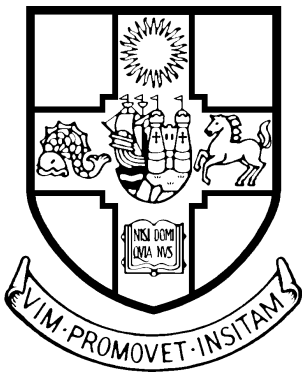
A Particle Markov Chain Monte Carlo Approach to Coalescent Inference



Patrick Cannon

School of Mathematics

University of Bristol

November 2018

A dissertation submitted to the University of Bristol
in accordance with the requirements for award of the degree of
Doctor of Philosophy in the Faculty of Science

60,000 words

Abstract

For some years Markov chain Monte Carlo (MCMC) has been regarded as a central tool in computational statistics. Its principle shortcoming is that it requires evaluation of a likelihood term, which is commonly infeasible in practice for reasons of intractability or high-dimensionality. Recent work on the so-called pseudo-marginal method [1, 2] has largely overcome this difficulty in settings where a positive, unbiased estimate of the likelihood is available. For state space models such an estimate is routinely calculated using sequential Monte Carlo methods. In combination the approach is termed particle Markov chain Monte Carlo (PMCMC).

This thesis is principally concerned with developing a PMCMC method suitable for the sequentially Markov coalescent (SMC) [3, 4], a model in population genetics for the ancestry of a set of sequences taken from a population. It is shown that the SMC can be recast as a piecewise deterministic Markov process and a novel particle filter is proposed for the case of two and three individuals. In particular the case of three individuals introduces a number of challenges that require bespoke solutions. These particle filter methods are then used in the context of a PMCMC algorithm to infer parameters of the biological model. Finally, backwards sampling is implemented in a non-trivial setting, and subsequently linked to very recent work [5] on pseudo-marginal algorithms that can scale well with the dimension of the problem, compared to traditional approaches.

Acknowledgements

I would like to thank my supervisors Christophe Andrieu and Mark Beaumont for their patience and kindness over the last four years. It has been incredibly rewarding to have worked with you both and I consider it a privilege.

I would like to express my gratitude to Jonty Rougier, for much wisdom and good humour over the years. I thank also Nick Whiteley, my first research supervisor, for an early push in the right direction.

I thank Sophie, Kathryn, Justin, Bertie, Tom, Leonie, Ailsa, Sam, Michael and all my other friends and colleagues in Bristol for making the journey more fun than it had any right to be.

I thank Brandon and Tom for being such pillars - sorry, pillocks. Here's to the next 10.

To my father, Nigel - thank you for giving so much towards my education. I couldn't have done without your support, then and now.

Finally, I thank Elizabeth, who has looked on tempests and was never shaken.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Contents

List of Figures

Chapter 1

Introduction

In almost every application imaginable, data is more freely available, and on a larger scale, than it has ever been. With this surfeit of information has come a desire for more realistic, faster, and better performing models and computational methods. Yet as more sophisticated models become commonplace we are faced with fresh challenges, one of which is the presence of intractable likelihoods. Such intractability typically arises in complex, high-dimensional models dependent on a latent state that is impractical or impossible to marginalise analytically.

A notable case is the field of population genetics. Experimental advances of the last twenty years have increased by orders of magnitude the quantity and quality of genome wide data, revolutionising the discipline. However, the complex, high-dimensional nature of the stochastic models underlying many frameworks for understanding reproduction prohibits the use of most simple statistical approaches. A celebrated model in population genetics is *the coalescent*, an elegant simulation method for the ancestral history of a sample of individuals from a population. Here, the problems of dimensionality and intractability are ever present. One would like to find the posterior distributions of the parameters of the stochastic model underlying the coalescent, but this is only possible when the likelihood can be computed. Since to do so would require marginalising a vast latent space of genealogies, the problem becomes extraordinarily computationally demanding.

One of the most promising strategies of the last decade has been 'exact approximate' methods, introduced in [1] and later formalised in [2], where it was termed the *pseudo-marginal* approach. The idea is as simple as replacing the intractable likelihood in a Metropolis-Hastings algorithm with a 'good' estimate. Provided the estimate is positive and unbiased (with respect to the true likelihood) then not only is one justified in using it, but the resulting algorithm is, somewhat miraculously, exact. Prior to this discovery, sequential Monte Carlo methods, sometimes termed particle filters, had become a popular approach for inference in high-dimensional state space models. Particle filters are capable of producing an unbiased

estimate of the likelihood, and a connection was soon established. When the likelihood estimate in the pseudo-marginal algorithm is provided by a particle filter, the resulting method is termed particle Markov chain Monte Carlo (PMCMC) and was first presented in [6].

This thesis aims to develop an original particle Markov chain Monte Carlo approach to inference under the coalescent model. Chapters 2 and 3 are concerned with developing a strategy to mitigate the worst behaviour of pseudo-marginal algorithms, and proving results about its performance. The remainder of the thesis is devoted to Monte Carlo methods for coalescent inference. Chapter 4 introduces some models from population genetics, including the coalescent. In Chapter 5 a particle filter approach is described for the sequentially Markov coalescent model, for two and three individuals. The setting is challenging as it is not easily described by a standard state space model and the observations are only weakly informative of the latent state. Finally, the algorithm developed is placed in the context of a PMCMC method and in Chapter 6 this is further developed through so-called backwards sampling to allow application of a recent, state-of-the-art, high-dimensional MCMC method. It is hoped that with further work a general and flexible approach of the type offered by PMCMC could prove useful to practitioners.

Chapter 2

Rejuvenation within pseudo-marginal Metropolis-Hastings algorithms

This chapter explores variations of pseudo-marginal Metropolis-Hastings (PsMMH) algorithms into which ancillary mechanisms have been built to prevent *sticking* of the Markov chain. A novel algorithm, termed λ-PsMMH, is proposed and shown in the main result to have greater efficiency than standard PsMMH. Firstly a brief survey of popular implementations of PsMMH algorithms is presented, along with a running example. Markov chain Monte Carlo performance measures are discussed and placed in the context of pseudo-marginal algorithms. Existing strategies to improve the performance of several PsMMH algorithms are explored and contrasted with the *rejuvenation* strategy at the heart of λ-PsMMH. Computer simulations are carried out, indicating that the qualitative asymptotic variance ordering results established can have impact in the field.

2.1 Algorithm descriptions

2.1.1 The Metropolis-Hastings algorithm

Following the notation of [2, 7] let π be a probability distribution, defined on a measurable space $(\mathsf{X}, \mathcal{B}(\mathsf{X}))$, from which one is interested in sampling. Suppose that it is infeasible to generate samples directly, for example due to its complexity or high-dimensionality, but that it is possible to evaluate the target distribution π point-wise. A traditional approach in Statistics on such occasions is to implement Markov chain Monte Carlo (MCMC) methods, a flexible, powerful sub-class of which are Metropolis-Hastings (MH) algorithms.

MH algorithms offer a technique of constructing a regular, reversible Markov kernel with a desired stationary distribution - in practice chosen to be one's target distribution. A defining

facet of the MH algorithm is its 'acceptance ratio', that is the probability with which a new state, proposed according to a proposal kernel q, is taken to be the next step in the Markov chain. Taking the current and proposed states to be $x, y \in \mathsf{X}$, this probability is: [8–10]

$$\alpha(x,y) := \min\left\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\right\} = \min\{1, r(x,y)\} \tag{2.1}$$

where the meaning of $r(x,y)$ is implicit. At each step of the algorithm the chain moves either to a new state through an accepted proposal else it remains in the same state through a rejected proposal. Thus the associated Markov kernel is given [7] by

$$P(x,\mathrm{d}y) := \alpha(x,y)q(x,\mathrm{d}y) + \delta_x(\mathrm{d}y)\rho(x)$$

where $\rho(x) := 1 - \int \alpha(x,y)q(x,\mathrm{d}y)$ is the probability of rejecting a proposed move to any state. We note that the MH kernel is reversible with respect to the target since it preserves detailed balance. For completeness this is briefly demonstrated. Ignoring inconsequentially the diagonal component, detailed balance is established directly as follows, where both sides of the equation are understood to be measures on the product space $(\mathsf{X}\times\mathsf{X}, \mathcal{B}(\mathsf{X})\times\mathcal{B}(\mathsf{X}))$,

$$\begin{aligned}\pi(\mathrm{d}x)\alpha(x,y)q(x,\mathrm{d}y) &= \min\{\pi(\mathrm{d}x)q(x,\mathrm{d}y), \pi(\mathrm{d}y)q(y,\mathrm{d}x)\} \\ &= \pi(\mathrm{d}y)\alpha(y,x)q(y,\mathrm{d}x)\end{aligned}$$

and a simple substitution of this identity suffices to show that for any $\mathsf{A} \in \mathcal{B}(\mathsf{X})$,

$$\int_{\mathsf{X}} \pi(\mathrm{d}x)P(x,\mathsf{A}) = \pi(\mathsf{A}).$$

We stress here that we require from the proposal kernel q that we can sample from it, and that it can be evaluated pointwise up to a multiplicative constant. Outside of these requirements any q is permissible, though a judicious choice of proposal kernel is crucial to the efficiency of the algorithm. In summary, the MH algorithm produces a reversible Markov chain by virtue of which the correct stationary distribution is targeted. Reversibility is clearly a sufficient condition for stationarity, and has been widely adopted for this reason, but it is not necessary. In line with the literature we will in this report refer to the MH algorithm presented here as the *marginal algorithm*. Algorithm 2.1 provides instructions for performing a single iteration of the marginal algorithm, taking $x \in \mathsf{X}$ to be the current state.

A toy example is now introduced that will act as the basis for many of the comparisons and explanations in this project. It is natural to explore stochastic algorithms in the first instance empirically, yet often their inherent stochasticity can lead to a frustrating lack of robustness in

Algorithm 2.1 Marginal algorithm: simulating from $P(x,\cdot)$

Input: current state $x \in \mathsf{X}$.

1. Sample $Y \sim q(x,\cdot)$.

2. With probability $\alpha(x,Y)$ given in (2.1):

   set $x' \leftarrow Y$,

   otherwise:

   set $x' \leftarrow x$.

Output: new state $x' \in \mathsf{X}$.

output diagnostic data or output statistics of the chain. Moreover, as we shall see, this problem presents a 'catch 22' - some of the most compelling questions in computational statistics relate to improving inference in highly complex settings, for instance where high dimensionality or intractable likelihoods are a feature, and it is in precisely such settings that computational experiments are most fraught with misleading noise and artifacts. In order, then, to allow as few artifacts as possible to distract our inference, and so to give us greater confidence in our results, we have chosen to use very basic models for our examples in this section.

2.1.2 Example: random walk Metropolis Hastings

In light of the considerations above, our primary example throughout this chapter is the one-dimensional target density $\pi(\cdot) = \mathcal{N}(\cdot \mid 0,1)$. Often in practice when using 'off the shelf' MCMC algorithms the choice of proposal distribution is a methodological bottleneck. Fortunately there exist several canonical forms of proposal distribution which are simple to implement and produce MH algorithms with well-studied properties. One such class is *random walk MH* algorithms (see e.g. [11]), wherein the proposal density q can be written as $q(x,y) = q(y-x)$, for any $x,y \in \mathsf{X}$. In particular, we will use a random walk with Gaussian innovations, that is $q(x,\cdot) = \mathcal{N}\left(\cdot \mid x, \sigma^2\right)$, indexed by a single scale parameter $\sigma \in [0,\infty)$. Any tuning of our proposal distribution will be performed by manipulating this scale parameter.

Figure 2.1 displays three typical MCMC diagnostic plots associated to a MH algorithm. In particular for this example $M = 5 \times 10^5$ iterations were used and a proposal standard deviation of $\sigma = 5$ was chosen. The trace plot and estimated density are standard, and the plot of estimated autocorrelations is produced using R's [12] built-in autocorrelation function (ACF). A 'mean acceptance rate' of the algorithm is given simply by the number of novel proposals

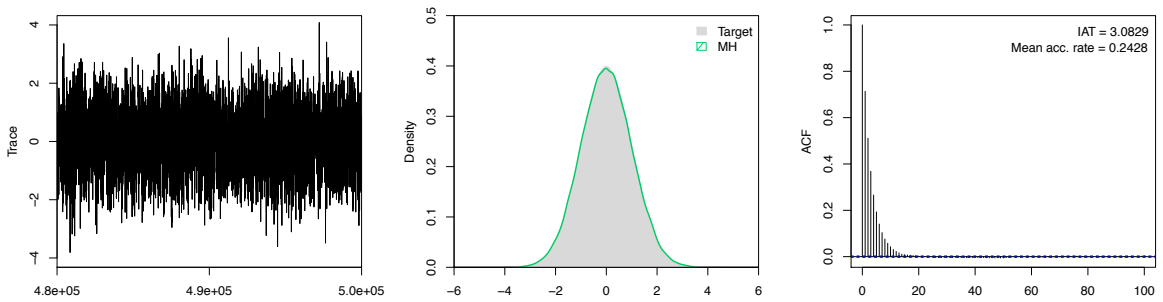


Figure 2.1: *Diagnostic plots for a Random Walk MH algorithm. From left to right they are a trace plot of the last* 2×10^4 *iterations of the Markov chain, an estimated density of the chain, and a plot of its estimated autocorrelations to a lag of 100 steps.*

accepted as a proportion of the total number of iterations, and the integrated autocorrelation time (IAT), discussed in depth in Section 2.2, is found using a function based on a method due to Sokal (see [13] and the implementation [14]).

2.1.3 The pseudo-marginal Metropolis-Hastings algorithm

In practice π may be intractable or prohibitively costly to evaluate, precluding evaluation of the acceptance ratio (2.1) and ruling out use of the marginal algorithm. Recent work [1, 2] motivated by problems in population genetics has sought to overcome this issue in settings where unbiased estimates of point evaluations of the target are available. That is, for any $x \in \mathsf{X}$, one has access to nonnegative estimates $\hat{\pi}(x)$ satisfying $\mathbb{E}[\hat{\pi}(x)] = \pi(x)$, where this expectation is taken with respect to the collection of random variables ψ implicitly used in computing the estimate $\hat{\pi}(x) = \hat{\pi}(x;\psi)$.

One intuitive approach is to exploit a law of large numbers, sampling independently $N \in \mathbb{N}^*$ estimates $\{\hat{\pi}(x)^{(i)}\}_{i=1}^N$ for every proposed state x, and substituting in the acceptance ratio of the marginal algorithm the estimate $N^{-1}\sum_i \hat{\pi}(x)^{(i)} \approx \pi(x)$. Certainly, to build on this idea, one can arbitrarily reduce the distance of the estimate from the true value by increasing N, so it is reasonable to think that by increasing N this 'averaged' algorithm can in some sense approach the 'exact' algorithm. In fact this intuition is well-founded and moreover, rather miraculously, these so-called *pseudo-marginal algorithms* can be shown to be exact for *any* $N \geq 1$. We now justify this claim, taking $N = 1$ without loss of generality.

A framework for describing and analysing pseudo-marginal Metropolis Hastings (PsMMH) algorithms is given by [2, 7] and we follow this notation. It is fruitful to write the unbiased estimates as their true value with multiplicative noise, i.e. $\hat{\pi}(x) = W_x \pi(x)$, where $W_x \sim Q_x(\cdot) \geq 0$ and $\mathbb{E}[W_x] = 1$ for any $x \in \mathsf{X}$. For definiteness we take the family of proba-

bility measures $\{Q_x\}_{x\in\mathsf{X}}$ to be defined on the measurable space $(\mathbb{R}_+, B(\mathbb{R}_+))$. We proceed by considering an augmented distribution on the product space $(\mathsf{X}\times\mathbb{R}_+, \mathcal{B}(\mathsf{X})\times\mathcal{B}(\mathbb{R}_+))$, given by $\tilde{\pi}(\mathrm{d}x,\mathrm{d}w) := \pi(\mathrm{d}x)\pi_x(\mathrm{d}w)$ where $\pi_x(\mathrm{d}w) := Q_x(\mathrm{d}w)w$. It is clear that $\tilde{\pi}$ has π as a marginal distribution since $\int \tilde{\pi}(x,\mathrm{d}w) = \pi(x)$. Let us imagine then a standard MH algorithm targeting $\tilde{\pi}$ with proposal kernel $\tilde{q}(x,w;\mathrm{d}y,\mathrm{d}u) := q(x,\mathrm{d}y)\,Q_y(\mathrm{d}u)$. The acceptance ratio of this algorithm is

$$\begin{aligned}\tilde{\alpha}(x,w;y,u) &= \min\left\{1, \frac{\tilde{\pi}(\mathrm{d}y,\mathrm{d}u)\tilde{q}(y,u;\mathrm{d}x,\mathrm{d}w)}{\tilde{\pi}(\mathrm{d}x,\mathrm{d}w)\tilde{q}(x,w;\mathrm{d}y,\mathrm{d}u)}\right\} \qquad (2.2)\\ &= \min\left\{1, \frac{\pi(\mathrm{d}y)Q_y(\mathrm{d}u)uq(y,\mathrm{d}x)\,Q_x(\mathrm{d}w)}{\pi(\mathrm{d}x)Q_x(\mathrm{d}w)wq(x,\mathrm{d}y)\,Q_y(\mathrm{d}u)}\right\}\\ &= \min\left\{1, \frac{\pi(\mathrm{d}y)q(y,\mathrm{d}x)}{\pi(\mathrm{d}x)q(x,\mathrm{d}y)}\frac{u}{w}\right\}\\ &= \min\left\{1, r(x,y)\frac{u}{w}\right\}. \qquad (2.3)\end{aligned}$$

In simplifying the acceptance ratio we find it is exactly that of the "noisy" version of the marginal algorithm, the pseudo-marginal algorithm, which can be seen by comparing this expression to that of the marginal algorithm, which is given in Equation (2.1). Here it is germane to emphasise that in describing the pseudo-marginal algorithm as noisy we make use of a heuristic. The pseudo-marginal algorithm, like the marginal algorithm, is exact; indeed we have shown it is a standard MH algorithm marginally targeting π. Finally, the Markov kernel associated to the algorithm is

$$\tilde{P}(x,w;\mathrm{d}y,\mathrm{d}u) := \tilde{\alpha}(x,w;y,u)\tilde{q}(x,w;\mathrm{d}y,\mathrm{d}u) + \delta_{x,w}(\mathrm{d}y,\mathrm{d}u)\,\tilde{\rho}(x,w)$$

where, analogously to the marginal algorithm, the probability of rejection is

$$\tilde{\rho}(x,w) := 1 - \iint \tilde{\alpha}(x,w;y,u)\tilde{q}(x,w;\mathrm{d}y,\mathrm{d}u).$$

There are few complications in extending this argument to averages of multiple unbiased estimates. Suppose we average N unbiased estimates at each step of the algorithm, giving an estimate of the target, for any $x\in\mathsf{X}$, of $N^{-1}\sum_i W_i\pi(x)$ where $W_{1:N} := (W_1,\ldots,W_N) \sim Q_x^N$, and Q_x^N is a probability measure on $\mathbb{R}_+^N$. Note that throughout the project we will make use of the rather loose terminology 'weight' or 'weights' to describe these unbiased draws. Then, analogously to the $N=1$ case, we can think of the pseudo-marginal algorithm using these averages as a MH algorithm targeting $\tilde{\pi}^N(\mathrm{d}x,\mathrm{d}w_{1:N}) := \pi(\mathrm{d}x)\pi_x^N(\mathrm{d}w_{1:N})$ where $\pi_x^N(\mathrm{d}w_{1:N}) := Q_x^N(\mathrm{d}w_1,\ldots,\mathrm{d}w_N)N^{-1}\sum_i w_i$, which still has π as a marginal distribution. The transition kernel of the N-weight version will in general hereafter be denoted $\tilde{P}_N$. Taking (x,w)

to be the current state, for some $x \in \mathsf{X}$ and $w \in \mathbb{R}_+^N$, instructions for performing one step of the PsMMH algorithm are now given.

Algorithm 2.2 Pseudo-Marginal algorithm: simulating from $\tilde{P}_N(x, w; \cdot)$

Input: current state (x, w).

1. Sample $Y \sim q(x, \cdot)$.
2. Sample $U \sim Q_Y^N(\cdot)$.
3. With probability $\tilde{\alpha}(x, w; Y, U)$ given in 2.2:
   set $(x', w') \leftarrow (Y, U)$,
   otherwise:
   set $(x', w') \leftarrow (x, w)$.

Output: new state (x', w').

We have chosen to leave the algorithms in an abstract form for clear exposition, however it is helpful to remember that, for example, step 2 of Algorithm 2.2 corresponds in real terms to drawing N unbiased estimates $\{\hat{\pi}(Y)\}_{i=1}^N$ and step 3 simply corresponds to substituting their average into the MH ratio (2.1) in place of the true target (as in Equation (2.3)).

Results concerning the inheritance of ergodicity of the marginal algorithm by the pseudo-marginal algorithm are now briefly discussed; ergodicity is defined and discussed more at the start of Section 2.2. Several important cases are presented in [7], including a theorem in which it is shown that the pseudo-marginal algorithm maintains polynomial ergodicity when the marginal algorithm is a random walk Metropolis and the weights satisfy a moment-bounding condition. It is also suggested as a general principle that geometric ergodicity is inherited wherever the weights are uniformly bounded.

2.1.4 Example: PsMMH

Suppose we are again in the setting described in Example 2.1.2 but that it is now impossible to sample directly from the target distribution π. Instead we have access only to a black box capable of providing, for any $x \in \mathsf{X}$, unbiased estimates of $\pi(x)$.

A number of modelling choices must be made before we can implement and test a PsMMH algorithm in this scenario. Firstly, as is common (e.g. [15, Assumption 2] and [16, Assumption 4]), we take the log-target distribution to be subject to additive Gaussian noise so that

$Q_x(\cdot) = \log \mathcal{N}\left(\cdot \mid -\gamma^2/2, \gamma^2\right)$ for some variance parameter γ^2. Using the log-normal assumption guarantees that $Q_x(W_x \geq 0) = 1$ holds, and the given relation between the parameters satisfies the requirement $\mathbb{E}[W_x] = 1$. It is noted in [16], and reiterated here, that supposing the weight W_x to be distributed independently of the point x at which it is generated is simplistic. It is however in line with our aim to conduct experiments in their most essential setting.

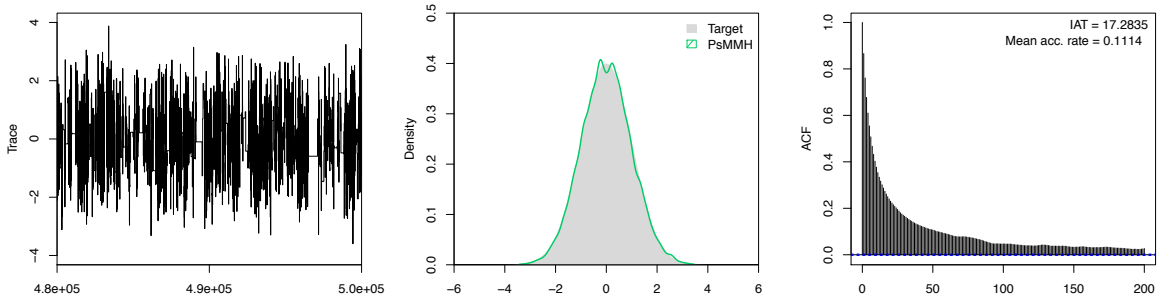


Figure 2.2: *Diagnostic plots for a Random Walk PsMMH algorithm. From left to right they are a trace plot of the last 2×10^4 iterations of the Markov chain, an estimated density of the chain, and a plot of its estimated autocorrelations to a lag of 200 steps. Note the adjusted lag scale in the ACF plot.*

In Figure 2.2 the log-noise variance was set to $\gamma^2 = 3$, the proposal standard deviation was $\sigma = 3$, and again $M = 5 \times 10^5$ iterations were used overall. For simplicity only $N = 1$ estimate was simulated at each step. Note that a smaller proposal standard deviation was chosen for this experiment as the value $\sigma = 5$ used in Example 2.1.2 was larger than is optimal for the current example; as one would expect, it is beneficial, broadly speaking, to share the variance across the weights and the proposal density. It is clear from the plots that the implicit noise introduced into the algorithm (through the use of unbiased estimates) has significantly reduced its efficiency. Indeed intervals in which the algorithm is 'stuck' are clearly visible in the trace plot, and this in turn is reflected in the diminished mean acceptance rate and inflated autocorrelations. It is the central aim of this chapter to explore strategies to mitigate precisely these issues - this process begins in earnest in Section 2.2.

2.1.5 Monte Carlo within Metropolis

A second possible implementation of the pseudo-marginal approach, namely *Monte Carlo within Metropolis (MCWM)* or *Noisy Metropolis Hastings*, has also been studied [2, 17]. This algorithm is superficially similar to the PsMMH algorithm, however in contrast the weight associated to the current state is not carried from one step to the next, rather it is recalculated at every step. Thus the algorithm constructs a Markov chain on $(\mathsf{X}, \mathcal{B}(\mathsf{X}))$ in which all the

auxiliary variables required are sampled within each step and only the state is recorded. Instructions for performing a single iteration are given in Algorithm 2.3, taking $x \in \mathsf{X}$ to be the current state.

Algorithm 2.3 Monte Carlo within Metropolis algorithm: simulating from $P^{\mathrm{MCWM}}(x,\cdot)$

Input: current state $x \in \mathsf{X}$.

1. Sample $Y \sim q(x,\cdot)$.
2. Sample independently $W \sim Q_x^N(\cdot)$ and $U \sim Q_Y^N(\cdot)$.
3. With probability $\tilde{\alpha}(x,W;Y,U)$ given in (2.2):

   set $x' \leftarrow Y$,

   otherwise:

   set $x' \leftarrow x$.

Output: new state x'.

To distinguish the acceptance ratio of the MCWM algorithm from that of the PsMMH algorithm, we can write its acceptance ratio as

$$\alpha_N^{\mathrm{MCWM}}(x,y) := \mathbb{E}_{Q_x^N \otimes Q_y^N}\left[\tilde{\alpha}(x,W;y,U)\right],$$

using the notation of [17]. It is then possible to write the transition kernel as

$$P^{\mathrm{MCWM}}(x,\mathrm{d}y) := \alpha^{\mathrm{MCWM}}(x,y)\, q(x,\mathrm{d}y) + \delta_x(\mathrm{d}y)\, \rho^{\mathrm{MCWM}}(x)$$

where $\rho^{\mathrm{MCWM}}(x) := 1 - \int_{\mathsf{X}} \alpha^{\mathrm{MCWM}}(x,y)\, q(x,\mathrm{d}y)$. Note that we have dropped the subscript N for notational convenience. One drawback associated with the MCWM algorithm is that, by recalculating all auxiliary variables at each step, it fails to achieve invariance under the target distribution π. Indeed it may not have an invariant distribution at all - there are cases for which the marginal and pseudo-marginal chains exhibit geometric ergodicity but the corresponding MCWM chain is transient [17]. Monte Carlo within Metropolis has received less attention than the pseudo-marginal method of Section 2.1.3 and less is known about its stability properties, though some interesting results are available; two such results are now summarised. In the epilogue of [2] it is shown that when the marginal algorithm is uniformly ergodic and the MCWM chain is invariant with respect to some probability distribution $\breve{\pi}_N$ then, under some reasonable conditions on the weights, the MCWM chain is also uniformly ergodic with

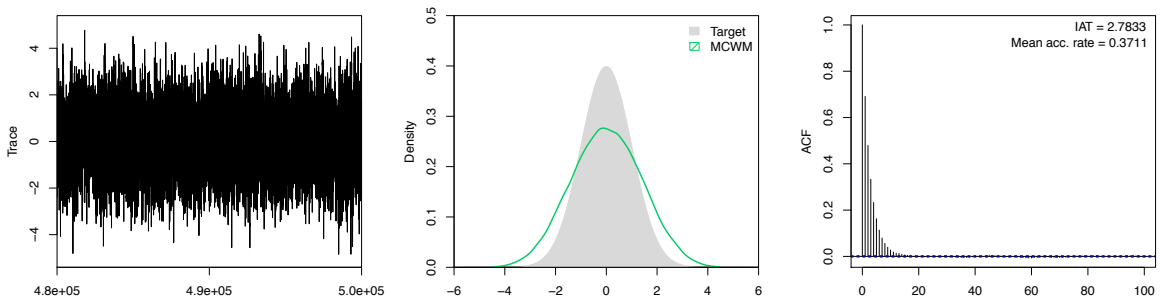


Figure 2.3: *Diagnostic plots for a Random Walk MCWM algorithm. From left to right they are a trace plot of the last* 2×10^4 *iterations of the Markov chain, an estimated density of the chain, and a plot of its estimated autocorrelations to a lag of 100 steps.*

respect to $\check{\pi}_N$, and moreover one can upper-bound the distance between $\check{\pi}_N$ and the true target. Motivated by scenarios in which the marginal chain achieves at best geometric ergodicity, [17] finds that when the marginal algorithm is geometrically ergodic then, under some conditions on the weights, and for large enough N, the MCWM algorithm with transition kernel P_N^{MCWM} is also geometrically ergodic (which is sufficient to guarantee it has an invariant distribution $\check{\pi}_N$). They find in addition under similar constraints on the weights that $\check{\pi}_N$ converges in total variation towards the true target, with tractable rates of convergence available in some settings.

2.1.6 Example: MCWM

Continuing from Example 2.1.4, we now run an MCWM algorithm with $N = 1$ weight at every step. Once again the parameters $\gamma^2 = 3$, $\sigma = 3$, and $M = 5 \times 10^5$ are used, and the result is recorded in Figure 2.3. As one might expect the algorithm attains a higher acceptance rate and lower IAT than the PsMMH algorithm, and the difference is profound. There is however a prominent bias in the chain which, in the context of conducting inference, may be undesirable. In short, the algorithm converges well to a distribution that is not the target. This reflects a key issue, that the N required to ensure that the MCWM algorithm converges to an invariant distribution, and further, an invariant distribution that is within an acceptable distance of the true target, could be intolerably large. In Section 2.3 we carry out computational experiments to study this ambiguity.

In practice the extent to which this bias matters depends on the object of inference; it may well be that the bias in the first order moment is reasonable for small N, but it may require a far larger N to guarantee similar accuracy in higher-order moments.

2.2 Improving performance

It is well known that in practice Markov chain Monte Carlo algorithms can require considerable expertise to tune and often converge or mix very slowly (move slowly around the support of the target distribution), especially in high dimensions. Poor mixing is of concern to the practitioner since to achieve ergodic averages of comparable quality to those of a more rapidly mixing chain, one must compensate by running the MCMC for more iterations [18]. To this end, a considerable body of literature is dedicated to the choice of proposal distribution, its structure having a profound impact on the success of the algorithm (see for example [19, 20]). In general, selecting a myopic proposal distribution, one making only very local moves, ensures a high probability of acceptance but also guarantees slow mixing of the chain. Conversely selecting a bold proposal distribution ensures more rapid mixing but guarantees a greater probability of rejection, resulting in 'stickiness', that is, a propensity of the chain to remain in the same state for many steps in a row. This trade-off is a fundamental factor in constructing the proposal distribution for a MH algorithm and there are some rules of thumb. For example, in the case of the random walk Metropolis Hastings algorithm, the received wisdom is to pick the variance of the proposal distribution to achieve an acceptance rate of roughly 0.44 in one dimension and 0.234 in the limit as the number of dimensions of the target distribution increases (e.g. [20, 21]).

Naturally the noisy estimates used in the acceptance ratio (2.2) of the pseudo-marginal algorithm exacerbate these problems since, in addition to the causes of poor mixing in the marginal algorithm, the pseudo-marginal algorithm suffers from additional stickiness induced by the stochasticity of the weights. To see this, consider the value of the acceptance probability

$$\tilde{\alpha}(x,w;y,u) = \min\left\{1, r(x,y)\,\frac{u}{w}\right\}$$

when the weight at the current step is $w = 50$, in other words the estimate of the target happens to be 50 times too large. This is entirely feasible; we require the weights to be positive, and they must satisfy $\mathbb{E}_{Q_x^N}[W_{1:N}] = 1$, but their variance is essentially unrestricted and may in practice cause difficulty. We see that the algorithm will struggle to move on from any state for which an overestimate of the target has been produced, and moreover the form of $\tilde{\alpha}$ encourages moves to such states, as they 'appear' to be states that are very likely under the target distribution. Before discussing possible solutions to these issues, we explore further performance measures for MCMC algorithms.

Two related concepts underpinning the performance of a Markov chain Monte Carlo algorithm are the rate of convergence and efficiency of estimation. One reason to distinguish between the two is that attempting to optimise both criteria can result in conflicting demands

on one's choice of Markov kernel, as noted in [22], which may suggest a switching strategy. We will see that the notion of statistical efficiency can be well described, for a process at equilibrium, by the 'asymptotic variance' of an ergodic average based on the chain. This is naturally a useful measure on its own, and a time-honoured strategy to compare two competing Markov kernels. In practice, however, it is generally not possible to start an MCMC algorithm at equilibrium since the target distribution is in principle unknown, so the rate at which the algorithm, from an arbitrary starting point, reaches equilibrium is also crucial.

Notation and definitions relevant to the performance of MCMC are now recalled (see, e.g., [10, 23]). Consider a Markov transition kernel Π with invariant distribution μ defined on a metric space $(\mathsf{E}, \mathcal{E})$ and let $\{\Phi_k; k \geq 0\}$ denote the Markov chain generated by Π from equilibrium, that is $\Phi_0 \sim \mu$. For a particular function $f : \mathsf{E} \to \mathbb{R}$ one typically makes use of the ergodic average $S_M := M^{-1} \sum_{k=1}^{M} f(\Phi_k)$ to estimate $\mu(f)$. Finally, we will use the standard notation $\mathsf{L}^2(\mu) := \{f : \mu(f^2) < \infty\}$ for the space of measurable, real-valued functions f that are square integrable with respect to μ.

Definition 1. *For $f \in \mathsf{L}^2(\mu)$ the asymptotic variance of the estimator S_M is*

$$v(f, \Pi) := \lim_{M \to \infty} M \operatorname{Var}_\mu (S_M) \in [0, \infty].$$

The asymptotic variance is a useful measure to compare the performance of Markov chains with the same invariant distribution.

Definition 2. *Taking ρ_k to be the correlation between the random variables $f(\Phi_0)$ and $f(\Phi_k)$, the integrated autocorrelation time is defined to be*

$$\tau(f, \Pi) := 1 + 2 \sum_{k=1}^{\infty} \rho_k.$$

A well known relationship exists between the asymptotic variance of the estimator and properties of the integrated autocorrelation time of the Markov chain. Whenever $\tau(f, \Pi)$ exists and is finite, it holds that

$$v(f, \Pi) = \sigma_f^2 \, \tau(f, \Pi) \in [0, \infty)$$

where $\sigma_f^2 := \mu f^2 - (\mu f)^2$, a proof of which can be found in Appendix A.1. In essence a factor of $\tau(f, \Pi)$ is incurred in the variance of the estimator as a penalty for using dependent samples rather than *iid* draws from the target distribution. There are connections here with the notion of *effective sample size* in the theory of Monte Carlo techniques. Were simulating directly from the target distribution feasible, the associated Monte Carlo average S_M would enjoy variance σ_f^2 / M. However, using auto-correlated MCMC samples the variance of the M-step average

becomes $\sigma_f^2 \tau(f,\Pi)/M = \sigma_f^2/(M/\tau(f,\Pi))$, thus by analogy the estimator is comparable in variance to one based on $M/\tau(f,\Pi)$ *iid* samples. In summary if two Markov chains with the same invariant distribution are both candidates for estimating some expectation of interest, the asymptotic variance of their respective estimators can help inform a choice between the two, at least when both chains start at equilibrium. Computationally, it will be useful to carry out this comparison using an estimate of the integrated autocorrelation time of the chain as a surrogate.

As discussed earlier, because the chain is unlikely to start from its invariant distribution it is crucial to consider the rate of convergence of the algorithm to equilibrium. Rates of convergence are well classified and we give now a brief overview, following for example [24], in which we describe several classes of chains in decreasing order of rate of convergence. Suppose $\{\Phi_k; k \geq 0\}$ is an ergodic Markov chain with invariant distribution μ, and that $\mathcal{L}_x(\Phi_n)$ denotes the probability law of the n-th state of the chain started from $x \in \mathsf{E}$. We will use $\| \mu \| := \sup_{|g|\leq 1} |\mu(g)|$ to denote the total variation distance of a signed measure μ [25]. Then the chain is said to be:

Uniformly ergodic if for some constants $V < \infty$ and $0 < r < 1$ it holds that

$$\| \mathcal{L}_x(\Phi_n) - \mu \| \leq V r^n$$

for all $x \in \mathsf{E}$ and $n \geq 1$.

Geometrically ergodic if for some function $V : E \to \mathbb{R}_+$ satisfying $\mu V < \infty$, and a constant $0 < r < 1$, it holds that

$$\| \mathcal{L}_x(\Phi_n) - \mu \| \leq V(x) r^n$$

for all $x \in \mathsf{E}$ and $n \geq 1$.

Polynomially ergodic if for some function $V : E \to \mathbb{R}_+$ satisfying $\mu V < \infty$, and a constant $\alpha > 0$ it holds that

$$\| \mathcal{L}_x(\Phi_n) - \mu \| \leq V(x) n^{-\alpha}$$

for all $x \in \mathsf{E}$ and $n \geq 1$.

Establishing rates of convergence is useful not just in comparing competing algorithms but also in guaranteeing certain central limit theorems are satisfied. A function f is said to satisfy a Central Limit Theorem (CLT) if the partial sums $M^{1/2}(S_M - \mu(f))$ converge weakly to a normal distribution (e.g. [25, 26]). Consequently, where CLTs hold we have an understanding of the error in an MCMC average S_M. The rates of convergence described above constitute convenient descriptions of sufficient conditions for some CLTs; we now give two well-known

examples to highlight their usefulness. If the Markov chain $\{\Phi_k; k \geq 0\}$ is uniformly ergodic with invariant distribution μ, then a CLT holds for all f satisfying $\mu(f^2) < \infty$, and in particular $M^{1/2}(S_M - \mu(f)) \to^d \mathcal{N}(0, v(f,\Pi))$. If the chain satisfies the weaker condition of geometric ergodicity then a CLT holds for all f with $\mu(|f|^{2+\delta}) < \infty$ for some $\delta > 0$. A plethora of similar results exist, each hingeing on slightly different requirements.

It is worth remarking that ergodicity is not the only criteria relating to CLTs that has been considered. A popular alternative is variance bounding (see [27]). A Markov kernel Π with unique stationary distribution μ is said to be *variance bounding* if there exists a constant $K < \infty$ such that $v(f,\Pi) \leq K \operatorname{Var}_\mu(f)$ for all $f : \mathsf{E} \to \mathbb{R}$. For reversible chains variance bounding is strictly weaker than geometric ergodicity, indeed geometric ergodicity implies variance bounding. Often, however, CLTs are satisfied even in the absence of geometric ergodicity - so a weaker concept is certainly justified. Variance bounding has a strong claim to being precisely the notion required, since to quote [27, Theorem 7], if Π is reversible, with unique stationary distribution μ, then Π is variance bounding if and only if every $f \in \mathsf{L}^2(\mu)$ satisfies a CLT for Π.

Strategies for improving the quality of the chain produced by a pseudo-marginal algorithm are now discussed. Our interests here are largely theoretical; computational tests and further discussion are deferred to Section 2.3.

2.2.1 Choosing N

Perhaps the most obvious method of improving the performance of a pseudo-marginal algorithm is increasing the number of weights averaged at each step. A reasonable intuition is to expect (perhaps monotonic) improvement in performance as N increases. Andrieu and Vihola [23] go some way towards proving this result, establishing that under certain conditions the asymptotic variance of a pseudo-marginal algorithm is non-increasing in N, and that the expected acceptance probability between any two states is non-decreasing in N. The result [23, Corollary 2.] is given here for completeness, after a number of definitions.

First define the simplex

$$\mathscr{S}_N := \{v \in [0,1]^N : \sum_{i=1}^N v(i) = 1\}$$

and suppose $W := \{W(1), \ldots, W(N)\}$ is a vector of exchangeable, non-negative, unit expectation random variables - analogous to the weights of Section 2.1.3. For two vectors $a, b \in \mathbb{R}^N$ we will use the inner product notation $(a,b) := \sum_{i=1}^N a(i)b(i)$, from which we define, for some $v \in \mathscr{S}_N$, the weight $W^{(v)} := (v, W)$ - analogous to an averaging of weights in Section 2.1.3,

though in this case not necessarily uniform averaging. The averaged weight $W^{(v)}$ is also non-negative with unit expectation, thus we can consider the PsMMH algorithm corresponding to this particular weight. Let us denote its transition kernel $\tilde{P}_v$. It is first shown [23, Corollary 2] that uniform weighting is optimal with respect to asymptotic variance. For uniform weighting we use the notation $u_k := (1/k, \cdots, 1/k, 0, \ldots, 0) \in \mathscr{S}_N$, the length-$N$ vector in which the first k components are $1/k$ and the rest are zero. The corollary concerns the asymptotic variance v, as described in Definition 1, and the conditional expected acceptance probability for a generic pseudo-marginal kernel $\tilde{P}$, $\alpha_{xy}(\tilde{P}) := \int_{\mathbb{R}_+^2} \tilde{\alpha}(x,w;y,u)\, Q_x(\mathrm{d}w) w Q_y(\mathrm{d}u)$. We now paraphrase the remainder of the corollary.

Corollary 1. *For any $x, y \in \mathsf{X}$ and $f \in \mathsf{L}^2(\pi)$, it holds that*

i) $k \longmapsto \alpha_{xy}\left(\tilde{P}_{u_k}\right)$ *is non-decreasing, and*

ii) $k \longmapsto v\left(f, \tilde{P}_{u_k}\right)$ *is non-increasing.*

Remark 1. The assumption of exchangeability on the weights is carried throughout this chapter and is standard [2, 23]. Its meaning is as follows: the set of weights $w_1, \ldots, w_N \sim Q_x^N(\cdot)$ is N-exchangeable when $\mathbb{E}[g(w_1, \ldots, w_N)] = \mathbb{E}[g(w_{\pi_1}, \ldots, w_{\pi_N})]$ for all bounded and continuous g, and all permutations $\{\pi_1, \ldots, \pi_N\}$ of $\{1, \ldots, N\}$. Loosely speaking, it is a claim that the labelling of the random variables conveys no information to us.

Thus a gain with respect to asymptotic variance is always derived from increasing the number of weights. Alas it is difficult to directly capitalise on such results - without a quantitative expression for the improvement it is impossible to determine in advance of experiments if there will be a net benefit to increasing N. In contrast recent work on Approximate Bayesian Computation MCMC [28] suggests there is little, if any, net benefit derived from using any number of weights $N > 1$, a claim that is also evaluated later in this chapter. One might expect to find results establishing whether or not a similar relation holds for the convergence rate of the algorithm as a function of N. To the best of our knowledge this is an open question; speculatively a positive result seems likely. Computational experiments described in Section 2.3.2 support this hypothesis.

A related, though distinct, question concerns ordering of the asymptotic variances of particle Markov chain Monte Carlo (PMCMC) algorithms (see [6]). Typically in a PMCMC algorithm a single unbiased estimate of the likelihood is used, that is $N = 1$, the variance of which is controlled by a new parameter m, the number of particles used in the particle filter at each step of the Markov chain. Recent work in this area [15, 16] has sought to provide guidelines for choosing a maximally efficient m, under the assumptions of Gaussian additive noise in the log-likelihood estimator, the variance of which is inversely proportional to m, and

where the distribution of the noise is independent of the current position. In [15] the optimal m, with respect to minimising upper bounds on computing time required to achieve a specified asymptotic variance, is found to be that which gives a log-likelihood standard deviation of "slightly greater than 1.0" (at least when the efficiency of the marginal algorithm is unknown). The result in [15] in fact holds for general pseudo-marginal algorithms, of which PMCMC are a class. Under a different criteria of efficiency, namely the expected squared jump distance in the limit as the number of dimensions of the target diverges to infinity, the optimal m is found in [16] to be that which gives a log-likelihood variance of 3.283 and an acceptance rate of around 7%. An interesting question, as yet unanswered to the best of our knowledge, is whether a result analogous to Corollary 1 holds for the number of particles m used in the particle filter; in other words, does every particle count in PMCMC?

2.2.2 Rejuvenation

Of primary concern in improving the efficiency of pseudo-marginal algorithms is preventing (entirely or with high probability) the weight, or average of weights, from bloating to such an extent that the algorithm regularly fails to change state for many consecutive steps. Increasing the number of weights N averaged at each step is one way in which this could be achieved. We now describe a process, introduced in [29], by which one can *rejuvenate*, or agitate, the set of weights $\{w_{1:N}\}$ of a given iteration in order to obtain a new set, crucially without disturbing detailed balance and spoiling the invariant distribution of the algorithm.

First notice that it is possible to expose a mixture structure in the target distribution $\tilde{\pi}^N$ of the algorithm:

$$\begin{aligned}\tilde{\pi}^N(x, w_{1:N}) &= \pi(x)\, Q_x^N(w_{1:N}) \frac{1}{N} \sum_{k=1}^{N} w_k \\ &= \sum_{k=1}^{N} \frac{1}{N} \pi(x)\, Q_x^N(w_{1:N}) w_k \\ &= \sum_{k=1}^{N} \mathring{\pi}(k, x, w_{1:N}) \end{aligned} \tag{2.4}$$

where $\mathring{\pi}(k, x, w_{1:N}) := \frac{1}{N} \pi(x)\, Q_x^N(w_{1:N}) w_k$. We now consider $\mathring{\pi}$ in its own right, noting that one can explicitly compute the full conditional distributions

$$\mathring{\pi}(k \mid x, w_{1:N}) = \frac{w_k}{\sum_{j=1}^{N} w_j}$$

and

$$\mathring{\pi}\left(w_{-k} \mid k, x, w_k\right) = Q_x^N\left(w_{-k} \mid w_k\right)$$

for any $k \in \{1, \ldots, N\}$. If we have access to the second full conditional, which is certainly the case for independent weights, then we are able to perform Gibbs updates which will naturally preserve $\mathring{\pi}$. In the seminal paper [6], analogous Gibbs updates on an extended state-space were termed *conditional sequential Monte Carlo* (cSMC).

Algorithm 2.4 Pseudo-Marginal algorithm with rejuvenation

Input: current state (x, w).

1. Sample $Y \sim q(x, \cdot)$.
2. Sample $U \sim Q_Y^N(\cdot)$.
3. *Rejuvenation step*:
   - Sample $k \sim \mathcal{P}(w_1, \ldots, w_N)$
   - Resample $w_{-k} \sim Q_x^N(\cdot \mid w_k)$
4. With probability $\tilde{\alpha}(x, w; Y, U)$ given in 2.2:

   set $(x', w') \leftarrow (Y, U)$,

   otherwise:

   set $(x', w') \leftarrow (x, w)$.

Output: new state (x', w').

For definiteness, we will review the technique step by step. Each iteration of the pseudo-marginal algorithm starts with a pair $(x, w_{1:N})$ distributed (approximately) according to $\tilde{\pi}^N$. We then recast our target $\tilde{\pi}^N$ as a marginal distribution of another distribution $\mathring{\pi}$ on an extended state space $(k, x, w_{1:N})$. Gibbs steps are then performed, updating (k, w_{-k}), using as proposals the full conditional distributions derived above. Marginalising over k, one is left with an updated pair $(x, u_{1:N})$. Therefore, for a given $x \in \mathsf{X}$ and weights $w_{1:N}$, the *rejuvenation kernel*, i.e. the probability kernel of the described weight updates, is given by

$$R_N\left(x, w_{1:N}; x', \mathrm{d}u_{1:N}\right) := \sum_{k=1}^{N} \mathring{\pi}\left(k \mid x, w_{1:N}\right) \mathring{\pi}\left(u_{-k} \mid k, x, w_k\right) \delta_{x, w_k}\left(x', \mathrm{d}u_k\right). \tag{2.5}$$

In this way $N-1$ of the N weights are rejuvenated producing a new, typically quite different, estimate for the target distribution at the previous iteration, which in turn reduces the chance of the chain becoming stuck. Pseudo code for the pseudo-marginal algorithm with a rejuvenation step is given in Algorithm (2.4). An appealing feature of the approach is its flexibility - the rejuvenation need not be carried out at every step. One could consider for instance an adaptive rule by which the weights are rejuvenated according to some degeneracy criteria, though care must be taken here to avoid disrupting detailed balance. Our approach, in contrast, will be to study PsMMH algorithms in which independently at each iteration the rejuvenation step is performed with a probability $\lambda \in [0,1]$, which we will term 'λ-PsMMH'. Thus 0-PsMMH refers to the standard PsMMH of Section 2.1.3, whereas 1-PsMMH refers to the PsMMH algorithm with rejuvenation at every step.

2.2.3 Example: rejuvenation

We now return to the setting of Example 2.1.4 to test this approach. Figure 2.4 presents the output diagnostics from two independent runs of the PsMMH algorithm, one with an additional rejuvenation step and the other without. The parameters for this example were set to impose a greater challenge on the algorithm; $M = 5 \times 10^5$ iterations were used again, but now with $N = 3$ weights at every step, a weight variance of $\gamma^2 = 4$ and a proposal standard deviation of $\sigma = 8$. It is clear from this experiment that rejuvenation can offer impressive benefits over standard PsMMH. Figure 2.4 shows for example a roughly 40% reduction with respect to the autocorrelations of the chain produced.

We now show that the target distribution $\tilde{\pi}^N$ is invariant with respect to such rejuvenating transitions. Note that, where no confusion is possible, we will hereafter often make use of the reduced notation

$$R_{x,N}\left(w_{1:N}, \mathrm{d}u_{1:N}\right) := \sum_{k=1}^{N} \mathring{\pi}\left(k \mid x, w_{1:N}\right) \mathring{\pi}\left(w_{-k} \mid k, x, w_k\right) \delta_{w_k}\left(\mathrm{d}u_k\right).$$

Lemma 1. *The rejuvenation kernel R_N leaves $\tilde{\pi}^N$ invariant.*

Proof. It is sufficient to show that $R_{x,N}\left(w_{1:N}; \mathrm{d}u_{1:N}\right) := \sum_{k=1}^{N} \frac{w_k}{\sum_{j=1}^{N} w_j} Q_x^N\left(u_{-k} \mid w_k\right) \delta_{w_k}\left(\mathrm{d}u_k\right)$ is

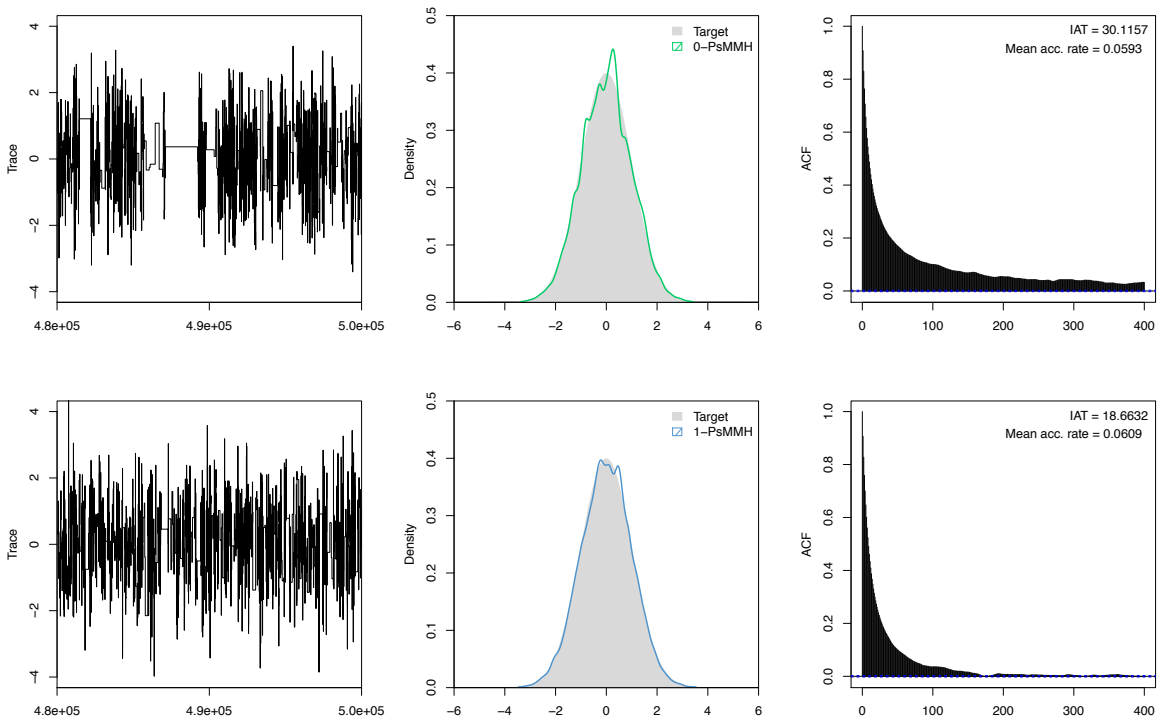


Figure 2.4: *Output diagnostics from two independent runs: (top) a standard PsMMH algorithm; (bottom) a PsMMH algorithm with rejuvenation at every step. From left to right they are a trace plot of the last* 2×10^4 *iterations of the Markov chain, an estimated density of the chain, and a plot of its estimated autocorrelations to a lag of 100 steps.*

reversible with respect to $\pi_x^N(\mathrm{d}w_{1:N})$ for any $x \in \mathrm{X}$. We observe the following equivalence

$$\begin{aligned}
\pi_x^N(\mathrm{d}w_{1:N}) R_{x,N}(w_{1:N}; \mathrm{d}u_{1:N}) &= Q_x^N(\mathrm{d}w_{1:N}) \frac{1}{N} \sum_{k=1}^{N} w_k Q_x^N(u_{-k} \mid w_k)\, \delta_{w_k}(\mathrm{d}u_k) \\
&= \frac{1}{N} \sum_{k=1}^{N} w_k Q_x^N(\mathrm{d}w_{1:N}) \frac{Q_x^N(u_{-k}, w_k)}{Q_x^N(w_k)} \delta_{w_k}(\mathrm{d}u_k) \\
&= \frac{1}{N} \sum_{k=1}^{N} u_k Q_x^N(w_{1:N}) \frac{Q_x^N(\mathrm{d}u_{1:N})}{Q_x^N(u_k)} \delta_{u_k}(\mathrm{d}w_k) \\
&= Q_x^N(\mathrm{d}u_{1:N}) \frac{1}{N} \sum_{k=1}^{N} u_k Q_x^N(w_{-k} \mid u_k)\, \delta_{u_k}(\mathrm{d}w_k) \\
&= \pi_x^N(\mathrm{d}u_{1:N}) R_{x,N}(u_{1:N}; \mathrm{d}w_{1:N})
\end{aligned}$$

where we have used the exchangeability of the weights, and in the third step interchanged w_k and u_k which is made possible by the presence of the Dirac measure.

□

Interestingly the arguments above also suggest a rejuvenation strategy for the case $N = 1$. It consists of picking some number $n \geq 2$ and, assuming a current state of the algorithm (x, w), sampling weights $w_{2:n} \sim Q^n_{x,}(\cdot \mid w)$ to form a *rejuvenation pool* $w_{1:n} := (w, w_{2:n})$. The weight succeeding w is then chosen to be w_k with probability $\mathring{\pi}(k \mid x, w_{1:n})$. In this way one preserves the target density $\tilde{\pi}(\mathrm{d}x, \mathrm{d}w) := \pi(\mathrm{d}x) Q_x(w)\, w$, a result formalised in Lemma 2. To ease the proof we introduce the terminology 'scale-up' and 'scale-down' kernel to describe the following two probability densities:

$$\hat{P}_n\left(x, w; x', \mathrm{d}v_{1:n}\right) := \frac{1}{n} \sum_{i=1}^{n} Q^n_x\left(\mathrm{d}v_{-i} \mid v_i\right) \delta_{x,w}\left(x', v_i\right)$$

$$\check{P}_n\left(x, v_{1:n}; x', u\right) := \sum_{i=1}^{n} \frac{v_i}{\sum_{j=1}^{n} v_j} \delta_{x,u}\left(x', v_i\right)$$

which correspond to the probability of scaling-up from 1 weight to a pool of n weights, and scaling-down from a pool of n weights to 1 weight respectively. In possession of this notation the rejuvenation strategy for $N = 1$, via a rejuvenation pool of size n, is described by the transition

$$R_n(x, w; x, u) := \int_{\mathbb{R}^n_+} \hat{P}_{x,n}(w, dv_{1:n})\, \check{P}_{x,n}(v_{1:n}, u),$$

where we have made use of a now overloaded notation $R(\cdot, \cdot)$. This strategy is now shown to preserve detailed balance.

