g_\varepsilon(u'_r)\, \hat{\xi}\left(u_t^{(r-1)}\right)}{\pi\left(\theta_t^{(r-1)}\right) q\left(\theta'_r|\theta_t^{(r-1)}\right) g_\varepsilon\left(u_t^{(r-1)}\right) \hat{\xi}(u'_r)}$$
        (c) Otherwise $\left(\theta_t^{(r)}, u_t^{(r)}\right) = \left(\theta_t^{(r-1)}, u_t^{(r-1)}\right)$
    End Loop
    2. Updating Step
        (a) Estimate the density of the samples $u_t^{(1)}, ..., u_t^{(M)}$ using a kernel density estimation method and evaluate the density at the grid points
            $$\zeta_t(z_i) = \frac{1}{M} \sum_{r=1}^{M} h_t^{-\frac{1}{2}} K\left(h_t^{-\frac{1}{2}}\left(z_i - u_t^{(r)}\right)\right)$$
            where $K(v)$ is a kernel density function
        (b) Normalize $\zeta_t(z_i)$ on the grid points by setting
            $$\zeta'_t(z_i) = \frac{\zeta_t(z_i)}{\sum_{i'=1}^{W} \zeta_t(z_{i'})}$$
        (c) Update $\hat{\xi}_t(z_i)$ such that
            $$\log \hat{\xi}_{t+1}(z_i) = \log \hat{\xi}_t(z_i) + \gamma_t\left(\zeta'_t(z_i) - \pi'(z_i)\right)$$
            where $\pi'(z_i) = \frac{\pi(z_i)}{\sum_{i'=1}^{W} \pi(z_{i'})}$
End Loop
Output: parameter values $\theta_1, ..., \theta_{\mathrm{N}}$, discrepancies $u_1, ..., u_{\mathrm{N}}$,estimate $\hat{\xi}_{\mathrm{N}}$
```

As in our SAMCABC algorithm, the CCABC algorithm avoids local trapping by adapting the kernel function to encourage visits to poorly explored regions of the sample space. Furthermore, the CCABC sampler improves upon the SAMCABC sampler by constructing the weights in a continuous way. This typically results in CCABC converging more quickly than SAMCABC because SAMCABC updates the weights of fixed subregions by using the gain factor γ_t which determines how large the updates should be. CCABC controls the updates by using not only the gain factor but also the bandwidth sequence which allows early simulations to penalize larger areas of the marginal density whilst later simulations only penalize smaller localized regions.

We optimise the acceptance rate of CCABC by automatically adapting the desired sampling distribution $\varpi(u)$ similar to adaptive SAMCABC. This adaptation has much more flexibility than under SAMCABC since we can specify $\varpi(u)$ as a continuous distribution. To adapt $\varpi(u)$ we append the update step found in Algorithm 3.4 to the CCABC algorithm found in Algorithm 3.3.

Algorithm 3.4 Adaptive CCABC Step

```
3. Updating Step II
    (a) Compute $\log(s_{t+1}) = \log(s_t) + \eta_{t+1}(\alpha^* - \alpha_{t+1})$
    (b) Evaluate $\pi'_{t+1}(s_{t+1})$ and normalise to $\pi_{t+1} = \frac{\pi'_{t+1}}{\sum \pi'_{t+1}}$
    (c) Set $\hat{\xi}_{t+1} = \hat{\xi}_{t+1} \times \frac{\pi_t}{\pi_{t+1}}$
```

The adaptive CCABC algorithm retains the benefits of adaptive SAMCABC, achieving desired acceptance rates whilst mitigating local trapping and reducing the ABC bias. Furthermore, the CCABC algorithm boasts improved convergence compared to the SAMCABC algorithm because of the continuous nature of CCABC's adaptive kernel. We note that the desired sampling distribution must be non-zero everywhere in U otherwise the updating step in Algorithm 3.4 fails. We find this failure can easily be avoided with sensible choices for the maximum tolerance and a slower cooling sequence for s than $\hat{\xi}$.

3.4 Model selection and the adaptive kernel of CCABC

The CCABC algorithm found in Algorithm 3.3 constructs an adaptive kernel $h\left(u\right)$ similar to the adaptive kernel of SAMCABC. We can write this kernel as

$$h_{\varepsilon}\left(u\right) \propto \frac{\varpi\left(u\right)}{\hat{\xi}\left(u\right)} g_{\varepsilon}\left(u\right) \tag{3.3}$$

Here we demonstrate how the adaptive kernel of CCABC can be used to determine the marginal likelihood.

Proposition 3.4.1. *Consider the marginal density kernel density estimate $\xi\left(u\right)$ estimated by $\varpi\left(u\right)/\hat{\xi}\left(u\right)$ under the CCABC framework. Then it follows that the ABC marginal likelihood $\pi_{\varepsilon}^{(g)}\left(y\right)$ can be calculated as:*

$$\pi_{\varepsilon}^{(g)}\left(y\right) = \int \frac{\varpi\left(u\right)}{C\hat{\xi}\left(u\right)} du$$

Where $\varpi\left(u\right)$ is a chosen desired sampling density and $\hat{\xi}\left(u\right)$ has been estimated as a piecewise linear function up to some multiplicative constant C so this integral can be solved analytically under sensible choices of the desired sampling density. Even under alternate choices the integral is univariate and so cheap to solve numerically. We can use an alternative estimate of the marginal likelihood to determine C and use it to detemine the CCABC marginal likelihood $\pi_{\varepsilon}^{(h)}\left(y\right)$.

Proposition 3.4.2. *The CCABC marginal likelihood $\pi_{\varepsilon}^{(h)}\left(y\right)$ can be calculated as*

$$\pi_{\varepsilon}^{(h)}\left(y\right) = \left(\int \frac{\varpi\left(u\right)}{C\hat{\xi}\left(u\right)} g_{\varepsilon}\left(u\right)\, du\right)^{-1}$$

Unlike the marginal likelihood estimate of the SAMCABC algorithm, the marginal likelihood estimate from CCABC is generally not analytically tractable. However, for certain choices of ABC kernel such as the uniform ABC kernel and desired sampling density the integral becomes much easier to evaluate. Let us consider an example where we use the uniform ABC kernel which is a popular choice in the literature. We will also specify the desired sampling density $\varpi(u)$ to be uniform which makes evaluation more straightforward and we have found to also be a good choice in practice.

If $g_\varepsilon(u)$ is the uniform ABC kernel on $u \in [0, \varepsilon]$ and the desired sampling density $\varpi(u)$ is uniform then the ABC marginal likelihood $\pi_\varepsilon^{(g)}(y)$ can be estimated as

$$\pi_\varepsilon^{(g)}(y) = \frac{1}{m} \sum_{i=1}^{m} \frac{\log\left|\hat{\xi}(\varepsilon_i)\right| - \log\left|\hat{\xi}(\varepsilon_{i-1})\right|}{\hat{\xi}(\varepsilon_i) - \hat{\xi}(\varepsilon_{i-1})}$$

where the ε_i are the gridpoints. The CCABC marginal likelihood $\pi_\varepsilon^{(h)}(y)$ can be estimated as

$$\pi_\varepsilon^{(h)}(y) = \left(\frac{1}{m} \sum_{i=1}^{m} \frac{\hat{\xi}(\varepsilon_i) + \hat{\xi}(\varepsilon_{i-1})}{2}\right)^{-1}$$

Derivations of these marginal likelihood estimates can be found in Appendix A.

Moreover, we show that the CCABC algorithm achieves ABC bias reduction under similar conditions to SAMCABC. Since the CCABC kernel produces an adaptive kernel (3.3) which satisfies the ABC kernel properties we have that Proposition 2.4.4 holds for CCABC. We revisit the uniform kernel example of Proposition 2.4.5 to demonstrate how this works for CCABC in practice. Hence we assume that $g_\varepsilon(u)$ is the uniform kernel with maximum tolerance ε. Clearly it follows that the variance of this kernel density is $\sigma_g^2 = \frac{1}{3}$.

Proposition 3.4.3. *For CCABC, we have that*

$$\sigma_h^2 = \frac{1}{3m^3} \sum_{j=1}^{m} \left(j\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right) - (j-1)\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3 - (j-1)^3\right)$$

and so it follows that the bias reduction factor is:

$$\frac{\hat{b}_{\varepsilon}^{(h)}\left(y|\theta\right)}{\hat{b}_{\varepsilon}^{(g)}\left(y|\theta\right)} = \frac{1}{m^3}\sum_{j=1}^{m}\left(j\hat{\xi}\left(\frac{\left(j-1\right)\varepsilon}{m}\right) - \left(j-1\right)\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3 - \left(j-1\right)^3\right)$$

Thus when this factor is less than one we know that the CCABC posterior has reduced ABC bias.

3.5 Automatic handling of model misspecification

Both of the algorithms introduced here depend on the absence of ABC model misspecification to converge. As described in Frazier, Robert and Rousseau (2020) we define $\varepsilon^* = \min_{\theta}\left(u\right)$ to be the minimum distance between the observed data and the simulated data under the model. If the model does not suffer from ABC misspecification then $\varepsilon^* = 0$ otherwise ε^* takes some positive value proportional to the severity of the ABC misspecification. Under model misspecification the acceptance rate may not change as expected when the kernel variance becomes small and lots of weight is assigned to the region $[0, \varepsilon^*]$ which is empty. Furthermore, under model misspecification the ABC posterior does not produce credible sets with valid coverage properties however Frazier, Robert and Rousseau (2020) showed that the posterior does concentrate mass on appropriate pseudo true parameter values.

If ε^* is non zero then we want to estimate ε^* and use a kernel which places as little weight on the $[0, \varepsilon^*]$ interval as possible. Here we propose a simple adaptation to the algorithms which estimates $\hat{\varepsilon}^*$ by the smallest discrepancy sampled so far in the chain. Then as $t \to \infty$ we have that $\hat{\varepsilon}^* \to \varepsilon^*$ provided the kernel family assigns enough weight to the $[0, \varepsilon^*]$ interval to still encourage the sampler to explore. Assigning appropriate weight to this region is difficult when using the adaptive MCMC-ABC algorithm since this must be done through appropriate kernel family choice and the marginal density is poorly understood in the $[0, \varepsilon^*]$ interval under misspecification. If too much weight is assigned then the sampler will become stuck if it enters the region, however if too little weight is assigned the sampler may never explore this region and additional ABC bias may be introduced.

Contrastingly the adaptive CCABC algorithm offers a direct solution to the problem. Since with CCABC we do not specify the kernel directly but instead adapt the kernel to achieve a desired sampling distribution we can more directly affect the weight assigned to a region of the sampling space. In particular we can determine the desired sampling distribution by some piecewise function that is flat in the $[0, \varepsilon^*]$ interval with value equal to the maximum value of the kernel in the $[\varepsilon^*, \varepsilon]$ interval. This results in a CCABC kernel which assigns the minimum weight to the $[0, \varepsilon^*]$ interval such that exploration of this interval is always encouraged and no more. This change can be made to Step 1 of Algorithm 3.3 by adding:

1 (d) If (θ', x') is accepted and $u' < \hat{\varepsilon}^*$ then set $\hat{\varepsilon}^* = u'$

and by using a kernel family that adjusts for changes in $\hat{\varepsilon}^*$.

In this chapter we use mixture that is uniform in $[0, \varepsilon^*)$ and half-normal in $[\varepsilon^*, \infty)$

$$\varpi(u) = \begin{cases} \frac{1}{\sqrt{2\pi s^2}} & u \leq \varepsilon^* \\ \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{(u-\varepsilon^*)^2}{2s^2}\right) & \varepsilon^* < u \leq \varepsilon \end{cases} \tag{3.4}$$

3.6 Benchmark examples

In this section we demonstrate the use of adaptive CCABC through some synthetic examples. Our numerical results demonstrate that the desired acceptance rate can be achieved by adapting the desired sampling distribution and that we can detect model misspecification by estimating ε^*. Since the ABC algorithm is a pseudomarginal algorithm we target an acceptance rate of 7% throughout both examples and the application following the results of Sherlock et al. (2015).

3.6.1 A g-and-k Numerical Example with Adaptive CCABC

We continue with the example detailed in Section 2.5.3 using the same sample size and parameter values. We then attempt to fit the g-and-k model to these data using the adaptive tolerance algorithm of Vihola and Franks (2020) and the adaptive CCABC method described in this paper. We use the four robust summary statistics suggested in Drovandi and Pettitt (2011):

$$\begin{aligned} S_a =& L_2; & S_g =& \left(L_3 + L_1 - 2L_2\right)/S_b; \\ S_b =& L_3 - L_1; & S_k =& \left(E_7 - E_5 + E_3 - E_1\right)/S_b \end{aligned}$$

where L_i is the ith quartile and E_j is the jth octile.

For both algorithms we use independent normal random walk proposals with standard deviations of 0.1 for A,B, and k and a standard deviation of 1 for g. For the adaptive CCABC algorithm we set the base ABC kernel to be uniform with a large maximum tolerance $\varepsilon = 5$. This should bound the acceptance rate above the desired value which is 0.07 for both algorithms. Finally we have 501 grid points z_i spaced evenly from $z_1 = 0$ to $z_{501} = 5$ with desired sampling density given by:

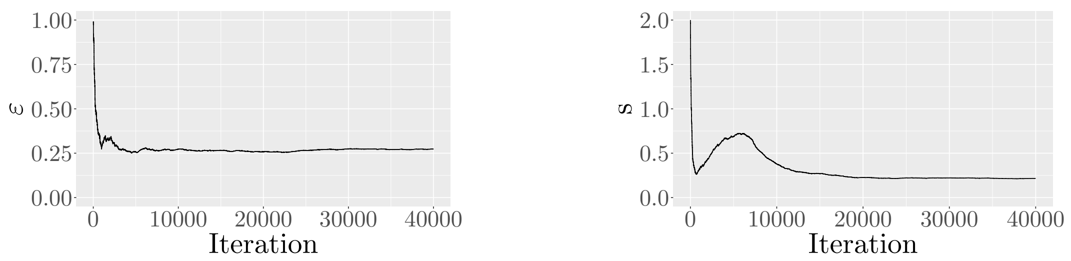
$$\pi'_s\left(z_i\right) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{z_i^2}{2s^2}\right)$$

and normalised to

$$\pi_s\left(z_i\right) = \frac{\pi'_s\left(z_i\right)}{\sum_{i=1}^{501} \pi'_s\left(z_i\right)}$$

For the cooling sequences of the CCABC algorithm we use the cooling sequences described in (3.2) with $t_0 = 1000$, $M = 1$ and $\gamma = 1$.

We run both algorithms for a burn in period of 2×10^4 iterations and then for a further 2×10^4 iterations and both algorithms converge quickly as shown in Figure 3.1b. The adaptive CCABC algorithm shown on the right converges more slowly because the CCMC weights must also converge.



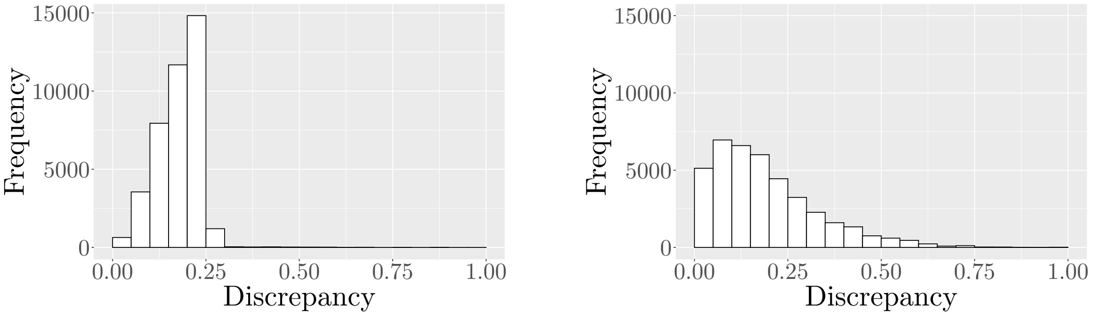
(a) The tolerance parameter of the Adaptive Tolerance MCMC-ABC algorithm

(b) The desired sampling parameter of the Adaptive CCABC algorithm

Figure 3.1: The adaptive tolerance algorithm of Vihola and Franks (2020) shown on the left converges to a maximum tolerance of about 0.252. The adaptive CCABC algorithm shown on the right converges to a half normal desired sampling distribution with standard deviation of approximately 0.212.

The adaptive CCABC algorithm appears to converge in a strange manner. First it converges very quickly but then it deviates for a few thousand iterations before converging again. This occurs because the sampler manages to push into a low discrepancy region of the sample space which it had previously not visited. This results in big updates to the CCMC weights and while the sampler is trying to explore this region the acceptance rate drops while these are updated and the adaptive mechanism increases s to compensate. Once the weights have adjusted the desired sampling density converges again to an appropriate value with the new weights.

The two algorithms produce quite different posterior samples. The difference is particularly clear in the histograms of the discrepancies which are shown in Figure 3.2.



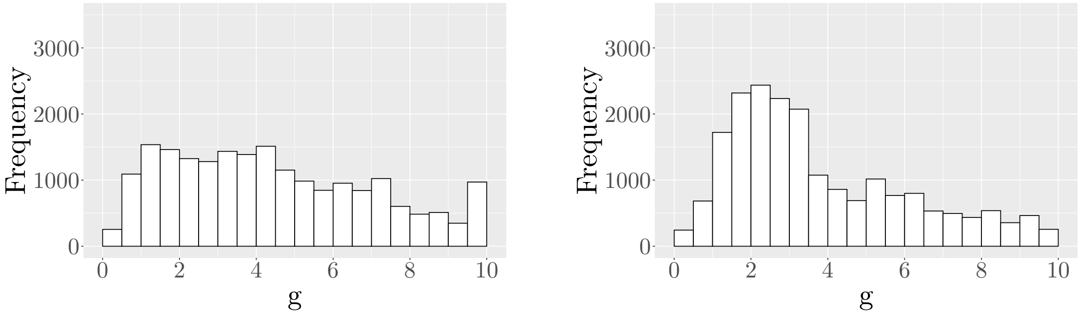
(a) The discrepancies sampled by the Adaptive Tolerance MCMC-ABC algorithm

(b) The discrepancies sampled by the Adaptive CCABC algorithm

Figure 3.2: The adaptive tolerance algorithm of Vihola and Franks (2020) shown on the left imposes a maximum tolerance of 0.252 and so there are not many visits to regions above this tolerance. The adaptive CCABC algorithm shown on the right converges to a half normal desired sampling distribution with standard deviation of approximately 0.212 which approximately matches the sampled discrepancy distribution as expected.

There are two immediate differences between the two discrepancy samples. The first clear difference is the maximum tolerance which is imposed by the adaptive tolerance algorithm. As a result the adaptive tolerance algorithm never samples higher discrepancy regions and instead traps itself in a region of low discrepancy. In contrast the adaptive CCABC algorithm does not impose this maximum tolerance and instead allows the sampler to visit higher discrepancy regions enabling the sampler to escape local traps. The other main difference between the two samples can be seen in the way the adaptive CCABC algorithm pushes the sampler towards lower discrepancies. This reduces the ABC bias of the posterior and so despite the adaptive CCABC sampler being allowed to visit higher discrepancies it does this much less often than it would under a uniform kernel that allowed this. Additionally this push means that despite spending fewer iterations in the acceptable region of the adaptive tolerance algorithm the adaptive CCABC sampler still visited the very lowest discrepancy regions much more often than the adaptive tolerance algorithm.

The advantages of the ACCABC algorithm are clear in histograms of all four marginal posterior samples where the additional ABC bias of the adaptive tolerance algorithm can be observed. However, these advantages are most clear in the histogram of the marginal estimates of g shown in Figure 3.3 where it can be observed that the adaptive tolerance algorithm became trapped in the long right tail.



(a) The posterior estimates of g obtained using the Adaptive Tolerance MCMC-ABC algorithm

(b) The posterior estimates of g obtained using the Adaptive CCABC algorithm

Figure 3.3: The adaptive tolerance algorithm struggles to identify the posterior mode and additionally some minor local trapping can be seen in the tails. By contrast the ACCABC algorithm identifies a strong mode around the true value of $g = 2$ and explores the long right tail without getting trapped.

There is a very clear difference between the two posterior estimates in Figure (3.3) where the adaptive tolerance algorithm fails to capture the mode of the posterior. Instead the adaptive tolerance sampler suffers from some local trapping in the long tail of the distribution. The adaptive CCABC algorithm on the other hand successfully reduces the ABC bias through its “pushing” mechanism whilst also displaying an immunity to the local trapping problem from which the other tolerance based algorithm suffers. These attributes enabled the sampler to identify the posterior mode near the true parameter value of $g = 2$. Additionally the sampler was still allowed to properly explore the tail of the posterior and did so without becoming trapped there.

3.6.2 An Example with Misspecification

We consider an example from Frazier, Robert and Rousseau (2020) where we simulate 100 data points from the normal mixture.

$$y_i \sim 0.9 \times N(1,2) + 0.1 \times N(7,2)$$

We now attempt to model this data using the g-and-k distribution. Whilst the g-and-k distribution is highly flexible it cannot capture the bi-modal feature of the true data generating process. While there are therefore no true parameters for the model Frazier, Robert and Rousseau (2020) show that for the Euclidean norm the "pseudo true" parameter values are $\theta^* = (1.17, 1.50, 0.41, 0.23)$. Frazier, Robert and Rousseau (2020) go on to show that several ABC algorithms, including their own regression adjusted method, perform poorly for this example. Here we show that our method can detect the misspecification and adjust for it to perform well.

For the model we take independent uniform priors on $[0, 10]$ for each of the parameters and for the Metropolis Hastings step we make independent Normal random walk proposals with standard deviations of 0.1 for A, B, and k and a standard deviation of 1 for g. Following Frazier, Robert and Rousseau (2020) we take the Euclidean distance between the first seven octiles of the data and pseudo data to be the discrepancy. For the adaptive CCABC algorithm we use a base ABC kernel with maximum tolerance $\varepsilon = 2$ and set a grid of 201 evenly spaced lattice points over the region $[0, 5]$. We set the desired sampling density as in (3.4) with $p = 0.99$. Finally for the cooling sequences we use the cooling sequences described in (3.2) with $t_0 = 400$, $M = 1$ and $\gamma = 1$. We run both algorithms for a burn in period of 2×10^4 iterations before running them for a further 2×10^4 iterations.

Following Frazier, Robert and Rousseau (2020) we run the experiment 1000 times and record the estimates of the posterior means and standard deviations, the 2.5% and 97.5% quantiles, the average length of the credible set and the Monte Carlo coverage of the pseudo true parameter values θ^*. In many of the experiments the adaptive tolerance algorithm of Vihola and Franks (2020) suffered from the local trapping problem and a typical example of this can be found in Figure 3.4.

As expected none of the adaptive CCABC samples displayed the local trapping problem.

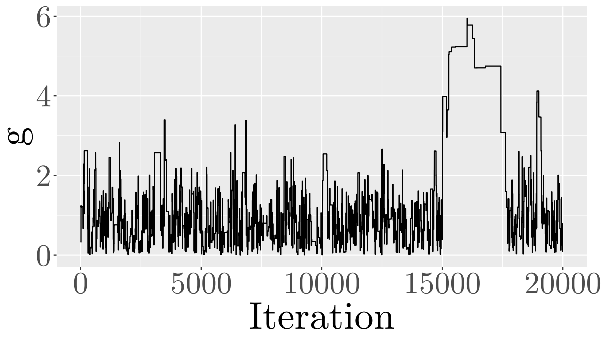


Figure 3.4: Typical trace plot for the samples of the g parameter using the adaptive tolerance algorithm

For reference we provide the values recorded by Frazier, Robert and Rousseau (2020) for their local linear regression ABC method. We have also included results using the adaptive SAMCABC algorithm detailed in Section 3.2. With some post processing we improve the results. This can be done with the three adaptive algorithms by applying a simple cut off to the discrepancies. As the adaptive algorithms naturally push the samples towards low discrepancies this cut off can be set much lower than is feasible for Accept-Reject based samples such as the regression method proposed by Frazier, Robert and Rousseau (2020). In Table 3.1 we provide the results of a cut off set at $\varepsilon = 0.8$ and provide the results of Frazier, Robert and Rousseau (2020) for reference.

As we see in Table 3.1 the two adaptive algorithms achieve good estimates of the parameters after some post processing. In addition to this the two algorithms achieve much tighter credible intervals after the cut off has been applied whilst achieving superior coverage properties. Moreover, the adaptive CCABC achieves much tighter credible intervals than the adaptive tolerance algorithm, most obviously displayed by the credible intervals for g. This is because the adaptive CCABC algorithm is able to push the sampler towards lower discrepancies much more effectively than

A	RegN	AT	SAMC	CC
Mean	1.1568	1.0590	1.1058	1.0915
Std Dev	0.0027	0.2586	0.2152	0.2104
CI Length	0.0112	1.0091	0.8485	0.8358
Coverage	0.0180	0.9670	0.9630	0.9510
2.5% Quantile	1.1509	0.5475	0.6688	0.6566
97.5% Quantile	1.1621	1.5566	1.5173	1.4942

(a) Estimates for A

B	RegN	AT	SAMC	CC
Mean	1.0143	1.3418	1.3871	1.3967
Std Dev	0.0104	0.3704	0.3023	0.3019
CI Length	0.0414	1.4125	1.1805	1.1887
Coverage	0.1730	0.9910	0.9770	0.9880
2.5% Quantile	0.9940	0.6732	0.8025	0.8027
97.5% Quantile	1.0354	2.0856	1.9830	1.9914

(b) Estimates for B

g	RegN	AT	SAMC	CC
Mean	5.3310	1.2202	0.9565	0.8190
Std Dev	1.5265	1.1763	0.8391	0.7731
CI Length	6.1010	4.3954	3.0760	2.8668
Coverage	0.3380	0.9630	0.9670	0.9720
2.5% Quantile	2.4728	0.0906	0.0954	0.0972
97.5% Quantile	8.5738	4.4859	3.2101	2.9640

(c) Estimates for g

k	RegN	AT	SAMC	CC
Mean	2.1790	0.5852	0.4945	0.4838
Std Dev	0.9426	0.4241	0.3145	0.3212
CI Length	3.8949	1.5531	1.1901	1.2229
Coverage	0.5760	0.9660	0.9510	0.9530
2.5% Quantile	0.3648	0.0421	0.0552	0.0537
97.5% Quantile	4.2597	1.5952	1.2453	1.2766

(d) Estimates for k

Table 3.1: Here we have the average recorded details for 1000 simulations using the Adaptive Tolerance(AT) algorithm and the adaptive SAMCABC(SAMC) and CCABC(CCMC) algorithms after applying a simple cut off at $\varepsilon = 0.8$. For reference we also have the local linear regression(RegN) results from Frazier, Robert and Rousseau (2020). Note the pseudo true values are $(A^*, B^*, g^*, k^*) = (1.17, 1.50, 0.41, 0.23)$

the adaptive tolerance algorithm.

3.7 Revisiting the ebola model

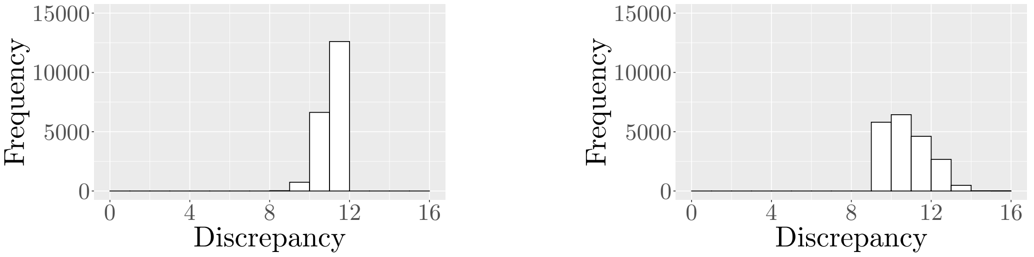
We apply the Adaptive CCABC algorithm to fit the compartmental model described in Section 2.6. For the set-up of the ACCABC we choose a large maximum tolerance of $\varepsilon = 16$ as we expect discrepancies in the high dimensional data space to be relatively large. We then set up the lattice of gridpoints as 161 evenly spaced points from 0 to 16. For the Metropolis Hastings step we make independent Normal random walk proposals with standard deviations of 0.05 for each of the parameters. We take the Euclidean distance between the data and pseudo data to be the discrepancy. We set the desired sampling density as in 3.4. Finally for the cooling sequences we use the cooling sequences described in (3.2) with $t_0 = 2000$, $M = 1$ and $\gamma = 1$. We run the algorithm for a burn in period of 2×10^4 iterations before it is run for a further 2×10^4 iterations.

The mean estimates of the four parameters are provided in Table 3.2 with 95% credible intervals. Additionally the estimates of Ponce et al. (2019) can also be found in Table (3.2) however the intervals for these are frequentist 95% confidence intervals. There is no confidence interval for the proportion of asymptomatic infections, δ, as Ponce et al. (2019) determined this parameter by trying four different values and choosing the one with the lowest AIC value for the model.

	AT	CC	Ponce et al. 2019
β_I	0.334 $(0.256, 0.397)$	0.343 $(0.255, 0.398)$	0.319 $(0.309, 0.330)$
β_H	0.142 $(0.036, 0.232)$	0.112 $(0.013, 0.233)$	0.191 $(0.185, 0.198)$
β_D	0.385 $(0.257, 0.537)$	0.412 $(0.286, 0.530)$	0.383 $(0.371, 0.396)$
δ	0.169 $(0.010, 0.395)$	0.163 $(0.010, 0.388)$	0.3

Table 3.2: Posterior estimates of the transmission parameters and their 95% credible intervals from both the adaptive tolerance algorithm and ACCABC. The estimates from Ponce et al. (2019) are also provided for reference along with their 95% confidence intervals. There is no confidence interval for the proportion of asymptomatic infections, δ, as Ponce et al. (2019) determined this parameter by trying four different values and choosing the one with the lowest AIC value for the model.

The three methods produce very similar estimates for the three transmission parameters. For the proportion of asymptomatic infections, δ, however the two ABC methods estimate a much lower proportion than the likelihood based estimates of Ponce et al. (2019). The model suffers from severe misspecification and the adaptive CCABC method estimated $\varepsilon^* = 8.46$. Indeed the histograms of discrepancies shown in Figure 3.5 from both the ABC methods show a large gap between the lowest discrepancies and zero.



(a) The discrepancies sampled by the adaptive tolerance algorithm

(b) The discrepancies sampled by the adaptive CCABC algorithm

Figure 3.5: Histograms of the discrepancies from the two algorithms. We see that the Adaptive Tolerance algorithm enforces a sharp cut off at $\varepsilon = 11.65$ while the adaptive CCABC algorithm allowed discrepancies larger than this to ensure immunity to local trapping. The adaptive CCABC algorithm also pushed more aggresively into lower discrepancy areas and so despite allowing larger discrepancies it also sampled more from the lowest discrepancy regions.

This suggests that results from this model should be treated cautiously as the method detected severe model misspecification. For comparison we also simulated 20000 pseudodata sets using the simulator with the estimates of Ponce et al. (2019) and recorded the discrepancies. These discrepancies are shown in Figure 3.6. These discrepancies are noticeably higher than those of the two ABC methods which may suggest that the likelihood method of Ponce et al. (2019) did not find estimates near the pseudo true values. In fact the mean discrepancy was 13.2 which is noticeably larger than the mean discrepancies of the adaptive tolerance and adaptive CCABC algorithms which were 11.1 and 10.8 respectively.

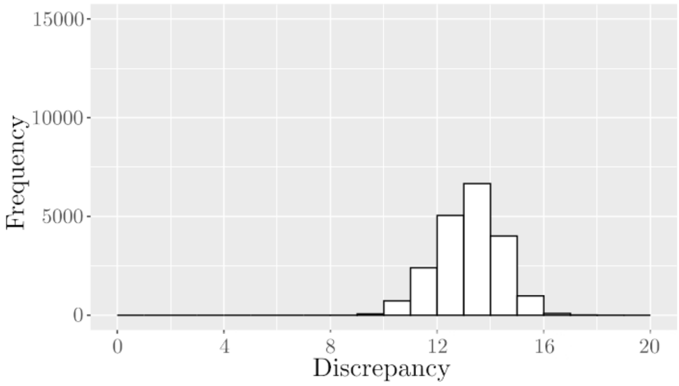


Figure 3.6: Histogram of the discrepancies sampled using the parameter estimates of Ponce et al. (2019). Note that these are almost all larger than the discrepancies sampled in the two ABC methods. Furthermore, notice the high variance of discrepancy despite the fixed input parameters.

Also noticeable in Figure 3.6 is the high variance in discrepancy. This is surprising since all of these simulations were completed using identical parameter settings. This perhaps suggests that for this model the data are not very informative. This would also explain why post processing of the ABC results by imposing a cut off at $\varepsilon = 10$ resulted in estimates that were largely unchanged.

	AT (Post-processed)	CC (Post-processed)	Ponce et al. 2019
β_I	0.337 $(0.257, 0.398)$	0.353 $(0.258, 0.399)$	0.319 $(0.309, 0.330)$
β_H	0.127 $(0.022, 0.219)$	0.103 $(0.017, 0.233)$	0.191 $(0.185, 0.198)$
β_D	0.404 $(0.256, 0.530)$	0.424 $(0.323, 0.510)$	0.383 $(0.371, 0.396)$
δ	0.155 $(0.013, 0.331)$	0.147 $(0.010, 0.379)$	0.3

Table 3.3: Posterior estimates of the transmission parameters and their 95% credible intervals from both the adaptive tolerance algorithm and the adaptive CCABC after imposing a cut-off at $\varepsilon = 10$. The estimates from Ponce et al. (2019) are also provided here for reference along with their 95% confidence intervals. There is no confidence interval for the proportion of asymptomatic infections, δ, because Ponce et al. (2019) determined this parameter by trying four different values and choosing the one with the lowest AIC value for the model.

The credible intervals observed in Table 3.3 became slightly smaller after post processing but the difference is very small. This is further evidence that the data are not informative for this model. This could perhaps be improved if the data were collected more frequently or if data about another model compartment were collected. In particular data about hospitalisations would likely be quite informative here,

Chapter 4

Multifidelity ABC

4.1 The multifidelity problem

For ABC methods it is necessary to simulate pseudodata from the model many times. In some cases we have multiple simulators available. Some of these can be very accurate but expensive to run whilst others may be less accurate but substantially cheaper. The proposed algorithm uses multiple cheap simulators to achieve computational improvements over an ABC algorithm which uses only the expensive simulator while still converging to the same approximate posterior distribution. This is partially offset by a reduction in statistical efficiency which depends on how well correlated the cheap simulators are to the expensive one. The algorithm also uses a stochastic approximation step to identify which cheap simulators are correlated best in which regions of the sample space. This allows the algorithm to minimise the reduction in statistical efficiency.

4.2 Early rejection MCMC-ABC

Partition the parameter space Θ, into subregions, $T_1, ..., T_k$, and to each subregion assign a vector of probabilities, $\exp\left(-\phi_{T,j}\right)$ for $j = 0, 1, ..., m$. Where $j = 0$ corresponds to skipping the low fidelity step and $j = 1, ..., m$ corresponds to using the jth low fidelity simulator. Then at each iteration we propose a new θ, select a low fidelity simulator at random with the corresponding probability vector, and then perform delayed acceptance MCMC with the selected low fidelity simulator. We then update the probability vector for that region based on the difference in approximate likelihood ratios

$$H\left(\phi, \theta\right) = \left| \frac{g_{\varepsilon,i}\left(u'_{lo}\right)}{g_{\varepsilon,i}\left(u^{(t)}_{lo}\right)} - \frac{g_{\varepsilon,hi}\left(u'_{hi}\right)}{g_{\varepsilon,hi}\left(u^{(t)}_{hi}\right)} \right|$$

using a stochastic approximation step to solve

$$\int H\left(\phi, \theta\right) \pi_{ABC}\left(\theta | x\right) \mathrm{d}\theta = 0$$

The full algorithm can be found in Algorithm 4.1.

Algorithm 4.1 Early Rejection MCMC-ABC