Lemma 2. *The rejuvenation kernel R_n leaves $\widetilde{\pi}$ invariant.*

Proof. Reversibility of $\pi_x(\mathrm{d}w) R_{x.n}(w; \mathrm{d}u)$ is established as in the proof of Lemma 1; the only additional task on this occasion is to marginalise out the ancillary weights $v_{1:n}$ comprising the rejuvenation pool. Firstly one finds

$$\begin{aligned}
\pi_x(\mathrm{d}w) R_{x.n}(w; \mathrm{d}u) &= Q_x(\mathrm{d}w)\, w \int_{\mathbb{R}^n_+} \hat{P}_{x,n}(w, \mathrm{d}v_{1:n})\, \check{P}_{x,n}(v_{1:n}, \mathrm{d}u) \\
&= \frac{w}{n} \int_{\mathbb{R}^n_+} \sum_{i,j=1}^{n} Q^n_x\left(\mathrm{d}v_{-i} \mid v_i\right) Q_x(\mathrm{d}w)\, \frac{v_j \delta_w(dv_i)\, \delta_{v_j}(\mathrm{d}u)}{\sum_{l=1}^{n} v_l}
\end{aligned}$$

at which point it is prudent to split the summands into diagonal and off-diagonal components,

giving for the right hand side:

$$\int_{\mathbb{R}_+^n} \frac{w}{n} \sum_{i=1}^{n} Q_x^n\left(\mathrm{d}v_{-i} \mid v_i\right) Q_x\left(\mathrm{d}w\right) \frac{v_i \delta_w\left(\mathrm{d}v_i\right) \delta_{v_i}\left(\mathrm{d}u\right)}{\sum_{l=1}^n v_l} + \int_{\mathbb{R}_+^n} \frac{w}{n} \sum_{\substack{i,j=1 \\ i\neq j}}^{n} Q_x^n\left(\mathrm{d}v_{-i} \mid v_i\right) Q_x\left(\mathrm{d}w\right) \frac{v_j \delta_w\left(\mathrm{d}v_i\right) \delta_{v_j}\left(\mathrm{d}u\right)}{\sum_{l=1}^n v_l}.$$

Carrying out the possible integrations yields

$$\frac{1}{n}\sum_{i=1}^{n} \int_{\mathbb{R}_+^{n-1}} \frac{Q_x^n\left(\mathrm{d}v_{-i}, \mathrm{d}w\right) uw \delta_w\left(\mathrm{d}u\right)}{\left[w + \sum_{\substack{l=1 \\ l\neq i}}^n v_l\right]} + \frac{1}{n}\sum_{\substack{i,j=1 \\ i\neq j}}^{n} \int_{\mathbb{R}_+^{n-2}} \frac{Q_x^n\left(\mathrm{d}v_{-i,j}, \mathrm{d}u, \mathrm{d}w\right) uw}{\left[u + w + \sum_{\substack{l=1 \\ l\neq i,j}}^n v_l\right]}$$

which, provided the weights satisfy the exchangeability assumption, is symmetric in u, w.

□

We have shown that for N weights it is straightforward to rejuvenate $N-1$ weights. Additionally, we have shown that for the case $N=1$ it is possible to rejuvenate the single weight through sampling a rejuvenation pool of size n. It is natural to ask if such rejuvenation schemes can be extended to include, for any number of weights N, strategies for fixing $k \in \{1,\ldots,N-1\}$ weights and rejuvenating $N-k$ weights. Such flexibility is desirable since it allows for greater control over the computational cost of executing the algorithm. For example if $N=10$ weights are used at each step, then rejuvenating 9 weights at every step doubles the cost of the algorithm. However, rejuvenating 3 weights at each step may prove to be beneficial as it only increases the computational cost of the algorithm by $1/3$. It is shown presently that whilst of theoretical interest, such strategies present unfortunate practical obstacles.

To facilitate our analysis we define new generalised scale-up and scale-down kernels. We consider as before the pseudo-marginal algorithm with N particles at every step, $N-k$ of which are to be rejuvenated (with probability λ). Now, the first task is to 'scale-down', that is, pick which weights are to be fixed, and calculate the respective probabilities with which this occurs. Just as in the construction of the earlier scale-up and scale-down kernels, we are inspired here by the form of the probability distribution $\mathring{\pi}$ introduced implicitly in Equation (2.4). Let $\alpha_{1:k} = \{\alpha_1,\ldots,\alpha_k\}$ be some permutation of $k < N$ elements of the set $\{1,\ldots,N\}$. Then for any $x \in \mathsf{X}$ and weights $w_{1:N} \in \mathbb{R}_+^N$, the probability of keeping fixed the weights of indices $\alpha_{1:k}$ is

$$\mathring{\pi}(\alpha_{1:k}|x, w_{1:N}) = \frac{\sum_{i=1}^k w_{\alpha_i}}{\sum_{j=1}^N w_j} \mathbb{I}\{\alpha_{1:k} \subset \{1,\ldots,N\}\}$$

and

$$\mathring{\pi}(w_{-\alpha_{1:k}}|\alpha_{1:k},x,w_{\alpha_{1:k}}) = Q_x^N(w_{-\alpha_{1:k}}|w_{\alpha_{1:k}}).$$

Provided for $v_{1:k} \in \mathbb{R}_+^k$ there exists an injection $\sigma : \{1,\ldots,k\} \to \{1,\ldots,N\}$ such that $v_i = w_{\sigma(i)}$, $i \in \{1,\ldots,k\}$, which we also write $v_{1:k} = w_\sigma$, one finds

$$\check{P}_{N,k}\left(x,w_{1:N};x',v_{1:k}\right) := \frac{1}{{}^kP_N}\sum_{\sigma\in\mathfrak{S}}\frac{\sum_{i=1}^k w_{\sigma(i)}}{\sum_{j=1}^N w_j}\delta_{x,w_\sigma}(x',v_{1:k})$$

where $\mathfrak{S}$ is the set of injections from $\{1,\ldots,k\}$ to $\{1,\ldots,N\}$; it is a standard result that there are ${}^kP_N = \frac{N!}{(N-k)!}$ such injections. The probability associated with a particular rejuvenation of the remaining $N-k$ is then given by the scale-up operator

$$\hat{P}_{k,N}\left(x,v_{1:k};x',u_{1:N}\right) := \frac{1}{{}^kP_N}\sum_{\sigma\in\mathfrak{S}} Q_x^N(u_{-\sigma}|u_\sigma)\delta_{x,v_{1:k}}(x',u_\sigma).$$

These results are interesting yet impracticable as presented since to implement the rejuvenation step would require calculation of the summation of ${}^kP_N$ different subsets of the N weights. Future work could consider somehow restricting the set of injections considered to reduce the computational complexity, for instance by ensuring k is either very close to 1 or very close to N.

We now move towards proving our main result, that the asymptotic variance of the rejuvenated PsMMH is no greater than that of the standard algorithm. The following definition (e.g. [30, 31]) will be useful.

Definition 3. *Suppose P_0 and P_1 are Markov transition kernels on a generic metric space $(\mathsf{X},\mathcal{X})$ with invariant distribution π. The kernel P_1 is said to* dominate P_0 in the covariance ordering, *written $P_1 \succcurlyeq P_0$, if for all $f \in L^2(\pi)$, $\langle f,P_1 f\rangle \leq \langle f,P_0 f\rangle$.*

This ordering is an extension of an earlier concept, central to [10]: we say P_1 *dominates P_0 off the diagonal*, written $P_1 \succeq P_0$, if for π-almost all $x \in \mathsf{X}$, $P_1(x,\mathsf{A}\setminus\{x\}) \geq P_0(x,\mathsf{A}\setminus\{x\})$ for all $\mathsf{A} \in \mathcal{X}$. It may be simpler in some cases to demonstrate off-diagonal domination than domination in the covariance ordering, and it is sufficient, as shown in [10, Lemma 3], since $P_1 \succeq P_0$ implies $P_1 \succcurlyeq P_0$. A powerful result from [31] is now quoted, and will be pivotal to proving our theorem.

Lemma 3. *Let $\tilde{\pi}$ be a probability measure on a metric space $(\mathsf{Y}\times\mathsf{U},\mathcal{Y}\otimes\mathcal{U})$ and let P_i and Q_i, $i \in \{0,1\}$, be kernels on the same space satisfying*

i) P_i and Q_i, $i \in \{0,1\}$, are $\tilde{\pi}$-reversible,

ii) $P_1 \succcurlyeq P_0$ and $Q_1 \succcurlyeq Q_0$.

Assume also that for all $(y,u) \in (\mathsf{Y} \times \mathsf{U})$,

$$P_i(y,u;\{y\} \times \mathsf{U}) = 1 \qquad (i \in \{0,1\}). \tag{2.6}$$

Then for all $f \in L^2(\tilde{\pi})$ *such that* $f(y,u) = h(y)$ *for some h, and such that*

$$\sum_{k=1}^{\infty} \left| \left\langle f, (P_i Q_i)^k f \right\rangle \right| < \infty \qquad (i \in \{0,1\})$$

it holds that

$$v(f, P_1 Q_1) = v(f, Q_1 P_1) \leq v(f, P_0 Q_0) = v(f, Q_0 P_0).$$

The result is a standard one when the compositions $P_0 Q_0$ and $P_1 Q_1$ are themselves $\tilde{\pi}$-reversible, the classical statement of which is found in [10], but such composite reversibility does not in general hold for $\tilde{\pi}$-reversible kernels P_i, Q_i, $i \in \{0,1\}$. In possession of this result we now show that, with respect to asymptotic variance, the λ-PsMMH algorithm can perform no worse than its 0-PsMMH counterpart.

Theorem 1. *Let* $\left\{ \left(X_k^{\lambda}, W_k^{\lambda} \right); k \in \mathbb{N} \right\}$ *and* $\{(X_k, W_k); k \in \mathbb{N}\}$, *with transition kernels* $\tilde{P}_N^{\lambda}$ *and* $\tilde{P}_N$, *be Markov chains generated by the N particle* λ*-PsMMH and 0-PsMMH algorithms respectively, where* $\lambda \in [0,1]$ *is a constant. Suppose that* $\left(X_0^{\lambda}, W_0^{\lambda} \right) \sim \tilde{\pi}$ *and* $(X_0, W_0) \sim \tilde{\pi}$. *Then for any function* $g \in \mathsf{L}^2(\pi)$ *satisfying*

$$\sum_{k=1}^{\infty} |\operatorname{Cov}(g(Z_0), g(Z_k))| < \infty \qquad (\textit{for } Z = X^{\lambda} \textit{ and } Z = X)$$

we have

$$v\left(g, \tilde{P}_N^{\lambda}\right) \leq v\left(g, \tilde{P}_N\right).$$

Proof. To benefit from Lemma 3 we write the transition kernel associated with each of the algorithms as the composition of two $\tilde{\pi}$-reversible kernels (we mean $\tilde{\pi}^N$, and merely suppress the N). First notice that the 0-PsMMH transition kernel may be written with a redundant 'identity' transition kernel as

$$\tilde{P}_N = \delta \tilde{P}_N$$

by which it is meant that $\tilde{P}_N(x,w;\mathrm{d}y,\mathrm{d}u) = \iint \delta_{(x,w)}(\mathrm{d}x',\mathrm{d}w') \tilde{P}_N(x',w';\mathrm{d}y,\mathrm{d}u)$. Secondly the λ-PsMMH transition kernel may be written as the composition of the rejuvenation kernel and the 0-PsMMH transition kernel, that is

$$\tilde{P}_N^{\lambda} = R_N \tilde{P}_N$$

where R_N is defined by Equation (2.5). That the conditions of the lemma are satisfied is straightforward to show. We establish first the reversibility criterion. The delta measure is trivially $\tilde{\pi}$-reversible since it is reversible with respect to any probability measure on $(\mathsf{X}, \mathcal{B}(\mathsf{X}))$. In Lemma 1 it is shown that R_N is $\tilde{\pi}$-reversible and $\tilde{P}_N$ is reversible with respect to $\tilde{\pi}$ by construction as it describes a Metropolis Hastings algorithm targeting $\tilde{\pi}$.

We now demonstrate the relevant covariance orderings using the sufficient criteria of domination off the diagonal. It holds by definition of the delta measure that for any $(x,w) \in \mathsf{X} \times \mathbb{R}_+^N$ and $\mathsf{A} \subseteq (\mathcal{B}(\mathsf{X}) \times \mathcal{B}(\mathbb{R}_+^N))$ we have $R_N(x,w;\mathsf{A} \setminus \{x,w\}) \geq \delta_{(x,w)}(\mathsf{A} \setminus \{x,w\}) = 0$. Of course we can add to this the trivial identity $\tilde{P}_N \succcurlyeq \tilde{P}_N$. In addition the restriction given in (2.6) is satisfied for the measures δ and R_N as required, since these kernels only affect the weights. Finally applying Lemma 3 to the function $f(x,w) = g(x)$ gives the result.

□

2.2.4 Random refreshment

We describe now an algorithm, introduced in [31], that can be thought of as an MCWM algorithm in which the proposal of the recalculated weights is 'Metropolised' through an additional accept/reject step. In this way the so-called *Random Refreshment* algorithm (in contrast to the systematic refreshment of MCWM) strikes a new balance, on the one hand reclaiming exactness, on the other sacrificing some of the desirable autocorrelation properties of the standard MCWM algorithm. A single step of the algorithm, whose transition kernel we term $\bar{P}_N$, is performed according to the instructions given in Algorithm 2.5.

With the random refreshment algorithm defined a picture begins to emerge of a complex web of possible pseudo-marginal algorithms. One can imagine for instance the standard 0-PsMMH algorithm and the MCWM algorithm as being polar extremes. By turning the λ-dial from 0 to 1 we obtain the 1-PsMMH algorithm that rejuvenates $N-1$ of N weights systematically, which is in some sense as close as it is possible to get to the MCWM algorithm without sacrificing detailed balance. The refreshment algorithm (we drop the 'random' for brevity) lies somewhere in the middle, in that it will only refresh the weights in perhaps half of the iterations, but exists, in a manner of speaking, on a slightly different plane in our mental image; crucially its refreshments are total, they are not rejuvenations, and they are *targeted* by virtue of the extra Metropolis step. One unfortunate facet of the refreshment algorithm is its relative lack of flexibility. For an equal number of iterations M the refreshment algorithm is virtually identical in cost to the MCWM algorithm, their cost being of the order $2NM$, since even for those iterations where the previous likelihood estimate is carried through and not recalculated, the N new weights must nonetheless be drawn at every step to calculate the probability of their own acceptance.

Algorithm 2.5 Random Refreshment algorithm: simulating from $\bar{P}_N(x,w;\cdot)$

Input: current state (x,w).

1. Sample $w' \sim Q_x^N(\cdot)$.
2. With probability $\rho(w,w') := 1 \wedge \frac{N^{-1}\sum_{i=1}^N w_i'}{N^{-1}\sum_{i=1}^N w_i}$ set $U \leftarrow w'$, otherwise set $U \leftarrow w$.
3. Sample $Y \sim q(x,\cdot)$.
4. Sample $\hat{U} \sim Q_Y^N(\cdot)$.
5. With probability $\tilde{\alpha}(x,U;Y,\hat{U})$ given in (2.2):

   set $(x',w') \leftarrow \left(Y,\hat{U}\right)$,

   otherwise:

   set $(x',w') \leftarrow (x,U)$.

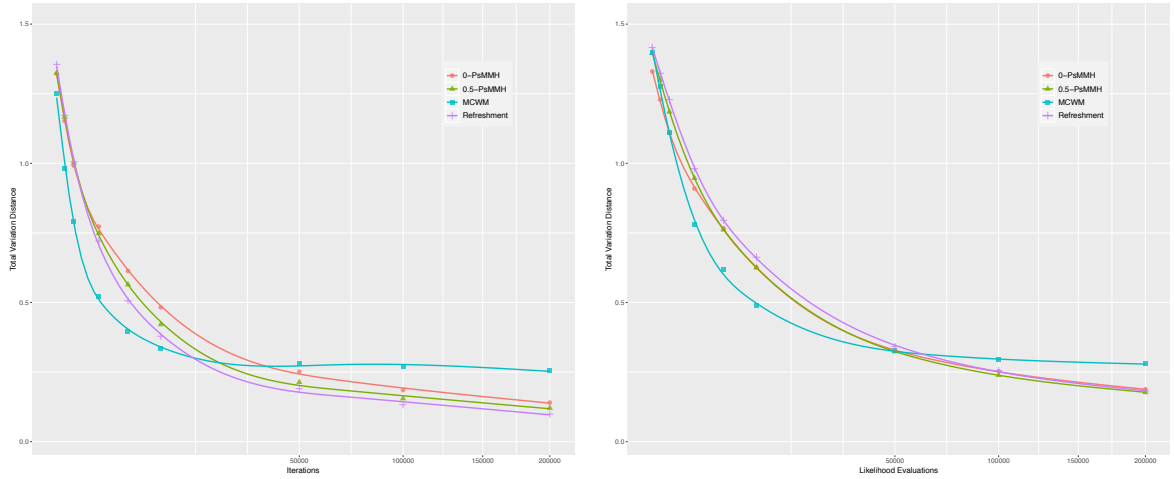
Output: new state (x',w').

2.3 Simulation studies

In this section the algorithms and techniques discussed throughout this chapter are tested with a view to establishing if they are in some settings worth implementing over standard algorithms.

The central aim of the tests that follow will be to ask whether or not it is possible to 'beat the cost' of augmenting standard algorithms with computationally intensive additional steps. In other words, we ask: is it possible through improvements in statistical efficiency, speed of convergence, or both to make up for the additional costs and thus make a gain *overall*? Certainly there will be examples for which an improvement can be observed by implementing, say, rejuvenation, but where it will be clear that a more efficient use of resources would be instead to simply run the original chain for longer. Other examples will be amenable to our additions and optimal strategies will be sought in these cases. Examples in this section will, as in earlier examples, target the standard normal distribution, as well as a more challenging asymmetric beta distribution.

For the normal distribution examples, we use a higher variance in the proposal density than would typically be used. The lower acceptance rate and increased autocorrelations of the algorithm caused by this choice are deleterious to its performance. We make this choice in an



(a) *Distance as a function of the number of iterations of the Markov chain.*

(b) *Distance as a function of the number of likelihood evaluations.*

Figure 2.5: *Estimated total variation distance of various pseudo-marginal algorithms with* $N = 2$*,* $\gamma^2 = 3.5$*,* $\sigma = 8$*. The splines are used solely for visual clarity.*

attempt to recreate in a simpler environment some of the conditions found in very challenging settings for statistical inference algorithms. It is not uncommon in a high-dimensional pseudo-marginal algorithm to observe an acceptance rate under 5%, for example. It may also be very difficult to choose an effective proposal density for the high-dimensional setting, or to tune the proposal appropriately. The beta distribution target, in contrast, is itself already more challenging for the algorithms presented here - it is asymmetric, U-shaped, and is defined on a compact support. It is therefore unnecessary to undermine the performance of the algorithm in any other way, and so we make a more natural proposal variance choice in these examples.

2.3.1 Rate of convergence comparison

We first concentrate on comparing the pseudo-marginal algorithms presented in this chapter by their rate of convergence for two test cases, a standard normal distribution and a Beta$(0.5, 0.7)$ distribution.

Figure 2.5a describes an experiment in which the total variation distance, estimated using functions in the 'distrEx' R package [32], for each algorithm is compared for a standard normal target. Some aspects of the graph agree with our intuitions; the MCWM algorithm converges fastest of all but with a prominent bias, and random refreshment and 1-PsMMH are both superior in rate of convergence to 0-PsMMH over any time scale.

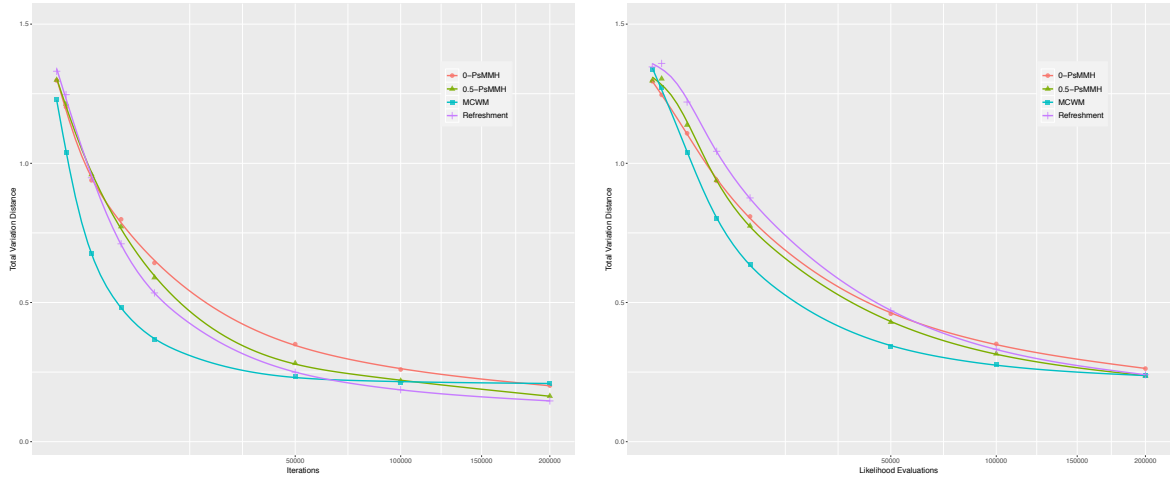
Figure 2.5b shows again the performance of each algorithm with respect to total variation distance from the target, but now with reference to a measure of the computation time. Here

we have chosen to use the number of likelihood evaluations as a surrogate for wall-clock time. It is a natural choice as in the majority of applications the likelihood calculation is the dominant computational cost, though this may not always be the case in practice. For our purposes, we suppose that in computational complexity, the PsMMH algorithm is $\mathcal{O}(MN)$, where M is the number of iterations of the Markov chain and N is the number of weights. For this set of experiments, the 0-PsMMH algorithm executes the fewest likelihood evaluations, N per iteration of the Markov chain. The 0.5-PsMMH algorithm uses on average $N + (N-1)/2$ likelihood evaluations at every iteration; $(N-1)/2$ for rejuvenating the previous weight with probability 0.5 and a further N for the next estimate. Both MCWM and the refreshment algorithm make use of $2N$ likelihood evaluations per iteration of the Markov chain. It is therefore possible to scale the number of MCMC iterations executed for each algorithm, so that each has made (in expectation) the same number of likelihood evaluations. This experimental procedure allows a user with knowledge of the computational cost of their likelihood evaluation, to 'look up' which algorithm offers the best performance in total variation distance, given the computational time available.

In 2.5b we again observe that MCWM offers competitive performance for a small number of likelihood evaluations but at higher numbers becomes unfavourable due to its bias. The remaining methods offer similar performance across a wide range of likelihood evaluations, with 0.5-PsMMH by a small margin enjoying the lowest total variation distance for more than 50,000 likelihood evaluations. Around 200,000 likelihood evaluations the performances of the algorithms become too close to distinguish.

An analogous experiment is now given in Figure 2.6 for the case of a Beta$(0.5, 0.7)$ target distribution. In Figure 2.6a, MCWM again exhibits a bias, evident through its performance plateau, though it fares better in comparison to its performance on the simpler example. The remaining three algorithms exhibit the same ordering as in the simpler normal distribution example. More interestingly, Figure 2.6b shows that with a more challenging target distribution, discounting for computational cost, MCWM can be an optimal choice, at least up to 2×10^5 likelihood evaluations (equivalently 1×10^5 MCMC iterations). The remaining methods again offer similar performance to each other, with the rejuvenated PsMMH algorithm marginally outperforming the refreshment algorithm and 0-PsMMH. When run for a greater number of likelihood evaluations, in line with expectations, the performance of MCWM (with respect to its total variation distance from the target) becomes less favourable due to its bias.

We have shown in both a simple and more challenging example that the use of some level of rejuvenation in PsMMH can improve the rate of convergence to the target, in some settings and when the chain is run for enough iterations. In light of this, rejuvenation strategies show promise as a beneficial addition to PsMMH. We therefore now turn our attention to whether



(a) *Distance as a function of the number of iterations of the Markov chain.*

(b) *Distance as a function of the number of likelihood evaluations.*

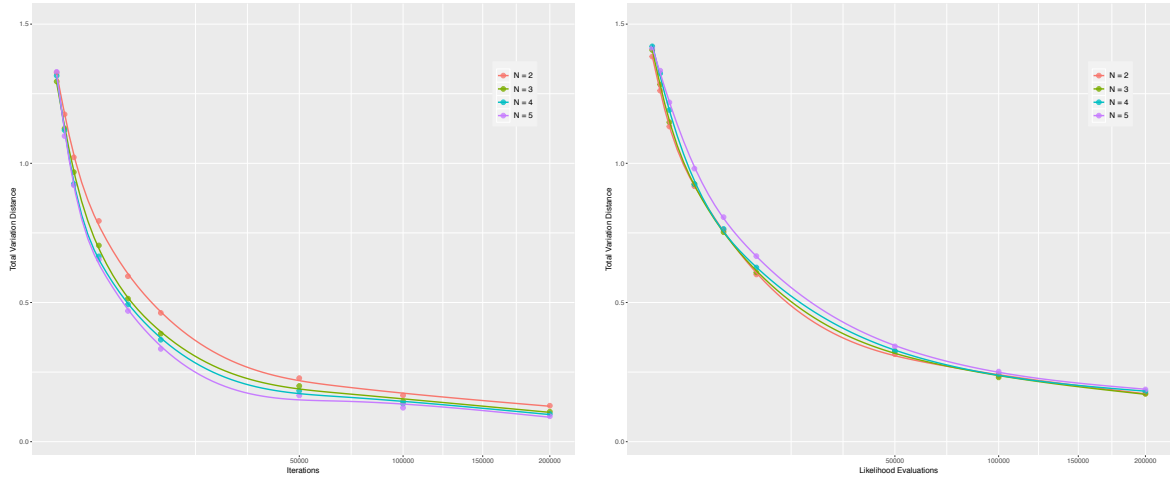
Figure 2.6: *Estimated total variation distance from* Beta$(0.5, 0.7)$ *target of various pseudo-marginal algorithms with* $N = 2$, $\gamma^2 = 3.5$, $\sigma = 3$. *The splines are used solely for visual clarity.*

there is an optimal choice of the number of weights N and the probability of rejuvenation λ in λ-PsMMH algorithms.

2.3.2 Optimising N

Having established the asymptotic variance results of Section 2.2.1, we concern ourselves here primarily with the difference in rate of convergence of the PsMMH algorithm with varying numbers of weights N. The examples in this section compare algorithms that have been run with the same proposal distribution and a constant variance parameter. This is almost certainly not a completely fair test - it is reasonable for example to suppose that better results could be achieved when N is larger by running the chain a little 'hotter', that is, using a proposal of higher variance. Nevertheless, it is hoped that this will have a modest effect, if any. Our belief here is supported to an extent by the argument presented in [21, p.10] that the marginal random walk Metropolis algorithm enjoys a relatively flat efficiency curve, as a function of the proposal standard deviation σ, and consequently is (in the high dimensional case) at least 80% efficient with any acceptance rate in the region $[0.15, 0.5]$, despite the optimal being 0.234.

Figure 2.7a shows the total variation distance between the empirical distribution of a 1-PsMMH algorithm and a standard normal target distribution for varying numbers of steps, with each line describing a different N. Predictably, larger N seems to guarantee monotonically faster convergence. In addition this improvement probably diminishes - the improvement in speed of convergence derived from increasing the number of weights from 2 to 3 is likely



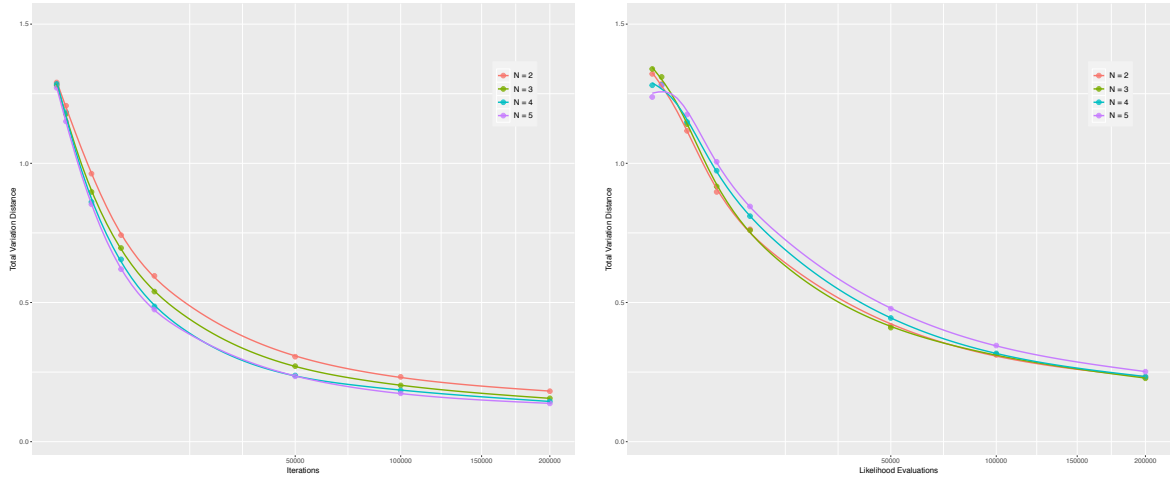
(a) *Distance as a function of the number of iterations of the Markov chain.*

(b) *Distance as a function of the number of likelihood evaluations.*

Figure 2.7: *Estimated total variation distance of the* 1*-PsMMH algorithm from a standard normal target. Using parameters* $\gamma^2 = 3.5$, $\sigma = 8$.

to be greater than the improvement derived from increasing the number of weights from 4 to 5. Since the computational cost of running the algorithm increases essentially linearly in N, it is likely that the choice of N that optimises the rate of convergence will be small in most circumstances. This claim is supported by Figure 2.7b which shows the total variation distance as a function of the number of likelihood evaluations. While the performance across the values of N shown is similar, it is clear that $N = 2$ consistently offers the lowest total variation distance per likelihood evaluation, in the range tested, with $N = 3$ a very close second and $N = 5$ the worst performer in total variation distance for any number of steps discounted by computational cost.

We return to the more challenging Beta$(0.5, 0.7)$ target in Figure 2.8. Figure 2.8a closely resembles the outcome of the standard normal target experiment. In Figure 2.8b the same experiment is expressed as a function of the number of likelihood evaluations, and this too looks similar to the experiment in which a standard normal target was used. There is very close performance between the algorithms with $N = 2$ and $N = 3$, and they overall out-perform the choice $N = 4$ which overall out-performs the choice $N = 5$. We may be tempted to surmise that the choice of N is invariant to the complexity of the problem, however this seems a priori unlikely. More plausibly, the choice of N is likely to depend in a complex way on the dimension of the problem, the precise structure of the target distribution, and so on.



(a) *Distance as a function of the number of iterations of the Markov chain.*

(b) *Distance as a function of the number of likelihood evaluations.*

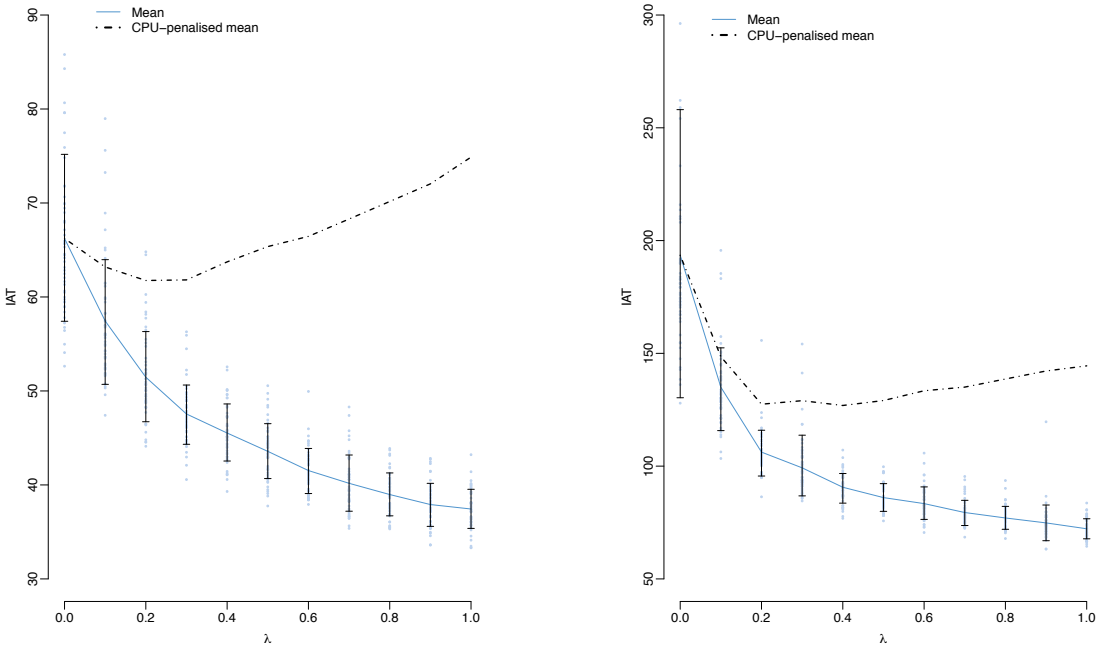
Figure 2.8: *Estimated total variation distance of the* 1*-PsMMH algorithm with* $\gamma^2 = 3.5$*,* $\sigma = 8$*.*

2.3.2.1 Optimising asymptotic variance

As discussed in Section 2.2 an appropriate measure of the asymptotic variance of the chain is given by the integrated autocorrelation time. One would like to minimise the IAT, since this minimises the asymptotic variance of ergodic averages based on the chain. However, in keeping with the principles of this set of simulations, it is also essential to penalise the performance of an algorithm as its cost increases to determine whether the additions perform better than simply running the original chain for longer.

Figure 2.9a shows the effect on the IAT of varying the rejuvenation probability λ in the λ-PsMMH Markov chain generated by $\tilde{P}^{\lambda}$ where one weight is drawn at every step and rejuvenation is carried out with a pool of two weights. The target distribution is a standard normal. The dash-dotted line on the plots represents the CPU-penalised IAT for $\lambda \in [0,1]$, which we observe achieves a non-trivial minimum somewhere in the region $[0.2, 0.3]$. This CPU-penalised IAT is found by scaling the IAT in proportion to its cost, that is in this case a factor of $1+\lambda$. Achieving a non-trivial minimum implies that an optimal strategy has been found with respect to IAT, since this strategy minimises the IAT penalised by computational cost. The practitioner could realistically carry out such a test, perhaps on a short run in advance of a more serious computation, to determine which value of λ may improve the performance of their algorithm.

By comparison Figure 2.9b targets a Beta$(0.5, 0.7)$. In this case the benefits of rejuvenation are far more pronounced. In contrast to Figure 2.9a, not only is an optimum of the computational-cost penalised IAT achievable here, but *any* choice $\lambda > 0$ is superior in cost-



(a) *Target distribution* $\mathcal{N}(0,1)$. *Using* $N = 1$, $n = 2$, $\gamma^2 = 3.5$, $\sigma = 10$.

(b) *Target distribution* $\text{Beta}(0.5, 0.7)$. *Using* $N = 1$, $n = 2$, $\gamma^2 = 3$, $\sigma = 3$.

Figure 2.9: *Plots exploring the effect of increasing the rejuvenation probability* λ *on the IAT of the corresponding* λ*-PsMMH chain. Each blue point, of which there are 100 for each* λ*, is an IAT estimate produced from running an entire PsMMH algorithm with* $M = 2 \times 10^6$ *steps. To be explicit, taking Figure (a), the blue points lying vertically above, e.g.,* $\lambda = 0.3$*, are 100 independent realisations of the Sokal estimator* $\hat{\tau}$ *of the true IAT* τ *of the* λ*-PsMMH algorithm in question.*

penalised IAT to the choice $\lambda = 0$. In other words, for our realistically complex target, any amount of rejuvenation is better than no rejuvenation. The dotted curve does however exhibit an optimum, and again it is found around $\lambda = 0.2$.

In some settings then it is clear that a certain amount of rejuvenation offers a net benefit. It is not yet apparent in which settings these benefits are greatest, though experiments seem to indicate a more striking improvement in settings featuring complex target distributions, as well as inappropriate or incorrectly calibrated proposal densities. If this is the case then it may be that rejuvenation has potential to offer benefits in some challenging settings.

It seems unlikely that rejuvenation will be necessary if the standard pseudo-marginal algorithm is performing well. Where the pseudo-marginal algorithm has mean acceptance rate greater than for example 0.2, rejuvenation is unlikely to offer much improvement. However, where the standard pseudo-marginal algorithm is struggling, for instance achieving a mean acceptance rate of less than 0.1, it seems more likely that rejuvenation could offer some improvement. One example of particular interest is PMCMC algorithms, wherein rejuvenation

can be carried out through a conditional SMC step. PMCMC methods are commonly used to conduct inference in highly complex settings in which it may be impossible to come close to an optimal proposal distribution. Indeed, if one is constructing a high-dimensional proposal distribution through many auxiliary proposal distributions, as is often the case in SMC, this is in some sense akin to a very high variance proposal density in the simple example discussed in this project - and such high variance proposal algorithms respond very well, in the simulations run so far, to rejuvenation.

Changing tack, there may also be alternative strategies to be sought in the particular implementation of rejuvenation. If a rejuvenation probability of say $\lambda = 0.2$ is desired at each step then there are at least two ways of achieving this. We have so far described a process by which the rejuvenation is carried out independently at each step with probability 0.2. Another strategy is to rejuvenate the weight deterministically every five steps. Recent work [33] exploring the optimality of random or deterministic cycling of reversible kernels may prove to be useful. A very brief description of the main result [33, Theorem 6] is as follows. Let π be a probability distribution and Π_1, Π_2 be two π-reversible Markov transition kernels (defined on appropriate measurable spaces). Define $P^{\text{rand}} := (\Pi_1 + \Pi_2)/2$, the transition kernel of an MCMC algorithm evolving according to Π_1 and Π_2, and denote by P^{strat} the MCMC algorithm evolving *deterministically* according to the composition $\Pi_1\Pi_2$. Andrieu shows [33] under reasonable conditions on the function f that $v(f, P^{\text{rand}}) \geq v(f, P^{\text{strat}})$, whence it is natural to ask - does the result hold for more than two kernels? To the best of our knowledge this is unknown and may be a fruitful direction for future research.

2.4 Discussion

In this chapter we presented a survey of popular pseudo-marginal approaches, and explored novel 'rejuvenation' techniques for pseudo-marginal algorithms, formalising and extending the ideas presented in [29]. The reversibility of the transition kernels implied by our rejuvenation strategies has been demonstrated, and moreover we have proved that our λ-PsMMH strategy outperforms the standard PsMMH algorithm in asymptotic variance. The simulation results presented seem to suggest that both λ-PsMMH and random refreshment are strategies of some promise - in our tests the two algorithms offer similar convergence rates, which are a marked improvement over standard algorithms. Critically we have shown that there are realistic scenarios in which the improvement in asymptotic variance of λ-PsMMH appears to justify its computational expense, as is illustrated by the kind of optimality found in Figure 2.9. We now briefly discuss potential avenues for future research in this area.

Our first thought is towards computational techniques. Throughout the report it is assumed

that all strategies are run in series - it was natural to test the algorithms in this way since it would be ideal if they could be shown to be optimal in series. Where parallel architectures are available they can potentially be used to great effect, depending on the particular setting. Computational advances are increasingly opening up the possibilities of efficient parallelisation, so it may be worthwhile exploring the ways in which rejuvenation can be frugally adapted to take advantage of these capabilities. If parallel computing is available (D cores say), is it preferable to run D simultaneous PsMMH algorithms and then average, or one PsMMH algorithm with D weights?

Secondly, there are potentially further ways to capitalise on the flexibility of λ-PsMMH. It would be possible for instance to use it as a strategy primarily intended to increase the rate of convergence of the algorithm. In this case, one could for example only use rejuvenation for the first 2×10^4 steps, then stop. This algorithm would make good use of the potential of rejuvenation for fast convergence, without requiring it to be the most efficient way to run the entire chain. Another, similar, idea would be to create a decreasing sequence of λs, and use them so that the probability of rejuvenation is larger at the beginning of the Markov chain and far less likely towards the end. There are many further adaptive strategies possible, for example using MCWM for the burn in and then switching to λ-PsMMH.

Chapter 3

Spectral analysis for Markov chains

In the last chapter we explored several strategies for improving the performance of pseudo-marginal type algorithms. Primarily we concerned ourselves with the efficiency of the chain, and for this reason studied results relating to the asymptotic variance of Markov chains. We now turn our attention to analysing the convergence rate of the rejuvenation strategy of Chapter 2 in some settings.

3.1 A brief survey of spectral theory

Before considering convergence rates of Markov chains it will be essential to have an understanding of some aspects of spectral theory. We embark upon this project now, guided by [34]. The results here will be described in the discrete state space setting. In particular we will study the eigenvalue and eigenvector relationship for square matrices, though many of the concepts generalise straightforwardly to continuous state spaces.

First define the *total variation distance* between two probability distributions μ and ν on a measurable space $(\Omega, \mathcal{F})$:

$$\| \mu - \nu \|_{\mathrm{TV}} := \max_{A \in \mathcal{F}} |\mu(A) - \nu(A)| . \tag{3.1}$$

where $|\cdot|$ represents the usual Euclidean norm. In some sense this describes the greatest possible disagreement between the two distributions about an event A. Of course conducting a search over all possible subsets of the sample space is prohibitive, and so one often uses the following alternative characterisation:

$$\| \mu - \nu \|_{\mathrm{TV}} = \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)| \tag{3.2}$$

the proof of which is well known. We note that it is clear from equation 3.2 that the total variation satisfies a triangle inequality since, for any new probability distribution η, we have

$$\begin{aligned} \| \mu - \nu \|_{\mathrm{TV}} &= \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \eta(x) + \eta(x) - \nu(x)| \\ &\leq \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \eta(x)| + |\eta(x) - \nu(x)| \\ &= \| \mu - \eta \|_{\mathrm{TV}} + \| \eta - \nu \|_{\mathrm{TV}} \end{aligned} \tag{3.3}$$

where the second line makes use of the triangle inequality enjoyed by the Euclidean norm. Where it is still impossible to use the characterisations 3.1 and 3.2, several alternatives exist. One such alternative is offered by the method of coupling.

Definition 4. *A pair of random variables (X,Y) defined on the same probability space are said to be a coupling of the probability distributions μ and ν if they are marginally distributed according to μ and ν. That is, any pair (X,Y) satisfying $\mathbf{P}(X = x) = \mu(x)$ and $\mathbf{P}(Y = y) = \nu(y)$ is a coupling of μ and ν.*

As shown in the following result, coupling is a useful construction because it directly relates the complex distributional notion of total variation distance to a statement about equality of random variables. A proof of the following proposition can be found in [34].

Proposition 1. *Suppose μ and ν are two probability distributions on Ω, and $\mathscr{C}(\mu,\nu)$ is the set of all couplings (X,Y) of μ and ν. Then*

$$\| \mu - \nu \|_{\mathrm{TV}} = \inf \{ \mathbf{P}(X \neq Y) \colon (X,Y) \in \mathscr{C}(\mu,\nu) \} .$$

In practice one hopes to construct a coupling (X,Y) that comes close to this lower bound, since for any coupling $(X,Y) \in \mathscr{C}(\mu,\nu)$ we certainly have the upper bound $\| \mu - \nu \|_{\mathrm{TV}} \leq \mathbf{P}(X \neq Y)$.

Ultimately we seek to understand the total variation distance between the probability distribution of a Markov chain's state and the stationary distribution of that chain. Access to this distance would give a clear picture of the convergence of the Markov chain, the convergence rate being simply how quickly this distance reduces as a function of the number of steps the chain has taken. Define a *coupling of Markov chains* with transition matrix P to be a process $\{(X_t,Y_t)\}_{t=0}^{\infty}$ such that both (X_t) and (Y_t) are Markov chains with transition matrix P, though not necessarily the same initial distribution. Let $\mathbf{P}_{x,y}$ denote the probability law of the chain, conditional on $X_0 = x$ and $Y_0 = y$. One can modify any coupling $\{(X_t,Y_t)\}_{t=0}^{\infty}$ of Markov chains with transition P so that once they meet the two chains remain together thereafter. That is, if

for some s we have $X_s = Y_s$, then

$$X_t = Y_t \text{ for all } t \geq s. \tag{3.4}$$

One can imagine running the chains according to the original coupling until their first meeting, after which they are run in unison. Now suppose the transition matrix P defines an irreducible and aperiodic Markov chain, and that the stationary distribution is denoted π. We now give another result, which follows directly from the preceding proposition and definitions.

Theorem 2. *Suppose $\{(X_t, Y_t)\}$, with initial state (x, y), transition matrix P, and law $\mathbf{P}_{x,y}$, is a coupling satisfying 3.4. Let $\tau := \min\{s : X_s = Y_s\}$ be the first time the chains meet. Then for any $t \geq 1$,*

$$\left\| P^t(x, \cdot) - P^t(y, \cdot) \right\|_{\mathrm{TV}} \leq \mathbf{P}_{x,y}(\tau > t).$$

With this result in hand, we are in a position to quantify the convergence rate of the chain. Define $d(t) := \max_{x \in \Omega} \|P^t(x, \cdot) - \pi\|_{\mathrm{TV}}$, that is, the distance between the Markov chain and its stationary distribution after t steps, conditional on starting at a point $x \in \Omega$. The following corollary holds, describing this distance in terms of a coupling probability - its proof can be found in [34].

Corollary 2. *Suppose for each pair of states $x, y \in \Omega$ there exists a coupling $\{(X_t, Y_t)\}$ with $X_0 = x$ and $Y_0 = y$. Then*

$$d(t) \leq \max_{x,y \in \Omega} \mathbf{P}_{x,y}(\tau > t).$$

The distance $d(t)$ has connections with the eigenvalues of the transition matrix P, an idea we now explore further.

Definition 5. *Let P be a reversible (that is, reversible with respect to some probability measure) transition matrix on Ω with ordered eigenvalues $-1 \leq \lambda_{|\Omega|} \leq \cdots \leq \lambda_2 < \lambda_1 = 1$. P is said to have spectral gap, or 'right' spectral gap, $\gamma := 1 - \lambda_2$, which we will occasionally denote* $\mathrm{Gap}(P)$. *The 'absolute' spectral gap is given by $\gamma_\star := 1 - \lambda_\star$ where $\lambda_\star := \max_{2 \leq i \leq |\Omega|} \{|\lambda_i|\}$.*

Remark 2. That P is reversible guarantees it has an eigenvalue of 1, associated with the eigenvector π (its stationary distribution). In addition, where P is aperiodic and irreducible it cannot have an eigenvalue of -1, a consequence of the Perron-Frobenius theorem, thus $\gamma_\star > 0$.

Definition 6. *The ε-mixing time of a Markov chain P is defined by*

$$t_{\mathrm{mix}}(\varepsilon) := \min\{t : d(t) \leq \varepsilon\}. \tag{3.5}$$

In addition, the relaxation time of a reversible Markov chain with absolute spectral gap $\gamma_\star$ is defined to be

$$t_{\text{rel}} := \frac{1}{\gamma_\star}.$$

The following theorem (proved in [34]) explains the relationship between these two quantities.

Theorem 3. *Let P be the transition matrix of a reversible, irreducible Markov chain with state space Ω and stationary distribution π, and let $\pi_{\min} := \min_{x\in\Omega} \pi(x)$. Then*

$$t_{\text{mix}}(\varepsilon) \leq \log\left(\frac{1}{\varepsilon\pi_{\min}}\right) t_{\text{rel}}.$$

In short, the ε-mixing time (i.e. the speed of convergence) of the Markov chain is bounded above by a quantity inversely proportional to the absolute spectral gap. We now define the 'Dirichlet form' - an object that will be central to our study of the spectral gap.

Definition 7. *Suppose P is a reversible transition matrix with invariant distribution π. Then for functions f and h on Ω, the Dirichlet form associated to the pair (P,π) is defined by*

$$\mathcal{E}_P(f,h) := \langle (I-P)f, h\rangle_\pi.$$

We will drop the subscript P where there is no confusion. The following result introduces a variation of the definition above and connects the two.

Lemma 4. *If we define*

$$\mathcal{E}_P(f) := \frac{1}{2}\sum_{x,y\in\Omega} [f(x)-f(y)]^2 \pi(x)P(x,y), \tag{3.6}$$

then $\mathcal{E}_P(f) = \mathcal{E}_P(f,f)$.

Proof. First expand the squared bracket as follows

$$\begin{aligned}\frac{1}{2}\sum_{x,y\in\Omega} [f(x)-f(y)]^2 \pi(x)P(x,y) &= \frac{1}{2}\sum_{x\in\Omega} f^2(x)\pi(x) - \sum_{x,y\in\Omega} f(x)f(y)\pi(x)P(x,y) \\ &\quad + \frac{1}{2}\sum_{y\in\Omega} f^2(y)\pi(y) \\ &= \sum_{x\in\Omega} f^2(x)\pi(x) - \sum_{x,y\in\Omega} f(x)f(y)\pi(x)P(x,y).\end{aligned}$$

Then, using the reversibility of P, one finds

$$\begin{aligned}\sum_{x\in\Omega} f^2(x)\pi(x) - \sum_{x,y\in\Omega} f(x)f(y)\pi(x)P(x,y) &= \sum_{x\in\Omega} f^2(x)\pi(x) - \sum_{y\in\Omega} f(y)\pi(y)Pf(y)\\ &= \langle f,f\rangle_\pi - \langle Pf,f\rangle_\pi\\ &= \langle (I-P)f,f\rangle_\pi .\end{aligned}$$

□

We now move towards showing the connection between the spectral gap γ and the Dirichlet form associated with the transition matrix P. First we require the following lemma from [34].

Lemma 5. *Suppose the transition matrix P is reversible with respect to π. Then the inner product space $(\mathbb{R}^\Omega, \langle\cdot,\cdot\rangle_\pi)$ has an orthonormal basis of real-valued eigenfunctions $\{f_j\}_{j=1}^{|\Omega|}$ with corresponding real eigenvalues $\{\lambda_j\}$.*

Proof. The matrix $A(x,y) := \pi(x)^{1/2}\pi(y)^{-1/2}P(x,y)$ is symmetric by virtue of the reversibility of P with respect to π. As a symmetric $|\Omega|\times|\Omega|$ matrix there exists an orthonormal basis of eigenfunctions $\{\varphi_j\}_{j=1}^{|\Omega|}$ with corresponding real eigenvalues $\{\lambda_j\}_{j=1}^{|\Omega|}$. Notice first that $\sqrt{\pi}$ is an eigenfunction of A with eigenvalue 1, since

$$A\pi^{1/2}(x) = \sum_{y\in\Omega} A(x,y)\pi^{1/2}(y) = \sum_{y\in\Omega} \pi^{1/2}(x)P(x,y) = \pi^{1/2}(x).$$

Label $\varphi_1 := \sqrt{\pi}$ and $\lambda_1 := 1$. Define the diagonal matrix D_π by the diagonal elements $D_\pi(x,x) = \pi(x)$, whence we may write $A = D_\pi^{1/2}PD_\pi^{-1/2}$. If $f_j := D_\pi^{-1/2}\varphi_j$ then f_j is an eigenfunction of P with eigenvalue λ_j, since

$$Pf_j = PD_\pi^{-1/2}\varphi_j = D_\pi^{-1/2}A\varphi_j = D_\pi^{-1/2}\lambda_j\varphi_j = \lambda_j f_j.$$

Now, $\langle\varphi_i,\varphi_j\rangle = \delta_{ij}$ since $\{\varphi_j\}_{j=1}^{|\Omega|}$ are orthonormal with respect to the usual inner product. This implies $\{f_j\}_{j=1}^{|\Omega|}$ are orthonormal with respect to the inner product $\langle\cdot,\cdot\rangle_\pi$ because $\langle\varphi_i,\varphi_j\rangle = \left\langle D_\pi^{1/2}f_i, D_\pi^{1/2}f_j\right\rangle = \langle f_i,f_j\rangle_\pi$.

□

Finally we are ready to combine the two notions. Throughout the remainder of the chapter we will work with Dirichlet forms; the following lemma clarifies why this object is of interest.

Lemma 6. *Let P be the transition matrix of a reversible Markov chain, with associated Dirichlet form $\mathcal{E}(\cdot) = \mathcal{E}_P(\cdot)$ defined by (3.6). Then, its spectral gap $\gamma = 1 - \lambda_2$ satisfies*

$$\gamma = \min_{f: f \perp_\pi 1, \|f\|_2 = 1} \mathcal{E}(f) = \min_{f: f \perp_\pi 1, f \not\equiv 0} \frac{\mathcal{E}(f)}{\|f\|_2^2}.$$

Proof. Let $n = |\Omega|$. By Lemma 5 if $f_1, f_2, \ldots, f_n$ are the eigenfunctions of P with associated eigenvalues $-1 \leq \lambda_n \leq \cdots \leq \lambda_2 < \lambda_1 = 1$, then any function f can be written as $f = \sum_{j=1}^n \langle f, f_j \rangle_\pi f_j$. We can always take $f_1 = 1$. The $\mathsf{L}^2(\pi)$ norm of such a function is

$$\|f\|_2^2 = \langle f, f \rangle_\pi = \sum_{i=1}^n |\langle f, f_i \rangle_\pi|^2 .$$

Therefore if $\|f\|_2 = 1$ and $f \perp \mathbf{1}$ (since $f_1 = 1$ this is equivalent to the condition $\langle f, f_1 \rangle_\pi = 0$), then $f = \sum_{j=2}^n \alpha_j f_j$ where $\sum_{j=2}^n \alpha_j^2 = 1$. Finally, this means

$$\begin{aligned}
\langle (I - P) f, f \rangle_\pi &= \langle f, f \rangle_\pi - \langle Pf, f \rangle_\pi \\
&= \sum_{j=2}^n a_j^2 - \left\langle \sum_{i=2}^n a_i P f_i, \sum_{j=2}^n a_j f_j \right\rangle_\pi \\
&= \sum_{j=2}^n a_j^2 - \sum_{i,j=2}^n a_i a_j \lambda_i \langle f_i f_j \rangle_\pi \\
&= \sum_{j=2}^n \alpha_j^2 (1 - \lambda_j) \geq 1 - \lambda_2
\end{aligned}$$

where the third line follows from the orthonormality of the $\{f_j\}$ with respect to $\langle \cdot, \cdot \rangle_\pi$.

□

In summary, we have found a connection between the spectral gap of the transition matrix P and its Dirichlet form. In the next section we seek to understand what implications follow from establishing ordering of Dirichlet forms.

3.2 Connections to convergence rate

To begin we show that the Dirichlet form has connections with the notion of asymptotic variance from the previous chapter.

Proposition 2. *(Uniform (over all functions) ordering of Dirichlet forms implies ordering asymptotic variance.) Let P_1 and P_2 be reversible transition matrices with the same invariant*

distribution π. *If* $\mathcal{E}_{P_1}(f) \geq \mathcal{E}_{P_2}(f)$ *for all* $f \in \mathsf{L}^2(\pi)$, *then for any such* $f(\cdot)$

$$v(f, P_1) \leq v(f, P_2).$$

Proof. Suppose $\mathcal{E}_{P_1}(f) \geq \mathcal{E}_{P_2}(f)$ for all $f \in \mathsf{L}^2(\pi)$. Using the definition of the Dirichlet form and simple algebra, the following statements hold for all $f \in \mathsf{L}^2(\pi)$ and are equivalent:

$$\begin{aligned} \langle (I - P_1)f, f \rangle_\pi &\geq \langle (I - P_2)f, f \rangle_\pi \\ \pi(f^2) - \langle P_1 f, f \rangle_\pi &\geq \pi(f^2) - \langle P_2 f, f \rangle_\pi \\ \langle P_1 f, f \rangle_\pi &\leq \langle P_2 f, f \rangle_\pi . \end{aligned}$$

The final line implies $P_1 \succcurlyeq P_2$, in other words, P_1 dominates P_2 in the covariance ordering. It follows from [10, Theorem 4.] that $v(f, P_1) \leq v(f, P_2)$.

□

In addition, the Dirichlet form is related to the right spectral gap of the transition matrix as is crystallised in the following proposition.

Proposition 3. *(Ordering of Dirichlet forms implies ordering of right spectral gap.) Let* P_1 *and* P_2 *be reversible transition matrices with the same invariant distribution* π. *If* $\mathcal{E}_{P_1}(f) \geq \mathcal{E}_{P_2}(f)$ *for all* $f \in \mathsf{L}^2(\pi)$, *then*

$$\mathrm{Gap}(P_1) \geq \mathrm{Gap}(P_2).$$

Proof. If $\mathcal{E}_{P_1}(f) \geq \mathcal{E}_{P_2}(f)$ for all $f \in \mathsf{L}^2(\pi)$ then, inspired by the functional relationship established in Lemma 6, it follows that

$$\min_{f \in \mathsf{L}^2(\pi)} \mathcal{E}_{P_1}(f) \geq \min_{f \in \mathsf{L}^2(\pi)} \mathcal{E}_{P_2}(f).$$

Naturally the inequality still holds over the particular subset of functions

$$\{f : f \in L^2(\pi), f \perp_\pi 1, \|f\|_2 = 1\},$$

whence it is clear that $\mathrm{Gap}(P_1) \geq \mathrm{Gap}(P_2)$.

□

We note here that an ordering of the right spectral gap does not necessarily imply any ordering on the speed of convergence, which depends on the absolute spectral gap. It has this implication only if the right spectral gap coincides with the absolute spectral gap. Where P is self-adjoint, equivalent to reversibility, then it must have a real spectrum. If in addition it

satisfies $\langle Pf, f\rangle \geq 0$ for all $f \in \mathsf{L}^2(\pi)$ then P is called *positive* and is guaranteed to have a non-negative spectrum [35]. Reversibility of P is key, since in this scenario there are strategies for forcing the spectrum to be non-negative, e.g. using so-called 'lazy' chains where $P(x,x) \geq 1/2, x \in \Omega$. Resolving this difference between right spectral gap and absolute spectral gap for non-reversible operators P directly, without using techniques like lazy chains, is a challenging avenue of current research and some headway has been made [35, 36].

3.3 Majorisation

We move towards proving a result concerning ordering of Dirichlet forms for the rejuvenated pseudo-marginal method. Some concepts relating to majorisation and Schur-convex functions must be introduced first as they will be essential to the proofs that follow. The results in this section are both found in [37].

Definition 8. *For $x, y \in \mathbb{R}^n$, we say x* majorises *y (written $x \prec y$) if*

$$\begin{cases} \sum_{i=1}^{k} x_{[i]} \leq \sum_{i=1}^{k} y_{[i]}, & k \in \{1, \ldots, n-1\}, \\ \sum_{i=1}^{n} x_{[i]} = \sum_{i=1}^{n} y_{[i]} \end{cases}$$

where $x_{[1]} \geq \ldots \geq x_{[n]}$ denotes the components of x in decreasing order.

Definition 9. *A real-valued function ϕ defined on a set $\mathscr{A} \subset \mathbb{R}^n$ is said to be* Schur-convex *on $\mathscr{A}$ if*

$$x \prec y \text{ on } \mathscr{A} \Rightarrow \phi(x) \leq \phi(y).$$

Similarly ϕ is said to be Schur-concave *on $\mathscr{A}$ if*

$$x \prec y \text{ on } \mathscr{A} \Rightarrow \phi(x) \geq \phi(y).$$

3.4 Rate of convergence of iterated rejuvenation

We are interested in what improvements can be made to the convergence rate of the pseudo-marginal algorithm when one implements a rejuvenation step. It will first be crucial however to understand the convergence properties of the rejuvenation kernel itself. Our efforts here are towards ordering Dirichlet forms related to the algorithm, since from this follows immediately the ordering of the right spectral gap and the asymptotic variance. The ordering of the Dirichlet forms will use the techniques of majorisation introduced in the previous section.

First define the N-simplex $\mathscr{S}_N := \{\upsilon \in [0,1]^N : \sum_{i=1}^N \upsilon(i) = 1\}$. For the pseudo-marginal algorithm in which one weight is used at each time step and N weights are used in the rejuvenation pool, the standard rejuvenation procedure is as follows. Suppose we start at (x,w) at time t, then

1. Set $w_1 \leftarrow w$ and sample $w_i \overset{iid}{\sim} Q_x(\cdot),\ i \in \{2,\ldots,N\}$
2. Renormalise the weights $\bar{w}_i := \frac{w_i}{\sum_{j=1}^N w_j}$
3. Sample $l \sim P(\bar{w}_1,\ldots,\bar{w}_N)$
4. Return (x, w_l).

This is an application of the scale-up and scale-down kernels. Recall $\hat{P}_N(w;dw_{1:N}) = \frac{1}{N}\sum_{k=1}^N Q_x^N(dw_{-k} \mid w_k)\delta_w(w_k)$ and $\check{P}_N(w_{1:N};u) = \sum_{l=1}^N \frac{w_l}{\sum_{j=1}^N w_j}\delta_u(w_l)$. We assume the probability measure associated with the weights is of the form

$$Q_x^N(\mathrm{d}w_{-k} \mid w_k) = \prod_{i=1, i\neq k}^N q(\mathrm{d}w_i)$$

as it will be e.g. in the importance sampling case. The marginal transition from w to u is given by

$$\begin{aligned} R_N(w;u) &= \int_{\mathbb{R}_+^N} \hat{P}_N(w;\mathrm{d}w_{1:N})\check{P}_N(w_{1:N};u) \\ &= \sum_{k,l=1}^N \frac{1}{N}\int_{\mathbb{R}_+^N} \prod_{i=1,i\neq k}^N q(\mathrm{d}w_i)\frac{w_l}{\sum_{j=1}^N w_j}\delta_u(w_l)\,\delta_w(\mathrm{d}w_k) \end{aligned}$$

whence the appropriate Dirichlet form for the rejuvenation kernel $R_N(w;u)$ is (half of):

$$\begin{aligned} \mathcal{E}(f) &= \int_{\mathbb{R}_+^2} \pi_x(\mathrm{d}w)\,R_N(w;\mathrm{d}u)\,[f(w)-f(u)]^2 \\ &= \int_{\mathbb{R}_+^2} \pi_x(\mathrm{d}w)\sum_{k,l=1}^N \frac{1}{N}\int_{\mathbb{R}_+^N} \prod_{i=1,i\neq k}^N q(\mathrm{d}w_i)\frac{w_l}{\sum_{j=1}^N w_j}\delta_u(w_l)\,\delta_w(\mathrm{d}w_k)\,[f(w)-f(u)]^2 \\ &= \pi_x(\mathrm{d}w_k)\sum_{k,l=1}^N \frac{1}{N}\int_{\mathbb{R}_+^N} \prod_{i=1,i\neq k}^N q(\mathrm{d}w_i)\frac{w_l}{\sum_{j=1}^N w_j}\,[f(w_k)-f(w_l)]^2 \\ &= \sum_{k,l=1}^N \int_{\mathbb{R}_+^N} \prod_{i=1}^N q(\mathrm{d}w_i)\frac{\frac{1}{N}w_k\frac{1}{N}w_l}{\sum_{j=1}^N \frac{1}{N}w_j}\,[f(w_k)-f(w_l)]^2. \end{aligned} \tag{3.7}$$

Inspired by the form of 3.7, for $\alpha = (\alpha_1, \ldots, \alpha_N)^{\mathrm{T}} \in \mathscr{S}_N$ define the general Dirichlet form

$$\Psi(\alpha) := \sum_{k,l=1}^{N} \int_{\mathbb{R}_+^N} \prod_{i=1}^{N} q(\mathrm{d}w_i) \frac{\alpha_k w_k \alpha_l w_l}{\sum_{j=1}^{N} \alpha_j w_j} [f(w_k) - f(w_l)]^2.$$

We now present a theorem on the structure of this function.

Theorem 4. *$\Psi(\alpha)$ is Schur concave, that is, for any $\alpha_1, \alpha_2 \in \mathscr{S}_N$,*

$$\alpha_1 \prec \alpha_2 \;\Rightarrow\; \Psi(\alpha_1) \geq \Psi(\alpha_2).$$

Proof. Define $F_{i,j} = (f(w_i) - f(w_j))^2$, $\mathbf{w} = (w_1, \ldots, w_N)^{\mathrm{T}}$, and $M = \mathrm{diag}(\mathbf{w}) \times F \times \mathrm{diag}(\mathbf{w})$. Notice that since $M_{ij} = w_i w_j F_{ij}$ we have $\alpha^{\mathrm{T}} M \alpha = \sum_{k,l=1}^{N} \alpha_k w_k \alpha_l w_l F_{kl}$ and so

$$\Phi(\alpha) := \frac{\alpha^{\mathrm{T}} M \alpha}{\alpha^T \mathbf{w}} = \sum_{k,l=1}^{N} \frac{\alpha_k w_k \alpha_l w_l}{\sum_{j=1}^{N} \alpha_j w_j} [f(w_k) - f(w_l)]^2. \tag{3.8}$$

That is,

$$\Psi(\alpha) = \int \Phi(\alpha) \prod_{i=1}^{N} q(\mathrm{d}w_i).$$

We are interested in the Hessian of $\Phi(\alpha)$, and to this end first derive its gradient. A straightforward application of the chain rule yields

$$\begin{aligned} \frac{\partial \Phi(\alpha)}{\partial \alpha_i} &= \frac{2w_i}{\alpha^T \mathbf{w}} \sum_{k=1}^{N} \alpha_k w_k F_{i,k} - \frac{w_i}{(\alpha^T \mathbf{w})^2} \sum_{k,l=1}^{N} \alpha_k w_k \alpha_l w_l F_{k,l} \\ &= \frac{2\,(M\alpha)_i}{\alpha^{\mathrm{T}} \mathbf{w}} - \frac{\left(\alpha^{\mathrm{T}} M \alpha\right) w_i}{\left(\alpha^{\mathrm{T}} \mathbf{w}\right)^2}, \end{aligned}$$

from which it follows that

$$\nabla \Phi(\alpha) = \frac{2M\alpha}{\left(\alpha^{\mathrm{T}} \mathbf{w}\right)} - \frac{\left(\alpha^T\right) w}{\left(\alpha^{\mathrm{T}} \mathbf{w}\right)^2}.$$

The second derivative is found similarly. First notice

$$\frac{\partial \Phi(\alpha)}{\partial \alpha_i} = \frac{1}{\alpha^{\mathrm{T}} \mathbf{w}} \left[2\,(M\alpha)_i - \frac{\left(\alpha^{\mathrm{T}} M \alpha\right) w_i}{\alpha^{\mathrm{T}} \mathbf{w}} \right] =: \frac{1}{\alpha^{\mathrm{T}} \mathbf{w}} G(\alpha)$$

where $G(\alpha) \coloneqq 2\left(M\alpha\right)_i - \frac{\left(\alpha^\mathsf{T} M\alpha\right) w_i}{\alpha^\mathsf{T}\mathbf{w}}$ so that

$$\frac{\partial^2\Phi(\alpha)}{\partial\alpha_j\partial\alpha_i} = \frac{1}{\alpha^\mathsf{T}\mathbf{w}}\left\{\frac{\partial G(\alpha)}{\partial\alpha_j} - \frac{w_j}{\alpha^\mathsf{T}\mathbf{w}}G(\alpha)\right\}. \tag{3.9}$$

Now,

$$\begin{aligned}\frac{\partial G(\alpha)}{\partial\alpha_j} &= 2w_iw_jF_{i,j} + \frac{\left(\alpha^\mathsf{T} M\alpha\right)w_iw_j}{\left(\alpha^\mathsf{T}\mathbf{w}\right)^2} - 2\frac{w_iw_j}{\alpha^\mathsf{T}\mathbf{w}}\sum_{k=1}^{N}\alpha_kw_kF_{j,k}\\ &= 2M_{i,j} + \frac{w_iw_j}{\left(\alpha^\mathsf{T}\mathbf{w}\right)^2}\left(\alpha^\mathsf{T} M\alpha\right) - 2\frac{w_i}{\alpha^\mathsf{T}\mathbf{w}}\left(M\alpha\right)_j.\end{aligned}$$

Substituting into equation 3.9 shows

$$\begin{aligned}\frac{\partial^2\Phi(\alpha)}{\partial\alpha_j\partial\alpha_i} &= \frac{1}{\alpha^\mathsf{T}\mathbf{w}}\left\{2M_{i,j} + \frac{w_iw_j}{\left(\alpha^\mathsf{T}\mathbf{w}\right)^2}\left(\alpha^\mathsf{T} M\alpha\right) - 2\frac{w_i}{\alpha^\mathsf{T}\mathbf{w}}\left(M\alpha\right)_j - \frac{w_j}{\alpha^\mathsf{T}\mathbf{w}}\left(2\left(M\alpha\right)_i - \frac{\left(\alpha^\mathsf{T} M\alpha\right)w_i}{\alpha^\mathsf{T}\mathbf{w}}\right)\right\}\\ &= \frac{2}{\alpha^\mathsf{T}\mathbf{w}}\left\{M_{i,j} + \frac{w_iw_j}{\left(\alpha^\mathsf{T}\mathbf{w}\right)^2}\left(\alpha^\mathsf{T} M\alpha\right) - \frac{1}{\alpha^\mathsf{T}\mathbf{w}}\left(w_i\left(M\alpha\right)_j + w_j\left(M\alpha\right)_i\right)\right\}.\end{aligned}$$

In summary, by a simple rearrangement where we have used the symmetry of M,

$$\begin{aligned}\nabla^2\Phi(\alpha) &= \frac{2}{\alpha^\mathsf{T}\mathbf{w}}\left\{M + \frac{\mathbf{w}\alpha^\mathsf{T} M\alpha\mathbf{w}^\mathsf{T}}{\left(\alpha^\mathsf{T}\mathbf{w}\right)^2} - \frac{M\alpha w^\mathsf{T} + w\alpha^\mathsf{T} M}{\alpha^\mathsf{T}\mathbf{w}}\right\}\\ &= 2\left(I - \frac{w\alpha^\mathsf{T}}{\alpha^\mathsf{T}\mathbf{w}}\right)\frac{M}{\alpha^\mathsf{T}\mathbf{w}}\left(I - \frac{\alpha w^\mathsf{T}}{\alpha^\mathsf{T}\mathbf{w}}\right).\end{aligned}$$

We now show that the Hessian matrix $\nabla^2\Phi(\alpha)$ is negative-semidefinite, i.e. $\mathbf{h}^T\nabla^2\Phi(\alpha)\mathbf{h} \le 0$ for any $\mathbf{h} \in \mathbb{R}^N$. To see this, introduce the notation $B \coloneqq \operatorname{diag}(\mathbf{w})\left(I - \frac{\alpha w^\mathsf{T}}{\alpha^\mathsf{T}\mathbf{w}}\right)$ from which it follows that $\nabla^2\Phi(\alpha) = \frac{2}{\alpha^\mathsf{T}\mathbf{w}}B^\mathsf{T} FB$. It is now prescient to observe that, for any $\mathbf{h} \in \mathbb{R}^N$, the

vector Bh is orthogonal to the vector consisting of all 1s, i.e. $(Bh)^\mathsf{T}\mathbf{1} = 0$. Indeed,

$$\begin{aligned}(Bh)^\mathsf{T}\mathbf{1} &= \sum_{i,j=1}^{N} B_{i,j}h_j \\ &= \sum_{i,j=1}^{N}\left(w_i\delta_{i,j} - \frac{w_i w_j \alpha_i}{\alpha^\mathsf{T}\mathbf{w}}\right)h_j \\ &= \mathbf{w}^\mathsf{T}h - \frac{\left(\sum_{i=1}^N w_i\alpha_i\right)\left(\sum_{j=1}^N w_j h_j\right)}{\alpha^\mathsf{T}\mathbf{w}} \\ &= 0.\end{aligned}$$

Denote $\mathbf{u} \coloneqq (f(y_1),\ldots,f(y_N))^\mathsf{T}$ and $\mathbf{v} \coloneqq (f^2(y_1),\ldots,f^2(y_N))^\mathsf{T}$, then $F = \mathbf{v}\mathbf{1}^\mathsf{T} + \mathbf{1}\mathbf{v}^\mathsf{T} - 2\mathbf{u}\mathbf{u}^\mathsf{T}$. Finally we have for any $\mathbf{h} \in \mathbb{R}^N$

$$\begin{aligned}\mathbf{h}^\mathsf{T}\nabla^2\Phi(\alpha)\mathbf{h} &= \frac{2}{\alpha^\mathsf{T}\mathbf{w}}\mathbf{h}^\mathsf{T}B^\mathsf{T}\left\{\mathbf{v}\mathbf{1}^\mathsf{T} + \mathbf{1}\mathbf{v}^\mathsf{T} - 2\mathbf{u}\mathbf{u}^\mathsf{T}\right\}B\mathbf{h} \\ &= \frac{2}{\alpha^\mathsf{T}\mathbf{w}}\left\{\mathbf{h}^\mathsf{T}B^\mathsf{T}\mathbf{v}\left(\mathbf{1}^\mathsf{T}B\mathbf{h}\right) + \left(\mathbf{h}^\mathsf{T}B^\mathsf{T}\mathbf{1}\right)\mathbf{v}^\mathsf{T}B\mathbf{h} - 2\mathbf{h}^\mathsf{T}B^\mathsf{T}\mathbf{u}\mathbf{u}^\mathsf{T}B\mathbf{h}\right\} \\ &= -\frac{4}{\alpha^\mathsf{T}\mathbf{w}}\mathbf{h}^\mathsf{T}B^\mathsf{T}\mathbf{u}\mathbf{u}^\mathsf{T}B\mathbf{h} \\ &= -\frac{4}{\alpha^\mathsf{T}\mathbf{w}}\left(\mathbf{u}^\mathsf{T}B\mathbf{h}\right)^\mathsf{T}\left(\mathbf{u}^\mathsf{T}B\mathbf{h}\right) \\ &\leq 0\end{aligned}$$

where we have used the fact that α is positive (by definition of the simplex) and so is $\mathbf{w}$ as it is a pseudo-marginal weight. Therefore $\Phi(\alpha)$ is concave in α for each fixed vector $\mathbf{w}$. Moreover, we assume (as is done throughout this thesis) the set of weights $w_1,\ldots,w_N \sim Q_x^N(\cdot)$ is N-exchangeable. Note also that $\Phi(\mathbf{w}\Pi;\alpha\Pi) = \Phi(\mathbf{w};\alpha)$ for all permutations Π, which can be seen by examining the expression 3.8, and that $\Phi(\mathbf{w};\alpha)$ is Borel measurable in $\mathbf{w}$ for each fixed α. Therefore $\Psi(\alpha) = \mathbb{E}_{Q_x^N}[\Phi(\mathbf{W};\alpha)]$ is symmetric and concave [37, Proposition B.1, p. 393] and it follows that $\Psi(\alpha)$ is Schur-concave [37, Proposition C.2, p. 97].

□

From the theorem we can conclude that $k \mapsto \Psi(u_k)$ is non-decreasing. We would like from this statement about the rejuvenation kernel to be able to say something about the spectral gap associated with the rejuvenated pseudo-marginal algorithm explored in Chapter 2. Recall that the pseudo-marginal averaging N weights at each step had a transition kernel denoted $\tilde{P}_N$ and that therefore the rejuvenated version of the same algorithm possessed a transition kernel $\tilde{P}_N^\lambda = R_N\tilde{P}_N$. Now, for example, we speculate that $k \mapsto \text{Gap}\left(\tilde{P}_{u_k}^\lambda\right)$ is non-decreasing, but this

is not yet clear. We may however be sure from the theorem that the iterated rejuvenation kernel has an asymptotic variance that is non-increasing in k and a right spectral gap that is non-decreasing in k. We hope that the techniques contained in this chapter may form the basis of future research that may, for example, show the function $k \mapsto \Psi(u_k)$ to be concave.

Chapter 4

The Coalescent

In this chapter we turn our attention away from pure statistical theory towards an important biological application of Monte Carlo methods. Experimental advances of the 1990s allowed for the first time sequencing of entire genomes, precipitating enormous general interest in the previously arcane study of population genetics [38]. Prior to this the discipline had been largely theoretical with a focus on probabilistic modelling. However with the sudden availability of huge quantities of genome data at the DNA level new statistical challenges arose, which along with access to rapidly improving computational capabilities gave life to the discipline of 'population genomics.' A panoply of techniques has since been developed, largely aiming to take current DNA sequence data of a population or subset thereof, and from it to infer facets of the ancestral past of that population, or the parameters of the stochastic model assumed to have generated the genetic data. Some of these techniques will be discussed in this chapter and the next. First we focus on some fundamental models in population genetics that underpin the more complex models we explore later. Much of the following review is inspired by [39].

4.1 The Wright-Fisher model

We begin by discussing the Wright-Fisher model, introduced in [40] and [41]; possibly the simplest model for describing the evolution of a finite population of genes through discrete generations. Consider a population of N diploid individuals (carriers of two sets of genetic information), which it is traditional to model as $2N$ haploid individuals (carriers of a single set of genetic information). At this stage we refer loosely to 'individuals' or 'members' of a population, by which we may mean a DNA sequence, gene, locus, or base - this distinction is presently of little importance. The primary assumptions (though there are other implicit assumptions) underlying a basic Wright-Fisher model are the following:

1. The generations in the model are discrete and do not overlap, that is, all individuals in a generation reproduce at the same time and simultaneously die.
2. The population size is fixed.
3. There are no considerations of fitness of individuals (selection).
4. There is no population structure, e.g. geographical or sexes.
5. There is no recombination or mutation.

Naturally none of these assumptions hold for real populations, but the model offers a simple base on top of which more complicated models may be imagined. Population size and structure, for example, are often regarded as crucial factors in the genetic history of a population and so are commonly included in practice. For our purposes the most limiting assumption of the model, and one that we will shortly relax, is that of no recombination or mutation. These additions do however significantly complicate the mathematics and for this reason are best avoided at first.

The Wright-Fisher model for reproduction proceeds, forward in time, as follows. Suppose we have reached generation $t \geq 1$, then generation $t+1$ comprises $2N$ genes copied uniformly at random from the $2N$ genes in generation t. In other words, each new generation is constituted of clones of members of the current generation sampled randomly with replacement. Thus, every individual in generation $t+1$ has a parent in generation t, but a randomly selected individual in generation t will not necessarily have offspring in generation $t+1$, and if it does not then its lineage is said to have died out. As generations elapse what emerges is a population of genes, each of which has an ancestry.

Summarising [39, Section 1.4], label the genes in generation t by $i \in \{1,2,\ldots,2N\}$. When generation t reproduces, individual i produces $v_i \in \{0,1,\ldots,2N\}$ offspring in generation $t+1$. Equivalently, for individual i to have k offspring in generation $t+1$, k members of generation $t+1$ must 'pick' individual i in generation t as their parent, which they do independently with probability $1/2N$. One can simulate such a model and represent its evolutionary dynamics using a simple graph, as is shown (on the left) in Figure 4.1; the colours represent distinct alleles, each of which in this case is unique in the population when the model begins since no colour is repeated. This example shows how, at least when the population size is small, in relatively few generations one gene can become ubiquitous in the population through *genetic drift* - i.e. changes in frequency of a particular allele or gene in a population due simply to random sampling and not, for example, the relative fitness of the allele. Generation $t+1$ selecting their parents in generation t is in this sense similar to drawing balls with replacement from an urn, a classic exercise in probability theory. The balls, too, possess no qualities

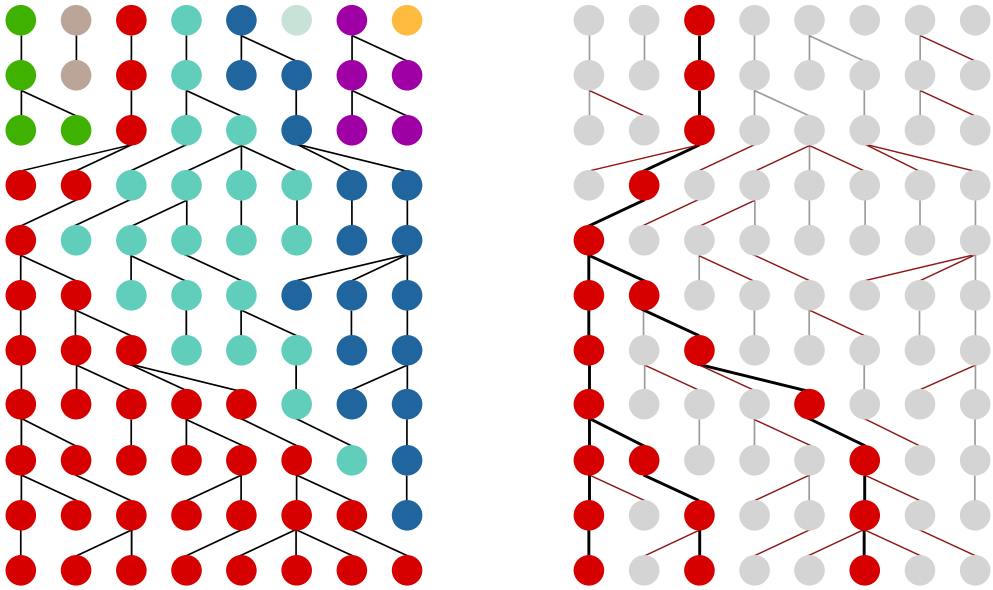


Figure 4.1: *Left: A realisation of a Wright-Fisher model for a population of eight individuals. After eleven generations the allele shown in red has become ubiquitous through genetic drift. Right: Tracing the ancestral history of three samples from the population. Sequences one and three coalesce in three generations (measured backwards in time from the present). Sequence six coalesces with this lineage three generations later.*

of 'fitness.' Yet if $2N$ balls of various colours were placed in an urn, and removed (with replacement) to form a new urn of balls, after a small number of iterations of this process the resulting urn would contain only one colour of ball.

A probabilistic framing of this drift is that the prevalence $v_i(t)$ of allele i considered across generations $\{t, t+1, \ldots\}$ may be described as an irreducible, finite state space Markov chain. It is irreducible since the state space (with two exceptions) constitutes one communicating class; this means that providing there is at least one member of the population with allele X, it is possible in a finite number of generations for all members of the population to possess allele X, and vice versa. The Markov chain also has absorbing states 0 and $2N$ - in other words, once one of these two states is reached the chain remains perpetually in that state. It is bound to reach one of these absorbing states in finite time since the chain is necessarily positive recurrent by virtue of being finite and irreducible [42].

Crucially, the interpretation of the model as having children 'pick' their parents provides instructions for running the model in reverse, that is, starting with a subset of the population and moving backwards in time disregarding other members of the population. With time moving by generations in reverse, when two individuals in a generation choose the same parent their lineages are said to *coalesce*; recording all such lineages and coalescences, one will

eventually find a most recent common ancestor (MRCA), a single individual from which all members of the present sample are descended. The right hand plot in Figure 4.1 shows the process of tracing the MRCA of 3 individuals from a population of eight.

Consider two genes in the present population, and the time T in generations until a MRCA is found. Whatever parent the first gene chooses, the second gene chooses the same with probability $1/2N$, that is, a Bernoulli trial with probability $p = 1/2N$ of 'success.' If the two do not share a common parent, then one simply looks back one generation further. Given that they do not share a common parent, whether they share a common grandparent is an independent Bernoulli trial with exactly the same probability of success/failure. In other words $T \sim \text{Geom}(p)$ and

$$P(T = k) = (1-p)^{k-1}p$$

where the probability of success is simply the probability of sharing a parent, that is $p = 1/2N$. Consider next a generalisation; the waiting time for k genes to have fewer than k ancestral lineages. Take first the complement of this event. In order for k genes to each have a distinct parent the following sequence of events must occur. The first gene is free to choose its parent, the second must choose a different parent with probability $(2N-1)/(2N)$, the third must choose a parent distinct from the first two with probability $(2N-2)/2N$, and so on. Therefore the probability that k genes have k distinct parents is

$$\begin{aligned}\prod_{i=1}^{k-1}\left(1-\frac{i}{2N}\right) &= 1-\sum_{i=1}^{k-1}\frac{i}{2N}+O\left(\frac{1}{N^2}\right)\\ &= 1-\binom{k}{2}\frac{1}{2N}+O\left(\frac{1}{N^2}\right).\end{aligned}$$

Assuming one considers only values of k that are small relative to $2N$ then, approximately speaking, the probability of no coalescence in k genes in one generation is $1-\binom{k}{2}\frac{1}{2N}$ and the probability of a coalescence event between two genes is $\binom{k}{2}\frac{1}{2N}$. Note that this approximation precludes more than one pair of genes simultaneously finding a common ancestor, though in practice it will make little appreciable difference for $N > 50$. Analogously to the $k = 2$ case we can talk about the distribution of the time T_k for the first coalescence of any pair of genes out of a subpopulation of size k. Again this is simply a geometric distribution, namely $T_k \sim \text{Geom}\left(\binom{k}{2}/2N\right)$.

It follows from the structure of the model that any subset of genes will find a MRCA in a finite number of generations - though this may be large if N is large. Indeed, the time Υ_k to the MRCA (which we will abbreviate to TMRCA) of k genes is given by $\Upsilon_k = T_k + T_{k-1} + \ldots + T_2$,

where $T_i \sim \text{Geom}\left(\binom{i}{2}/2N\right)$, and therefore

$$E(\Upsilon_k) = \sum_{i=2}^{k} E(T_i) = 4N(1-1/k) < \infty.$$

It follows that $\Upsilon_k < \infty$ a.s. for all $k < \infty$.

4.2 The coalescent model

We consider now a particular limiting case of this model. Fix a generation $j \in \{1,2,\ldots\}$, an integer $M \in \mathbb{N}_+$, and $p \in (0,1)$. Define a constant $a := pM$, a rescaled time $t := j/M$ and a rescaled TMRCA $T^c := T/M$. Recall that for any real a we have

$$e^a = \lim_{M\to\infty}\left(1+\frac{a}{M}\right)^M,$$

and that from the properties of the geometric distribution, for $T \sim \text{Geom}(p)$ one has that $P(T \geq j) = (1-p)^{j-1}$. Substituting the change of variables one finds

$$\begin{aligned}\mathbf{P}(T^c \geq t) &= \mathbf{P}(T \geq j)\\ &= (1-p)^{tM-1}\\ &= \left[\left(1-\frac{a}{M}\right)^M\right]^t\left(1-\frac{a}{M}\right)^{-1}\end{aligned}$$

and that, taking the limit, $\mathbf{P}(T^c \geq t) = e^{-at}$ as $M \to \infty$. Thus, the rescaled TMRCA is well approximated in distribution by an exponential random variable with rate a. This describes a Wright-Fisher model that in the limit is continuous in time. Indeed, let t be scaled by $M = 2N$ when N is sufficiently large, so that time is measured in units of $2N$ generations. Then, one can convert a number of generations j into a continuous time t by the identity $t = j/2N$. Equally, a continuous time t is converted to generations by $j = \lfloor 2Nt \rfloor$. We also have

$$a = \frac{\binom{k}{2}}{2N} \times 2N = \binom{k}{2}$$

so that $\mathrm{P}(T_k \geq t) = \exp(-\binom{k}{2}t)$, i.e. $T_k \sim \text{Exp}(\binom{k}{2})$. Algorithm 4.1, which generates stochastically a genealogy for k genes under the continuous coalescent model, is now adapted from [39]. We refer to it as the *basic coalescent* (in Kingman's work [43] this is referred to as the n-coalescent). The limit argument presented above is the essence of coalescent models -

Algorithm 4.1 The basic coalescent

Input: a sample of k genes.

1. Sample $T_k \sim \text{Exp}(\binom{k}{2})$, the time to the next coalescent event.
2. Sample uniformly at random a pair (i, j), $1 \leq i < j \leq k$.
3. Genes i and j coalesce after time T_k and the sample size is reduced by one, that is, $k \leftarrow k-1$.
4. If $k \geq 2$ return to step 1, else stop.

Output: an ancestral history of the k genes; the coalescence times and which lineages coalesce.

they are population level models of reproduction that view time as moving in reverse from the present towards the past continuously. They arise naturally as the continuous-time limit to a range of discrete-time models like the Wright-Fisher and Moran models [44], and are for this reason considered to be robust [39, 43, 45].

The coalescent method offers at least two key insights [46]. The first insight is that there is a deep consequence of supposing mutations to be selectively neutral. Mutations need not be generated, as presented here, as a competing process to coalescence that is generated at the same time as the topology, but can instead be overlaid after the fact. They are independent of the structure, or topology, of the ancestral history. Mutations can be simulated according to a Poisson process and superimposed on the genealogy after the fact [47, Chapter 20]. This is equivalent to algorithm 4.2 when the mutations are selectively neutral.

The second insight is that sampling the ancestral history of a sample of genes, or individuals, does not require sampling the history of the entire population. This follows from viewing the Wright-Fisher model as each member of a generation 'picking' their parent. In fact the coalescent is extraordinarily computationally efficient, at least when compared to simulating the entire population forward in time and then tracing lineages back from the final generation [46]. Rather than pick out the ancestral tree of interest from the vast web of population parent-child relationships, it is instead drawn directly; this simplicity is incredibly useful but also offers a profound heuristic. Returning to Figure 4.1, the left figure shows a forward in time Wright-Fisher style model for a population of size eight. On the other hand, the right figure shows in a fashion similar to the coalescent the ancestral history of individuals one, three, and

six in the present day population. Note that in going from one view to the other, most of the population reproduction information becomes irrelevant.

4.3 Mutations

In the Wright-Fisher model, genes were related at the population reproduction level through a shared ancestry. To understand the biology of reproduction at a deeper level it is necessary to consider the effect of mutations. There are several biological models of mutations including the infinite alleles model, the infinite sites model, and the finite sites model [48–50]. We focus here on the infinite sites model.

4.3.1 The infinite sites model

Under an infinite sites model it is assumed that, relative to the length of the genes considered, the number of single nucleotide polymorphisms (SNPs) is small. If this assumption holds, it is reasonable to expect in practice that each new mutation will occur at a new position in the sequence. The model therefore assumes that mutations cannot occur at the same site twice. Equivalently, every mutation affects a new site. This is in some sense quite natural in the context of the coalescent model described above where the sequence is taken to be a continuous interval, with mutations being placed on top by a continuous probability distribution.

A number of consequences follow from the infinite sites assumptions. First, every mutation that has occurred in the ancestral history of the population is known - in particular it is observed through the mutations in the genes. This is in contrast to a finite sites model for example, where a mutation could occur at the same site twice, say A $\rightarrow$ G and G $\rightarrow$ A, which would then be imperceptible through observation alone.

Secondly, considering the same position in the genome across a set of sequences (a sample from a population, say) there will only ever be at most two states observed, since more would require more than one mutation to occur at this base. For this reason, under the infinite sites model, sets of sequences are often written in terms of 0s and 1s. 'Translating' from genetic sequences (ATGC...) to this code is simple; fix the DNA corresponding to one sample as the zero-type, i.e. all zeros. Then proceed base by base with each additional sample. Where the bases are identical - A,A or T,T for example - then the sample being compared would also receive a zero in this position. Where the bases differ, the sample would receive a one. Take

as an example the fragment of genetic sequence for two individuals

GACATC...
TACAGC...

which can be converted to

000000
100010

where we have assumed the first sequence to be the 'ancestral' state - the state considered to be primary and from which others are derived by mutation.

Throughout this thesis we will work with an infinite sites model and under the assumption that mutations are 'selectively neutral' - in other words, no genetic configuration out-performs any other in terms of fitness. The probability of passing down a mutation does not depend on the type held by the parent, and inheriting a mutation does not affect survival.

4.3.2 The basic coalescent with mutation

Mutation can be factored into the model similarly to coalescence. Assume that a mutation at a given base occurs with constant probability μ per individual per generation; a *scaled mutation rate* is typically defined by $\theta := 4N\mu$. Then, measured in generations, the time T_m until the next mutation event for a single individual will be a geometric random variable with expected value $1/\mu$ and is approximated in continuous time by an exponential random variable with rate $\theta/2$. To see this, define a constant $a := 2\mu N$, rescaled time $t := j/2N$, and a rescaled time to the first mutation of an individual $T_m^c := T_m/2N$. Let $N \to \infty$ and $\mu \to 0$, and observe

$$\begin{aligned}
\mathbf{P}(T_m^c \geq t) &= \mathbf{P}(T_m \geq j) \\
&= (1-\mu)^{2Nt-1} \\
&= \left[\left(1-\frac{a}{2N}\right)^{2N}\right]^t \left(1-\frac{a}{2N}\right)^{-1} \\
&\approx e^{-at}.
\end{aligned}$$

where $a = 2\mu N = \theta/2$.

Let us return to the basic coalescent algorithm and augment it with mutations. Beginning again with a present subpopulation of k individuals, and looking backwards in time, the first event that occurs could be a mutation in any of the k lineages or a coalescence between any

pair. Since for independent random variables $Y_i \sim \text{Exp}(\lambda_i)$, $i \in \{1,2,\ldots,n\}$ one has

$$\min\{Y_1,\ldots,Y_n\} \sim \text{Exp}(\sum_{i=1}^{n} \lambda_i)$$

it follows that the distribution of the first event time in our basic coalescent model with mutation is exponential with rate

$$k\frac{\theta}{2} + \binom{k}{2} = \frac{k(k-1+\theta)}{2}.$$

The particular event that will take place at this time is chosen with probability proportional to its contribution to the rate. In other words, the event will be a mutation with probability

$$\frac{k\theta/2}{\binom{k}{2} + k\theta/2} = \frac{\theta}{k-1+\theta}$$

and a coalescence with probability

$$\frac{\binom{k}{2}}{\binom{k}{2} + k\theta/2} = \frac{k-1}{k-1+\theta}.$$

An algorithm for simulating a genealogy of k genes under the basic coalescent with mutation is given by Algorithm 4.2.

We dedicate time here to labour an important point. Coalescent algorithms like Algorithm 4.2 are generic procedures in which is fed an input, namely k genes from a present subpopulation of interest, and which stochastically generate backwards in time a genealogy, or *history*, $\mathcal{H} \in \mathscr{T}(k)$. Here we define $\mathscr{T}(k)$ to be the entire space of such genealogies or *trees* that are permissible under the particular coalescent model of interest, starting from a present subpopulation of k individuals. Once such a tree is generated it can be run forward in time from the MRCA (with any sequence as the MRCA) and the resulting genealogy and offspring will be perfectly coherent with respect to the model. However, in practice our interest will often be in asking, given a particular realisation of observed genetic data on k present individuals (suppose for instance that we have sequenced the genomes of k people), what genealogy or set of genealogies is likely to have generated those data. There is no simple adaptation of the coalescent to this requirement. Indeed, a naive approach may fail at the first hurdle - imagine running Algorithm 4.2 in reverse from an observed sample of k genes and further that the first event is randomly chosen to be a coalescence between two non-identical individuals. In summary, for some problems of inference we may require access to the subset $\mathscr{T}(k;\mathcal{H}_0) \subset \mathscr{T}(k)$ of trees, those compatible with a particular present sample history $\mathcal{H}_0$, and this refinement is non-trivial.

Algorithm 4.2 The basic coalescent with mutation

Input: a sample of k genes.

1. Start with sample of k genes.
2. Sample $T_k \sim \text{Exp}\left(\frac{k(k-1+\theta)}{2}\right)$, the time to the next event.
3. With probability $\frac{k-1}{k-1+\theta}$ the event at time T_k is a coalescence, otherwise it is a mutation.
4. If it is a coalescence, sample uniformly at random a pair (i, j), $1 \leq i < j \leq k$ that will coalesce, set $k \leftarrow k-1$.
5. If it is a mutation, pick uniformly one of the k lineages on which to place a mutation at T_k.
6. If $k \geq 2$ return to step 2, else stop.

Output: an ancestral history of the k genes.

4.4 The coalescent with recombination

In addition to mutation, an essential feature of real reproduction (of almost all multicellular organisms on Earth [51]) is recombination. It describes the biological process by which a child may inherit a 'shuffled' version of the genetic information of the parents. Imagining a coalescent model, recombination when considered backwards in time has the opposite effect to coalescence, in that it allows a bifurcation of lineages into the past. We will assume this splitting process occurs at a single point in the sequence, referred to as 'crossing-over' at a point [46]. A coalescent model with recombination is shown in Figure 4.2 where, for example, sequence three first experiences a recombination event before eventually all the lineages coalesce further back in time.

Introducing recombination into the model has two major consequences. Primarily, it turns what was a tree topology into a graph, significantly complicating the interpretation and mathematics of the model. In addition, by splitting the ancestral information over many chromosomes backwards in time, the total space of possible graphs that explain the present sample becomes very large. Crucially, we note that the ancestral history of a single position in the sequence is still described by a tree - there is no way a single position can be split across two chromosomes - and it follows that the full ancestral history can be described either as a graph

or a collection of such trees. Mathematically it is challenging to model recombination, and the phenomenon is less well understood than say mutation and several interpretations exist.

The coalescent with recombination was first presented in [52]. Here we describe the closely related *ancestral recombination graph* (ARG) as described in [53] wherein an infinite sites model is adopted. Consider first a discrete Wright-Fisher type model in which is modelled the genetic evolution of a constant population of $2N$ sequences. It is assumed that recombination occurs with constant probability r per individual per generation. The individuals comprising the population are represented by a continuous length of gene or DNA data, denoted by the interval [0,1]. This is simply a convenient representation of a sequence but could easily be mapped back to discrete bases if necessary. Each member of generation $t+1$ samples with probability $1-r$ a single parent from generation t in the usual way, and with probability r samples two parents uniformly at random from generation t and a recombination occurs. A recombination event consists of sampling a break point Z on the gene, where Z takes some continuous distribution over [0,1], and attributing the region $[0,Z)$ to the first parent and $[Z,1]$ to the second. A typical choice is $Z \sim U(0,1)$ to reflect the scenario in which each base on the gene has an equal probability of recombination.

Define the *scaled recombination rate* by $\rho := 4Nr$. Using an analogous limit argument to subsection 4.3.2 it follows that the distribution of the first event time in the ARG is exponential with rate

$$k\frac{\rho}{2} + \binom{k}{2} = \frac{k(k-1+\rho)}{2}.$$

The particular event that will take place at this time is chosen with probability proportional to its contribution to the rate. In other words, the event will be a recombination with probability

$$\frac{k\rho/2}{\binom{k}{2} + k\rho/2} = \frac{\rho}{k-1+\rho}$$

and a coalescence with probability

$$\frac{\binom{k}{2}}{\binom{k}{2} + k\rho/2} = \frac{k-1}{k-1+\rho}.$$

Algorithm 4.3 (adapted from [39]) describes the process for simulating an ancestral recombination graph for a sample of n individuals from a population. With the knowledge that we are free to overlay mutations after the fact, we do not include them here.

Note that the process is guaranteed to reach a most recent common ancestor in finite time since it may be recast as a birth/death process with birth rate proportional to k and death rate proportional to k^2 [54]. If mutations are to be included in the analysis, then they are placed

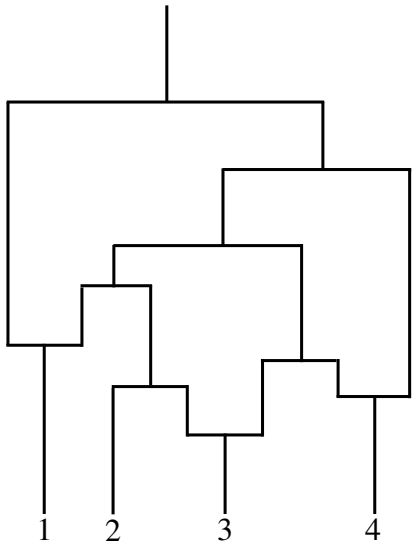


Figure 4.2: *An ancestral recombination graph for four sequences. There are three recombination events and six coalescent events.*

on the edges of the ARG according to a Poisson process of rate $\theta/2$. The position of the mutation on the sequence is chosen uniformly at random and independent of all other such choices. Note it is possible for a mutation to fall in *non-ancestral material*, a section of the gene that is not ancestral to any extant sequence.

Finally, as was briefly alluded to above, when referring to the ancestry of a particular base there is no recombination to consider. Thus each position $x \in [0,1]$ within the length of gene possesses an ancestral tree $\mathscr{T}(x)$ which is simply a realisation of a coalescent tree simulated by the basic coalescent algorithm with mutation (see Algorithm 4.2). This tree is imbedded within the full ARG and can be found as follows: trace backwards in time the ancestry of the sequence on which the base is located, and if a recombination occurs at a point $Z \in [x,1]$ (resp. $Z \in [0,x)$) follow the left (resp. right) path of the bifurcation.

The complexity of the ancestral recombination graph causes some difficulties in practice. Mathematically the process becomes much harder to work with as the likelihood function is intractable, and estimating it is "notoriously difficult" [3] largely due to the enormity of the space of possible latent states (the graphs).

4.5 Recombination as a process along sequences

In [55] it was shown that there is another way to view the coalescent with recombination; not as whole sequences tracing their ancestry backwards through time, but instead moving along the sequences from left to right and modifying the ancestral history when a recombination

Algorithm 4.3 The ancestral recombination graph

Input: a sample of n genes.

1. Set $k = n$.
2. For k sequences with ancestral material, sample $T_k \sim \text{Exp}\left(\frac{k(k-1+\rho)}{2}\right)$, the time to the next event.
3. With probability $(k-1)/(k-1+\rho)$ the event at time T_k is a coalescence, otherwise it is a recombination event.
4. If it is a recombination, sample a random sequence and a random point Z on the sequence. Create two ancestral sequences: one with the ancestral material in $[0,Z)$ and another with the ancestral material in $[Z,1]$. Set $k \leftarrow k+1$ and go to 1.
5. If it is a coalescence, choose at random two of the ancestral sequences and merge them. Set $k \leftarrow k-1$.
6. If $k = 1$ **end**, otherwise go to 1.

Output: an ancestral recombination graph relating the n genes.

point occurs. This 'spatial' algorithm begins at position 0 in the sequence with a standard coalescent tree (see Algorithm 4.1), describing the ancestral history of the sequences at 0. Moving (spatially) across the sequence new recombination points are proposed at which the 'local tree' is modified. Thus the ancestral history of the full sequence may be viewed as a sequence of embedded trees. This approach is essentially equivalent to the ARG described above, a brief argument for which is given in [55]. In essence, conditional on a total branch length measured in generations of $2Nb$, and considering a length of L nucleotides, the number n of recombination points in the length-L segment is binomially distributed $n \sim \text{B}(2NbL, r)$. For large N and L, by the usual limiting arguments used for the coalescent, this number tends to a $\text{Poi}(b\rho/2)$ distribution. It follows that the distance from a point on the sequence to the next recombination point on the sequence is exponentially distributed. This fact is exploited by the spatial algorithm for its movements across the sequence.

We do not go into great depth here about this particular model as in the next chapter we will discuss at length a particular approximation to it. Note however that the approximation

will arise out of a desire to cast the coalescent with recombination as a 'spatial', Markovian state space model. The ARG is an inherently Markovian process viewed backwards in time - one need only think of the interpretation of children picking their parents at each generation. However, the spatial algorithm is not Markovian. Long-range dependencies are induced because the probability distribution of the next local tree depends on previous trees as a sequence created in a recombination event is permitted to coalesce with branches of earlier trees.

With a sufficient understanding of coalescent models we begin in the next chapter to discuss the sequentially Markov coalescent (SMC) and SMC' approximations [3, 4] to the full ARG.

Chapter 5

Monte Carlo inference for SMC'

5.1 Introduction

There have been many proposed computational methods for analysing genetic variation data, and as the volume and complexity of the available data is ever increasing the problem remains as challenging as ever. Many promising strategies have used Monte Carlo techniques. For example, importance sampling approaches have been developed in [53, 56–58] and sequential importance sampling in [38, 59]. The use of Markov chain Monte Carlo (MCMC) and approximate Bayesian computation has also been popular. Some early methods are given in [60, 61], as well as later [1, 62–64].

More recently, in 2005 the highly influential sequentially Markov coalescent model (SMC) was proposed in [3]; it offers a Markovian interpretation of the spatial algorithm of [55] discussed in Chapter 4 and offers a good approximation to the full coalescent machinery of Kingman and Hudson. In [4] an adjustment to the SMC model is made that improves its likeness to the full coalescent and runs faster (see [65] for a study into its accuracy). The pairwise sequentially Markov coalescent was described in [66], and is a particular implementation of the SMC for two chromosomes, though it requires discretisation of time and along the sequence length. A powerful Gibbs-sampling approach termed 'threading' was introduced in [67] to sample from the hidden Markov model underpinning the SMC. In contrast, we will explore here a general and flexible particle MCMC approach.

We begin by reviewing the SMC and closely related SMC' algorithms. These algorithms are commonly used in population genetics to generate possible sequence-level ancestral histories of a set of individuals from a population. Our motivation will be to calculate the likelihood of an observed set of mutations on a genetic sequence, marginal to the ancestral history by which these mutations were brought about. In other words we seek to estimate $\Pr_\vartheta(\mathcal{M})$, where $\mathcal{M}$ denotes the observed mutations assumed to have been generated by a coalescent

model with latent parameters ϑ. Given a particular ancestral history $\mathcal{H}$, which for now may simply be thought of as a graph describing the 'ancestry' of a number of individuals (with respect to this particular genetic sequence), $\Pr_\vartheta(\mathcal{M}|\mathcal{H})$ is tractable. A simple Monte Carlo strategy, then, is to generate many histories $\{\mathcal{H}^{(i)}\}_{i\in[\![1,N]\!]}$ consistent with $\mathcal{M}$ and estimate the quantity of interest by

$$\Pr_\vartheta(\mathcal{M}) \approx \frac{1}{N}\sum_{i=1}^{N} \Pr_\vartheta(\mathcal{M}|\mathcal{H}^{(i)}). \tag{5.1}$$

Here we encounter at least two significant issues. First whilst the proposal of generic ancestral histories for any number of individuals is straightforward under the coalescent (and even simpler under SMC'), generating ancestral histories consistent with a given set of observed mutations is non-trivial. It will be shown that this problem does not arise for two individuals, but will be a central concern for three and above. We propose a solution for the three individual case.

Secondly the realm of possible ancestral histories is vast, so a naive approach (as in Equation (5.1)) may suffer from impracticably large variance (if it is guaranteed to converge at all). To mitigate this issue, we propose a sequential Monte Carlo approach (hereafter referred to as *particle filter* and never SMC). This is motivated by recasting the SMC' dynamics as a piecewise deterministic process (PDP) and investigating recent literature on the PDP particle filter [68, 69]. This will provide the basis for exploration of a novel approach to using particle filters for coalescent inference.

In addition, if an unbiased estimate of $\Pr_\vartheta(\mathcal{M})$ is available, it is possible to perform inference on the latent parameters ϑ by exploiting the pseudo-marginal method [2], for example using a particle Metropolis-Hastings or particle Gibbs algorithm [6].

In this chapter we implement a pseudo-marginal Metropolis-Hastings algorithm for the two sequence SMC' algorithm and find an interesting feature of the model. The algorithm is then expanded to include the case of three sequences, and a solution is found to the problem of generating ancestral topologies inconsistent with the mutation data. In the last section we turn our attention towards smoothing via backwards sampling for the particle filter method developed. With this technique in hand, we exploit recent advances in efficient particle Gibbs that make use of samples from the smoothing distribution.

5.2 The two sequence case

Coalescent models [43, 52] have been at the centre of population genetics research for over thirty years, yet they are of limited use in inference over large regions of the genome. For this reason several approximations to the coalescent have been developed, possibly the most

well known of which is the sequentially Markov coalescent (SMC) algorithm introduced by McVean and Cardin [3]. In particular we direct our attention towards an iteration of the algorithm termed SMC' [4]; an algorithmic description of the model is given in Algorithm 5.1.

Algorithm 5.1 The SMC' algorithm

1. Generate a coalescent tree for $x = 0$. Denote the tree by $T(x)$ and its total branch length by $L(x)$.
2. Simulate a distance $\gamma \sim \text{Exp}(\frac{\rho}{2}L(x))$ along the sequence to the next recombination point. If $x + \gamma > 1$ **stop**.
3. Simulate a 'cut point' uniformly along the total branch length $L(x)$ of the tree $T(x)$.
4. Add a recombination event. The existing (left) branch above the cut point remains where it is. A line emerges (right) from the point and evolves according to the standard coalescent dynamics, i.e. proceeding backwards in time and coalescing with existing lineages at a rate proportional to the number of lineages present.
5. Delete the left branch above the cut point, leaving a tree instead of a graph.
6. Set $x = x + \gamma$. Denote the length of the new tree by $T(x)$ and its total branch length by $L(x)$.
7. Return to 2.

Output: a sample of an approximation to the full ARG.

Note that the original SMC algorithm differs from Algorithm (5.1) only in the reversal of steps 4 and 5, which precludes the possibility of the new branch coalescing back to the original branch it came from (since it has been deleted already). We now derive an expression for the probability density associated with the addition of a new recombination point under SMC' for two individuals. This is achieved through examination of pseudo-code describing the SMC' method, given in Algorithm 5.1.

Consider without loss of generality the first recombination point $\tau_0 = 0$, and suppose the

two branches of the associated tree coalesce at a time s_0 in the past. Denote by τ_1 and s_1 the position along the sequence of the first recombination point and the coalescence time of its branches. Notice that the dynamics of the algorithm imply the following structure for the transition:

$$p(\tau_1, s_1|\tau_0, s_0) = f(\tau_1|\tau_0, s_0)q(s_1|s_0).$$

According to Algorithm 5.1, the first recombination point is sampled according to an exponential distribution with rate $\rho L(\tau_0)/2$, therefore it is straightforward to write

$$f(\tau_1|\tau_0, s_0) = \mathbb{1}\{\tau_1 \geq \tau_0\}\frac{\rho L(\tau_0)}{2}\exp(-\frac{\rho L(\tau_0)}{2}(\tau_1 - \tau_0)).$$

In the case of two individuals, $L(\tau_0) = 2s_0$ and thus it follows that

$$f(\tau_1|\tau_0, s_0) = \mathbb{1}\{\tau_1 \geq \tau_0\}\rho s_0 \exp(-\rho s_0(\tau_1 - \tau_0)).$$

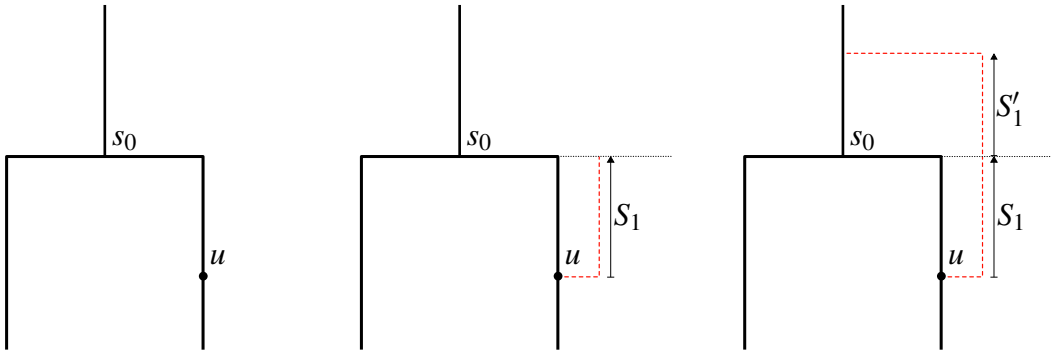
Establishing the transition density q of the coalescence time is more involved. Consider Figure 5.1, which describes events relevant to a transition $q(\cdot|s_0)$. First, as in Figure 5.1a, a cut point is picked uniformly along the total branch length. We may suppose this cut point is measured from the bottom of the left branch of Figure 5.1a. In fact, as will be demonstrated below, the particular branch chosen for the cut point makes no difference to the transition probabilities here, as they are symmetrical with respect to the branch choice. For this reason we refer hereafter only to the height of the cut point, denoted u and shown in Figure 5.1.

Next, an exponential random variable S_1 of rate 2 (the number of branches) is simulated backwards in time from the height u of the cut point, as in Figure 5.1b. We now consider the possible development of the recombination event conditional on the realisation of S_1.

Suppose first that the new branch created from the cut point during the recombination event does not exceed the current two-branch epoch. In other words, suppose $u \leq s_0$ and $0 \leq S_1 < s_0 - u$. Then there are two possible outcomes; with probability $1/2$ the new branch coalesces back to itself (the branch from which it came), else it coalesces with the other branch. If it coalesces with itself, the coalescence time is unchanged and therefore $s_1 = s_0$. If it coalesces with the other branch, the new coalescence time is $s_1 = u + S_1$ and we have $s_1 < s_0$.

Now suppose instead $S_1 > s_0 - u$, that is, the branch generated from the cut point exceeds the previous coalescence time s_0. In this case the coalescent dynamics dictate that a second exponential random variable, S_1', is drawn at rate 1, since only one lineage remains to which it may coalesce (see Figure 5.1c). The new coalescence time is $s_1 = s_0 + S_1'$ and $s_1 > s_0$.

Therefore three outcomes must be considered: $s_1 < s_0$, $s_1 = s_0$, and $s_1 > s_0$. Suppose first that $s_1 < s_0$. As described above, this outcome occurs when $S_1 < s_0 - u$ and the recombination



(a) $u \sim \mathrm{U}(0, 2s_0)$ **(b)** $S_1 \sim \mathrm{Exp}(2)$ **(c)** $S_1' \sim \mathrm{Exp}(1)$

Figure 5.1: *A schematic explanation of adding a recombination event in the SMC' algorithm for two individuals.*

branch does not coalesce with itself. The density of such a transition is given by

$$\begin{aligned} q_1(s_1|s_0) &= \int_0^{s_0} q(s_1|u,s_0)q(u|s_0)\,du \\ &= \frac{1}{s_0}\int_0^{s_0} q(s_1|u,s_0)\,du \\ &= \frac{1}{s_0}\int_0^{s_1} e^{-2(s_1-u)}\,du \\ &= \frac{1}{2s_0}(1-e^{-2s_1}). \end{aligned} \tag{5.2}$$

Notice that this calculation does not depend on the branch of the cut point. To see this, observe that sampling a uniform random variable on $[0, 2s_0]$ is equivalent to first picking the half-interval $[0, s_0]$ or $(s_0, 2s_0]$ each with probability 1/2, then sampling a uniform random variable on the chosen half-interval. In equations this is trivially conveyed:

$$\frac{1}{2s_0} = \frac{1}{2}\frac{1}{s_0}.$$

In our present situation, this means that sampling a cut point on $[0, 2s_0]$ is equivalent to picking a branch $B \in \{1, 2\}$ randomly and sampling a uniform random variable $u \sim U(0, s_0)$ to lie on that branch. We can write this as $q_1(u, B|s_0) = 1/2s_0$. Moreover, given that a cut point has been placed on branch B and at a height u, we know that the outcome $s_1 < s_0$ occurs with probability

$$q_1(s_1|u, B, s_0) = 2e^{-2(s_1-u)} \times \frac{1}{2},$$

where the factor of $1/2$ corresponds to choosing to coalesce with the 'other' branch. Together,

this implies $q_1(s_1, u, B|s_0) = \frac{1}{2s_0} e^{-2(s_1-u)}$ so that

$$\begin{aligned} q_1(s_1|s_0) &= \sum_{B \in [\![1,2]\!]} \int_0^{s_0} \mathbb{1}_{\{u \leq s_1\}} \frac{1}{2s_0} e^{-2(s_1-u)} \, du \\ &= \frac{1}{2s_0}(1 - e^{-2s_1}). \end{aligned}$$

In summary, Equation (5.2) gives the 'one-branch' equation. Marginalising over the branch choice is equivalent to multiplying by 1/2 for the choice of branch, and then by a factor of two since both branches were possible and would lead to equivalent outcomes.