```
**Inputs:** observed data $y$, kernel parameter $\varepsilon$, number of iterations N, initial parameter value $\theta^{(1)}$, initial discrepancy $u^{(1)}$, subregions $T_j$
**Loop over** $t = 1, ..., \mathbf{N}$
    1. Sampling Step($\theta'$)
        (a) Sample $\theta' \sim q\left(\theta'|\theta_t\right)$
    2. Augmentation Step
        (a) Identify the subregion $T$ of $\Theta$ which contains $\theta'$
        (b) Sample $i \sim \pi_{lofi}\left(i|T\right)$ where $i = 1, ..., m$ and $\pi_{lofi}\left(j|T\right) = \exp\left(-\phi^t_{T,j}\right)$
    3. Sampling Step($x'_{lo}$)
        (a) Sample $x'_{lo} \sim f_i\left(x'_{lo}|\theta'\right)$
        (b) Calculate $\alpha^{(t+1)}_{lo}\left(\theta'|\theta^{(t)}\right) = \min\left(R_{lo}, 1\right)$ with:
            $$R_{lo} = \frac{\pi\left(\theta'\right) q\left(\theta^{(t)}|\theta'\right) g_{\varepsilon,i}\left(u'_{lo}\right)}{\pi\left(\theta^{(t)}\right) q\left(\theta'|\theta^{(t)}\right) g_{\varepsilon,i}\left(u^{(t)}_{lo}\right)}$$
        (c) With probability $\alpha^{(t+1)}_{lo}$ go to step 4. Otherwise set $\left(\theta^{(t+1)}, x^{(t+1)}_{lo}, x^{(t+1)}_{hi}\right) = \left(\theta^{(t)}, x^{(t)}_{lo}, x^{(t)}_{hi}\right)$ and go to the next iteration.
    4. 4. Sampling Step($x'_{hi}$)
        (a) Sample $x'_{hi} \sim f_{hi}\left(x'_{hi}|\theta', x'_{lo}\right)$
        (b) Calculate $\alpha^{(t+1)}_{hi}\left(\theta'|\theta^{(t)}\right) = \min\left(R_{hi}, 1\right)$ with:
            $$R_{hi} = \frac{\pi\left(\theta'\right) q^*\left(\theta^{(t)}|\theta'\right) g_{\varepsilon,hi}\left(u'_{hi}\right)}{\pi\left(\theta^{(t)}\right) q^*\left(\theta'|\theta^{(t)}\right) g_{\varepsilon,hi}\left(u^{(t)}_{hi}\right)}$$
        (c) Set $\left(\theta^{(t+1)}, x^{(t+1)}_{lo}, x^{(t+1)}_{hi}\right) = \left(\theta', x'_{lo}, x'_{hi}\right)$ with probability $\alpha^{(t+1)}_{hi}\left(\theta'|\theta^{(t)}\right)$.
            Otherwise set $\left(\theta^{(t+1)}, x^{(t+1)}_{lo}, x^{(t+1)}_{hi}\right) = \left(\theta^{(t)}, x^{(t)}_{lo}, x^{(t)}_{hi}\right)$
    5. Updating step for augmentation
        (a) Calculate $p_{t+1} = \left|\frac{g_{\varepsilon,i}\left(u'_{lo}\right)}{g_{\varepsilon,i}\left(u^{(t)}_{lo}\right)} - \frac{g_{\varepsilon,hi}\left(u'_{hi}\right)}{g_{\varepsilon,hi}\left(u^{(t)}_{hi}\right)}\right|$
        (b) Compute $\phi^{(t+1)}_{T,i} = \phi^{(t)}_{T,i} + \gamma_{t+1}\left[p_{t+1} - \delta\right]_i$
        (c) Normalise $\phi^{(t+1)}_T$ such that $\sum^m_{j=1} \phi^{(t+1)}_{T,j} = 1$
**End Loop**
**Output:** parameter values $\theta^{(1)}, ..., \theta^{(N)}$, estimate $\phi^{(N)}$
```

where $\delta \geq 0$ is some tolerance for the difference in likelihood approximation and m is the number of low fidelity simulators. The obvious choice for δ is zero as this corresponds to a perfect approximation of the high fidelity simulator by the low fidelity one. However $\delta = 0$ would always push away from using the low fidelity simulator used in the current iteration. A small non-zero tolerance enables convergence to a particular low fidelity simulator. Furthermore $q^* \left(\theta' | \theta^{(t)}\right)$ and $r\left(\theta^{(t)}\right)$ are as follows to preserve detailed balance.

$$q^* \left(\theta' | \theta^{(t)}\right) = \alpha_{lo}^{(t+1)} \left(\theta' | \theta^{(t)}\right) q\left(\theta' | \theta^{(t)}\right) + \left(1 - r\left(\theta^{(t)}\right)\right) \delta_{\theta^{(t)}} \left(\theta'\right)$$

and:

$$r\left(\theta^{(t)}\right) = \int \alpha_{lo}^{(t+1)} \left(\theta' | \theta^{(t)}\right) q\left(\theta' | \theta^{(t)}\right) \mathrm{d}\theta'$$

and $\delta_{\theta^{(t)}} \left(\theta'\right)$ denotes the Dirac mass at $\theta^{(t)}$. As described in Robert, Casella and Casella (1999) this preserves the detailed balance condition of the Metropolis Hastings step of the algorithm. This then consequently preserves the convergence guarantees of the SAMC-based algorithm.

4.3 Multifidelity MCMC-ABC

Algorithm 4.2 uses Stochastic Approximation to find model selection probabilities that minimise $H\left(\phi, \theta\right) = \left| \frac{g_{\varepsilon,i}\left(u'_{lo}\right)}{g_{\varepsilon,i}\left(u_{lo}^{(t)}\right)} - \frac{g_{\varepsilon,hi}\left(u'_{hi}\right)}{g_{\varepsilon,hi}\left(u_{hi}^{(t)}\right)} \right| > 0$. This allows the algorithm to favour the low fidelity models which offer better approximations in the early rejection step. For further computational gains it would be beneficial to also use this information in an early acceptance step. Unfortunately this cannot be done without introducing some error to the posterior as we will accept samples early that would have been rejected by the high fidelity model. Nonetheless, if we perform early acceptance only when the average $H\left(\phi, \theta\right)$ for the model in that region is low then we can keep this approximation error small whilst enjoying large computational gains.

In fact we can find an estimate $\Delta_{T,i}$ of the average relative error in $g_{\varepsilon,hi}\left(u_{hi}^{(t+1)}\right)$ for model i in subregion T . Furthermore, in the case of uniform ABC kernels the average absolute error reduces to $\Delta_{T,i}$ and gives the frequency with which $g_{\varepsilon,hi}\left(u_{hi}^{(t+1)}\right)$ is estimated as 1 where it should be 0. The threshold η gives the frequency that the approximation should be used given that the error is below ϵ. This controls the trade off between continuing the stochastic approximation in that region and using the learned information to make computational savings. The parameter η should increase to 1 asymptotically to ensure convergence. This means that asymptotically as the stochastic approximation converges and there is less uncertainty about the expected error of the approximation we decide to use the approximation or not deterministically using only the expected error. Note this is not a true early acceptance step, when the approximation is used there is still the possibility of rejection in the high fidelity accept/reject step, particularly if the acceptance probability in the low fidelity step was small. However it does remove the need to always simulate from the expensive high fidelity simulator in every acceptance step.

Algorithm 4.2 Multifidelity MCMC-ABC

```
Inputs: observed data $y$, kernel parameter $\varepsilon$, number of iterations N, initial
parameter value $\theta^{(1)}$, initial discrepancy $u^{(1)}$, subregions $T_j$, early acceptance
threshold parameters $\epsilon$ and $\eta$

Loop over $t = 1, ..., \mathbf{N}$

  1. Sampling Step($\theta'$)

      (a) Sample $\theta' \sim q\left(\theta' | \theta_t\right)$

  2. Augmentation Step

      (a) Identify the subregion $T$ of $\Theta$ which contains $\theta'$

      (b) Sample $i \sim \pi_{lofi}\left(i|T\right)$ where $i = 1, ..., m$ and $\pi_{lofi}\left(j|T\right) = \exp\left(-\phi_{T,j}^{t}\right)$
```

3. Sampling Step(x'_{lo})

   (a) Sample $x'_{lo} \sim f_i\left(x'_{lo}|\theta'\right)$

   (b) Calculate $\alpha_{lo}^{(t+1)}\left(\theta'|\theta^{(t)}\right) = \min\left(R_{lo}, 1\right)$ with:

$$R_{lo} = \frac{\pi\left(\theta'\right) q\left(\theta^{(t)}|\theta'\right) g_{\varepsilon,i}\left(u'_{lo}\right)}{\pi\left(\theta^{(t)}\right) q\left(\theta'|\theta^{(t)}\right) g_{\varepsilon,i}\left(u_{lo}^{(t)}\right)}$$

   (c) With probability $\alpha_{lo}^{(t+1)}$ go to step 4 . Otherwise set $\left(\theta^{(t+1)}, x_{lo}^{(t+1)}, x_{hi}^{(t+1)}\right) = \left(\theta^{(t)}, x_{lo}^{(t)}, x_{hi}^{(t)}\right)$ and go to the next iteration.

4. Sampling Step(x'_{hi})

   (a) Sample $U \sim \text{Uniform}\left(0,1\right)$

   (b) If $\Delta_{T,i} < \epsilon$ and U $< \eta$ then approximate $g_{\varepsilon,hi}\left(u'_{hi}\right) = \frac{g_{\varepsilon,i}\left(u'_{lo}\right) g_{\varepsilon,hi}\left(u_{hi}^{(t)}\right)}{g_{\varepsilon,i}\left(u_{lo}^{(t)}\right)}$

   (c) Otherwise sample $x'_{hi} \sim f_{hi}\left(x'_{hi}|\theta', x'_{lo}\right)$ and calculate $g_{\varepsilon,hi}\left(u'_{hi}\right)$ using x'_{hi}

   (d) Calculate $\alpha_{hi}^{(t+1)}\left(\theta'|\theta^{(t)}\right) = \min\left(R_{hi}, 1\right)$ with:

$$R_{hi} = \frac{\pi\left(\theta'\right) q^*\left(\theta^{(t)}|\theta'\right) g_{\varepsilon,hi}\left(u'_{hi}\right)}{\pi\left(\theta^{(t)}\right) q^*\left(\theta'|\theta^{(t)}\right) g_{\varepsilon,hi}\left(u_{hi}^{(t)}\right)}$$

   (e) Set $\left(\theta^{(t+1)}, x_{lo}^{(t+1)}, x_{hi}^{(t+1)}\right) = \left(\theta', x'_{lo}, x'_{hi}\right)$ with probability $\alpha_{hi}^{(t+1)}\left(\theta'|\theta^{(t)}\right)$. Otherwise set $\left(\theta^{(t+1)}, x_{lo}^{(t+1)}, x_{hi}^{(t+1)}\right) = \left(\theta^{(t)}, x_{lo}^{(t)}, x_{hi}^{(t)}\right)$

5. Updating step for augmentation

   (a) If $\Delta_{T,i} < \epsilon$ and $U < \eta$ go to the next iteration

   (b) Calculate $p_{t+1} = \left|\frac{g_{\varepsilon,i}\left(u'_{lo}\right)}{g_{\varepsilon,i}\left(u_{lo}^{(t)}\right)} - \frac{g_{\varepsilon,hi}\left(u'_{hi}\right)}{g_{\varepsilon,hi}\left(u_{hi}^{(t)}\right)}\right|$

   (c) Update $\Delta_{T,i} = \frac{1}{k+1}\left(k\Delta_{T,i} + p_{t+1}\right)$ where k is the number of times we have previously chosen model i in subregion T

   (d) Compute $\phi_{T,i}^{(t+1)} = \phi_{T,i}^{(t)} + \gamma_{t+1}\left[p_{t+1} - \delta\right]_i$

```
    (e) Normalise $\phi_T^{(t+1)}$ such that $\sum_{j=1}^{m} \phi_{T,j}^{(t+1)} = 1$
End Loop
Output: parameter values $\theta^{(1)}, ..., \theta^{(N)}$, estimate $\phi^{(N)}$
```

4.4 Demonstrating the algorithm with a toy example

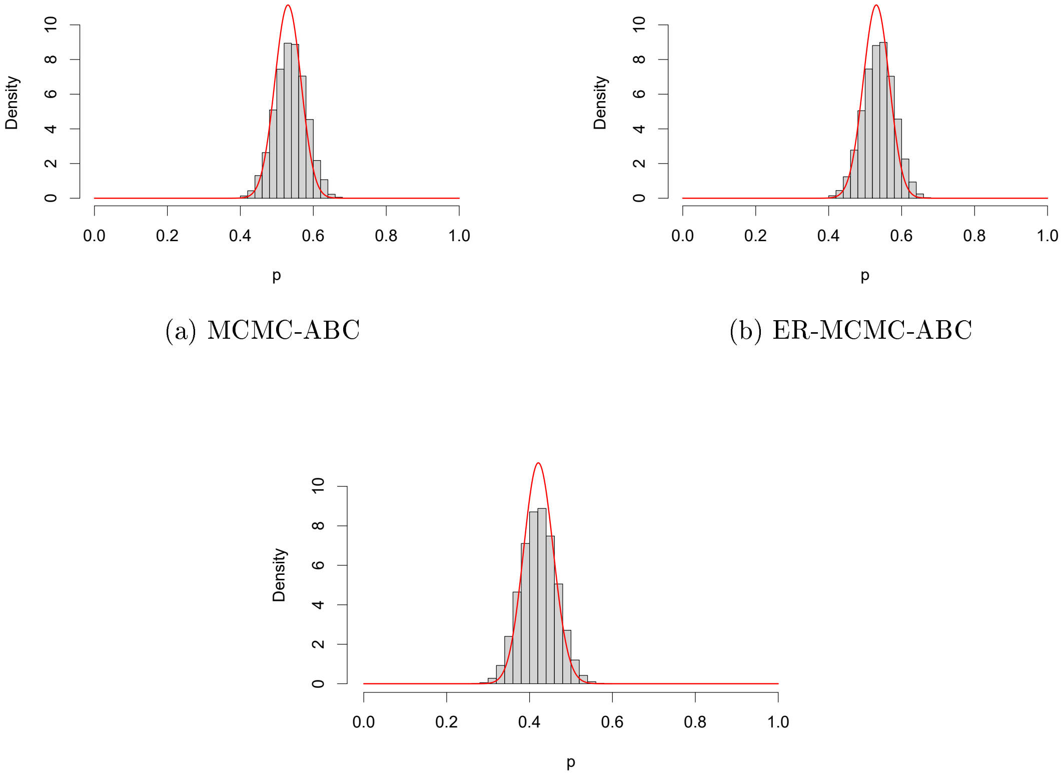
We demonstrate the algorithm on a normal mixture model. For the toy example the two normal components have known means and variances and the mixture is strongly bimodal. Inference is to be made about the mixture parameter p. The mixture distribution is as described in (4.1) with $p = 0.5$. We sample 200 data points from the mixture and perform ABC using the Euclidean distance between the order statistics as the discrepancy. For the prior on p we take a uniform distribution on $[0, 1]$.

$$X_{hi} \sim p \times N(0,1) + (1-p) \times N(5,1) \tag{4.1}$$

First we perform standard MCMC-ABC sampling using only the high fidelity model which is the true model with unknown parameter p. We use a uniform ABC kernel with a maximum tolerance of 6. We run the sampler for a total of 2×10^5 iterations. Then we run the early rejection and multifidelity ABC samplers with the two normal components of the mixture as low fidelity models

$$X_{lo}^{(1)} \sim N(0,1)$$
$$X_{lo}^{(2)} \sim N(5,1)$$

and the true model again as the high fidelity model. The ABC kernels for the two low fidelity models are uniform with a maximum tolerances of 43.5. The early rejection and multifidelity ABC samplers are also run for a total of 2×10^5 iterations each. For the multifidelity algorithm we choose an error threshold of $\epsilon = 0.3$ for the early approximation step. Histograms of the samples are shown below in Figure 4.1 with the true posterior overlayed in red.



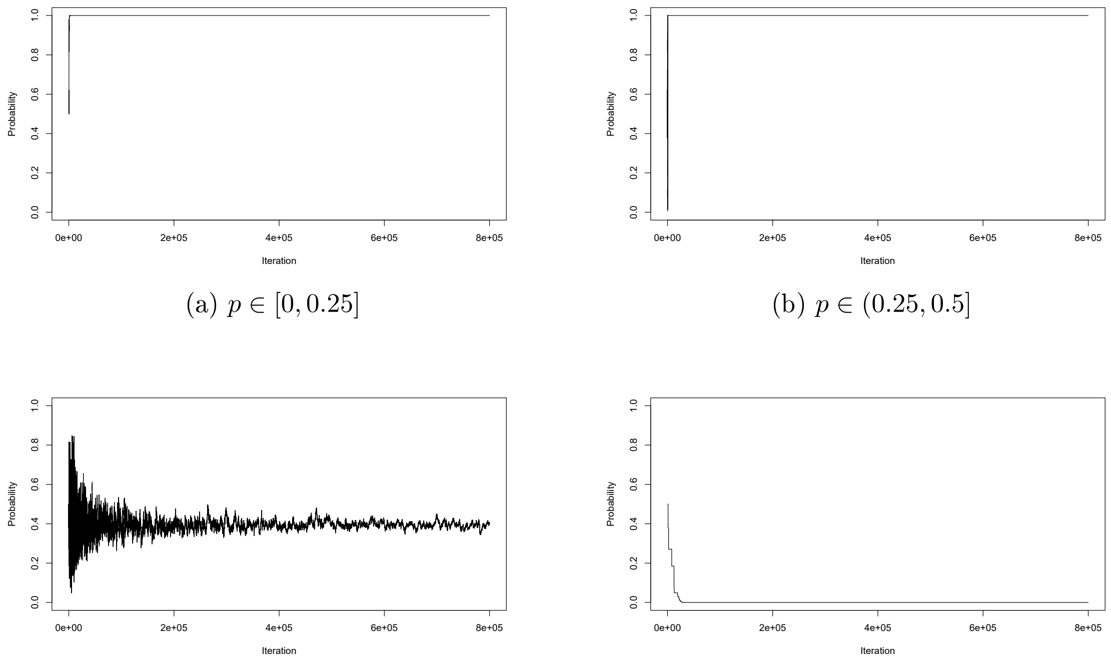
(a) MCMC-ABC

(b) ER-MCMC-ABC

(c) MF-MCMC-ABC

Figure 4.1: Histograms of posterior estimates of the mixture parameter using standard MCMC ABC with the high fidelity simulator (a), using early rejection MCMC-ABC (b), and using Multifidelity MCMC-ABC (c)

The sampling methods produce samples that match well with the true posterior, demonstrating that the early rejection and multifidelity algorithms may sample from the ABC posterior without introducing additional bias even with very poor low fidelity models if the low fidelity models are good approximations within a region of the parameter space. We also plot the convergence of the low fidelity model selection probabilities. We plot the probability that $X_{lo}^{(2)} \sim N(5, 1)$ will be used as the low fidelity model in each of the four subregions in Figure 4.2. Note that when p is small the sampler should prefer $X_{lo}^{(2)} \sim N(5,1)$ and when p is large the sampler should prefer $X_{lo}^{(1)} \sim N(0,1)$.



(a) $p \in [0, 0.25]$

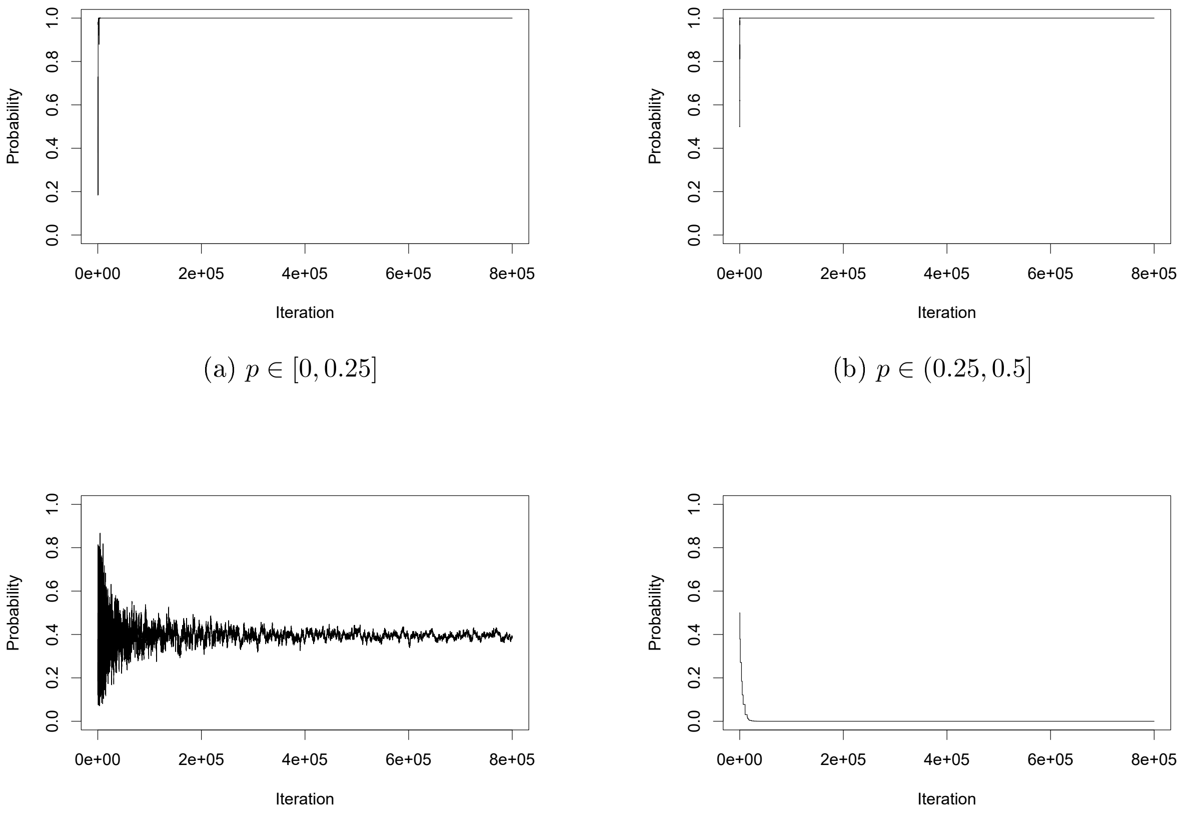
(b) $p \in (0.25, 0.5]$

(c) $p \in (0.5, 0.75]$

(d) $p \in (0.75, 1]$

Figure 4.2: Probability of selecting $X_{lo}^{(2)} \sim N(5,1)$ as the low fidelity model in each subregion at iteration t using the early rejection algorithm

As can be seen in Figure 4.2 the early rejection sampler correctly favours $X_{lo}^{(1)} \sim N(0,1)$ in the regions where p is large and favours $X_{lo}^{(2)} \sim N(5,1)$ instead in the regions where p is small. Moreover, we can see that since visits to the subregion $(0.5, 0.75]$ are proposed much more often this probability is updated much more often. If there were not such a strong preference for one model in the other subregions it is unlikely that the probabilities would have converged. Nonetheless, this is not a problem. Algorithm 4.2 clearly gains the most efficiency by learning in the most often proposed subregions and these are the subregions which converge well. In Figure (4.3) we see that the multifidelity sampler also correctly chooses $X_{lo}^{(1)} \sim N(0,1)$ in regions where p is large and $X_{lo}^{(2)} \sim N(5,1)$ elsewhere.



(a) $p \in [0, 0.25]$

(b) $p \in (0.25, 0.5]$

(c) $p \in (0.5, 0.75]$

(d) $p \in (0.75, 1]$

Figure 4.3: Probability of selecting $X_{lo}^{(2)} \sim N(5, 1)$ as the low fidelity model in each subregion at iteration t using the early rejection algorithm

As observed in the early rejection sampler outputs the selection probability is updated most often in the subregion $(0.5, 0.75]$ where visits are proposed much more often. In the other regions the algorithm develops a strong preference for the appropriate low fidelity simulator very quickly. We observe that convergence is not impeded by the use of the early approximation step used to accept proposals without simulating from the high fidelity simulator. This suggests that the multifidelity ABC algorithm can achieve a good approximation of the high fidelity model posterior whilst enjoying considerable computational advantages.

Chapter 5

Conclusion

5.1 Adaptive Kernel ABC

We introduced a new framework for approximate Bayesian computation which aims to mitigate the local trapping problem whilst yielding reduced approximation bias. This methodology uses stochastic approximation to estimate the marginal density of the discrepancy and adapt the kernel using this estimate. We further extend this methodology to the continuous discrepancy domain where we can introduce an automatic method to determine the weights used in the stochastic approximation. This automatic method additionally optimises the Metropolis-Hastings acceptance rate and improves convergence. Moreover. we present a method which can both detect and adapt to ABC model misspecification. We demonstrate these methods on challenging benchmark examples where we see the expected improvements over existing methods. Furthermore, we use our proposed methods to fit a sophisticated SEIR-type model to data from the Ebola outbreak in Liberia in 2014. We found that our methods compare favourably to the existing methodology previously used to fit this model. Future work within the SAMCABC and CCABC frameworks may aim to adaptively construct the SAMC partition and CCMC grids respectively.

Adaptive construction of the partition or grid would enable full automation of the method.

5.2 Multifidelity ABC

In Chapter 4 we introduced methodology to extend ABC to the multifidelity setting. In particular our method enables the comparison of multiple competing low fidelity simulators to optimise the efficiency whilst fitting a multifidelity model. More impressively, our method can even favour different low fidelity simulators in different regions of the parameter space, enabling the use of more expensive approximations in more challenging regions whilst retaining the computational benefits of a cheaper approximation in regions where the cheaper approximation can suffice. We demonstrate our method on a benchmark example where the method correctly identifies the appropriate low fidelity approximation to use in each subregion.

Part II

Model selection by stochastic reversible jumps

Chapter 6

Overview and context

6.1 Stochastic gradient methods for big data

Given a large set of data $x \in \mathcal{X}$, and a collection of models $m_k = \{f_k(\cdot|\theta_k); \theta_k \in \Theta_k\} \in \mathcal{M}$ indexed by k we aim to perform model selection whilst making inference about the parameters $\theta_k \in \Theta_k$ of each model. We assign a prior $\pi(\theta_k, m_k)$ and write the likelihood function of model m_k as $f_k(x|\theta_k, m_k)$. This leads to a posterior distribution

$$\pi(\theta_k, m_k|x) \propto \pi(\theta_k, m_k) f(x_k|\theta_k, m_k)$$

where the normalising constant is typically intractable. For now we focus on methods which aim to make this inference for a fixed model m. Thus we suppress the model index k in notation for brevity. Later in Section 6.4 we extend these methods to the model selection problem. Several popular MCMC methods such as the Metropolis Adjusted Langevin Algorithm (MALA) (Roberts and Stramer 2002) and the Hamiltonian Monte Carlo (HMC) algorithm (Duane et al. 1987; Neal et al. 2011) aim to improve upon the efficiency of standard MCMC methods by exploiting the gradient of the posterior to make informed proposals. These methods can greatly improve the statistical efficiency over other MCMC methods such as the Metropolis-Hastings algorithm but when the size of the data becomes large these gradients can become too computationally expensive to evaluate repeatedly. Recent advances have introduced scalable MCMC algorithms such as the Stochastic Gradient Langevin Dynamics (SGLD) algorithm (Welling and Teh 2011) and the Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) algorithm (T. Chen, E. Fox and Guestrin 2014) which aim to improve the computational efficiency of MALA and HMC respectively. Both stochastic algorithms achieve improved computational efficiency by evaluating the gradient of the likelihood on only a subset of the data at each iteration to reduce the cost of each iteration. Moreover, by taking asymptotically decreasing step sizes we can guarantee convergence whilst skipping the still expensive Metropolis-Hastings accept/reject step (Welling and Teh 2011; T. Chen, E. Fox and Guestrin 2014; C. Chen, Ding and Carin 2015).

6.1.1 Stochastic Gradient Langevin Dynamics

The Stochastic Gradient Langevin Dynamics (SGLD) algorithm uses a stochastic estimate for the gradient of the target distribution to make proposals from a discrete time approximation of the Langevin diffusion (Welling and Teh 2011; Nemeth and Fearnhead 2020; Y.-A. Ma, T. Chen and E. B. Fox 2015). By using an unbiased estimate of the gradient of the log-likelihood function based on a subsample size n which can be much smaller than the total sample size N we can greatly reduce the computational cost of evaluating the gradient. This in turn greatly reduces the cost of the whole algorithm since the gradient is typically the computational bottleneck for the algorithm (Nemeth and Fearnhead 2020). To describe the algorithm let's first consider the target distribution $\pi\left(\theta|x\right)$ that admits the form:

$$\pi\left(\theta|x\right) \propto \exp\left(-U\left(\theta|x\right)\right)$$

where $U(\theta|x)$ is called the energy function. For brevity we write $U(\theta)$ for the energy function hereafter. Then we are interested in the gradient of the energy $\nabla U(\theta)$. In particular we can write

$$\nabla \hat{U}(\theta) = \nabla \log(\pi(\theta)) + \frac{N}{n} \sum_{x_i \in S} \nabla \log f(x_i|\theta) \tag{6.1}$$

for the unbiased estimator of the gradient evaluated using a subsample, S, of the data. Now we can compute the updating equation for $\theta^{(t+1)}$ from $\theta^{(t)}$ given stepsize, $\varepsilon^{(t)}$, and hence the full SGLD algorithm is given in Algorithm 6.1.

Algorithm 6.1 Stochastic Gradient Langevin Dynamics

```
Input: number of iterations K, subsample size $n$, step sizes $\varepsilon$, initial parameter $\theta^{(1)}$
Loop over $t = 1, ..., \mathbf{K}$
    1. Subsample $S$ from the data without replacement
    2. Estimate $\nabla \hat{U}\left(\theta^{(t)}\right)$ using Equation 6.1
    3. Draw $\eta^{(t)} \sim N\left(0, \varepsilon^{(t)}\right)$
    4. Calculate $\theta^{(t+1)} = \theta^{(t)} - \frac{\varepsilon}{2} \nabla \hat{U}(\theta) + \eta^{(t)}$
End Loop
Output: parameters $\theta^{(1)}, ..., \theta^{(\mathrm{K})}$
```

Stochastic gradient methods such as SGLD can massively reduce the computational costs of the efficient but sometimes expensive gradient based MCMC methods. Furthermore, given an appropriately decaying step size the stochastic gradient methods still have convergence guarantees. In practice however, a decaying step size $\varepsilon^{(t)}$ is not used. Instead the decay is either halted once the step size reaches some small value or the step size is fixed at some small value $\varepsilon^{(t)} = \varepsilon$ for all iterations t (Welling and Teh 2011; Nemeth and Fearnhead 2020). This trades off some approximation error in the Metropolis-Hastings acceptance ratio in exchange for better long term exploration by the algorithm (Welling and Teh 2011; Nemeth and Fearnhead 2020). By making this trade off we no longer have guaranteed convergence to the true target density but instead to some approximation of it, where for a sufficiently small step size that approximation error is considered negligible.

6.1.2 Stochastic Gradient Hamiltonian Monte Carlo

The Hamiltonian Monte Carlo (HMC) algorithm extends the Metropolis-adjusted Langevin algorithm (MALA) by introducing auxillary momentum variables, r, to the dynamics used to make proposals (T. Chen, E. Fox and Guestrin 2014). Together with a mass matrix, M, which is typically set to be an identity matrix we write the joint target distribution

$$\pi\left(\theta, r\right) \propto \exp\left(-H\left(\theta, r\right)\right)$$

where $H\left(\theta, r\right) = U\left(\theta\right) - \frac{1}{2}r^T M^{-1} r$. Similarly to MALA the bottleneck of the HMC algorithm is the evaluation of the gradient $\nabla H\left(\theta, r\right)$. Thus the computational burden of the algorithm can be reduced by evaluating that gradient using an unbiased estimator as in SGLD (T. Chen, E. Fox and Guestrin 2014; Nemeth and Fearnhead 2020; Y.-A. Ma, T. Chen and E. B. Fox 2015). Then with stepsize $\varepsilon^{(t)}$ we have updating equations

$$\theta^{(t+1)} = \theta^{(t)} + \varepsilon^{(t)} M^{-1} r^{(t-1)} \tag{6.2}$$

$$r^{(t+1)} = r^{(t)} - \frac{\varepsilon^{(t)}}{2} \nabla \hat{U}\left(\theta^{(t+1)}\right) - \frac{\varepsilon^{(t)}}{2} C M^{-1} r^{(t-1)} + \eta^{(t)} \tag{6.3}$$

where C is a user specified friction matrix that reduces the impact of the noise (T. Chen, E. Fox and Guestrin 2014; C. Chen, Ding and Carin 2015). Then using (6.2) and (6.3) the Stochastic Gradient HMC algorithm is shown in Algorithm 6.2.

Algorithm 6.2 Stochastic Gradient Hamiltonian Monte Carlo

```
Input: number of iterations K, subsample size $n$, step sizes $\varepsilon$, initial parameter $\theta^{(1)}$,
friction matrix $C$
Loop over $t = 1, ..., \mathbf{K}$
    1. Subsample $S$ from the data without replacement
    2. Estimate $\nabla\hat{U}(\theta)$ using Equation 6.1
    3. Draw $\eta^{(t)} \sim N\left(0, C\varepsilon^{(t)}\right)$
    4. Update $\theta$ and $r$ using Equations 6.2 and 6.3
End Loop
Output: parameters $\theta^{(1)}, ..., \theta^{(\mathrm{K})}$
```

Similarly to SGLD a fixed step size is often used in practice to improve the long term exploration of the algorithm at the cost of some small approximation error (T. Chen, E. Fox and Guestrin 2014).

6.2 Variance reduction for stochastic gradient methods

The stochastic gradient algorithms detailed in Section 6.1 reduce the computational burden associated with the exact Monte Carlo methods. Given appropriately decreasing step sizes and an unbiased estimator for the gradient the convergence of the algorithm is still guaranteed (C. Chen, Ding and Carin 2015; T. Chen, E. Fox and Guestrin 2014; Dubey et al. 2016). However, the estimator for the gradient often has high variance since only a small subsample of the data is used and this can cause poor mixing and slow convergence (Dubey et al. 2016; Li et al. 2019; Chatterji et al. 2018). To improve the mixing of the algorithm several variance reduction methods have been proposed (Dubey et al. 2016; Chatterji et al. 2018). In this section we focus on two algorithms called Stochastic Average Gradient Accelerated (SAGA) and Stochastic Variance Reduced Gradient (SVRG) and in particular we focus on applying these algorithms to SGLD. Both algorithms attempt to address the problem of high variance by retaining information about the entire dataset between iterations.

The SAGA algorithm retains the most recently evaluated gradient at each data point so that an estimate of the gradient across the full dataset can be used as a control variate. To do this initialise $\alpha_i^{(0)} = \theta^{(0)}$, $g_i^{(0)} = \nabla \log \left(f\left(x_i|\theta\right)\right)$, and $g^{(0)} = \sum g_i^{(0)}$ for $i = 1, ..., N$. Then when we update $\theta^{(t+1)}$ we also update $\alpha^{(t+1)}$ and $g_i^{(t+1)}$ at the data points in the subset used at that iteration. The $g_i^{(t+1)}$ not used at that iteration are set as $g_i^{(t)}$. Then the estimate of the gradient given in (6.1) becomes:

$$\nabla \hat{U}\left(\theta\right) = \nabla \log\left(\pi\left(\theta\right)\right) + \frac{N}{n}\sum_{x_i \in S}\left(\nabla \log f\left(x_i|\theta\right) - \nabla \log f\left(x_i|\alpha\right)\right) - g \tag{6.4}$$

and the SAGA Langevin Dynamics (SAGA-LD) algorithm is given in Algorithm 6.3.

Algorithm 6.3 SAGA-LD