Now suppose $s_1 = s_0$. This outcome occurs when $S_1 < s_0 - u$ and the recombination branch coalesces with itself. If the branch coalesces to itself, it does not matter precisely where S_1 is realised in the region $[u, s_0]$. For this reason the density must be marginalised with respect to S_1. In this case the transition is described by the probability

$$\begin{aligned} q_2(s_1 = s_0|s_0) &= \int_0^{s_0} \int_u^{s_0} q(s_1 = t|u, s_0) q(u|s_0) \, dt \, du \\ &= \frac{1}{s_0} \int_0^{s_0} \int_u^{s_0} q(s_1 = t|u, s_0) \, dt \, du \\ &= \frac{1}{s_0} \int_0^{s_0} \int_u^{s_0} e^{-2(t-u)} \, dt \, du \\ &= \frac{1}{2s_0} \left(s_0 - \frac{1}{2} \left(1 - e^{-2s_0} \right) \right) \end{aligned}$$

where again we have used the one-branch simplification. Finally suppose $s_1 > s_0$, corresponding to Figure 5.1c. The associated density is given by

$$\begin{aligned} q_3(s_1|s_0) &= \frac{1}{s_0} \int_0^{s_0} q(s_1|u, s_0) \, du \\ &= \frac{1}{s_0} \int_0^{s_0} e^{-2(s_0-u)} e^{-(s_1-s_0)} \, du \\ &= \frac{1}{2s_0} e^{-(s_1-s_0)} \left(1 - e^{-2s_0} \right). \end{aligned}$$

In summary the density, with respect to Lebesgue measure, for a transition corresponding to a

recombination event can be written

$$\begin{aligned} q(ds_1|s_0) = \mathbb{1}_{\{s_1<s_0\}}\frac{1}{2s_0}(1-e^{-2s_1})\,ds_1 \\ + \delta_{s_0}(s_1)\frac{1}{2s_0}\left(s_0 - \frac{1}{2}\left(1-e^{-2s_0}\right)\right) \\ + \mathbb{1}_{\{s_1>s_0\}}\frac{1}{2s_0}e^{-(s_1-s_0)}\left(1-e^{-2s_0}\right)\,ds_1. \end{aligned} \tag{5.3}$$

Notice that this indeed defines a probability density since

$$\begin{aligned} \int_0^\infty q(ds_1|s_0) &= \frac{1}{2s_0}\left\{\int_0^{s_0}(1-e^{-2v})\,dv + s_0 - \frac{1}{2}\left(1-e^{-2s_0}\right) + \left(1-e^{-2s_0}\right)\int_{s_0}^\infty e^{-(v-s_0)}\,dv\right\} \\ &= \frac{1}{2s_0}\left\{2s_0 - \left(1-e^{-2s_0}\right) + \left(1-e^{-2s_0}\right)\right\} \\ &= 1. \end{aligned}$$

Our method here involved considering every possible outcome of the recombination process. This is feasible in the current setting as there are only two sequences and the number of possible outcomes of a recombination event is small. It is possible to find the transition density $q(\cdot|s_0)$ of the coalescence time through another, more direct, route. Namely, it can be written in analogy with the algorithm that generates the model. Suppose we are currently at point x in the sequence and the associated coalescence time is s_0. Then, for two sequences, the pseudo-code given in Algorithm 5.2 describes the coalescence time transition brought about by a recombination event.

Algorithm 5.2 Two-sequence recombination event

Input: The current position x and coalescence time s_0 in the sequence.

1. Simulate a cut point along the total branch length $L(x)$ of the tree $T(x)$ and denote by u its height.

2. Simulate $S_1 \sim \text{Exp}(2)$.

   if $u + S_1 \leq s_0$

   Select uniformly at random one of the following two outcomes:

   Set $s_1 \leftarrow S_1 + u$, **go to** (3).

   Set $s_1 \leftarrow s_0$, **go to** (3).

   else

   Simulate $S_1' \sim \text{Exp}(1)$.

   Set $s_1 \leftarrow s_0 + S_1'$, **go to** (3).

3. Return s_1.

Output: the coalescence time s_1 after a recombination event.

Suppose the current tree is $T(0)$, with coalescence time s_0, so that the total branch length is given by $L(0) = 2s_0$. A direct reading of the pseudo-code implies the following expression, conditional on the height u of the cut point:

$$q(s_1|u,s_0) = \int_u^\infty 2e^{-2(t-u)} \Big\{ \mathbb{1}\{u \leq t \leq s_0\} \frac{1}{2} \left[\delta_{s_1}(t) \mathbb{1}\{u \leq s_1\} \mathbb{1}\{s_1 < s_0\} + \delta_{s_0}(s_1) \right] + \mathbb{1}\{t > s_0\} \mathbb{1}\{s_1 > s_0\} e^{-(s_1 - s_0)} \Big\} dt.$$

Performing the integration yields

$$\begin{aligned} q(s_1|u,s_0) &= \mathbb{1}\{s_1 < s_0\} \mathbb{1}\{u < s_1\} e^{-2(s_1-u)} \\ &\quad + \delta_{s_0}(s_1) \int_u^{s_0} e^{-2(t-u)}\, dt + \mathbb{1}\{s_1 > s_0\} e^{-(s_1-s_0)} \int_{s_0}^\infty 2e^{-2(t-u)}\, dt \\ &= \mathbb{1}\{s_1 < s_0\} \mathbb{1}\{u < s_1\} e^{-2(s_1-u)} \\ &\quad + \delta_{s_0}(s_1) \frac{1}{2} \left(1 - e^{-2(s_0-u)} \right) + \mathbb{1}\{s_1 > s_0\} e^{-(s_1-s_0)} e^{-2(s_0-u)}. \end{aligned}$$

After integrating out all possible cut points, this gives

$$\begin{aligned}\int_0^{s_0} q(s_1|u,s_0)q(du|s_0) &= \mathbb{1}\{s_1 < s_0\}\frac{1}{s_0}\int_0^{s_1} e^{-2(s_1-u)}\,du \\ &\quad + \delta_{s_0}(s_1)\frac{1}{2s_0}\int_0^{s_0}\left(1-e^{-2(s_0-u)}\right)du \\ &\quad + \mathbb{1}\{s_1 > s_0\}\frac{1}{s_0}e^{-(s_1-s_0)}\int_0^{s_0} e^{-2(s_0-u)}\,du \\ &= \mathbb{1}\{s_1 < s_0\}\frac{1}{2s_0}\left(1-e^{-2s_1}\right) \\ &\quad + \delta_{s_0}(s_1)\frac{1}{2s_0}\left(s_0 - \frac{1}{2}\left(1-e^{-2s_0}\right)\right) \\ &\quad + \mathbb{1}\{s_1 > s_0\}\frac{1}{2s_0}e^{-(s_1-s_0)}\left(1-e^{-2s_0}\right)\end{aligned}$$

which is identical to equation (5.3).

5.2.1 Verifying the transition density

Finally it is necessary to demonstrate agreement between samples drawn from the generative description of the recombination event given in Algorithm 5.2, and the derived transition density given in Eq. (5.3). Several methods are suited to this task. We first carry out a visual examination of the empirical sample quantiles against the quantiles of the cumulative distribution function (CDF) associated with the transition density. The approach is as follows:

1. Fix the current coalescence time s_0 to an arbitrary value in its support, i.e. $[0,\infty)$.
2. Conditioned on starting at s_0, simulate independently R recombination events giving new coalescence times $\{s_1^i\}_{i\in[\![1,R]\!]}$.
3. Compare using a Q-Q plot the empirical quantiles of the sample $\{s_1^i\}_{i\in[\![1,R]\!]}$ against the theoretical quantiles provided by the CDF associated with density (5.3).

The process should of course be repeated for several values of s_0, which may be sampled according to an $\text{Exp}(2)$ corresponding to the normal coalescent dynamics for two lineages. Figure 5.2 shows the outcome of six independent runs of this experiment, with the initial coalescence time s_0 shown in red. Note that one observes a discontinuity in the Q-Q plot as a consequence of the discontinuity (both left and right) of the transition density $q(s_1|s_0)$ at s_0.

The Q-Q plots appear to strongly support the correctness of the transition density, however this approach relies solely on a visual appraisal. We now replace step three in the above description with a one-sample Kolmogorov-Smirnov (KS) test to interrogate the hypothesis

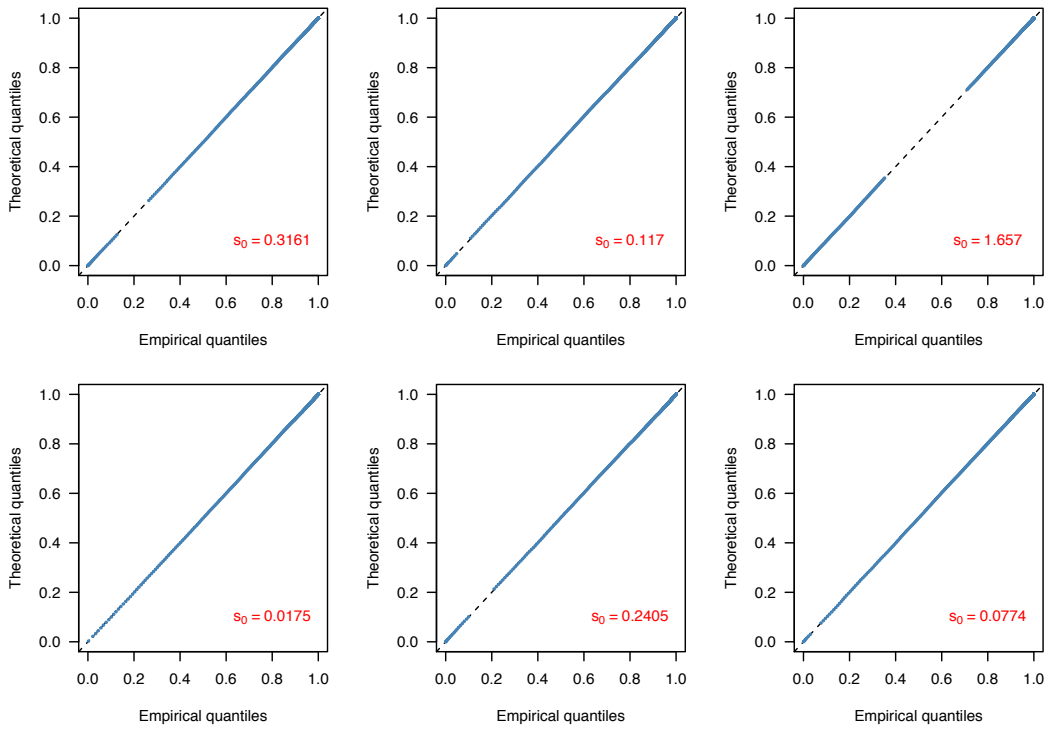
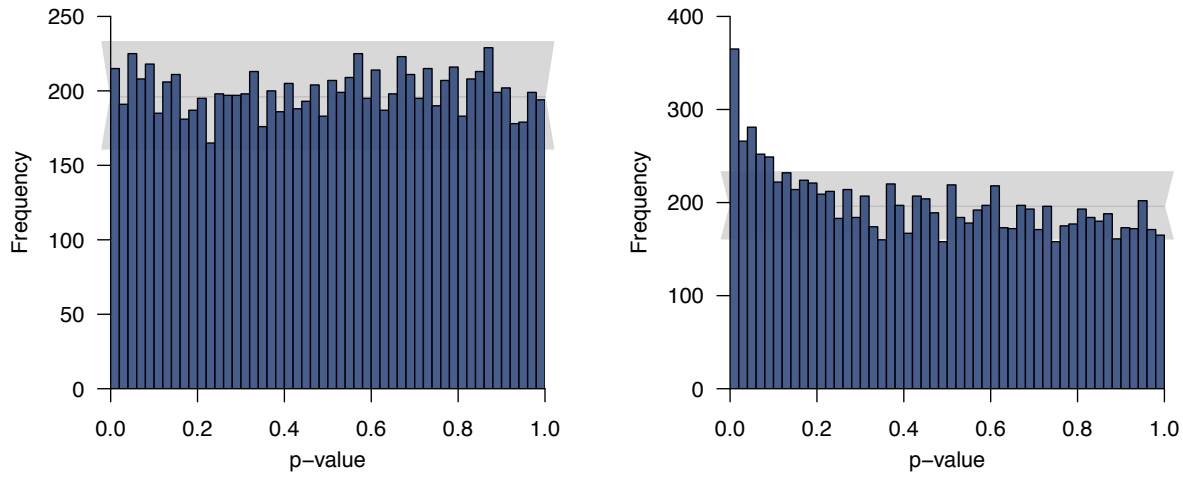


Figure 5.2: *Q-Q plots comparing new coalescence times* $\{s_1^i\}_{i\in[\![1,5\times10^4]\!]}$ *generated by Algorithm 5.2, conditional on* s_0 *shown in red, to the theoretical CDF derived above. The initial values* s_0 *were sampled from an* $\mathrm{Exp}(2)$.

that the empirical CDF of the samples is sufficiently close to the CDF of the transition density (5.3). Note that the traditional KS test may be used only with continuous CDFs. In this case a mixed version of the test is required, capable of handling multiple discontinuities, or jumps. Fortunately such an analogue exists [70] and is implemented in an R package [71].

The null hypothesis in a single experiment of this type is that the samples generated by the recombination algorithm, conditional on starting at s_0, are drawn from the density $q(s_1|s_0)$, as given in Equation (5.3). Naturally, the test must be carried out in a principled fashion for many initial values s_0; not rejecting the null hypothesis in an experiment conditioned on a single value of s_0 provides evidence of the correctness of the algorithm only for that value of s_0. We use here the following approach:

1. Sample the current coalescence time $s_0 \sim \mathrm{Exp}(2)$.
2. Conditional on starting at s_0, simulate independently R recombination events giving new coalescence times $\{s_1^i\}_{i\in[\![1,R]\!]}$.
3. Taking the empirical quantiles of the sample $\{s_1^i\}_{i\in[\![1,R]\!]}$, and the theoretical quantiles provided by the CDF, compute the p-value associated with the KS test statistic.



(a) *Correct CDF* **(b)** *Error in CDF*

Figure 5.3: *Each histogram shows* $10,000$ *replications of the p-value under the Kolmogorov-Smirnov test. For each replication,* $R = 100$ *samples from the generative process were used to construct the empirical CDF. The uniformity in plot (a) provides strong evidence for the null hypothesis; namely, that the generative samples (Algorithm 5.2) are being drawn from the same distribution as the CDF derived from Equation (5.3). Plot (b) shows the sensitivity of the analysis to a small error introduced in the CDF.*

Repeating this procedure, one can generate many p-values which under the null hypothesis are uniformly distributed on the interval $[0,1]$. The result of generating 10,000 p-values by this approach are shown in Figure 5.3, complete with 99% confidence interval for the uniform distribution on the histogram bins shown, calculated using the binomial distribution. The number of recombination events, R, simulated in step 2 must be large enough to ensure some level of smoothness in the empirical CDF. In Figure 5.3 it is shown that the uniform histogram test is sensitive to small errors for sufficiently large R. In particular, the rate of the exponential random variable was coded as 2.3 instead of the true value 2, resulting in a profound deviation from uniformity in the histogram. This lends weight to our belief in the technique as a useful diagnostic method. We note that in this case it did not require a particularly large number of samples R to detect a fairly subtle error, which is encouraging. There will of course be a lower bound to this. Repeating the experiment with the error present, we observed no deviation from uniformity when only $R = 20$ samples were used per replication. It follows that some errors may only be picked up by the method for large enough R, and it is reasonable to hypothesise that the smaller the error one is attempting the find, the larger the R required to find it. This follows too from the definition of the KS test statistic in combination with Donsker's theorem, which provides the rate at which the error decreases as the number of observations

increases[72]. A pragmatic approach then is to simply run the process for as large a value of R as is computationally feasible. We repeated the experiment in Figure 5.3 with $R = 1000$ and found as before no noticeable deviation from uniformity. This does not constitute a proof of correctness of the CDF with respect to the generative algorithm, only that if any errors persist they are so small as to be effectively negligible.

Finally we note, for completeness, that the sample of p-values itself (from the correct generative process) was not rejected in a KS test of uniformity at the 5% significance level. In conclusion, this series of investigations provided no evidence of any incorrectness in the probability density derived from the generative algorithm and so we proceed with some confidence in its correctness.

5.2.2 Summary

We have now an expression for the transition density $p(\tau_1, s_1 | \tau_0, s_0)$ describing a single recombination event in the SMC' model. It is useful to have this expression for two reasons. First, it would be essential to have an explicit expression if one intended to use importance sampling in the particle filter, for example as a variance reduction technique. This approach is not explored here for the two sequence case, as sampling from the prior distribution is found to be relatively efficient. However, performing the above calculations will also prove to be useful preparation for performing analogous calculations for the three sequence case, where it will be necessary to use techniques from importance sampling to overcome an interesting property of the coalescent models under consideration. In particular the idiosyncrasy concerns a characterisation of the mutations that are possible relative to a given ancestral history. Alternatively, to use a more statistical phrase, it will be necessary to consider what observations it is possible to observe given a particular latent state. For two sequences, or two individuals, when we refer to 'mutations' we are comparing two variations of the same section of genome. For example, if the two sequences began

ATACGTA...
AGACGCA...

then the first two mutation positions would be two and six with, for example, associated mutations $(1,0)$ and $(0,1)$. Here 0 represents the ancestral state and 1 the derived or mutated state. In this example, since the two sequences have in their second position (T, G) respectively, if the mutation associated with this position is $(1,0)$ then this means G is considered to be the ancestral state and T the derived state. Note that the scientist is required to know which state is ancestral and which derived; often this decision is made on the basis of biological evidence,

but there are also methods in population genetics that treat it as another unknown and attempt to infer it (see for example [56]).

It is crucial to observe that a mutation with respect to two sequences, for example $(1,0)$, does not rule out any particular ancestral history. In other words, given that a mutation m has been observed, all underlying topologies are still possible explanations for this mutation. Indeed in the two sequence case all topologies are essentially the same - two lines joined at a point - only differentiated by the height at which they meet, that is, the coalescence time. This property is exclusively enjoyed by the two sequence model. It is in general not true for an arbitrary number of sequences that any underlying topology can explain a given mutation. Therefore sampling directly from the prior for more than two sequences is extremely computationally expensive, if not impossible, as the vast majority of simulations will be incompatible with the observed mutation and so must be discarded. This idiosyncrasy will be a central concern when considering the three sequence case in Section 5.6, where an importance sampling solution is derived.

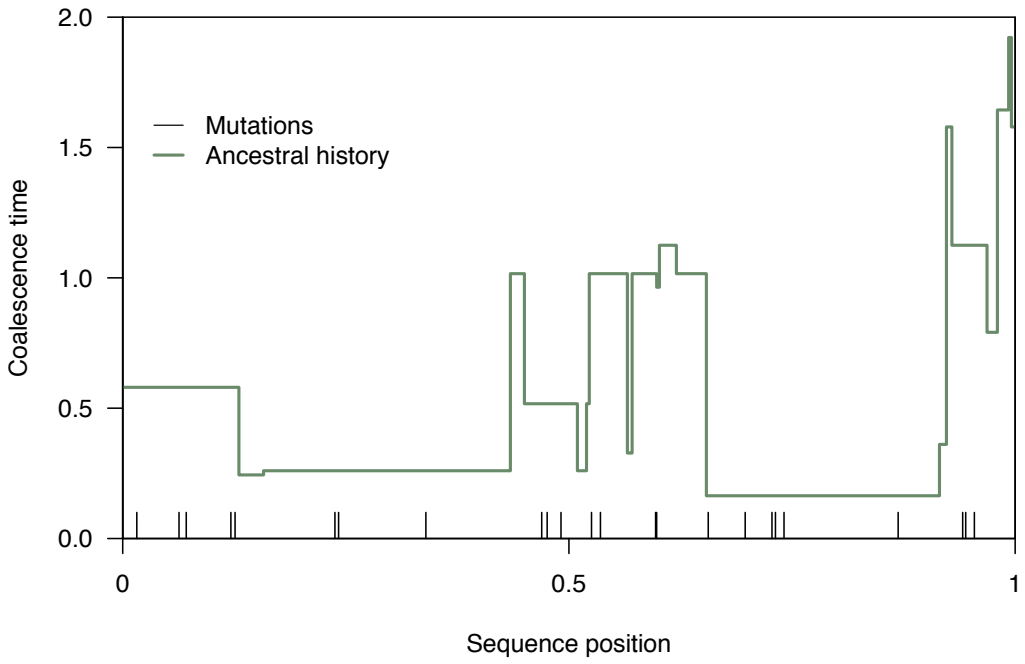


Figure 5.4: *Viewing the evolution of the SMC' algorithm as a piecewise deterministic Markov process. In green is the ancestral history associated with a sample; its jumps are the recombination points and the height of the line at each point is the coalescence time associated with that point the sequence. The short black lines convey the position of the mutations present in the sequences in the sample.*

Consider again now the Markov process $(\tau_i, s_i)_{i=0,1,\ldots}$ analysed in this section. Figure 5.4

shows a particular realisation of the recombination process for two individuals with mutations overlaid. The evolution of the state dynamics can be viewed as a piecewise deterministic process (PDP), i.e. a continuous-time stochastic process that jumps at a countable number of jump times, between which it is deterministic. Such processes were first proposed in [73]. What we consider here is in fact a piecewise constant process - the recombination points (τ_i) are the random times at which the stochastic jumps (s_i) occur, and between the recombination points the process is constant (and so deterministic). We now turn our attention more seriously to these piecewise deterministic Markov processes and the challenges of applying standard particle filter algorithms to them.

5.3 Piecewise deterministic processes

We now describe a formal framework for the analysis of the piecewise deterministic processes (PDPs) discussed above. The following introduction follows the notation and ideas of [69] and [68].

Consider a Markov chain $\left(\tau_j, \phi_j\right)_{j\in\mathbb{N}\cup\{0\}}$ comprising an increasing set of real *jump times* $\tau_j \in \mathbb{R}^+$, *jump sizes* ϕ_j taking values in a non-empty set Φ, and step-j transition kernel of the form

$$p(\mathrm{d}(\tau_j, \phi_j)|\tau_{j-1}, \phi_{j-1}) = f(\mathrm{d}\tau_j|\tau_{j-1}, \phi_{j-1})q(\mathrm{d}\phi_j|\phi_{j-1})$$

with support $(\tau_{j-1}, \infty) \times \Phi$. Note that it is possible [69] to work with more general forms of f and q, but the present forms are fit for our purpose. By convention, $\tau_0 = 0$ and $\phi_0 \sim q_0(\cdot)$. Let $\nu_t := \sup\{j \in \mathbb{N}\cup\{0\} : \tau_j \leq t\}$ denote the index of the last jump before time t, so that τ_{ν_t} is the time of the last jump before t. Use θ to denote the set of static parameters of the model. Then we have the following definition.

Definition 10. *A continuous-time stochastic process $\zeta := (\zeta_t)_{t\geq 0}$ with initial condition $\zeta_0 = \phi_0$ is said to be a* piecewise deterministic process (PDP) *when*

$$\zeta_t = F^\theta(t, \tau_{\nu_t}, \phi_{\nu_t})$$

for some deterministic function $F^\theta : [0,\infty)^2 \times \Phi \to \Phi$ satisfying $F^\theta(t,t,\phi) = \phi$ for any $t \geq 0, \phi \in \Phi$.

Let $0 = t_0 < t_1 < t_2 < \ldots$ be a deterministic series of times and denote $K_n := \nu_{t_n}$, with realisations k_n and convention $k_0 = 0$, the stochastic number of jumps made by the process $\left(\tau_j, \phi_j\right)$ before time t_n.

The PDP (ζ_t) for $t \in [0, t_n]$ is determined by the function F^θ and the collection of random variables $X_n := (K_n, \tau_{1:K_n}, \phi_{0:K_n})$ taking values in the disjoint union $\tilde{\mathsf{E}}_n := \bigcup_{k=0}^{\infty} \left(\{k\} \times \mathbb{T}_{n,k} \times \Phi^{k+1}\right)$, where $\mathbb{T}_{n,k} := \{\tau_{1:k} \in (0,\infty)^k : 0 < \tau_1 < \ldots < \tau_k \leq t_n\}$. A joint prior distribution is induced on the number of jumps k_n in $[0, t_n]$, their times $\tau_{1:k_n}$, and sizes $\phi_{0:k_n}$.

$$p_n(k_n, \tau_{1:k_n}, \phi_{0:k_n}) = S(t_n, \tau_{k_n}; \phi_{k_n}) q_0(\phi_0) \prod_{j=1}^{k_n} f(\tau_j | \tau_{j-1}, \phi_{j-1}) q(\phi_j | \phi_{j-1}) \mathbb{1}_{(0,t_n]}(\tau_{k_n}) \tag{5.4}$$

where $S(\tau, t; \phi) := 1 - \int_\tau^t f(ds|\tau, \phi)$ is the probability of no jump occurring in the interval $(\tau, t]$. The state process can be observed in the nth epoch $[t_{n-1}, t_n)$ only through noisy observations $y_{[t_{n-1},t_n)}$, a collection of random variables with density $g(y_{[t_{n-1},t_n)} | \zeta_{[t_{n-1},t_n)})$. Given the PDP, observations in non-overlapping time intervals are assumed to be conditionally independent.

Up to a constant of proportionality the posterior distribution of the process $(\zeta_t)_{t\in[0,t_n]}$, given the observations $y_{(0,t_n]}$, is of the form

$$\begin{aligned}\tilde{\pi}_n(x_n) &= \tilde{\gamma}(x_n)/\mathcal{Z}_n \\ &:= p_n(k_n, d\tau_{n,1:k_n}, d\phi_{n,0:k_n}) g(y_{(0,t_n]} | \zeta_{(0,t_n]})/\mathcal{Z}_n\end{aligned}$$

where $\mathcal{Z}_n > 0$ is a normalising constant and the density $g(y_{(0,t_n]} | \zeta_{(0,t_n]})$ is assumed to factorise as

$$g(y_{(0,t_n]} | \zeta_{(0,t_n]}) = g(y_{[\tau_{k_n},t_n]} | \tau_{k_n}, \phi_{k_n}) \prod_{j=1}^{k_n} g(y_{[\tau_{j-1},\tau_j)} | \tau_{j-1}, \phi_{j-1}).$$

Having described a generic PDP it is clear that this is precisely the recombination process discussed above, only with a slight change in notation. Here the state process is denoted ϕ rather than s, and we are considering in Equation (5.4) not the probability distribution of a single recombination event, but all recombination events occurring in the interval $[0, t_n]$. In our case the t here does not literally refer to a series of times, but of positions along the genetic sequence being considered. The observations $y_{(0,t_n]}$ correspond simply to the positions and types of mutations occurring on the sequence between position 0 and position t_n.

In summary, the SMC' algorithm (Algorithm 5.1) can be cast as a PDP. If one imagines overlaying deterministic epochs $0 = t_0 < t_1 < t_2 < \ldots < t_P$ on top of the sequence, then a prior distribution over the number of recombination points, their times, and their sizes is given by Equation (5.4). Having expressed the recombination process as a particular kind of state space model, we now seek to apply particle filter methods to it. A modest literature exists exploring the application of sequential Monte Carlo methods to PDPs, one of the first examples of which is the variable rate particle filter (VRPF) of Godsill et al. [74] (see also [68] for a review). Algorithm 5.3 is a pseudo-code description of the VRPF for SMC' (see e.g. [75] for a review

of particle filtering). An instance of the notation (i) implies that the operation should be performed for all particles $i \in [\![1,N]\!]$ where N is the total number of particles in the filter. We use a multinomial resampling scheme for all subsequent experiments, it is however not optimal and *stratified* or *systematic* resampling are to be favoured in general for their lower computational complexity[76].

Algorithm 5.3 VRPF for SMC'

Input: observations $y_{(0,t_P]}$, number of particles N, number of epochs T.

1. Sample $\phi_0^{(i)} \sim q_0(\cdot)$. Set $x_0^{(i)} \leftarrow (\tau_0 = 0, \phi_0^{(i)})$.
2. Set $w_0(x_0^{(i)}) \leftarrow 1$ and $W_0^i \propto w_0(x_0^{(i)})$.
3. For $n \in [\![1,T]\!]$:
   - Sample $\phi_n^{(i)} \sim q(\cdot|\bar{\phi}_{n-1}^{(i)})$, $\tau_n^{(i)} \sim f(\cdot|\bar{\tau}_{n-1}^{(i)}, \bar{\phi}_{n-1}^{(i)})$.
   - Set $x_n^{(i)} \leftarrow (\tau_n^{(i)}, \phi_n^{(i)})$ and $x_{0:n}^{(i)} \leftarrow (\bar{x}_{0:n-1}^{(i)}, x_n^{(i)})$.
   - Compute $w_n(x_{0:n}^{(i)}) = g(y_{(t_{n-1},t_n]}|x_{0:n}^{(i)})$ and $W_n^i \propto w_n(x_{0:n}^{(i)})$.
   - Resample according to weights W_n^i obtaining equally weighted particles $\{\frac{1}{N}, \bar{x}_{0:n}^{(i)}\}$.

Output: likelihood estimate $\hat{p}(y_{(0,t_P]}) = \prod_{n\in[\![1,P]\!]} \hat{p}(y_{(t_{n-1},t_n]}|y_{(0,t_{n-1}]})$ where $\hat{p}(y_{(t_{n-1},t_n]}|y_{(0,t_{n-1}]}) \leftarrow \frac{1}{N}\sum_{i\in[\![1,N]\!]} w_n(x_{0:n}^{(i)})$.

More advanced applications of particle filter methods to PDPs do exist, a notable example of which is the PDP particle filter introduced in [69]. An interesting feature of this approach is its flexibility, for example (using the transition kernels suggested in the paper) at each step of the particle filter a particle may make a new move, termed *birth*, but it may instead go back and change a previous move, which is referred to as an *adjustment*. This is a broad and powerful approach but comes at the cost of potentially adding considerable complexity. Whiteley [69, p.6] suggests there is particular inefficiency in using standard particle filters "when the expected jump arrival rate is low relative to the rate at which observations are made." It is doubtful that this is the situation we find here for the SMC' model. There are typically very few observations (mutations), and they arrive at a similar rate to the jumps (recombination events). Moreover, the mutations are only very weakly informative of the underlying ancestral topology. This is an issue in itself, and one central to the difficulty of coalescent inference, but it also implies that an ability to adjust particle trajectories based on

new observations may not be very beneficial in this setting.

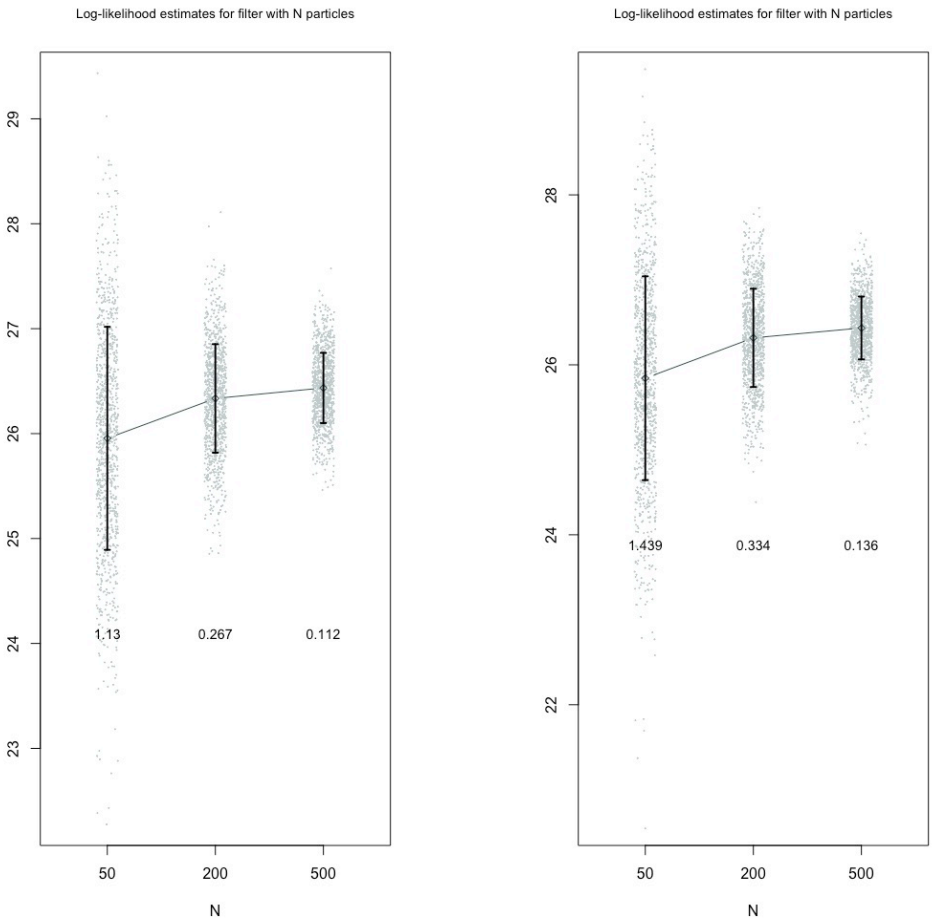
Another issue with the PDP filter is that it may produce an approximation. As described above, the "generic moves" used in [69] consist of an adjustment move and a birth move. In the birth move, the number of events is incremented, i.e. new jumps are created in the PDP. Unfortunately, if the number of jumps is by construction finite (as is the case with the toy example of 1 birth) then an approximation is induced. One is targeting a different distribution to the target, namely a distribution on the space of PDPs of no more than one jump event per epoch of the particle filter. Whiteley says of this property [69, p.13] that "a forward kernel capable of proposing any positive number of births in the interval $(t_{n-1}, t_n]$ must be employed if there is a positive probability associated with such configurations under the target distribution" and that feasibly a computationally cheap way to do this is to allow the forward kernel to propose $1 + D$ birth events per epoch where D follows a Poisson distribution of low intensity.

Ultimately, despite the features offered by the more advanced model, we will hereafter make use of the VRPF because of its simplicity. A key aim of the remainder of this work is to establish a backward smoothing method for PDPs which can be applied to the coalescent models under consideration. Sadly the nested construction of the PDP particle filter does not lend itself naturally to backwards sampling - though Finke et al. address this in [68]. However, backwards sampling can be written down relatively simply using the VRPF as the VRPF provides almost the standard setting for backward sampling on a SSM, albeit with a slight complication related to the likelihood term. This is explored further in Section 6.

5.4 Performance of VRPF for two sequences

We now analyse the performance of the VRPF for the SMC' (Algorithm 5.3). Figure 5.5 shows the convergence of log-likelihood estimates provided by the particle filter for increasing numbers of particles N. As N increases, the log-likelihood estimates are contained in the support of the log-likelihood estimates produced by the particle filter using a smaller N. Standard errors are drawn as black bars with the corresponding value written below in black. The standard deviation decreases like $1/N$, for example we notice that $1.13/4 \approx 0.27$ and that $0.27/2.5 \approx 0.11$. Moreover as N increases the estimates appear to converge on a mean value as they should (the true log-likelihood).

There is also a slight upward trend to be seen in both plots in Figure 5.5. This may seem unusual, after all the likelihood estimates produced by the particle filter are unbiased for any number of particles N and so should have the same mean for each input in the figure. Since we are considering log-likelihood estimate here, the unbiasedness property no longer holds.



(a) $T = 20, \rho = 80, \theta = 80$ **(b)** $T = 60, \rho = 80, \theta = 80$

Figure 5.5: *Convergence of the log-likelihood estimates for the two-individual SMC' particle filters.*

In fact, one would expect to see the kind of upward correction displayed here. See Appendix A.2 for some further details and experiments.

Figure 5.6 shows the likelihood curves for the SMC' model. Interestingly the likelihood curve for θ, the mutation rate, seems to be more informative than the curve for ρ, the recombination rate. Certainly the likelihood curve for ρ is much flatter. In addition the MLE is much closer to the true value for θ than ρ, and so in this sense too the likelihood is less informative about ρ. This quirk will appear again later in the chapter when discussing the use of this VRPF within particle MCMC.

Finally we consider the effect on the performance of the VRPF of increasing the number of epochs used. One way of measuring this is through the effective sample size. Here, taking the particle weights in each epoch to be given by the vector $w = (w_1, \ldots, w_N)$, and assuming

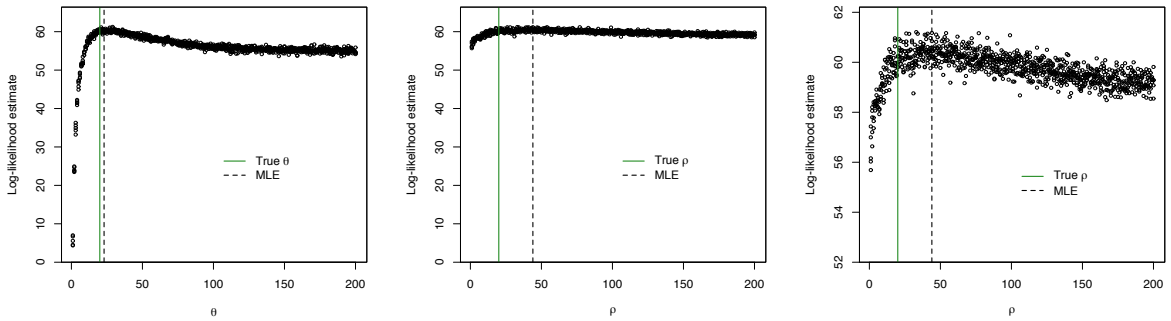


Figure 5.6: *Approximate likelihood curves of the SMC' model. First a synthetic set of data (mutations) for two sequences was produced using the SMC' model and the parameters* $(\theta, \rho) = (20, 20)$. *Then one parameter, say* ρ, *is fixed and a VRPF is run using this* ρ *and a range of values of* θ *- the log-likelihood estimate of the particle filter is recorded. From left to right the plots are: (a)* $\rho = 20, \theta \in (0, 200)$, *(b)* $\theta = 20, \rho \in (0, 200)$, *and plot (c) is identical to plot (b) but with a different y-axis scale. This is intended to show the similarity of the* θ *and* ρ *curves but also the relative flatness of the* ρ *likelihood.* $N = 128$ *particles were used in the particle filters.*

they are normalised, we define the ESS by a commonly used approximation (e.g. [77, 78]):

$$\mathrm{N}_{\mathrm{eff}}(w) = \frac{1}{\sum_{i=1}^{N} w_i^2}.$$

Otherwise, if the weight vector w is not normalised, we have the expression

$$\mathrm{N}_{\mathrm{eff}}(w) = \frac{(\sum_{i=1}^{N} w_i)^2}{\sum_{i=1}^{N} w_i^2}.$$

It is a commonly used measure of efficiency in sequential Monte Carlo. For example it is often used to determine whether or not to perform a resampling step [75] since a low ESS (close to zero) indicates a high level of degeneracy among the particle weights, whereas a high ESS (close to N) indicates that the particles are all of a similar weight. Clearly, as shown in Figure 5.7, increasing the number of epochs over which the particle filter works increases the average ESS. In practice it would be sensible to tune the algorithm by optimising with respect to the trade off between ESS and CPU time.

5.5 Comparison with ARGweaver

In the following experiments we compare on simulated data sets results obtained from the two-sequence particle filter just described and the popular ARGweaver software [67]. The data we use is simulated from the SMC' algorithm (Algorithm 5.1) with mutations overlaid

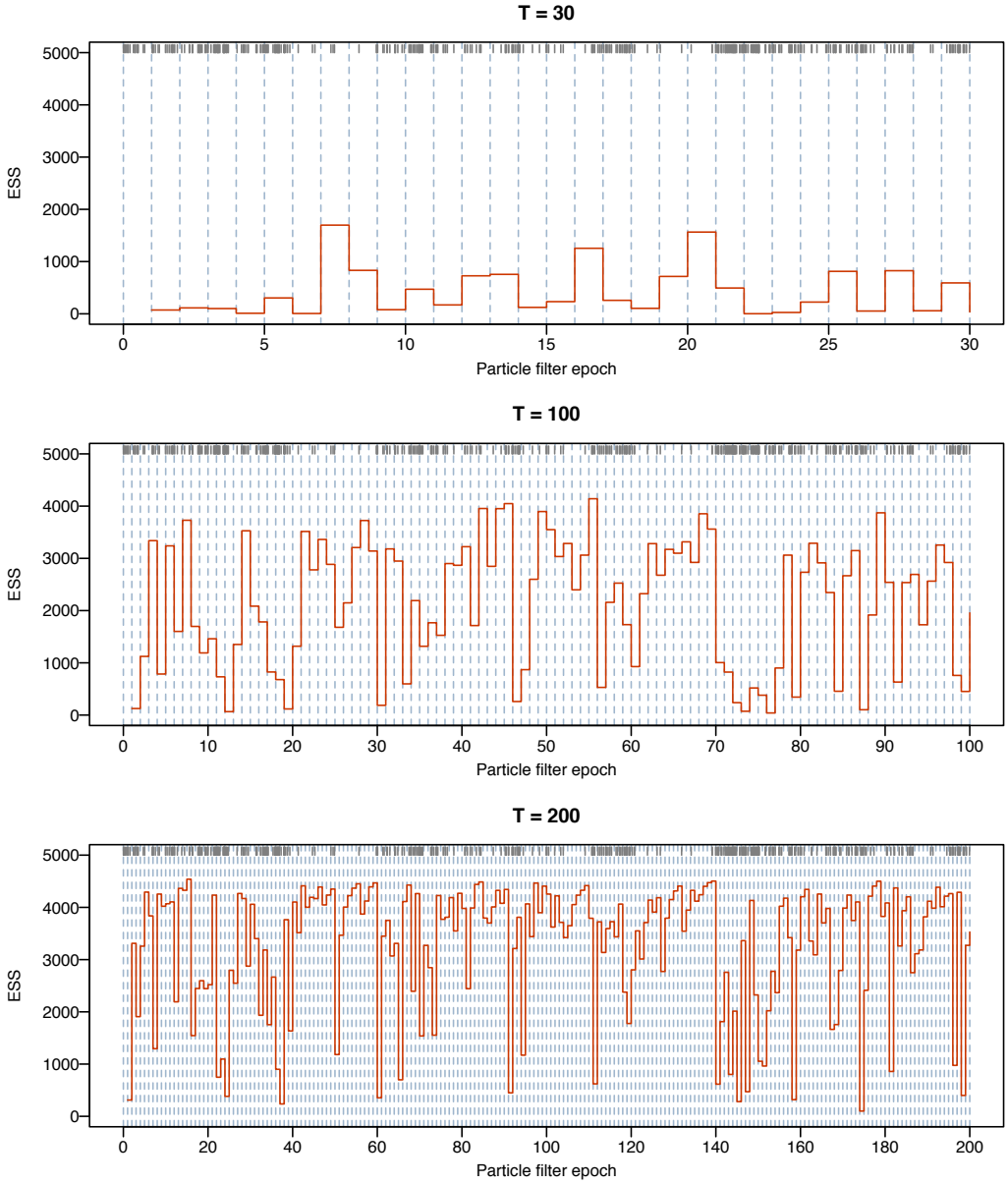


Figure 5.7: *The effective sample size (ESS) associated with three runs of the two-sequence algorithm. From top to bottom the plots show: (i) the VRPF with 30 epochs, (ii) the VRPF with 100 epochs and (iii) the VRPF with 200 epochs. The ESS is calculated from the weights at each epoch of the algorithm and is shown in red. The dotted blue lines represent the epochs of the particle filters and the grey dashes at the top of the plots represent the positions of the mutations in the sequence. The sequence itself is synthetic data simulated according to the SMC' model with* $\theta = \rho = 200$.

according to the usual dynamics. Our comparison follows an approach similar to that of the recent work [79] of Heine et al. in which a 'bridging' method is compared to ARGweaver by the number of recombinations in posterior samples. ARGweaver is in essence a Markov chain Monte Carlo method, capable of generating correlated posterior samples of ancestral recombination graphs (ARGs) given mutation data, as well as mutation and recombination rate parameters (θ, ρ). The distribution therefore of the ARGs it produces is $p(x_{1:P}|y_{1:P})$, where x represents the ARG across all sites and y represents the mutation data across the sequence. This is of course precisely the filtering distribution reached in the final state of the particle filter described earlier. Therefore by sampling from the latent process $x_{1:P}$ of ARGs using a particle filter, we have a valid means of comparing the two approaches. The only detail is that rather than comparing the full ARGs, we compare the number of recombination events, which is a projection of the full ARG. The comparison proceeds as follows:

1. Sample a mutation rate $\rho \sim \mathcal{U}(5,50)$. Set $\theta = \rho$.
2. Conditional on the parameters (ρ, θ), generate an ARG and mutation data $\mathcal{M}$ according to the SMC'.
3. Conditional on $\mathcal{M}$, run ARGweaver for 10^3 iterations to gain a sample of posterior ARGs.
4. Conditional on $\mathcal{M}$, run the two sequence VRPF independently 10^3 times to gain a sample of ARGs from the filtering distribution.
5. Summarise both sets of posterior ARGs by the number of recombinations and plot on the same histogram.

Note that ARGweaver is run with a fixed set of parameters (ρ, θ), as is the VRPF. We do not use for this purpose the 'true' parameters (ρ, θ) with which the data were generated. Rather, we use a normal perturbation to the true values. This is a realistic test in the setting of MCMC, in which often across the course of many iterations the particle filter will be run with parameter values fairly close to the 'true' values. In addition, the samples from ARGweaver are correlated as they originate from a Markov chain. In contrast, independent draws are being taken from the particle filter. This appears to have little consequence in practice as ARGweaver suffered from no appreciable autocorrelation in the examples shown here.

Figure 5.8 demonstrates the result of six realisations of this experiment. Encouragingly, the posterior distributions of the number of recombinations appear to be very similar for both approaches, and both are close to the true number of recombination events (shown as a dashed line). This consistency between the approaches is to be expected as they both rely on the

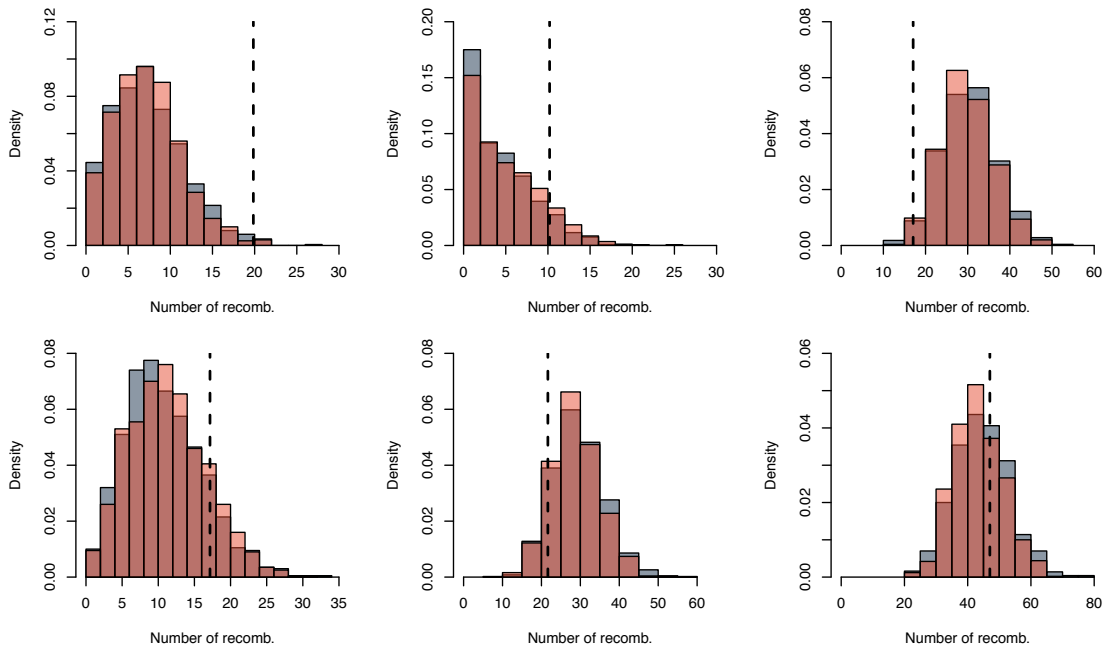


Figure 5.8: *Six realisations of the posterior number of recombinations, given distinct mutation data. ARGweaver is shown in red, while the two-sequence VRPF is shown in grey. $N = 128$ particles were used in the particle filter.*

sequentially Markov coalescent model, though ARGweaver implements a Jukes-Cantor model of mutation and not the infinite-sites model as we have implemented here and this may account for some variation.

One further comparison of the approaches is given in Figure 5.9. In this simulation, many experiments like the one described above were repeated. Each run, for example of ARGweaver, was summarised by the posterior mean of the number of recombination events. Similarly a posterior mean was taken for each set of 10^3 particle filter runs. These point estimates were then compared in mean squared error (MSE) to the true number of recombination events. This experiment was repeated in total 100 times. The plot shows that in general the two approaches perform very similarly, with the VRPF perhaps achieving a slightly lower median MSE. It is we believe more constructive to compare the posterior distributions qualitatively as in Figure 5.8, since there is no reason why the posterior distribution should concentrate on the true value. However, the MSE comparison is reassuring in confirming the particle filter method proposed here for SMC' has similar properties to ARGweaver by this particular recombination metric.

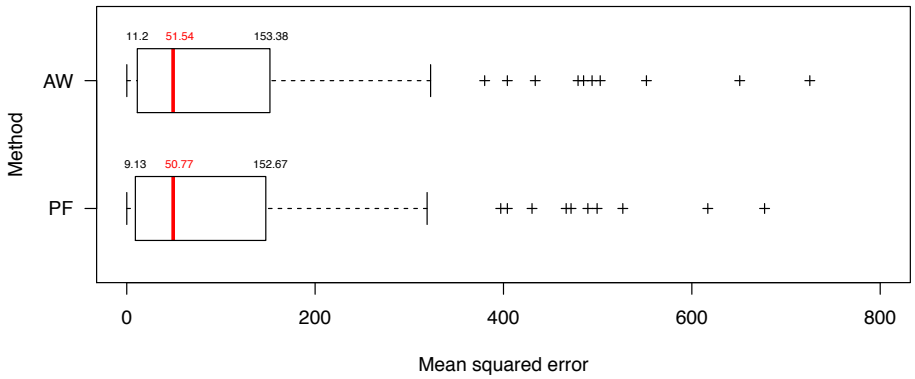


Figure 5.9: *The mean squared error associated with the posterior mean point-estimate of the number of recombination events. The two-sequence particle filter is denoted PF, and ARGweaver is denoted AW.*

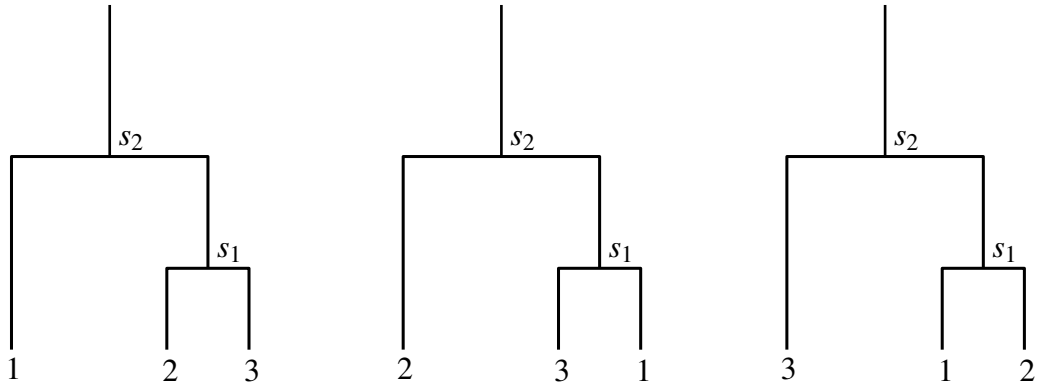
5.6 The three sequence case

We now consider the SMC' algorithm for three sequences. In the case of three individuals we are considering ancestral histories corresponding to topologies like those in Figure 5.10. Note that we have referred to the three topologies in the figure as the *essential* topologies since, although there are 3! ways of arranging the numbers 1 to 3, the symmetries of the model mean it is indifferent to certain permutations.

We now discuss the key differences between the two sequence and three sequence settings. First observe that, in contrast to the two sequence topology with its one-dimensional state $\phi = s$, a three sequence ancestral topology is described by a three-dimensional state $\phi = (s_1, s_2, \nu)$. Here s_1 denotes the coalescence time of the first two sequences to coalesce, and s_2 denotes the coalescence time of the remaining *lone* branch with the coalesced pair. Finally, ν represents the index of the lone branch itself. Secondly, we again consider what position-wise differences may be observed between three genetic sequences, using as an example the following sequences:

ATACGTA...
AGACGCA...
ATACGCA...

Here, the first two mutations would be at positions two and six since in these positions the three sequences do not all agree. Recall that a mutation can only be said to have occurred with respect to some ancestral sequence, that is to say, an agreed starting point from which



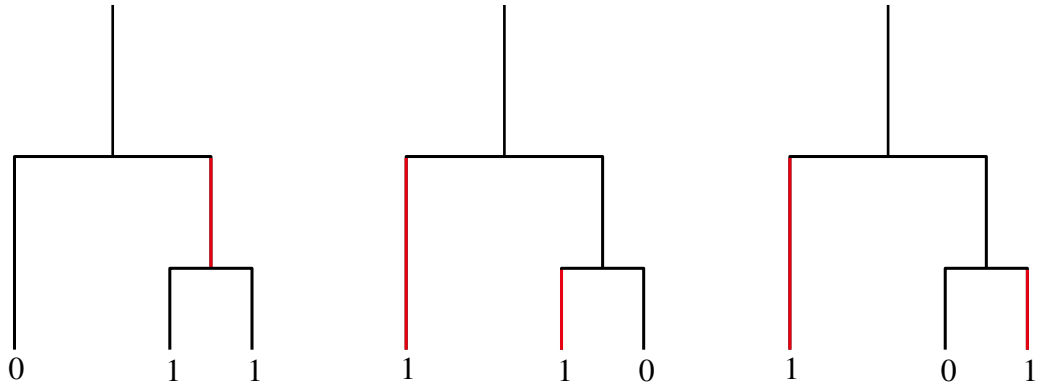
(a) *Type 1 topology* **(b)** *Type 2 topology* **(c)** *Type 3 topology*

Figure 5.10: *The essential topologies in the SMC' algorithm for three individuals. The numbers below the trees describe each sequence or individual, while the branches describe their ancestral relationship. For example, going from left to right, at the time of the first coalescence there are only three possibilities; (a) individuals 2 and 3 have coalesced, (b) individuals 1 and 3 have coalesced, or (c) individuals 1 and 2 have coalesced. Notice the tree itself can be drawn identically and we may simply move the labels to describe every possible outcome.*

the mutation was derived. If for this example we take the ancestral sequence to be the second (AGACGCA...), then the mutations associated with positions two and six in the sequence are $(1,0,1)$ and $(1,0,0)$ because in position two the first and third sequence differ from the ancestral sequence and in position six only the first differs from it.

Since $(0,0,0)$ and $(1,1,1)$ would not be mutations at all, there are $2^3-2=6$ possible mutation types. Three of these mutation types feature one mutation: $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. The other three feature two mutations: $(1,1,0)$, $(1,0,1)$ and $(0,1,1)$. The first class of mutation (only one individual mutated) is compatible with any underlying topology. To clarify, consider the mutation $(0,1,0)$ - this means a mutation has been observed only in individual two. Compare this with the possible ancestral histories the three individuals could have, shown as the numbers under the branches in Figure 5.10. Any of the ancestral histories is a possible explanation of this mutation; indeed it would only require the mutation to have happened on a different part of the tree.

The second class of mutation (two individuals mutated) are incompatible with some topologies. To see this, consider the mutation (0,1,1) and interrogate whether it is compatible with the leftmost topology in Figure 5.10. To be compatible the mutation must be the result of *only one* mutation, a requirement of the infinite sites model in genetics (the probability of a mutation occurring at the same site at different points in the ancestral history is vanishingly small). In this case it is compatible - it is explained by a mutation having happened on the section of branch above where individuals 2 and 3 coalesce. In contrast, the mutation $(1,1,0)$



(a) *Type 1 topology* **(b)** *Type 2 topology* **(c)** *Type 3 topology*

Figure 5.11: *The essential topologies in the SMC' algorithm for three individuals overlaid with a single mutation (0,1,1). Branches of the topology on which a mutation must have occurred (to explain the mutation) are shown in red. Notice that only in topology (a) is there exactly one red branch, whereas (b) and (c) have two.*

is incompatible with the leftmost topology in Figure 5.10, since the mutation would have to be placed on the tree twice for this outcome to occur.

5.6.1 The VRPF for three sequences

The key difference between the two and three sequence SMC' model, discussed above, presents an issue for the particle filter methodology we have developed. It is argued above that if a SNP is encountered at which exactly two of the three individuals have a mutation, the precise structure of this mutation will rule out some number of ancestral topologies for that SNP. We now take the example further. Consider again the compatibility of Figure 5.10a with respect to an observed mutation (0,1,1). It indeed offers a possible explanation, but notice that the other two essential topologies in Figure 5.10 do not.

To better understand this, in Figure 5.11 the mutation (0,1,1) has been transposed onto the topologies of Figure 5.10. Highlighted in red are the branches where a mutation would have had to occur for the topology to explain the observed mutation (0,1,1). Recall that it is a requirement of the infinite sites model in population genetics that the probability of a mutation occurring twice at the same position is taken to be vanishingly small. Only in topology (a) is it possible to achieve the observed mutation with just a single mutation on a branch. In other words, for topologies (b) and (c) the likelihood of observing the mutation (0,1,1) is zero, since it would require a violation of this assumption.

This is deleterious to the performance of a particle filter for the following reason. Suppose the particle filter proposes recombination points and topologies according to the model

(see Algorithm 5.1). Suppose further that the current interval of the particle filter contains a *problematic* mutation, i.e. of the type described above - a mutation shared by exactly two of the three individuals. Then, if the particle being considered reaches the problematic mutation whilst its current state ϕ is one of the two non-compatible topologies, the likelihood term in the weight of this particle will vanish. Of course there is in principle nothing preventing the particle weights in a particle filter from reaching zero, but it causes at least two difficulties for the approach. First it will very rapidly lead to so-called sample impoverishment, a common issue in particle filtering whereby only very few particles enjoy significant weight. This in turn will lead to the filter producing log-likelihood estimates of high variance. A second, perhaps more pressing, consequence of particle weights dropping to zero with high probability is that, in a given epoch of the particle filter, *all* of the particles may die, in which case the algorithm fails.

In practice this is a serious obstacle. Applying a classical particle filter with resampling to the three sequence model, our experiments found that on average 80% of particles die in an epoch containing a problematic mutation. And since the number that die is itself of course a random variable, while it may be as little as 60% in an early epoch, it may later in the filter happen to be 100%, at which point the algorithm collapses and the computational effort is wasted.

There are at least two possible solutions to this difficulty. First we could discard the infinite sites model for another model of mutation. The finite sites model, for instance, allows 'recurrent mutations' - more than one mutation happening in the same position [39]. This model is adopted in, for example, [38, §5.2], but we do not pursue the approach here. Instead we aim to develop now an algorithm that proposes only topologies that it knows to be consistent with future mutations. For this reason we now repeat the calculations made for the transition probabilities of the two individual case for three individuals (requiring a normalising constant too). With this transition density in hand, we will explore an importance sampling method through which we may control the probability of proposing a topology that is inconsistent with a future mutation.

5.7 A different approach

We now describe a novel approach to the particle filtering for the SMC' model, designed to work around the pitfalls of the VRPF in this setting. The new algorithm has two key features that differentiate it from the VRPF. First it facilitates the proposal of particles suited to the observed mutations, in other words particles guaranteed to give a positive weight. Secondly it allows the user to control the probability of a particle undergoing no events in an epoch. Both

of these offer an improvement in stability of the algorithm and the variance associated with the likelihood estimates.

An importance sampling approach will be taken, requiring an expression for the transition densities associated with the three sequence SMC' model. We begin, as for the two sequence case of Section 5.2, by constructing a pseudo-code description of the SMC' algorithm.

5.7.1 Pseudo-code for the three sequence case

Analogously to the two sequence case, the SMC' dynamics (see Algorithm 5.1) imply a transition density for the jth recombination event of the form

$$p(\tau_j, \phi_j | \tau_{j-1}, \phi_{j-1}) = f(\tau_j | \tau_{j-1}, \phi_{j-1}) q(\phi_j | \phi_{j-1}).$$

Pseudo-code is now developed that describes algorithmically the evolution of the state process in a recombination event in the three-sequence case, i.e. the dynamics of $q(\phi_j|\phi_{j-1})$. This will facilitate in the sequel writing down the transition density. Later, in Section 5.7.5, we return to the function f and the evolution of the jump times τ_j.

Earlier in the chapter we demonstrated that for a three sequence ancestral history we need to track the three-dimensional state $\phi = (s_1, s_2, \nu)$, corresponding respectively to the time of the first coalescence, the second coalescence, and the index of the *lone* branch. For a visual example, suppose for some $k \in \mathbb{N}$ the current tree is $T(\phi_k)$, as depicted in Figure 5.12, with coalescence times (s_k^1, s_k^2) so that the total branch length is given by $L(\phi_k) = 2s_k^2 + s_k^1$. Denote the index of the *lone* branch, i.e. the branch that does not coalesce first, by $\nu \in [\![1,3]\!]$, or specifically ν_k for this iteration.

In a recombination event for three sequences, recalling the instructions of Algorithm 5.1, a cut point must be simulated somewhere on the current topology. Let the height of the cut point be given by u, while the index of the branch on which it occurs will be given by B. Unlike in the two sequence setting, u and B here are not independent, and have a non-trivial joint distribution. For conceptual clarity we introduce the random variable A, taking values in the set $\{a, b\}$ which stands for *above* and *below*. The event $A = a$ occurs when the cut point lies somewhere in the top section of the topology, that is in the two-branch epoch. On the other hand, for a cut point in the bottom section of the topology, that is in the three-branch epoch, we have $A = b$. Now, Algorithm 5.1 requires the cut point to be drawn uniformly along the total branch length $L(\phi_k)$ of the current tree $T(\phi_k)$. We can write

$$q(A = a | s_k^1, s_k^2) = \frac{2\bar{s}_k^2}{3s_k^1 + 2\bar{s}_k^2} = \frac{2(s_k^2 - s_k^1)}{2s_k^2 + s_k^1}$$

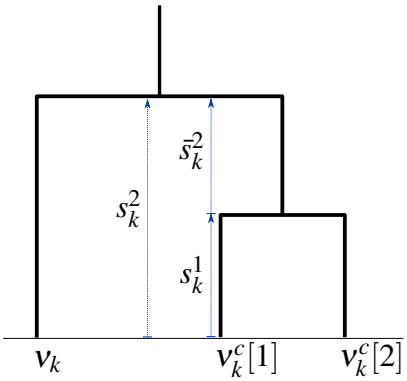


Figure 5.12: *A three-sequence topology $T(\phi_k) = T(s_k^1, s_k^2, \nu_k)$ demonstrating the height and branch nomenclature.*

since this is just the total branch length of the *above* section as a proportion of the total branch length $L(\phi_k)$. Similarly we can write

$$q(A = b|s_k^1, s_k^2) = \frac{3s_k^1}{3s_k^1 + 2\bar{s}_k^2} = \frac{3s_k^1}{2s_k^2 + s_k^1}.$$

Notice that when $A = a$ we must have by construction $B \in \{1,2\}, u \in [s_k^1, s_k^2]$. Conversely when $A = b$ we must have $B \in \{1,2,3\}, u \in [0, s_k^1]$. Furthermore, u and B are conditionally independent given A. This implies for example

$$q(u, B|A = a, s_k^1, s_k^2) = \frac{\mathbb{1}\{B \in [\![1,2]\!]\}}{2} \frac{\mathbb{1}\{s_k^1 \leq u \leq s_k^2\}}{s_k^2 - s_k^1}.$$

Thus the probability of generating an initial cut point on the topology on branch B and at a height u is given by

$$\begin{aligned} q(u, B|s_k^1, s_k^2) &= q(A = a, u, B|s_k^1, s_k^2) + q(A = b, u, B|s_k^1, s_k^2) \\ &= q(A = a|s_k^1, s_k^2) q(u, B|A = a, s_k^1, s_k^2) + q(A = b|s_k^1, s_k^2) q(u, B|A = b, s_k^1, s_k^2) \\ &= \frac{2(s_k^2 - s_k^1)}{2s_k^2 + s_k^1} \frac{\mathbb{1}\{B \in [\![1,2]\!]\}}{2} \frac{\mathbb{1}\{s_k^1 \leq u \leq s_k^2\}}{s_k^2 - s_k^1} \\ &\quad + \frac{3s_k^1}{2s_k^2 + s_k^1} \frac{\mathbb{1}\{B \in [\![1,3]\!]\}}{3} \frac{\mathbb{1}\{u \leq s_k^1\}}{s_k^1} \end{aligned}$$

and we have $\int_0^{s_k^2} \sum_{B \in [\![1,3]\!]} q(u, B|s_k^1, s_k^2) = 1$ as required. Note that hereafter we will sometimes abuse notation by writing v_k when what is really meant is the singleton $\{v_k\}$. For example, in Figure 5.12 the index of the lone branch is (in this case) $v_k = 1$ and we sometimes use

$\nu_k^c := \{2,3\}$ to ease notation as well as $\nu_k^c := \{\nu_k^c[1], \nu_k^c[2]\}$. We will also make use of the notation $\nu_{k+1} = \nu_k^c \backslash B$ as a shorthand for choosing $\nu_{k+1} \in \{\nu_k\}^c \cap \{B\}^c$, where all complements are taken with respect to the universal set $[\![1,3]\!] := \{1,2,3\}$. There is no confusion here as $\nu_k^c \backslash B$ will only be used when $B \in \nu_k^c$ and so we will always have $|\{\nu_k\}^c \cap \{B\}^c| = 1$.

A single recombination event for the three sequence SMC' model is now given in pseudo-code in Algorithm 5.4, starting from a present state $\phi_0 = (s_0^1, s_0^2, \nu_0)$. It is the three sequence analogue to Algorithm 5.2. The pseudo-code is written for step $k = 0$ simply for visual clarity; really it describes a general move $\phi_k \to \phi_{k+1}$, that is, it describes the procedure for sampling from the distribution $q(\phi_{k+1}|\phi_k)$.

Also given here are pseudo-code descriptions of two alternative versions of this algorithm. The first (Algorithm 5.5) describes a recombination event under the three sequence SMC' model where the outcome is conditioned on $\nu_{k+1} = \nu_k$. In other words it describes the procedure for sampling from the distribution $q(s_{k+1}^1, s_{k+1}^2 | \phi_k, \nu_{k+1} = \nu_k)$. The second (Algorithm 5.6) describes a recombination event under the three sequence SMC' model where the outcome is conditioned on $\nu_{k+1} = \nu$ where ν is given and $\nu \neq \nu_k$. In other words it describes sampling from the distribution $q(s_{k+1}^1, s_{k+1}^2 | \phi_k, \nu_{k+1} = \nu)$ where $\nu \neq \nu_k$. These two additional algorithms can be thought of as restrictions on Algorithm 5.4 in the sense that they observe the same dynamics but their output $(s_{k+1}^1, s_{k+1}^2, \nu_{k+1})$ is restricted to a subset of the possible outputs of that algorithm.

5.7.2 Sampling the next state

Recall that throughout this section we are concerned with finding the transition density

$$q(\phi_1|\phi_0) = q(s_1^1, s_1^2, \nu_1 | s_0^1, s_0^2, \nu_0)$$

associated with the state transition in the three-sequence SMC' model. That is, we aim to find the probability density of the next topology (after a recombination event) given the current topology. An explicit pseudo-code description of the model, as was developed above, allows us in some sense to simply write down the transition density directly. We will do this by first conditioning on the particular cut point generated in the algorithm and then marginalising over all possible cut points.

There are in fact three categories of cut point. We start with the case $\mathbb{1}\{u \leq s_0^1\}\mathbb{1}\{B \neq \nu_0\}$, followed by $\mathbb{1}\{u \leq s_0^1\}\mathbb{1}\{B = \nu_0\}$, then finally $\mathbb{1}\{u > s_0^1\}$. To be explicit, we first consider the transition density when the cut point is on the lower three branches, and not on the lone branch. Then, we consider the density when the cut point is on the lone branch, in the lower section. Finally we consider the transition density associated with a cut point on the upper branches.

Algorithm 5.4 Three sequence recombination event

Input: a current state $\phi_0 = (s_0^1, s_0^2, \nu_0)$.

1. Simulate uniformly a cut point (u, B) along the total branch length of the tree $T(\phi_0)$.

2. **if** $u \leq s_0^1$
   Simulate $H_1 \sim \text{Exp}(3)$.
   if $u + H_1 \leq s_0^1$
   if $B = \nu_0$
   Select uniformly at random one of the following three outcomes:
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0$, **go to** (3).
   Set $s_1^1 \leftarrow u + H_1, s_1^2 \leftarrow s_0^1, \nu_1 \leftarrow \nu_0^c[1]$, **go to** (3).
   Set $s_1^1 \leftarrow u + H_1, s_1^2 \leftarrow s_0^1, \nu_1 \leftarrow \nu_0^c[2]$, **go to** (3).
   else
   Select uniformly at random one of the following three outcomes:
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0$, **go to** (3).
   Set $s_1^1 \leftarrow u + H_1, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0$, **go to** (3).
   Set $s_1^1 \leftarrow u + H_1, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0^c \backslash B$, **go to** (3).
   else
   Simulate $H_2 \sim \text{Exp}(2)$.
   if $s_0^1 + H_2 \leq s_0^2$
   if $B = \nu_0$
   Select uniformly at random one of the following two outcomes:
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0$, **go to** (3).
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^1 + H_2, \nu_1 \leftarrow \nu_0$, **go to** (3).
   else
   Select uniformly at random one of the following two outcomes:
   Set $s_1^1 \leftarrow s_0^1 + H_2, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0$, **go to** (3).
   Set $s_1^1 \leftarrow s_0^1 + H_2, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0^c \backslash B$, **go to** (3).
   else
   Simulate $H_3 \sim \text{Exp}(1)$.
   if $B = \nu_0$, set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2 + H_3, \nu_1 \leftarrow \nu_0$, **go to** (3).
   else set $s_1^1 \leftarrow s_0^2, s_1^2 \leftarrow s_0^2 + H_3, \nu_1 \leftarrow B$, **go to** (3).

   else
   Simulate $H_1 \sim \text{Exp}(2)$.
   if $u + H_1 \leq s_0^2$
   Select uniformly at random one of the following two outcomes:
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, \nu_1 \leftarrow \nu_0$, **go to** (3).
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow u + H_1, \nu_1 \leftarrow \nu_0$, **go to** (3).
   else
   Simulate $H_2 \sim \text{Exp}(1)$ and set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2 + H_2, \nu_1 \leftarrow \nu_0$, **go to** (3).

3. Return s_1^1, s_1^2, ν_1.

Output: the next state $\phi_1 = (s_1^1, s_1^2, \nu_1)$.

Algorithm 5.5 Three sequence recombination event, conditional on $v_1 = v_0$.

Input: a current state $\phi_0 = (s_0^1, s_0^2, v_0)$.

1. Simulate uniformly a cut point (u, B) along the total branch length of the tree $T(\phi_0)$.

2. **if** $u \leq s_0^1$
   Simulate $H_1 \sim \text{Exp}(3)$.
   if $u + H_1 \leq s_0^1$
   if $B = v_0$
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v_0$, **go to** (3).
   else
   Select uniformly at random one of the following two outcomes:
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v_0$, **go to** (3).
   Set $s_1^1 \leftarrow u + H_1, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v_0$, **go to** (3).
   else
   if $B = v_0$
   Simulate $H_2 \sim \text{Exp}(2)$.
   if $s_0^1 + H_2 \leq s_0^2$
   Select uniformly at random one of the following two outcomes:
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v_0$, **go to** (3).
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^1 + H_2, v_1 \leftarrow v_0$, **go to** (3).
   else
   Simulate $H_3 \sim \text{Exp}(1)$. Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2 + H_3, v_1 \leftarrow v_0$, **go to** (3).
   else
   Simulate $H_2 \sim \text{Exp}(2)$ such that $s_0^1 + H_2 \leq s_0^2$.
   Set $s_1^1 \leftarrow s_0^1 + H_2, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v_0$, **go to** (3).

   else
   Simulate $H_1 \sim \text{Exp}(2)$.
   if $u + H_1 \leq s_0^2$
   Select uniformly at random one of the following two outcomes:
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v_0$, **go to** (3).
   Set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow u + H_1, v_1 \leftarrow v_0$, **go to** (3).
   else
   Simulate $H_2 \sim \text{Exp}(1)$ and set $s_1^1 \leftarrow s_0^1, s_1^2 \leftarrow s_0^2 + H_2, v_1 \leftarrow v_0$, **go to** (3).

3. Return s_1^1, s_1^2, v_1.

Output: the next state $\phi_1 = (s_1^1, s_1^2, v_1)$.

Algorithm 5.6 Three sequence recombination event, conditional on $v_1 = v$ where $v \in [\![1,3]\!] \setminus \{v_0\}$

Input: a current state $\phi_0 = (s_0^1, s_0^2, \nu_0)$.

1. Simulate uniformly a cut point (u, B) along the total branch length of the tree $T(\phi_0)$ such that $u \leq s_0^1$.

2. **if** $B = v_0$
   Simulate $H_1 \sim \text{Exp}(3)$ such that $u + H_1 \leq s_0^1$.
   Set $s_1^1 \leftarrow u + H_1, s_1^2 \leftarrow s_0^1, v_1 \leftarrow v$, **go to** (3).
   else if $B = v$
   Simulate $H_3 \sim \text{Exp}(1)$.
   Set $s_1^1 \leftarrow s_0^2, s_1^2 \leftarrow s_0^2 + H_3, v_1 \leftarrow v$, **go to** (3).
   else
   Simulate $H_1 \sim \text{Exp}(3)$.
   if $u + H_1 \leq s_0^1$
   Set $s_1^1 \leftarrow u + H_1, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v$, **go to** (3).
   else
   Simulate $H_2 \sim \text{Exp}(2)$ such that $s_0^1 + H_2 \leq s_0^2$.
   Set $s_1^1 \leftarrow s_0^1 + H_2, s_1^2 \leftarrow s_0^2, v_1 \leftarrow v$, **go to** (3).

3. Return s_1^1, s_1^2, v_1.

Output: the next state $\phi_1 = (s_1^1, s_1^2, \nu_1)$.

As there are only two branches in the upper section (by definition) this last calculation will be analogous to the two-sequence transition density calculations of Section 5.2.

Case (1) $\mathbb{1}\{u \leq s_0^1\}\mathbb{1}\{B \neq v_0\}$. In words, we now calculate the transition density for a recombination event starting at a cut point on the lower three branches, and not on the lone branch. We have

$$q(\phi_1|A=b,u,B\neq v_0,\phi_0) = q(s_1^1,s_1^2,v_1|A=b,u,B\neq v_0,s_0^1,s_0^2,v_0)$$

is given by

$$\begin{aligned}
&\int_u^\infty 3e^{-3(t-u)}\Bigg\{ \\
&\mathbb{1}\{u\leq t\leq s_0^1\}\frac{1}{3}\delta_{s_0^2}(s_1^2)\left[\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)+\delta_{s_1^1}(t)\mathbb{1}\{s_1^1<s_0^1\}\mathbb{1}\{u\leq s_1^1\}\left(\delta_{v_0}(v_1)+\mathbb{1}\{v_1=v_0^c\backslash B\}\right)\right] \\
&+\mathbb{1}\{s_0^1<t\}\Bigg\{\int_{s_0^1}^\infty 2e^{-2(t'-s_0^1)}\Big[\mathbb{1}\{t'\leq s_0^2\}\mathbb{1}\{s_0^1<s_1^1<s_0^2\}\delta_{s_0^2}(s_1^2)\delta_{s_1^1}(t')\frac{1}{2}\left(\delta_{v_0}(v_1)+\mathbb{1}\{v_1=v_0^c\backslash B\}\right) \\
&\qquad\qquad +\mathbb{1}\{t'>s_0^2\}\mathbb{1}\{s_1^2>s_0^2\}\delta_{s_0^2}(s_1^1)\mathbb{1}\{v_1=B\}e^{-(s_1^2-s_0^2)}\Big]\,dt'\Bigg\} \\
&\qquad\Bigg\}\,dt
\end{aligned}$$

Performing the integrals here gives

$$\begin{aligned}
&=\frac{1}{3}(1-e^{-3(s_0^1-u)})\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2) \\
&\quad+e^{-3(s_1^1-u)}\mathbb{1}\{s_1^1<s_0^1\}\mathbb{1}\{u\leq s_1^1\}\delta_{s_0^2}(s_1^2)\left(\delta_{v_0}(v_1)+\mathbb{1}\{v_1=v_0^c\backslash B\}\right) \\
&\qquad+e^{-3(s_0^1-u)}\Big[\mathbb{1}\{s_0^1<s_1^1<s_0^2\}\delta_{s_0^2}(s_1^2)\left(\delta_{v_0}(v_1)+\mathbb{1}\{v_1=v_0^c\backslash B\}\right)e^{-2(s_1^1-s_0^1)} \\
&\qquad\quad+\mathbb{1}\{s_1^2>s_0^2\}\delta_{s_0^2}(s_1^1)\mathbb{1}\{v_1=B\}e^{-(s_1^2-s_0^2)}e^{-2(s_0^2-s_0^1)}\Big]
\end{aligned}$$

Then, integrating over $u\in[0,s_0^1]$ gives

$$\begin{aligned}
&=\frac{1}{3}\left(s_0^1-\frac{1}{3}(1-e^{-3s_0^1})\right)\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2) \\
&\quad+\frac{1}{3}(1-e^{-3s_1^1})\mathbb{1}\{s_1^1<s_0^1\}\delta_{s_0^2}(s_1^2)\left(\delta_{v_0}(v_1)+\mathbb{1}\{v_1=v_0^c\backslash B\}\right) \\
&\qquad+\frac{1}{3}(1-e^{-3s_0^1})\Big[\mathbb{1}\{s_0^1<s_1^1<s_0^2\}\delta_{s_0^2}(s_1^2)\left(\delta_{v_0}(v_1)+\mathbb{1}\{v_1=v_0^c\backslash B\}\right)e^{-2(s_1^1-s_0^1)} \\
&\qquad\quad+\mathbb{1}\{s_1^2>s_0^2\}\delta_{s_0^2}(s_1^1)\mathbb{1}\{v_1=B\}e^{-(s_1^2-s_0^2)}e^{-2(s_0^2-s_0^1)}\Big]
\end{aligned}$$

Case (2) $\mathbb{1}\{u \leq s_0^1\}\mathbb{1}\{B = v_0\}$. In words, we calculate here the transition density for a recombination event starting at a cut point on the lower three branches, and in particular on the lone branch. We have that $q(\phi_1|A = b, u, B = v_0, \phi_0)$ is given by

$$\int_u^\infty 3e^{-3(t-u)} \Bigg\{$$
$$\mathbb{1}\{u \leq t \leq s_0^1\}\frac{1}{3}\left[\delta_{s_0^2}(s_1^2)\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1) + \delta_{s_0^1}(s_1^2)\delta_{s_1^1}(t)\mathbb{1}\{s_1^1 < s_0^1\}\mathbb{1}\{u \leq s_1^1\}\left(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}\right)\right]$$
$$+\mathbb{1}\{s_0^1 < t\}\Bigg\{\int_{s_0^1}^\infty 2e^{-2(t'-s_0^1)}\Bigg[\mathbb{1}\{t' \leq s_0^2\}\frac{1}{2}\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\left(\delta_{s_0^2}(s_1^2) + \mathbb{1}\{s_1^2 < s_0^2\}\delta_{s_1^2}(t')\right)$$
$$+\mathbb{1}\{t' > s_0^2\}\mathbb{1}\{s_1^2 > s_0^2\}\delta_{s_0^1}(s_1^1)\delta_{v_0}(v_1)e^{-(s_1^2-s_0^2)}\Bigg]dt'\Bigg\}$$
$$\Bigg\}dt$$

Performing the integrals here gives

$$= \frac{1}{3}(1 - e^{-3(s_0^1-u)})\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2)$$
$$+ e^{-3(s_1^1-u)}\delta_{s_0^1}(s_1^2)\mathbb{1}\{u \leq s_1^1\}\mathbb{1}\{s_1^1 < s_0^1\}\left(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}\right)$$
$$+ e^{-3(s_0^1-u)}\Bigg[\frac{1}{2}(1 - e^{-2(s_0^2-s_0^1)})\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2)$$
$$+ e^{-2(s_1^2-s_0^1)}\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2 < s_0^2\}$$
$$+ e^{-2(s_0^2-s_0^1)}e^{-(s_1^2-s_0^2)}\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2 > s_0^2\}\Bigg]$$

Then, integrating over $u \in [0, s_0^1]$ gives

$$= \frac{1}{3}(s_0^1 - \frac{1}{3}(1 - e^{-3s_0^1}))\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2)$$
$$+ \frac{1}{3}(1 - e^{-3s_1^1})\delta_{s_0^1}(s_1^2)\mathbb{1}\{s_1^1 < s_0^1\}\left(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}\right)$$
$$+ \frac{1}{3}(1 - e^{-3s_0^1})\Bigg[\frac{1}{2}(1 - e^{-2(s_0^2-s_0^1)})\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2)$$
$$+ e^{-2(s_1^2-s_0^1)}\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2 < s_0^2\}$$
$$+ e^{-2(s_0^2-s_0^1)}e^{-(s_1^2-s_0^2)}\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2 > s_0^2\}\Bigg]$$

Case (3) $\mathbb{1}\{u > s_0^1\}$. In words, we calculate here the transition density for a recombination event starting at a cut point anywhere on the upper branches. This situation is analogous to the two sequence calculations already shown in Section 5.2. Using the approach demonstrated

there, $q(\phi_1|A=a,u,B,\phi_0)$ is given by

$$\int_u^\infty 2e^{-2(t-u)}\delta_{s_0^1}(s_1^1)\delta_{v_0}(v_1)\Big\{\mathbb{1}\{u\le t\le s_0^2\}\frac{1}{2}\left[\delta_{s_1^2}(t)\mathbb{1}\{u\le s_1^2\}\mathbb{1}\{s_1^2<s_0^2\}+\delta_{s_0^2}(s_1^2)\right] \\ +\mathbb{1}\{t>s_0^2\}\mathbb{1}\{s_1^2>s_0^2\}e^{-(s_1^2-s_0^2)}\Big\}\,dt$$

$$=\delta_{s_0^1}(s_1^1)\delta_{v_0}(v_1)\Bigg[\mathbb{1}\{s_1^2<s_0^2\}\mathbb{1}\{u\le s_1^2\}e^{-2(s_1^2-u)} \\ +\delta_{s_0^2}(s_1^2)\frac{1}{2}\left(1-e^{-2(s_0^2-u)}\right)+\mathbb{1}\{s_1^2>s_0^2\}e^{-(s_1^2-s_0^2)}e^{-2(s_0^2-u)}\Bigg]$$

Then, integrating over $u\in[s_0^1,s_0^2]$ gives

$$\delta_{s_0^1}(s_1^1)\delta_{v_0}(v_1)\Bigg[\mathbb{1}\{s_1^2<s_0^2\}\frac{1}{2}(1-e^{-2(s_1^2-s_0^1)}) \\ +\delta_{s_0^2}(s_1^2)\frac{1}{2}\left(s_0^2-s_0^1-\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})\right) \\ +\mathbb{1}\{s_1^2>s_0^2\}\frac{1}{2}e^{-(s_1^2-s_0^2)}(1-e^{-2(s_0^2-s_0^1)})\Bigg]$$

We have written down tractable expressions for three conditional probability density functions, they are $q(\phi_1|A=b,u,B\neq v_0,\phi_0)$, $q(\phi_1|A=b,u,B=v_0,\phi_0)$, and $q(\phi_1|A=a,u,B,\phi_0)$. With these expressions, we may write down the marginal transition density $q(\phi_1|\phi_0)$ as follows. First recall that the distribution $q(\phi_1,B|\phi_0)$ may equivalently be written

$$\int_0^{s_0^2}q(\phi_1,u,B|\phi_0)\,du=\int_0^{s_0^2}\Big[q(\phi_1,A=a,u,B|\phi_0) \\ +q(\phi_1,A=b,u,B|\phi_0)\Big]\,du$$

$$=\int_0^{s_0^2}\Big[q(\phi_1|A=a,u,B,\phi_0)q(A=a,u,B|\phi_0) \\ +q(\phi_1|A=b,u,B,\phi_0)q(A=b,u,B|\phi_0)\Big]\,du. \tag{5.5}$$

Moreover, it was shown previously that

$$q(A=a,u,B|\phi_0)=\frac{2(s_0^2-s_0^1)}{2s_0^2+s_0^1}\frac{\mathbb{1}\{B\in[\![1,2]\!]\}}{2}\frac{\mathbb{1}\{s_0^1\le u\le s_0^2\}}{s_0^2-s_0^1}$$

and

$$q(A=b,u,B|\phi_0)=\frac{3s_0^1}{2s_0^2+s_0^1}\frac{\mathbb{1}\{B\in[\![1,3]\!]\}}{3}\frac{\mathbb{1}\{u\leq s_0^1\}}{s_0^1}.$$

Substituting these expressions into Equation (5.5), as well as those of the three probability

densities derived above, one finds that $q(\phi_1, B|\phi_0) = \int_0^{s_0^2} q(\phi_1, u, B|\phi_0)\, du$ is given by

$$
\begin{aligned}
&= \frac{1}{2s_0^2 + s_0^1} \mathbb{1}\{B \in [\![1,2]\!]\} \int_{s_0^1}^{s_0^2} q(\phi_1 | A = a, u, B, \phi_0)\, du \\
&\quad + \frac{1}{2s_0^2 + s_0^1} \mathbb{1}\{B \in [\![1,3]\!]\} \int_0^{s_0^1} q(\phi_1 | A = b, u, B, \phi_0)\, du \\
&= \frac{1}{2s_0^2 + s_0^1} \mathbb{1}\{B \in [\![1,2]\!]\} \frac{1}{2} \delta_{v_0}(v_1) \delta_{s_0^1}(s_1^1) \Bigg[\mathbb{1}\{s_1^2 < s_0^2\}(1 - e^{-2(s_1^2 - s_0^1)}) \\
&\qquad + \delta_{s_0^2}(s_1^2) \left(s_0^2 - s_0^1 - \frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)}) \right) \\
&\qquad\quad + \mathbb{1}\{s_1^2 > s_0^2\} e^{-(s_1^2 - s_0^2)} (1 - e^{-2(s_0^2 - s_0^1)}) \Bigg] \\
&\quad + \frac{1}{2s_0^2 + s_0^1} \mathbb{1}\{B \in [\![1,3]\!]\} \Bigg\{ \\
&\qquad \mathbb{1}\{B \neq v_0\} \Bigg[\frac{1}{3} \left(s_0^1 - \frac{1}{3}(1 - e^{-3s_0^1}) \right) \delta_{v_0}(v_1) \delta_{s_0^1}(s_1^1) \delta_{s_0^2}(s_1^2) \\
&\qquad + \frac{1}{3}(1 - e^{-3s_1^1}) \mathbb{1}\{s_1^1 < s_0^1\} \delta_{s_0^2}(s_1^2) \left(\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c \backslash B\} \right) \\
&\qquad\quad + \frac{1}{3}(1 - e^{-3s_0^1}) \Big[\mathbb{1}\{s_0^1 < s_1^1 < s_0^2\} \delta_{s_0^2}(s_1^2) \left(\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c \backslash B\} \right) e^{-2(s_1^1 - s_0^1)} \\
&\qquad\qquad + \mathbb{1}\{s_1^2 > s_0^2\} \delta_{s_0^2}(s_1^1) \mathbb{1}\{v_1 = B\} e^{-(s_1^2 - s_0^2)} e^{-2(s_0^2 - s_0^1)} \Big] \\
&\qquad\qquad \Bigg] \\
&\qquad + \mathbb{1}\{B = v_0\} \Bigg[\frac{1}{3}(s_0^1 - \frac{1}{3}(1 - e^{-3s_0^1})) \delta_{s_0^2}(s_1^2) \delta_{v_0}(v_1) \delta_{s_0^1}(s_1^1) \\
&\qquad + \frac{1}{3}(1 - e^{-3s_1^1}) \delta_{s_0^1}(s_1^2) \mathbb{1}\{s_1^1 < s_0^1\} \left(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\} \right) \\
&\quad + \frac{1}{3}(1 - e^{-3s_0^1}) \delta_{v_0}(v_1) \Big[\frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)}) \delta_{s_0^1}(s_1^1) \delta_{s_0^2}(s_1^2) \\
&\qquad\quad + e^{-2(s_1^2 - s_0^1)} \delta_{s_0^1}(s_1^1) \mathbb{1}\{s_1^2 < s_0^2\} \\
&\qquad\qquad + e^{-2(s_0^2 - s_0^1)} e^{-(s_1^2 - s_0^2)} \delta_{s_0^1}(s_1^1) \mathbb{1}\{s_1^2 > s_0^2\} \Big] \\
&\qquad\qquad \Bigg] \\
&\qquad \Bigg\}.
\end{aligned}
$$

Marginalising over the possible cut point branch choices $B \in [\![1,3]\!]$, we find that $q(\phi_1|\phi_0) = \sum_{B\in[\![1,3]\!]} p(\phi_1, B|\phi_0)$ is given by

$$
\begin{aligned}
&= \frac{1}{2s_0^2 + s_0^1}\Bigg\{ \delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\Bigg[\mathbb{1}\{s_1^2 < s_0^2\}(1 - e^{-2(s_1^2 - s_0^1)}) \\
&\qquad\qquad + \delta_{s_0^2}(s_1^2)\left(s_0^2 - s_0^1 - \frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)})\right) \\
&\qquad\qquad + \mathbb{1}\{s_1^2 > s_0^2\}e^{-(s_1^2 - s_0^2)}(1 - e^{-2(s_0^2 - s_0^1)})\Bigg] \\
&\quad + \left(s_0^1 - \frac{1}{3}(1 - e^{-3s_0^1})\right)\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2) \\
&\quad + \frac{1}{3}(1 - e^{-3s_1^1})\mathbb{1}\{s_1^1 < s_0^1\}\delta_{s_0^2}(s_1^2)\,(2\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + \frac{1}{3}(1 - e^{-3s_0^1})\Big[\mathbb{1}\{s_0^1 < s_1^1 < s_0^2\}\delta_{s_0^2}(s_1^2)\,(2\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\})\,e^{-2(s_1^1 - s_0^1)} \\
&\quad + \mathbb{1}\{s_1^2 > s_0^2\}\delta_{s_0^2}(s_1^1)\,(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\})\,e^{-(s_1^2 - s_0^2)}e^{-2(s_0^2 - s_0^1)}\Big] \\
&\quad + \frac{1}{3}(1 - e^{-3s_1^1})\delta_{s_0^1}(s_1^2)\mathbb{1}\{s_1^1 < s_0^1\}\,(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + \frac{1}{3}(1 - e^{-3s_0^1})\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1)\Big[\frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)})\delta_{s_0^2}(s_1^2) \\
&\qquad\qquad + e^{-2(s_1^2 - s_0^1)}\mathbb{1}\{s_1^2 < s_0^2\} \\
&\qquad\qquad + e^{-2(s_0^2 - s_0^1)}e^{-(s_1^2 - s_0^2)}\mathbb{1}\{s_1^2 > s_0^2\}\Big]\Bigg\}. \qquad (5.6)
\end{aligned}
$$

In summary we have found the full transition density $q(\phi_{k+1}|\phi_k) = q(s_{k+1}^1, s_{k+1}^2, v_{k+1}|s_k^1, s_k^2, v_k)$, given by Equation (5.6). Note again that the specific case $k = 0$ was taken in the above derivations to minimise visual confusion between adjacent indices; in fact the derivations hold for a general kth move, $k \in \mathbb{N}$.

5.7.3 Verifying the transition density

Analogously to the two-sequence case, it is necessary to verify the transition density developed for the three-sequence algorithm. For completeness, we first check that the integral of the density over its support evaluates to one, i.e. that

$$\int q(\phi_1|\phi_0)\,d\phi_1 = \int_0^\infty \int_{s_1^1}^\infty \sum_{v_1\in[\![1,3]\!]} q(s_1^1, s_1^2, v_1|s_0^1, s_0^2, v_0)\,ds_1^2\,ds_1^1 = 1$$

First, summing over $v_1 \in [\![1,3]\!]$:

$$
\begin{aligned}
&= \frac{1}{2s_0^2+s_0^1}\Bigg\{ \delta_{s_0^1}(s_1^1)\Bigg[\mathbb{1}\{s_1^2<s_0^2\}(1-e^{-2(s_1^2-s_0^1)}) \\
&\quad + \delta_{s_0^2}(s_1^2)\left(s_0^2-s_0^1-\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})\right) \\
&\qquad + \mathbb{1}\{s_1^2>s_0^2\}e^{-(s_1^2-s_0^2)}(1-e^{-2(s_0^2-s_0^1)})\Bigg] \\
&\quad + \left(s_0^1-\frac{1}{3}(1-e^{-3s_0^1})\right)\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2) \\
&\quad + \frac{4}{3}(1-e^{-3s_1^1})\mathbb{1}\{s_1^1<s_0^1\}\delta_{s_0^2}(s_1^2) \\
&\quad + \frac{1}{3}(1-e^{-3s_0^1})\Big[\mathbb{1}\{s_0^1<s_1^1<s_0^2\}\delta_{s_0^2}(s_1^2)4e^{-2(s_1^1-s_0^1)} \\
&\qquad + \mathbb{1}\{s_1^2>s_0^2\}\delta_{s_0^2}(s_1^1)2e^{-(s_1^2-s_0^2)}e^{-2(s_0^2-s_0^1)}\Big] \\
&\quad + \frac{2}{3}(1-e^{-3s_1^1})\delta_{s_0^1}(s_1^2)\mathbb{1}\{s_1^1<s_0^1\} \\
&+ \frac{1}{3}(1-e^{-3s_0^1})\Big[\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2) \\
&\qquad + e^{-2(s_1^2-s_0^1)}\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2<s_0^2\} \\
&\qquad\quad + e^{-2(s_0^2-s_0^1)}e^{-(s_1^2-s_0^2)}\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2>s_0^2\}\Big]\Bigg\}
\end{aligned}
$$

Integrating $s_1^2 \in (s_1^1, \infty)$:

$$
\begin{aligned}
&= \frac{1}{2s_0^2 + s_0^1} \Bigg\{ \delta_{s_0^1}(s_1^1) \Bigg[s_0^2 - s_1^1 - \frac{1}{2}(e^{-2(s_1^1 - s_0^1)} - e^{-2(s_0^2 - s_0^1)}) \\
&\quad + \left(s_0^2 - s_0^1 - \frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)}) \right) \\
&\qquad + (1 - e^{-2(s_0^2 - s_0^1)}) \Bigg] \\
&\quad + \left(s_0^1 - \frac{1}{3}(1 - e^{-3s_0^1}) \right) \delta_{s_0^1}(s_1^1) \\
&\quad + \frac{4}{3}(1 - e^{-3s_1^1}) \mathbb{1}\{s_1^1 < s_0^1\} \\
&\quad + \frac{1}{3}(1 - e^{-3s_0^1}) \Big[\mathbb{1}\{s_0^1 < s_1^1 < s_0^2\} 4 e^{-2(s_1^1 - s_0^1)} \\
&\qquad + \delta_{s_0^2}(s_1^1) 2 e^{-2(s_0^2 - s_0^1)} \Big] \\
&\quad + \frac{2}{3}(1 - e^{-3s_1^1}) \mathbb{1}\{s_1^1 < s_0^1\} \\
&+ \frac{1}{3}(1 - e^{-3s_0^1}) \Big[\frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)}) \delta_{s_0^1}(s_1^1) \\
&\qquad + \delta_{s_0^1}(s_1^1) \frac{1}{2}(e^{-2(s_1^1 - s_0^1)} - e^{-2(s_0^2 - s_0^1)}) \\
&\qquad\quad + e^{-2(s_0^2 - s_0^1)} \delta_{s_0^1}(s_1^1) \Big] \Bigg\}
\end{aligned}
$$

Integrating $s_1^1 \in (0,\infty)$:

$$
\begin{aligned}
&= \frac{1}{2s_0^2+s_0^1}\Bigg\{ s_0^2 - s_0^1 - \frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) \\
&\quad + \left(s_0^2 - s_0^1 - \frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) \right) \\
&\quad\quad + (1-e^{-2(s_0^2-s_0^1)}) \\
&\quad + \left(s_0^1 - \frac{1}{3}(1-e^{-3s_0^1}) \right) \\
&\quad + \frac{4}{3}(s_0^1 + \frac{1}{3}(e^{-3s_0^1}-1)) \\
&\quad + \frac{1}{3}(1-e^{-3s_0^1}) \Big[2(1-e^{-2(s_0^2-s_0^1)}) \\
&\quad\quad + 2e^{-2(s_0^2-s_0^1)} \Big] \\
&\quad + \frac{2}{3}(s_0^1 + \frac{1}{3}(e^{-3s_0^1}-1)) \\
&\quad + \frac{1}{3}(1-e^{-3s_0^1}) \Big[\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) \\
&\quad\quad + \frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) \\
&\quad\quad\quad + e^{-2(s_0^2-s_0^1)} \Big] \Bigg\}
\end{aligned}
$$

Simplifying the above expression yields

$$
\begin{aligned}
&\frac{1}{2s_0^2+s_0^1}\left\{ 2s_0^2 - s_0^1 + \frac{4}{3}s_0^1 + \frac{2}{3}s_0^1 + (1-e^{-3s_0^1})[-\frac{3}{9} - \frac{4}{9} + \frac{6}{9} - \frac{2}{9} + \frac{3}{9}] \right\} \\
&= \frac{1}{2s_0^2+s_0^1}\left\{ 2s_0^2 + s_0^1 + (1-e^{-3s_0^1})[-\frac{3}{9} - \frac{4}{9} + \frac{6}{9} - \frac{2}{9} + \frac{3}{9}] \right\} \\
&= \frac{1}{2s_0^2+s_0^1}\left\{ 2s_0^2 + s_0^1 \right\} \\
&= 1.
\end{aligned}
$$

It is now shown by simulation that samples drawn from the generative description given in Algorithm 5.4 agree in distribution with Equation (5.6). As in the two-sequence case, our first simulation is a visual examination of the empirical sample quantiles against the quantiles of the CDF associated with the transition density. The approach is now the following:

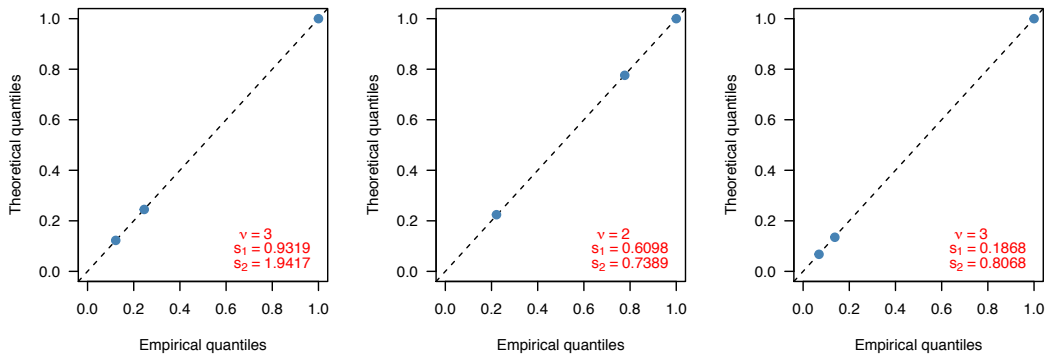


Figure 5.13: *Q-Q plots showing the new lone branch in the topology* $\{\nu_1(i)\}_{i\in[\![1,5\times10^4]\!]}$ *generated by Algorithm 5.4, conditional on the starting state* (ν_0, s_0^1, s_0^2) *shown in red, compared to the theoretical CDF derived above. There is strong agreement.*

1. Fix the current state (ν_0, s_0^1, s_0^2) to an arbitrary value in its support, i.e. $\{1,2,3\}\times[0,\infty)\times(s_0^1,\infty)$.

2. Conditioned on starting at (ν_0, s_0^1, s_0^2), simulate independently R recombination events giving new states $\{\nu_1(i), s_1^1(i), s_1^2(i)\}_{i\in[\![1,R]\!]}$.

3. Compare the empirical quantiles of the marginals of the sample $\{\nu_1(i), s_1^1(i), s_1^2(i)\}_{i\in[\![1,R]\!]}$ against the theoretical quantiles provided by the marginals of the CDF associated with density (5.6).

In Figure 5.13, three independent runs of this experiment are shown. The initial states (ν_0, s_0^1, s_0^2) were sampled randomly by first drawing $s_0^1 \sim \text{Exp}(3)$, then $s_0^2 = s_0^1 + Y$ where $Y \sim \text{Exp}(2)$. Finally, the lone branch ν_0 was picked uniformly at random from the set $\{1,2,3\}$. The Q-Q plots are based on the empirical CDF formed from 5×10^4 samples from the generative algorithm. Analogous experiments are shown in Figures 5.14 and 5.15 for the first and second coalescence times. All three sets of Q-Q plots show almost no observable deviance from a straight line, which provides some support for the correctness of the marginals of Equation (5.6).

Moving now to a more rigorous test, we again carry out a similar test to the two-sequence case by calculating a p-value for a large number of initial states and assessing the uniformity. We adopt the following process:

1. Sample the current state (ν_0, s_0^1, s_0^2) by first drawing $s_0^1 \sim \text{Exp}(3)$, then $s_0^2 = s_0^1 + Y$ where $Y \sim \text{Exp}(2)$, and finally $\nu_0 \sim \mathcal{U}\{1,3\}$.

2. Conditional on starting at (ν_0, s_0^1, s_0^2), simulate independently R recombination events according to the generative description in Algorithm 5.4, giving new states $\{\nu_1(i), s_1^1(i), s_1^2(i)\}_{i\in[\![1,R]\!]}$.

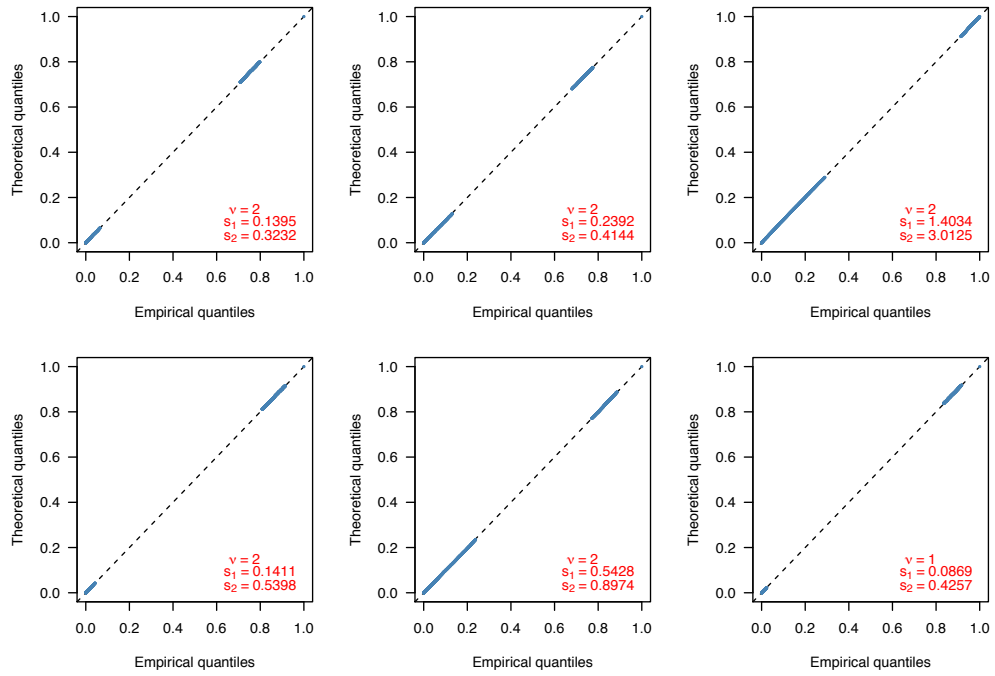


Figure 5.14: *Q-Q plots showing the simulated (first) coalescence times* $\{s_1^1(i)\}_{i\in[\![1,5\times10^4]\!]}$ *generated by Algorithm 5.4, conditional on the starting state* (ν_0, s_0^1, s_0^2) *shown in red, compared to the theoretical CDF derived above.*

3. Taking the empirical marginal quantiles of the sample $\{s_1^i\}_{i\in[\![1,R]\!]}$, and the theoretical marginal quantiles provided by the CDF, and compute the associated p-value with an appropriate test statistic.

We used here for simplicity the specific case of the first coalescence time, but in practice the experiment is repeated to give an analysis of each marginal. As before, the procedure above is repeated, giving many p-values which under the null hypothesis are uniformly distributed on the interval $[0,1]$. For the s_1 and s_2 experiments, a (mixed) Kolmogorov-Smirnov (KS) test statistic [70] was used, while for the ν experiment we adopted Pearson's chi-squared test.

The result of generating 10,000 p-values for each variable in the state by this approach are shown in Figure 5.16, complete with 99% confidence interval for the uniform distribution on the histogram bins shown, calculated using the binomial distribution. The plots provide strong support for the correctness of the marginal distributions of the generative algorithm, though of course they do not constitute a proof of correctness. We note that it is possible for the marginal distribution to be correct and the joint distribution incorrect. Evidence in support of the correctness of the joint distribution is not presented here, as we are aware of no simple approach in the literature for testing samples from a joint distribution with both discrete components and mixed discrete/continuous components. However, it is hoped that

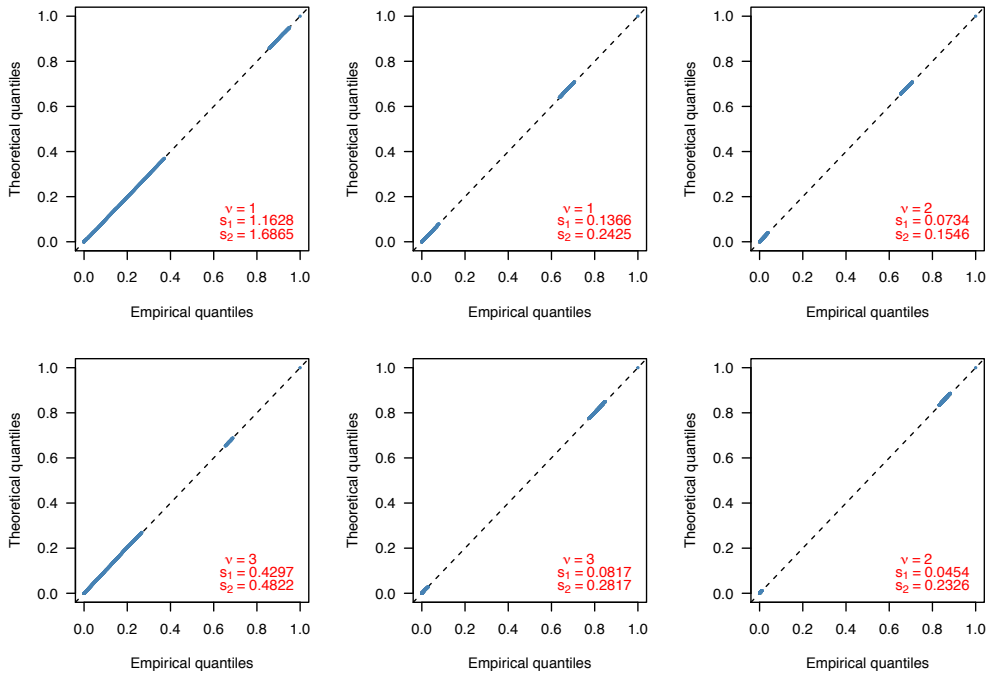
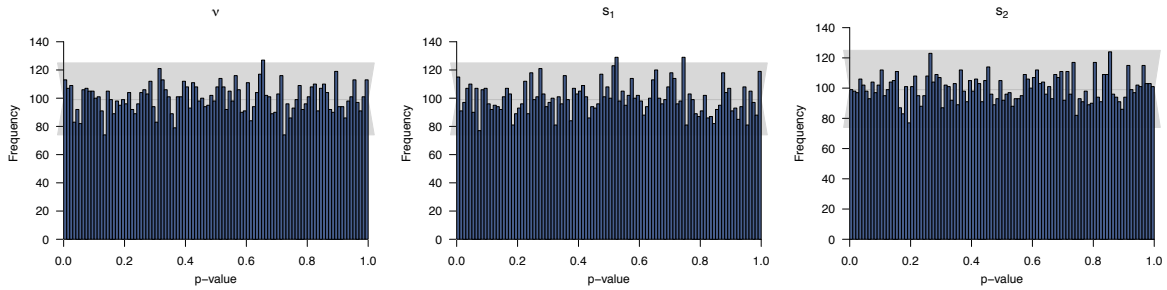


Figure 5.15: *Q-Q plots showing the simulated (second) coalescence times* $\{s_1^2(i)\}_{i\in[\![1,5\times10^4]\!]}$ *generated by Algorithm 5.4, conditional on the starting state* (ν_0, s_0^1, s_0^2) *shown in red, compared to the theoretical CDF derived above.*



(a) *P-values calculated using a Pearson chi-squared test.*

(b) *P-values calculated using a mixed KS test.*

(c) *P-values calculated using a mixed KS test.*

Figure 5.16: *Each histogram shows* $10,000$ *replications of the p-value under the tests described in each sub-figure. For each replication,* $R = 100$ *samples from the generative process were used to construct the empirical CDF. The uniformity in the plots provides strong evidence for the null hypothesis; namely, that the generative samples (Algorithm 5.4) are being drawn (marginally) from the same distribution as the CDF derived from Equation (5.6).*

with evidence for the correctness of the marginal distributions of the generative algorithm, together with a full Bayesian calibration of the inference algorithm presented later in this work, there will be sufficient evidence to consider the density correct or subject to inconsequential error.

5.7.4 Overcoming the three-sequence problem

We now concern ourselves with overcoming the drawbacks, discussed in Section 5.6.1, of a particle filter that proposes particles according to the SMC' model. It is desirable to be able sample topologies of a given shape, which we have denoted v, since using this strategy can obviate the proposal of particles incompatible with mutations in the sequence. In practice this solution would involve *looking ahead* to the next problematic mutation event - that is a SNP where exactly two of the three sequences have the mutation. Earlier it was shown that only one topology (of the three *essential* topologies; see Figure 5.11) will be compatible with any problematic mutation. This means if the last state ϕ_k proposed before the problematic mutation had one of the two incompatible topologies then that particle will die (its weight will be zero). However, if it were possible to sample from some relevant conditional distributions related to $q(\phi_{k+1}|\phi_k)$, then in principle the algorithm could 'look' ahead to see which topology v_k will be compatible with the problematic mutation. It would then sample the heights (s_k^1, s_k^2) conditional on the compatible v_k. In sum, it would be possible to propose particles on the support of the posterior distribution. For this reason we now turn our attention to calculating the marginal distribution $q(v_{k+1}|s_k^1, s_k^2, v_k)$ and the conditional distribution $q(s_{k+1}^1, s_{k+1}^2|v_{k+1}, s_k^1, s_k^2, v_k)$.

Again, we use the notation of the $k = 0$ case for visual clarity. Notice that $q(v_1|s_0^1, s_0^2, v_0) = \int\int q(ds_1^1, ds_1^2, v_1|s_0^1, s_0^2, v_1)$, and therefore firstly integrating $s_1^2 \in (s_1^1, \infty)$ yields:

$$
\begin{aligned}
&= \frac{1}{2s_0^2+s_0^1}\Bigg\{ \delta_{s_0^1}(s_1^1)\delta_{v_0}(v_1)\Bigg[s_0^2 - s_1^1 + \frac{1}{2}(e^{-2(s_0^2-s_0^1)} - e^{-2(s_1^1-s_0^1)}) \\
&\qquad\qquad + s_0^2 - s_0^1 - \frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) + (1-e^{-2(s_0^2-s_0^1)})\Bigg] \\
&\quad + \left(s_0^1 - \frac{1}{3}(1-e^{-3s_0^1})\right)\delta_{v_0}(v_1)\delta_{s_0^1}(s_1^1) \\
&\quad + \frac{1}{3}(1-e^{-3s_1^1})\mathbb{1}\{s_1^1 < s_0^1\}\,(2\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + \frac{1}{3}(1-e^{-3s_0^1})\Big[\mathbb{1}\{s_0^1 < s_1^1 < s_0^2\}e^{-2(s_1^1-s_0^1)}\,(2\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + \delta_{s_0^2}(s_1^1)e^{-2(s_0^2-s_0^1)}\,(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\})\Big] \\
&\quad + \frac{1}{3}(1-e^{-3s_1^1})\mathbb{1}\{s_1^1 < s_0^1\}\,(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + \frac{1}{3}(1-e^{-3s_0^1})\delta_{v_0}(v_1)\Big[\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})\delta_{s_0^1}(s_1^1) \\
&\qquad\qquad + \frac{1}{2}\left(e^{-2(s_1^1-s_0^1)} - e^{-2(s_0^2-s_0^1)}\right)\delta_{s_0^1}(s_1^1) + e^{-2(s_0^2-s_0^1)}\delta_{s_0^1}(s_1^1)\Big]\Bigg\}
\end{aligned}
$$

Then integrating $s_1^1 \in (0,\infty)$:

$$
\begin{aligned}
&= \frac{1}{2s_0^2+s_0^1}\Bigg\{ \delta_{v_0}(v_1)\Bigg[s_0^2 - s_0^1 - \frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) \\
&\qquad\qquad + s_0^2 - s_0^1 - \frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) + (1-e^{-2(s_0^2-s_0^1)})\Bigg] \\
&\quad + \left(s_0^1 - \frac{1}{3}(1-e^{-3s_0^1})\right)\delta_{v_0}(v_1) \\
&\quad + \frac{1}{3}(s_0^1 - \frac{1}{3}(1-e^{-3s_0^1}))\,(2\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + \frac{1}{3}(1-e^{-3s_0^1})\Big[\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})\,(2\delta_{v_0}(v_1) + \mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + e^{-2(s_0^2-s_0^1)}\,(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\})\Big] \\
&\quad + \frac{1}{3}(s_0^1 - \frac{1}{3}(1-e^{-3s_0^1}))\,(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}) \\
&\quad + \frac{1}{3}(1-e^{-3s_0^1})\delta_{v_0}(v_1)\Big[\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)}) + \frac{1}{2}\left(1-e^{-2(s_0^2-s_0^1)}\right) + e^{-2(s_0^2-s_0^1)}\Big]\Bigg\}
\end{aligned}
$$

Rearranging, we find

$$
\begin{aligned}
&q(v_1|s_0^1, s_0^2, v_0) \\
&\quad = \frac{1}{2s_0^2 + s_0^1}\left\{ \delta_{v_0}(v_1)\left[2s_0^2 - \frac{1}{3}s_0^1 + \frac{1}{9}(1-e^{-3s_0^1})\left(1-3e^{-2(s_0^2-s_0^1)}\right)\right]\right. \\
&\qquad \left. + (\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\})\left[\frac{2}{3}s_0^1 - \frac{1}{18}(1-e^{-3s_0^1})\left(1-3e^{-2(s_0^2-s_0^1)}\right)\right]\right\}.
\end{aligned}
$$

This may be written as $q(v_1|s_0^1, s_0^2, v_0) = a_1\delta_{v_0}(v_1) + a_2\left(\mathbb{1}\{v_1 = v_0^c[1]\} + \mathbb{1}\{v_1 = v_0^c[2]\}\right)$, or

$$
q(v_1|s_0^1, s_0^2, v_0) = \begin{cases} a_1 & \text{if } v_1 = v_0 \\ a_2 & \text{if } v_1 \in \{v_0\}^c \end{cases}
$$

where

$$
a_1 := \frac{2s_0^2 - \frac{1}{3}s_0^1 + \frac{1}{9}(1-e^{-3s_0^1})\left(1-3e^{-2(s_0^2-s_0^1)}\right)}{2s_0^2 + s_0^1},
$$

and

$$
a_2 := \frac{\frac{2}{3}s_0^1 - \frac{1}{18}(1-e^{-3s_0^1})\left(1-3e^{-2(s_0^2-s_0^1)}\right)}{2s_0^2 + s_0^1},
$$

and $a_1 + 2a_2 = 1$.