```
Input: number of iterations K, subsample size $n$, step sizes $\varepsilon$, initial parameter $\theta^{(1)}$
Loop over $t = 1, ..., \mathbf{K}$
    1. Subsample $S$ from the data without replacement
    2. Estimate $\nabla \hat{U}\left(\theta^{(t)}\right)$ using Equation 6.4
    3. Draw $\eta^{(t)} \sim N\left(0, \varepsilon^{(t)}\right)$
    4. Calculate $\theta^{(t+1)} = \theta^{(t)} - \frac{\varepsilon}{2}\nabla \hat{U}\left(\theta\right) + \eta^{(t)}$
    5. Set $\alpha_i^{(t+1)} = \theta^{(t+1)}$ for $i$ such that $x_i \in S$ and $\alpha_i^{(t+1)} = \alpha_i^{(t)}$ for $i$ such that $x_i \notin S$
    6. Calculate $g_i^{(t+1)} = \nabla \log f\left(x_i|\alpha_i^{(t+1)}\right)$ for $i$ such that $x_i \in S$ and set $g_i^{(t+1)} = g_i^{(t)}$ for $i$ such that $x_i \notin S$
    7. Calculate $g^{(t+1)} = \sum_{i=1}^{N} g_i^{(t+1)}$
End Loop
Output: parameters $\theta^{(1)}, ..., \theta^{(\mathrm{K})}$
```

The SAGA-LD algorithm has a much improved convergence rate compared to standard SGLD at the cost of high memory overhead (Dubey et al. 2016; Chatterji et al. 2018). When the dataset is very large and the memory costs of storing every subgradient g_i becomes prohibitive the SVRG Langevin Dynamics (SVRG-LD) algorithm can be used instead (Dubey et al. 2016; Chatterji et al. 2018). Instead of storing the N individual gradients like SAGA-LD does, the SVRG-LD algorithm instead evaluates the gradient on the full dataset every m iterations and stores the full gradient of the log likelihood, $\tilde{g}$, and $\tilde{\theta} = \theta$ at the full evaluation. Thus the estimate of the gradient given in (6.1) becomes:

$$\nabla \hat{U}(\theta) = \nabla \log(\pi(\theta)) + \frac{N}{n} \sum_{x_i \in S} \left(\nabla \log f(x_i|\theta) - \nabla \log f\left(x_i|\tilde{\theta}\right)\right) - \tilde{g} \tag{6.5}$$

Thus SVRG-LD trades memory overhead for computational cost. This cost becomes small as m becomes large(Dubey et al. 2016). However, as θ moves further away from $\tilde{\theta}$, $\tilde{g}$ becomes less useful as a control variate (Chatterji et al. 2018). Hence if m is too large then variance reduction will not be achieved. The SVRG algorithm is given in Algorithm 6.4.

Algorithm 6.4 SVRG-LD step

```
Input: number of iterations K, subsample size $n$, step sizes $\varepsilon$, initial parameter $\theta^{(1)}$,
gradient update iteration $m$
Loop over $t = 1, ..., \mathbf{K}$
    1. If $t \mod m = 0$ then set $\tilde{\theta} = \theta^{(t)}$ and calculate $\tilde{g} = \sum_{i=1}^{N} \log f\left(x_i|\tilde{\theta}\right)$
    2. Subsample $S$ from the data without replacement
    3. Estimate $\nabla \hat{U}(\theta)$ using Equation 6.5
    4. Draw $\eta^{(t)} \sim N\left(0, \varepsilon^{(t)}\right)$
    5. Calculate $\theta^{(t+1)} = \theta^{(t)} - \frac{\varepsilon}{2} \nabla \hat{U}(\theta) + \eta^{(t)}$
End Loop
Output: parameters $\theta^{(1)}, ..., \theta^{(\mathrm{K})}$
```

6.3 Bias reduction for MCMC under uncertainty

Stochastic gradient methods typically skip the Metropolis-Hastings accept-reject step since the acceptance probability is assumed to be near one for small step sizes and calculating the acceptance ratio requires expensive evaluation of the likelihood on the full dataset (Welling and Teh 2011). This can result in very slow exploration of the parameter space since ideally the step size for full gradient based MCMC methods is large. One possible compromise that allows for larger steps is to calculate the acceptance ratio using only the subset of the data used to estimate the gradient. This is much cheaper than evaluating the acceptance ratio exactly but it introduces additional bias into the algorithm via the acceptance ratio such that we no longer converge to the target distribution.

The bias can be reduced using methods for MCMC under uncertainty. One such method makes assumptions about the distribution of the log acceptance ratio under the uncertainty and then attempts to estimate the bias and apply an appropriate penalty term to the acceptance ratio (Ceperley and Dewing 1999). In particular, if we can assume that the subsampling estimate of the log acceptance ratio is normally distributed then by calculating the log acceptance ratio, $\log(r)$, for n_p different subsamples $S_1, ..., S_{n_p}$we can construct the following estimators:

$$\delta = \frac{\sum_{i=1}^{n_p} \log(r_i)}{n_p}$$
$$\chi^2 = \frac{\sum_{i=1}^{n_p} (\log(r_i) - \delta)^2}{n_p(n_p - 1)}$$

for the log acceptance ratio and its variance respectively. Then the corrected acceptance probability can be constructed as

$$\alpha_p = \min(1, \exp(-\delta - u_B))$$

where $u_B = \frac{\chi^2}{2} + \frac{\chi^4}{4(n_p+1)} + \frac{\chi^6}{3(n_p+1)(n_p+3)} + ...$ is the penalty term (Ceperley and Dewing 1999). In practice when $\frac{\chi^2}{2}$ is small then a penalty using only the lowest order term, $u_B = \frac{\chi^2}{2}$, is sufficient to achieve very low bias.

6.4 The reversible jump algorithm

The Reversible Jump algorithm extended the scope of MCMC methods to include transdimensional sampling problems (Green 1995). By using a technique called dimension matching, the reversible jump MCMC (RJ-MCMC) methodology enables an MCMC sampler to transition between different parameter spaces. This is done by extending those spaces using auxillary variables, u, and then constructing bijective transformations, $G_{k \to k'}$ between the extended spaces. Then denoting the probability density on the auxillary variables as $\varphi_{k \to k'}\left(u_{k \to k'}\right)$ and the Jacobian of the transformation as $J_{k \to k'}$ we can write the standard RJ-MCMC update at state (θ_k, k) in Algorithm 6.5.

Algorithm 6.5 RJ-MCMC step

```
Input: current parameter $\theta_k$, current model $k$
    1. Sample $k' \sim q\left(k^{(t)}\right)$
    2. Sample $u_{k \to k'} \sim \varphi\left(\cdot\right)$
    3. Apply the transformation $\theta_{k'} = G_{k \to k'}\left(\theta_k, u_{k \to k'}\right)$
    4. Accept $(\theta_{k'}, k')$ with probability $\alpha = \min\left(r_{k \to k'}, 1\right)$ where
           $$r_{k \to k'} = \frac{\pi\left(\theta_{k'}, k'\right) \varphi_{k' \to k}\left(u_{k' \to k}\right) q\left(k', k\right)}{\pi\left(\theta_k, k\right) \varphi_{k \to k'}\left(u_{k \to k'}\right) q\left(k, k'\right)} J_{k \to k'}$$
    5. Otherwise set $(\theta_{k'}, k') = (\theta_k, k)$
Output: parameter $\theta_{k'}$, model $k'$
```

In practice, choosing $G_{k \to k'}$ and $\varphi_{k \to k'}\left(u_{k \to k'}\right)$ can be difficult, and poor choices lead to slow mixing, with high rejection rates even between comparable model spaces. As a result of these practical limitations, transdimensional MCMC using standard

RJ-MCMC is often extremely inefficient and improving that efficiency can be taxing on the user.

6.5 Annealed Importance Sampling RJ

To achieve more efficient RJ-MCMC sampling various methods have been proposed, including the annealed importance sampling reversible jump algorithm (AISRJ) (Karagiannis and Andrieu 2013). Let us first consider an idealised Metropolis-Hastings algorithm where the model transition probabilities are known given in Algorithm 6.6. The acceptance rate of the idealised algorithm is greater than or equal to the the acceptance rate of any reversible jump sampler $\mathbb{E}\left(\alpha^{ID}\right) \geq \mathbb{E}\left(\alpha^{RJ}\right)$. Although this algorithm is typically unavailable since the model transition probabilities are usually not known, the algorithm has guaranteed convergence under only mild assumptions without the inefficiency introduced in the RJ method. Hereafter we will refer to Algorithm 6.6 as the Ideal algorithm and compare both AISRJ and the proposed methodology in Chapter 7 to this algorithm.

Algorithm 6.6 Ideal Model Transition MH step

```
Input: current model $k$
  1. Sample $k' \sim q(\cdot|k)$
  2. Accept $k'$ with probability $\alpha = \min(r_{k\to k'}, 1)$ where
         $r_{k\to k'} = \frac{\pi(k')\, q(k', k)}{\pi(k)\, q(k, k')}$
  3. Otherwise set $k' = k$
Output: model $k'$
```

Returning to RJ methods then, the AISRJ algorithm aims to improve upon the efficiency of the RJ-MCMC algorithm by implementing Annealed Importance Sampling (Jarzynski 1997a; Jarzynski 1997b; Neal 2005) ideas in the context of reversible jump (Karagiannis and Andrieu 2013). By incorporating these ideas into the reversible jump algorithm a ‘bridge’ can be constructed between models which consists of intermediate distributions which the sampler transitions through. By using this bridge the AISRJ has a much smoother transition between models and in fact it can be shown that AISRJ converges to the Ideal algorithm as the number of intermediate distributions $T \to \infty$. Given intermediate densities $\rho_t \left(\theta_{k'}, u_{k' \to k} | k \to k'\right)$ the AISRJ algorithm is as shown in Algorithm 6.7. The algorithm used SGLD to transition between the intermediate densities to produce a path $\left(\theta_{k'}^{(1)}, u_{k' \to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k' \to k}^{(T-1)}\right)$ at each iteration. This path can be used to calculate the annealed importance sampling weight $r_{k \to k'}^{(0:T-1)}$ and thus the acceptance probability $a_{k,k'}^{(0:T-1)}$ for the transition. Throughout this thesis we consider geometric annealing distributions such that the intermediate densities take the form

$$\begin{aligned} \rho_t \left(\theta_{k'}, u_{k' \to k} | k \to k'\right) \propto & \left(\pi\left(\theta_k, k\right) \varphi_{k \to k'}\left(u_{k \to k'}\right) J_{k \to k'}\right)^{(1-\gamma_t)} \\ & \times \left(\pi\left(\theta_{k'}, k'\right) \varphi_{k' \to k}\left(u_{k' \to k}\right)\right)^{\gamma_t} \end{aligned} \tag{6.6}$$

where $\gamma_t = \frac{t}{T}$ and our results assume that these intermediate densities are used. Alternative choices can be made (Gelman and Meng 1998) but our results do not hold for these alternatives which are not considered hereafter.

Algorithm 6.7 AISRJ step

```
Input: current parameter $\theta_k$, current model $k$
  1. Sample $k' \sim q(k)$
  2. Set $\theta_k^{(0)} = \theta_k$ and draw $u_{k\to k'}^{(0)} \sim \varphi_{k\to k'}(\cdot)$
  3. Compute $\left(\theta_{k'}^{(0)}, u_{k'\to k}^{(0)}\right) = G_{k\to k'}\left(\theta_k^{(0)}, u_{k\to k'}^{(0)}\right)$
  4. Generate a path $\left(\theta_{k'}^{(1)}, u_{k'\to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k'\to k}^{(T-1)}\right)$
  5. Compute the annealing importance weight $r_{k\to k'}^{(0:T-1)}$:
       $r_{k\to k'}^{(0:T-1)} = \frac{\pi\left(\theta_{k'}^{(T-1)}, k'\right)}{\pi\left(\theta_k^{(0)}, k\right)} \frac{\varphi_{k'\to k}\left(u_{k'\to k}^{(T-1)}\right)}{\varphi_{k\to k'}\left(u_{k\to k'}^{(T-1)}\right)} J_{k'\to k}^{-1}\left(\theta_{k'}^{(0)}, u_{k'\to k}^{(0)}\right) \times \prod_{t=1}^{T-1} \frac{\rho_t\left(\theta_{k'}^{(t-1)}, u_{k'\to k}^{(t-1)}; k \to k'\right)}{\rho_t\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}; k \to k'\right)}$
  6. Accept the proposed value $\left(k', \theta_{k'}^{(T-1)}\right)$ with acceptance probability:
       $a_{k,k'}^{(0:T-1)} = \min\left\{1, \frac{q(k|k')}{q(k'|k)} r_{k\to k'}^{(0:T-1)}\right\}$
  7. Otherwise set $\left(k', \theta_{k'}^{(T-1)}\right) = (k, \theta_k)$
Output: parameter $\theta_{k'}^{(T-1)}$, model $k'$
```

The AISRJ algorithm has strong convergence guarantees and as $T \to \infty$ we later show that the algorithm converges to the desireable Ideal algorithm. Furthermore, since the 'bridge' smooths the transition by adjusting the proposed parameters, the algorithm can achieve this even with poor choices for $G_{k\to k'}$ and $\varphi_{k\to k'}(u_{k\to k'})$ (Karagiannis and Andrieu 2013). However, the algorithm does not scale well to big data. For each intermediate step the likelihood must be evaluated on the full dataset. Moreover, the transitions between intermediate steps may require further evaluations of the log-likelihood or the gradient of the log-likelihood. Hence, as the size of the data grows large, the computational costs of AISRJ quickly become prohibitive.

Chapter 7

Stochastically Annealed Reversible Jump

7.1 Preliminary results about AISRJ

To address the issues of the AISRJ algorithm we will propose a stochastic annealing mechanism for the reversible jump and show that it has similar convergence guarantees whilst being much less computationally burdensome. To facilitate this proposal we define the following Wasserstein ergodicity assumption found in Rudolf and Schweizer (2018) which is key to many of the results in this chapter.

Assumption 1. *For a transition kernel P_α we assume that there are constants $C \in (0, \infty)$and $\rho \in (0, 1)$ for which:*

$$\tau\left(P_\alpha\right) = \sup_{\theta, \theta' \in \Theta, \theta \neq \theta'} \frac{W\left(P_\alpha^j\left(\theta, \cdot\right), P_\alpha^j\left(\theta', \cdot\right)\right)}{d\left(\theta, \theta'\right)} \leq C\rho^n$$

where $W\left(\cdot,\cdot\right)$ is the Wasserstein distance and $d\left(\cdot,\cdot\right)$ is the metric with which the Wasserstein distance has been defined.

We also provide the following preliminary result regarding AISRJ given approximation error in the acceptance ratio of AISRJ compared to the ideal sampler $\mathcal{E}\left(\theta,\theta'\right)$. The proof of Proposition 7.1.1 and proofs of all the other results in this chapter can be found in Appendix B.

Proposition 7.1.1. *Assume that the ideal sampler in Algorithm 6.6 satisfies the assumption defined in Assumption 1. Then writing P_α for the transition kernel of the AISRJ algorithm and $Q^*\left(\cdot,\cdot\right)$ for the transition kernel of the proposal density of the reversible jump step we have the following bound for the Wasserstein distance between the jth iterations*

$$\begin{aligned} W\left(P^j_{\alpha^*}\left(\theta,\cdot\right),P^j_\alpha\left(\theta,\cdot\right)\right) &= \frac{C\left(1-\rho^j\right)}{\left(1-\rho\right)} \\ &\times \sup_{\theta\in\Theta}\left(\left(\left(\left(\int_\Theta d\left(\theta,\theta'\right)^2 Q\left(\theta,d\theta'\right)\right)\left(\int_\Theta \mathcal{E}\left(\theta,\theta'\right)^2 Q\left(\theta,d\theta'\right)\right)\right)^{\frac{1}{2}}\right)\end{aligned}$$

Since the AISRJ acceptance ratio is a consistent estimator of the idealised acceptance ratio in T (Karagiannis and Andrieu 2013) it follows that this bound can be made arbitrarily small by increasing T and thus the AISRJ algorithm converges to the ideal algorithm in the Wasserstein distance as T grows large. Later we will show that the proposed algorithm converges to the AISRJ algorithm in a similar manner and hence also to the ideal.

7.2 The Stochastically Annealed Reversible Jump algorithm

The AISRJ algorithm becomes computationally expensive as the data grows large. Both the generation of the annealing path and the evaluation of the annealing weight require computation of the full likelihood and/or the gradient of the likelihood. The proposed algorithm, called Stochastically Annealed Reversible Jump (SARJ), aims to reduce the computational costs of both steps. This is required to since reducing the computational burden of only one step will result in the other becoming a bottleneck and hence in failure to meaningfully reduce the computational costs of the full algorithm.

To reduce the cost of the annealing path generation we can simply make each transition using the SGLD algorithm. This allows the acceptance ratio of the transition kernels to be ignored. Furthermore, if we were to evaluate the annealing importance weight on the full data using a path generated by SGLD we would still converge to the ideal sampler with the bound given in Proposition 7.1.1. That is we can reduce the costs of the path generation step without impacting the convergence of the algorithm. However, we must also reduce the cost of computing the annealing importance weight in order to reduce the overall cost of the algorithm. To do this we propose to sample T subsets $x^{(0)}, ..., x^{(T-1)}$ each of size m from the data. Each subset $x^{(t)} = \left(x_1^{(t)}, ..., x_m^{(t)}\right)$ is sampled independently from the full data without replacement. We construct the following stochastic annealing importance weight:

$$\begin{aligned} r_{k\to k'}^{(0:T-1),m} =& \frac{\pi\left(k', \theta_{k'}^{(T-1)} | x^{(0)}\right)}{\pi\left(k, \theta_k^{(0)} | x^{(0)}\right)} \frac{\varphi_{k'\to k}\left(u_{k'\to k}^{(T-1)}\right)}{\varphi_{k\to k'}\left(u_{k\to k'}^{(T-1)}\right)} J_{k'\to k}^{-1}\left(\theta_{k'}^{(0)}, u_{k'\to k}^{(0)}\right) \\ &\times \prod_{t=1}^{T-1} \frac{\rho_t\left(\theta_{k'}^{(t-1)}, u_{k'\to k}^{(t-1)}; k \to k', x^{(t)}\right)}{\rho_t\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}; k \to k', x^{(t)}\right)} \end{aligned}$$

where each likelihood evaluation is done using only a subset of the full data. The full algorithm is given in Algorithm 7.1.

Algorithm 7.1 Stochastically Annealed Reversible Jump (SARJ)

```
Input: current parameter $\theta_k$, current model $k$
  1. Propose model $k'$ with probability $q(\cdot|k)$
  2. Set $\theta_k^{(0)} = \theta_k$ and draw $u_{k\to k'}^{(0)} \sim \varphi_{k\to k'}(\mathrm{d}u_{k\to k'})$
  3. Compute $\left(\theta_{k'}^{(0)}, u_{k'\to k}^{(0)}\right) = G_{k\to k'}\left(\theta_k^{(0)}, u_{k\to k'}^{(0)}\right)$
  4. Sample $T$ minibatches $x^{(0)}, ..., x^{(T-1)}$ of size $m$ from the data
  5. Generate a path $\left(\theta_{k'}^{(1)}, u_{k'\to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k'\to k}^{(T-1)}\right)$ where $\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}\right)$ is drawn
     using Stochastic Gradient Langevin Dynamics
  6. Compute the stochastic annealing importance weight $r_{k\to k'}^{(0:T-1,m)}$:
     $$r_{k\to k'}^{(0:T-1),m} = \frac{\pi\left(k', \theta_{k'}^{(T-1)}|x^{(0)}\right)}{\pi\left(k, \theta_k^{(0)}|x^{(0)}\right)} \frac{\varphi_{k'\to k}\left(u_{k'\to k}^{(T-1)}\right)}{\varphi_{k\to k'}\left(u_{k\to k'}^{(T-1)}\right)} J_{k'\to k}^{-1}\left(\theta_{k'}^{(0)}, u_{k'\to k}^{(0)}\right)$$
     $$\times \prod_{t=1}^{T-1} \frac{\rho_t\left(\theta_{k'}^{(t-1)}, u_{k'\to k}^{(t-1)}; k\to k', x^{(t)}\right)}{\rho_t\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}; k\to k', x^{(t)}\right)}$$
  7. Accept the proposed value $\left(k', \theta_{k'}^{(T-1)}\right)$ with acceptance probability:
     $$a_{k,k'}^{(0:T-1)} = \min\left\{1, \frac{q(k|k')}{q(k'|k)} r_{k\to k'}^{(0:T-1),m}\right\}$$
Output: parameter $\theta_{k'}^{(T-1)}$, model $k'$, stochastic annealing importance weight $r_{k\to k'}^{(0:T-1,m)}$
```

7.3 Theoretical Investigation

To demonstrate the convergence guarantees of the proposed algorithm let's first consider the distribution of the log stochastic annealing importance weight, $\log r_{k\to k'}^{(0:T-1,m)}$, as T grows large. The following results show that under mild conditions the log stochastic annealing importance weight is asymptotically normally distributed. We remind the reader that proofs of these results can be found in Appendix B.

Proposition 7.3.1. *Given a path* $\left(\theta_{k'}^{(1)}, u_{k'\to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k'\to k}^{(T-1)}\right)$*, a minibatch size* $m < N$*, and that the cumulative distribution function of the likelihood is Lipschitz continuous then the log stochastic annealing importance weight,* $\log r_{k\to k'}^{(0:T-1),m}$*, converges in distribution to a Normal distribution* $N\left(\mu_T, \sigma_T^2\right)$ *as* $T \to \infty$

Furthermore we have that the log stochastic annealing importance weight $\log r_{k\to k'}^{(0:T-1),m}$ is an unbiased estimator of the full annealing importance weight.

Proposition 7.3.2. *Given a path* $\left(\theta_{k'}^{(1)}, u_{k'\to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k'\to k}^{(T-1)}\right)$ *and a minibatch size,* m*, then* $\mathbb{E}\left[\log r_{k\to k'}^{(0:T-1),m}\right] = \log r_{k\to k'}^{(0:T-1),N}$ *where* $r_{k\to k'}^{(0:T-1),m}$ *is the ratio achieved with minibatch size* m*.*

So it follows from Propositions 7.3.1 and 7.3.2 that the stochastic annealing importance weight $r_{k\to k'}^{(0:T-1),m}$ is also a consistent estimator of the full annealing importance weight $r_{k\to k'}^{(0:T-1),N}$.

Corollary 7.3.3. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Then* $\sigma_T^2 = Var\left(\log r_{k\to k'}^{(0:T-1),m}\right) \to 0$ *as* $T \to \infty$ *and hence* $\log r_{k\to k'}^{(0:T-1),m}$ *is a consistent estimator of* $\log r_{k\to k'}^{(0:T-1),N}$*.*

We now consider the stochastic annealing importance weight, $r_{k\to k'}^{(0:T-1),m}$, which is used to evaluate the Metropolis-Hastings accept-reject ratio. This estimator is biased however the bias goes to zero as T grows large or as m grows to N.

Proposition 7.3.4. *The stochastic annealed importance weight* $r_{k\to k'}^{(0:T-1),m}$ *is biased. Under the conditions of Proposition 7.3.1 this bias asymptotically tends to a factor of* $e^{\frac{\sigma_T^2}{2}}$ *as* T *grows large.*

Corollary 7.3.5 follows from Proposition 7.3.4 since $\sigma_T^2 \to 0$ as the subsample size m or number of intermediate steps T grow large.

Corollary 7.3.5. *As* $m \to N$ *or* $T \to \infty$ *the bias in* $r_{k\to k'}^{(0:T-1),m}$ *tends to zero.*

Finally we show that the Wasserstein distance between the jth step of the SARJ Markov chain and the jth iteration of the Ideal sampler is bounded. We do this by showing that the distance between the stochastic and standard AISRJ algorithms is bounded.

Proposition 7.3.6. *Assume that the standard Annealed Importance Sampling reversible jump algorithm (Algorithm 6.7) satisfies the Wasserstein ergodicity assumption (Assumption 1). Then writing $P_{\tilde{\alpha}}$ for the transition kernel of the SARJ algorithm and $Q(\cdot,\cdot)$ for the transition kernel of the proposal density we have the following bound for the Wasserstein distance between the jth iterations*

$$W\left(P_{\alpha}^{j}(\theta,\cdot), P_{\tilde{\alpha}}^{j}(\theta,\cdot)\right) \leq \frac{C\left(1-\rho^{j}\right)}{(1-\rho)} \sup_{\theta\in\Theta}\left(\left(\left(\int_{\Theta} d\left(\theta,\theta'\right)^{2} Q\left(\theta, d\theta'\right)\right)\left(\int_{\Theta}\left(\exp\left(\frac{\sigma_{T}^{2}}{2}\right)-1\right)^{2} Q\left(\theta, d\theta'\right)\right)\right)^{\frac{1}{2}}\right)$$

The bound in Proposition 7.3.6 can be controlled by controlling σ_T^2. Thus we can make this bound arbitrarily small by increasing m or T. The following Corollary follows by applying the triangle inequality to the results of Propositions 7.1.1 and 7.3.6.

Corollary 7.3.7. *The Wasserstein distance between the jth iterations of the SARJ algorithm and the ideal algorithm $W\left(P_{\tilde{\alpha}}^{j}(\theta,\cdot), P_{\alpha^*}^{j}(\theta,\cdot)\right)$ is bounded and the can be made smaller by increasing the number of intermediate steps, T.*

Hence the SARJ algorithm converges to the ideal sampler in the Wasserstein distance as T grows large.

7.4 Improving convergence with bias and variance reduction

Similarly to the stochastic gradient methods discussed in Chapter 6 the introduction of stochasticity to the AISRJ algorithm can introduce new challenges to replace the computational cost of the full algorithm. Like those stochastic gradient algorithms the convergence of SARJ may be slow if the variance of the stochastic annealing importance weight is large. More problematically, convergence may not be guaranteed if there is not enough computational budget to sufficiently control the bias with large T. In this section we present mechanisms to reduce both the variance and the bias of the estimator alongside theoretical results demonstrating the effectiveness of these methods. Proofs of these results can be found in Appendix B.

7.4.1 Variance reduction for SARJ

We introduce variance reduction to the SARJ algorithm. Unlike in Dubey et al. (2016), and indeed in most of the stochastic gradient literature, we apply the variance reduction to the stochastic estimate of the likelihood itself instead of to the estimates of its gradient. This way we directly target the additional variance which affects the annealing importance weights. Let's consider the variance reduced log likelihood estimator of SAGA in (6.4) and apply the SAGA mechanism to the log likelihood function

$$\tilde{f}\left(x|\theta\right)=\frac{N}{n}\sum_{x_i\in S}\left(\log f\left(x_i|\theta\right)-\log f\left(x_i|\alpha\right)\right)-g \tag{7.1}$$

with $g=\sum_{i=1}^{N}\log f\left(x_i|\alpha_i^{(t+1)}\right)$. We show that when we replace the standard log likelihood estimator in the SARJ algorithm with our variance reduced estimator (7.1) as shown in Algorithm 7.2, we achieve variance reduction in the likelihood which propogates to the stochastic annealing importance weight.

Proposition 7.4.1. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Variance reduction in $\hat{f}_k^{(t)}$ and $\hat{f}_{k'}^{(t)}$ for all $t=0,1,...,T-1$ implies variance reduction in $r_{k\to k'}^{(0:T-1),m}$*

The Variance Reduced SARJ algorithm (VRSARJ) shown in Algorithm 7.2 can therefore use the variance reduced SAGA estimator of the likelihood in (7.1) to reduce the variance of the annealed importance sampling weight. This reduces the approximation error introduced by subsampling and improves the sampling output.

Algorithm 7.2 Variance Reduced SARJ

```
Input: current parameter $\theta_k$, current model $k$

 1. Propose model $k'$ with probability $q\left(\cdot|k\right)$
 2. Set $\theta_k^{(0)}=\theta_k$ and draw $u_{k\to k'}^{(0)}\sim\varphi_{k\to k'}\left(\mathrm{d}u_{k\to k'}\right)$
 3. Compute $\left(\theta_{k'}^{(0)},u_{k'\to k}^{(0)}\right)=G_{k\to k'}\left(\theta_k^{(0)},u_{k\to k'}^{(0)}\right)$
 4. Sample a $T$ minibatches $x^{(0)},...,x^{(T-1)}$ of size $m$ from the data
 5. Generate a path $\left(\theta_{k'}^{(1)},u_{k'\to k}^{(1)}\right),...,\left(\theta_{k'}^{(T-1)},u_{k'\to k}^{(T-1)}\right)$ where $\left(\theta_{k'}^{(t)},u_{k'\to k}^{(t)}\right)$ is drawn using Stochastic Gradient Langevin Dynamics
 6. Compute $\tilde{f}_k$ and $\tilde{f}_{k'}$ for all $t$ where

        $$\tilde{f}\left(x^{(t)}|\theta^{(t)}\right)=\frac{N}{n}\sum_{x_i\in S}\left(\log f\left(x_i^{(t)}|\theta^{(t)}\right)-\log f\left(x_i^{(t)}|\alpha_i^{(t)}\right)\right)-g_i^{(t)}$$

 7. Compute the stochastic annealing importance weight $r_{k\to k'}^{(0:T-1)}$using the variance reduced log-likelihoods $\tilde{f}_k$ and $\tilde{f}_{k'}$ of models $k$ and $k'$ respectively:

        $$r_{k\to k'}^{(0:T-1)}=\frac{\exp\left(\hat{f}_{k'}\left(x^{(0)}|\theta_{k'}^{(T-1)}\right)\right)\pi\left(\theta_{k'}^{(0)}|k'\right)\pi\left(k'\right)}{\exp\left(\hat{f}_k\left(x^{(0)}|\theta_k^{(0)}\right)\right)\pi\left(\theta_k^{(0)}|k\right)\pi\left(k\right)}$$
        $$\times\frac{\varphi_{k'\to k}\left(u_{k'\to k}^{(T-1)}\right)}{\varphi_{k\to k'}\left(u_{k\to k'}^{(0)}\right)}\times J_{k'\to k}^{-1}\left(\theta_{k'}^{(0)},u_{k'\to k}^{(0)}\right)$$
        $$\times\prod_{t=1}^{T-1}\frac{\rho_t\left(\theta_{k'}^{(t-1)},u_{k'\to k}^{(t-1)};k\to k',x^{(t)}\right)}{\rho_t\left(\theta_{k'}^{(t)},u_{k'\to k}^{(t)};k\to k',x^{(t)}\right)}$$

 8. Accept the proposed value $\left(k',\theta_{k'}^{(T-1)}\right)$ with acceptance probability:

        $$a_{k,k'}^{(0:T-1)}=\min\left\{1,\frac{q\left(k|k'\right)}{q\left(k'|k\right)}r_{k\to k'}^{(0:T-1)}\right\}$$

 9. Set $\alpha_i^{(t+1)}=\theta^{(t+1)}$ for $i$ such that $x_i\in S$ and $\alpha_i^{(t+1)}=\alpha_i^{(t)}$ for $i$ such that $x_i\notin S$
10. Calculate $g_i^{(t+1)}=\log f\left(x_i|\alpha_i^{(t+1)}\right)$ for $i$ such that $x_i\in S$ and set $g_i^{(t+1)}=g_i^{(t)}$ for $i$ such that $x_i\notin S$
11. Calculate $g^{(t+1)}=\sum_{i=1}^{N}g_i^{(t+1)}$

Output: parameter $\theta_{k'}^{(T-1)}$, model $k'$, stochastic annealing importance weight $r_{k\to k'}^{(0:T-1,m)}$
```

Furthermore, it follows from Proposition 7.4.1 that the bias in the Metropolis-Hastings acceptance ratio is also reduced since this bias depends on the variance of the stochastic annealing importance weight.

Proposition 7.4.2. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Then variance reduction in $\hat{f}_k$ reduces σ_T^2 and thus since the bias in $r_{k\to k'}^{(0:T-1),m}$ is asymptotically equal to $e^{\frac{\sigma_T^2}{2}}$ there is bias reduction in $r_{k\to k'}^{(0:T-1),m}$*

Finally since the bias in the Metropolis-Hastings acceptance probability is reduced by variance reduction, the Wasserstein distance between the exact and stochastic AISRJ algorithms also decreases and convergence to the ideal sampler improves.

Corollary 7.4.3. *Variance reduction in $\hat{f}_k$ results in an improved bound on the Wasserstein distance*

$$W\left(P_{\alpha}^{j}\left(\theta,\cdot\right),P_{\tilde{\alpha}}^{j}\left(\theta,\cdot\right)\right)\leq\frac{C\left(1-\rho^{j}\right)}{\left(1-\rho\right)}\sup_{\theta\in\Theta}\left(\left(\left(\int_{\Theta}d\left(\theta,\theta'\right)^{2}Q\left(\theta,d\theta'\right)\right)\left(\int_{\Theta}\left(\exp\left(\frac{\sigma_T^2}{2}\right)-1\right)^{2}Q\left(\theta,d\theta'\right)\right)\right)^{\frac{1}{2}}\right)$$

between the exact and SARJ algorithms since σ_T^2 is reduced.

7.4.2 Bias reduction for SARJ

We further show in Algorithm 7.3 that the uncertainty penalty method discussed in Section 6.3 can be applied to the SARJ algorithm to directly target the bias in the SARJ Metropolis-Hastings acceptance ratio. By calculating n estimates of the annealed importance sampling weight $r_{k\to k'}^{(0:T-1),m}$ we estimate the bias in the estimator and introduce a correction term. Moreover, we find an upper bound for the expected remaining bias after the penalty has been applied and an upper bound in probability for the remaining bias itself.

Proposition 7.4.4. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Given the uncertainty penalty of Algorithm*

7.3 we can upper bound the expected remaining bias factor $\zeta_{T,n} = \exp\left(\frac{\sigma_T^2}{2} - \frac{\hat{\sigma}_{T,n}^2}{2}\right) \leq 1 + \frac{\sigma_T^4}{4(n-1)} \max\left\{1, \frac{\sigma_T^2}{2} - \frac{\hat{\sigma}_{T,n}^2}{2}\right\}$.

Note that if we consider the distribution of $\frac{\hat{\sigma}_{T,n}^2}{2}$ under the conditions of Proposition 7.3.1 we have $\frac{\hat{\sigma}_{T,n}^2}{2} \sim \Gamma\left(\frac{n-1}{2}, \frac{n-1}{\sigma_T^2}\right)$ and we can find a bound K such that:

$$\mathbb{P}\left(\mathbb{E}\left[\zeta_{T,n}\right] \geq K\right) = p$$

for arbitrarily small p. Figure 7.1 shows that for $p = 0.99$ the upper bound on the remaining bias factor $\zeta_{T,n}$ grows large as σ_T^2 grows large. Furthermore, Figure 7.1 shows that the rate of this growth is reduced as n grows large and the estimator of the bias correction factor becomes more precise. The bias factor with no correction is also shown for comparison.

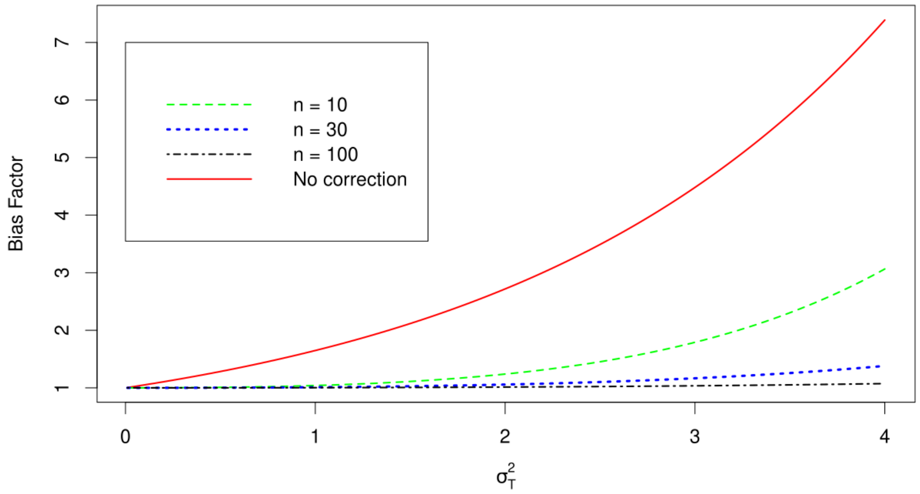


Figure 7.1: The bias factor $\zeta_{T,n}$ is shown on the y axis with the variance of the annealed importance sampling weight estimator shown on the x axis. The growth of the bias factor is shown for no correction (red), and correction using 10 (green), 30 (blue) and 100 (black) estimates of the annealed importance sampling weight $r_{k \to k'}^{(0:T-1),m}$. Note that the growth slows as n grows large.

Algorithm 7.3 SARJ with uncertainty penalty

Input: current parameter θ_k, current model k

1. Sample model $k' \sim q\left(\cdot|k\right)$
2. Set $\theta_k^{(0)} = \theta_k$ and draw $u_{k\to k'}^{(0)} \sim \varphi_{k\to k'}\left(\mathrm{d}u_{k\to k'}\right)$
3. Compute $\left(\theta_{k'}^{(0)}, u_{k'\to k}^{(0)}\right) = G_{k\to k'}\left(\theta_k^{(0)}, u_{k\to k'}^{(0)}\right)$
4. Sample a set of T minibatches $\left(x^{(0)}, ..., x^{(T-1)}\right)$ of size m from the data
5. Generate a path $\left(\theta_{k'}^{(1)}, u_{k'\to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k'\to k}^{(T-1)}\right)$ where $\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}\right)$ is drawn using Stochastic Gradient Langevin Dynamics
6. Sample n more sets of T minibatches $\left(x^{(0)}, ..., x^{(T-1)}\right)$ of size m from the data
7. For each set of minibatches $\boldsymbol{x}^{(i)}$, compute the stochastic annealing importance weight $r_{k\to k'}^{(0:T-1,i)}$ for $i = 1, ..., n$

$$r_{k\to k'}^{(0:T-1,i)} = \frac{\pi\left(k', \theta_{k'}^{(T-1)}|x^{(0)}\right)}{\pi\left(k, \theta_k^{(0)}|x^{(0)}\right)} \frac{\varphi_{k'\to k}\left(u_{k'\to k}^{(T-1)}\right)}{\varphi_{k\to k'}\left(u_{k\to k'}^{(T-1)}\right)} J_{k'\to k}^{-1}\left(\theta_{k'}^{(0)}, u_{k'\to k}^{(0)}\right)$$
$$\times \prod_{t=1}^{T-1} \frac{\rho_t\left(\theta_{k'}^{(t-1)}, u_{k'\to k}^{(t-1)}; k \to k', x^{(t)}\right)}{\rho_t\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}; k \to k', x^{(t)}\right)}$$

8. For each stochastic annealing importance weight $r_{k\to k'}^{(0:T-1,i)}$ calculate the negative log acceptance ratio $w^{(i)}$:

$$w^{(i)} = -\log\left(\frac{q\left(k|k'\right)}{q\left(k'|k\right)} r_{k\to k'}^{(0:T-1,i)}\right)$$

9. Calculate estimates of the mean, δ, and variance, σ^2 of the $y^{(i)}$:

$$\delta = \frac{\sum_{i=1}^n w^{(i)}}{n}, \ \sigma^2 = \frac{\sum_{i=1}^n \left(w^{(i)} - \delta\right)^2}{n\left(n-1\right)}$$

10. Accept the proposal with acceptance probability:

$$a_{k\to k'}^{(0:T-1)} = \min\left(1, \exp\left(-\delta - \frac{\sigma^2}{2}\right)\right)$$

Output: parameter $\theta_{k'}^{(T-1)}$, model k', stochastic annealing importance weight $r_{k\to k'}^{(0:T-1,m)}$

In addition to the upper bound on the expected remaining bias factor we also present a bound on the remaining bias factor itself, given in probability, with a constant rate of convergence in T and also in n if the variance σ^2 satisfies a minor condition.

Proposition 7.4.5. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Given the uncertainty penalty of Algorithm 7.3 we have an upper bound for the bias factor* $\zeta_{T,n} = \exp\left(\frac{\sigma_T^2}{2} - \frac{\hat{\sigma}_{T,n}^2}{2}\right)$. *Consider the sequence* $Y_{T,n} = \zeta_{T,n} - 1$ *which describes the difference between the bias factor and unbiasedness as* n *or* T *grows. Then* $Y_{T,n} = o_P(1)$ *in* T *and when a further condition on the variance* σ_T^2 *of the annealed importance sampling weight estimator* $r_{k\to k'}^{(0:T-1),m}$ *is satisfied then* $Y_{T,n} = o_P(1)$ *in* n *as well.*

From these propositions it follows that the Wasserstein distance between the stochastic and standard AISRJ algorithms also decreases to zero as T grows large.

Corollary 7.4.6. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Given the uncertainty penalty of Algorithm 7.3 the Wasserstein distance* $W\left(P_{\alpha}^{j}(\theta,\cdot), P_{\tilde{\alpha}}^{j}(\theta,\cdot)\right) = o_p(1)$ *in* T*, and* $W\left(P_{\alpha}^{j}(\theta,\cdot), P_{\tilde{\alpha}}^{j}(\theta,\cdot)\right) = o_p(1)$ *in* n *for large enough* T *such that the variance* σ_T^2 *of the annealed importance sampling weight estimator* $r_{k\to k'}^{(0:T-1),m}$ *satisfies* $\sigma_T^2 \leq 2\log(1+\epsilon)$

7.5 Demonstrating convergence with a toy example

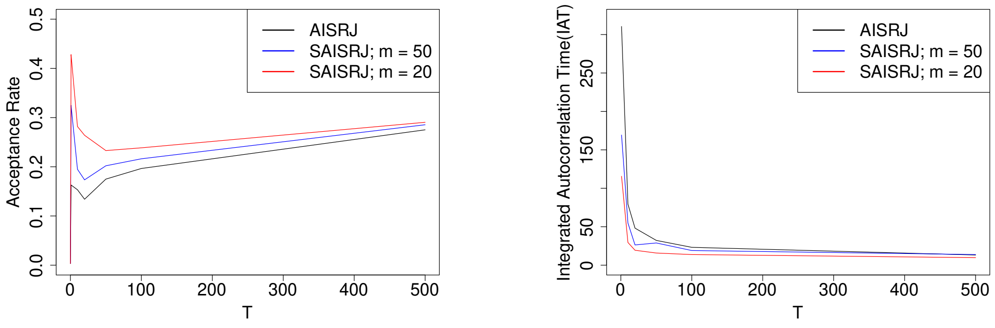
We demonstrate the performance of our algorithm against benchmark examples. We apply the method to a model selection problem with a generalised linear model. Consider first the logistic regression model with two independent variables x_1 and x_2. Furthermore, consider a uniform prior on the models and weak normal priors on the parameters centred at zero with standard deviations of 100. We apply the algorithm to compare two models k_1 and k_2 where k_2 contains x_2 and k_1 does not.

$$\text{logit}\left(\pi_{k_1}\left(y_i = 1|\mathbf{x}_i\right)\right) = \beta_0 + \beta_1 x_{1,i};$$

$$\text{logit}\left(\pi_{k_2}\left(y_i = 1|\mathbf{x}_i\right)\right) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i},$$

To construct the artificial data we generate 100 simulations from model k_2 with true parameters $(\beta_0, \beta_1, \beta_2) = (-0.2, 4, 1)$.

For the reversible jump we move to model k_1 from k_2 by removing β_2 and move to model k_2 from k_1 by proposing a new parameter value $\beta_2 \sim N(2, 1)$. When running the standard reversible jump with this proposal the mixing is extremely poor with an acceptance rate less than 0.01. Moreover, the standard reversible jump algorithm gets stuck in model k_1 despite the data having been generated from model k_2. We demonstrate that even with poorly chosen proposals the SARJ algorithm performs well. We find that as T grows large and the algorithm converges to the ideal we approach an acceptance rate of 0.3 with $\pi(k_2|y, x_1, x_2) = 0.80$. We also apply the exact AISRJ algorithm to compare it with the SARJ algorithm and show that the results are comparable.

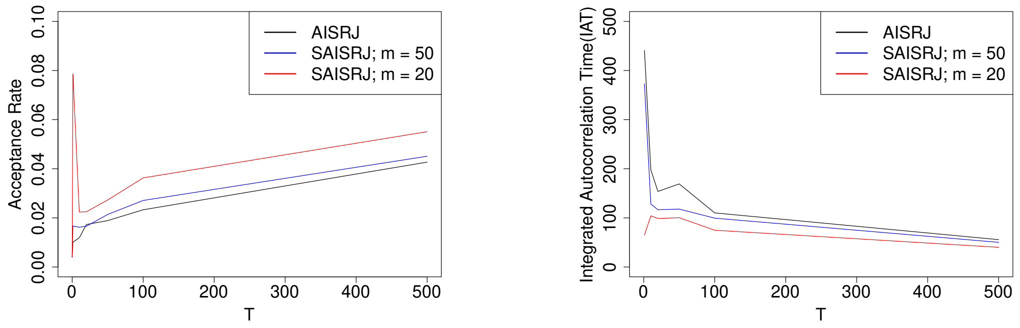


(a) Acceptance Rate

(b) Integrated Autocorrelation Time

Figure 7.2: Acceptance rate and IAT by T for the exact AISRJ algorithm (Black); the stochastic AISRJ algorithm using only 50% of the data (Blue) and the stochastic AISRJ algorithm using only 20% of the data (Red)

Now we consider simulations from a logistic regression model with 10 independent variables $x_1, ..., x_{10}$ and we want to use reversible jump for full variable selection. We generate 100 simulations from the model with true parameters $\beta = (-0.2, 4, 2, 2, 0.3, 1, 0.5, 0.1, 0, 0, 0)$ such that the variables x_8, x_9 and x_{10} are not included in the true model. For the reversible jump we index the models with indicator vectors, k, describing the variable inclusion. A new model, k_2, is proposed from the current model, k_1, by choosing a variable, x_i, from $x_1, ..., x_{10}$ uniformly at random. Then for the reversible jump step a new parameter value $\beta_i \sim N(6, 2)$ is proposed. This proposal is deliberately chosen to be poor since in more complex problems choosing an appropriate proposal can be very difficult. The standard reversible jump algorithm samples have very poor mixing with an acceptance rate less than 0.005. Again we apply both the standard and stochastic AISRJ algorithms and find that as T grows large the mixing improves and we achieve an acceptance rate of about 0.04 with exact AISRJ.

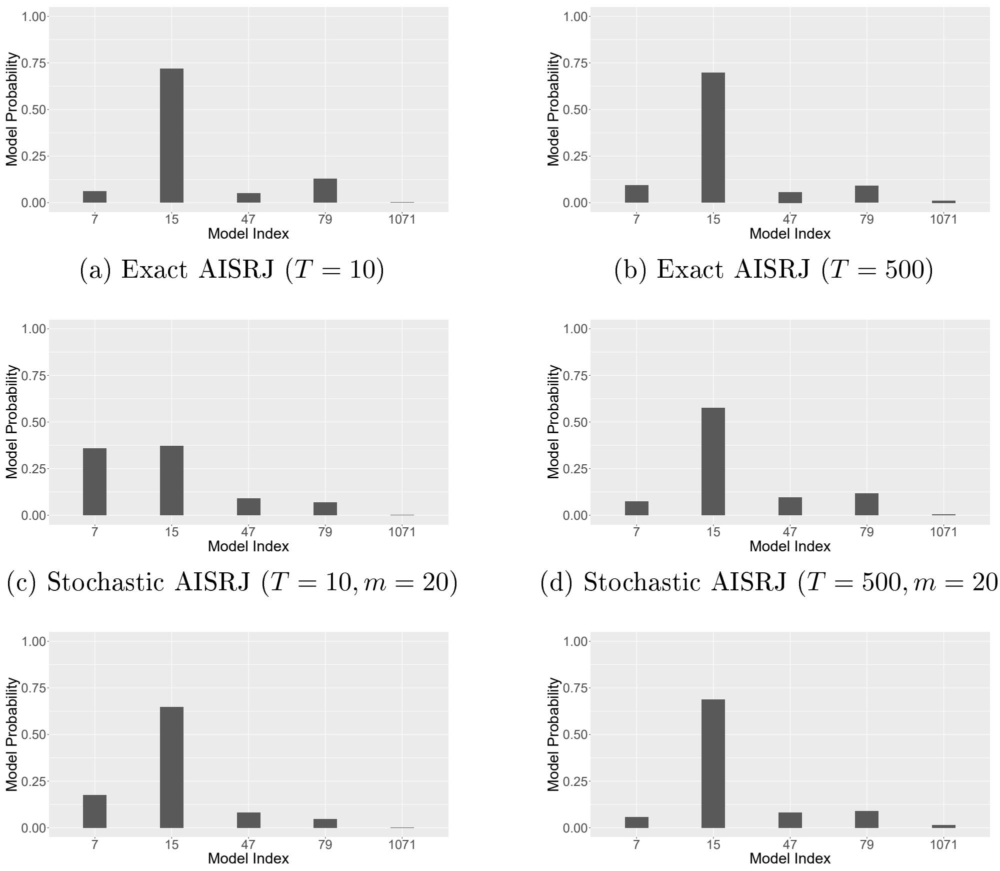


(a) Acceptance Rate

(b) Integrated Autocorrelation Time

Figure 7.3: Acceptance rate and IAT by T for the exact AISRJ algorithm (Black); the stochastic AISRJ algorithm using only 50% of the data (Blue) and the stochastic AISRJ algorithm using only 20% of the data (Red)

It is clear from Figure 7.3 that as T grows large, the acceptance rate and integrated autocorrelation time (IAT) of the SARJ algorithm converge to the AISRJ algorithm. However, we also see in Figure 7.3 that for small T there are large spikes in the acceptance rate and IAT which result from the large variance in the stochastic annealed importance sampling weight $r_{k\to k'}^{(0:T-1),m}$ of the SARJ algorithm using very few intermediate densities, especially when only a very small proportion of the data are used. We can further see the impact of this variance in Figure 7.4 where it can clearly be seen that the stochastic AISRJ algorithm produces a biased estimate of the probability mass function of the model distribution when T and m are small as expected by Proposition 7.3.4. Furthermore, we see that as T grows large this bias shrinks and we recover the same estimate of the probability mass function of the model distribution as produced by exact AISRJ. This demonstrates the behaviour expected from Proposition 7.3.6.



(a) Exact AISRJ ($T = 10$)

(b) Exact AISRJ ($T = 500$)

(c) Stochastic AISRJ ($T = 10, m = 20$)

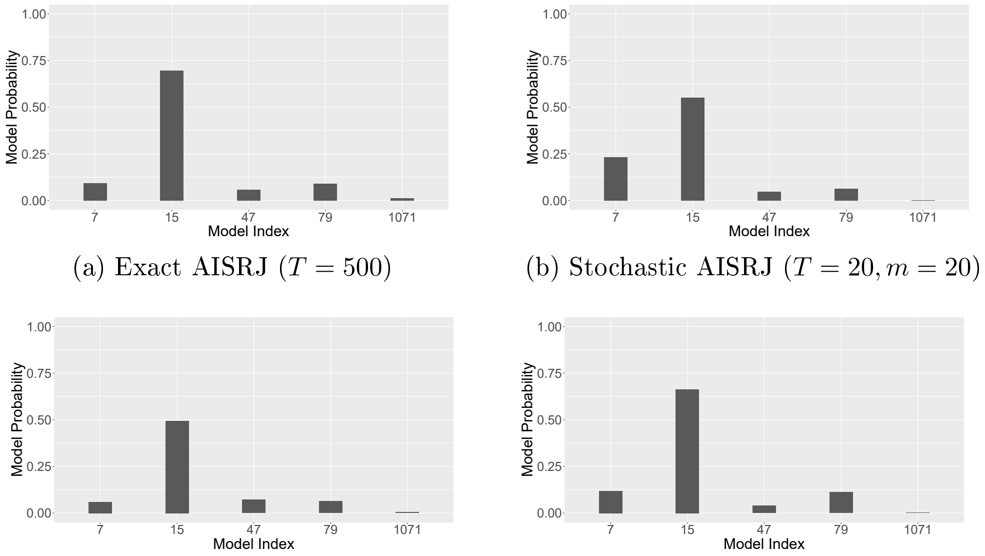
(d) Stochastic AISRJ ($T = 500, m = 20$)

(e) Stochastic AISRJ ($T = 10, m = 50$)

(f) Stochastic AISRJ ($T = 500, m = 50$)

Figure 7.4: Estimated probability mass of the five most probable models with model indices on the x axes and model probabilities on the y axes. The posterior model probabilities of exact AISRJ are provided as a reference.

In Figure 7.4 we see probability mass functions of the posterior model distributions $\pi\left(m_k|x,\theta_k\right)$. Taking the posterior of the exact AISRJ algorithm with many intermediate steps shown in Figure 7.5a as a reference close to the ideal, we observe in Figure 7.4 that using a very small subsample of the data can lead to considerable bias in the posterior if a small number of intermediate steps are used. While Figure 7.4 also shows that this can be overcome with a larger number of intermediate steps we can also demonstrate the capacity to prevent this bias using variance reduction and uncertainty penalty based methods.



(a) Exact AISRJ ($T = 500$)

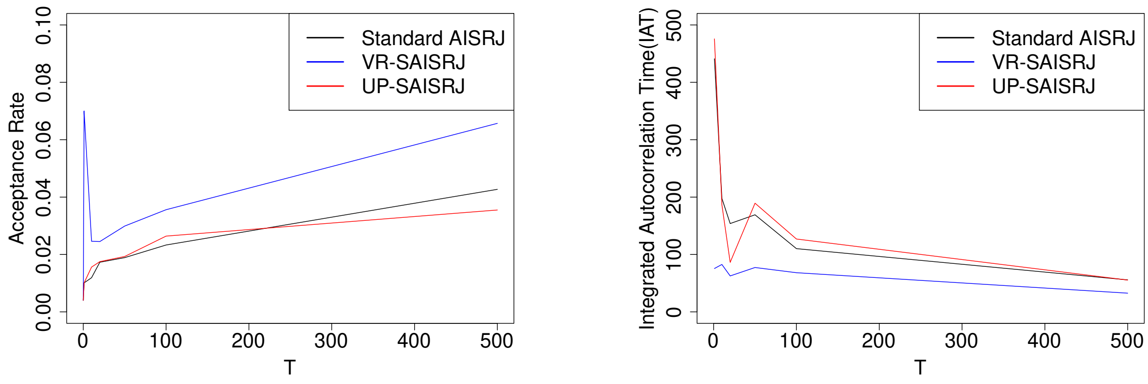
(b) Stochastic AISRJ ($T = 20, m = 20$)

(c) Stochastic AISRJ with Variance Reduction($T = 20, m = 20$)

(d) Stochastic AISRJ with Uncertainty Penalty ($T = 20, m = 20$)

Figure 7.5: Estimated probability mass of the 5 most probable models with model indices on the x axes and model probabilities on the y axes using Variance Reduction and Uncertainty Penalty methods. The posterior model probabilities of exact AISRJ are provided as a reference.

In Figure 7.5 we present probability mass functions of the posterior model distributions $\pi\left(m_k|x,\theta_k\right)$ using the variance reduction and bias correction methods. We further present in Figure 7.5a the model posterior $\pi\left(m_k|x,\theta_k\right)$ of the exact AISRJ algorithm with many intermediate steps shown in Figure 7.5a as a reference close to the ideal. It can be observed that variance reduction is only partially successful in reducing the posterior bias and achieving a posterior similar to exact AISRJ. However, the uncertainty penalty algorithm is much more successful, achieving results that are competitive with the exact AISRJ algorithm whilst using only 20 intermediate steps instead of 500 and also only using subsamples of 20% of the data. Moreover, we can compare the convergence of these algorithms with exact AISRJ.



(a) Acceptance Rate

(b) Integrated Autocorrelation Time

Figure 7.6: Acceptance rate and IAT by T for the exact AISRJ algorithm (Black); the stochastic AISRJ algorithm using only 20% of the data and variance reduction (Blue) and the stochastic AISRJ algorithm using only 20% of the data and the uncertainty penalty (Red)

We demonstrate that the uncertainty penalty results in convergence that is very similar to that of exact AISRJ. This is shown in Figure 7.6 where we also see that the variance reduction algorithm continues to mix much faster than the exact AISRJ algorithm. This is a result of the remaining bias that was not corrected by the variance reduction.

7.6 Using the SARJ algorithm to fit a non-stationary Gaussian process model to real data

We apply the algorithm to fit the complex Bayesian Spatially Clustered Coefficients (BSCC) model (Luo, Sang and Mallick 2021a) to real oceanic data. The BSCC model uses a minimum spanning tree T to partition the data into clusters for each coefficient. A priori the number of clusters is not known and so reversible jump methods are used to sample from models with varying numbers of clusters. The model described by Luo, Sang and Mallick (2021a) is a linear model but we apply the ideas to fit a non-stationary Gaussian process to the data. This model has challenging transitions and typically suffers from poor mixing. We demonstrate that our algorithm achieves much better mixing and copes well with the challenging model jumps.

We use spatial data describing the salinity of the Atlantic ocean downloaded from the National Oceanographic Data Center (https://www.nodc.noaa.gov/OC5/woa13/) and presented in Figure 7.7. We take data along a single line of longitude so that we model a two dimensional slice of the ocean. We then scale the latitude and depth so that we have spatial coordinates in a $[0, 1] \times [0, 1]$ square. The data are clearly nonstationary, with almost no variation in salinity at greater depths whilst having clear structure in shallower waters.

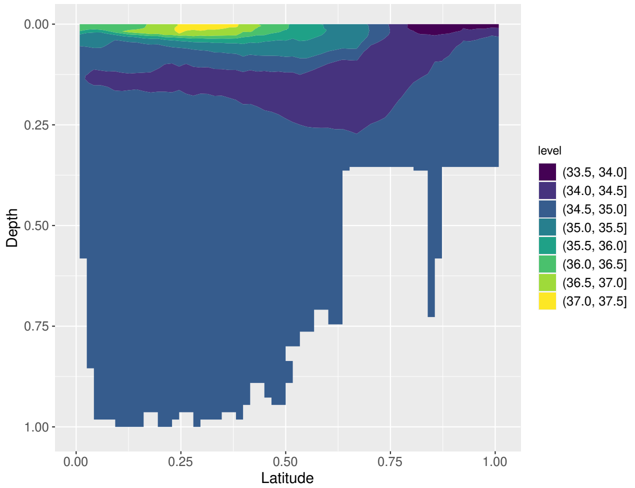


Figure 7.7: Latitude is shown along the x axis and Depth is shown on the y axis. The salinity of the water is presented as a colour gradient with higher salinity regions in yellow and lower salinity regions in purple.

Given output data $y \in \mathcal{Y}$; spatial data $z \in \mathcal{Z}$; a mean function μ; and a covariance function Σ with parameters σ^2,ϕ and τ^2; the model is a spatially clustered Gaussian process written in (7.2). We write $s(z_i)$ for the cluster label of the point z_i. Two data points within the same cluster share the same mean function which is constant within that cluster. Furthermore, the covariance matrix of the model is $\Sigma = R + \tau^2 I$ where I is the $n \times n$ identity matrix, τ^2 is a nugget parameter and the i, jth element of R is given by $R_{i,j} = \sigma^2_{s(z_i)} \exp\left(d\left(z_i, z_j\right)/\phi_{s(z_i)}\right)$ if $s\left(z_i\right) = s\left(z_j\right)$ and $R_{i,j} = 0$ otherwise. Thus each cluster has three parameters describing its mean and covariance functions and there is one additional global parameter describing the nugget variance.

$$y_k\left(\cdot\right) \sim \text{GP}\left(\mu\left(\cdot|\beta\right), \Sigma\left(\cdot, \cdot|\sigma^2_k, \phi_k, \tau^2\right)\right) \tag{7.2}$$

We assign a normal prior on the linear coefficients, β, with hyperparameters mean $u_0 = 0$ and variance $v_0^2 = 10^6$. We set independent inverse gamma priors on each of the covariance parameters $\sigma^2_k, \phi_k,$ and τ^2 which each have shape and rate hyperparameters $p = 0.5$ and $q = 0.5$ respectively. We further set a truncated geometric prior on the number of clusters such that

$$\pi\left(K = k\right) \propto \left(1 - c\right)^k, \qquad \text{for } k = 1, ..., n_S,\ 0 \leq c < 1 \tag{7.3}$$

This helps to regulate the model and prevent over-fitting. We use the subsampling based estimator of the gradient which is biased for the Gaussian Process model but has strong convergence guarantees for stochastic gradient methods (H. Chen et al. 2022).

For the reversible jump step we propose one of three possible moves. A birth step can occur by removing an edge from the minimum spanning tree, T. This splits one cluster into two new clusters. A death step can occur by adding a missing edge from the complete minimum spanning tree. This merges two clusters into a single cluster. The birth move is accepted with probability $\alpha = \min(1, r)$ such that

$$\begin{aligned} r =& \frac{\pi(T', \phi_t' | y_t)}{\pi(T, \phi_t | y_t)} \frac{q(T|T')}{q(T'|T)} q(\phi|\phi') \\ q(T'|T) =& q_S(K) \frac{1}{n_s - K} \\ q(T|T') =& q_M(K+1) \frac{1}{K} \end{aligned}$$

where q_S is the probability of proposing a split, q_M is the probability of proposing a merge, and K is the number of clusters. The death move is accepted with probability $\alpha = \min(1, r^{-1})$.

Finally an adjustment step can occur which updates the minimum spanning tree while conserving the current clusters. This enables the sampler to reach previously unreachable configurations of the model. This step is accepted with probability 1. The birth and death steps are proposed with probability 0.475 each and the adjustment step is proposed with probability 0.05. When a new cluster is made by the birth step we propose a new mean function parameter $\beta_i \sim N(35, 2)$ and new covariance function parameters σ_i^2 and ϕ_i from independent inverse gamma distributions with shape and rate parameters equal to 8.

We split the data into two halves of 2664 data points by randomly selecting half of the spatial locations. We use one half to fit the model and the other half as test data. To fit the model we use the SARJ algorithm with 10 intermediate steps and a batchsize of 400 which is approximately 15% of the training data. We run 10000 iterations of the algorithm and obtain predictive values for the test data at each iteration. We then take the MSE of the predictive errors and as shown in Figure 7.8 we find that this is very small everywhere. Even the largest MSEs are very small relative to the magnitude of the predicted values.

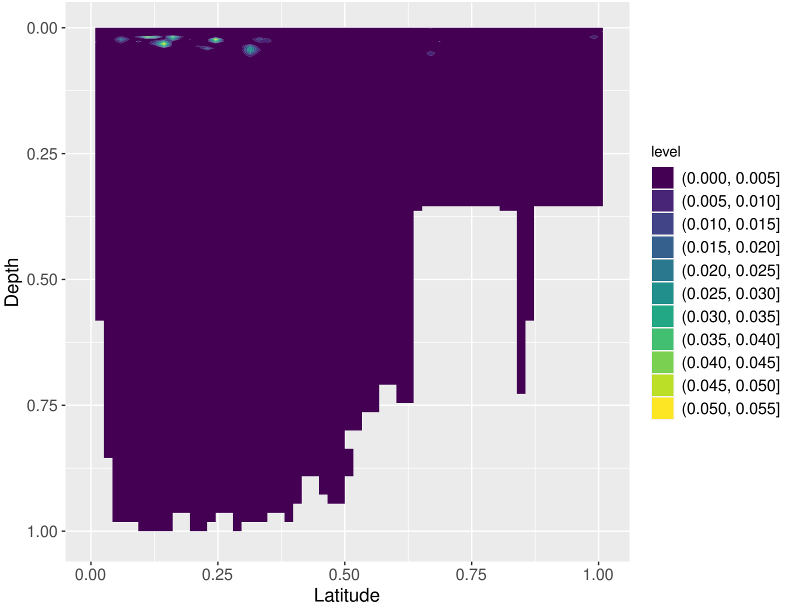
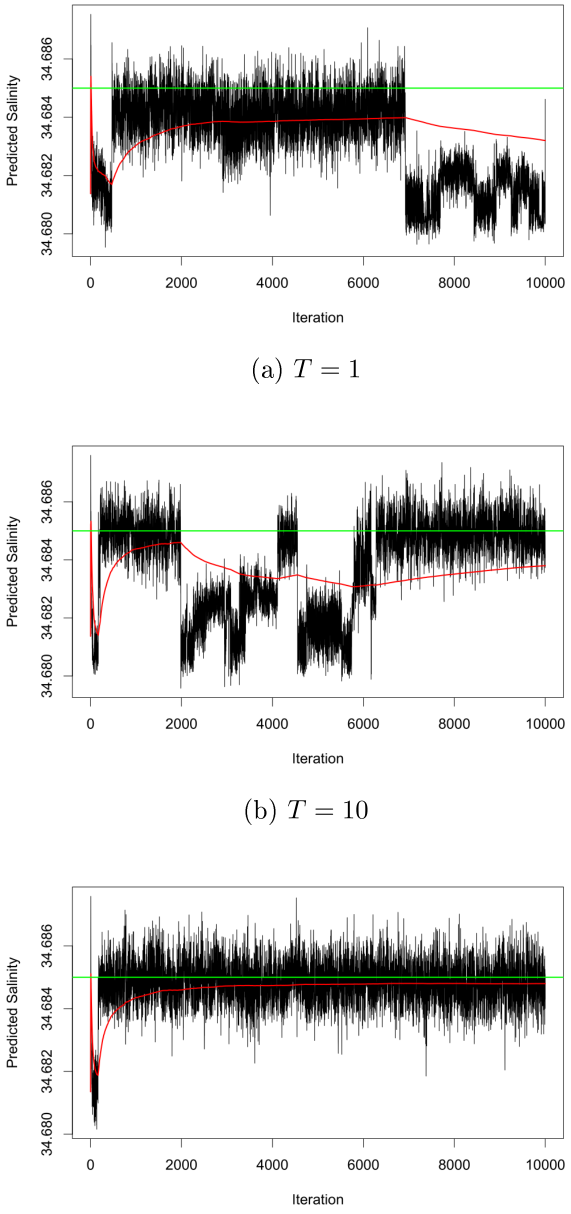


Figure 7.8: Mean Squared Error of the Predicted Values

We further consider the mixing of the algorithm by examining the predictive performance of the model at a single location. In Figure 7.9 we see that for a small number of intermediate steps the prediction is close to the true value but the mixing is poor. As the number of intermediate steps increases the mixing improves greatly as the sampler can more freely explore the model space. This is further evidenced by improvements in the integrated autocorrelation time which is 240.8, 91.5 and 50.7 for SARJ with $T = 1, 10$, and 100 intermediate steps respectively.



(a) $T = 1$

(b) $T = 10$

(c) $T = 100$

Figure 7.9: Predicted salinity for a typical test point using $1, 10$, and 100 intermediate densities respectively. The true salinity is shown in green and the averaged prediction is shown in red.

For the real data example we further see impressive computational savings. The SARJ algorithm was run in 5.667 hours, whereas the exact AISRJ algorithm took approximately two and a half weeks to run on the same computer to achieve nearly identical predictive accuracy.

Chapter 8

Conclusion

8.1 Summary

In this Part we proposed a stochastically annealed reversible jump(SARJ) algorithm. The new algorithm aimed to achieve the statistical efficiency of exact AISRJ whilst also achieving much greater computational efficiency for large datasets. We proved that the algorithm achieved these aims with theoretical guarantees and then further showed that the algorithm succeeds in practice with demonstrations of the algorithm being applied to synthetic and real datasets. The theoretical results provided guarantees about the convergence of the algorithm by showing that it converges to an ideal sampler. Furthermore, we present theoretical results showing that this convergence is improved by reducing the variance of the stochastic likelihood estimate with additional intermediate steps and/or subsamples. We continued development of the algorithm by exploring two mechanisms which can be used in tandem to reduce the variance and bias introduced by the stochastic estimator and improve the algorithm further. The numerical examples presented showed the theoretical convergence of the SARJ algorithm to the exact AISRJ algorithm with a known example. Furthermore, we demonstrated that both the variance reduction and bias reduction mechanisms reduce the variance of the stochastic likelihood estimate and consequently the bias in the model posterior estimates. These benchmark results demonstrate that our theoretical results are observed in practice. Finally the real data application to the non stationary Gaussian process model yielded extremely impressive computational savings with a runtime of less than six hours compared to the exact algorithm's runtime of two and a half weeks. This reduction in runtime came with little to no loss of predictive accuracy which further demonstrates the convergence properties of the SARJ algorithm in practice.

8.2 Future research aims

In the application of the SARJ algorithm to the real data application we found that the statistical efficiency of the algorithm could be improved by using a stratified subsampling scheme to ensure data were subsampled from each cluster of the model instead of naïvely subsampling with a simple random sampling scheme. Future work might examine different subsampling schemes and look for theoretical results regarding the impact of these schemes on the variance of the stochastic annealing importance weight. Another avenue for future work is the annealing schedule. Throughout this chapter we have exclusively used the geometric schedule given by Equation 6.6. Further work could be done to examine other annealing schedules which might outperform the geometric schedule in practice. Theoretical results for other schedules may be challenging and likely require direct proof of convergence from SARJ to the ideal sampler where they break the reversibility of AISRJ.

Part III

Spatially clustered Gaussian process regression

Chapter 9

Overview and context

9.1 Spatially clustered modelling

Given an input domain $\mathcal{X}$, spatial domain $\mathcal{S}$, and an output domain $\mathcal{Y}$, we aim to build a model to make inference about the relationship between inputs $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$ given a location $s \in \mathcal{S}$. Gaussian process regression is a popular and effective approach to making this inference.

The Gaussian process model is nonparametric and can be flexible, allowing it to model a diverse range of problems. However, the computational cost of the Gaussian process model can become prohibitively expensive when the dataset is large. Moreover, the model can be challenging to use if nonstationarity is present in the data.

Various methods exist to reduce the computational costs of Gaussian process regression including tapered covariance functions (Furrer, Genton and D. Nychka 2006; Kaufman, M. J. Schervish and D. W. Nychka 2008; Du, H. Zhang and Mandrekar 2009) and lower dimensional space process approximations (Wikle and Cressie 1999; Cressie and Johannesson 2008). Methods also exist to tackle non stationary problems such as by using nonstationary covariance functions (Paciorek and M. Schervish

2003). Alternatively spatial clustering can enable nonstationary Gaussian process modelling by splitting the space into homogenous regions. This simultaneously reduces the computational cost by introducing natural sparsity to the covariance model and models the nonstationarity in an easily interpretable manner. To introduce spatial clustering to the Gaussian process model we aim to identify a cluster structure in $\mathcal{S}$ such that two points s and s' which share a cluster $\mathcal{D}$ give the same relationship between x and y. Then the model makes inference about x and y given a cluster $\mathcal{D} \subset \mathcal{S}$.

There exist many methods in the literature which can be applied to find such clusters and model local relationships. These include binary treed methods which recursively split sections of the spatial domain into two parts. Such models include the Classification and Regression Tree (CART) model (Breiman and Ihaka 1984; Denison, Mallick and Smith 1998) popular in machine learning, the Bayesian Additive regression tree (Chipman, George and McCulloch 2010) and treed Gaussian processes (Gramacy and H. K. H. Lee 2008; B. Konomi et al. 2014). Binary treed methods can function well but suffer from a restrictive assumption that the spatial clusters must be rectangular in shape. An alternative approach to binary treed methods uses Voronoi tesselations to identify clusters (Knorr-Held and Raßer 2000). This approach defines region centers such that points are included in a region determined by their nearest centre. Whilst Voronoi tesselation based clustering relaxes the restrictive assumption of binary treed methods and allows for non-rectangular clusters, it is still required that the clusters must be convex and contiguous in space.

Recently, spatially varying coefficients models have been proposed which use minimum spanning trees (MST) to cluster the spatial domain (Luo, Sang and Mallick 2021a). These methods can identify contiguous clusters in space of any arbitary shape and massively relax the assumptions of the model relative to both binary treed and Voronoi tesselation based methods. Nonetheless, the Bayesian Spatially Clustered Coefficients (BSCC) model described in the literature is underdeveloped in terms of prediction and completely lacks any method by which the model can be extended to include new data. In this Part we introduce a method by which we can naturally extend BSCC models to make predictions to new data whilst extending the scope of the clustering method to include multifidelity Gaussian processes. By enabling prediction whilst extending the methodology to multifidelity Gaussian processes we vastly expand the scope of problems to which the BSCC methodology can be applied.

9.2 Multifidelity Gaussian process regression

Computer simulation experiments are becoming increasingly popular for performing experiments that are unfeasible to perform repeatedly otherwise. As computer experiments become more detailed and precise however, even the computer simulations can become prohibitively expensive. The Gaussian process model is a popular choice for computer model emulation. Computer model emulation involves fitting a cheaper to run statistical model to a limited set of computer simulator outputs in order to use the statistical model as a cheap emulator. Oftentimes there is not a singular computer simulator available but instead several, where some simulators are more detailed than others and hence both more accurate and more expensive to obtain simulations from. In these cases it is desirable to use outputs from all of the computer models to fit the statistical emulator. This can be done with a model framework called cokriging.

Traditional cokriging involves fitting hierarchical Gaussian processes which model the lowest fidelity computer model, and then the discrepancy between each model and the next lowest fidelity model. Consider data $y \in \mathcal{Y}$, covariates $x \in \mathcal{X}$, spatial coordinates $s \in S$, and fidelity levels $t = 1, ..., m$, this model is given in (9.1),(9.2), and (9.3)

$$y_t(\cdot) = \gamma_{t-1}(\cdot)\, y_{t-1}(\cdot) + \delta_t(\cdot), \qquad t = 2, ..., m \tag{9.1}$$

$$y_1(\cdot) \sim \mathrm{GP}\left(\mu_1(\cdot|\beta_1), \sigma_1^2 R(\cdot,\cdot|\phi_1)\right), \tag{9.2}$$

$$\delta_t(\cdot) \sim \mathrm{GP}\left(\mu_t(\cdot|\beta_t), \sigma_t^2 R(\cdot,\cdot|\phi_t)\right), \qquad t = 2, ..., m \tag{9.3}$$

where $\gamma_{t-1}(\cdot)$ is a scale discrepancy between y_{t-1} and y_t and $\delta_t(\cdot)$ is a location discrepancy that models local differences between y_{t-1} and y_t. For ease of presentation we consider the case of a constant scale discrepancy $\gamma_{t-1}(\cdot) = \gamma_{t-1}$. Moreover, μ_t is the mean function of the Gaussian process at fidelity t whilst σ_t^2 and ϕ_t are correlation function parameters for the correlation function R at fidelity level t. Many correlation functions can be used (Williams and Rasmussen 2006), for illustration we focus on the Matérn family of stationary correlation functions. In particular for the examples we use the square exponential correlation function given in (9.4)

$$R\left((x,s),(x',s')\,|\phi_t\right) = \exp\left(-\frac{1}{2}\left((x,s)-(x',s')\right)\mathrm{diag}\left(\phi_t\right)\left((x,s)-(x',s')\right)\right) \tag{9.4}$$

This statistical emulator can become expensive to fit when the size of the data grows large. Suppose the data y_t is of size n_t at each fidelity level t. Then the cost of fitting the model is typically dominated by the inversion of an $n_t \times n_t$ covariance matrix at each level of the model. Furthermore, data augmentation is required to fit the model when the model designs are not also hierarchically nested. For simplicity of illustration we focus here on nested designs and note that data augmentation can be applied to all of the methods described here and proposed later to solve non-nested problems.

9.3 Partial Parallel cokriging

By making appropriate additional assumptions about the covariance structure the model described in (9.1),(9.2), and (9.3) can be simplified further. In particular we assume the spatial covariance structure is independent of the covariate covariance structure. Then we can design the covariance function R, in an entirely separable manner composed of the spatial covariance function R_S and the covariate covariance function R_x as shown in (9.5)

$$R\left(\cdot,\cdot|\phi_t\right) = R_S\left(\cdot,\cdot|\rho_t\right) \otimes R_x\left(\cdot,\cdot|\phi_t\right) \tag{9.5}$$

We can now consider the dimension of the data separately in the spatial domain and in the input domain. Let n_s be the number of spatial locations s in the data and n_x be the number of covariate combinations in the data at fidelity level t. Then at fidelity level t the data y_t are of size $n_s n_x$ and inverting the matrix output of the kronecker product (9.5) reduces the bottleneck step from inverting an $n_s n_x \times n_s n_x$ to separately inverting both an $n_s \times n_s$ and an $n_x \times n_x$ matrix by the properties of the kronecker product. This reduces the complexity of the bottleneck from $O\left(n_t^3\right)$ to $O\left(n_s^3 + n_x^3\right)$ which can be considerably cheaper.

It can then further be shown that if all that is required from the emulator is the predictive mean and predictive variance then we can further assume that the spatial covariance matrix R_S can be replaced with the identity matrix (Gramacy and H. K. H. Lee 2008; P. Ma et al. 2019). That is we can assume that the computer outputs given the covariates are independent across space without loss of accuracy in the estimates of the predictive mean and variance. This further reduces the bottleneck as it is now only necessary to invert an $n_x \times n_x$ matrix which is massively cheaper than inverting the original $n_s n_x \times n_s n_x$ matrix. Unfortunately these computational gains come at a cost of restrictive assumptions that severely limit the usage of the model when we are interested in uncertainty quantification beyond the predictive mean and variance. In particular if we are interested in spatially infilling the computer model to improve the granularity of the output then the Partial Parallel (PP) cokriging model is considerably overfitted to the spatial locations with existing data and requires ad-hoc adjustments to make even sensible predictions that still lack uncertainty quantification. In the next Chapter we propose a method that retains the computational benefits of PP-cokriging whilst addressing its overfitting issues.

Chapter 10

Spatially clustered multifidelity Gaussian process

We propose a new cokriging model which relaxes the spatial covariance assumptions of the PP-cokriging model and does not require that we can replace R_S with the identity matrix. The proposed model retains much of the computational savings of PP-cokriging whilst enabling the emulator to model spatial covariance within groups of outputs. The new model includes the fully separable model using (9.5) with no clustering as a special case.

10.1 The model

10.1.1 The Statistical Model

To relax the assumptions of PP-cokriging we propose extending the Bayesian Spatially Clustered Coefficients (BSCC) model (Luo, Sang and Mallick 2021a) used in Chapter 9 to the multifidelity Gaussian process model. Consider data $y \in \mathcal{Y}$, covariates $x \in \mathcal{X}$, spatial coordinates $s \in S$, and fidelity levels $t = 1, ..., m$, and the cokriging model given in (9.1),(9.2), and (9.3) with a separable covariance function

(9.5). We introduce a connected graph $\mathcal{G}$ with vertices at the spatial points s connected by edges $e \in E$ that we obtain via Delauney triangulation (D.-T. Lee and Schachter 1980) and removing edges longer than some threshold υ_0. Delauney triangulation connects vertices in triangles with edges such that no vertex lies within the circumcircle of each triangle. We then obtain an initial minimum spanning tree T with edges $E_T \subset E$ from this graph by the application of Prim's algorithm (Prim 1957). Prim's algorithm builds a tree by starting at a vertex and choosing the edge with lowest weight that connects a new vertex to the tree. It does this iteratively until every vertex is connected and the resulting tree is a minimum spanning tree. This minimum spanning tree will be used to induce a partition on S.

Clustering can be achieved by adding and removing edges e from E_T such that each cluster is contained within a connected subgraph T_k of T. Moreover, T_k is itself a minimum spanning tree on the points s_k contained within the kth cluster. We assume that the spatial covariance function takes the form

$$R_s\left(s, s'|\phi\right) = \mathrm{I}\left(s = s'\right) \tag{10.1}$$

where $\mathrm{I}\left(s = s'\right)$ is an indicator function. To enable computation we assume that the covariance structure within each cluster can be modelled independently of the other clusters. To extend the method to the multifidelity setting we further assume that each cluster can be propogated to all of the fidelity levels such that each spatial point belongs to the same cluster regardless of the fidelity level being examined. Later this assumption will enable us to analytically intergrate out parameters to allow for more efficient model fitting. For a given cluster k, we can write the statistical model as

$$y_{k,t}(\cdot) = \gamma_{t-1} y_{k,t-1}(\cdot) + \delta_{k,t}(\cdot) \tag{10.2}$$

$$y_{k,1}(\cdot) \sim \mathrm{GP}\left(\mu_1(\cdot|\beta_{k,1}), \sigma_{k,1}^2 R(\cdot,\cdot|\phi_{k,1})\right) \tag{10.3}$$

$$\delta_{k,t}(\cdot) \sim \mathrm{GP}\left(\mu_t(\cdot|\beta_{k,t}), \sigma_{k,t}^2 R(\cdot,\cdot|\phi_{k,t})\right) \tag{10.4}$$

Furthermore, the covariance matrix R has a diagonal block structure. This can be exploited to reduce computational costs since the main bottleneck when fitting the model is the matrix inversion of R. In particular if $n = \sum n_k^3 = \sum \left(n_{s_k} \times n_x\right)^3$ is the sum of the cubed cluster sizes then the bottleneck for fitting the model is reduced to $O(n_S)$ which can be considerably cheaper than even the seperable model. This reduction of the computational complexity is especially large if the spatial size of the data n_s, dominates the size of the data in the input domain n_x, such that $n_s >> n_x$. It also clearly follows from (10.2), (10.3), and (10.4) that the fully separable model is a special case where every spatial point exists within the same cluster.

10.1.2 The Prior

We denote a minimum spanning tree (MST) T with weights w as $T = \mathrm{MST}(w)$ and specify a prior on the edge weights $w = \{w_{ij}\}_{(s_i,s_j)\in S}$ where w_{ij} is the weight of an edge between points s_i and s_j such that

$$w_{ij} \sim \mathrm{Pr}_{ij}(\cdot) \tag{10.5}$$

with $w_{ij} \in [0,1]$ without loss of generality. Consider a model with a set of clusters T_k. We can consider the edges of the underlying graph $\mathcal{G}$ as a union of two sets, the within cluster edges $e_c \in E_c$ which connect points that share a cluster and the between cluster edges $e_k \in E_k$ which need to be removed to produce the partition. To enable the formation of clusters T_k the minimum spanning tree T must contain edges from the within cluster edge set E_c such that each cluster can be contained within a minimum spanning subtree using only these edges. Then the complete minimum spanning tree T must include exactly k edges from the between cluster edges E_k. These edges connect the clusters to form the MST and induce the partition into clusters T_k when removed. Similarly we split the edge weights w into the within cluster edge weights w_c and between cluster edge weights w_k. Thus assigning priors to the edge weights such that the within cluster edge weights are smaller than the between cluster edge weights with high probability effects a prior which places large prior mass on spanning trees that admit the clusters T_k. If prior information about the clusters is not available then we can assign independent uniform priors to the edge weights to produce an uninformative prior on the clusters. The ability to admit an informative prior on the clusters is a major advantage for our model. Other clustering methods such as binary tree or Voronoi tesselation based methods do not have the capability to include prior information about the cluster structure.

We construct the minimum spanning tree on the graph $\mathcal{G}$ given edge weights w using Prim's algorithm, $T = \text{MST}(w)$. There are alternative algorithms for finding the MST such as Kruskal's algorithm (Kruskal 1956) and Dijkstra's algorithm (Dijkstra 1959). Since the MST is unique these algorithms will produce the same tree. However, as the number of spatial locations n_s becomes large, Prim's algorithm becomes considerably faster. It should additionally be noted that for any given spanning tree of $\mathcal{G}$ there exists edge weights such that Prim's algorithm produces that spaning tree and thus this prior on the edge weights constructs a prior on the entire space of spanning trees (Luo, Sang and Mallick 2021a). We complete the prior on the model space M, by assigning a prior on the number of clusters K, such that

$$\pi\left(K = k\right) \propto \left(1 - c\right)^k, \qquad \text{for } k = 1, ..., n_S,\ 0 \leq c < 1 \tag{10.6}$$

where c is a hyperparameter that penalises models with a large number of clusters.

This prior can be interpreted as a geometric distribution on the number of splits in the tree. Hence we can interpret $\pi\left(K=k\right)$ as the probability that we remove k edges before it no longer improves the model to remove more. As c becomes closer to one the penalisation on k is strong whereas when c becomes zero the penalty vanishes and the prior becomes uniform on the number of clusters. We assign this prior on K independently of the prior on w to construct a prior on the full model space.

To complete the prior we assign Normal-Inverse Gamma priors on the within model parameters of the statistical model $\beta_{k,t}\in\mathbb{R}^p, \gamma_{k,t-1}\in\mathbb{R}, \sigma^2_{k,t}\in(0,\infty)$ described in (10.2),(10.3), and (10.4) such that

$$\begin{aligned}\beta_{k,t},\gamma_{k,t-1}|\sigma^2_{k,t} \sim&\mathrm{N}\left(\left[b_t,g_{t-1}\right],\sigma^2_{k,t}\mathrm{diag}\left(B_t,G_{t-1}\right)\right) && (10.7)\\ \sigma^2_{k,t} \sim&\mathrm{IG}\left(\lambda_t,\chi_t\right) && (10.8)\end{aligned}$$

with hyperparameters $b_t, g_{t-1}, B_t, G_{t-1}, \lambda_t,$ and χ_t. These priors are conjugate in the case of a nested design. Furthermore, we assign a Gamma mixture prior on the covariance parameter $\phi_t\in(0,\infty)$ such that

$$\phi_{k,t}\sim 0.5\mathrm{G}\left(1,10\right)+0.5\mathrm{G}\left(5,2\right) \qquad (10.9)$$

for $t=1,...,m$ which places positive mass on both small and large values of $\phi_{k,t}$ as shown in Figure 10.1. This completes the model.

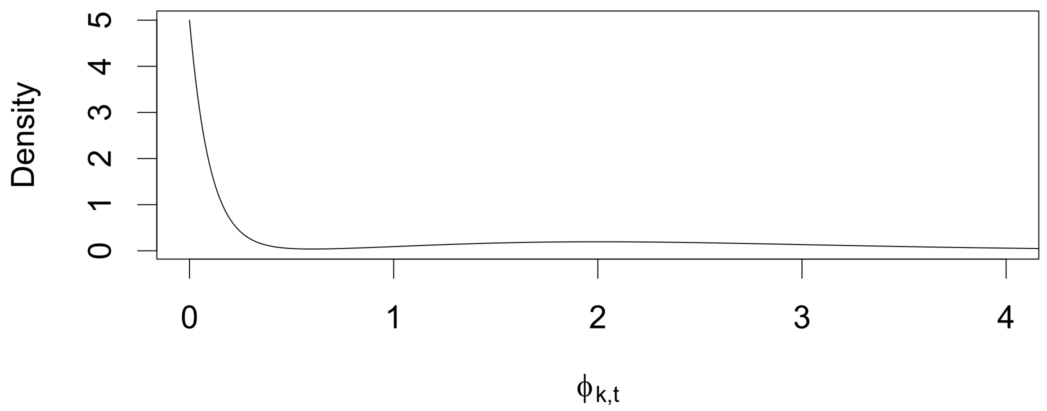


Figure 10.1: A plot of the prior density assigned to the parameter $\phi_{k,t}$

10.2 Computational strategy

To facilitate inference we use a reversible jump MCMC sampler since the posterior distribution is intractable. To sample each new edge weights for the Adjustment step we follow Algorithm 10.1 which samples new weights ordered such that the clusters are preserved.