5.7.5 Sampling the next recombination point

We now consider the question of how to sample the next recombination point. Let the recombination points be denoted $(\tau_j)_{j\in\mathbb{N}\cup\{0\}}$ with the convention that $\tau_0 = 0$, and let $\phi_j := (s_j^1, s_j^2, v_j)$ for $j \in \mathbb{N}\cup\{0\}$ be the coalescent history associated with the sequence in the interval $(\tau_j, \tau_{j+1}]$. Recall that the SMC' dynamics (Algorithm 5.1) imply a transition density for the jth recombination event of the form

$$
p(\tau_j, \phi_j|\tau_{j-1}, \phi_{j-1}) = f(\tau_j|\tau_{j-1}, \phi_{j-1})q(\phi_j|\phi_{j-1}).
$$

Currently we are able to sample from

$$
q(\phi_j|\phi_{j-1}) = p(s_j^1, s_j^2, v_j|s_{j-1}^1, s_{j-1}^2, v_{j-1})
$$

directly (Algorithm 5.4), and alternatively by first conditioning on a particular value of v_j then updating (s_j^1, s_j^2) given ϕ_{j-1} and v_j (Algorithms 5.5, 5.6). We show presently that, for the purposes of proposing particles that are compatible with observed mutations, sampling recombination points from the prior distribution $f(\cdot|\tau_{j-1}, \phi_{j-1})$ is unsatisfactory. It remains to design a method of proposing state transitions at useful recombination points.

The difficulty in proposing recombination points τ_j can be demonstrated with a simple example. Suppose the SMC' algorithm has undergone a recombination event at τ_j and is subsequently in a state $\phi_j = (s_j^1, s_j^2, v_j)$. Now suppose that further along the sequence a mutation of an incompatible type is encountered at a position denoted $m \in (0,1)$, and of course $m > \tau_j$. The mutation at m is assumed to be compatible only with topologies of type $v_{(m)}$. According to the usual SMC' dynamics, a recombination point τ_{j+1} is generated stochastically according to

$$\tau_{j+1} - \tau_j \sim \text{Exp}(\frac{\rho}{2} L(\phi_j))$$

where $L(\phi_j) = 2s_j^2 + s_j^1$ is the total length of the tree associated with the state ϕ_j. Suppose this recombination point occurs before the mutation at m, i.e. $\tau_{j+1} < m$. Next a state ϕ_{j+1} must be generated, but it is unclear by what method. If no more recombination points will be generated under the model before the mutation at m then we would like ϕ_{j+1} to be generated conditional on $v_{j+1} = v_{(m)}$ to ensure the particle survives. On the other hand if it were possible to know in advance that another recombination point τ_{j+2} would be proposed before point m then we would instead allow ϕ_{j+1} to be generated according to the unrestricted SMC' dynamics and insist that ϕ_{j+2} be generated conditional on $v_{j+2} = v_{(m)}$. Underlying our argument here is a preference to sample recombination times from their prior distribution under the model. Unfortunately proposing times in this way offers no guarantee that a recombination event will occur before an anticipated problematic mutation. For this reason we must make what are in some sense *unnatural* changes to this prior to ensure a recombination event happens before the mutation. Nevertheless our aim is to explore a strategy making as few such interventions as possible. One might think of generating first all the recombination points in the interval $(\tau_j, m]$ and afterwards sampling the concomitant states ϕ. This solution is however infeasible in the situation where the proposal of the next recombination point depends on the most recent state ϕ, as is the case with $f(\tau_j|\tau_{j-1}, \phi_{j-1})$.

A promising approach is now described. First, given the interval $(\tau_j, m]$ on the sequence, generate recombination events (τ_i, ϕ_i) according to the prior until one exceeds m. To be precise, suppose having started from a state (τ_j, ϕ_j) that further events $(\tau_{j+i}, \phi_{j+i})_{i \in \{1,2,\ldots\}}$ are proposed according to SMC' dynamics, and that the first recombination point to exceed m is τ_{j+k+1} for some $k \in \mathbb{N}$. In other words, we have $\tau_j < \tau_{j+1} < \ldots < \tau_{j+k} < m$ and $m < \tau_{j+k+1}$. Then return to recombination point τ_{j+k}, the last before m, and sample a new state ϕ_{j+k} condi-

tional on $v_{j+k} = v_{(m)}$. It is clear that this will introduce a bias that must be corrected to retain exactness of the algorithm.

For notational simplicity, suppose the epoch under consideration is $(t_i, m]$, and suppose the particle filter is constructed so that m is the next point at which resampling takes place. Here we make use of the notation $x_{[t_i,m]}$ to describe the entire PDP, that is the set of recombination points and coalescence times, from t_i (including the state in which the PDP arrived at t_i) up to the mutation at point m. The mutations in the same region are describe by the observation variable $y_{(t_i,m]}$. Then the weight of the particle described in the interval $[t_i, m]$ is given by

$$\begin{aligned}&\frac{p_{[t_i,m]}(k,\tau_{1:k},\phi_{0:k})g(y_{(t_i,m]}|x_{[t_i,m]})}{p_{[t_i,m]}(k,\tau_{1:k},\phi_{0:k-1})q(s_k^1,s_k^2|v_k = v_{(m)},\phi_{k-1})}\\&= \frac{p_{[t_i,m]}(k,\tau_{1:k},\phi_{0:k})}{\int p_{[t_i,m]}(k,\tau_{1:k},\phi_{0:k-1},\phi_k^*)\,d\phi_k^*} \times \frac{g(y_{(t_i,m]}|x_{[t_i,m]})}{q(s_k^1,s_k^2|v_k = v_{(m)},\phi_{k-1})}\end{aligned}$$

where ϕ_k^* may be thought of as the 'forgotten' last state, the value of which is overwritten by a value consistent with the problematic mutation following it. Substituting the prior distribution defined in Equation (5.4), the first fraction is equivalent to

$$\frac{S(\tau_k,m|\phi_k)q_0(\phi_0)\left(\prod_{j=1}^k f(\tau_j|\tau_{j-1},\phi_{j-1})\right)\left(\prod_{j=1}^k q(\phi_j|\phi_{j-1})\right)}{\int S(\tau_k,m|\phi_k^*)q_0(\phi_0)\left(\prod_{j=1}^k f(\tau_j|\tau_{j-1},\phi_{j-1})\right)\left(\prod_{j=1}^{k-1} q(\phi_j|\phi_{j-1})\right)q(\phi_k^*|\phi_{k-1})\,d\phi_k^*}$$

which, cancelling all terms independent of ϕ_k^*, reduces to

$$\frac{S(\tau_k,m|\phi_k)q(\phi_k|\phi_{k-1})}{\int S(\tau_k,m|\phi_k^*)q(\phi_k^*|\phi_{k-1})\,d\phi_k^*}.$$

Combining results, the weight may be expressed as

$$\begin{aligned}&\frac{p_{[t_i,m]}(k,\tau_{1:k},\phi_{0:k})g(y_{(t_i,m]}|x_{[t_i,m]})}{p_{[t_i,m]}(k,\tau_{1:k},\phi_{0:k-1})q(s_k^1,s_k^2|v_k = v_{(m)},\phi_{k-1})}\\&= \frac{S(\tau_k,m|\phi_k)q(\phi_k|\phi_{k-1})}{\int S(\tau_k,m|\phi_k^*)q(\phi_k^*|\phi_{k-1})\,d\phi_k^*}\frac{g(y_{(t_i,m]}|x_{[t_i,m]})}{q(s_k^1,s_k^2|v_k = v_{(m)},\phi_{k-1})}.\end{aligned} \tag{5.7}$$

Notice that since, by the definition of conditional probability,

$$q(\phi_k|\phi_{k-1}) = q(s_k^1,s_k^2|v_k,\phi_{k-1})q(v_k|\phi_{k-1})$$

it follows that when $\phi_k = (s_k^1, s_k^2, v_{(m)})$ we have

$$q(\phi_k|\phi_{k-1}) = q(s_k^1, s_k^2|v_{(m)}, s_{k-1}^1, s_{k-1}^2, v_{k-1}) q(v_{(m)}|s_{k-1}^1, s_{k-1}^2, v_{k-1}).$$

Note that we need only consider the weight associated with a final state $\phi_k = (s_k^1, s_k^2, v_k)$ with $v_k = v_{(m)}$, since if $v_k \neq v_{(m)}$ then the weight of the particle is guaranteed to be zero by the likelihood term $g(y_{(0,t_n]}|x_{0:n})$. This argument allows us to make the simplification

$$\frac{q(\phi_k|\phi_{k-1})}{q(s_k^1, s_k^2|v_k = v_{(m)}, \phi_{k-1})} = q(v_k = v_{(m)}|\phi_{k-1}).$$

Now, defining a quantity from Equation (5.7):

$$\begin{aligned} H(\phi_{k-1}, \tau_k, m) &:= \int \Pr(\tau_{k+1} > m|\tau_k, \phi_k) q(\phi_k|\phi_{k-1})\, d\phi_k \\ &= \int S(\tau_k, m|\phi_k) q(\phi_k|\phi_{k-1})\, d\phi_k \\ &= \int_0^\infty \int_{s_k^1}^\infty \sum_{v_k \in [\![1,3]\!]} S(\tau_k, m|s_k^1, s_k^2, v_k) q(s_k^1, s_k^2, v_k|s_{k-1}^1, s_{k-1}^2, v_{k-1})\, ds_k^2\, ds_k^1. \end{aligned} \tag{5.8}$$

one has the following expression for the weights

$$\begin{aligned} &\frac{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k}) g(y_{(t_i,m]}|x_{[t_i,m]})}{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k-1}) q(\phi_k|v_k = v_{(m)}, \phi_{k-1})} \\ &= \frac{S(\tau_k, m|\phi_k) q(\phi_k|\phi_{k-1})}{H(\phi_{k-1}, \tau_k, m)} \frac{g(y_{(t_i,m]}|x_{[t_i,m]})}{q(\phi_k|v_k = v_{(m)}, \phi_{k-1})} \\ &= \frac{S(\tau_k, m|\phi_k) q(v_k = v_{(m)}|\phi_{k-1})}{H(\phi_{k-1}, \tau_k, m)} g(y_{(t_i,m]}|x_{[t_i,m]}). \end{aligned} \tag{5.9}$$

It will be shown in the sequel that H is tractable and so our weights are well defined in practice.

We now examine the no-event case. To retain exactness of the particle filter it is crucial that in any interval the algorithm allow for no event to occur. In particular, this implies that for any interval ending in a problematic mutation there must be a non-zero probability of no new recombination events in the interval, and so a non-zero probability that a particle will die. However, as discussed above, if too many particles have weight zero then the performance of the algorithm deteriorates and it may fail altogether. It is now shown that the probability with which a particle will have no recombination events can be controlled using importance sampling techniques. Consider the following decomposition of the prior distribution of k

jumps and their positions:

$$\begin{aligned} p(k,\tau_{1:k}) &= \delta_0(k)p(k=0) + \mathbb{1}\{k\geq 1\}p(k\geq 1)p(k,\tau_1,\ldots,\tau_k|k\geq 1) \\ &= \delta_0(k)p(k=0) + \mathbb{1}\{k\geq 1\}(1-p(k=0))p(k,\tau_1,\ldots,\tau_k|k\geq 1). \end{aligned}$$

Now define the proposal density

$$p_\alpha(k,\tau_{1:k}) \coloneqq \alpha\delta_0(k) + (1-\alpha)\mathbb{1}\{k\geq 1\}p(k,\tau_1,\ldots,\tau_k|k\geq 1)$$

for some $\alpha \in (0,1)$. Recall from Section 5.3 the definition

$$\mathbb{T}_{n,k} \coloneqq \{\tau_{1:k} \in (0,\infty)^k : 0 < \tau_1 < \ldots < \tau_k \leq t_n\}$$

of the space of length-k sequences of positive, ordered jump times bounded by some t_n . We have for any $t_n \in (0,1)$, and any $\alpha \in (0,1)$,

$$\begin{aligned} \sum_{k=0}^{\infty}\int_{\mathbb{T}_{n,k}} p_\alpha(k,\tau_{1:k})\,d\tau_{1:k} &= \alpha + (1-\alpha)\sum_{k=1}^{\infty}\int_{\mathbb{T}_{n,k}} p(k,\tau_1,\ldots,\tau_k|k\geq 1)\,d\tau_{1:k} \\ &= \alpha + 1 - \alpha \\ &= 1 \end{aligned}$$

thus p_α is a probability density function for any α and in particular any $\alpha \in [0,1]$.

With these results, we introduce a strategy we refer to as *boosting* that will ensure correctness of the algorithm whilst permitting control over the probability of loss of particles in intervals that end with problematic mutations. Suppose the present interval of the particle filter, denoted here $(t_i,m]$, ends with a problematic mutation m. Then, for each particle, with probability α no recombination events will occur in the interval. With probability $1-\alpha$ some positive number of recombination events will occur. In other words, with probability α the number of recombination events will be chosen according to the density $\delta_0(k)$ and with probability $1-\alpha$ the number and position of recombination events will be chosen according to the density $\mathbb{1}\{k\geq 1\}p(k,\tau_1,\ldots,\tau_k|k\geq 1)$. The particle filter is *boosted* in the sense that the number of particles still alive at the end of an epoch with a problematic mutation is increased, and so the accuracy of the estimators is improved as is the stability of the algorithm.

The weights associated with the two moves are now derived. First consider the case of no event, or $k=0$. The correct particle weight in this case is given by

$$w_{[t_i,m]}(k=0) = \frac{p_{[t_i,m]}(k=0)}{\alpha} g(y_{(t_i,m]}|x_{[t_i,m]}). \tag{5.10}$$

Secondly we consider the weight of a particle for which the number of events in the interval $(t_i, m]$ is $k > 0$. In this case

$$w_{[t_i,m]}(k) = \frac{1}{1-\alpha} \frac{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k}) g(y_{(t_i,m]}|x_{[t_i,m]})}{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k-1}|k \geq 1) q(\phi_k | v_k = v_{(m)}, \phi_{k-1})}.$$

Notice that since

$$\begin{aligned} p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k-1}|k \geq 1) &= \mathbb{1}\{k \geq 1\} \frac{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k-1})}{p_{[t_i,m]}(k \geq 1)} \\ &= \mathbb{1}\{k \geq 1\} \frac{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k-1})}{1 - p_{[t_i,m]}(k = 0)} \end{aligned}$$

and using properties of the exponential distribution we have

$$1 - p_{[t_i,m]}(k = 0) = 1 - e^{-\frac{\rho L(t_i)}{2}(m - t_i)}$$

where $L(t_i)$ simply refers to the total branch length established at the last recombination event before t_i. Finally then the weight is found analogously to Equation (5.9):

$$\begin{aligned} w_{[t_i,m]}(k) &= \frac{1}{1-\alpha} \frac{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k}) g(y_{(t_i,m]}|x_{(t_i,m]})}{p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k-1}|k \geq 1) q(\phi_k | v_k = v_{(m)}, \phi_{k-1})} \\ &= \frac{S(\tau_k, m|\phi_k) q(v_k = v_{(m)}|\phi_{k-1})}{(1-\alpha) H(\phi_{k-1}, \tau_k, m)} \left(1 - e^{-\frac{\rho L(t_i)}{2}(m - t_i)}\right) g(y_{(t_i,m]}|x_{(t_i,m]}). \end{aligned} \tag{5.11}$$

We now turn to calculation of the quantity $H(\phi_{k-1}, \tau_k, m)$. Since $S(\tau, t|\phi) := 1 - \int_\tau^t f(ds|\tau, \phi)$, we have

$$\begin{aligned} S(\tau_k, m|s_k^1, s_k^2, v_k) &= e^{-\frac{\rho}{2} L(\phi_k)(m - \tau_k)} \\ &= e^{-\frac{\rho}{2}(2s_k^2 + s_k^1)(m - \tau_k)}. \end{aligned}$$

Consider without loss of generality the case $k = 1$. Notice that since $S(\tau_1, m|s_1^1, s_1^2, v_1) = S(\tau_1, m|s_1^1, s_1^2) = e^{-\frac{\rho}{2}(2s_1^2 + s_1^1)(m - \tau_1)}$, writing $\lambda := \frac{\rho}{2}(m - \tau_1)$,

$$
\begin{aligned}
H(\phi_0,\tau_1,m) &= \int_0^\infty \int_{s_1^1}^\infty \sum_{v_1\in[\![1,3]\!]} S(\tau_1,m|s_1^1,s_1^2,v_1)q(s_1^1,s_1^2,v_1|s_0^1,s_0^2,v_0)\,ds_1^2\,ds_1^1 \\
&= \int_0^\infty \int_{s_1^1}^\infty e^{-\lambda(2s_1^2+s_1^1)} \sum_{v_1\in[\![1,3]\!]} q(s_1^1,s_1^2,v_1|s_0^1,s_0^2,v_0)\,ds_1^2\,ds_1^1 \\
&= \int_0^\infty \int_{s_1^1}^\infty e^{-\lambda(2s_1^2+s_1^1)} q(s_1^1,s_1^2|s_0^1,s_0^2,v_0)\,ds_1^2\,ds_1^1. \qquad (5.12)
\end{aligned}
$$

Now, simplifying an expression derived above, we have already the density

$$
\begin{aligned}
q(s_1^1,s_1^2|\phi_0) = \frac{1}{2s_0^2+s_0^1}\Bigg\{ & \delta_{s_0^1}(s_1^1)\Bigg[\mathbb{1}\{s_1^2<s_0^2\}(1-e^{-2(s_1^2-s_0^1)}) \\
& + \delta_{s_0^2}(s_1^2)\left(s_0^2-\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})-\frac{1}{3}(1-e^{-3s_0^1})\right) \\
& \quad + \mathbb{1}\{s_1^2>s_0^2\}e^{-(s_1^2-s_0^2)}(1-e^{-2(s_0^2-s_0^1)})\Bigg] \\
& + \frac{1}{3}(1-e^{-3s_1^1})\mathbb{1}\{s_1^1<s_0^1\}\left(4\delta_{s_0^2}(s_1^2)+2\delta_{s_0^1}(s_1^2)\right) \\
& \quad + \frac{1}{3}(1-e^{-3s_0^1})\Big[\mathbb{1}\{s_0^1<s_1^1<s_0^2\}\delta_{s_0^2}(s_1^2)4e^{-2(s_1^1-s_0^1)} \\
& \qquad + \mathbb{1}\{s_1^2>s_0^2\}\delta_{s_0^2}(s_1^1)2e^{-(s_1^2-s_0^2)}e^{-2(s_0^2-s_0^1)}\Big] \\
& + \frac{1}{3}(1-e^{-3s_0^1})\Big[\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})\delta_{s_0^1}(s_1^1)\delta_{s_0^2}(s_1^2) \\
& \quad + e^{-2(s_1^2-s_0^1)}\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2<s_0^2\} \\
& \qquad + e^{-2(s_0^2-s_0^1)}e^{-(s_1^2-s_0^2)}\delta_{s_0^1}(s_1^1)\mathbb{1}\{s_1^2>s_0^2\}\Big] \\
& \Bigg\}
\end{aligned}
$$

Multiplying through by the factor $S(\tau_1,m|s_1^1,s_1^2,v_1)$ and integrating, as in Equation (5.12), gives

$$
\begin{aligned}
&\int_{s_1^1}^{\infty} S(\tau_1, m|\phi_1) q(s_1^1, s_1^2|\phi_0)\, ds_1^2 \\
&= \frac{1}{2s_0^2 + s_0^1} \Bigg\{ \delta_{s_0^1}(s_1^1) \Bigg[\frac{1}{2\lambda} e^{-\lambda s_1^1} (e^{-2\lambda s_1^1} - e^{-2\lambda s_0^2}) \\
&\qquad - \frac{1}{2(1+\lambda)} e^{-\lambda s_1^1} e^{2s_0^1} (e^{-2(1+\lambda)s_1^1} - e^{-2(1+\lambda)s_0^2}) \\
&\qquad + \left(s_0^2 - \frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)}) - \frac{1}{3}(1 - e^{-3s_0^1}) \right) e^{-\lambda(2s_0^2 + s_1^1)} \\
&\qquad + \frac{1}{1+2\lambda} e^{-\lambda(2s_0^2 + s_1^1)} (1 - e^{-2(s_0^2 - s_0^1)}) \Bigg] \\
&\quad + \frac{1}{3}(1 - e^{-3s_1^1}) \mathbb{1}\{s_1^1 < s_0^1\} \left(4e^{-\lambda(2s_0^2 + s_1^1)} + 2e^{-\lambda(2s_0^1 + s_1^1)} \right) \\
&\quad + \frac{1}{3}(1 - e^{-3s_0^1}) \Big[\mathbb{1}\{s_0^1 < s_1^1 < s_0^2\} 4e^{-\lambda(2s_0^2 + s_1^1)} e^{-2(s_1^1 - s_0^1)} \\
&\qquad + \frac{1}{1+2\lambda} e^{-\lambda(2s_0^2 + s_1^1)} 2\delta_{s_0^2}(s_1^1) e^{-2(s_0^2 - s_0^1)} \Big] \\
&\quad + \frac{1}{3}(1 - e^{-3s_0^1}) \delta_{s_0^1}(s_1^1) \Big[\frac{1}{2}(1 - e^{-2(s_0^2 - s_0^1)}) e^{-\lambda(2s_0^2 + s_1^1)} \\
&\qquad + \frac{1}{2(1+\lambda)} e^{2s_0^1 - \lambda s_1^1} (e^{-2(1+\lambda)s_1^1} - e^{-2(1+\lambda)s_0^2}) \delta_{s_0^1}(s_1^1) \\
&\qquad + e^{-2(s_0^2 - s_0^1)} \frac{1}{1+2\lambda} e^{-\lambda(2s_0^2 + s_1^1)} \Big] \\
&\quad \Bigg\}
\end{aligned}
$$

Finally, integrating this expression with respect to s_1^1 gives

$$
\begin{aligned}
&H(\phi_0,\tau_1,m)\\
&=\frac{1}{2s_0^2+s_0^1}\Bigg\{\frac{1}{2\lambda}e^{-\lambda s_0^1}(e^{-2\lambda s_0^1}-e^{-2\lambda s_0^2})\\
&\quad-\frac{1}{2(1+\lambda)}e^{-(\lambda-2)s_0^1}(e^{-2(1+\lambda)s_0^1}-e^{-2(1+\lambda)s_0^2})\\
&\quad+\left(s_0^2-\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})-\frac{1}{3}(1-e^{-3s_0^1})\right)e^{-\lambda(2s_0^2+s_0^1)}\\
&\quad+\frac{1}{1+2\lambda}e^{-\lambda(2s_0^2+s_0^1)}(1-e^{-2(s_0^2-s_0^1)})\\
&\quad+\frac{1}{3}(\frac{1}{\lambda}(1-e^{-\lambda s_0^1})-\frac{1}{3+\lambda}(1-e^{-(3+\lambda)s_0^1}))(4e^{-2\lambda s_0^2}+2e^{-2\lambda s_0^1})\\
&\quad+\frac{1}{3}(1-e^{-3s_0^1})\Big[\frac{4}{2+\lambda}e^{-2\lambda s_0^2}e^{2s_0^1}(e^{-(2+\lambda)s_0^1}-e^{-(2+\lambda)s_0^2})\\
&\qquad\qquad+\frac{2}{1+2\lambda}e^{-\lambda(2s_0^2+s_0^2)}e^{-2(s_0^2-s_0^1)}\Big]\\
&\quad+\frac{1}{3}(1-e^{-3s_0^1})\Big[\frac{1}{2}(1-e^{-2(s_0^2-s_0^1)})e^{-\lambda(2s_0^2+s_0^1)}\\
&\qquad\qquad+\frac{1}{2(1+\lambda)}e^{2s_0^1-\lambda s_0^1}(e^{-2(1+\lambda)s_0^1}-e^{-2(1+\lambda)s_0^2})\\
&\qquad\qquad+e^{-2(s_0^2-s_0^1)}\frac{1}{1+2\lambda}e^{-\lambda(2s_0^2+s_0^1)}\Big]\\
&\quad\Bigg\}
\end{aligned}
$$

In summary, we have found that the expression for the correct particle weights is tractable. In the case of an interval not ending in a problematic mutation, the correct weight is given by Equation (5.9). If an interval ends in a problematic mutation, then the correct weight is given by Equation (5.10) in the case of choosing no event (with probability α) and by Equation (5.11) in the case of choosing a positive number of events (with probability $1-\alpha$).

5.7.6 A new algorithm

The techniques explored above are now combined into a novel particle filter for the SMC' model. First we choose a set of positions along the sequence that will define the intervals of the particle filter. Any choice is valid and the method we propose here is as follows: decide on an initial partition of the interval $[0,1]$ and denote it $\mathcal{T}' := \{0 = t'_0, t'_1, \ldots, t'_{P'} = 1\}$. Then take the set of mutation positions $\{m_1, m_2, \ldots, m_r\}$ corresponding to all mutations of *problematic*

type and add them to this set. This results in an ordered set of times

$$\mathcal{T} := \{0 = t_0, t_1, \ldots, t_P = 1\}$$

with an associated indicator vector $\mathcal{I} := (I_1, \ldots, I_P)$ where

$$I_s := \begin{cases} 0 & t_s \in \mathcal{T}' \\ 1 & t_s \notin \mathcal{T}' \end{cases}$$

describes whether epoch $(t_{s-1}, t_s]$ ends with a problematic mutation ($I_s = 1$) or not ($I_s = 0$). If $I_s = 0$, the epoch $(t_{s-1}, t_s]$ is such that $t_s \in \mathcal{T}'$. For such epochs the particle filter generates paths according to the model, as described in Algorithm 5.1 with particles weighted simply by their likelihood. If $I_s = 1$, the epoch $(t_{s-1}, t_s]$ is such that t_s is a problematic mutation. For these epochs the particle filter will follow the *boosting* approach described in Section 5.7.5. With probability α a particle will experience no recombination events in the epoch - this may lead to it being given a weight zero if the topology with which it entered the epoch is not compatible with the problematic mutation at t_s. Otherwise, with probability $1 - \alpha$, the particle will experience a positive number of events, and the last of these will have by design a state ϕ compatible with the mutation at t_s. A precise description is found in Algorithm 5.7, wherein an instance of the notation (i) implies that the operation should be performed for all particles $i \in [\![1, N]\!]$.

5.7.7 Implementation details

There are several details of the implementation of Algorithm 5.7 that require further consideration.

We first explore the choice of epochs. Since for a particular set of observations (mutations) the positions of the problematic mutations are fixed, there is no way to change these interval end-points. There is however control over all others, as the user may pick any initial partition $\mathcal{T}' := \{0 = t'_0, t'_1, \ldots, t'_{P'} = 1\}$ of $[0, 1]$. Note that we must have as a minimum $\{0, 1\} \subseteq \mathcal{T}'$ for Algorithm 5.7 to make sense. Our experiments to date have made use of a regular grid, evenly dividing the interval into P' non-overlapping sub intervals. This approach is principled - no large region of the sequence is left without a point where resampling can take place in the particle filter - though it may not be optimal. Perhaps it is most effective to set $\mathcal{T}' = \{0, 1\}$, so that $\mathcal{T} = \{0, m_1, m_2, \ldots, m_r, 1\}$ and every interval end-point is a problematic mutation, save for the last. Certainly this approach will be computationally efficient, it is in fact the minimum number of resampling points possible for the particle filter as we have defined it (since $\mathcal{T}'$

Algorithm 5.7 Particle filter (with boosting) for the SMC' model

Input: number of particles N, indicator vector $\mathcal{I}$, epoch times $\mathcal{T}$, probability of no event α, parameters (θ, ρ).

1. Sample $\phi_0^{(i)} \sim q_0(\cdot)$. Set $x_0^{(i)} \leftarrow (\tau_0 = 0, \phi_0^{(i)})$.
2. Set $w_0(x_0^{(i)}) \leftarrow 1$ and $W_0^i \propto w_0(x_0^{(i)})$.
3. Set $n \leftarrow 1$.
4. **if** $I_n = 0$
   Sample events $(k_n^{(i)}, \tau^{(i)}_{1:k_n^{(i)}}, \phi^{(i)}_{1:k_n^{(i)}}) \sim p_{[t_{n-1},t_n]}(x_n|\bar{x}_{n-1}^{(i)})$
   if $I_n = 1$
   With probability α set $k_n^{(i)} \leftarrow 0$,
   otherwise sample events

$$(k_n^{(i)}, \tau^{(i)}_{1:k_n^{(i)}}, \phi^{(i)}_{1:k_n^{(i)}}) \sim p_{[t_{n-1},t_n]}(k_n^{(i)}, \tau^{(i)}_{1:k_n^{(i)}}, \phi^{(i)}_{1:k_{n-1}^{(i)}}|k_n^{(i)} \geq 1, \bar{x}_{n-1}^{(i)}) q(s^{(i),1}_{k_n^{(i)}}, s^{(i),2}_{k_n^{(i)}}|v^{(i)}_{k_n^{(i)}} = v_{(t_n)}, \phi^{(i)}_{k_{n-1}^{(i)}}).$$

5. Set $x_n^{(i)} \leftarrow (k_n^{(i)}, \tau^{(i)}_{1:k_n^{(i)}}, \phi^{(i)}_{1:k_n^{(i)}})$ and $x_{0:n}^{(i)} \leftarrow (\bar{x}_{0:n-1}^{(i)}, x_n^{(i)})$.
6. Compute the particle weight:

if $I_n = 0$, $w_n(x_{0:n}^{(i)}) = g(y_{(t_{n-1},t_n]}|x_{0:n}^{(i)})$,
if $I_n = 1$ and $k_n^{(i)} = 0$,

$$w_n(x_{0:n}^{(i)}) = \frac{p_{[t_{n-1},t_n]}(k_n^{(i)} = 0)}{\alpha} g(y_{(t_{n-1},t_n]}|x_{0:n}^{(i)})$$

if $I_n = 1$ and $k_n^{(i)} > 0$,

$$w_n(x_{0:n}^{(i)}) = \frac{S(\tau_{k_n^{(i)}}, t_n|\phi_{k_n^{(i)}}) q(v_{k_n^{(i)}} = v_{(t_n)}|\phi_{k_{n-1}^{(i)}})}{(1-\alpha) H(\phi_{k_{n-1}^{(i)}}, \tau_{k_n^{(i)}}, t_n)} \left(1 - e^{-\frac{\rho L(t_{n-1})}{2}(t_n - t_{n-1})}\right) g(y_{(t_{n-1},t_n]}|x_{0:n}^{(i)}).$$

7. Set $\hat{p}(y_{(t_{n-1},t_n]}|y_{(0,t_{n-1}]}) \leftarrow \frac{1}{N}\sum_{i\in[\![1,N]\!]} w_n(x_{0:n}^{(i)})$
8. Calculate $W_n^i \propto w_n(x_{0:n}^{(i)})$.
9. Resample according to weights W_n^i obtaining equally weighted particles $\{\frac{1}{N}, \bar{x}_{0:n}^{(i)}\}$.
10. **if** $n < P$: set $n \leftarrow n+1$, **go to** (5).

Output: Likelihood estimate obtained via $\hat{p}(y_{(0,t_P]}) = \prod_{n\in[\![1,P]\!]} \hat{p}(y_{(t_{n-1},t_n]}|y_{(0,t_{n-1}]})$.

must contain the points $\{0,1\}$). Future research would look further into this.

The relative effectiveness of a fixed grid $\mathcal{T}'$ can be explored through the experiment shown in Figure 5.17. Each of the three plots shows the effective sample size (ESS) within each epoch

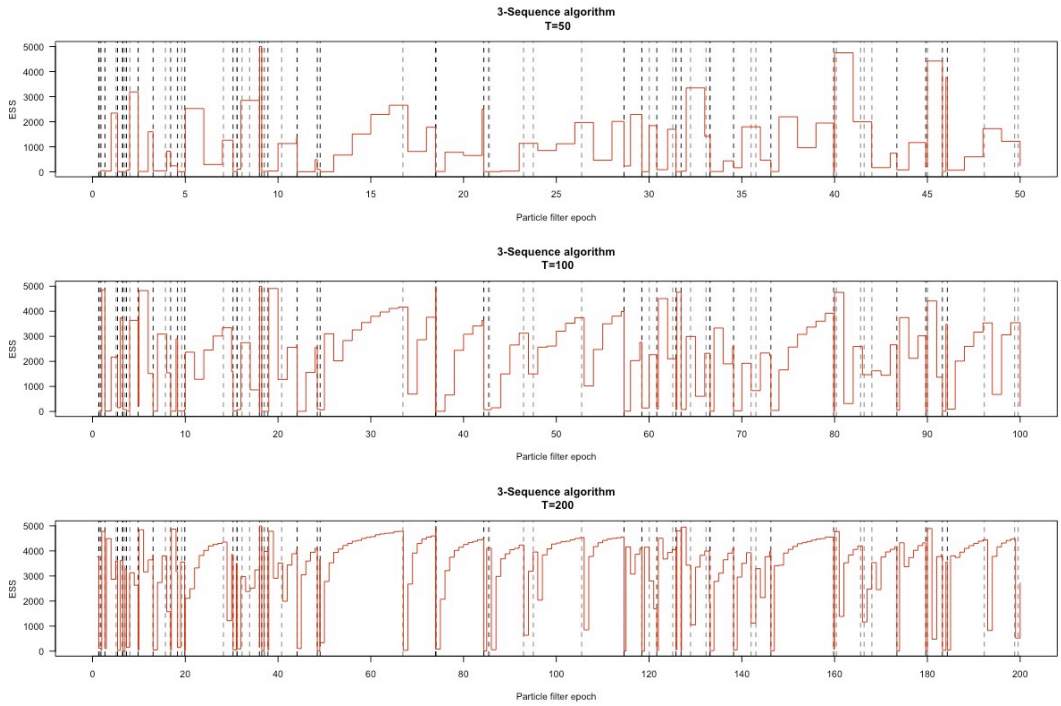


Figure 5.17: *The three sequence particle filter with boosting applied with initial partition* $\mathcal{T}' := \{0 = t'_0, t'_1, \ldots, t'_{P'} = 1\}$ *where* $P' \in \{50, 100, 200\}$. *Recall* P' *is simply the number of epochs in the initial partition, before including problematic mutations.*

of the algorithm. The three algorithms use an initial partition $\mathcal{T}' := \{0 = t'_0, t'_1, \ldots, t'_{P'} = 1\}$ with $P' = 50, 100, 200$ respectively. Clearly, increasing the density of the initial partition increases the average ESS.

Secondly we consider the choice of α, the probability of having no events in an epoch ending with a problematic mutation. A sensible choice here will be important for reducing the weight variance and improving the overall efficiency of algorithm. Taking too small an α will compromise the performance of the algorithm; too large an α and many particles will be wasted in intervals with problematic mutations. An appealing strategy is to use a fixed probability $\alpha \in (0,1)$ for all particles. This is simple to implement and crucially offers a high level of mathematical tractability. Consider an epoch ending in a problematic mutation. Suppose each particle independently undergoes no recombination in the epoch with probability α. Then, assuming the particle filter uses N particles, the total number of no-event particles in the epoch, N^0 , is binomially distributed $N^0 \sim \mathrm{B}(N, \alpha)$. Since the $N - N^0$ remaining particles are *boosted* to be compatible with the problematic mutation in the interval, the number

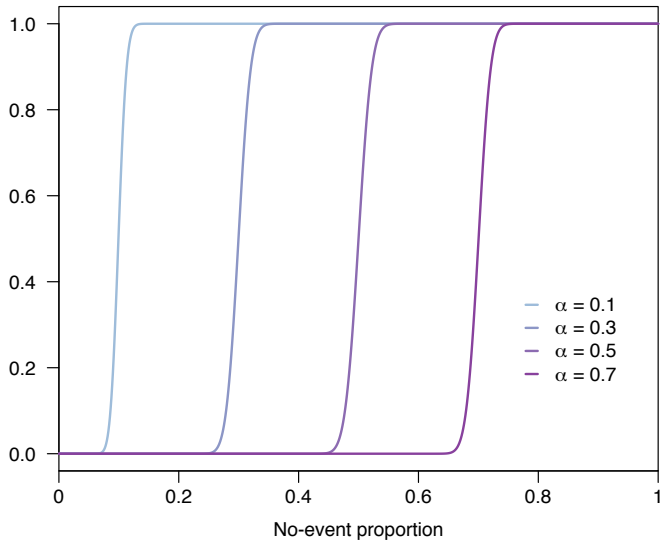


Figure 5.18: *For a fixed-α strategy, the cumulative distribution function of the proportion of particles with no event in an interval. Here we considered $N = 800$ particles as an example.*

of zero-weight particles is bounded above by N^0 and so an understanding of its distribution is useful. Figure 5.18 shows the cumulative distribution function of N^0/N, the proportion of particles experiencing no recombination in the epoch, for $\alpha \in \{0.1, 0.3, 0.5, 0.7\}$. The figure is based on $N = 800$ particles but it changes very little with changes in N. Effectively it shows that the variance of the number of no-event particles is small. For example, if we choose $\alpha = 0.3$, the proportion of no-event particles is bounded above by 0.35, at least with probability 0.999 (around four standard deviations from the mean). Of course other schemes are possible. One could take α to be state-dependent or even observation dependent. It is perhaps desirable to select no events for particles where it makes most sense, for example using the strategy $\alpha \propto p(k = 0)$ where only the particles most likely under the prior to undergo no events actually do so. We do not pursue this strategy further here.

Finally there is a programming consideration. Computationally, there are several ways to sample from $p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k-1} | k \geq 1)$ and specifically $p_{[t_i,m]}(k, \tau_{1:k} | k \geq 1)$. The first is a simple rejection method in which PDP trajectories are proposed according to the model density $p_{[t_i,m]}(k, \tau_{1:k}, \phi_{1:k})$ and those for which $k = 0$ are simply rejected. This could prove computationally expensive in cases where the probability of no event is large. We use instead here a truncated exponential distribution to simulate the position of the first recombination event. In particular we take the following approach. Suppose again the particle filter is presently at an interval $(t_i, m]$, and moreover (with probability $1 - \alpha$) the particle under consideration has been chosen to receive a positive number of recombination events. Then the position τ_1 of this

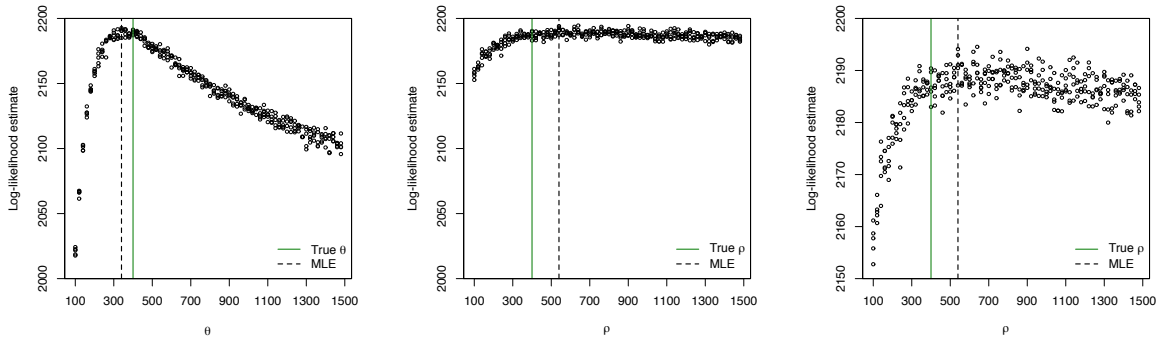


Figure 5.19: *Approximate likelihood curves of the SMC' model for three sequences. First a synthetic set of data (mutations) for three sequences was produced using the SMC' model and the parameters* $(\theta,\rho) = (400,400)$. *Then one parameter, say* ρ, *is fixed and a boosted particle filter is run using this* ρ *and a range of values of* θ *- the log-likelihood estimate of the particle filter is recorded. From left to right the plots are: (a)* $\rho = 400, \theta \in (100,1500)$, *(b)* $\theta = 400, \rho \in (100,1500)$, *and plot (c) is identical to plot (b) but with a different y-axis scale. This is intended to show the similarity of the* θ *and* ρ *curves but also the relative flatness of the* ρ *likelihood.*

first recombination event is proposed according to the truncated version of the model density $f(\tau_1|t_i,\phi_0,m)$:

$$\tilde{f}(\tau_1|t_i,\phi_0,m) = \frac{\mathbb{1}\{t_i \leq \tau_1 \leq m\}\frac{\rho L(t_i)}{2}e^{-\frac{\rho L(t_i)}{2}(t_i-\tau_0)}}{1-e^{-\frac{\rho L(t_i)}{2}(m-t_i)}}$$

from which it is straightforward to simulate using an inverse CDF method (see e.g. [80]).

5.7.8 Performance

First we examine the likelihood curves produced by the three sequence boosted approach (Algorithm 5.7), which are shown in Figure 5.19. The plots are fairly similar to those of the two sequence algorithm, shown in Figure 5.6, though here we take a far more computationally demanding example. From the leftmost plot in the figure, it seems the algorithm determines θ well, though again the likelihood curve for ρ is much less informative. In Section 5.8 we tackle the uninformative likelihood through the conditional dependence structure of the prior distribution $p(\theta,\rho)$.

We now compare the performance of the naive VRPF to the boosted algorithm presented here. In principle we are interested in comparing a metric like the effective sample size (ESS) in Figure 5.17. Unfortunately we have found the ESS to be a poor indicator of performance when comparing approaches where a large number of the particles have weight zero. To be precise, for a single particle filter (featuring zero-weight particles) there is some meaning to

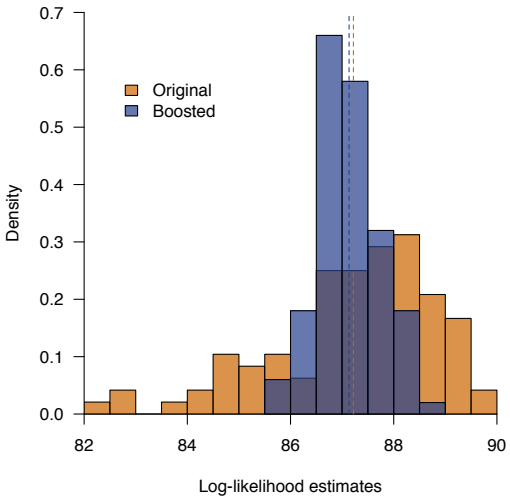


Figure 5.20: *A comparison of the original VRPF approach and our 'boosted' algorithm. For each method, 100 particle filters were run independently. The resulting log-likelihood estimates are shown in the histograms above. The dotted lines show the sample means for the two sets of 100 estimates. Based on a fixed grid of size* $P' = 40$*, and choosing* $\alpha = 0.2$*.*

the measure when comparing the ESS between epochs. However in comparing between two different particle filters for a single epoch, the inference one would like to make from the ESS is confounded by the difference in the proportion of zero-weight particles between the two approaches. The ESS is effectively 'blind' to zero-weight particles (further criticism of the ESS in particle filtering can be found in [81]).

Instead we can compare the VRPF to the boosted algorithm more directly, using the variance of the log likelihood estimates produced. Figure 5.20 shows the possible reduction in variance of the likelihood estimates produced by the particle filter achieved through using boosting, compared to the VRPF method. For this example, the variance of the VRPF estimated log-likelihoods was 2.578 and the variance of the boosted algorithm's estimated log-likelihoods was 0.369; a near seven-fold improvement. In practice this makes an enormous difference in the pseudo-marginal method where, as discussed in Chapter 2, the success of the Markov chain Monte Carlo using an unbiased estimate in place of the true likelihood depends to a large extent on the variance of the 'noise' term associated with the estimate.

Figure 5.21 shows the enormous benefits of the boosting algorithm for stability. The original VRPF approach comes very close in some epochs to failing altogether because the number of wasted particles (zero-weight particles) reaches almost the total number N, even with $N = 5000$. For this reason the algorithm is unreliable. In contrast the boosted algorithm limits the extent to which its performance is affected by the proposal of particles that will be wasted. By restricting the probability with which a 'wasted' particle may be proposed, the algorithm

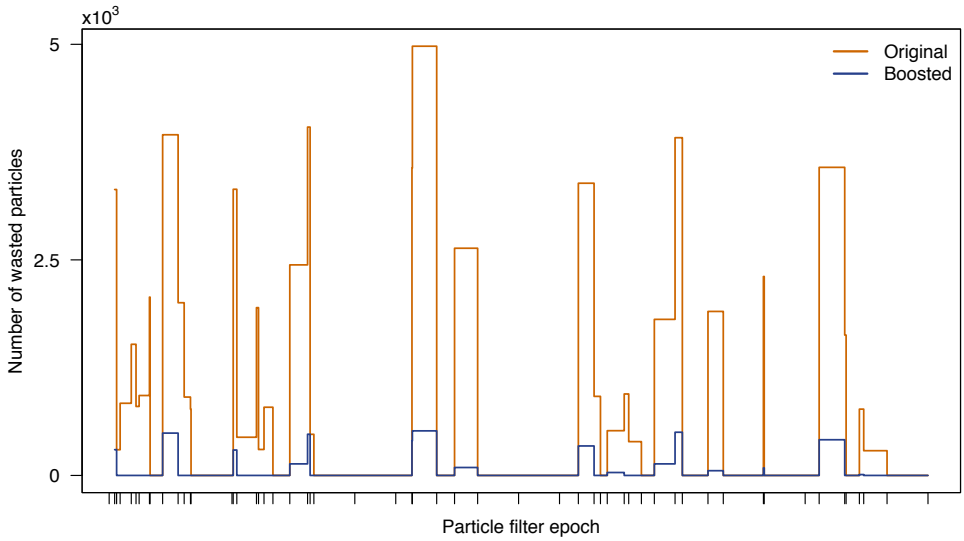


Figure 5.21: *In orange is the number of wasted particles (zero-weight particles) in each epoch of the particle filter for the original algorithm, and shown in blue is the new (boosted) algorithm. For this experiment* $\alpha = 0.1$ *was used, resulting in a more stable output.*

retains exactness while improving stability and variance of its output.

5.7.9 Comparison with ARGweaver

As was done for the two-sequence particle filter, we now compare the three-sequence particle filter to the ARGweaver software. We use a similar approach again, comparing the posterior number of recombination events under each of the two approaches, given simulated data.

Figure 5.22 shows six realisations of the experiment as detailed in Section 5.5. Again, the posterior numbers of recombinations are similar in each case and both are reasonably close to the number of recombination events of the true ARG. While the mean squared error of the posterior means generated by ARGweaver was higher on average for these six runs than for the posterior means using the three-sequence particle filter developed here, it was not computationally feasible to run the full MSE experiment.

5.8 Particle MCMC for SMC'

As briefly discussed in the introduction to this chapter, given an unbiased estimate of the likelihood of the mutations given the static parameters it is possible to run a particle Markov chain Monte Carlo algorithm (introduced in [6]) targeting the posterior distribution of the parameters.

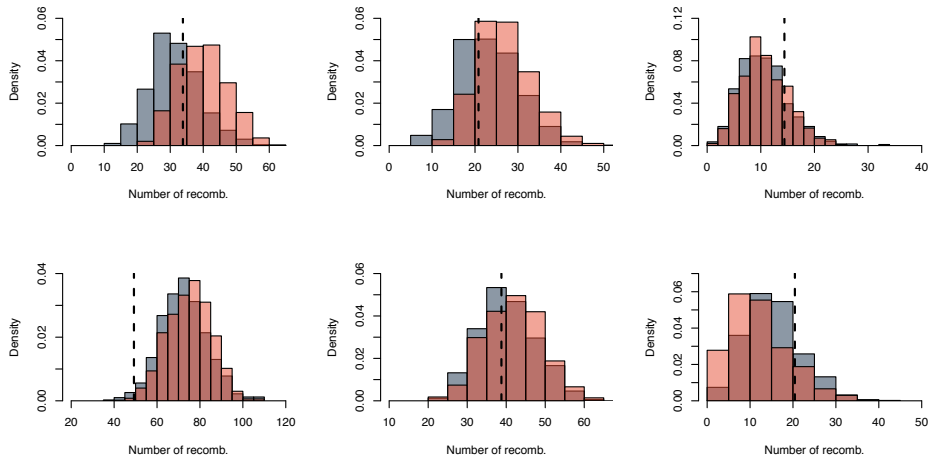


Figure 5.22: *Six realisations of the posterior number of recombinations, given distinct mutation data generated by the SMC' algorithm. ARGweaver is shown in red, while the three-sequence particle filter proposed in this work is shown in grey. $N = 256$ particles were used in each particle filter.*

Recall from Section 2.1 that the acceptance ratio of the Metropolis-Hastings algorithm is given by

$$\alpha(\vartheta, \vartheta') = \min\left\{1, \frac{\pi(\vartheta')q(\vartheta', \vartheta)}{\pi(\vartheta)q(\vartheta, \vartheta')}\right\}. \tag{5.13}$$

Where only an unnormalised version of the target distribution π is available, denoted

$$\tilde{\pi}(\vartheta) = Z\pi(\vartheta)$$

with normalising constant Z, notice that this may also be used since the normalising constant cancels. The target distribution of interest is the posterior density of the SMC' model parameters $\vartheta = (\theta, \rho)$ given the observed mutations $\mathcal{M}$. Therefore, defining $\pi(\vartheta) := p(\vartheta|\mathcal{M})$, we have by Bayes' theorem:

$$\pi(\vartheta) = \frac{p_\vartheta(\mathcal{M})p(\vartheta)}{p(\mathcal{M})}.$$

Thus an unnormalised target is given by

$$\tilde{\pi}(\vartheta) = p_\vartheta(\mathcal{M})p(\vartheta)$$

where $p_\vartheta(\mathcal{M})$ is used as a shorthand for $p(\mathcal{M}|\vartheta)$. Moreover the pseudo-marginal method [1, 2] permits the use of a positive, unbiased estimate $\hat{p}_\vartheta(\mathcal{M})$ in place of the true likelihood

$p_{\vartheta}(\mathcal{M})$. In summary, the acceptance ratio we employ is of the form

$$\hat{\alpha}(\vartheta, \vartheta') = \min\left\{1, \frac{\hat{p}_{\vartheta'}(\mathcal{M})p(\vartheta')q(\vartheta', \vartheta)}{\hat{p}_{\vartheta}(\mathcal{M})p(\vartheta)q(\vartheta, \vartheta')}\right\}.$$

For our purposes, $\hat{p}_{\vartheta}(\mathcal{M})$ is an unbiased, positive likelihood estimate obtained as a result of running a particle filter with parameter values $\vartheta = (\theta, \rho)$ and observations $\mathcal{M}$ (as in Algorithm 5.7). A prior distribution $p(\vartheta)$ on the parameter space Θ must also be specified, as well as a proposal density $q(\vartheta, \cdot)$ for the parameter vector, and a thoughtful choice here is essential to the performance of the algorithm. A priori the only physical constraint on the parameters is $(\theta, \rho) \in \mathbb{R}_+^2$ since both θ and ρ are rates. In the absence of further information, this might suggest use of the improper prior $\Pr(\theta, \rho) \propto 1$. We avoid this here to ensure we are guaranteed a proper posterior distribution. Instead we start by considering a uniform prior distribution for both parameters, bounded above by a large (but essentially arbitrary) value. The choice of prior distribution will be discussed in more depth in the following sections, as will the choice of proposal, which will require careful thought to overcome a quirk of the SMC' model.

5.8.1 Validating a Bayesian algorithm

Before exploring the performance of the two and three-sequence particle filters in the context of particle MCMC, it is essential to test the correctness of the algorithms. Many algorithms in computational statistics are esoteric and highly complex. Yet, in the absence of an established and well-tested software package or library, users must write their own (often substantial) code bases to facilitate use of a particular algorithm. Even when the mathematics is well understood, and this is by no means certain, it is extremely likely for bugs to be found in even short sections of code written from scratch. The problem is made no easier by the stochastic nature of many simulation heavy algorithms, like Markov chain Monte Carlo, which can mask small errors to a degree. Without a rigorous validation process, simulations from an algorithm cannot be taken to be reliable. Despite this, code validation is seldom referred to explicitly in methodological papers in computational statistics. Many Bayesian statisticians will be familiar with some form of posterior checking. For example, in a repeated experiment one takes a sample from the prior distribution, generates data conditional on it, and observes whether or not the true value lies within, for example, the associated 95% highest posterior density region, as it should in 95% of such experiments. While this is correct, such approaches lack rigour when performed in an ad hoc fashion and so a more principled Bayesian workflow must be sought.

One of the earliest works in Bayesian validation is the Gibbs sampling approach of Geweke

[82]. The technique relies on the comparison of two distinct Gibbs samplers for simulating from the joint distribution $p(\vartheta, y)$. Though still used, the approach lacks flexibility as it requires the user to change the code that is being tested so that it can resample the data given the observation, precluding the evaluation of black-box inference methods. Another criticism is that it is difficult to identify when the Gibbs sampler used in the method has converged sufficiently, and only once this has occurred are the scores it produces meaningful [83]. More recently, two popular approaches to Bayesian validation are those of Cook et al. [84] and Talts et al. [83]. Both rely on posterior quantiles, though the latter has now superseded the former, which contains errors [85] and is known in some cases to perform poorly [86].

We now describe the *simulation-based calibration* (SBC) method of [83]. In a Bayesian model, the joint distribution over parameters ϑ and data y can be written as the product of the prior and likelihood terms, that is, $p(\vartheta, y) = p(y|\vartheta)p(\vartheta)$. It is desirable to exploit a kind of internal self-consistency present in the relationship between the joint and posterior distributions. Consider drawing a 'true' parameter value $\tilde{\vartheta} \sim p(\vartheta)$, then generating data $\tilde{y} \sim p(y|\tilde{\vartheta})$, and finally drawing samples from the posterior distribution given those observations $p(\vartheta|\tilde{y})$. Investigating the distribution of such posterior samples, marginal to the parameter and observation drawn, one finds

$$\begin{aligned}\int p(\tilde{\vartheta})p(\tilde{y}|\tilde{\vartheta})p(\vartheta|\tilde{y})\,\mathrm{d}\tilde{\vartheta}\,\mathrm{d}\tilde{y} &= \int p(\tilde{\vartheta})p(\tilde{y}|\tilde{\vartheta})\,\mathrm{d}\tilde{\vartheta}\,p(\vartheta|\tilde{y})\,\mathrm{d}\tilde{y} \\ &= \int p(\tilde{y})p(\vartheta|\tilde{y})\,\mathrm{d}\tilde{y} \\ &= p(\vartheta). \end{aligned} \tag{5.14}$$

Thus in any Bayesian model, the data-averaged posterior is identical to the prior distribution. This is the desired self-consistency result, and indeed leads directly to a straightforward validation procedure. Suppose L samples $\{\vartheta_1, \ldots, \vartheta_L\}$ are taken from the posterior distribution following the process just described. By Equation (5.14) they will be distributed according to the prior distribution in a properly executed program. In order to make this assessment in practice, we define for any $f : \Theta \to \mathbb{R}$ the *rank statistic* of the prior sample relative to the posterior samples as

$$r(\{f(\vartheta_1), \ldots, f(\vartheta_L)\}, f(\tilde{\vartheta})) = \sum_{l=1}^{L} \mathbb{1}[f(\vartheta_l) < f(\tilde{\vartheta})] \in [\![0, L]\!]$$

This rank statistic is uniformly distributed across the integers $[\![0, L]\!]$ ([83, Theorem 1.]). The rank statistic, as it is defined here, is a strictly one-dimensional concept and so for a higher-dimensional parameter ϑ the rank statistic must be computed independently for each dimen-

Algorithm 5.8 Simulation Based Calibration

Input: A histogram initialised with bins centred on the integers $\{1,\dots,L\}$

1. Sample from the prior $\tilde{\vartheta} \sim p(\vartheta)$
2. Generate data $\tilde{y} \sim p(y|\tilde{\vartheta})$
3. Sample from the posterior $\{\vartheta_1,\dots,\vartheta_L\} \sim p(\vartheta|\tilde{y})$
4. **for** each dimension of the parameter **do**
   Increment the histogram with the rank statistic $r(\{f(\vartheta_1),\dots,f(\vartheta_L)\}, f(\tilde{\vartheta}))$
5. **go to** (1).

Output: The histogram of rank statistics, to be checked for uniformity.

sion of ϑ. There are many possible methods to test the uniformity of the rank statistic. In [83] a visual inspection of a histogram of rank statistics is employed for its diagnostic properties; particular deviations from uniformity in the histogram can signal causes of error, not merely that an error is present. The simulation based calibration algorithm is given in Algorithm 5.8, which should be run for as many iterations as is feasible. We note here that model checking is a valuable counterpart to model validation and should also be undertaken by the practitioner. One's model can of course be a very poor representation of the real world, and still be internally consistent with respect to Bayesian inference.