```
Algorithm 10.1 Adjustment Step Edge Weight Sampling
Input: Spatial points $s_i$ and $s_j$
  1. Determine the clustering of $s_i$ and $s_j$
     (a) If $s_i$ and $s_j$ are in the same cluster then scale the bounds of the weight
         prior, $w'_{ij} \sim \Pr_{ij}(\cdot)$ to $\left[0, \frac{1}{2}\right]$
     (b) Else if $s_i$ and $s_j$ are in the different clusters then scale the bounds of the
         weight prior, $w'_{ij} \sim \Pr_{ij}(\cdot)$ to $\left[\frac{1}{2}, 1\right]$
  2. Sample a new weight from the scaled prior
Output: edge weight $w'_{ij}$
```

Following Algorithm 10.2 we sample from the marginal posterior using the RJ-MCMC moves.

```
Algorithm 10.2 MCMC method for SCGP
Input: current tree, current parameters
  1. With probability $p = (0.475, 0.475, 0.05)$ perform one of the split, merge, or
     adjust moves detailed below
  2. Update $\phi$ using Metropolis-Hastings
Output: updated tree, updated parameters
```

We could further employ the SARJ method described in Chapter 7 to more efficiently explore the model space if necessary by constructing intermediate densities which can be used to smooth the model transition and thus improve the efficiency of the reversible jump.

Consider a current state of the model given by a MST T, currently split into subtrees T_k where the subtree T_k connects the kth cluster. We have a set of edges E_c contained within the subtrees and separately a set of edges E_r that belong to T but are not contained within any subtree. Then at each iteration one of three reversible jump moves is performed with probability $p = (0.475, 0.475, 0.05)$.

Split *An existing cluster k is split into two new clusters k^* and k' by moving one randomly selected edge from the set E_c to the set E_r. This splits the subtree T_k into two subtrees $T_{k'}$ and T_{k^*}. One of the new subtrees selected at random retains the parameters ϕ_k whilst the other new cluster has new parameters ϕ' generated from the prior.*

The split move is accepted with probability $\alpha = \min\left(1, r\right)$ such that

$$\begin{aligned}
r =& \frac{\pi\left(T', \phi'_t | y_t\right)}{\pi\left(T, \phi_t | y_t\right)} \frac{q\left(T | T'\right)}{q\left(T' | T\right)} q\left(\phi | \phi'\right) \\
q\left(T' | T\right) =& q_S\left(K\right) \frac{1}{n_s - K} \\
q\left(T | T'\right) =& q_M\left(K + 1\right) \frac{1}{K}
\end{aligned}$$

where q_S is the probability of proposing a split and q_M is the probability of proposing a merge.

Merge *Two existing clusters k^* and k' are merged into one new cluster k by moving one randomly selected edge from the set E_r to the set E_c and consequently joining T_{k^*} and $T_{k'}$. The parameters of one of the preexisting clusters are randomly selected to be kept as the parameters for the new cluster ϕ_k*

The merge move is accepted with probability $\alpha = \min\left(1, r^{-1}\right)$ where r^{-1} is equal to the reciprocal of the acceptance ratio of the corresponding split.

Adjustment *The clustering remains the same and the edge weights w are updated by sampling a new set of edge weights w' such that the same partition is induced by the resulting minimum spanning tree T' using Algorithm 10.1 which is detailed above.*

The adjustment move is accepted with probability $\alpha = \min\left(1, t\right)$ such that

$$t = \frac{\pi\left(w'\right) q\left(w|w'\right)}{\pi\left(w\right) q\left(w'|w\right)}$$

where w and w' are the unscaled edge weights. In this thesis we make proposals independently using the prior and so the adjustment move is always accepted with probability $\alpha = 1$.

To improve the efficiency of the reversible jump we integrate out the model parameters $\beta_{k,t}, \gamma_{k,t-1}$, and $\sigma^2_{k,t}$. If inference about these parameters is required we can use a Gibb's sampler together with the conditional posteriors. We focus here on inference about the clustering as well as predictive inference and instead target the marginal posterior

$$\begin{aligned}\pi\left(T,\phi_t|y_t\right) \propto& \pi\left(T\right)\prod_{k=1}^{K}\pi\left(y_{k,1},\phi_{k,1}|T\right)\prod_{t=2}^{m}\pi\left(y_{k,t},\phi_{k,t}|y_{k,t-1},T\right), \qquad (10.10)\\ \pi\left(y_{k,t},\phi_{k,t}|y_{k,t-1},T\right) =& \pi\left(\phi_{k,t}\right)\frac{\left|\hat{A}_{k,t}\left(\phi_{k,t}\right)\right|^{\frac{1}{2}}}{\left|B_t\right|^{\frac{1}{2}}\left|G_t\right|^{\frac{1}{2}}}\frac{\chi_t^{\lambda_t}}{\pi^{\frac{n_{k,t}}{2}}}\frac{\Gamma\left(\lambda_t+\frac{n_{k,t}}{2}\right)}{\Gamma\left(\lambda_t\right)}\\ &\times\left(\mathrm{SSE}_{k,t}\left(\phi_{k,t}\right)\right)^{-\lambda_t-\frac{n_{k,t}}{2}} \qquad (10.11)\end{aligned}$$

where n_k is the number of data y contained within the kth cluster. Furthermore, writing $\xi = (x, s)$ for the spatial input pair we have that $\mathrm{SSE}_{k,t}\left(\phi_{k,t}\right) = \left(n_{k,t}+2\lambda_t-2\right)\sigma_{k,t}^2\left(\phi_{k,t}\right)$ and that

$$\begin{aligned}\hat{A}_{k,t}\left(\phi_{k,t}\right) =& \Big[L_t\left(\xi_{k,t};y_{t-1}\right)^T R_t^{-1}\left(\xi_{k,t},\xi_{k,t}|\phi_{k,t}\right)L_t\left(\xi_{k,t};y_{t-1}\right)\\ &+\mathrm{diag}\big(B_t^{-1},G_t^{-1}\big)\Big]^{-1} \qquad (10.12)\\ \hat{\alpha}_{k,t}\left(\phi_{k,t}\right) =& \hat{A}_{k,t}\left(\phi_{k,t}|s_k\right)\Big(L_t\left(\xi_{k,t};y_{t-1}\right)^T R_t^{-1}\left(\xi_{k,t},\xi_{k,t}|\phi_{k,t}\right)\\ &+\left[b_t^T B_t^{-1}, g_{t-1}^T G_t^{-1}\right]^T\Big) \qquad (10.13)\\ \hat{\sigma}_{k,t}^2\left(\phi_{k,t}\right) =& \frac{1}{2\lambda_t+n_k-2}\left(2\chi_t+y_t^T R_t^{-1}\left(\xi_{k,t},\xi_{k,t}|\phi_{k,t}\right)y_t+b_t^T B_t^{-1}b_t\right.\\ &\left.+g_{t-1}^T G_t^{-1}g_{t-1}-\hat{\alpha}_{k,t}^T\left(\phi_{k,t}\right)\hat{A}_{k,t}^{-1}\left(\phi_{k,t}\right)\hat{\alpha}_{k,t}\left(\phi_{k,t}\right)\right) \qquad (10.14)\end{aligned}$$

where $L_t\left(\xi_{k,t};y_{t-1}\right)$ is a matrix constructed by vertically stacking the design matrices $H_{k,t}$, of the inputs $x_{k,t}$ at the spatial locations s_k and appending the data y_{t-1} as a final column vector to the matrix.

10.3 Prediction

Prediction under our model at new spatial locations is not straightforward since there is uncertainty about the clustering of the new location. We propose a novel approach to make predictions at unseen points which makes several weak assumptions about how these points can be included. We first assume that each new point x' is connected to exactly one cluster k. This is necessary to enable predictive inference, and also to prevent the connection of two separate clusters. To maintain the consistency of the Bayesian model we must also assume that the existing clusters are preserved when the new points are added. That is that the inclusion of the new points would not cause some other points to be removed from their cluster and adjoined to another. This amounts to the assumption of independence of irrelevant alternatives, common in categorical predictive modelling and enables predictive inference without requiring us to refit the model.

Now consider the simple case of prediction for a single new location s'. The new point is added to a cluster by including one of the edges that connect the point to the graph. We write $c\left(s'\right) = k$ to denote the assignment of point s to cluster k. According to the model we have that each of these edges e, have uniform weights w, with which the minimum spanning tree can be constructed. Thus to include the new point whilst preserving the existing clusters we must include one of the connecting edges uniformly at random. Hereafter we omit the conditionality on y, s, x, ϕ, and T for brevity and write the posterior predictive distribution $\pi\left(y\left(s', x'\right)|y, s, x, \phi, T\right)$ as $\pi\left(y\left(\xi'\right)\right)$ and the conditional predictive distribution for each cluster $\pi\left(y\left(s', x'\right)|y, s, x, \phi, T, c\left(s'\right) = k\right)$ as $\pi\left(y\left(\xi'\right)|c\left(s'\right) = k\right)$. The posterior predictive distribution can be calculated by first calculating the conditional predictive distributions. We then combine these to find the full posterior with the probabilities of connecting the new point to each cluster such that

$$\pi\left(y\left(\xi'\right)\right) = \sum_{k=1}^{K} \pi\left(y\left(\xi'\right)|c\left(s'\right) = k\right)\pi\left(c\left(s'\right) = k\right) \qquad (10.15)$$

where $\pi\left(c\left(s'\right) = k\right)$ is the proportion of edges connecting s' to the kth cluster.

An example is shown in Figure 10.2 where the new point x can be connected to one of the points $\{4, 5, 6, 10, 13, 14\}$ which are close enough to have edges shorter than the longest edge threshold v_0 and correspond to three separate clusters. Hence we calculate the posterior conditional on assigning x to each of these three clusters and then find the full posterior summing each of these conditionals multiplied by one third.

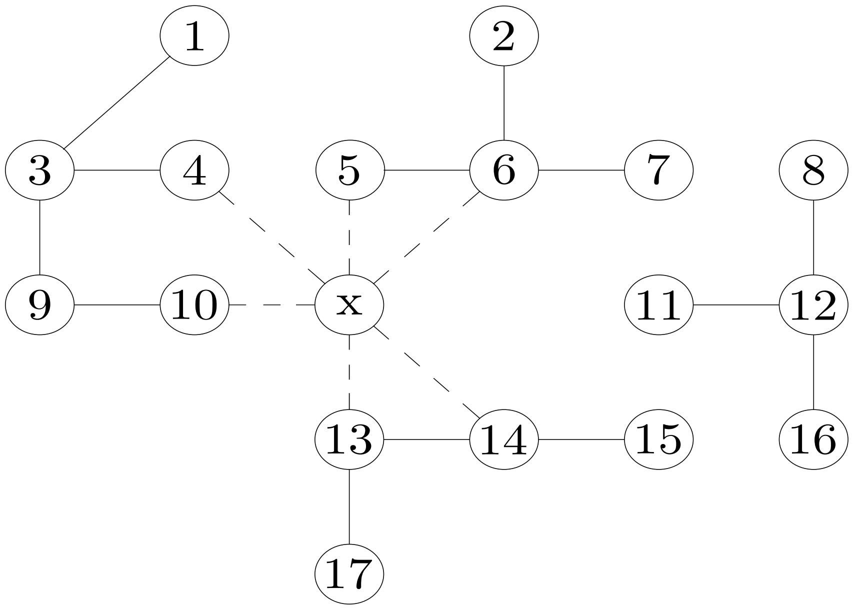


Figure 10.2: The simple case of adding a single new data point x to one of the existing clusters for prediction. One of the dotted edges must be added at random to connect x to a cluster.

Consider the case where we have two new data points. If these points are close then we must consider the inclusion of the edge that connects the two new points. To calculate the joint predictive distribution we must count the pairs of edges connecting the new points to each pair of clusters to calculate the connection probabilities and combine the conditional joint predictive distributions accordingly. Figure 10.3 presents an example where a pair of edges must be chosen. Clearly as the number of new points grows enumerating all of the possible sets of new edges becomes computationally infeasible.

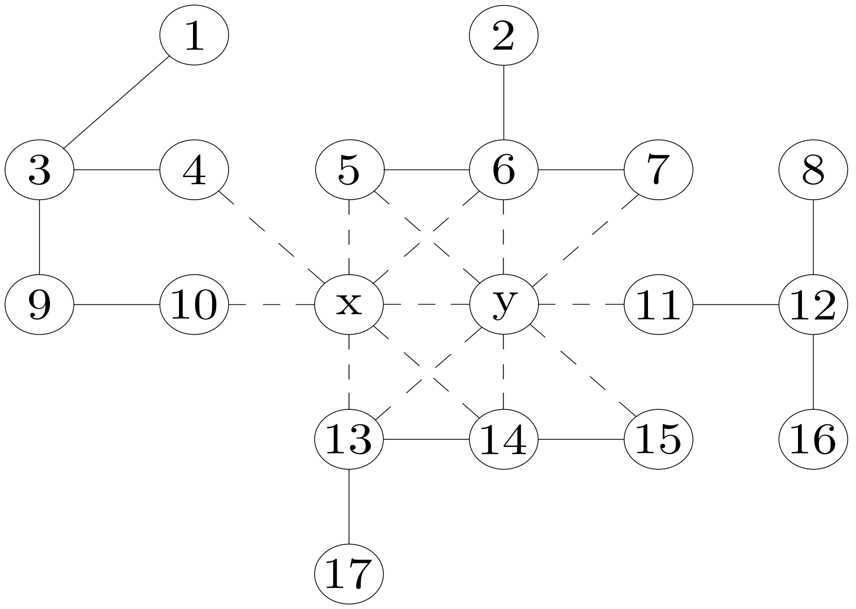


Figure 10.3: The case of adding a two new data points x and y to the existing clusters for prediction. Two of the dotted edges must be added at random such that x and y are both connected to a cluster.

To avoid the computational burden of enumerating the different clusterings we propose to sample from the posterior predictive distribution by sampling the edge weights. When fitting the model one of the updating steps updates the tree by sampling new weights for each of the edges using Algorithm 10.1.

For the adjustment step in Algorithm 10.1 we sample weights uniformly from $(0, 0.5]$ for edges connecting points that are within the same cluster and uniformly from $(0.5, 1]$ for edges connecting points which are in separate clusters we ensure that the resulting MST preserves the clusters. We propose an extension to Algorithm 10.1 by rescaling these weights to $\left(0, \frac{1}{3}\right]$ for edges within the same cluster, $\left(\frac{1}{3}, \frac{2}{3}\right]$ for edges between clusters and then sampling weights uniformly from $\left(\frac{2}{3}, 1\right]$ for the edges connecting the new points to the graph as shown in Algorithm 10.3.