In MCMC, only correlated samples from the posterior distribution are available, which will affect the performance of the SBC method. One can for example easily imagine the first $L = 100$ samples from a posterior distribution generated using Metropolis-Hastings obtaining only one or two distinct values, from which it follows that the highest and lowest bins in the SBC histogram will be over-represented. To combat this, thinning strategies are adopted to reduce the autocorrelation present in the sample as far as possible. Recall from Section 2.2 that for a Markov chain $\{\vartheta_1,\vartheta_2,\dots\}$ at equilibrium, with transition kernel Π, M samples have an effective sample size of $M/\tau(f,\Pi)$. Here, f is a square integrable function with respect to the stationary distribution and $\tau(f,\Pi) := 1+2\sum_{k=1}^{\infty}\rho_k[f]$, where $\rho_k[f]$ is the correlation between the random variables $f(\vartheta_0)$ and $f(\vartheta_k)$, in other words the lag-k autocorrelation. Motivated by this, a strategy is proposed in [83] for generating sufficiently many correlated samples from an MCMC posterior. Suppose L effective samples are desired. First, sample L' steps of the Markov chain, and then assess the number of effective samples

$$M_{\text{eff}}[f] := \frac{L'}{1+2\sum_{k=1}^{\infty}\rho_k[f]}. \tag{5.15}$$

Algorithm 5.9 Simulation Based Calibration for MCMC

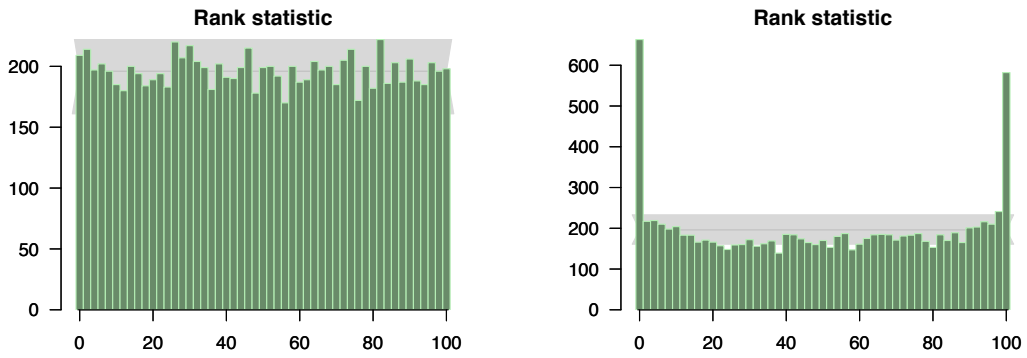
Input: A histogram initialised with bins centred on the integers $\{1,\dots,L\}$

1. Sample from the prior $\tilde{\vartheta} \sim p(\vartheta)$
2. Generate data $\tilde{y} \sim p(y|\tilde{\vartheta})$
3. Sample L' iterations from the posterior $\{\vartheta_1,\dots,\vartheta_{L'}\} \sim p(\vartheta|\tilde{y})$
4. Compute the effective sample size $M_{\text{eff}}[f]$ of $\{\vartheta_1,\dots,\vartheta_{L'}\}$ for the function f using (5.15)
5. **if** $M_{\text{eff}}[f] < L$ **then**
   Rerun the Markov chain for $L' \cdot L/M_{\text{eff}}[f]$ iterations
6. Thin the chain uniformly to obtain L samples.
7. **for** each dimension of the parameter **do**
   Increment the histogram with the rank statistic $r(\{f(\vartheta_1),\dots,f(\vartheta_L)\}, f(\tilde{\vartheta}))$
8. **go to** (1).

Output: The histogram of rank statistics, to be checked for uniformity.

It is then suggested that the chain is rerun for $L' \cdot L/M_{\text{eff}}[f] = L(1+2\sum_{k=1}^{\infty}\rho_k[f])$ iterations. Certainly it is plausible that the rerun chain will achieve an effective sample size close to L and so mitigate to a large extent the artifacts of autocorrelation in the SBC histogram. However, it is implied that the original L' samples are discarded in this procedure, when they could be reused. If it was not intended for the samples to be discarded, another explanation is that the first L' iterations are considered a kind of *burn-in* for the Markov chain, yet this seems unlikely as it is not suggested that the 'rerun' chain begin from the position at which the original chain ended. The second approach here is a practical addition to the method that we would encourage, as it avoids a needless repetition of the burn-in period. Nevertheless, in our simulations we implement the approach as it is written, and as is quoted in Algorithm 5.9. Note that for a model with multiple parameters it is sensible to take the minimum of the effective samples sizes when thinning the chain.

We now briefly explore some properties of the SBC method in a toy example. Consider a standard Metropolis-Hastings algorithm targetting the posterior distribution $p(\vartheta|y) \propto p(y|\vartheta)p(\vartheta)$ with mixture prior $p(\vartheta) = \gamma N(\vartheta;-1,2^2) + (1-\gamma)N(\vartheta;10,3^2)$ and likelihood $p(y|\vartheta) = N(y;\vartheta,8^2)$. For the following experiments we set $\gamma = 0.4$, and used as proposal kernel a Gaussian random walk with standard deviation $\sigma = 3$. An application of the SBC procedure in Algorithm 5.9 on this example is shown in Figure 5.23. In keeping with many



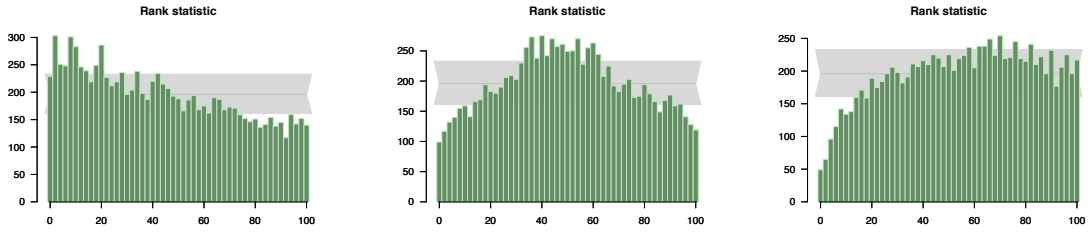
(a) *Conservative thinning* **(b)** *No thinning*

Figure 5.23: *A toy implementation of SBC, targetting* $p(\vartheta|y)$ *where* $p(\vartheta) = \gamma N(\vartheta; -1, 2^2) + (1-\gamma)N(\vartheta; 10, 3^2)$ *and* $p(y|\vartheta) = N(y; \vartheta, 8^2)$*. The mixture proportion was* $\gamma = 0.4$ *and the algorithm was run for* $10,000$ *steps.*

examples in [83], we used 10,000 replications to populate the histograms. In addition, $L = 101$ posterior samples were drawn in each replication to facilitate grouping of the rank statistics into 51 bins in the histogram, each covering two integer outcomes. No advice is given in [83] on the choice of L', however in our experiments the integrated autocorrelation times were almost never small enough to render a choice of $L' = 500$ wasteful, and it is advisable to select a reasonably large L' to ensure accuracy of the IAT estimate.

Figure 5.23a demonstrates the method applied using the IAT to inform the length of chain required. In contrast, Figure 5.23b was run for exactly 101 iterations, i.e. $L = L' = 101$, and therefore has no thinning applied. In the samples without thinning, autocorrelation is captured quite clearly in the plots and we observe the anticipated U-shape. Many small errors can be detected in a similar way. Figure 5.24 depicts the effect on the diagnostic histogram of several typical coding errors. It demonstrates that small errors can be easily detected at this number of replications of the procedure, and the shape of the resulting histogram can be a good indication of where to search for an error.

It is to the best of our knowledge unknown to what extent replacing an exact algorithm with an exact-approximate, or pseudo-marginal, alternative affects the performance of the SBC method. To provide some insight into possible effects, we adapt the example above into a pseudo-marginal method. To be precise, we append to the true likelihood term a multiplicative factor, namely a non-negative random variable with expectation 1. In particular, in place of $p(y|\vartheta) = N(y; \vartheta, 8^2)$ we make use of the unbiased estimate $\hat{p}(y|\vartheta) = N(y; \vartheta, 8^2)W$ where, using the shape/scale parameterisation, $W \sim Gamma(1/r, r)$ so that $\mathbb{E}[W] = 1$ and $\mathrm{Var}(W) = r$. In this way, we have a single parameter r through which we can degrade the performance of



(a) *Mixture parameter in the proposal coded as* $\gamma = 0.6$ *and not* $\gamma = 0.4$.

(b) *Likelihood coded with a standard deviation of 4 rather than 8.*

(c) *First mixture component of prior coded with mean 1 and not -1.*

Figure 5.24: *Simulating the effect of a number of common mistakes on the SBC histogram.*

the pseudo-marginal algorithm. With respect to the particle MCMC algorithms presented in this thesis, increasing r is somewhat similar to reducing the number of particles used in the particle filters. Naturally this is a simplification, though the experiment may provide some insight regardless.

Figure 5.25 shows three experiments for a pseudo-marginal algorithm modelled by the process described with $r = 1$, and three similar experiments for a pseudo-marginal algorithm modelled by the process with $r = 8$. In Figures 5.25 (a, b, c), we see that a pseudo-marginal process enjoying a low-variance likelihood estimate passes the calibration process with moderate thinning, to the degree of one in three, or one in five samples. On the other hand, in (d, e, f), we observe that a pseudo-marginal algorithm with a high-variance likelihood estimate still shows signs of autocorrelation with thinning to the degree of one in five samples. After thinning to one in ten samples the autocorrelation appears to have very little effect, if any.

These tests demonstrate that the SBC method can be extremely computationally intensive, especially for high-dimensional or complex target distributions when integrated autocorrelation times are typically much higher. However, it also seems clear that this is unavoidable. If an algorithm requires a large number of iterations (and subsequent thinning) to pass the validation test, then equally it will require a large number of iterations in practice to yield reasonable posterior samples or expectations. In this sense, the SBC approach is not only applicable to validating the correctness of Bayesian algorithms, but in the context of Markov chain Monte Carlo it is a promising convergence diagnostic. Indeed, manipulating the free parameters of an MCMC proposal mechanism, or prior distribution, can have a profound impact on the uniformity of the associated SBC histogram and this provides a useful heuristic for tuning one's sampler. If an algorithm passes the calibration test convincingly, I can have some confidence that it will perform well in practice in a setting similar to that in which it was tested. If it fails to pass the calibration test convincingly, then, whether or not it is correct, it cannot be trusted

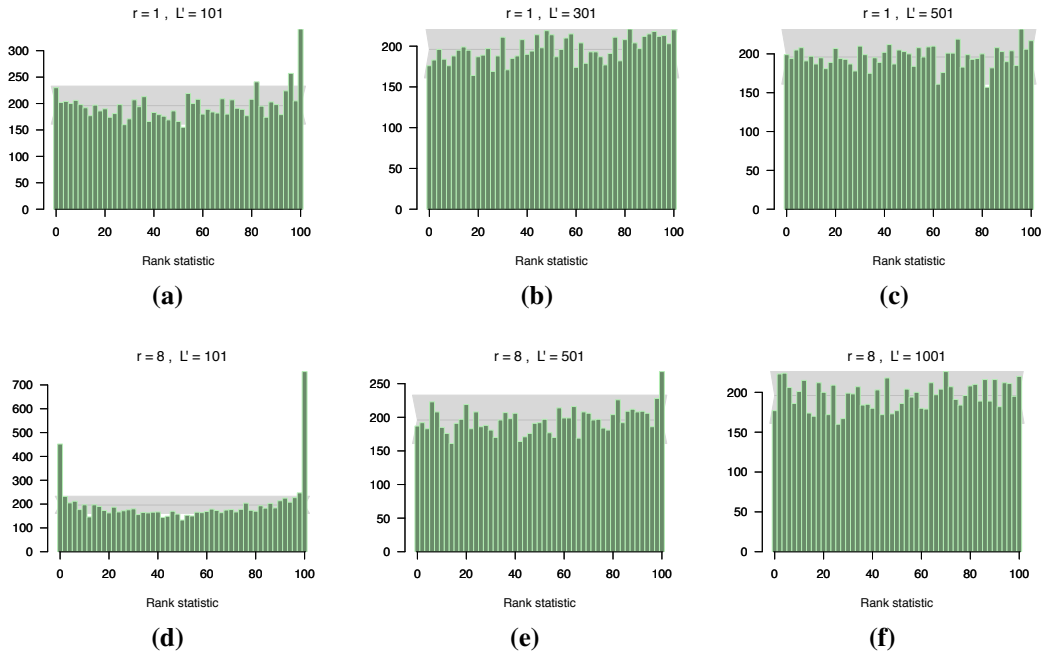


Figure 5.25: *Simulations exploring the effect of pseudo-marginal type approximations on SBC histogram diagnostics.*

at all in practice.

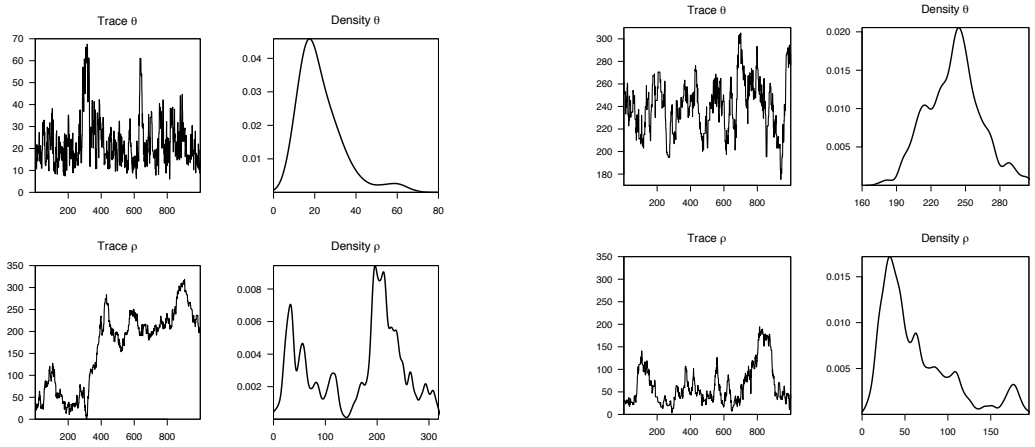
With this preparation in hand, we will make use of this validation procedure throughout the remainder of the thesis.

5.8.2 Two sequence algorithm

First we discuss the performance of the two sequence VRPF (Algorithm 5.3) applied in the particle MCMC setting. For our first experiments we take the prior distribution over the parameters to be $\theta, \rho \sim^{iid} U(0, 400)$, and the proposal density to be $\theta' \sim \mathcal{N}(\theta, \sigma_\theta)$ and $\rho' \sim \mathcal{N}(\rho, \sigma_\rho)$.

Figure 5.26 demonstrates a feature of the SMC' model we believe not to have been discussed in the literature. As seen in Figure 5.26a, when the mutation and recombination rates are similar, that is when there are relatively few observations, the model has a propensity to egregiously overestimate ρ. The effect appears to be a kind of overfitting; when there is very little data to be explained, the model favours an extremely high recombination rate so that every mutation in the sequence will have a recombination event almost tailored to it. Whereas, as in figure 5.26a, when there are a relatively large number of mutations the algorithm favours much more realistic values of ρ. This most probably is due to the combination of a uniform

prior and the relatively flat likelihood of ρ shown in Figure 5.6.



(a) $\vartheta = (\rho, \theta) = (20, 20)$ **(b)** $\vartheta = (\rho, \theta) = (20, 250)$

Figure 5.26: *Posterior trace and density* $\Pr(\rho, \theta | \mathcal{M})$ *where* $\mathcal{M} \sim \Pr_{\vartheta}(\cdot)$*, generated by the particle marginal Metropolis-Hastings algorithm with likelihood estimate provided by the two sequence VRPF.*

Though this behaviour is perhaps not accurately described as overfitting, the parameter ρ has an identifiability issue for certain values of θ. Later we will seek a regularisation solution to the problem.

5.8.3 Prior elicitation

Now we reconsider the issue of prior elicitation. In our first tests (Figure 5.26) we took the parameters to be independent, so that the conditional dependence structure of the pair $\vartheta = (\theta, \rho)$ was described by

$$p(\vartheta) = p(\theta)p(\rho)$$

and $\theta, \rho \sim^{iid} U(0, 400)$. Other choices of prior distribution are possible, as are other dependence structures. Let us now consider more closely the prior distribution for the mutation rate θ. An approach suggests itself that is loosely inspired by so-called empirical Bayes - where hyperparameter values are estimated from the data - but which carries information from the likelihood to the prior, and so is not Bayesian in spirit. We are motivated by the following line of reasoning. If the observed genetic sequences differ for example at 500 SNPs, it is inconceivable that the sequences were generated by the SMC' model with mutation rate $\theta = 30$. Such an outcome would require an implausible concurrence of many independently unlikely events. Intuitively we expect the number of mutations to scale, possibly linearly, with the

mutation rate of the process. This knowledge is highly informative and should therefore be incorporated into the prior distribution of the parameter; it remains to construct a principled method for doing so. Since we know the process by which mutations are generated according to the model, we may generate for a single value of θ a large number of full sets of mutations, say n, which we denote $\mathcal{M}^{\theta}_{1:n}$. Repeating this exercise for a range of values of θ, say $\{\theta_1,\ldots,\theta_{n_\theta}\}$, yields a collection of mutations $\{\mathcal{M}^{\theta_i}_{1:n}\}_{i=1,\ldots,n_\theta}$. Let $|\mathcal{M}^{\theta_i}_j|$ denote the number of SNPs, or mutations, generated in the jth set of mutations generated at rate θ_i. Such an experiment is visualised in Figure 5.27, where $\{\theta_1,\ldots,\theta_{n_\theta}\}=\{10,30,\ldots,990\}$. To each θ_i is associated a boxplot of the lengths $|\mathcal{M}^{\theta_i}_j|$, $j=1:5000$. Among other features, the boxplots represent the median number of SNPs with a black line and the interquartile range (that is, the 25th and 75th percentile) with a grey box. In addition the 15th and 85th percentile are shown in yellow, so that 70% of the sets of mutations generated for a value θ_i have a size (i.e. number of SNPs) that lies between the two yellow lines.

Suppose now that the real sequences one is interested in comprise in total 500 SNPs - this is represented by the solid red line in Figure 5.27. Where this line intersects the yellow lines is visualised by dotted red lines. In this case the dotted red lines (in the top figure) define an interval given approximately by $\Theta=[420,740]$. We can say: for all $\theta\in\Theta$, and for $\rho=100$, sets of mutations 'similar' to those of the real sequences are 'not unlikely' under the SMC' model with parameters (ρ,θ). Here, 'similar' simply means sets of mutations with the same number of SNPs, and since the real sequences lie in the most likely 70% of outcomes for all $\theta\in\Theta$, they are in this sense not unlikely, though for a given $\theta\in\Theta$ there may of course be far more likely outcomes. Notice that when the recombination rate under which the mutations are simulated is increased from $\rho=100$ to $\rho=500$ the number of SNPs seems to concentrate around the median, and the small density peak towards zero diminishes. This has the effect in Figure 5.27 of reducing the interval generated by the above procedure to approximately $\Theta=[435,580]$. In other words, for a fixed θ, as ρ is increased a concentrating behaviour is observed in the number of SNPs generated by the SMC' model with parameters (θ,ρ). A particular case (in effect a cross-section of Figure 5.27) is shown in Figure 5.28. The explanation behind this concentrating behaviour is unclear. A possible explanation is that with a higher recombination rate, any ancestral history generated is more likely to look like an 'average' ancestral history. This follows because with a high recombination rate there are many chances (recombination points) to alter the PDP that describes the ancestral history. On the other hand, when the recombination rate is low one would expect to see a PDP with very sparse jumps, and so perhaps more chance to be 'stuck' in a small or large coalescence time state. Then, once mutations are overlaid on the topology, you are more likely to see very low or very high numbers of mutations.

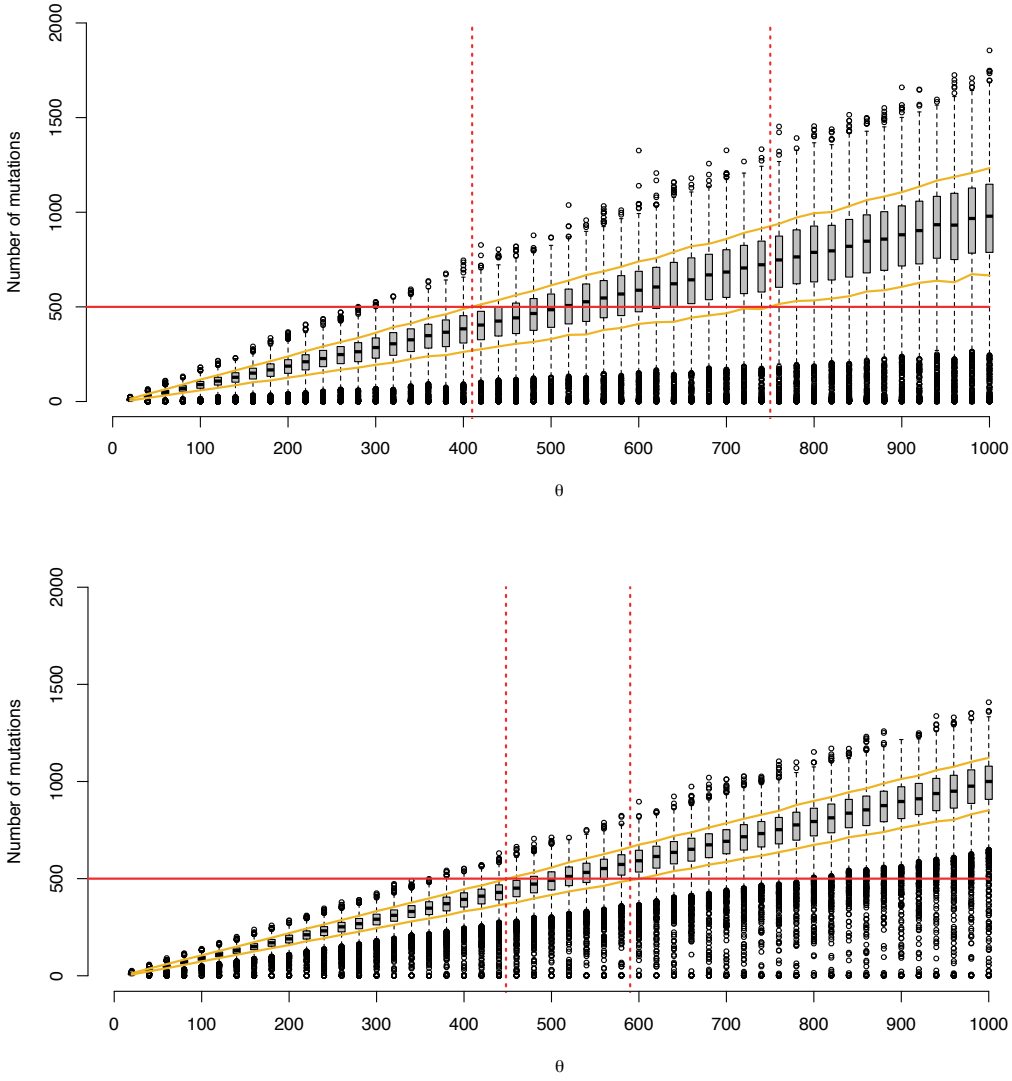


Figure 5.27: *Top:* $\rho = 100$. *Bottom:* $\rho = 500$. *Prior elicitation of* θ*; further explanation is found in the text.*

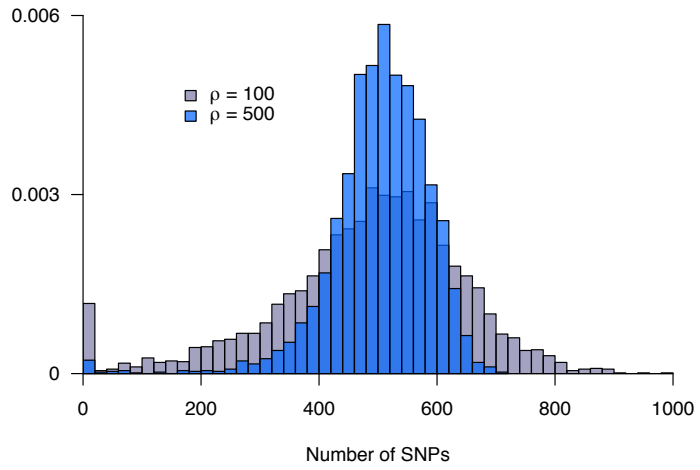


Figure 5.28: *Histograms showing the empirical density of numbers of SNPs generated by the SMC' model when the mutation rate is* $\theta = 520$ *and the recombination rate is* $\rho = 100$ *or* $\rho = 500$.

It is clear from Figures 5.27 and 5.28 that the distribution of the number of SNPs, conditional on the mutation rate θ, is bi-modal. For every value of θ considered there is a peak in the density around the very lowest numbers of SNPs, for example the peak in Figure 5.28 for $\theta \leq 20$. One implication of this feature is that real sequences with very few or no SNPs will be very difficult inference problems since they could be explained by a large range of parameters and so are subject to great uncertainty. This issue is compounded in the low-SNP setting as the bi-modal structure of the density is exacerbated at low values of ρ, and low values of ρ are typically associated with low values of θ.

It may not be prudent to use the interval Θ generated by the above procedure as the support for the prior distribution, for example by using a uniform distribution over this region. We recommend however that the prior distribution used should have a high concentration of probability density in this interval. One simple choice is to use a normal distribution with mean and variance such that, e.g.,

$$\int_{\Theta} p(\theta)\, d\theta \geq 0.8 \tag{5.16}$$

Note that this suggestion is a departure from the spirit of Bayesian statistics; a Bayesian approach here would be to simply use the reasoning above to pick promising initial values for (θ, ρ) in conjunction with a vague prior. The mixing of the Markov chain should not suffer from the choice of a vague prior, provided the chain starts in an area of high likelihood concentration. For example, in Figure 5.30 a normal distribution truncated to the region $(0, \infty)$ is used for both parameters, with its mean chosen to approximately satisfy (5.16).

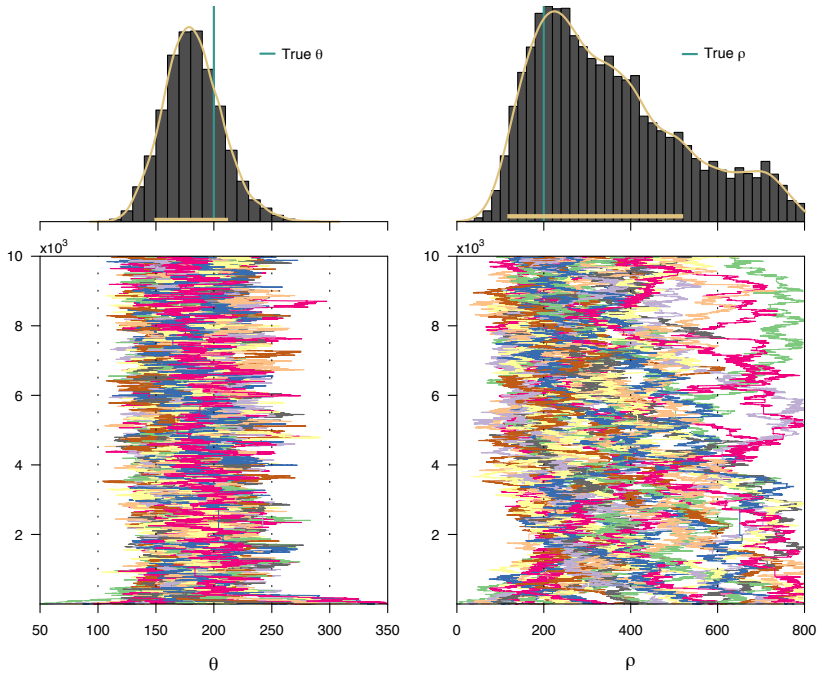


Figure 5.29: *Posterior distribution trace plots and density estimates for the two sequence algorithm. In the particle MCMC, $M = 10^4$ steps were sampled independently 30 times. In addition the θ and ρ proposals follow Gaussian random walks with standard deviations $\sigma_\theta = 6.5$ and $\sigma_\rho = 7.5$. The prior distributions $\theta \sim U(50,350)$ and $\rho \sim U(0,800)$ were used, where $p(\theta,\rho) = p(\theta)p(\rho)$.*

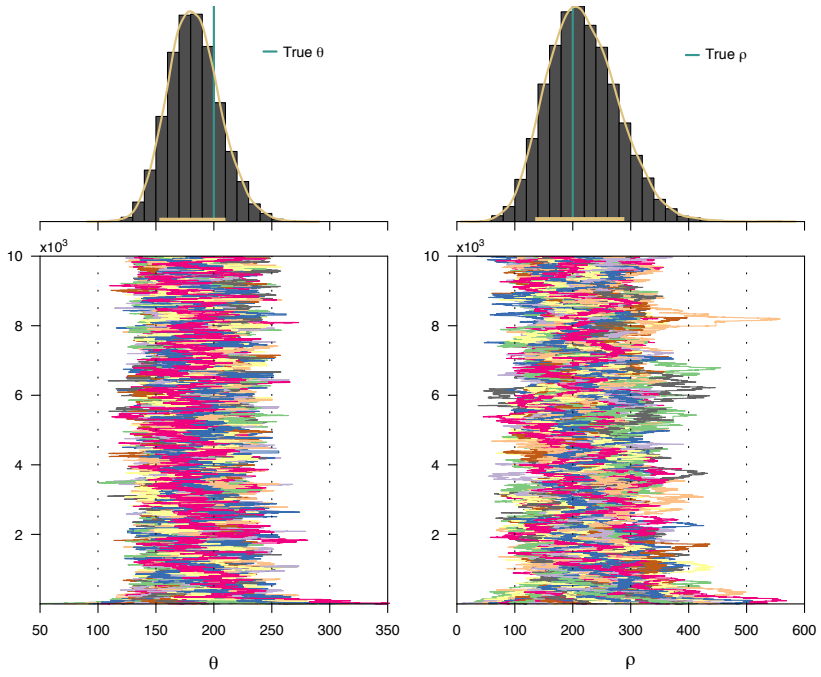


Figure 5.30: *Posterior distribution trace plots and density estimates for the two sequence algorithm. In the particle MCMC, $M = 10^4$ steps were sampled independently 30 times. In addition the θ and ρ proposals follow Gaussian random walks with standard deviations $\sigma_\theta = 6.5$ and $\sigma_\rho = 7.5$. The prior distributions $\theta \sim \mathcal{N}(200, 40)$ truncated to $\theta \in (0, \infty)$, and $\rho \sim \mathcal{N}(\theta, 80)$ truncated to $\rho \in (0, \infty)$, were used. Notice the conditional dependence structure $p(\theta, \rho) = p(\theta)p(\rho|\theta)$.*

Returning to the regularisation issue, it may be appropriate to consider an alternative dependence structure between the two parameters. It is known in biology [46] that in most real world examples θ and ρ are likely to be of a similar magnitude. For this reason we now consider the class of models described by the equation $p(\vartheta) = p(\theta)p(\rho|\theta)$. In particular this allows us to exploit the domain knowledge by featuring in our prior distribution for ρ that it should be in some sense 'close' to θ. We use this approach in Figure 5.30; taking $\theta \sim \mathcal{N}(200, 40)$ truncated to $\theta \in (0, \infty)$, we use conditional on this θ the prior distribution $\rho \sim \mathcal{N}(\theta, 80)$ truncated to $\rho \in (0, \infty)$. A relatively large standard deviation of 80 is used here; the aim is not to entirely constrain ρ, though this standard deviation should be small enough that the parameter is encouraged to converge.

In Figures 5.29 and 5.30 each colour in the trace plots represents an independent particle MCMC run, instantiated from a range of initial positions that span the region shown. The histogram and density estimate plot includes (in gold on the x-axis) an estimate of the 80% credible interval, i.e. the 80% highest posterior density interval, calculated using the R package [87]. Both plots are particle MCMC runs based on a two sequence VRPF (Algorithm 5.3) with 50 epochs and $N = 350$ particles and based on mutation data simulated under the SMC' model (Algorithm 5.1) with parameters $\theta = \rho = 200$. Notice that while the posterior trace and density for θ is fairly similar in both plots, the parameter ρ benefits enormously from reconsidering its prior distribution. In 5.29, where a uniform prior distribution was used, one can see that some Markov chains clearly do not converge to the stationary distribution, but move unconstrained around the parameter space for long stretches. A very different picture emerges in Figure 5.30 when ρ is encouraged by its prior distribution to remain close to θ.

Turning to validation, we now explore the correctness of the algorithm through simulation based calibration, described in Section 5.8.1. In the MCMC runs, the prior distribution structure $p(\theta, \rho) = p(\theta)p(\rho|\theta)$ was used with $\theta \sim \mathcal{N}(100, 60)$ truncated to $\theta \in (0, 200)$, and $\rho \sim \mathcal{N}(\theta, 60)$ truncated to $\rho \in (0, 200)$. Initial values for the Markov chain were chosen uniformly on the same interval. The proposal density for each parameter was a normal random walk with standard deviations 12 and 18 for θ and ρ respectively. We used $N = 256$ particles in each particle filter, which ran over $P = 20$ epochs. Figure 5.31 shows the resulting histograms. Assessing the counts visually, there appears to be an acceptable conformity to the expected behaviour of the uniform distribution. In only two bins does a count exceed the 99% confidence interval, which meets expectations for uniform sampling. It is possible to group together bins when, as is the case here by necessity, there are fewer realisations that one would like. However we find the ample performance of the algorithm on bins covering only a single integer to be encouraging with respect to the power of the test and so the grouping of bins seems unnecessary. The right-most bin on both plots is considerably higher than the

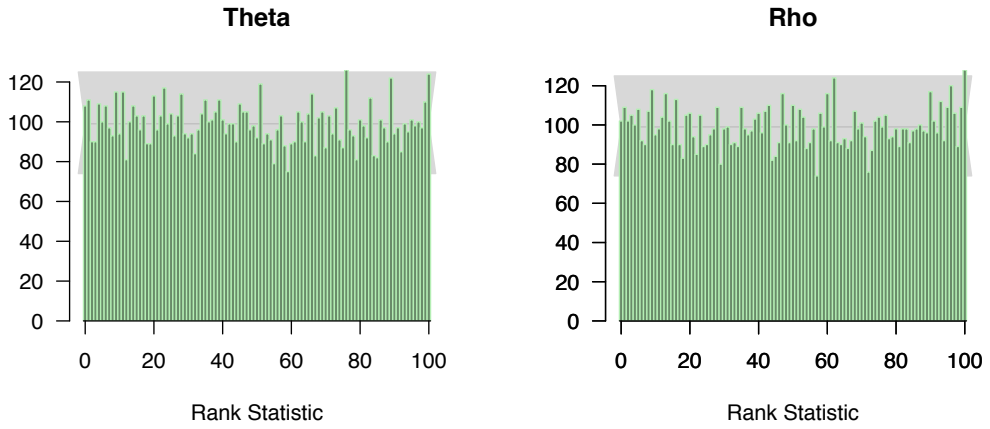


Figure 5.31: *SBC histograms examining the correctness of posterior samples taken from the two-sequence particle filter.*

expected count for the bin, the simplest explanation for which is the presence of some residual autocorrelation in the samples. On the other hand, this may simply be random variation.

Conducting the calibration process in practice for a real inference procedure is deceptively challenging, given the simple specification of the SBC algorithm. We found in practice that while it is relatively easy to obtain a histogram that is fairly uniform in appearance for low numbers of realisations, say 100-500, to preserve this behaviour for a large number of realisations can require a considerable number of MCMC iterations, such as would be infeasible without the use of parallel computation, and a careful choice of prior distribution. This is testament to the strength of the method as a convergence diagnostic. Chains that appeared in trace plots or posterior histograms to have converged often fared badly when executed inside the SBC algorithm. One heuristic we gained from this process was to pick a larger L' in complex settings. The motivation for this is simple. In more complex settings, for example when there is very dense mutation data, the autocorrelations of the chain are likely to be higher. There may be noticeable autocorrelation at lags of hundreds of steps. In this case, if the original run of the chain is too short, one will underestimate the integrated autocorrelation time associated with the chain, indeed the autocorrelations associated with larger lags will not be seen at all and may cumulatively contribute enormously to the integrated autocorrelation time (of an imagined infinitely long chain). Then, when one executes the main run, the IAT of which is compensated for in additional iterations in proportion to this IAT, it will fail to be run for long enough to eliminate the artifacts of autocorrelation. In practice too we found the choice of prior distribution to be subtle. Original experiments carried out with a uniform prior over both parameters proved very difficult to validate by SBC, despite visually appearing to

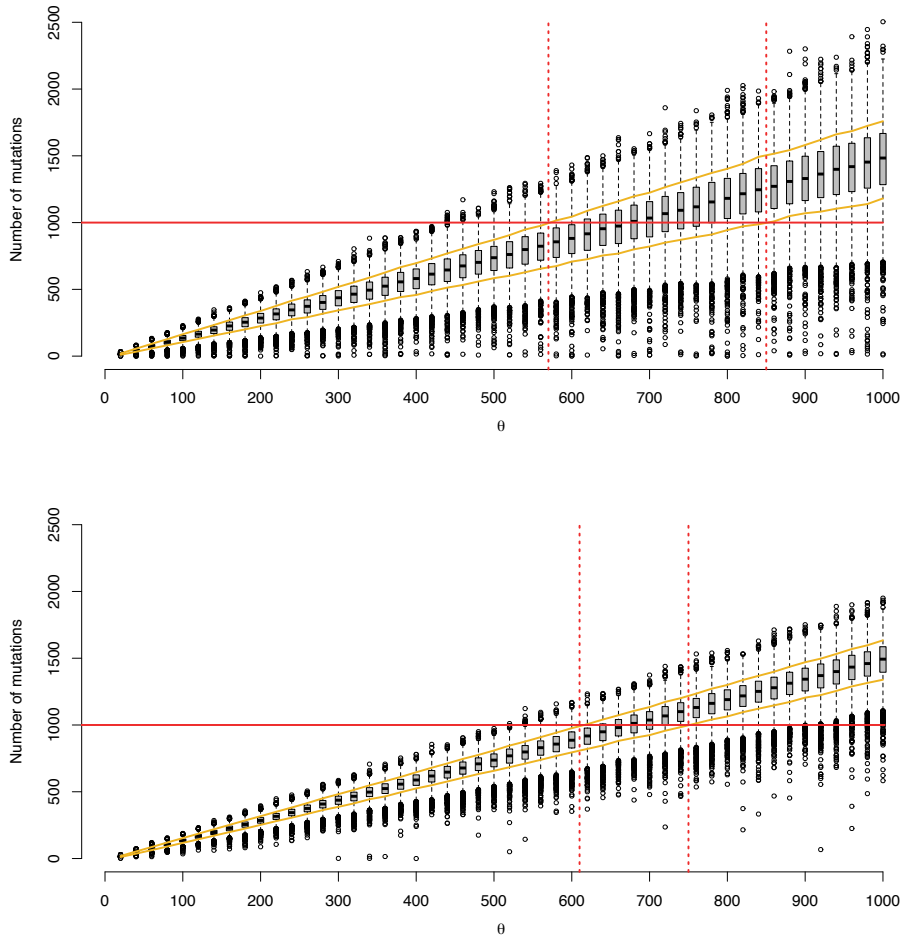


Figure 5.32: *Top:* $\rho = 100$. *Bottom:* $\rho = 500$. *Prior elicitation of* θ*; further explanation is found in the text.*

converge, and so a normal prior was adopted. Even when considering a normal distribution prior, the choice of standard deviation was of course crucial to the success of the algorithm. Too high, and the distribution is effectively similar to a uniform; too low, and the performance of the sampler deteriorates.

5.8.4 Three sequence algorithm

The performance of the three sequence (boosted) algorithm in the context of a particle MCMC algorithm is now briefly discussed. There are many similarities to the two sequence case. First we repeat the prior elicitation experiment for the three sequence case in Figure 5.32. Unsurprisingly the average number of mutation events for a given mutation rate θ is higher than for the two sequence case. In addition, it is considerably less likely that we will observe

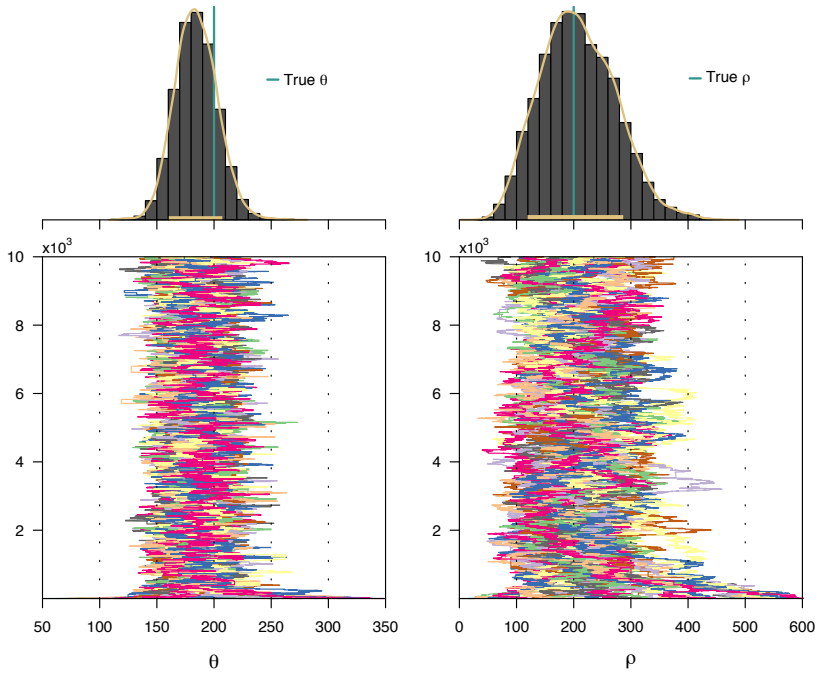


Figure 5.33: *Posterior distribution trace plots and density estimates for the three sequence algorithm. In the particle MCMC, $M = 10^4$ steps were sampled independently 30 times. In addition the θ and ρ proposals follow Gaussian random walks with standard deviations $\sigma_\theta = 6$ and $\sigma_\rho = 6$. The prior distributions $\theta \sim \mathcal{N}(200, 30)$ truncated to $\theta \in (0, \infty)$, and $\rho \sim \mathcal{N}(\theta, 80)$ truncated to $\rho \in (0, \infty)$, were used. Notice again the conditional dependence structure $p(\theta, \rho) = p(\theta)p(\rho|\theta)$.*

a zero-mutation sequence. A similar result is found in Figure 5.33 to Figure 5.30, and they are based on the same data.

We now focus on validating the three sequence algorithm using the simulation based calibration framework (Algorithm 5.8). For these tests the prior distribution structure $p(\theta, \rho) = p(\theta)p(\rho|\theta)$ was used with $\theta \sim \mathcal{N}(50, 20)$ truncated to $\theta \in (0, 100)$, and $\rho \sim \mathcal{N}(\theta, 30)$ truncated to $\rho \in (0, 100)$. Initial values for the Markov chain were chosen uniformly on the same interval. The proposal density for each parameter was a normal random walk with standard deviations 7 and 9 for θ and ρ respectively. We used $N = 512$ particles in each particle filter, which had 31 fixed time steps in addition to those defined by the problematic mutations. The results of the calibration simulation are shown in Figure 5.34.

The histograms summarise 3000 realisations of the Markov chain. More realisations would be preferable but this was not computationally feasible in this case. Note that there are 51 bins in this histogram as a result of pairing neighbouring bins, a practice advocated in [83] for noise reduction, provided the ratio N/B of number of samples to number of bins remains, they suggest, around the value 20. Even with pairing of bins, the resulting histograms are fairly

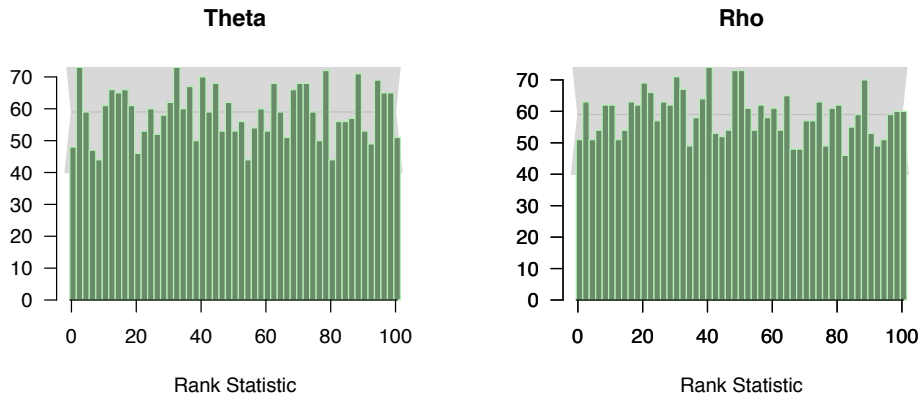


Figure 5.34: *SBC histograms examining the correctness of posterior samples taken from the three-sequence particle filter.*

noisy as a consequence of the low number of realisations. The large variations in bin count indicate that the test may be underpowered. Nevertheless, while the histograms cannot prove correctness, they do not highlight any significant failure in the algorithm. The histograms are within the bounds of uniformity, and show no obvious trend or bias.

Once again, challenges were encountered in carrying out the SBC method. Principally, as before, the choice of the number of iterations in the initial run was an issue. In addition, we encountered the following problem. Though there are predictable signs one may look for in the SBC histograms of posterior autocorrelations, where high autocorrelation is present and where a small number of realisations must be used, it is less clear that one is looking at the signs of autocorrelation, and therefore much time may be wasted on verifying code with no errors. In addition, the choice of prior was again laboured over. This was partly a consequence of the prohibitive cost of running the algorithm for large values of θ. We therefore chose to constrain the support of the parameters to the interval $[0, 100]$. A consequence of truncating the support in this way was that the performance of the algorithm became worse. This was most likely due to using priors with the same variance as for a larger range of parameter values and simply truncating their support. This has the effect of making the prior much flatter on the remaining interval and so we reduced the standard deviation of the prior distributions, which was beneficial for simulation based calibration. Finally, one feature of the approach became quite apparent in this test. After carrying out a short run of the chain, the IAT estimate is calculated and a new chain is run for (typically more) steps. As a result, the run time of the calibration algorithm is highly stochastic. A chain with a sensible initial value and a little luck could be run for twice as long after checking its integrated autocorrelation time. Another chain may need to be run for 100 times longer. One might be tempted to simply end the calibration

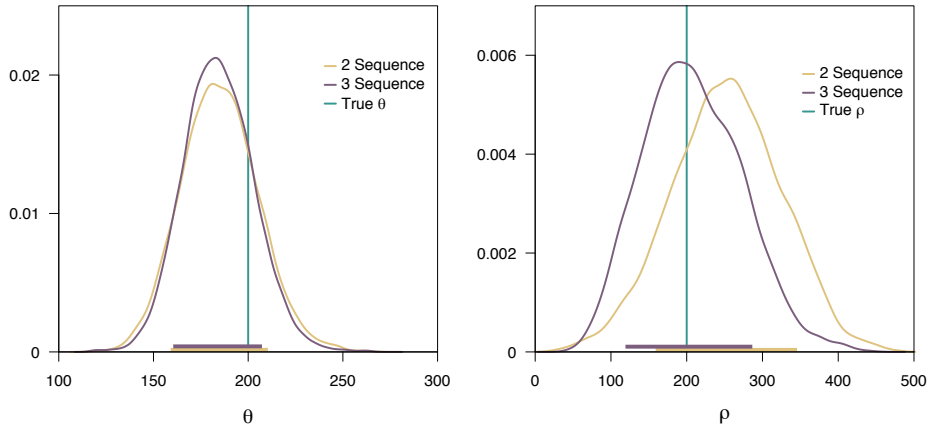


Figure 5.35: *A comparison of the posterior distribution of the parameters* (θ, ρ) *derived from running a two sequence pMMH algorithm, then incorporating an additional sequence from the same population and running a three sequence pMMH algorithm. Notice the reduction in variance of the three sequence algorithm.*

after a finite number of hours, discarding and realisations that have not yet converged, yet this seems unwise and mathematically indefensible as it biases the histograms towards high-performing chains.

5.8.5 An additional sequence

We now consider the difference between the information gain in going from two sequences to three sequences in the same population. More precisely, if one takes three sequences from a population and runs the (boosted, three sequence) particle MCMC, then one removes one of those sequences and runs the (VRPF based) particle MCMC for two sequences, what difference does this make to the posterior distribution of the parameters?

Figure 5.35 shows the posterior distributions derived from such an experiment. For both parameters there is a decrease in the posterior standard deviation. One might expect this to be the case; a priori three sequences contain more information about the genetic history of the population than two, and so the posterior distribution ought to be more 'certain' in its inference. On the other hand, as noted earlier, the strong genetic similarity between members of the same population means there may only be a small gain to be made by considering more sequences. The difference can be quantified by comparing the size of the HPD regions for each approach in each plot of Figure 5.35. Another feature present that one might expect is that the support for the posterior distribution of (θ, ρ) associated with using three sequences lies within the support of the posterior distribution found through using two sequences.

It is possible, and we speculate, that perhaps this concentration of posterior distributions

with the number of sequences implies that with more sequences one could relax the assumptions made in the choice of prior distributions. For instance if, as above, a normal distribution was used for the prior distribution, then the variance of that normal distribution could be raised as the number of samples considered increases. This would provide the practitioner with usefully precise results whilst only requiring of them a realistic level of certainty when eliciting the prior distribution.

Chapter 6

Backward sampling and Metropolis within particle Gibbs

In this chapter an adjustment is made to the particle filter algorithms established in Chapter 5 to allow sampling from smoothed trajectories of the PDPs, performed through so-called backward sampling (introduced in [88]). An application of the method to PDPs is highly non-trivial, though it has been tried in [68]. We derive again the approach for PDPs in full and apply this to the two sequence VRPF discussed earlier. Backwards sampling for PDPs has many potential applications outside of the models considered here, and so is of general methodological interest. We are not aware of any other work applying smoothing approaches to state space inference for coalescent models, and so there is novelty here.

Access to smoothed samples from the posterior distribution facilitates the use of very recent, advanced particle MCMC techniques [5] that scale efficiently in the dimensions of the data compared to traditional approaches. This property is naturally of keen interest to the population geneticist; an algorithm that scales well in dimension could in theory be applied relatively cheaply to a larger region of the genome, a challenge that is impracticable for many common techniques of genomic inference. With further research it is hoped that such benefits could be established for the particular case of the particle filters developed here for SMC'. We present the concept and some computational experiments.

6.1 Smoothing

We now describe Monte Carlo smoothing (which we may also refer to as 'backward sampling') as developed in [88]. Consider a state-space model (SSM) with unobserved, Markovian state $(x_t)_{t\geq 1}$ assumed to evolve according to $x_{t+1} \sim f(x_{t+1}|x_t)$ and observations $(y_t)_{t\geq 1}$

with density $y_{t+1} \sim g(y_{t+1}|x_{t+1})$. We suppose f and g may be non-linear and non-Gaussian but that they can be evaluated pointwise. The joint distribution of states and observations is easily shown to be

$$p(x_{1:t}, y_{1:t}) = f(x_1)g(y_1|x_1)\prod_{i=2}^{t} f(x_i|x_{i-1})g(y_i|x_i)$$

where $x_1 \sim f(\cdot)$ is the initial state distribution. In many state-space problems the object of inference is the so-called *filtering* density $p(x_t|y_{1:t})$. In contrast, *smoothing* aims to estimate densities of the form $p(x_t|y_{1:T})$ for any $t \in [\![1, T]\!]$. To this end, first consider the standard factorisation

$$p(x_{1:T}|y_{1:T}) = p(x_T|y_{1:T}) \prod_{t=1}^{T-1} p(x_t|x_{t+1:T}, y_{1:T}). \tag{6.1}$$

The Markovian structure of the model permits the simplification

$$\begin{aligned} p(x_t|x_{t+1:T}, y_{1:T}) &= p(x_t|x_{t+1}, y_{1:t}) \\ &= \frac{p(x_t|y_{1:t})f(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} \\ &\propto p(x_t|y_{1:t})f(x_{t+1}|x_t). \end{aligned}$$

Running a particle filter forward in time on this SSM would produce at each time step $t \in \{1, \ldots, T\}$ a sequence of weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i \in [\![1,N]\!]}$ approximating the filtering distribution by the empirical distribution

$$p(x_t|y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta_{x_t^{(i)}}(x_t).$$

Therefore a similar approximation may be found to $p(x_t|x_{t+1:T}, y_{1:T})$ through a simple modification of the existing forward-pass weights:

$$w_{t|t+1}^{(i)} := \frac{w_t^{(i)} f(x_{t+1}|x_t^{(i)})}{\sum_{j=1}^{N} w_t^{(j)} f(x_{t+1}|x_t^{(j)})}.$$

Assume a particle filter has been run forwards to obtain a sequence of weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i \in [\![1,N]\!]}$ for $t \in \{1, \ldots, T\}$, then the backward smoothing algorithm is given in Algorithm 6.1. Note that the algorithm outputs a sample approximately drawn from $p(x_{1:T}|y_{1:T})$.

Note that although the exact structure of this algorithm rests on the Markovian nature

Algorithm 6.1 Backward smoothing

Input: A sequence of weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i\in[\![1,N]\!]}$ for $t \in \{1,\dots,T\}$ from a forward run of the particle filter.

1. With probability $w_T^{(i)}$ set $\tilde{x}_T = x_T^{(i)}$.
2. For $t = T-1$ to 1:
   - Compute weights $w_{t|t+1}^{(i)} \propto w_t^{(i)} f(\tilde{x}_{t+1}|x_t^{(i)})$.
   - With probability $w_{t|t+1}^{(i)}$ set $\tilde{x}_t = x_t^{(i)}$.

Output: The sample $\tilde{x}_{1:T} \coloneqq (\tilde{x}_1,\dots,\tilde{x}_T)$

of the state process, this is in general not essential (e.g. [89]) and indeed a more elaborate backwards filter is now presented for the case of particle filters for PDPs. Again consider a deterministic series of times $0 = t_0 < t_1 < \ldots < t_P = 1$, and let $X_n \coloneqq \left(K_n, \tau_{0:K_n}^n, \phi_{0:K_n}^n\right)$ be the events occurring only in epoch n, that is the interval $(t_{n-1}, t_n]$, for $n \in \mathbb{N}\backslash\{0\}$. For any integer $s \in [\![1,P]\!]$ let

$$\mu(s) \coloneqq \min\{l \in [\![s+1,P]\!] : k_l \neq 0\}$$

denote the first epoch after the sth to contain an event, with the convention that $\min\emptyset = P+1$. Similarly, let

$$\nu(s) \coloneqq \max\{l \in [\![1,s-1]\!] : k_l \neq 0\}$$

denote the last epoch before the sth to contain an event, with the convention that $\max\emptyset = 0$. We will use the convention that $X_0 = (K_0 = 0, \tau_0^0 = 0, \phi_0^0)$ and $(\tau_0^n, \phi_0^n) = (t_{n-1}, \phi_{K_{\nu(n)}}^{\nu(n)})$ for all n and a particular realisation of the nth epoch of the process will be denoted x_n. This defines a non-homogenous, non-Markovian system with transitions

$$F_n(x_n|x_{n-1}) \coloneqq \begin{cases} S(\tau_{k_n}^n, t_n; \phi_{k_n}^n) \prod_{i=1}^{k_n} f(\tau_i^n|\tau_{i-1}^n, \phi_{i-1}^n) q(\phi_i^n|\phi_{i-1}^n) & \text{for } k_n > 0 \\ S(t_{n-1}, t_n; \phi_{k_{\nu(n)}}^{\nu(n)}) & \text{for } k_n = 0 \end{cases}$$

Along with the initial distribution

$$M(x_0) \coloneqq M(k_0 = 0, \tau_0^0 = 0, \phi_0^0) = \mathbb{1}_{\{k_0=0\}} \mathbb{1}_{\{\tau_0^0=0\}} q_0(\phi_0^0)$$

the posterior distributions of interest may be written

$$\pi_n(x_{0:n}) \propto g(y_{(0,t_n]}|x_{0:n})M(x_0)\prod_{j=1}^{n} F_j(x_j|x_{j-1}).$$

An application of the memoryless property of the Poisson process shows that for any $\tau \leq t \leq \tau'$ the simplification $S(\tau,t)f(\tau'|t) = f(\tau'|\tau)$ holds. Many such products occur in the posterior distributions of interest, which now admit the following representation

$$\pi_n(x_{0:n}) \propto g(y_{(0,t_n]}|x_{0:n})S(\tau^n_{k_n},t_n;\phi^n_{k_n})M(x_0)\prod_{j=1}^{n}\prod_{i=1}^{k_j} f(\tau^j_i|\tau^j_{i-1},\phi^j_{i-1})q(\phi^j_i|\phi^j_{i-1})$$

where we have replaced the previous convention $\tau^n_0 = t_{n-1}$ by a new convention $\tau^n_0 = \tau^{\nu(n)}_{k_{\nu(n)}}$. In words, under the new convention the 0th transition of an epoch is the last event that happened. The likelihood is assumed to factorise over any interval $[0,T]$ as

$$g(y_{[0,T]}|x_{0:P}) = \prod_{i=1}^{m} g(y_{[\alpha_{i-1},\alpha_i)}|x_{0:P})$$

for any partition $0 = \alpha_0 < \alpha_1 < \ldots < \alpha_m = T$ and $m \in \mathbb{N}\backslash\{0\}$, and for any $0 \leq r < s \leq P$ we have the conditional independence result

$$g(y_{[t_r,t_s)}|x_{0:P}) = g(y_{[t_r,t_s)}|x_{\nu(r+1):s}). \tag{6.2}$$

Note that it is in fact possible to make the further simplification

$$g(y_{[t_r,t_s)}|x_{\nu(r+1):s}) = g(y_{[t_r,t_s)}|x_{r+1:s}).$$

This is because we have established convention that $\phi^{r+1}_0 = \phi^{\nu(r+1)}_{k_{\nu(r+1)}}$ and therefore the information from which the particle 'sets off' at t_r is contained in x_{r+1}. However, this simplification will be omitted hereafter for the sake of clarity.