```
Algorithm 10.3 Prediction Step Edge Weight Sampling
Input: Spatial points $s_i$ and $s_j$
  1. Determine the clustering of $s_i$ and $s_j$
    (a) If $s_i$ or $s_j$ is a new point then scale the bounds of the weight prior, $w'_{ij} \sim \Pr(\cdot)$ to $\left[\frac{2}{3}, 1\right]$
    (b) Else if $s_i$ and $s_j$ are in the same cluster then scale the bounds of the weight prior, $w'_{ij} \sim \Pr(\cdot)$ to $\left[0, \frac{1}{3}\right]$
    (c) Else if $s_i$ and $s_j$ are in the different clusters then scale the bounds of the weight prior, $w'_{ij} \sim \Pr(\cdot)$ to $\left[\frac{1}{3}, \frac{2}{3}\right]$
  2. Sample a new weight from the scaled prior
Output: edge weight $w'_{ij}$
```

By ordering the weights such that the lowest weights are contained within clusters and the highest weights connect the new points we similarly ensure that the new MST T', preserves the existing tree whilst adding the new points in such a way that obeys the assumptions described above. Thus by sampling L sets of edge weights (and hence the trees) in this way we can get Monte Carlo estimates of the connection probabilities to estimate

$$\pi\left(c\left(s'\right) = k\right) = \frac{\sum_{l=1}^{L} 1\left(c_l\left(s'\right) = k\right)}{L} \tag{10.16}$$

Moreover, the conditional posterior predictive distribution $\pi\left(y_t\left(\xi'\right)|c\left(s'\right) = k\right)$ at the location s' and fidelity level t is a Student-T process

$$y_t\left(s', x'\right)|y, s, x, \phi, T, k \sim \text{STP}\left(\mu^*_{k,t}\left(x'|\phi_{k,t}\right), \hat{\sigma}^2_{k,1} R^*_{k,1}\left(x', x''|\phi_{k,t}\right), 2\lambda_t + n_{k,t}\right) \tag{10.17}$$

where λ_t is one of the hyperparameters, and $n_{k,t}$ is the size of the data in cluster k at fidelity level t. Given observed spatial input pairs $\xi_{k,t}$ and unobserved spatial input pairs ξ', we have that $\mu^*_{k,t}$ and $R^*_{k,1}$ can be calculated as

$$\begin{aligned}\mu^*_{k,t}\left(x|\phi_{k,t}\right) =& L_t\left(\xi_{k,t};y_{t-1}\right)^T \hat{\alpha}_{k,t}\left(\phi_{k,t}\right) + R_t\left(\xi',\xi_{k,t}|\phi_{k,t}\right) R_t^{-1}\left(\xi_{k,t},\xi_{k,t}|\phi_{k,t}\right)\\ &\times \left[y_t\left(\xi_{k,t}\right) - L_t\left(\xi_{k,t};y_{t-1}\right)^T \hat{\alpha}_{k,t}\left(\phi_{k,t}\right)\right] \qquad (10.18)\end{aligned}$$

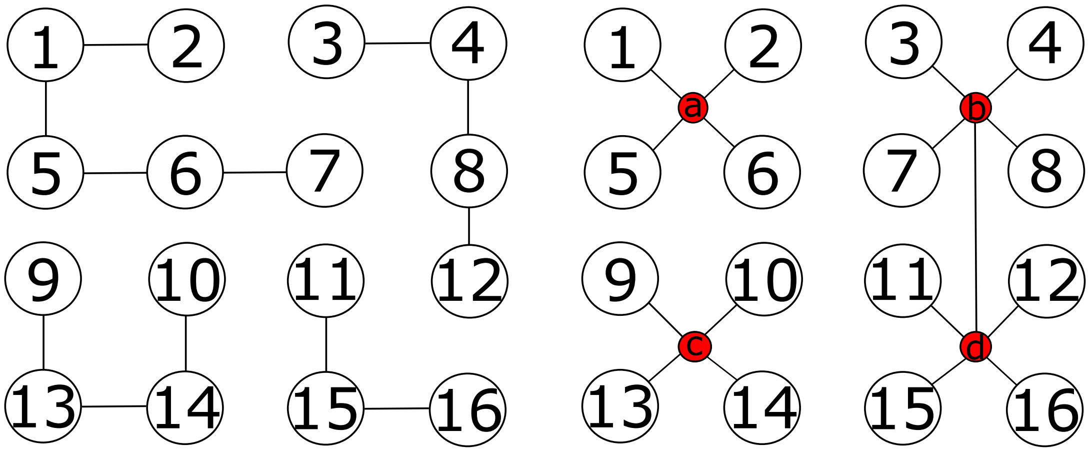
$$\begin{aligned}R^*_{k,1}\left(\xi',\xi''|\phi_{k,t}\right) =& R_t\left(\xi',\xi''|\phi_{k,t}\right) - R_t\left(\xi',\xi_{k,t}|\phi_{k,t}\right) R_t^{-1}\left(\xi_{k,t},\xi_{k,t}|\phi_{k,t}\right) R_t^T\left(\xi',\xi_{k,t}|\phi_{k,t}\right)\\ &+ D\left(\xi'|\phi_{k,t}\right)\hat{A}_{k,t} D^T\left(\xi''|\phi_{k,t}\right) \qquad (10.19)\end{aligned}$$

$$D\left(\xi'|\phi_{k,t}\right) = \left[L_t\left(\xi';y_{t-1}\right) - R_t\left(\xi',\xi_{k,t}|\phi_{k,t}\right) R_t^{-1}\left(\xi_{k,t},\xi_{k,t}|\phi_{k,t}\right) L_t\left(\xi_{k,t};y_{t-1}\right)\right] \qquad (10.20)$$

Together with the Monte Carlo estimates of the assignment probabilities $\pi\left(c\left(s'\right)=k\right)$ in Equation (10.16) we can construct joint posterior predictive densities as mixtures of Student-T processes.

10.3.1 The model using nodes

A new method has been proposed (Luo, Sang and Mallick 2021b) which extends the BSCC model using nodes. These nodes, N, are then used to construct the clusters by associating spatial points with the nearest node. Then the tree T can be built by constructing edges between the nodes instead of the spatial points. Thereafter the model can be fitted using Algorithm 10.2 where we consider the cluster membership $c\left(s\right)$ of a spatial point s to be the cluster membership $c\left(N\right)$ of the nearest node N. In Figure 10.4 we present an example of a clustering obtained using our method with the spatial points as vertices of the graph next to an example using the nodes method with four evenly spaced nodes as the vertices. We observe in Figure 10.4 that the placement of the nodes must be done carefully since they restrict the model space and force the clustering of their nearest spatial points. The restriction on the model space prevents the nodes method from achieving the configuration of clusters observed using the spatial points as vertices of the MST.



(a) Using the spatial points as the vertices of the MST

(b) Using the nodes as the vertices of the MST

Figure 10.4: The spatial points (white) are clustered using the Minimum spanning tree with spatial points as vertices (left) and the nodes (red) as vertices (right).

To enable prediction the clusters can be expanded to include new points by assigning the new points to the nearest node. This method was proposed to enable predictions to new spatial locations with the BSCC model which previously had no predictive method other than treating the locations of predictions as missing data and refitting the whole model. We note that the method using nodes can also be extended to the Gaussian process model in the multifidelity setting however we also propose an alternative predictive method that does not require the use of nodes. Furthermore, it should be noticed that when each point is assigned to its nearest node then the method is again a special case of the method described in Section 10.1; where cluster membership of groups of points is forced to match a priori. Thus it is desireable to avoid the use of nodes since the information required to make good node choices is not typically availabe a priori.

10.4 A toy example

We first demonstrate the clustering ability of the new algorithm on a simple example where the functions at each spatial location are planes such that

$$y_1(x,s) = c_1(s) + x_1 \tag{10.21}$$

$$y_2(x,s) = c_2(s) + x_1 \tag{10.22}$$

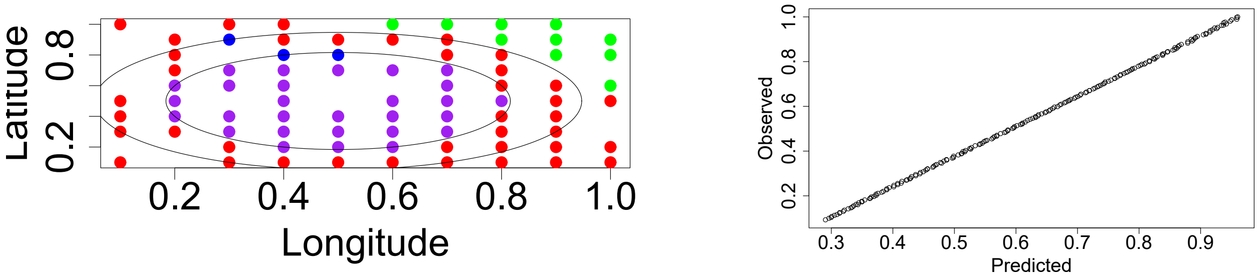
where $c_i(s)$ is a constant determined by the cluster which s belongs to, taking y_1 to be low fidelity and y_2 to be high fidelity. We generate simulated data at 100 evenly spaced spatial locations in the spatial domain $\mathcal{S} := [0,1] \times [0,1]$ partitioning $\mathcal{S}$ into three true clusters such that for a given spatial location $s = (s_1, s_2)$ and writing $s_c = (s_1 - 0.5)^2 + (s_2 - 0.5)^2$ we have that $c_1(s) = 1.2 + 4I(s_c > 0.2) + 2I(0.2 \geq s_c > 0.1)$ and $c_2(s) = 1.2 + 2.4I(s_c > 0.2) + 0.6I(0.2 \geq s_c > 0.1)$.

In the input domain, $\mathcal{X}$, we generate 200 inputs $x = (x_1, x_2)$ using Latin Hypercube Sampling (LHS) (McKay, Beckman and Conover 2000) and finally generate the outputs y_1 and y_2 for each combination of s and x using (10.21) and (10.22) respectively. At both the low and high fidelity levels we split the data into a training set of 80 randomly chosen spatial locations and reserve the remaining 20 as a validation set. We further remove the high fidelity outputs at another 20 spatial locations randomly chosen from the 80 training locations and add these outputs to the validation set.

We complete the Bayesian model by setting priors as in Section 10.1 with hyperparameters $c = 0.5, b_t = (0, ..., 0)$, $g_{t-1} = 0$, $B_t^{-1} = \text{diag}(0.01, ..., 0.01)$, $G_{t-1}^{-1} = 0.01$, and $\lambda_t = \chi_t = 0.2$. We sampe from the marginal posterior using 10000 iterations of Algorithm (10.2).

We observe from Figure 10.5a that the method correctly identifies the clusters. Furthermore, the method produced estimates with a predictive Root Mean Squared

Error (RMSE) of only 0.1 as it performs well with the simple example.



(a) The points indicate observed spatial locations at the corresponding longitude and latitude. The colours indicate which cluster the points belong to.

(b) The observed validation data on the y-axis against the predicted values on the x-axis.

Figure 10.5

For comparison we apply the PP-cokriging method described in Section 9.3 to demonstrate the capability of existing methodology. As expected the PP-cokriging method completely fails to cope with the problem and results in a predictive RMSE of 2.35, more than 20 times greater than the predictive RMSE of the new Spatially clustered Clustered GP method. At some spatial locations PP-cokriging fails so badly that the predictions are not even in the vicinity of the true output such as those observed in Figure 10.6 where the solid line indicating equality between prediction and observation is not near any of the prediction points.

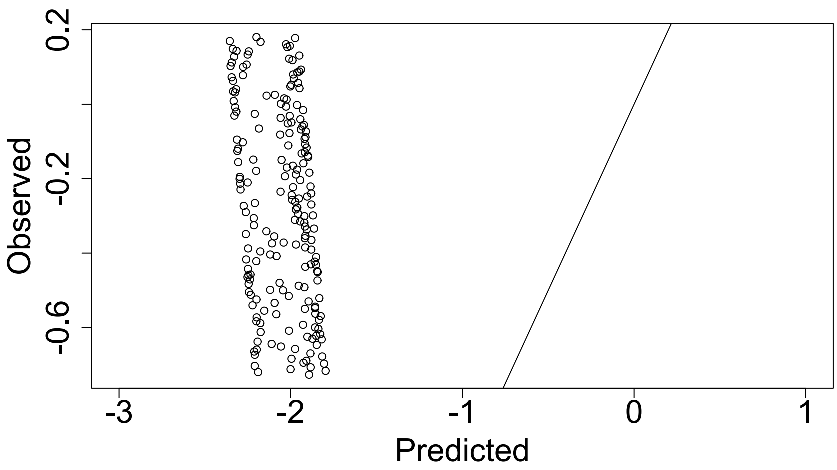


Figure 10.6: A plot of the observed data on the y-axis against the predicted values using PP-cokriging on the x-axis.

We now demonstrate the model on a more challenging example by adapting the numerical example in B. A. Konomi and Karagiannis (2021) by introducing a spatial component and clustering within the spatial domain. Consider functions

$$y_1(x,s) = 2x_1 \exp\left(-x_1^2 - x_2^2\right) + 0.5 \exp\left\{\sin\left(\left(0.9\left(\frac{x_1+2}{8} + 0.48\right)^{10}\right)\right)\right\} + c_1(s) \tag{10.23}$$

$$y_2(x,s) = 4x_1 \exp\left(-x_1^2 - x_2^2\right) + 0.2 \exp\left\{\sin\left(\left(0.9\left(\frac{x_1+2}{8} + 0.48\right)^{10}\right)\right)\right\} + c_2(s) \tag{10.24}$$

where $c_i(s)$ is a constant determined as in the simple example of (10.21) and (10.22). In the input domain we again generate 200 inputs $x = (x_1, x_2)$ using Latin Hypercube Sampling (LHS) and finally generate the outputs y_1 and y_2 for each combination of s and x using (10.23) and (10.24) respectively. We split the data into a training set of 80 randomly chosen spatial locations and reserve the remaining 20 as a validation set. We further remove the high fidelity outputs at another 20 spatial locations randomly chosen from the 80 training locations and add these outputs to the validation set. We complete the Bayesian model with hyperparameters $c = 0.5, b_t = (0, ..., 0)$, $g_{t-1} = 0$, $B_t^{-1} = \text{diag}(0.01, ..., 0.01)$, $G_{t-1}^{-1} = 0.01$, and $\lambda_t = \chi_t = 0.2$. We sample from the marginal posterior with 10000 iterations of Algorithm (10.2).

As can be observed in Figure 10.7 our method correctly identifies the clusters.

Moreover, the method performs well with a predictive RMSE of 0.28.

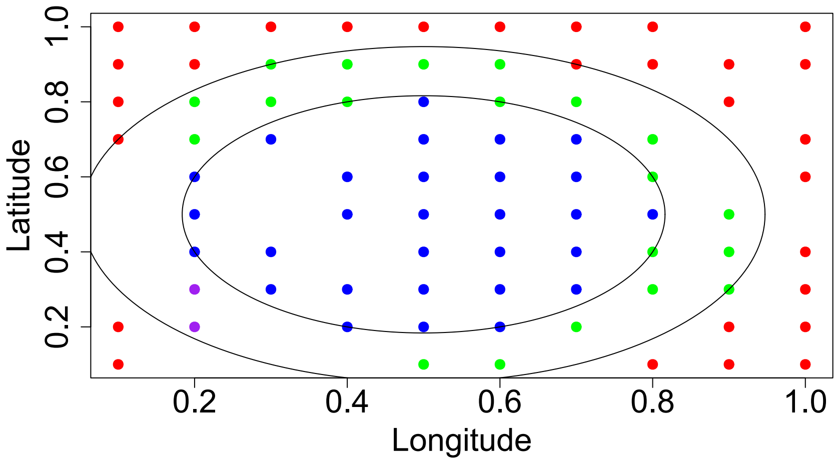
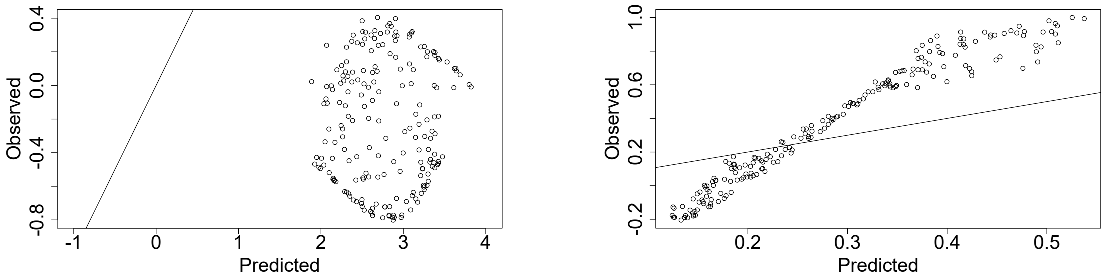


Figure 10.7: The points indicate observed spatial locations at the corresponding longitude and latitude. The colours indicate which cluster the points belong to. The black lines indicate the boundaries of the true clusters.

We observe in Figure 10.8a that the PP-cokriging model fails to emulate the simulator to a large degree with a predictive RMSE of 2.43 which is more than 8 times greater than the RMSE of SCGP. PP-cokriging fails because the method fits different parameters at every spatial location. This results in severe overfitting and very poor performance at new locations. In contrast we can observe in Figure 10.8b that SCGP offers a marked improvement upon existing methods. Despite the improvement there still remains a small portion of the bias that has not been eliminated. This is because Gaussian process models can suffer from an identifiability problem. Since the GP uses both a mean function and a covariance function to make predictions it is typically not possible to make direct inference about the mean function parameters. Ordinarily this does not impair prediction because observed output data can be used to calibrate predictions. When we wish to infill to unseen locations however, the new spatial locations have no output data that can be used and so the model cannot fully calibrate itself. We propose two potential remedies to this problem and demonstrate their efficacy.

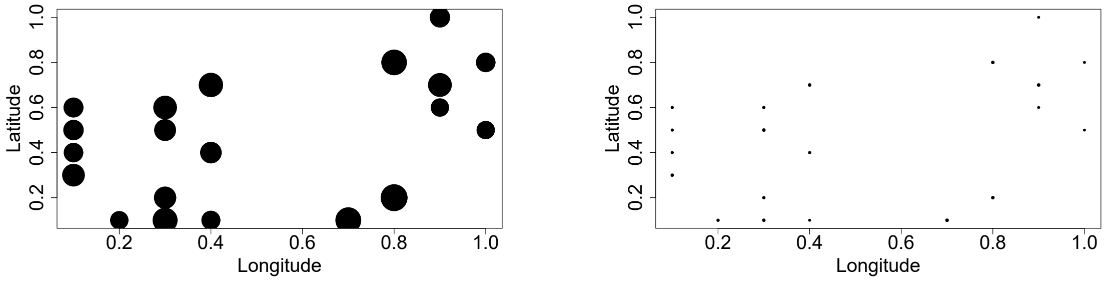


(a) The PP-cokriging prediction results

(b) The SCGP prediction results

Figure 10.8: Plots of the observed data on the y-axis against the predicted values on the x-axis. A straight line has been added to indicate where Observed = Predicted.

Figure 10.8 illustrates the difference in performance at a particular spatial location we can further see a large difference in the predictive error across all spatial locations. We observe in Figure 10.9 that our method performs well across the entire spatial domain whilst the existing PP-cokriging method performs much worse everywhere.



(a) The PP-cokriging prediction error across space. The overall predictive RMSE is 2.43

(b) The SCGP prediction error across space. The overall predictive RMSE is 0.28

Figure 10.9: Plots of the observed data on the y-axis against the predicted values on the x-axis. The size of the points indicates the magnitude of the RMSE at that spatial location.

One possible remedy to calibrate the predictive model can be used if the computational budget exists to supplement the data with new simulations at the unseen spatial locations. Existing methods such as PP-cokriging would require many new simulations and refitting of the model to estimate parameters for the unseen points. In contrast because of the clustering in our model we do not need to refit the model and can use the existing cluster parameters. Our model merely needs the new simulations to resolve the Gaussian Process' nonidentifiability problem. The aforementioned nonidentifiability issue can also be resolved with very few data and so the computational budget required to solve this problem can be minimised. Denoting the supplemental simulations as $\xi_{k,t}^{(s)}$ we can incorporate these into the conditional posterior predictive distribution (10.17) such that the parameters can be calculated as

$$\begin{aligned}\mu_{k,t}^{*}\left(\xi'|\phi_{k,t}\right) =& L_t\left(\xi_{k,t}^{(s)}; y_{t-1}\right)^T \hat{\alpha}_{k,t}\left(\phi_{k,t}\right) + R_t\left(\xi', \xi_{k,t}^{(s)}|\phi_{k,t}\right) R_t^{-1}\left(\xi_{k,t}^{(s)}, \xi_{k,t}^{(s)}|\phi_{k,t}\right) \\ & \times \left[y_t\left(\xi_{k,t}^{(s)}\right) - L_t\left(\xi_{k,t}^{(s)}; y_{t-1}\right)^T \hat{\alpha}_{k,t}\left(\phi_{k,t}\right)\right] \qquad (10.25)\\ R_{k,1}^{*}\left(\xi', \xi''|\phi_{k,t}\right) =& R_t\left(\xi', \xi''|\phi_{k,t}\right) - R_t\left(\xi', \xi_{k,t}^{(s)}|\phi_{k,t}\right) R_t^{-1}\left(\xi_{k,t}^{(s)}, \xi_{k,t}^{(s)}|\phi_{k,t}\right) R_t^T\left(\xi', \xi_{k,t}^{(s)}|\phi_{k,t}\right) \\ & + D\left(\xi'|\phi_{k,t}\right) \hat{A}_{k,t} D^T\left(\xi''|\phi_{k,t}\right) \qquad (10.26)\\ D\left(\xi'|\phi_{k,t}\right) =& \left[L_t\left(\xi'; y_{t-1}\right) - R_t\left(\xi', \xi_{k,t}^{(s)}|\phi_{k,t}\right) R_t^{-1}\left(\xi_{k,t}^{(s)}, \xi_{k,t}^{(s)}|\phi_{k,t}\right) L_t\left(\xi_{k,t}^{(s)}; y_{t-1}\right)\right] \qquad (10.27)\end{aligned}$$

Even just a pair of simulations can improve predictions considerably as can be seen in Figure 10.10 where a pair of supplemental inputs reduced the RMSE to 0.18.

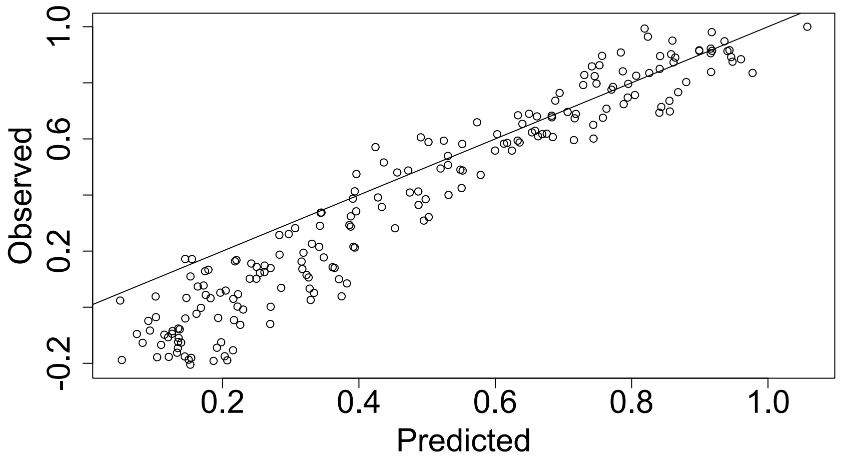


Figure 10.10: A plot of the observed data on the y-axis against the predicted values on the x-axis. A straight line has been added to indicate where Observed = Predicted.

With only a pair of simulations the predictive inference at unobserved spatial locations can be massively improved since these simulations provide calibration data. Further computational budget can be used to offer additional supplemental data and improve inference at unseen locations further. In this way the method can be used to infill the spatial mesh with minimal new computer simulations and without needing to refit the emulator.

An alternative solution when the remaining computational budget does not allow for new simulations is to interpolate the seen locations within the same cluster as the new location and use the interpolated data to calibrate the model. Similarly to the supplemental data we can denote the interpolated data as $\xi_{k,t}^{(s)}$ and incorporate these into the conditional posterior predictive distribution (10.17) using (10.25),(10.26), and (10.27). For this toy example interpolation between the spatial locations given the clusters can be done exactly and so because our method correctly identified the correct clusters this method outperforms the supplemental simulations and reduces the RMSE to only 0.13 performing well as can be observed in Figure 10.11.

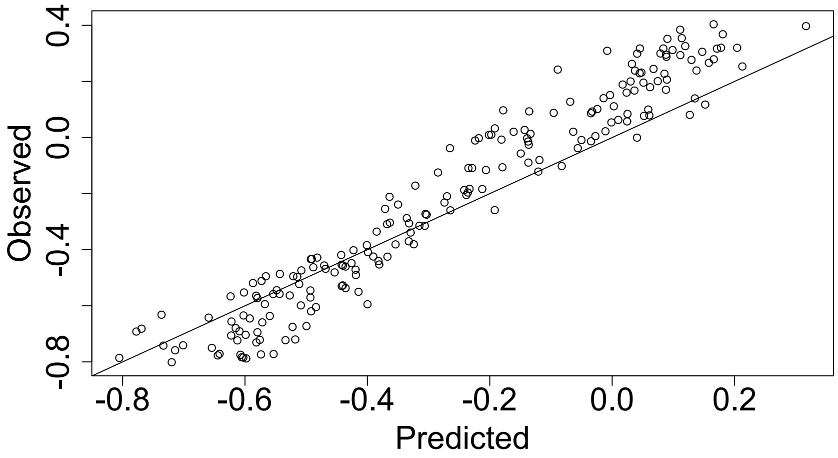


Figure 10.11: A plot of the observed data on the y-axis against the predicted values on the x-axis. A straight line has been added to indicate where Observed = Predicted.

In practice we would not expect interpolation to outperform supplemental data.

We additionally compare our method to the method using nodes as described in Subsection 10.3.1. We use 12 nodes scattered uniformly across the spatial domain to build the MST. We observe in Figure 10.12 that the model using nodes fails to identify the correct clusters. This happens because the nodes restrict the model space, and the true clustering no longer exists within an available model.

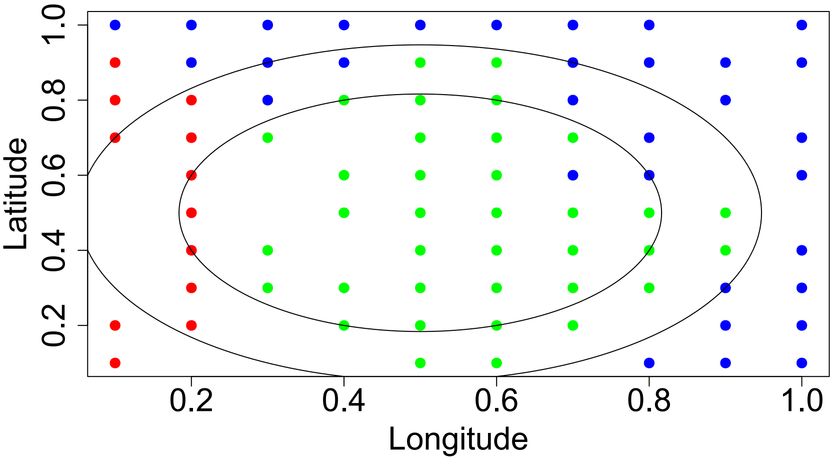
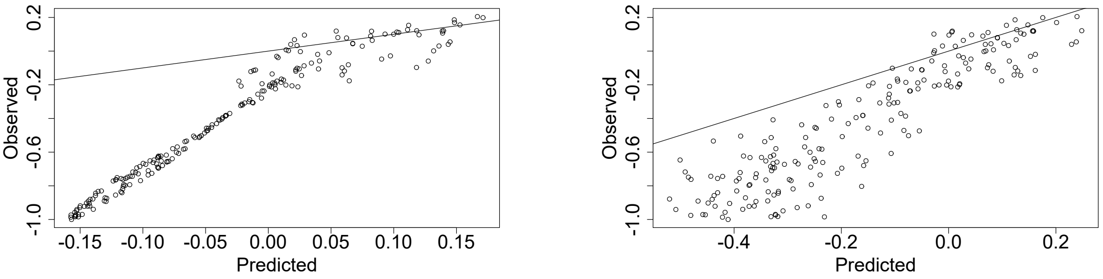


Figure 10.12: The points indicate observed spatial locations at the corresponding longitude and latitude. The colours indicate which cluster the points belong to. The black lines indicate the boundaries of the true clusters.

As we would expect the missclustering of points leads to poor predictive performance at missclustered spatial points such as can be observed in Figure 10.13. As with our SCGP method the nodes model can improve its predictive performance with a small amount of supplemental or interpolated data. Like the SCGP method these supplemental data can be incorporated without refitting the model.



(a) The nodes model prediction results

(b) The nodes model prediction results using supplemental data

Figure 10.13: A plot of the observed data on the y-axis against the predicted values on the x-axis. A straight line has been added to indicate where Observed = Predicted.

While we see improvement in the results following the incorporation of supplemental data the nodes model is still outperformed by our SCGP model.

Chapter 11

An application to Cape Coral stormsurge data

Computer modelling is an important component in the defence against stormsurge flooding caused by hurricanes. These models help to inform the required level of flood defences in areas that experience this stormsurge flooding. The physical simulation models required are extremely computationally expensive, a single run can take thousands of hours on a supercomputer. Instead of relying on the computer model to simulate the physical process it is common to replace the simulator with an emulator, typically in the form of a sophisticated Gaussian process model.

Standard Gaussian process models struggle with the nonstationarity that is common in these problems and so adaptations to the standard model have been proposed to directly address the nonstationarity. Modelling the complete covariance structure across both the spatial and input spaces together is normally impractical computationally, even for single fidelity models. As a compromise it is normal to completely separate the spatial and input covariance structures, and assume that these structures are independent of one another (Gu, Wang and Berger 2018). More recently PP-cokriging was proposed that makes the further assumption that the spatial covariance can be modelled by a diagonal matrix (P. Ma et al. 2019). This assumption massively reduces the computational work and comes with theoretical guarantees that if the quantities of interest are restricted to the predictive mean and the predictive variance at already observed spatial locations then the model gives near equivalent results to separable cokriging.

In this section we apply our new spatially clustered cokriging method described in Chapter 10. This relaxes the assumptions of PP-cokriging whilst retaining much of the computational benefits. In doing so we hope to broaden the scope of the outputs to include the predictive mean and variance at previously unobserved spatial locations. In particular this will enable us to upscale the mesh used at each fidelity of the model. We test our model using physical simulation data at 9284 spatial locations that form a mesh over the Cape Coral region of Florida. The input design is a latin hypercube in 5 dimensions of the input space with an additional input chosen at various even spacings for each combination of parameters. These low and high fidelity physical simulated data are obtained from the Advanced Circulation (ADCIRC) (Luettich and Westerink 2004) simulator and the ADCIRC+SWAN (J. Dietrich et al. 2011; J Casey Dietrich et al. 2012) simulators respectively. The ADCIRC simulator uses the finite elements method to numerically solve a series of equations for water levels over a grid. The ADCIRC+SWAN simulator couples the ADCIRC simulator with the Simulating WAves Near-shore (SWAN) (Booij, Ris and Holthuijsen 1999; Zijlema 2010) wave model which can simulate wave physics for near-shore waves that ride atop of stormsurge waves. We present the high fidelity output across all of the spatial locations with for one of the combinations of input parameters in Figure 11.1.

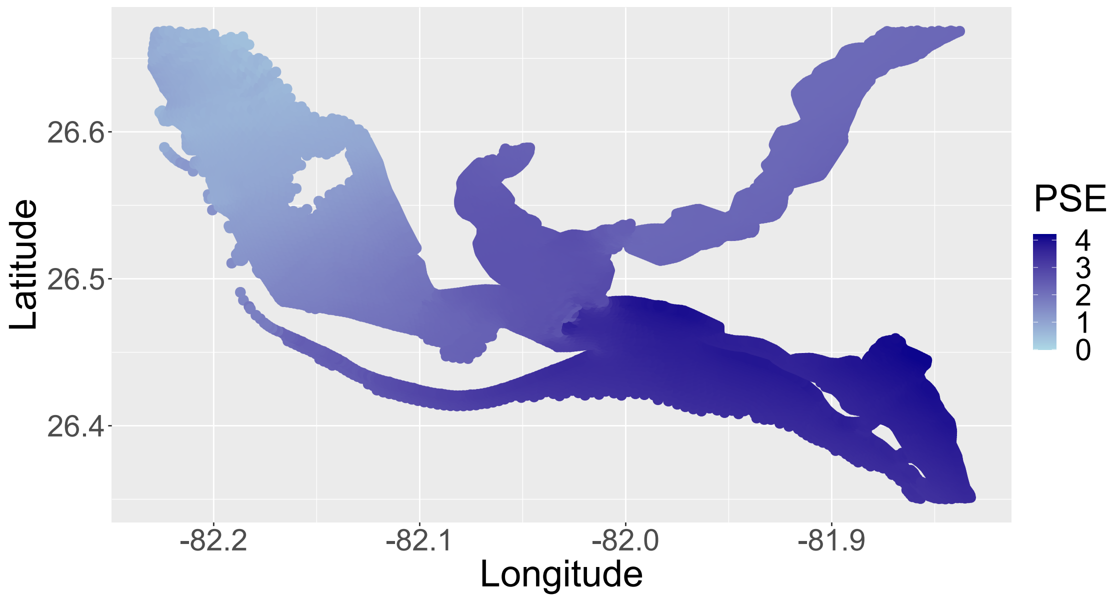


Figure 11.1: The output of the high fidelity simulator at an input, x_1. The axis show the longitude and latitude from the data whilst higher stormsurge is shown in darker blue.

To complete the Bayesian model we assign Normal-Inverse Gamma priors on the mean and variance parameters $\beta_t \in R, \gamma_t \in R$, and $\sigma_t^2 \in (0, \infty)$ at each fidelity level $t = 1, 2$ with hyperparameters $b_t = (0, ..., 0)$, $g_{t-1} = 0$, $B_t^{-1} = \text{diag}\,(0.01, ..., 0.01)$, $G_{t-1}^{-1} = 0.01$, and $\lambda_t = \chi_t = 0.2$. We further specify a Gamma mixture prior on the covariance parameter $\phi_t \in (0, \infty)$ such that

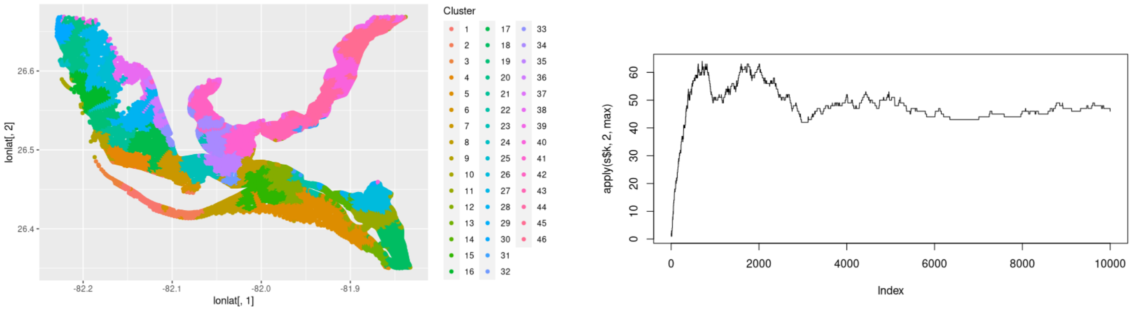
$$\phi_{k,t} \sim 0.5\text{G}\,(1, 10) + 0.5\text{G}\,(5, 2) \tag{11.1}$$

which allows for both small and large values of $\phi_{k,t}$. Finally we set a truncated geometric prior on the number of clusters k as described in (10.6) with $c = 0.5$.

We remove 284 spatial locations from the data to be used as a validation set. We further select another 2000 at the high fidelity level only which are also removed and added to the validation set. The remaining spatial locations are used as training data for each of the fitting and prediction methods. We will compare the methods using the predictive root mean square error (RMSE), average length of the 95% credible intervals (ALCI) on the predictive mean, the coverage of these same intervals (CVG), and the Nash-Sutcliffe model efficiency coefficient (NSME) (Nash and Sutcliffe 1970) calculated using the validation set. The Nash-Sutcliffe model efficiency coefficient is a goodness-of-fit statistic that measures how well the model predicts the output variable. The coefficient takes values less than or equal to one. A NSME coefficient of one indicates that the model perfectly predicts the output data, a NSME coefficient of zero indicates that the model predicts the output data as accurately as the sample mean would and values greater (less) than zero indicate that the model is a better (worse) predictor than the model. For those model fitted using reversible jump methods we additionally measure the acceptance rate of these jumps to evaluate how well the sampler is mixing.

11.1 Fitting the model

We first fit the model using standard reversible jump methodology. We use the prior for $\phi_{k,t}$ given in (11.1) as the dimension matching proposal and exploit the block diagonal structure of the model to reduce the necessary computations. We run the reversible jump sampler for 10000 iterations with a burn-in period of 1000 iterations and evaluate the model using the prediction criteria set out above. For comparison to the existing methodology we also fit the PP-cokriging model to the training data and attempt to make predictions to the test set with that also. In Figure 11.2a we observe the spatial clustering of the space at the final iteration of the sampling chain. There are lots of small to medium sized clusters with the smallest clusters typically near the boundary of the data. The large number of clusters indicates a large degree of nonstationarity in the spatial domain. Moreover, in Figure 11.2b we observe a trace plot of the number of clusters at each iteration and we can see that the method quickly splits the space into clusters and then mixes well around 40-50 clusters.



(a) The spatial clusters of the model. The axis show the longitude and latitude from the data whilst each cluster is represented by a different colour.

(b) A trace plot of the number of clusters.

Figure 11.2: Plots of the model output fitted with standard reversible jump

For the test metrics we find that our method performs well, with a RMSE that is very low relative to the magnitude of the output. Furthermore, the credible intervals perform exceptionally well, with coverage very close to the ideal 95%. In contrast PP-cokriging performs abysmally when applied to new spatial locations as we would expect. Both sets of performance metrics can be found in the Table 11.1.

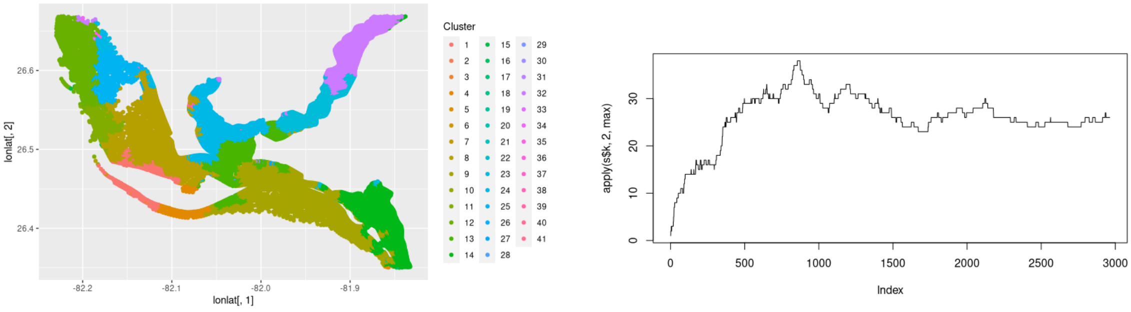
	PP-Cokriging	Spatially Clustered cokriging
RMSE	6.09	0.044
ALCI	202.3	0.172
CVG	1	0.94
NSME	-28	0.998
Runtime(hours)	11	14
RJ acceptance rate	NA	0.017

Table 11.1: The performance metrics for the PP-cokriging model and our Spatially Clustered cokriging model.

In Table 11.1 we see a vast difference between our proposed method and the PP-cokriging method. The predictive metrics RMSE and NSME illustrate an extremely large difference in accuracy whilst the uncertainty quantification also shows remarkable improvements both in the size of the credible intervals and in their calibration. We can observe that our new method Spatially Clustered cokriging has successfully broadened the scope of Stormsurge emulation and could be used to upscale the emulation mesh in this and other similar problems. This is a vast improvement over the existing methodology which is even magnitudes worse than just taking the data mean with a Nash-Sutcliffe model efficiency coefficient of -28.

11.2 Fitting the model with SARJ

We can see from Table 11.1 that the reversible jump acceptance rate is very low at only 1.7%. We will fit our model using the Stochastically Annealed Reversible Jump (SARJ) method from Chapter 8. With 10 intermediate steps each iteration of the algorithm takes longer to complete but proposes jumps that are much less likely to be rejected. Hence we run only 3000 iterations of the sampling algorithm with a burn-in period of 500 iterations and find that it converges much more quickly than standard reversible jump. Furthermore, when we compare the clustering of the SARJ fitted model in Figure 11.3a to the clustering of the standard RJ fitted model in Figure 11.2a we can see a smaller number of larger clusters with many small clusters around the periphary. This suggests the improved mixing of the SARJ method enabled the merging of larger clusters in a way that is difficult to achieve with standard RJ.



(a) The spatial clusters determined using SARJ. The axis show the longitude and latitude from the data whilst each cluster is represented by a different colour.

(b) A trace plot of the number of clusters.

Figure 11.3: Plots of the model output fitted with SARJ.

When we compare the test metrics to those using reversible jump in Table 11.2

we can observe similar predictive performance with a reversible jump acceptance rate that is almost 4 times greater than standard reversible jump.

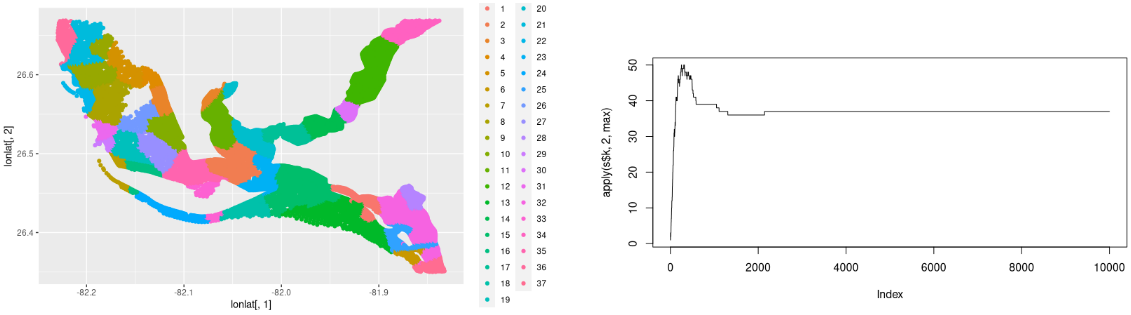
	Spatially Clustered GP (RJ)	Spatially Clustered GP (SARJ)	Nodes Clustered GP
RMSE	0.044	0.044	0.044
ALCI	0.172	0.136	0.133
CVG	0.94	0.89	0.88
NSME	0.998	0.998	0.998
Runtime (hours)	14	17	12
RJ acceptance rate	0.017	0.065	<0.001

Table 11.2: We observe that both our method and the alternative nodes based method have strong predictive performance based on the RMSE and NSME metrics shown. The methods differ in the RJ acceptance rate where the nodes based method fails to mix properly whilst the SARJ method improves upon the mixing rate of standard RJ.

We observe a decrease in the 95% CI coverage that we would expect to rectify with a larger computational budget. It should be noted that both of these methods would benefit greatly from parallelisation which could be used to more efficiently use the existing budget to close that gap.

11.3 Fitting the model with nodes

We additionally fit the nodes model of Subsection 10.3.1 to the data. The locations of the nodes can be difficult to assign since we have no information about good node locations a priori. We choose the nodes such that they are uniformly distributed throughout an alpha-hull that encloses the data. The alpha-hull constructs a tight non-convex boundary around the data so by choosing the nodes within this boundary we can ensure that the nodes are chosen such that there are data near each node. This ensures that the nodes both have good coverage and are contained within the data space such that there are no ‘dead’ nodes with no nearby points. We place 100 nodes in this way and then fit the model using standard reversible jump. We observe in Figure 11.4a that almost all of the clusters are medium to large even at the edges and have much smoother boundaries between clusters as a result of the nearest node clustering. We can also see from Figure 11.4b that while the nodes method splits the space into clusters very quickly it offers very poor mixing once the clusters are determined. This is likely to be a result of the inflexibility in spatial clustering within each node's space. Since two points which share a nearest node must be clustered together the nodes method is essentially a restricted version of the spatially clustered GP method and we would expect the performance to nearly match as the number of nodes approaches the number of spatial locations.



(a) The spatial clusters determined using the nodes. The axis show the longitude and latitude from the data whilst each cluster is represented by a different colour.

(b) A trace plot of the number of clusters.

Figure 11.4: Plots of the nodes based model output fitted with standard reversible jump

In Table 11.2 we compare the performance metrics of the nodes based method to the two spatially clustered models. We observe that in terms of RMSE the three models are near identical whilst the quality of uncertainty quantification varies.

We can see that the uncertainty quantification of the nodes method mostly matches the results of the SCGP model. Unlike the SCGP fitted model however, we would not expect the uncertainty quantification of the nodes to improve with additional computational budget because the sampler effectively stopped mixing once it found a near optimal node configuration. This trapping phenomenon is a frequent occurence for this method and occurs because the nodes effectively constrain the model space. Two spatial locations which share a nearest node must belong to the same cluster and so the sampler cannot explore models where those two locations are split. Moreover, the sampler cannot even use these models as a ‘bridge’ to escape local traps. We would expect to improve the method by introducing additional nodes and re-running the algorithm from scratch.

Chapter 12

Conclusion

12.1 Summary

We proposed a new spatially clustered Gaussian process model for handling nonstationarity. The new model aims to cope with a large degree of spatial nonstationarity whilst not inflicting the computational burden of a fully separable Gaussian process model. We further introduced a predictive process to the model that can be easily extended to other spatially clustered models. We showed that the model performs well, successfully modelling spatial nonstationarity in both a numerical example and in an application to stormsurge data from the Cape Coral region of Florida. Moreover, we demonstrated that the new model massively outperforms the existing PP-cokriging method when it is desired to predict to unobserved spatial locations. We further explored the model by using the SARJ algorithm that we proposed in Chapter 8 and found that this yielded much improved mixing quality compared to the standard reversible jump. Additionally we compared our method to an alternative nodes based method that has been proposed and found that our method lends itself to much better reversible jump mixing whereas the nodes based methodology suffers from severe local trapping problems when fitted with reversible jump.

12.2 Future research aims

Currently the model allows for nonstationarity in the spatial domain but assumes stationarity in the the input domain. The next steps to develop the model could involve introducing a second clustering process to this domain so that nonstationarity in the data could be modelled separably in both the spatial and input domains. This extension may introduce challenging non-identifiability issues and will not be straightforward. Moreover, SARJ and other sophisticated sampling schemes will be necessary to fit such a model since the mixing of two cluster structures simultaneously will be both difficult and slow.

Appendix A

Proofs of ABC results

A.1 Results of Chapter 2

Derivation of (2.4)

$$\begin{aligned}
\pi_\varepsilon(y) &= \int \pi(\theta)\, p(u|\theta)\, g_\varepsilon(u)\, \mathrm{d}u\mathrm{d}\theta \\
&= \int \pi(\theta)\, p(u|\theta)\, g_\varepsilon(u)\, \frac{q(\theta|\theta^*)\, p(u|\theta)}{q(\theta|\theta^*)\, p(u|\theta)} \mathrm{d}u\mathrm{d}\theta \\
&= \int \frac{\pi(\theta)\, g_\varepsilon(u)}{q(\theta|\theta^*)} q(\theta|\theta^*)\, p(u|\theta)\, \mathrm{d}u\mathrm{d}\theta \\
&\approx \frac{1}{N}\sum_{i=1}^{N} \frac{\pi(\theta_i)\, g_\varepsilon(u_i)}{q(\theta_i|\theta^*)} \quad \textit{where} \quad \theta_i \sim q(\theta_i|\theta^*)\,, u_i \sim p(u_i|\theta_i)
\end{aligned}$$

Proposition 2.4.2. *Consider the subregion weights $\varpi_j/\hat{w}_j$ calculated under the SAMC-ABC framework. We have that the $\hat{w}_j$ are the estimated normalising constants within each subregion E_j and that the subregions E_j form a partition on the parameter space. Then it follows that the ABC marginal likelihood $\pi_\varepsilon^{(g)}(y)$ can be calculated as:*

$$\pi_\varepsilon^{(g)}(y) = \sum_{j=1}^{m} C\hat{w}_j$$

Setting $\sum_{j=1}^{m} C\hat{w}_j$ to be equal to an alternative estimate such as (2.4) gives us the constraint required to determine an estimate of C.

Proof. Using the SAMCABC algorithm we can easily estimate the normalizing constant of the ABC posterior. Consider that $C\hat{w}_j = \int\int_{E_j} \pi(\theta) f(x|\theta) g_\varepsilon(||y-x||)\,\mathrm{d}x\mathrm{d}\theta$ and that the subregions E_j are a partition on the sample space. Then we can write the ABC normalizing constant $\pi_\varepsilon^{(g)}(y)$ as:

$$\begin{aligned}\pi_\varepsilon^{(g)}(y) =& \int_\Theta\int_\mathcal{X} \pi(\theta) f(x|\theta) g_\varepsilon(||y-x||)\,\mathrm{d}x\mathrm{d}\theta \\ =&\sum_{j=1}^{m}\int\int_{E_j} \pi(\theta) f(x|\theta) g_\varepsilon(||y-x||)\,\mathrm{d}x\mathrm{d}\theta \\ =&\sum_{j=1}^{m} C\hat{w}_i\end{aligned} \tag{A.1}$$

□

Proposition 2.4.3. *Consider the SAMCABC adaptive kernel*

$$h_\varepsilon(u) \propto \sum_{j=1}^{m} \frac{\varpi_j}{w_j} g_\varepsilon(u) I_{E_j}(u)$$

where $\frac{\varpi_j}{w_j}$ is the SAMC weight for subregion E_j and $g_\varepsilon(u)$ is the ABC kernel. The SAMCABC marginal likelihood $\pi_\varepsilon^{(h)}(y)$ can be calculated as

$$\pi_\varepsilon^{(h)}(y) = \left(\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\int_{E_j} g_\varepsilon(u)\,du\right)^{-1}$$

Proof. Consider the SAMCABC posterior:

$$\pi_{SAMCABC}(\theta, u|x) = \frac{\sum_{j=1}^{m}\pi(\theta) f(u|\theta)\frac{\varpi_j}{\hat{w}_j Z} g_\varepsilon(u)\,\mathrm{I}((\theta,u)\in E_j)}{\pi_\varepsilon^{(h)}(y)}$$

where $Z = \int \sum_{j=1}^{m}\frac{\varpi_j}{w_j} g_\varepsilon(u) I(u\in E_j)\,\mathrm{d}u$ is a normalizing constant which ensures that $k_\varepsilon(u) = \sum_{j=1}^{m}\frac{\varpi_j}{w_j Z} g_\varepsilon(||y-x||)\,\mathrm{I}((\theta,x)\in E_j)$ is a valid ABC kernel and $\pi_\varepsilon^{(h)}(y) = \int_\Theta\int_\mathcal{U}\sum_{j=1}^{m}\pi(\theta) f(u|\theta)\frac{\varpi_j}{w_j Z} g_\varepsilon(u)\,\mathrm{I}((\theta,u)\in E_j)\mathrm{d}u\mathrm{d}\theta$ is the SAMCABC marginal likelihood. Then we have:

$$\begin{aligned}
f_{SAMCABC}\left(y\right)=&\int_{\Theta}\int_{\mathcal{X}}\sum_{j=1}^{m}\pi\left(\theta\right)f\left(x|\theta\right)k_{\varepsilon}\left(||y-x||\right)\mathrm{I}((\theta,x)\in E_j)\mathrm{d}x\mathrm{d}\theta\\
=&\sum_{j=1}^{m}\int\int_{E_j}\pi\left(\theta\right)f\left(x|\theta\right)k_{\varepsilon}\left(||y-x||\right)\mathrm{d}x\mathrm{d}\theta\\
=&\sum_{j=1}^{m}\frac{\varpi_j}{w_jZ}\int\int_{E_j}\pi\left(\theta\right)f\left(x|\theta\right)g_{\varepsilon}\left(||y-x||\right)\mathrm{d}x\mathrm{d}\theta\\
=&\sum_{j=1}^{m}\frac{\varpi_j}{w_jZ}w_j\\
=&\frac{\sum_{j=1}^{m}\varpi_j}{Z}
\end{aligned}\tag{A.2}$$

Furthermore we have that $\sum_{j=1}^{m}\varpi_j=1$ and then the normalizing constant for the SAMC-ABC posterior is:

$$\begin{aligned}
\pi_{\varepsilon}^{(h)}\left(y\right)=&Z^{-1}\\
=&\left(\int\sum_{j=1}^{m}\frac{\varpi_j}{w_j}g_{\varepsilon}\left(u\right)I\left(u\in E_j\right)\mathrm{d}u\right)^{-1}\\
=&\left(\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\int_{E_j}g_{\varepsilon}\left(u\right)\mathrm{d}u\right)^{-1}
\end{aligned}\tag{A.3}$$

and so we can use the estimates of the subregion normalizing constants from the original ABC posterior to calculate the marginal likelihood of the SAMC-ABC posterior. □

Proposition 2.4.4. *For small tolerance, ε, ABC kernel variance, σ_g^2, and SAMC-ABC kernel variance, σ_h^2, where $\sigma_h^2\leq\sigma_g^2$ we have that the point-wise posterior bias of the original ABC posterior is greater in magnitude than that of the SAMCABC posterior:*

$$\left|\hat{a}_{\varepsilon}^{(h)}\left(y|\theta\right)\right|\leq\left|\hat{a}_{\varepsilon}^{(g)}\left(y|\theta\right)\right|$$

with equality everywhere if and only if $\sigma_h^2 = \sigma_g^2$*.*

Proof. Writing the ABC likelihood as $f_\varepsilon\left(y|\theta\right)$ let us consider the normalizing constants:

$$\begin{aligned}
\pi_\varepsilon^{(g)}\left(y\right) =& \int \pi\left(\theta\right) f_\varepsilon\left(y|\theta\right) \mathrm{d}\theta \\
=& \int \pi\left(\theta\right) f\left(y|\theta\right) + b_\varepsilon\left(y|\theta\right) \pi\left(\theta\right) \mathrm{d}\theta \\
=& \pi\left(y\right) + \int b_\varepsilon\left(y|\theta\right) \pi\left(\theta\right) \mathrm{d}\theta
\end{aligned}$$

Now consider the ratio of the absolute pointwise posterior bias in the ABC and SAMC-ABC/CCABC posteriors:

$$\begin{aligned}
\frac{\left|a_\varepsilon^{(g)}\left(\theta|y\right)\right|}{\left|a_\varepsilon^{(h)}\left(\theta|y\right)\right|} =& \frac{\left|\frac{b_\varepsilon^{(g)}(y|\theta)\pi(\theta)+\left(\pi(y)-\pi_\varepsilon^{(g)}(y)\right)\pi(\theta|y)}{\pi_\varepsilon^{(g)}(y)}\right|}{\left|\frac{b_\varepsilon^{(h)}(y|\theta)\pi(\theta)+\left(\pi(y)-\pi_\varepsilon^{(h)}(y)\right)\pi(\theta|y)}{\pi_\varepsilon^{(h)}(y)}\right|} \\
=& \frac{\left|\pi_\varepsilon^{(h)}\left(y\right)\right|}{\left|\pi_\varepsilon^{(g)}\left(y\right)\right|} \frac{\left|b_\varepsilon^{(g)}\left(y|\theta\right)\pi\left(\theta\right) + \left(\pi\left(y\right) - \pi_\varepsilon^{(g)}\left(y\right)\right)\pi\left(\theta|y\right)\right|}{\left|b_\varepsilon^{(h)}\left(y|\theta\right)\pi\left(\theta\right) + \left(\pi\left(y\right) - \pi_\varepsilon^{(h)}\left(y\right)\right)\pi\left(\theta|y\right)\right|} \\
=& \frac{\pi\left(y\right) + \int b_\varepsilon^{(h)}\left(y|\theta\right)\pi\left(\theta\right)\mathrm{d}\theta}{\pi\left(y\right) + \int b_\varepsilon^{(g)}\left(y|\theta\right)\pi\left(\theta\right)\mathrm{d}\theta} \frac{\left|b_\varepsilon^{(g)}\left(y|\theta\right)\pi\left(\theta\right) - \pi\left(\theta|y\right)\int b_\varepsilon^{(g)}\left(y|\theta\right)\pi\left(\theta\right)\mathrm{d}\theta\right|}{\left|b_\varepsilon^{(h)}\left(y|\theta\right)\pi\left(\theta\right) - \pi\left(\theta|y\right)\int b_\varepsilon^{(h)}\left(y|\theta\right)\pi\left(\theta\right)\mathrm{d}\theta\right|}
\end{aligned}$$

and if we take the second order approximation of the bias in the likelihood and writing $k = \frac{1}{2}\varepsilon^2 \int f''\left(y|\theta\right)\pi\left(\theta\right)\mathrm{d}\theta$ we have:

$$\begin{aligned}
\frac{\left|a_\varepsilon^{(g)}\left(\theta|y\right)\right|}{\left|a_\varepsilon^{(h)}\left(\theta|y\right)\right|} \approx & \frac{c+\sigma_h^2 k}{c+\sigma_g^2 k}\frac{\left|\frac{1}{2}\varepsilon^2\sigma_g^2 f''\left(y|\theta\right)\pi\left(\theta\right)-\pi\left(\theta|y\right)\sigma_g^2 k\right|}{\left|\frac{1}{2}\varepsilon^2\sigma_h^2 f''\left(y|\theta\right)\pi\left(\theta\right)-\pi\left(\theta|y\right)\sigma_h^2 k\right|} \\
= & \frac{c+\sigma_h^2 k}{c+\sigma_g^2 k}\frac{\sigma_g^2}{\sigma_h^2}\frac{\left|\frac{1}{2}\varepsilon^2 f''\left(y|\theta\right)\pi\left(\theta\right)-\pi\left(\theta|y\right)k\right|}{\left|\frac{1}{2}\varepsilon^2 f''\left(y|\theta\right)\pi\left(\theta\right)-\pi\left(\theta|y\right)k\right|} \\
= & \frac{c+\sigma_h^2 k}{c+\sigma_g^2 k}\frac{\sigma_g^2}{\sigma_h^2} \\
> & \frac{\sigma_h^2 k}{\sigma_g^2 k}\frac{\sigma_g^2}{\sigma_h^2} \\
= & 1
\end{aligned} \tag{A.4}$$

As $\varepsilon \to 0$ both $a_\varepsilon^{(g)}\left(\theta|y\right) \to 0$ and $a_\varepsilon^{(h)}\left(\theta|y\right) \to 0$ however we also have from (A.4) that as $\varepsilon \to 0$ and the higher order terms in $b_\varepsilon\left(y|\theta\right)$ become negligible then $\left|a_\varepsilon^{(g)}\left(\theta|y\right)\right| > \left|a_\varepsilon^{(h)}\left(\theta|y\right)\right|$. □

Proposition 2.4.5. *For SAMC-ABC, we have that*

$$\sigma_h^2 = \frac{1}{3m^3}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(j^3-(j-1)^3\right)$$

and so it follows that the bias reduction factor is:

$$\frac{\hat{b}_\varepsilon^{(h)}\left(y|\theta\right)}{\hat{b}_\varepsilon^{(g)}\left(y|\theta\right)} = \frac{1}{m^3}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(j^3-(j-1)^3\right)$$

Proof. Here we provides results for the commonly used uniform ABC kernel $g_\varepsilon\left(u\right) = \frac{1}{2\varepsilon}I\left(|u|<\varepsilon\right)$. Then $\sigma_g^2 = \underset{g}{\mathrm{Var}}\left(U\right)$ where $g_\varepsilon\left(u\right) = \frac{1}{\varepsilon}g\left(\frac{u}{\varepsilon}\right)$. Hence $g\left(u\right) = \frac{1}{2}I\left(|u|<1\right)$ and $\sigma_A^2 = \frac{1}{3}$. We now calculate the variance of the SAMCABC, σ_h^2, kernel. Consider the SAMC-ABC kernel:

$$h_\varepsilon\left(u\right) = \frac{1}{2\varepsilon}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}I\left(\varepsilon_{j-1}\leq |u| < \varepsilon_j\right)$$

which with the ε_j evenly spaced becomes:

$$h_\varepsilon\left(u\right)=\frac{1}{2\varepsilon}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}I\left(\frac{\left(j-1\right)\varepsilon}{m}\leq\left|u\right|<\frac{j\varepsilon}{m}\right)$$

and hence:

$$h\left(u\right)=\frac{1}{2}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}I\left(\frac{\left(j-1\right)}{m}\leq\left|u\right|<\frac{j}{m}\right)$$

So we then have that:

$$\begin{aligned}\sigma_h^2 =&\underset{h}{\mathrm{Var}}\left(U\right)\\ =&\int_{-1}^{1}u^2\frac{1}{2}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}I\left(\frac{\left(j-1\right)}{m}\leq\left|u\right|<\frac{j}{m}\right)\mathrm{d}u\\ =&\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(\int_{-\frac{j}{m}}^{-\frac{j-1}{m}}\frac{u^2}{2}\mathrm{d}u+\int_{\frac{j-1}{m}}^{\frac{j}{m}}\frac{u^2}{2}\mathrm{d}u\right)\\ =&\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\int_{\frac{j-1}{m}}^{\frac{j}{m}}u^2\mathrm{d}u\\ =&\frac{1}{3m^3}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(j^3-\left(j-1\right)^3\right)\end{aligned}$$

Which allows us to calculate the pointwise bias reduction factor for the SAMC-ABC likelihood which is:

$$\begin{aligned}\frac{\hat{b}_\varepsilon^{(h)}\left(y|\theta\right)}{\hat{b}_\varepsilon^{(g)}\left(y|\theta\right)} =&\frac{\sigma_h^2}{\sigma_g^2}\\ =&\frac{\frac{1}{3m^3}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(j^3-\left(j-1\right)^3\right)}{\frac{1}{3}}\\ =&\frac{1}{m^3}\sum_{j=1}^{m}\frac{\varpi_j}{w_j}\left(j^3-\left(j-1\right)^3\right)\end{aligned}$$

□

A.2 Results of Chapter 3

Proposition 3.4.2. *The CCABC marginal likelihood $\pi_\varepsilon^{(h)}(y)$ can be calculated as*

$$\pi_\varepsilon^{(h)}(y) = \left(\int \frac{\varpi(u)}{C\hat{\xi}(u)} g_\varepsilon(u)\ du\right)^{-1}$$

Proof. Consider the CCABC algorithm. The CCABC algorithm produces an estimate of the marginal distribution shown in (A.5).

$$\xi(u) = \int_\Theta \int_{\{x:||y-x||=u\}} \pi(\theta) f(x|\theta) g_\varepsilon(||y-x||) \,\mathrm{d}x\mathrm{d}\theta \tag{A.5}$$

Then the ABC marginal likelihood can be written as:

$$\begin{aligned}\pi_\varepsilon^{(g)}(y) &= \int_U \xi(u)\,\mathrm{d}u \\ &= \int_U \frac{\tilde{\varpi}(u)}{\hat{\xi}(u)}\mathrm{d}u\end{aligned}$$

where U is the bounded region $[-\varepsilon,\varepsilon]$ upon which we have defined the lattice for the CCABC and $\tilde{\varpi}(u)$ is the desired sampling distribution on U. □

Proposition 3.4.2. *The CCABC marginal likelihood $\pi_\varepsilon^{(h)}(y)$ can be calculated as*

$$\pi_\varepsilon^{(h)}(y) = \left(\int \frac{\varpi(u)}{C\hat{\xi}(u)} g_\varepsilon(u)\ du\right)^{-1}$$

Proof. Consider the CCABC posterior:

$$\pi_\varepsilon^{(h)}(\theta|y) = \frac{1}{\pi_\varepsilon^{(h)}(y)}\pi(\theta) f(x|\theta) \frac{\varpi(||y-x||)}{\xi(||y-x||)\,Z} h_\varepsilon(||y-x||)$$

where $Z = \int \frac{\varpi(u)}{\xi(u)} g_\varepsilon(u)\,\mathrm{d}u$ is a normalizing constant which ensures that $k_\varepsilon(u) =$

$\frac{\varpi(u)}{\xi(u)Z}g_{\varepsilon}(u)$ is a valid ABC kernel and

$$\pi_{\varepsilon}^{(h)}(y)=\int_{\Theta}\int_{\mathcal{X}}\pi(\theta)\,f(x|\theta)\,\frac{\varpi(||y-x||)}{\xi(||y-x||)\,Z}g_{\varepsilon}(||y-x||)\,\mathrm{d}x\mathrm{d}\theta$$

is the CCABC marginal likelihood. Then the marginal likelihood for the CCABC posterior can be written as:

$$\begin{aligned}\pi_{\varepsilon}^{(h)}(y)=&\int_{U}\int_{\Theta}\int_{\{x:||y-x||=u\}}\pi(\theta)\,f(x|\theta)\,\frac{\varpi(u)}{\xi(u)\,Z}h_{\varepsilon}(||y-x||)\,\mathrm{d}x\mathrm{d}\theta\mathrm{d}u\\ =&\int_{U}\frac{\varpi(u)}{\xi(u)\,Z}\int_{\Theta}\int_{\{x:||y-x||=u\}}\pi(\theta)\,f(x|\theta)\,h_{\varepsilon}(||y-x||)\,\mathrm{d}x\mathrm{d}\theta\mathrm{d}u\\ =&\int_{U}\frac{\varpi(u)}{\xi(u)\,Z}\xi(u)\,\mathrm{d}u\\ =&\frac{1}{Z}\int_{U}\varpi(u)\,\mathrm{d}u\\ =&Z^{-1}\\ =&\left(\int\frac{\varpi(u)}{\xi(u)}g_{\varepsilon}(u)\,\mathrm{d}u\right)^{-1}\end{aligned}$$

□

Proposition 3.4.3. *For CCABC, we have that*

$$\sigma_h^2=\frac{1}{3m^3}\sum_{j=1}^{m}\left(j\hat{\xi}\left(\frac{(j-1)\,\varepsilon}{m}\right)-(j-1)\,\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3-(j-1)^3\right)$$

and so it follows that the bias reduction factor is:

$$\frac{\hat{b}_{\varepsilon}^{(h)}(y|\theta)}{\hat{b}_{\varepsilon}^{(g)}(y|\theta)}=\frac{1}{m^3}\sum_{j=1}^{m}\left(j\hat{\xi}\left(\frac{(j-1)\,\varepsilon}{m}\right)-(j-1)\,\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3-(j-1)^3\right)$$

Proof. Consider the pointwise bias reduction factor for the CCABC likelihood with the commonly used uniform ABC kernel $g_{\varepsilon}(u)=\frac{1}{2\varepsilon}I(|u|<\varepsilon)$. First we consider the CCABC kernel:

$$\begin{aligned} h_\varepsilon(u) =& \hat{\xi}(u)\frac{1}{2\varepsilon}I(|u|<\varepsilon) \\ =& \sum_{j=1}^{m}\left(\frac{\hat{\xi}(\varepsilon_{j-1})-\hat{\xi}(\varepsilon_j)}{\varepsilon_{j-1}-\varepsilon_j}(u-\varepsilon_j)+\hat{\xi}(\varepsilon_j)\right)\frac{1}{2\varepsilon}I(\varepsilon_{j-1}\leq|u|<\varepsilon_j) \end{aligned}$$

which with the ε_j evenly spaced becomes:

$$\begin{aligned} h_\varepsilon(u) =& \frac{1}{2\varepsilon}\sum_{j=1}^{m}\left(\frac{\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)-\hat{\xi}\left(\frac{j\varepsilon}{m}\right)}{\frac{(j-1)\varepsilon}{m}-\frac{j\varepsilon}{m}}\left(u-\frac{j\varepsilon}{m}\right)+\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)I\left(\frac{(j-1)\varepsilon}{m}\leq|u|<\frac{j\varepsilon}{m}\right) \\ =& \frac{1}{2\varepsilon}\sum_{j=1}^{m}\left(m\left(\hat{\xi}\left(\frac{j\varepsilon}{m}\right)-\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)\right)\left(\frac{u}{\varepsilon}-\frac{j}{m}\right)+\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)I\left(\frac{(j-1)}{m}\leq\left|\frac{u}{\varepsilon}\right|<\frac{j}{m}\right) \end{aligned}$$

and hence:

$$h(u) = \frac{1}{2}\sum_{j=1}^{m}\left(m\left(\hat{\xi}\left(\frac{j\varepsilon}{m}\right)-\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)\right)\left(u-\frac{j}{m}\right)+\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)I\left(\frac{(j-1)}{m}\leq|u|<\frac{j}{m}\right)$$

So we then have that:

$$\begin{aligned} \sigma_h^2 =& \underset{h}{\operatorname{Var}}(U) \\ =& \int_{-1}^{1}u^2h(u)\,\mathrm{d}u \\ =& \frac{1}{2}\sum_{j=1}^{m}\left(m\left(\hat{\xi}\left(\frac{j\varepsilon}{m}\right)-\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)\right)\left(\int_{-\frac{j}{m}}^{-\frac{j-1}{m}}\left(u^3-\frac{j}{m}u^2\right)\mathrm{d}u+\int_{\frac{j-1}{m}}^{\frac{j}{m}}\left(u^3-\frac{j}{m}u^2\right)\mathrm{d}u\right)+2\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\int_{\frac{j-1}{m}}^{\frac{j}{m}}u^2\mathrm{d}u\right) \\ =& \sum_{j=1}^{m}\left(\left(\hat{\xi}\left(\frac{j\varepsilon}{m}\right)-\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)\right)\left(\frac{j(j-1)^3-j^4}{3m^3}\right)+\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\left(\frac{j^3-(j-1)^3}{3m^3}\right)\right) \\ =& \frac{1}{3m^3}\sum_{j=1}^{m}\left(-j\hat{\xi}\left(\frac{j\varepsilon}{m}\right)+j\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)+\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3-(j-1)^3\right) \\ =& \frac{1}{3m^3}\sum_{j=1}^{m}\left(j\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)-(j-1)\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3-(j-1)^3\right) \end{aligned}$$

Which allows us to calculate the pointwise bias reduction factor for the CCABC likelihood which is:

$$
\begin{aligned}
\frac{\hat{b}_{\varepsilon}^{(h)}\left(y|\theta\right)}{\hat{b}_{\varepsilon}^{(g)}\left(y|\theta\right)} =&\frac{\sigma_h^2}{\sigma_g^2} \\
=&\frac{\frac{1}{3m^3}\sum_{j=1}^{m}\left(j\hat{\xi}\left(\frac{(j-1)\varepsilon}{m}\right)-\left(j-1\right)\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3-\left(j-1\right)^3\right)}{\frac{1}{3}} \\
=&\frac{1}{m^3}\sum_{j=1}^{m}\left(j\hat{\xi}\left(\frac{\left(j-1\right)\varepsilon}{m}\right)-\left(j-1\right)\hat{\xi}\left(\frac{j\varepsilon}{m}\right)\right)\left(j^3-\left(j-1\right)^3\right)
\end{aligned}
$$

□

Appendix B

Proofs of SARJ results

For geometric intermediate densities, ρ_t, as defined in Equation B.1 we have the following results:

$$\rho_t\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}; k \to k', x^{(t)}\right) = \left(\exp\left(\hat{f}_k\left(x^{(t)}|\theta_k^{(t)}\right)\right)\pi\left(\theta_k^{(t)}|k\right)\pi\left(k\right)\varphi_{k\to k'}\left(u_{k\to k'}^{(t)}\right)J_{k'\to k}^{-1}\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}\right)\right)^{1-\frac{t}{T}} \tag{B.1}$$

$$\times\left(\exp\left(\hat{f}_{k'}\left(x^{(t)}|\theta_{k'}^{(t)}\right)\right)\pi\left(\theta_{k'}^{(t)}|k'\right)\pi\left(k'\right)\varphi_{k'\to k}\left(u_{k'\to k}^{(t)}\right)\right)^{\frac{t}{T}}$$

Proposition 7.3.1. *Given a path* $\left(\theta_{k'}^{(1)}, u_{k'\to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k'\to k}^{(T-1)}\right)$*, a minibatch size* $m < N$*, and that the cumulative distribution function of the likelihood is Lipschitz continuous then the log stochastic annealing importance weight,* $\log r_{k\to k'}^{(0:T-1),m}$*, converges in distribution to a Normal distribution* $N\left(\mu_T, \sigma_T^2\right)$ *as* $T\to\infty$

Proof. Let

$$\nu_t = \log\rho_t\left(\theta_{k'}^{(t-1)}, u_{k'\to k}^{(t-1)}; k\to k', x^{(t)}\right) - \log\rho_t\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}; k\to k', x^{(t)}\right) \tag{B.2}$$

for $t = 0, ..., T-1$ then for $\eta_t = T\nu_t$ we can write

$$S_T = T \log r_{k\to k'}^{(0:T-1),m} = \sum_{t=0}^{T-1} \eta_t$$

and we have that η_t does not depend on T. Moreover for the given path we have that the η_t are independent by the independence of the minibatches and so we can write the variance of S_T as:

$$s_T^2 = \mathrm{Var}\left(S_T\right) = \sum_{t=0}^{T-1} \mathrm{Var}\left(\eta_t\right)$$

Now by the Lipschitz continuity of the CDF of the likelihood we know that each η_t is bounded and since the expectations are finite we have that $\eta_t - \mathbb{E}\left[\eta_t\right]$ is also bounded by some constant K_t. Thus the $\eta_t - \mathbb{E}\left[\eta_t\right]$ are uniformly bounded by $K = \max K_t$. Hence we can write the following :

$$\frac{1}{s_T^3} \sum_{t=0}^{T-1} \mathbb{E}\left[\left|\eta_t - \mathbb{E}\left[\eta_t\right]\right|^3\right] \leq \frac{K}{s_T^3} \sum_{t=0}^{T-1} \mathbb{E}\left[\left(\eta_t - \mathbb{E}\left[\eta_t\right]\right)^2\right] = \frac{K s_T^2}{s_T^3} = \frac{K}{s_T} \to 0$$

as $T \to \infty$ since $s_T \to \infty$. Clearly then the Lyapounov condition is satisfied for $\delta = 1$ and therefore S_T obeys a central limit theorem and converges in distribution to a normal distribution. Consequently $\log r_{k\to k'}^{(0:T-1),m} = \frac{S_T}{T}$ also converges to a normal distribution with mean and variance equal to $\mu_T = \sum_{t=1}^{T} \mathbb{E}\left[\eta_t\right]$ and $\sigma_T^2 = \frac{s_T^2}{T^2} = \frac{1}{T^2} \sum_{t=0}^{T-1} \mathrm{Var}\left(\eta_t\right)$ respectively. □

Proposition 7.3.2. *Given a path* $\left(\theta_{k'}^{(1)}, u_{k'\to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k'\to k}^{(T-1)}\right)$ *and a minibatch size, m, then* $\mathbb{E}\left[\log r_{k\to k'}^{(0:T-1),m}\right] = \log r_{k\to k'}^{(0:T-1),N}$ *where* $r_{k\to k'}^{(0:T-1),m}$ *is the ratio achieved with minibatch size m.*

Proof. Consider $\rho_t^{(m)} = \rho_t\left(\theta_{k'}^{(t)}, u_{k'\to k}^{(t)}; k \to k', x^{(t,m)}\right)$ the intermediate density constructed with a subsample of size m. Then we have that

$$
\begin{aligned}
\log\rho_t^{(m)} =& \left(1-\frac{t}{T}\right)\left(\hat{f}_k^{(m)}\left(x^{(t,m)}|\theta_k^{(t)}\right)+\log\left(\pi\left(\theta_k^{(t)}|k\right)\pi\left(k\right)\varphi_{k\to k'}\left(u_{k\to k'}^{(t)}\right)J_{k'\to k}^{-1}\left(\theta_{k'}^{(t)},u_{k'\to k}^{(t)}\right)\right)\right)\\
&+\frac{t}{T}\left(\hat{f}_{k'}^{(m)}\left(x^{(t,m)}|\theta_{k'}^{(t)}\right)+\log\left(\pi\left(\theta_{k'}^{(t)}|k'\right)\pi\left(k'\right)\varphi_{k'\to k}\left(u_{k'\to k}^{(t)}\right)\right)\right)
\end{aligned}
\tag{B.3}
$$

Since neither of these terms depend on the subsample or on the subsample size we denote the following expression as C for simplicity:

$$
\begin{aligned}
C =& \left(1-\frac{t}{T}\right)\log\left(\pi\left(\theta_k^{(t)}|k\right)\pi\left(k\right)\varphi_{k\to k'}\left(u_{k\to k'}^{(t)}\right)J_{k'\to k}^{-1}\left(\theta_{k'}^{(t)},u_{k'\to k}^{(t)}\right)\right)\\
&+\frac{t}{T}\log\left(\pi\left(\theta_{k'}^{(t)}|k'\right)\pi\left(k'\right)\varphi_{k'\to k}\left(u_{k'\to k}^{(t)}\right)\right)
\end{aligned}
$$

Now the expectation of $\log\rho_t^{(m)}$ can be written as:

$$
\begin{aligned}
\mathbb{E}\left[\log\rho_t^{m}\right] =& \mathbb{E}\left[\left(1-\frac{t}{T}\right)\hat{f}_k^{(m)}\left(x^{(t,m)}|\theta_k^{(t)}\right)+\frac{t}{T}\hat{f}_{k'}^{(m)}\left(x^{(t,m)}|\theta_{k'}^{(t)}\right)+C\right]\\
=& \left(1-\frac{t}{T}\right)\mathbb{E}\left[\hat{f}_k^{(m)}\left(x^{(t,m)}|\theta_k^{(t)}\right)\right]+\frac{t}{T}\mathbb{E}\left[\hat{f}_{k'}^{(m)}\left(x^{(t,m)}|\theta_{k'}^{(t)}\right)\right]+C
\end{aligned}
$$

Thus since $\hat{f}_k^{(m)}$ is an unbiased estimator of $\hat{f}_k^{(N)}$ we have:

$$
\begin{aligned}
\mathbb{E}\left[\log\rho_t^{(m)}\right] =& \left(1-\frac{t}{T}\right)\hat{f}_k^{(N)}\left(x^{(t,m)}|\theta_k^{(t)}\right)+\frac{t}{T}\hat{f}_{k'}^{(N)}\left(x^{(t,m)}|\theta_{k'}^{(t)}\right)+C\\
=& \log\rho_t^{(N)}
\end{aligned}
$$

Now by linearity we can see that the expectation of the log ratio is unbiased:

$$
\mathbb{E}\left[\log r_{k\to k'}^{(0:T-1),m}\right] = \log r_{k\to k'}^{(0:T-1),N}
$$

□

Proposition 7.3.4. *The stochastic annealed importance weight $r_{k\to k'}^{(0:T-1),m}$ is biased. Under the conditions of Proposition 7.3.1 this bias asymptotically tends to a factor of $e^{\frac{\sigma_T^2}{2}}$ as T grows large.*

Proof. Given Propositions 7.3.1 and 7.3.2 and as $T \to \infty$ we have that:

$$\log r_{k\to k'}^{(0:T-1),m} \sim \mathrm{N}\left(\mu_T^{(m)}, \sigma_{T,m}^2\right)$$

where $\log r_{k\to k'}^{(0:T-1),m}$ is an unbiased estimator of $\log r_{k\to k'}^{(0:T-1),N}$. We can further write the distribution of $r_{k\to k'}^{(0:T-1),m}$ as:

$$r_{k\to k'}^{(0:T-1),m} \sim \mathrm{LogNormal}\left(\mu_T, \sigma_T^2\right)$$

which has mean $\exp\left(\mu_T + \frac{\sigma_T^2}{2}\right)$. Note now that $r_{k\to k'}^{(0:T-1),N} = \exp\left(\mu_T\right)$ and so $r_{k\to k'}^{(0:T-1),m}$ is biased by a factor of $\exp\left(\frac{\sigma_T^2}{2}\right)$. □

Proposition 7.3.6. *Assume that the standard Annealed Importance Sampling reversible jump algorithm (Algorithm 6.7) satisfies the Wasserstein ergodicity assumption (Assumption 1). Then writing $P_{\tilde{\alpha}}$ for the transition kernel of the SARJ algorithm and $Q(\cdot,\cdot)$ for the transition kernel of the proposal density we have the following bound for the Wasserstein distance between the jth iterations*

$$W\left(P_{\alpha}^{j}(\theta,\cdot), P_{\tilde{\alpha}}^{j}(\theta,\cdot)\right) \leq \frac{C\left(1-\rho^{j}\right)}{(1-\rho)} \sup_{\theta\in\Theta}\left(\left(\left(\int_{\Theta} d\left(\theta,\theta'\right)^2 Q\left(\theta, d\theta'\right)\right)\left(\int_{\Theta}\left(\exp\left(\frac{\sigma_T^2}{2}\right)-1\right)^2 Q\left(\theta, d\theta'\right)\right)\right)^{\frac{1}{2}}\right)$$

Proof. From Corollary 4.1 of Rudolf and Schweizer (2018) that the Wasserstein distance between the jth iterations is bounded:

$$W\left(P_{\alpha}^{j}(\theta,\cdot), P_{\tilde{\alpha}}^{j}(\theta,\cdot)\right) \leq \frac{\gamma C\left(1-\rho^{j}\right)}{(1-\rho)}$$

where C and ρ are the constants satisfying the Wasserstein ergodicity assumption (Assumption 1), and

$$\gamma = \sup_{\theta\in\Theta}\int_{\Theta} d\left(\theta,\theta'\right)\mathcal{E}\left(\theta,\theta'\right)Q\left(\theta,\mathrm{d}\theta'\right)$$

where $\mathcal{E}\left(\theta,\theta'\right) = \left|\alpha\left(\theta,\theta'\right)-\tilde{\alpha}\left(\theta,\theta'\right)\right|$ is the bias in the acceptance ratio. From Proposition 7.3.4 we have that $\tilde{\alpha}\left(\theta,\theta'\right) = \alpha\left(\theta,\theta'\right)\times\exp\left(\frac{\sigma_T^2}{2}\right)$ and so $\mathcal{E}\left(\theta,\theta'\right) = \alpha\left(\theta,\theta'\right)\left|1-\exp\left(\frac{\sigma_T^2}{2}\right)\right| \leq \exp\left(\frac{\sigma_T^2}{2}\right)-1$. Finally we follow Rudolf and Schweizer (2018) and apply Cauchy-Schwartz to the integral $\int_\Theta d\left(\theta,\theta'\right)\mathcal{E}\left(\theta,\theta'\right)Q\left(\theta,\mathrm{d}\theta'\right)$ to note that

$$\begin{aligned}\gamma \leq & \sup_{\theta\in\Theta}\left(\left(\left(\int_\Theta d\left(\theta,\theta'\right)^2 Q\left(\theta,\mathrm{d}\theta'\right)\right)\cdot\left(\int_\Theta \mathcal{E}\left(\theta,\theta'\right)^2 Q\left(\theta,\mathrm{d}\theta'\right)\right)\right)^{\frac{1}{2}}\right)\\ \leq & \sup_{\theta\in\Theta}\left(\left(\left(\int_\Theta d\left(\theta,\theta'\right)^2 Q\left(\theta,\mathrm{d}\theta'\right)\right)\cdot\left(\int_\Theta \left(\exp\left(\frac{\sigma_T^2}{2}\right)-1\right)^2 Q\left(\theta,\mathrm{d}\theta'\right)\right)\right)^{\frac{1}{2}}\right)\end{aligned}$$

and the bound on $W\left(P_\alpha^j\left(\theta,\cdot\right),P_{\tilde{\alpha}}^j\left(\theta,\cdot\right)\right)$ follows. □

Proposition 7.4.1. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Variance reduction in* $\hat{f}_k^{(t)}$ *and* $\hat{f}_{k'}^{(t)}$ *for all* $t = 0,1,...,T-1$ *implies variance reduction in* $r_{k\to k'}^{(0:T-1),m}$

Proof. First let us again consider $\log\rho_t^{(m)}$ as defined in Equation B.3. Writing $\hat{f}_k^{(t)} = \hat{f}_k^{(m)}\left(x^{(t,m)}|\theta_k^{(t)}\right)$ for brevity we have that the variance of $\log\hat{\rho}_t^{(m)}$ can be calculated as:

$$\mathrm{Var}\left(\log\hat{\rho}_t^{(m)}\right) = \left(1-\frac{t}{T}\right)^2\mathrm{Var}\left(\hat{f}_k^{(t)}\right)+\left(\frac{t}{T}\right)^2\mathrm{Var}\left(\hat{f}_{k'}^{(t)}\right)+\left(1-\frac{t}{T}\right)\left(\frac{t}{T}\right)\mathrm{Cov}\left(\hat{f}_k^{(t)},\hat{f}_{k'}^{(t)}\right)$$

now recall that $\hat{f}_k^{(m,t)} = f_k^{(m,t)}+\tilde{f}^{(m,t)}-\mathbb{E}\left[\tilde{f}^{(m,t)}\right]$ where $f_k^{(m,t)}$ is the log likelihood estimate without variance reduction and $\tilde{f}^{(m,t)}$ is the control variate used for variance reduction. Then from the variance reduction in $f_k^{(t)}$ for both k and k' we have:

$$\mathrm{Var}\left(\hat{f}_k^{(t)}\right) = \mathrm{Var}\left(f_k^{(t)}\right) + \mathrm{Var}\left(\tilde{f}^{(t)}\right) - 2\mathrm{Cov}\left(f_k^{(t)}, \tilde{f}^{(t)}\right) \leq \mathrm{Var}\left(f_k^{(t)}\right)$$

Now we look at ν_t and $\hat{\nu}_t$ where ν_t is defined as in Equation B.2 and $\hat{\nu}_t$ is the equivalent estimator using the variance reduced log likelihood. Then writing $\log \hat{\rho}_t \left(\theta^{(t-1)}\right) = \log \hat{\rho}_t \left(\theta_{k'}^{(t-1)}, u_{k' \to k}^{(t-1)}; k \to k', x^{(t)}\right)$ for brevity we have:

$$\begin{aligned}
\mathrm{Var}\left(\hat{\nu}_t\right) =& \mathrm{Var}\left(\log \hat{\rho}_t\left(\theta^{(t-1)}\right) - \log \hat{\rho}_t\left(\theta^{(t)}\right)\right) \\
=& \mathrm{Var}\left(\log \hat{\rho}_t\left(\theta^{(t-1)}\right)\right) + \mathrm{Var}\left(\log \hat{\rho}_t\left(\theta^{(t)}\right)\right) - \mathrm{Cov}\left(\log \hat{\rho}_t\left(\theta^{(t-1)}\right), \log \hat{\rho}_t\left(\theta^{(t)}\right)\right) \\
=& \mathrm{Var}\left(\log \rho_t\left(\theta^{(t-1)}\right)\right) + \mathrm{Var}\left(\log \rho_t\left(\theta^{(t)}\right)\right) - \mathrm{Cov}\left(\log \rho_t\left(\theta^{(t-1)}\right), \log \rho_t\left(\theta^{(t)}\right)\right) \\
& - \frac{t^2 - 2Tt + T^2}{T^2}\mathrm{Cov}\left(f_k^{(t)}\left(\theta^{(t-1)}\right), \tilde{f}^{(t)}\right) - \frac{t^2 - 2Tt + T^2}{T^2}\mathrm{Cov}\left(f_k^{(t)}\left(\theta^{(t)}\right), \tilde{f}^{(t)}\right) \\
& - \frac{t^2}{T^2}\mathrm{Cov}\left(f_{k'}^{(t)}\left(\theta^{(t-1)}\right), \tilde{f}^{(t)}\right) - \frac{t^2}{T^2}\mathrm{Cov}\left(f_{k'}^{(t)}\left(\theta^{(t)}\right), \tilde{f}^{(t)}\right) + \frac{2t^2 - 2Tt + T^2}{T^2}\mathrm{Var}\left(\tilde{f}^{(t)}\right) \\
\leq& \mathrm{Var}\left(\nu_t\right) - 2\left(\frac{2t^2 - 2Tt + T^2}{T^2}\right) C_{min} + \left(\frac{2t^2 - 2Tt + T^2}{T^2}\right) \mathrm{Var}\left(\tilde{f}^{(t)}\right) \\
\leq& \mathrm{Var}\left(\nu_t\right)
\end{aligned}$$

where

$$C_{min} = \min\left[\mathrm{Cov}\left(\hat{f}_k^{(t)}\left(\theta^{(t-1)}\right), \tilde{f}^{(t)}\right), \mathrm{Cov}\left(\hat{f}_k^{(t)}\left(\theta^{(t)}\right), \tilde{f}^{(t)}\right), \mathrm{Cov}\left(\hat{f}_{k'}^{(t)}\left(\theta^{(t-1)}\right), \tilde{f}^{(t)}\right), \mathrm{Cov}\left(\hat{f}_{k'}^{(t)}\left(\theta^{(t)}\right), \tilde{f}^{(t)}\right)\right]$$

Then the first inequality follows from the positivity of $\frac{2t^2 - 2Tt + T^2}{T^2}$ and the last inequality follows from the variance reduction in both $f_k^{(t)}$ and $f_{k'}^{(t)}$. Now since the $\hat{\nu}_t$ are independent given the path $\left(\theta_{k'}^{(1)}, u_{k' \to k}^{(1)}\right), ..., \left(\theta_{k'}^{(T-1)}, u_{k' \to k}^{(T-1)}\right)$ and $\log \rho_t^{(m)}$ is simply the sum of all the ν_t it clearly follows that the variance of $\log \rho_t^{(m)}$ will also be reduced. Finally as $T \to \infty$ we have that $\log r_{k \to k'}^{(0:T-1),m} \overset{D}{\to} N\left(\mu_T, \sigma_T^2\right)$ by Proposition 7.3.1. Note that the variance reduction leaves μ_T unchanged and that:

$$\mathrm{Var}\left(r_{k \to k'}^{(0:T-1),m}\right) = \left[\exp\left(\sigma_T^2\right) - 1\right] \exp\left(2\mu_T + \sigma_T^2\right)$$

Thus since the variance of the stochastic annealed importance weight is monotonically increasing in the variance of $\log r_{k\to k'}^{(0:T-1),m}$ it follows that the variance reduction of $\log r_{k\to k'}^{(0:T-1),m}$ causes variance reduction in $r_{k\to k'}^{(0:T-1),m}$. □

Proposition 7.4.2. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Then variance reduction in $\hat{f}_k$ reduces σ_T^2 and thus since the bias in $r_{k\to k'}^{(0:T-1),m}$ is asymptotically equal to $e^{\frac{\sigma_T^2}{2}}$ there is bias reduction in $r_{k\to k'}^{(0:T-1),m}$*

Proof. Consider the bias factor of Proposition 7.3.4, $\exp\left(\frac{\sigma_T^2}{2}\right)$. This factor increases monotonically with the variance of the log ratio. Hence since we showed in Proposition 7.4.1 that the Variance Reduction we apply to $\hat{f}_k$ results in reduced variance of the log ratio it follows that this bias factor is also reduced. □

Proposition 7.4.4. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Given the uncertainty penalty of Algorithm 7.3 we can upper bound the expected remaining bias factor $\zeta_{T,n} = \exp\left(\frac{\sigma_T^2}{2} - \frac{\hat{\sigma}_{T,n}^2}{2}\right) \leq 1 + \frac{\sigma_T^4}{4(n-1)} \max\left\{1, \frac{\sigma_T^2}{2} - \frac{\hat{\sigma}_{T,n}^2}{2}\right\}$.*

Proof. First let's write $X_{T,n} = -\frac{\hat{\sigma}_{T,n}^2}{2}$ and consider the expectation:

$$\mathbb{E}\left[\zeta_{T,n}\right] = \frac{\mathbb{E}\left[\exp\left(X\right)\right]}{\exp\left(\mathbb{E}\left[X\right]\right)}$$

Then Taylor expansion of $\exp(X)$ about $\mathbb{E}[X]$ with mean value remainder gives us:

$$\begin{aligned}\mathbb{E}\left[\exp\left(X\right)\right] =& \mathbb{E}\left[e^{\mathbb{E}[X]} + e^{\mathbb{E}[X]}\left(X - \mathbb{E}\left[X\right]\right) + e^{z}\frac{\left(X - \mathbb{E}\left[X\right]\right)^2}{2}\right] \\ =& e^{\mathbb{E}[X]} + e^{z}\frac{\mathrm{Var}\left(X\right)}{2}\end{aligned}$$

for some z between $\frac{\sigma_T^2}{2}$ and $\frac{\hat{\sigma}_{T,n}^2}{2}$. Now recall that under the conditions of Proposition 7.3.1 $\hat{\sigma}_T^2$ is the sample variance of n iid normal random variables. Hence:

$$\begin{aligned}\mathbb{E}\left[\exp\left(X\right)\right] =& e^{\mathbb{E}[X]} + e^{z}\frac{\mathrm{Var}\left(\hat{\sigma}^2_{T,n}\right)}{8}\\ =& e^{\mathbb{E}[X]} + e^{z}\frac{\sigma^4_T}{4\left(n-1\right)}\\ \leq& e^{\mathbb{E}[X]} + \frac{\sigma^4_T}{4\left(n-1\right)}\exp\left(\max\left\{\mathbb{E}\left[X\right], X\right\}\right)\end{aligned}$$

And finally we have the upper bound on the expected remaining bias factor $\zeta_{T,n}$:

$$\begin{aligned}\mathbb{E}\left[\zeta_{T,n}\right] =& \frac{\mathbb{E}\left[\exp\left(X\right)\right]}{\exp\left(\mathbb{E}\left[X\right]\right)}\\ \leq& \frac{e^{\mathbb{E}[X]} + \frac{\sigma^4_T}{4(n-1)}\exp\left(\max\left\{\mathbb{E}\left[X\right], X\right\}\right)}{e^{\mathbb{E}[X]}}\\ =& 1 + \frac{\sigma^4_T}{4\left(n-1\right)}\max\left\{1, X - \mathbb{E}\left[X\right]\right\}\\ =& 1 + \frac{\sigma^4_T}{4\left(n-1\right)}\max\left\{1, \frac{\sigma^2_T}{2} - \frac{\hat{\sigma}^2_{T,n}}{2}\right\}\end{aligned}$$

□

Proposition 7.4.5. *Assume that the AISRJ sampler satisfies the Wasserstein ergodicity assumption (Assumption 1). Given the uncertainty penalty of Algorithm 7.3 we have an upper bound for the bias factor* $\zeta_{T,n} = \exp\left(\frac{\sigma^2_T}{2} - \frac{\hat{\sigma}^2_{T,n}}{2}\right)$*. Consider the sequence* $Y_{T,n} = \zeta_{T,n} - 1$ *which describes the difference between the bias factor and unbiasedness as* n *or* T *grows. Then* $Y_{T,n} = o_P\left(1\right)$ *in* T *and when a further condition on the variance* σ^2_T *of the annealed importance sampling weight estimator* $r^{(0:T-1),m}_{k\to k'}$ *is satisfied then* $Y_{T,n} = o_P\left(1\right)$ *in* n *as well.*

Proof. Under the conditions of Proposition 7.3.1 we have that $X_{T,n} = \frac{\hat{\sigma}^2_{T,n}}{2} \sim \Gamma\left(\frac{n-1}{2}, \frac{n-1}{\sigma^2_T}\right)$. From this it follows that the cumulative distribution function of $Y_{T,n}$ is given by:

$$F_Y(y) = 1 - F_X\left(\frac{\sigma_T^2}{2} - \log(y+1)\right)$$

where F_X is the cumulative distribution function of $X_{T,n}$. Then we can write:

$$\begin{aligned}
\mathbb{P}\left(|Y_{T,n}| \geq \epsilon\right) =& 1 - \left(F_Y(\epsilon) - F_Y(-\epsilon)\right) \\
=& 1 - \left(1 - F_X\left(\frac{\sigma_T^2}{2} - \log(1+\epsilon)\right) - 1 + F_X\left(\frac{\sigma_T^2}{2} - \log(1-\epsilon)\right)\right) \\
=& 1 - \frac{1}{\Gamma\left(\frac{n-1}{2}\right)}\left(\int_0^{\frac{n-1}{2} - \frac{n-1}{\sigma_T^2}\log(1-\epsilon)} t^{\frac{n-1}{2}-1} e^{-t} \mathrm{d}t\right. \\
& \left. - \int_0^{\max\left\{0, \frac{n-1}{2} - \frac{n-1}{\sigma_T^2}\log(1+\epsilon)\right\}} t^{\frac{n-1}{2}-1} e^{-t} \mathrm{d}t\right) \\
=& 1 - \frac{\int_{\max\left\{0, \frac{n-1}{2} - \frac{n-1}{\sigma_T^2}\log(1+\epsilon)\right\}}^{\frac{n-1}{2} - \frac{n-1}{\sigma_T^2}\log(1-\epsilon)} t^{\frac{n-1}{2}-1} e^{-t} \mathrm{d}t}{\int_0^\infty t^{\frac{n-1}{2}-1} e^{-t} \mathrm{d}t}
\end{aligned}$$

It is clear that as $\sigma_T^2 \to 0$ the limits on the integral in the numerator go to 0 and ∞ respectively and so $\lim\limits_{\sigma_T^2 \to 0} \mathbb{P}\left(|Y_{T,n}| \geq \epsilon\right) = 0$. Hence since $\sigma_T^2 \to 0$ as $T \to \infty$ by Corollary 7.3.3 we have that $\lim\limits_{T\to\infty} \mathbb{P}\left(|Y_{T,n}| \geq \epsilon\right) = 0$ and thus $Y_{T,n} = o_P(1)$ in T.

Now consider the behaviour of $\mathbb{P}\left(|Y_{T,n}| \geq \epsilon\right)$ as $n \to \infty$. If $\frac{\log(1+\epsilon)}{\sigma_T^2} < \frac{1}{2}$ then the lower limit of the integral in the numerator grows with n and $\lim\limits_{n\to\infty} \mathbb{P}\left(|Y_{T,n}| \geq \epsilon\right) \neq 0$. However for large enough T we have that $\sigma_T^2 \leq 2\log(1+\epsilon)$ and thus the lower limit is 0 since for $\sigma_T^2 \leq 2\log(1+\epsilon)$ it follows that $\frac{n-1}{2} - \frac{n-1}{\sigma_T^2}\log(1+\epsilon) < 0$. Then it is clear that $\lim\limits_{n\to\infty} \mathbb{P}\left(|Y_{T,n}| \geq \epsilon\right) = 0$ and hence $Y_{T,n} = o_P(1)$ in n for such T. □

Bibliography

Allingham, David, Robert AR King and Kerrie L Mengersen (2009). "Bayesian estimation of quantile distributions". In: *Statistics and Computing* 19.2, pp. 189–201.

Andrieu, Christophe and Johannes Thoms (2008). "A tutorial on adaptive MCMC". In: *Statistics and computing* 18.4, pp. 343–373.

Beaumont, Mark A (2010). "Approximate Bayesian computation in evolution and ecology". In: *Annual review of ecology, evolution, and systematics* 41, pp. 379–406.

Beaumont, Mark A, Wenyang Zhang and David J Balding (2002). "Approximate Bayesian computation in population genetics". In: *Genetics* 162.4, pp. 2025–2035.

Blum, Michael GB (2010). "Approximate Bayesian computation: A nonparametric perspective". In: *Journal of the American Statistical Association* 105.491, pp. 1178–1187.

Booij, NRRC, Roeland C Ris and Leo H Holthuijsen (1999). "A third-generation wave model for coastal regions: 1. Model description and validation". In: *Journal of geophysical research: Oceans* 104.C4, pp. 7649–7666.

Breiman, Leo and Ross Ihaka (1984). *Nonlinear discriminant analysis via scaling and ACE.* Department of Statistics, University of California.

Butler, A et al. (2006). *A Latent Gaussian Model for Compositional Data with Structural Zeroes (Biomathematics and Statistics Scotland, Edinburgh).* Tech. rep. technical report.

Cao, Yang, Daniel T Gillespie and Linda R Petzold (2006). "Efficient step size selection for the tau-leaping simulation method". In: *The Journal of chemical physics* 124.4, p. 044109.

CDC (2014). *CDC 2014 Ebola Outbreak in West Africa: Reported Cases Graphs Ebola Hemorrhagic Fever.* https://www.cdc.gov/vhf/ebola/history/2014-2016-outbreak/cumulative-cases-graphs.html.

Ceperley, DM and Mark Dewing (1999). "The penalty method for random walks with uncertain energies". In: *The Journal of chemical physics* 110.20, pp. 9812–9820.

Chatterji, Niladri et al. (2018). "On the theory of variance reduction for stochastic gradient Monte Carlo". In: *International Conference on Machine Learning.* PMLR, pp. 764–773.

Chen, Changyou, Nan Ding and Lawrence Carin (2015). "On the convergence of stochastic gradient MCMC algorithms with high-order integrators". In.

Chen, Hao et al. (2022). "Gaussian process parameter estimation using mini-batch stochastic gradient descent: convergence guarantees and empirical benefits". In: *The Journal of Machine Learning Research* 23.1, pp. 10298–10356.

Chen, Tianqi, Emily Fox and Carlos Guestrin (2014). "Stochastic gradient hamiltonian monte carlo". In: *International conference on machine learning.* PMLR, pp. 1683–1691.

Chipman, Hugh A, Edward I George and Robert E McCulloch (2010). "BART: Bayesian additive regression trees". In: *The Annals of Applied Statistics* 4.1, pp. 266–298.

Cressie, Noel and Gardar Johannesson (2008). “Fixed rank kriging for very large spatial data sets”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.1, pp. 209–226.

Denison, David GT, Bani K Mallick and Adrian FM Smith (1998). “A bayesian cart algorithm”. In: *Biometrika* 85.2, pp. 363–377.

Dietrich, J Casey et al. (2012). “Performance of the unstructured-mesh, SWAN+ ADCIRC model in computing hurricane waves and surge”. In: *Journal of Scientific Computing* 52, pp. 468–497.

Dietrich, JC et al. (2011). “Modeling hurricane waves and storm surge using integrally-coupled, scalable computations”. In: *Coastal Engineering* 58.1, pp. 45–65.

Dijkstra, Edsger W (1959). “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1, pp. 269–271.

Drovandi, Christopher C and Anthony N Pettitt (2011). “Likelihood-free Bayesian estimation of multivariate quantile distributions”. In: *Computational Statistics & Data Analysis* 55.9, pp. 2541–2556.

Du, Juan, Hao Zhang and VS2549562 Mandrekar (2009). “Fixed-domain asymptotic properties of tapered maximum likelihood estimators”. In: *the Annals of Statistics* 37.6A, pp. 3330–3361.

Duane, Simon et al. (1987). “Hybrid monte carlo”. In: *Physics letters B* 195.2, pp. 216–222.

Dubey, Avinava et al. (2016). “Variance reduction in stochastic gradient Langevin dynamics”. In: *Advances in neural information processing systems* 29, p. 1154.

Ebola Response Team, WHO (2014). “Ebola virus disease in West Africa: the first 9 months of the epidemic and forward projections”. In: *New England Journal of Medicine* 371.16, pp. 1481–1495.

Frazier, David T, Christian P Robert and Judith Rousseau (2020). “Model misspecification in approximate Bayesian computation: consequences and diagnostics”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology).*

Furrer, Reinhard, Marc G Genton and Douglas Nychka (2006). “Covariance tapering for interpolation of large spatial datasets”. In: *Journal of Computational and Graphical Statistics* 15.3, pp. 502–523.

Gelman, Andrew and Xiao-Li Meng (1998). “Simulating normalizing constants: From importance sampling to bridge sampling to path sampling”. In: *Statistical science*, pp. 163–185.

Gillespie, Daniel T (1977). “Exact stochastic simulation of coupled chemical reactions”. In: *The journal of physical chemistry* 81.25, pp. 2340–2361.

— (2001). “Approximate accelerated stochastic simulation of chemically reacting systems”. In: *The Journal of Chemical Physics* 115.4, pp. 1716–1733.

Gramacy, Robert B and Herbert K H Lee (2008). “Bayesian treed Gaussian process models with an application to computer modeling”. In: *Journal of the American Statistical Association* 103.483, pp. 1119–1130.

Green, Peter J (1995). “Reversible jump Markov chain Monte Carlo computation and Bayesian model determination”. In: *Biometrika* 82.4, pp. 711–732.

Gu, Mengyang, Xiaojing Wang and James O Berger (2018). “Robust Gaussian stochastic process emulation”. In: *The Annals of Statistics* 46.6A, pp. 3038–3066.

Jarzynski, Christopher (1997a). “Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach”. In: *Physical Review E* 56.5, p. 5018.

— (1997b). “Nonequilibrium equality for free energy differences”. In: *Physical Review Letters* 78.14, p. 2690.

Karagiannis, Georgios and Christophe Andrieu (2013). “Annealed importance sampling reversible jump MCMC algorithms”. In: *Journal of Computational and Graphical Statistics* 22.3, pp. 623–648.

Kaufman, Cari G, Mark J Schervish and Douglas W Nychka (2008). “Covariance tapering for likelihood-based estimation in large spatial data sets”. In: *Journal of the American Statistical Association* 103.484, pp. 1545–1555.

Knorr-Held, Leonhard and Günter Raßer (2000). “Bayesian detection of clusters and discontinuities in disease maps”. In: *Biometrics* 56.1, pp. 13–21.

Konomi, Bledar et al. (2014). “Bayesian treed multivariate gaussian process with adaptive design: Application to a carbon capture unit”. In: *Technometrics* 56.2, pp. 145–158.

Konomi, Bledar A and Georgios Karagiannis (2021). “Bayesian Analysis of Multifidelity Computer Models With Local Features and Nonnested Experimental Designs: Application to the WRF Model”. In: *Technometrics* 63.4, pp. 510–522.

Kruskal, Joseph B (1956). “On the shortest spanning subtree of a graph and the traveling salesman problem”. In: *Proceedings of the American Mathematical society* 7.1, pp. 48–50.

Lee, Der-Tsai and Bruce J Schachter (1980). “Two algorithms for constructing a Delaunay triangulation”. In: *International Journal of Computer & Information Sciences* 9.3, pp. 219–242.

Li, Zhize et al. (2019). “Stochastic gradient hamiltonian monte carlo with variance reduction for bayesian inference”. In: *Machine Learning* 108.8, pp. 1701–1727.

Liang, Faming (2007). “Continuous contour Monte Carlo for marginal density estimation with an application to a spatial statistical model”. In: *Journal of Computational and Graphical Statistics* 16.3, pp. 608–632.

— (2009). “On the use of stochastic approximation Monte Carlo for Monte Carlo integration”. In: *Statistics & Probability Letters* 79.5, pp. 581–587.

— (2014). “An overview of stochastic approximation Monte Carlo”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 6.4, pp. 240–254.

Liang, Faming, Chuanhai Liu and Raymond Carroll (2011). *Advanced Markov chain Monte Carlo methods: learning from past samples.* Vol. 714. John Wiley & Sons.

Liang, Faming, Chuanhai Liu and Raymond J Carroll (2007). “Stochastic approximation in Monte Carlo computation”. In: *Journal of the American Statistical Association* 102.477, pp. 305–320.

Liang, Faming and Mingqi Wu (2013). “Population SAMC vs SAMC: Convergence and applications to gene selection problems”. In: *Journal of Biometrics & Biostatistics.*

Luettich, Richard Albert and Joannes J Westerink (2004). *Formulation and numerical implementation of the 2D/3D ADCIRC finite element model version 44. XX.* Vol. 20. R. Luettich Chapel Hill, NC, USA.

Luo, Zhao Tang, Huiyan Sang and Bani K Mallick (2021a). “A Bayesian Contiguous Partitioning Method for Learning Clustered Latent Variables.” In: *J. Mach. Learn. Res.* 22, pp. 37–1.

— (2021b). “A Nonstationary Soft Partitioned Gaussian Process Model via Random Spanning Trees”.

Ma, Yi-An, Tianqi Chen and Emily B Fox (2015). “A complete recipe for stochastic gradient MCMC”. In: *arXiv preprint arXiv:1506.04696.*

Ma, Pulong et al. (2019). “Multifidelity computer model emulation with high-dimensional output: An application to storm surge”. In: *arXiv preprint arXiv:1909.01836*.

Marin, Jean-Michel et al. (2014). “Relevant statistics for Bayesian model choice”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76.5, pp. 833–859.

Marjoram, Paul et al. (2003). “Markov chain Monte Carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences* 100.26, pp. 15324–15328.

McKay, Michael D, Richard J Beckman and William J Conover (2000). “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code”. In: *Technometrics* 42.1, pp. 55–61.

Nash, J Eamonn and Jonh V Sutcliffe (1970). “River flow forecasting through conceptual models part I-A discussion of principles”. In: *Journal of hydrology* 10.3, pp. 282–290.

Neal, Radford M (2005). “Taking bigger Metropolis steps by dragging fast variables”. In: *arXiv preprint math/0502099*.

Neal, Radford M et al. (2011). “MCMC using Hamiltonian dynamics”. In: *Handbook of markov chain monte carlo* 2.11, p. 2.

Nemeth, Christopher and Paul Fearnhead (2020). “Stochastic gradient markov chain monte carlo”. In: *Journal of the American Statistical Association*, pp. 1–18.

Nunes, Matthew A and David J Balding (2010). “On optimal selection of summary statistics for approximate Bayesian computation”. In: *Statistical applications in genetics and molecular biology* 9.1.

Paciorek, Christopher and Mark Schervish (2003). “Nonstationary covariance functions for Gaussian process regression”. In: *Advances in neural information processing systems* 16.

Ponce, Joan et al. (2019). “Assessing the effects of modeling the spectrum of clinical symptoms on the dynamics and control of Ebola”. In: *Journal of theoretical biology* 467, pp. 111–122.

Prim, Robert Clay (1957). “Shortest connection networks and some generalizations”. In: *The Bell System Technical Journal* 36.6, pp. 1389–1401.

Robert, Christian P, George Casella and George Casella (1999). *Monte Carlo statistical methods.* Vol. 2. Springer.

Robert, Christian P, Jean-Marie Cornuet et al. (2011). “Lack of confidence in approximate Bayesian computation model choice”. In: *Proceedings of the National Academy of Sciences* 108.37, pp. 15112–15117.

Roberts, Gareth O and Osnat Stramer (2002). “Langevin diffusions and Metropolis-Hastings algorithms”. In: *Methodology and computing in applied probability* 4.4, pp. 337–357.

Rudolf, Daniel and Nikolaus Schweizer (2018). “Perturbation theory for Markov chains via Wasserstein distance”. In: *Bernoulli* 24.4A, pp. 2610–2639.

Sherlock, Chris et al. (2015). “On the efficiency of pseudo-marginal random walk Metropolis algorithms”. In: *The Annals of Statistics* 43.1, pp. 238–275.

Silk, Daniel, Sarah Filippi and Michael PH Stumpf (2013). “Optimizing threshold-schedules for sequential approximate Bayesian computation: applications to molecular systems”. In: *Statistical applications in genetics and molecular biology* 12.5, pp. 603–618.

Simola, Umberto et al. (2021). “Adaptive approximate Bayesian computation tolerance selection”. In: *Bayesian analysis* 16.2, pp. 397–423.

Sisson, Scott A, Yanan Fan and Mark Beaumont (2018). *Handbook of approximate Bayesian computation*. Chapman and Hall/CRC.

Sisson, Scott A, Yanan Fan and Mark M Tanaka (2007). “Sequential monte carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences* 104.6, pp. 1760–1765.

Tanaka, Mark M et al. (2006). “Using approximate Bayesian computation to estimate tuberculosis transmission parameters from genotype data”. In: *Genetics* 173.3, pp. 1511–1520.

Vihola, Matti and Jordan Franks (2020). “On the use of approximate Bayesian computation Markov chain Monte Carlo with inflated tolerance and post-correction”. In: *Biometrika* 107.2, pp. 381–395.

Welling, Max and Yee W Teh (2011). “Bayesian learning via stochastic gradient Langevin dynamics”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, pp. 681–688.

WHO (2014). *WHO, Ebola Situation Reports.* http://apps.who.int/ebola/ebola-situation-reports.

Wikle, Christopher K and Noel Cressie (1999). “A dimension-reduced approach to space-time Kalman filtering”. In: *Biometrika* 86.4, pp. 815–829.

Wilkinson, Richard David (2013). “Approximate Bayesian computation (ABC) gives exact results under the assumption of model error”. In: *Statistical applications in genetics and molecular biology* 12.2, pp. 129–141.

Williams, Christopher KI and Carl Edward Rasmussen (2006). *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA.

Wood, Simon N (2010). “Statistical inference for noisy nonlinear ecological dynamic systems”. In: *Nature* 466.7310, pp. 1102–1104.

Zijlema, M (2010). “Computation of wind-wave spectra in coastal waters with SWAN on unstructured grids”. In: *Coastal Engineering* 57.3, pp. 267–277.