We now address our specific approach to backwards sampling for PDPs. Recall that we aim to use information from a forward run of a particle filter to sample approximate draws from the smoothing distribution of the latent process. That is, we would like to be able to sample approximately from the posterior distribution $\pi_P(x_{0:P}) := \pi_P(x_{0:P}|y_{(t_0,t_P]})$, where the dependence on the observations y will be suppressed for brevity. In the setting outlined above, access to conditional distributions of the form $\pi_P(x_m|x_{m+1:P})$ for $m \in [\![0, P-1]\!]$ was assumed and these distributions led naturally to the smoothing distribution through the decomposition

6.1. In the PDP setting such conditional distributions are not available. A distinct strategy is as follows: by the definition of conditional probability, for any $m \in [\![0, P-1]\!]$, we have

$$\pi_P(x_{0:P}) = \pi_P(x_{0:m}|x_{m+1:P})\pi_P(x_{m+1:P}) \tag{6.3}$$

where $x_{0:P} = (x_0, x_1, \ldots, x_P)$. If the conditional distribution $\pi_P(x_{0:m}|x_{m+1:P})$ is tractable and can be evaluated up to a normalising constant then it is possible to obtain approximate samples from the posterior distribution. Rather than using a single decomposition, as in equation 6.1, we apply the decomposition 6.3 for each m in sequence. To be precise, suppose a standard forward run of the particle filter is performed and a final state x_P drawn (approximately) from the filtering distribution $\pi_P(x_P)$. Then it is possible to sample a history $x'_{0:P-1}$ according to $\pi_P(x'_{0:P-1}|x_P)$, giving together a particle $x'_{0:P} = (x'_{0:P-1}, x_P)$ approximately distributed according to the smoothing distribution. Most of this particle however is simply taken in order from the history of a particle in the forward pass of the algorithm. A more representative sample from the smoothing distribution is found by applying this idea sequentially, backwards through steps P to 0.

We now derive an expression for the conditional distribution $\pi_P(x_{0:m}|x_{m+1:P})$. Notice that since the m-step model prior can be written

$$F_{0,m}(x_{0:m}) = M(x_0)\prod_{i=1}^{m} F_i(x_i|x_{i-1})$$

it follows that

$$F_{m+1,P}(x_{m+1:P}|x_{0:m}) = \prod_{i=m+1}^{P} F_i(x_i|x_{i-1}).$$

Using the notation $g(y_{[t_i,t_j)}|x_{0:P}) \coloneqq G(t_i,t_j|x_{0:P})$, the smoothing distributions can be expressed as

$$\begin{aligned}
&\pi_P(x_{0:m}|x_{m+1:P}) \\
&\qquad \propto G(0,t_m|x_{0:m})G(t_m,t_P|x_{\nu(m+1):P})M(x_0)F_{0,m}(x_{0:m})F_{m+1,P}(x_{m+1:P}|x_{0:m}) \\
&\qquad \propto \pi_m(x_{0:m})G(t_m,t_P|x_{\nu(m+1):P})F_{m+1,P}(x_{m+1:P}|x_{0:m}).
\end{aligned}$$

Employing identity 6.2, and since $\nu(\mu(m)+1) = \mu(m)$,

$$G(t_m,t_P|x_{\nu(m+1):P}) = G(t_m,t_{\mu(m)}|x_{\nu(m+1):\mu(m)})G(t_{\mu(m)},t_P|x_{\mu(m):P})$$

where $\nu(m+1) < \mu(m)$ by definition. Finally, considering the dependence on $x_{0:m}$, we find

$$\begin{aligned}
&\pi_P(x_{0:m}|x_{m+1:P}) \\
&\quad \propto \pi_m(x_{0:m}) G(t_m, t_P | x_{\nu(m+1):P}) F_{m+1,P}(x_{m+1:P}|x_{0:m}) \\
&\quad \propto \pi_m(x_{0:m}) G(t_m, t_{\mu(m)} | x_{\nu(m+1):\mu(m)}) F_{m+1,P}(x_{m+1:P}|x_{0:m}) \\
&\quad \propto \pi_m(x_{0:m}) G(t_m, t_{\mu(m)} | x_{\nu(m+1):\mu(m)}) \begin{cases} f(\tau_1^{\mu(m)} | t_m, \phi_{k_{\nu(m+1)}}^{\nu(m+1)}) q(\phi_1^{\mu(m)} | \phi_{k_{\nu(m+1)}}^{\nu(m+1)}) & \mu(m) \leq P \\ S(t_m, t_P | \phi_{k_{\nu(m+1)}}^{\nu(m+1)}) & \text{otherwise} \end{cases}
\end{aligned}$$

This suggests the following backwards sampling algorithm, where w_m corresponds to the weights originating from the forward pass of the particle filter.

Algorithm 6.2 Backward sample smoothing for SMC'

Input: A sequence of weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i \in [\![1,N]\!]}$ at each time step $t \in \{0, \ldots, P\}$ from a forward run of the particle filter.

1. Pick $x_P^{(i)}$ with probability $w_P^{(i)}$.

2. Iterate over epochs $m \in \{P-1, \ldots, 0\}$:

   - Compute the normalised weights

$$\begin{aligned}
w_{m|P}^{(j)} \propto w_m^{(j)} G(t_m, t_{\mu(m)} | \tau_1^{\mu(m)}, \phi_{k_{\nu(m+1)}}^{\nu(m+1)}(j)) \\
\times \begin{cases} f(\tau_1^{\mu(m)} | t_m, \phi_{k_{\nu(m+1)}}^{\nu(m+1)}(j)) q(\phi_1^{\mu(m)} | \phi_{k_{\nu(m+1)}}^{\nu(m+1)}(j)) & \mu(m) \leq P \\ S(t_m, t_P | \phi_{k_{\nu(m+1)}}^{\nu(m+1)}(j)) & \text{otherwise} \end{cases}
\end{aligned}$$

   - Sample $i \sim \mathcal{P}(w_{m|P}^{(1)}, \ldots, w_{m|P}^{(N)})$

Output: The reconstructed particle $x_{1:P}$.

Note that the algorithm produces an approximate realisation from $\pi_n(x_{0:n})$. For clarity we now describe how this backward sampling algorithm works in the simplest case, that is $P = 2$, the case of only two intervals. Suppose we run a forward particle filter and $x_2^{(3)}$, the third particle in the Pth interval is chosen by multinomial sampling according to the final weights $w_P^{(i)}$. Then, according to the equations given in Algorithm 6.2, one computes the backwards

weights $w^{(j)}_{P-1|P}$ as follows:

$$\begin{aligned} w^{(j)}_{1|2} &\propto w^{(j)}_1 G(t_1, t_{\mu(1)} | \tau^{\mu(1)}_1, \phi^{\nu(2)}_{k_{\nu(2)}}(j)) \\ &\quad \times \begin{cases} f(\tau^{\mu(1)}_1 | t_1, \phi^{\nu(2)}_{k_{\nu(2)}}(j)) q(\phi^{\mu(1)}_1 | \phi^{\nu(2)}_{k_{\nu(2)}}(j)) & \mu(1) \leq P \\ S(t_1, t_2 | \phi^{\nu(2)}_{k_{\nu(2)}}(j)) & \text{otherwise} \end{cases} \\ &= w^{(j)}_1 G(t_1, t_2 | \tau^2_1(3), \phi^1_{k_1}(j)) \\ &\quad \times \begin{cases} f(\tau^2_1(3) | t_1, \phi^1_{k_1}(j)) q(\phi^2_1(3) | \phi^1_{k_1}(j)) & \mu(1) \leq P \\ S(t_1, t_2 | \phi^1_{k_1}(j)) & \text{otherwise} \end{cases} \end{aligned}$$

where for expository reasons we have included here an explicit dependence on the third particle in the 2nd interval with the notation (3) where appropriate. Using as a reference figure 6.1 it is clear that particle 3 in the second interval undergoes a positive number of recombination events. Therefore the final weighs in this case would be given by

$$w^{(j)}_{1|2} \propto w^{(j)}_1 G(t_1, t_2 | \tau^2_1(3), \phi^1_{k_1}(j)) f(\tau^2_1(3) | t_1, \phi^1_{k_1}(j)) q(\phi^2_1(3) | \phi^1_{k_1}(j)).$$

Notice that for $G(t_1, t_2 | \tau^2_1(3), \phi^1_{k_1}(j))$ we need only calculate the quantity

$$G(t_1, \tau^2_1(3) | \tau^2_1(3), \phi^1_{k_1}(j))$$

i.e. the likelihood from the start of the second interval to the time of the third particle's first event, since from this point the likelihood is the same for every particle considered, regardless of what position it arrived from in the prior interval.

A word of caution may be timely here. Algorithm 6.2 is easy to misunderstand when viewing the SMC' as a collection of lines. Visualising the SMC' process as a PDP was an insight of great benefit to this point, and plots like Figure 6.1 serve as useful heuristics. However now the SMC' process is best thought of as a set of points and not as a step function. The reason this can be misleading is that if one imagines the state x_n carrying all the information about the state in the region $(t_{n-1}, t_n]$ and that this state is represented by a step function stretching from exactly t_{n-1} to t_n, then it is possible to confuse oneself with thoughts like "if I select a new antecedent state x_{n-1} to my now fixed x_n, the lines won't meet at t_n." This confusion never arises if the process is thought of simply as a collection of jump times and heights (with a recipe for visualising a step function from them, if you like).

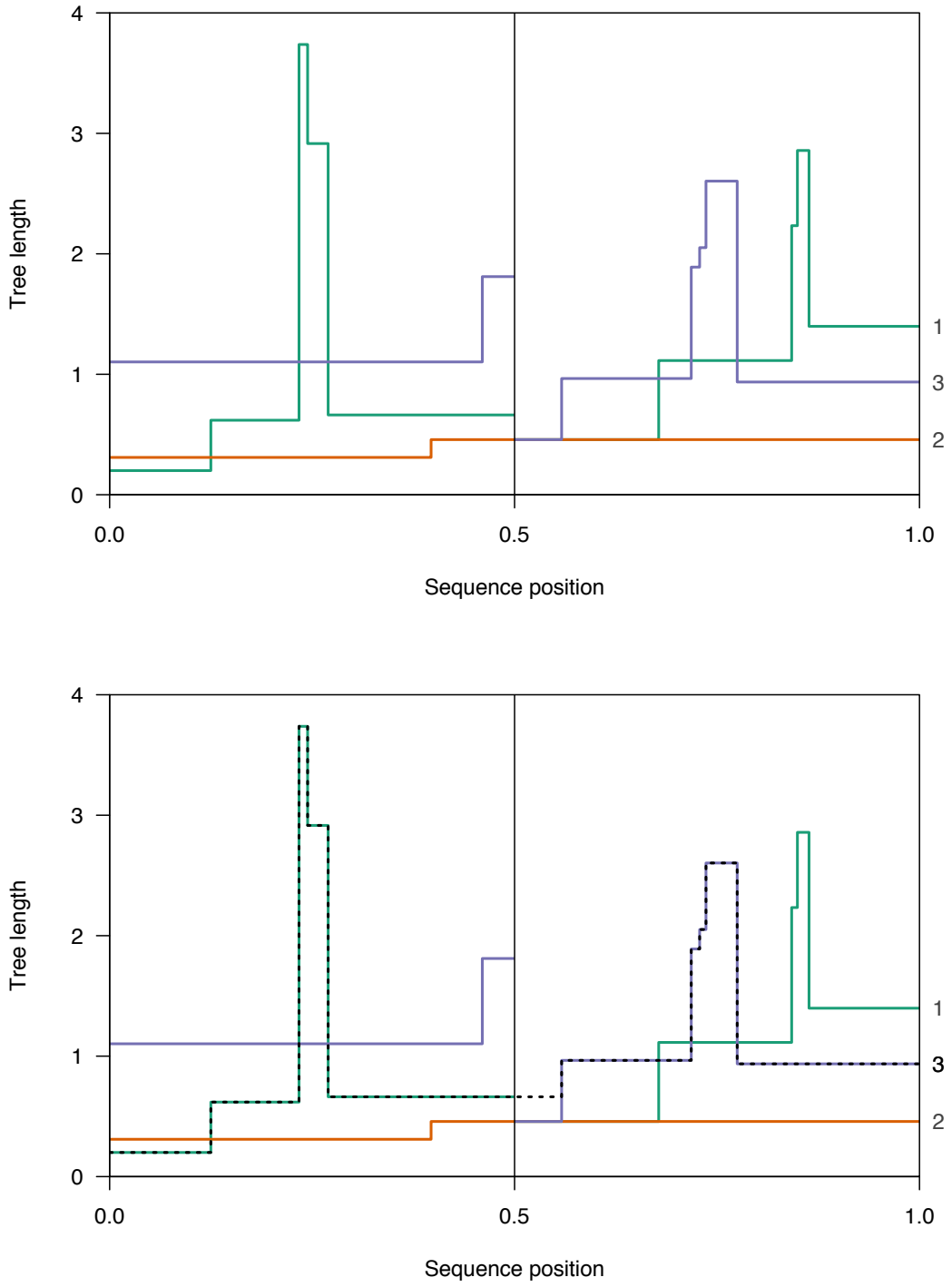


Figure 6.1: *Generating a sample from the smoothing distribution in a simple case; note that the positions of mutations have been omitted. Here, there are $N = 3$ particles and only two intervals, $[0, 0.5]$ and $(0.5, 1]$. On the right hand side the particles are labelled 1,2 and 3 - this is simply the order in which they were generated. The lower plot shows (as a dotted line) an example of a sample generated from the smoothing distribution; notice this sample is not present in the forward paths generated by the particle filter.*

6.2 Estimating the likelihood ratio

A recent advance in particle MCMC methodology is now explored which makes use of the smoothed samples, obtained through backwards sampling, to construct a very efficient sampler. Recall that the acceptance ratio of the Metropolis Hastings algorithm is given for $\vartheta, \vartheta' \in \Theta$ by

$$\alpha(\vartheta, \vartheta') = \min\left\{1, \frac{p(\vartheta')q(\vartheta', \vartheta)}{p(\vartheta)q(\vartheta, \vartheta')}\mathfrak{L}(\vartheta, \vartheta')\right\}$$

where

$$\mathfrak{L}(\vartheta, \vartheta') := \frac{p_{\vartheta'}(\mathcal{M})}{p_{\vartheta}(\mathcal{M})} \tag{6.4}$$

is referred to as the likelihood ratio. In Chapter 5 a pseudo-marginal approach was adopted in which the likelihood ratio was estimated by the ratio of estimates produced by independent particle filters. That is, we estimated the likelihood ratio by $\hat{\mathfrak{L}}(\vartheta, \vartheta') := \hat{p}_{\vartheta'}(\mathcal{M})/\hat{p}_{\vartheta}(\mathcal{M})$ where $\vartheta, \vartheta' \in \Theta$ are points in parameter space, and $\hat{p}_{\vartheta}(\mathcal{M})$ is the positive, unbiased estimate provided by the particle filter of the likelihood of observing mutations $\mathcal{M}$. Note that the estimator $\hat{p}_{\vartheta}(\mathcal{M})$ scales linearly in T, the dimension of the latent process (see e.g. [90]). In contrast, [5] proposes an algorithm in which the likelihood ratio $\mathfrak{L}(\vartheta, \vartheta')$ is itself estimated directly. A key benefit of this method is its efficiency, due to the stability afforded by estimating a stochastic ratio of two random quantities using a unique source of randomness. Moreover it is shown in [5] that it is possible to use the method to design algorithms that scale well in the dimension of the problem. Their approach relies on conditional sequential Monte Carlo (cSMC) [6] in combination with annealed importance sampling (AIS) [91, 92]. We now summarise the derivation of the likelihood ratio estimate.

Suppose the latent process (X_t) takes values in $(\mathsf{X}, \mathcal{X})$. Denote $\pi_{\vartheta}(x_{1:T})$ the posterior distribution of interest given a parameter ϑ. The approach first requires for any $\vartheta, \vartheta' \in \Theta$ the choice of a family of 'bridging' probability distributions $\mathscr{P}_{\vartheta,\vartheta'} = \{\pi_{\vartheta,\vartheta',\varsigma}, \varsigma \in [0,1]\}$ defined on $(\mathsf{X}^T, \mathcal{X}^{\otimes T})$, as well as a non-decreasing function $\varsigma(\cdot) : [0,1] \to [0,1]$ such that $\varsigma(0) = 0$ and $\varsigma(1) = 1$. They are bridging in the sense that they satisfy the end point conditions $\pi_{\vartheta,\vartheta',0}(\cdot) = \pi_{\vartheta}(\cdot)$ and $\pi_{\vartheta,\vartheta',1}(\cdot) = \pi_{\vartheta'}(\cdot)$. Further, we require the following condition on the support of the distributions: for any $A \in \mathcal{X}^{\otimes T}$, $\varsigma, \varsigma' \in [0,1]$ with $\varsigma \leq \varsigma'$ we have

$$\pi_{\vartheta,\vartheta',\varsigma'}(A) > 0 \Rightarrow \pi_{\vartheta,\vartheta',\varsigma}(A) > 0.$$

In addition, define a family of transition probabilities $\mathscr{R}_{\vartheta,\vartheta'} = \{R_{\vartheta,\vartheta',\varsigma}(\cdot,\cdot) : \mathsf{X}^T, \mathcal{X}^{\otimes T} \to [0,1], \varsigma \in [0,1]\}$, such that for any $\varsigma \in [0,1]$, $R_{\vartheta,\vartheta',\varsigma}(\cdot,\cdot)$ leaves $\pi_{\vartheta,\vartheta',\varsigma}(\cdot)$ invariant. Now it is possible to choose a number of bridging distributions K and define the sub family of

distributions

$$\mathscr{P}_{\vartheta,\vartheta',K} \coloneqq \{\pi_{\vartheta,\vartheta',\varsigma(\frac{k}{K+1})}, k \in [\![0, K+1]\!]\} \subset \mathscr{P}_{\vartheta,\vartheta'}$$

with concomitant transitions $\mathscr{R}_{\vartheta,\vartheta'} = \{R_{\vartheta,\vartheta',k}(\cdot,\cdot) : \mathsf{X}^T, \mathcal{X}^{\otimes T} \to [0,1], k \in [\![0, K+1]\!]\}$. Hereafter we write $\pi_{\vartheta,\vartheta',k} = \pi_{\vartheta,\vartheta',\varsigma(\frac{k}{K+1})}$ for simplicity. Setting $\mathbf{u} \coloneqq x_{1:T}$, a Markov chain is defined by $\mathbf{U}_0 \sim \pi_\vartheta$ and for $k \geq 1$, $(\mathbf{U}_k | \mathbf{U}_{k-1} = \mathbf{u}_{k-1}) \sim R_{\vartheta,\vartheta',k}(\mathbf{u}_{k-1}, \cdot)$. Observe that the joint distribution of the chain is given by

$$\pi_\vartheta(u_0) R_{\vartheta,\vartheta',1}(u_0,u_1) \cdots R_{\vartheta,\vartheta',K}(u_{K-1},u_K)$$

so that by taking the integrals with respect to $u_0, u_1, \ldots$ in order it is easy to show

$$\begin{aligned}
&\mathrm{E}\left(\prod_{k=0}^{K} \frac{\pi_{\vartheta,\vartheta',k+1}(\mathbf{U}_k)}{\pi_{\vartheta,\vartheta',k}(\mathbf{U}_k)}\right) \\
&= \int \prod_{k=0}^{K} \frac{\pi_{\vartheta,\vartheta',k+1}(u_k)}{\pi_{\vartheta,\vartheta',k}(u_k)} \times \pi_\vartheta(u_0) R_{\vartheta,\vartheta',1}(u_0,u_1) \cdots R_{\vartheta,\vartheta',K}(u_{K-1},u_K)\, du_0 \cdots du_K \\
&= \int \left[\frac{\pi_\vartheta(u_0)}{\pi_{\vartheta,\vartheta',0}(u_0)} \times \frac{\pi_{\vartheta,\vartheta',1}(u_0) R_{\vartheta,\vartheta',1}(u_0,u_1)}{\pi_{\vartheta,\vartheta',1}(u_1)} \times \right. \\
&\qquad \left. \frac{\pi_{\vartheta,\vartheta',K}(u_{K-1}) R_{\vartheta,\vartheta',K}(u_{K-1},u_K)}{\pi_{\vartheta,\vartheta',K}(u_K)} \times \pi_{\vartheta,\vartheta',K+1}(u_K)\right] du_0 \cdots du_K \\
&= 1.
\end{aligned}$$

Where the normalised density is unavailable, but we have access to pointwise evaluations of an unnormalised version $\gamma_{\vartheta,\vartheta',\varsigma} = Z_{\vartheta,\vartheta',\varsigma} \pi_{\vartheta,\vartheta',\varsigma}$, notice that we have the identity

$$\begin{aligned}
\prod_{k=0}^{K} \frac{\gamma_{\vartheta,\vartheta',k+1}(\mathbf{U}_k)}{\gamma_{\vartheta,\vartheta',k}(\mathbf{U}_k)} &= \prod_{k=0}^{K} \frac{Z_{\vartheta,\vartheta',k+1} \pi_{\vartheta,\vartheta',k+1}(\mathbf{U}_k)}{Z_{\vartheta,\vartheta',k} \pi_{\vartheta,\vartheta',k}(\mathbf{U}_k)} \\
&= \prod_{k=0}^{K} \frac{Z_{\vartheta,\vartheta',k+1}}{Z_{\vartheta,\vartheta',k}} \prod_{k=0}^{K} \frac{\pi_{\vartheta,\vartheta',k+1}(\mathbf{U}_k)}{\pi_{\vartheta,\vartheta',k}(\mathbf{U}_k)} \\
&= \frac{Z_{\vartheta,\vartheta',K+1}}{Z_{\vartheta,\vartheta',0}} \prod_{k=0}^{K} \frac{\pi_{\vartheta,\vartheta',k+1}(\mathbf{U}_k)}{\pi_{\vartheta,\vartheta',k}(\mathbf{U}_k)}
\end{aligned}$$

whereupon it follows that an unbiased estimate of $Z_{\vartheta,\vartheta',K+1}/Z_{\vartheta,\vartheta',0}$ is given by $\prod_{k=0}^{K} \frac{\gamma_{\vartheta,\vartheta',k+1}(\mathbf{U}_k)}{\gamma_{\vartheta,\vartheta',k}(\mathbf{U}_k)}$. In other words, when $\gamma_{\vartheta,\vartheta',0}(x_{1:T}) = p_\vartheta(x_{1:T}, y_{1:T})$ and $\gamma_{\vartheta,\vartheta',1}(x_{1:T}) = p_{\vartheta'}(x_{1:T}, y_{1:T})$, we have a method of obtaining an unbiased estimate $\hat{\mathfrak{L}}(\vartheta,\vartheta')$ of the likelihood ratio $\mathfrak{L}(\vartheta,\vartheta')$ given in Equation (6.4). The AIS algorithm is quoted from [5] in Algorithm 6.3, which to avoid bias should be started from a path $x_{1:T} \sim \pi_\vartheta(\cdot)$. Notice that the variance of the estimate will be

Algorithm 6.3 AIS$(x_{1:P}, P_{\vartheta,\vartheta'}, R_{\vartheta,\vartheta'}, K)$

Input: $x_{1:P}, P_{\vartheta,\vartheta'}, R_{\vartheta,\vartheta'}, K$

1. Set $\mathbf{u}_0 = x_{1:P}$
2. **for** $k = 1, \ldots, K$:
   - Sample $\mathbf{u}_k \sim R_{\vartheta,\vartheta',k}(\mathbf{u}_{k-1}, \cdot)$
3. Compute the estimate

$$\hat{\mathfrak{L}}(\vartheta, \vartheta') = \prod_{k=0}^{K} \frac{\gamma_{\vartheta,\vartheta',k+1}(\mathbf{u}_k)}{\gamma_{\vartheta,\vartheta',k}(\mathbf{u}_k)}$$

Output: $(\hat{\mathfrak{L}}(\vartheta, \vartheta'), \mathbf{u}_K)$.

smaller when ϑ and ϑ' are close, provided the model is sufficiently smooth in the parameter ϑ, since the end point densities $\pi_\vartheta(\cdot)$ and $\pi_{\vartheta'}(\cdot)$ will be 'similar'. Indeed it may be that smaller values of K can be tolerated in this case. We now explain our interest in this context in cSMC algorithms. The cSMC update, proposed in [6], is a sequential Monte Carlo algorithm in which one of the path trajectories is fixed. Its transition kernel is π_ϑ invariant and so it is now often studied in the context of MCMC algorithms [93, 94]. As with all sequential Monte Carlo methods, the resampling within cSMC can cause sample impoverishment or degeneracy. A crucial consequence of this is that for particle paths $x_P^{(i)}$ at time P, while there may be great diversity in the paths at the more recent end of the paths, there will be little diversity across the particles for $x_m^{(i)}$ where m is much less than P. One solution, suggested in [95], involves combining cSMC with backwards sampling - this is given in Algorithm 6.4 for the two sequence SMC' algorithm. Note that BS is chosen to be true when backwards sampling is desired. The cSMC algorithm is known to be reversible with respect to the target density $\pi_\vartheta(\cdot)$ with backwards sampling or without [93, 96]. Finally, the AIS (Algorithm 6.3) with transition kernels given by the cSMC with BS = True is referred to as AIS-cSMC.

6.3 Scalable inference

The ideas presented in this section is largely a summary of others' work. However, the relevance of this new methodological work to the coalescent application is to the best of our knowledge original and we believe of scientific interest.

A central issue with the pseudo-marginal method, or more specifically the particle marginal Metropolis-Hastings (PMMH) algorithm explored in Chapter 2, is that its performance depends heavily on the quality of the likelihood estimate used in the ratio (6.4) in place of the true likelihood. In the context of PMMH this means that often in practice large numbers of particles are required in the filter to guarantee a likelihood estimate of low enough variance. Moreover to maintain a given likelihood estimate variance as the dimension P of the state space model increases, N should increase linearly with P.

Algorithm 6.5 is proposed in [5] as an alternative method for sampling from the joint distribution $\pi(\vartheta, x_{1:P})$. Note that q represents the proposal distribution of the parameter and η the prior distribution. They show that the algorithm is reversible with respect to $\pi(\vartheta, x_{1:P})$ for any $K \geq 0$. In fact, taking $K = 0$ one obtains a reducible algorithm where $x_{1:P}$ is not updated, only ϑ. Following this reasoning, one can imagine (as is done in Gibbs sampling) alternately updating ϑ using Algorithm 6.5 with $K = 0$, then updating $x_{1:P}$ using a cSMC update (Algorithm 6.4). One could use the cSMC with or without backwards sampling for this purpose, though it is shown in [5] experimentally that MCMC-AIS is only efficient when cSMC-BS can be implemented and moreover only where cSMC-BS is an efficient choice. They term this approach Metropolis-within-Particle-Gibbs (MwPG) and its application to SMC' is now explored further. It may be possible, as is the case for models satisfying certain assumptions in [5], that the computational cost of the algorithm as the length of sequence increases could be favourable compared to the PMMH approach. If this is correct, it may follow that large regions of genome could be considered for a small increase in computing time. This is by no means certain but would be an idea well worth pursuing in future research.

We now apply the MwPG to real genetic data sequenced from cichlid fish from Lake Itamba, an isolated crater lake in Tanzania. The data itself comes from the study [97]. For computational ease genetic data only up to the first 200 SNPs was used for these simulations. Figure 6.2 shows the posterior distributions derived from the approach developed in this chapter for two sequences. We note that the convergence of the algorithm is excellent, if only for a short sequence and that for the particular sequences chosen, at least in the region of the sequences considered, the algorithm seems to show fairly confidently that the recombination rate is lower than the mutation rate.

Finally, we address validation of the MwPG algorithm using the simulation based calibration approach described in Section 5.8.1. The results are shown in Figure 6.3, which was generated using particle filters with $N = 256$ particles and $P = 20$ epochs. The prior distribution used for the parameters was $p(\theta, \rho) = p(\rho|\theta)p(\theta)$ where $\theta \sim N(100, 60)$, truncated to $[0, 200]$, and $\rho \sim N(\theta, 60)$, also truncated to $[0, 200]$. A normal random walk was used for the proposal density, with standard deviations 6 and 8 for θ and ρ respectively. Initial values for

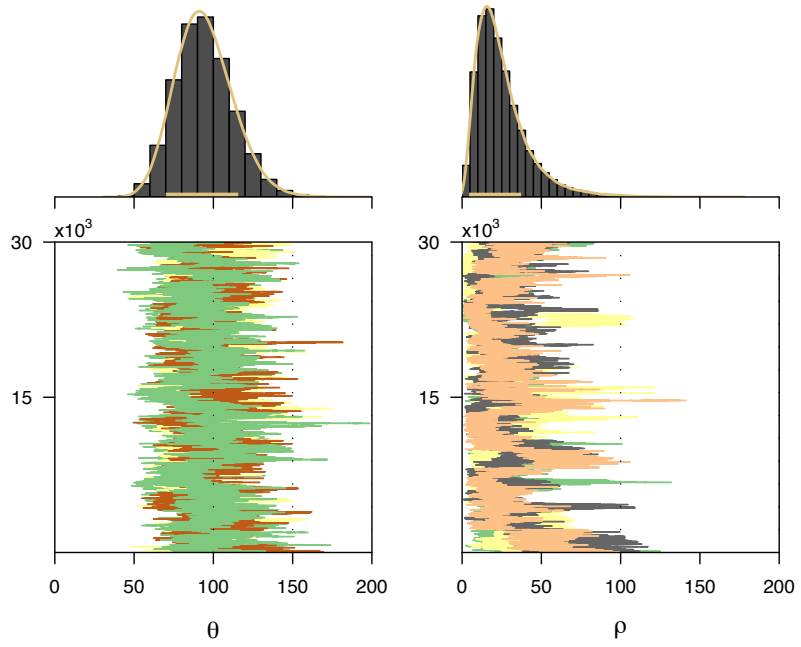


Figure 6.2: *Posterior distribution trace plots and density estimates for the two sequence algorithm on two individuals from the Itamba dataset. $M = 3 \times 10^4$ steps of the algorithm were sampled independently 30 times (of which four are shown). In addition the θ and ρ proposals follow Gaussian random walks with standard deviations $\sigma_\theta = 6.5$ and $\sigma_\rho = 7.5$. The prior distributions $\theta \sim \mathcal{N}(50, 30)$ truncated to $\theta \in (0, \infty)$, and $\rho \sim \mathcal{N}(\theta, 60)$ truncated to $\rho \in (0, \infty)$, were used. Notice again the conditional dependence structure $p(\theta, \rho) = p(\theta)p(\rho|\theta)$.*

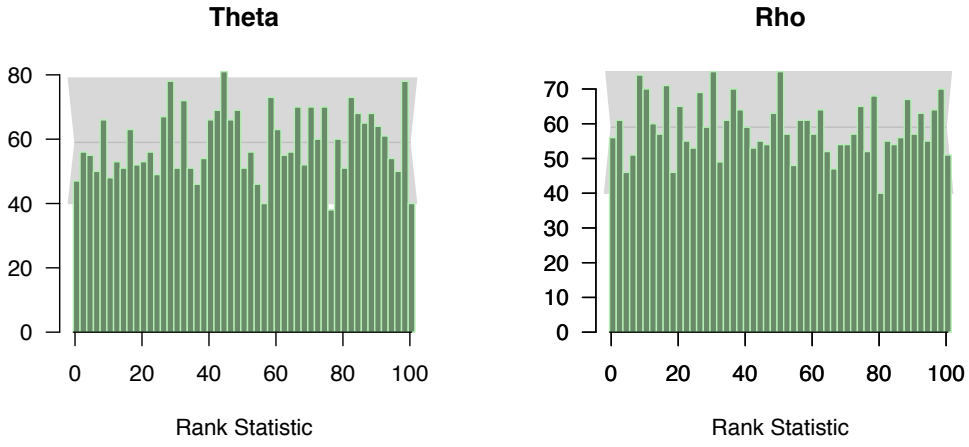


Figure 6.3: *SBC histograms examining the correctness of posterior samples taken from the MwPG algorithm.*

the Markov chain were chosen uniformly at random on the support of the prior distributions.

Appraising the results visually, there appear to be no drastic deviations from uniformity, which is reassuring. Due to computational limits, the test was only run for 3000 realisations, and so it is not surprising that the histograms exhibit fairly large variability even when neighbouring bins have been paired. There is however no clear pattern or trend to be observed in the counts. Therefore we conclude that there is no major issue, for example a large bias, present in the algorithm.

Algorithm 6.4 Two sequence cSMC for SMC'

Input: Number of particles N, number of epochs P, a particle $x_{1:P}$.

A superscript (i) denotes performing the action for all $i \in \{2,\dots,N\}$.

1. Set $z_m^{(1)} = x_m$ for $m \in [\![1,P]\!]$
2. Sample $\phi_0^{(i)} \sim q_0(\cdot)$. Set $z_0^{(i)} \leftarrow (\tau_0 = 0, \phi_0^{(i)})$
3. Set $w_0(z_0^{(i)}) \leftarrow 1$ and $W_0^i \propto w_0(z_0^{(i)})$
4. **for** $m \geq 1$:

   Sample $a_{m-1}^{(i)} \sim \mathcal{P}(W_{m-1}^1,\dots,W_{m-1}^N)$ and $\phi_m^{(i)} \sim q(\cdot|\phi_{m-1}^{a_{m-1}^{(i)}})$, $\tau_m^{(i)} \sim f(\cdot|\tau_{m-1}^{a_{m-1}^{(i)}}, \phi_{m-1}^{a_{m-1}^{(i)}})$

   Set $z_m^{(i)} \leftarrow (\tau_m^{(i)}, \phi_m^{(i)})$

   Compute $w_m(z_{0:m}^{(i)}) = g(y_{(t_{m-1},t_m]}|z_{0:m}^{(i)})$ and $W_m^i \propto w_m(z_{0:m}^{(i)})$
5. Sample $k_P \sim P(W_p^{(1)},\dots,W_P^{(N)})$ and set $x'_P \leftarrow z_P^{(k_P)}$
6. **for** $m = P-1,\dots,1$:

   if $\neg BS$:

   $k_t = a_t^{(k_{t+1})}$

   else:

   for $i \in \{1,\dots,N\}$, compute the forward weight :

$$w_{m|P}^{(i)} \propto w_m^{(i)} G(t_m, t_{\mu(m)}|\tau_1^{\mu(m)}, \phi_{k_{\nu(m+1)}}^{\nu(m+1)}(i)) \times \begin{cases} f(\tau_1^{\mu(m)}|t_m, \phi_{k_{\nu(m+1)}}^{\nu(m+1)}(i)) q(\phi_1^{\mu(m)}|\phi_{k_{\nu(m+1)}}^{\nu(m+1)}(i)) & \mu(m) \leq P \\ S(t_m, t_P|\phi_{k_{\nu(m+1)}}^{\nu(m+1)}(i)) & \text{otherwise} \end{cases}$$

   Sample $k_m \sim P(w_{m|P}^{(1)},,w_{m|P}^{(N)})$

   Set $x'_m \leftarrow z_m^{(k_m)}$

Output: $x'_{1:P}$

Algorithm 6.5 AIS

Input: Current sample $(\vartheta, x_{1:P})$

1. Sample $\vartheta' \sim q(\vartheta, \cdot)$
2. Generate $(x'_{1:P}, \hat{\mathfrak{L}}(\vartheta, \vartheta'))$ according to AIS$(x_{1:P}, P_{\vartheta,\vartheta'}, R_{\vartheta,\vartheta'}, K)$ (Algorithm 6.3)
3. Return $(\vartheta', x'_{1:P})$ with probability

$$\min\left\{1, \frac{q(\vartheta', \vartheta)}{q(\vartheta, \vartheta')} \frac{\eta(\vartheta')}{\eta(\vartheta)} \hat{\mathfrak{L}}(\vartheta, \vartheta')\right\}$$

   otherwise return $(\vartheta, x_{1:P})$.

Output: New sample $(\vartheta', x'_{1:P})$

Chapter 7

Conclusion

The work presented in this thesis has primarily centred around the search for a general and flexible particle Markov chain Monte Carlo method that can be applied to the sequentially Markov coalescent model from population genomics.

In Chapter 2 we explored pseudo-marginal Metropolis-Hastings (PMMH) algorithms [1, 2] and discussed existing strategies to mitigate any poor performance resulting from use of a likelihood estimate of high variance. We demonstrated that a particular strategy, rejuvenation, has desirable properties and we proved that in particular the efficiency of this strategy can be no worse than the original algorithm (and in practice may be substantially better).

In Chapter 3 a review of some topics in discrete spectral theory was presented. We explored the utility of the Dirichlet form as a mechanism for understanding the spectral gap of an operator. Using strategies from the theory of majorisation we proved an ordering on the asymptotic variance and right spectral gap of the iterated importance-sampling rejuvenation kernel.

Chapter 4 comprised a review of some essential population genetics models and their role in the development of coalescent theory, providing the necessary background for the work of Chapter 5, in which a particle MCMC approach was developed. First, we recast the SMC' model as a piecewise deterministic Markov process and studied existing algorithms for such processes. An algorithm termed the *variable rate particle filter* was shown to work well for the two sequence case. For three sequences extensions to existing models were required. We developed an approach termed the *boosted particle filter* that was able to 'look ahead' at upcoming mutations in the sequence and propose particle paths consistent with these mutations, thus 'boosting' the number of particles of positive weight in intervals with problematic mutations. This was shown to reduce considerably the variance of the likelihood estimated when compared to a naive algorithm.

Following this, the particle filter methods were placed in the context of a PMMH in order to

carry out posterior inference on the mutation and recombination rate parameters that generated the model, satisfying our original aim in this project.

In Chapter 6 we derived a backwards sampling approach for the piecewise deterministic Markov processes considered. In turn this allowed us to consider a recent state-of-the-art pseudo-marginal method, an alternative to the PMMH framework that can scale very favourably with the dimension of the problem.

7.1 Future research

A natural question is whether the particle filter approach derived is amenable to generalisation with respect to the number of sequences. Certainly, we believe there is no restriction in principle to the number of sequences that one may consider simultaneously using this technique. In practice, because for three or more sequences an importance sampling proposal will be required (a consequence of the problematic mutations), the challenge in increasing the number of sequences will be in calculating the state transition density. As was evident in Chapter 5, calculating the transition density for three sequences was cumbersome and the problem only grows as the number of sequences increases. However we are optimistic that there may exist strategies to facilitate increasing the number of sequences. For example, since the ancestral recombination graph (ARG) for four individuals contains the ARG for three, it seems natural that there is some kind of recursive structure to coalescent models that could be exploited to find tractable expressions for the transition density more easily. Another possibility is that the process could be automated. As was demonstrated in Chapter 5, the transition density may be written down directly from the description of the coalescent algorithm (SMC' in this case). Once it has been 'transcribed' from its algorithmic form, following through the calculations is mechanical and there exists software for this purpose. It is possible that the calculations following from transcribing the algorithm may contain intractable integrals, precluding such an approach. It is unclear whether this will occur in reality, though it seems unlikely as for the two and three sequence cases almost all the terms in the transition density were exponential functions or convolutions of exponential functions, which are in general straightforward to integrate in many scenarios.

We must also ask whether there is a limit to the number of sequences that it is beneficial to consider. Increasing the dimension of a statistical problem will always incur a penalty of some type, so one must carefully select the number of sequences so that it remains beneficial overall. For coalescent type models it is possible to argue that this number may not be very large. For example, a result from [98] shows that for a simple Wright-Fisher model, considering a sample of i individuals in the present day, the probability that their common ancestor is also

the common ancestor of the whole population t time units in the past is given by

$$e^{-t}\frac{i-1}{i+1}$$

This value is, remarkably, independent of the size of the population and may give us a sense of the size of sample needed for inferences about a sample to hold for the entire population.

i	2	3	4	5	6	7
$\frac{i-1}{i+1}$	0.333	0.5	0.6	0.666	0.714	0.75

We note that it only requires four sequences to be more sure than not that the most recent common ancestor (MRCA) for your sample is the MRCA for the entire population from which they come. Moreover it has long been known [58] that increasing the number of sequences considered is less important than making use of all the information in each sequence. This is because we study members sampled from a single population, who naturally will have very strong correlation in their DNA since they are closely related. In summary, though of course it is desirable to be able to perform (for example) whole population genome studies using coalescent techniques, we must also ask what opportunities are there to make best use of the data one has, as opposed to using more? This is arguably particularly salient for the coalescent as it seems unavoidable that more sequences will entail a considerable increase in computational complexity, even if it is possible.

Finally, it would be particularly interesting in future research to look deeper into the results of Chapter 6, and the possibility that the particle filters developed may possess properties that allow them to benefit from the improved scaling of the annealed importance sampling MCMC approach of [5]. This is of course by no means guaranteed but the asymptotic variance results observed in simulations do suggest some improvement is being conferred.

Bibliography

[1] M. A. Beaumont, "Estimation of Population Growth or Decline in Genetically Monitored Populations," *Genetics*, vol. 164, no. 3, pp. 1139–1160, 2003.

[2] C. Andrieu and G. O. Roberts, "The pseudo-marginal approach for efficient Monte Carlo computations," *Annals of Statistics*, vol. 37, no. 2, pp. 697–725, 2009.

[3] G. A. T. McVean and N. J. Cardin, "Approximating the coalescent with recombination." *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 360, no. 1459, pp. 1387–93, 2005.

[4] P. Marjoram and J. D. Wall, "Fast "coalescent" simulation," *BMC Genetics*, vol. 7, 2006.

[5] S. Yildirim, C. Andrieu, and A. Doucet, "Scalable Monte Carlo inference for state-space models," 2017.

[6] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 72, pp. 269–342, 2010.

[7] C. Andrieu and M. Vihola, "Convergence properties of pseudo-marginal Markov chain Monte Carlo algorithms," *The Annals of Applied Probability*, vol. 25, no. 2, pp. 1030–1077, 2015. [Online]. Available: http://arxiv.org/abs/1210.1484

[8] N. Metropolis, A. Rosenbluth, M. N. Rosenbluth, T. A.H., and E. Teller, "Equations of state calculations by fast computing machine," *Journal of Chemical Physics*, 1953.

[9] W. K. Hastings, "Monte carlo sampling methods using Markov chains and their applications," *Biometrika*, 1970.

[10] L. Tierney, "A note on Metropolis-Hastings kernels for general state spaces," *Annals of Applied Probability*, vol. 8, no. 1, pp. 1–9, 1998.

[11] G. O. Roberts, "Markov chain concepts related to sampling algorithms," in *Markov Chain Monte Carlo in Practice*, W. Gilks, S. Richardson, and D. Spiegelhalter, Eds. Chapman & Hall, 1996, ch. 3, pp. 45–58.

[12] R. core team, "R: A language and environment for statistical computing." *R Foundation for Statistical Computing, Vienna, Austria.*, 2017.

[13] A. D. Sokal, "Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms," *Physics New York*, vol. 34, no. June 1989, 1996.

[14] G. Ambler, D. Hastie, and P. J. Green, "https://github.com/eturro/mmseq/blob/master/src/sokal.cc," 2003. [Online]. Available: https://github.com/eturro/mmseq/blob/master/src/sokal.cc

[15] A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn, "Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator," *Biometrika*, vol. 102, no. 2, pp. 295–313, 2015.

[16] C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal, "On the efficiency of pseudo-marginal random walk Metropolis algorithms," *The Annals of Statistics*, vol. 43, no. 1, pp. 238–275, 2015.

[17] F. J. Medina-Aguayo, A. Lee, and G. O. Roberts, "Stability of Noisy Metropolis-Hastings," mar 2015. [Online]. Available: http://arxiv.org/abs/1503.07066

[18] W. R. Gilks, S. Richardson, and D. Spiegelhalter, "Introducing Markov chain Monte Carlo," in *Markov Chain Monte Carlo in Practice*, W. R. Gilks, S. Richardson, and D. Spiegelhalter, Eds. Chapman & Hall, 1996, ch. 1, pp. 1–20.

[19] J. S. Rosenthal, "Optimal Proposal Distributions and Adaptive MCMC," *Handbook of Markov Chain Monte Carlo*, 2011.

[20] G. O. Roberts, A. Gelman, and W. R. Gilks, "Weak convergence and optimal scaling of random walk Metropolis algorithms," *Ann. Appl. Probab.*, vol. 7, no. 1, pp. 110–120, 1997.

[21] G. O. Roberts and J. S. Rosenthal, "Optimal scaling for various Metropolis-Hastings algorithms," *Statist. Sci.*, vol. 16, no. 4, pp. 351–367, 2001.

[22] J. Besag and P. Green, "Spatial statistics and Bayesian computation," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 55, no. 1, pp. 25–37, 1993.

[23] C. Andrieu and M. Vihola, "Establishing some order amongst exact approximations of MCMCs," p. 20, apr 2014. [Online]. Available: http://arxiv.org/abs/1404.6909

[24] L. Tierney, "Introduction to general state-space Markov chain theory," in *Markov Chain Monte Carlo in Practice*, W. R. Gilks, S. Richardson, and D. Spiegelhalter, Eds. Chapman & Hall, 1996, ch. 4, pp. 59–74.

[25] S. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*, 2nd ed. New York, NY, USA: Cambridge University Press, 2009.

[26] G. O. Roberts, J. S. Rosenthal, and Others, "General state space Markov chains and MCMC algorithms," *Probability Surveys*, vol. 1, pp. 20–71, 2004.

[27] G. O. Roberts and J. S. Rosenthal, "Variance bounding Markov chains," *The Annals of Applied Probability*, pp. 1201–1214, 2008.

[28] L. Bornn, N. Pillai, A. Smith, and D. Woodard, "The Use of a Single Pseudo-Sample in Approximate Bayesian Computation," apr 2014. [Online]. Available: http://arxiv.org/abs/1404.6298

[29] C. Andrieu, "Theoretical and methodological aspects of computations with noisy MCMC algorithms / likelihoods," in *Forthcoming book*, 2015, ch. 1, pp. 1–20.

[30] A. Mira, "Ordering and Improving the Performance of Monte Carlo Markov Chains," *Statist. Sci.*, vol. 16, no. 4, pp. 340–350, 2001.

[31] F. Maire, R. Douc, and J. Olsson, "Comparison of asymptotic variances of inhomogeneous Markov chains with application to Markov chain Monte Carlo methods," *Ann. Statist.*, vol. 42, no. 4, pp. 1483–1510, 2014.

[32] P. Ruckdeschel, M. Kohl, T. Stabla, and F. Camphausen, "S4 Classes for Distributions," *R News*, vol. 6, no. 2, pp. 2–6, may 2006. [Online]. Available: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/distr.pdf

[33] C. Andrieu, "A note on one of the Markov chain Monte Carlo novice's questions," apr 2015. [Online]. Available: http://arxiv.org/abs/1504.03467

[34] D. A. Levin, Y. Peres, and E. L. Wilmer, "Markov Chains and Mixing Times," *Book*, 2009.

[35] J. Rosenthal and P. Rosenthal, "Spectral bounds for certain two-factor non-reversible MCMC algorithms," *Electron. Commun. Probab.*, vol. 20, p. 10 pp., 2015.

[36] D. Rudolf and M. Ullrich, "Positivity of hit-and-run and related algorithms," *Electron. Commun. Probab.*, vol. 18, p. 8 pp., 2013.

[37] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*, ser. Springer Series in Statistics. Springer New York, 2011.

[38] M. Stephens and P. Donnelly, "Inference in molecular population genetics," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2000.

[39] J. Hein, M. H. Schierup, and C. Wiuf, *Gene Genealogies, Variation and Evolution A primer in coalescent theory*, 2003.

[40] R. A. Fisher, *The Genetical Theory of Natural Selection*. Clarendon Press, 1930.

[41] S. Wright, "Evolution in Mendelian populations," *Genetics*, vol. 16, p. 97, 1931.

[42] J. R. Norris, *Markov Chains*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[43] J. F. C. Kingman, "On the Genealogy of Large Populations," *Journal of Applied Probability*, vol. 19, p. 27, 1982.

[44] P. A. P. Moran, "A general theory of the distribution of gene frequencies-I. Overlapping generations," *Proc. R. Soc. Lond. B*, vol. 149, no. 934, pp. 102–112, 1958.

[45] M. Möhle, "Robustness results for the coalescent," *Journal of Applied Probability*, 1998.

[46] M. Nordborg, *Coalescent Theory*, 2000.

[47] D. J. Balding, M. Bishop, and C. Cannings, Eds., *Handbook of Statistical Genetics, Volume 2.*, 2nd ed. Chichester: John Wiley & Sons, 2003.

[48] M. Kimura and J. F. Crow, "The number of alleles that can be maintained in a finite population," *Genetics*, 1964.

[49] M. Kimura, "The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations." *Genetics*, 1969.

[50] T. H. Jukes and C. R. Cantor, "Evolution of Protein Molecules," in *Mammalian Protein Metabolism*, 1969.

[51] J. Stapley, P. G. Feulner, S. E. Johnston, A. W. Santure, and C. M. Smadja, "Recombination: The good, the bad and the variable," *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2017.

[52] R. R. Hudson, "Properties of a neutral allele model with intragenic recombination," *Theoretical Population Biology*, vol. 23, no. 2, pp. 183–201, 1983.

[53] R. C. Griffiths and P. Marjoram, "Ancestral inference from samples of DNA sequences with recombination." *Journal of computational biology : a journal of computational molecular cell biology*, vol. 3, no. 4, pp. 479–502, 1996.

[54] ——, "An ancestral recombination graph," *Progress in population genetics and human evolution (Minneapolis, MN, 1994)*, 1997.

[55] C. Wiuf and J. Hein, "Recombination as a point process along sequences." *Theoretical population biology*, vol. 55, no. 3, pp. 248–59, 1999.

[56] R. C. Griffiths and S. Tavare, "Ancestral Inference in Population Genetics," 1994.

[57] R. Nielsen, "Maximum likelihood estimation of population divergence times and population phylogenies under the infinite sites model," *Theoretical Population Biology*, 1998.

[58] P. Fearnhead and P. Donnelly, "Estimating recombination rates from population genetic data." *Genetics*, 2001.

[59] M. D. Iorio and R. C. Griffiths, "Importance Sampling on Coalescent Histories. I," *Advances in Applied Probability*, vol. 36, no. 2, pp. 417–433, 2004.

[60] M. K. Kuhner, J. Yamato, and J. Felsenstein, "Estimating effective population size and mutation rate from sequence data using metropolis-hastings sampling," *Genetics*, 1995.

[61] M. A. Beaumont, "Detecting population expansion and decline using microsatellites," *Genetics*, 1999.

[62] M. A. Beaumont, W. Zhang, and D. J. Balding, "Approximate Bayesian computation in population genetics," *Genetics*, 2002.

[63] Y. Wang and B. Rannala, "Bayesian inference of fine-scale recombination rates using population genomic data," *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2008.

[64] J. M. Cornuet, F. Santos, M. A. Beaumont, C. P. Robert, J. M. Marin, D. J. Balding, T. Guillemaud, and A. Estoup, "Inferring population history with DIY ABC: A user-friendly approach to approximate Bayesian computation," *Bioinformatics*, 2008.

[65] P. R. Wilton, S. Carmi, and A. Hobolth, "The SMC' is a highly accurate approximation to the ancestral recombination graph," *Genetics*, 2015.

[66] H. Li and R. Durbin, "Inference of human population history from individual whole-genome sequences." *Nature*, vol. 475, no. 7357, pp. 493–496, 2011.

[67] M. D. Rasmussen, M. J. Hubisz, I. Gronau, and A. Siepel, "Genome-Wide Inference of Ancestral Recombination Graphs," *PLoS Genetics*, 2014.

[68] A. Finke, A. M. Johansen, and D. Spanò, "Static-parameter estimation in piecewise deterministic processes using particle Gibbs samplers," in *Annals of the Institute of Statistical Mathematics*, vol. 66, no. 3, 2014, pp. 577–609.

[69] N. Whiteley, A. M. Johansen, and S. Godsill, "Monte carlo filtering of piecewise deterministic processes," *Journal of Computational and Graphical Statistics*, vol. 20, no. 1, pp. 119–139, 2011.

[70] D. S. Dimitrova, V. K. Kaishev, and S. Tan, "Computing the Kolmogorov-Smirnov distribution when the underlying cdf is purely discrete, mixed or continuous," 2017.

[71] ——, *KSgeneral: Computing P-Values of the K-S Test for (Dis)Continuous Null Distribution*, 2018. [Online]. Available: https://cran.r-project.org/package=KSgeneral

[72] D. Pollard, "Beyond the heuristic approach to Kolmogorov-Smirnov theorems," *Journal of Applied Probability*, 1982.

[73] M. H. A. Davis, "Piecewise-Deterministic Markov Processes: A General Class of Non-Diffusion Stochastic Models," *Journal of the Royal Statistical Society. Series B (Methodological)*, 1984.

[74] S. J. Godsill, J. Vermaak, W. Ng, and J. F. Li, "Models and algorithms for tracking of maneuvering objects using variable rate particle filters," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 925–952, 2007.

[75] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later," in *Handbook of Nonlinear Filtering*, 2011, pp. 656–704.

[76] J. D. Hol, T. B. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *NSSPW - Nonlinear Statistical Signal Processing Workshop 2006*, 2006.

[77] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, 2001.

[78] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag New York, 2008.

[79] K. Heine, A. Beskos, A. Jasra, D. Balding, and M. De Iorio, "Bridging trees for posterior inference on ancestral recombination graphs," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2018.

[80] L. Devroye, *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[81] V. Elvira, L. Martino, and C. P. Robert, "Rethinking the Effective Sample Size," *ArXiv e-prints*, sep 2018.

[82] J. Geweke, "Getting it right: Joint distribution tests of posterior simulators," *Journal of the American Statistical Association*, 2004.

[83] S. Talts, M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman, "Validating Bayesian Inference Algorithms with Simulation-Based Calibration," apr 2018. [Online]. Available: http://arxiv.org/abs/1804.06788

[84] S. R. Cook, A. Gelman, and D. B. Rubin, "Validation of software for Bayesian models using posterior quantiles," *Journal of Computational and Graphical Statistics*, 2006.

[85] A. Gelman, "Bigshot statistician keeps publishing papers with errors; is there anything we can do to get him to stop???" 2017. [Online]. Available: https://statmodeling.stat.columbia.edu/2017/08/10/bigshot-statistician-keeps-publishing-papers-errors-anything-can-get-stop/

[86] D. Simpson, "You better check yo self before you wreck yo self," 2018.

[87] Statisticat and LLC., *LaplacesDemon: Complete Environment for Bayesian Inference*, 2018. [Online]. Available: https://web.archive.org/web/20150206004624/http://www.bayesian-inference.com/software

[88] S. J. Godsill, A. Doucet, and M. West, "Monte carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, 2004.

[89] R. Douc, A. Garivier, E. Moulines, and J. Olsson, "On the forward filtering backward smoothing particle approximations of the smoothing distribution in general state spaces models," *arXiv preprint arXiv:0904.0316*, 2009.

[90] F. Cérou, P. Del Moral, and A. Guyader, "A nonasymptotic theorem for unnormalized Feynman Kac particle models," *Ann. Inst. H. Poincaré Probab. Statist.*, vol. 47, no. 3, pp. 629–649, 2011. [Online]. Available: https://doi.org/10.1214/10-AIHP358

[91] G. E. Crooks, "Nonequilibrium Measurements of Free Energy Differences for Microscopically Reversible Markovian Systems," *J. Stat. Phys.*, 1998.

[92] R. M. Neal, "Annealed importance sampling," *Statistics and Computing*, 2001.

[93] N. Chopin and S. S. Singh, "On particle Gibbs sampling," *Bernoulli*, vol. 21, no. 3, pp. 1855–1883, 2015. [Online]. Available: https://doi.org/10.3150/14-BEJ629

[94] C. Andrieu, A. Lee, and M. Vihola, "Uniform ergodicity of the iterated conditional SMC and geometric ergodicity of particle Gibbs samplers," *Bernoulli*, 2018.

[95] N. Whiteley, "Discussion of 'Particle Markov chain Monte Carlo methods' by Andrieu et al." *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 72(3), pp. 306–307, 2010.

[96] F. Lindsten, R. Douc, and E. Moulines, "Uniform Ergodicity of the Particle Gibbs Sampler," *Scandinavian Journal of Statistics*, vol. 42, no. 3, pp. 775–797.

[97] M. Malinsky, R. J. Challis, A. M. Tyers, S. Schiffels, Y. Terai, B. P. Ngatunga, E. A. Miska, R. Durbin, M. J. Genner, and G. F. Turner, "Genomic islands of speciation separate cichlid ecomorphs in an East African crater lake," *Science*, 2015.

[98] I. W. Saunders, S. Tavaré, and G. A. Watterson, "On the Genealogy of Nested Subsamples from a Haploid Population," *Advances in Applied Probability*, vol. 16, no. 3, pp. 471–491, 1984.

Appendix A

Omitted proofs

A.1 Asymptotic variance and integrated autocorrelation time

Recall first that by definition, and using the shorthand $\bar{f}(\Phi_i) \coloneqq f(\Phi_i) - \mu f$,

$$\begin{aligned}\mathrm{Var}_\mu\left(S_M\right) &= M^{-2}\,\mathrm{Var}_\mu\left(\sum_{k=1}^{M} f\left(\Phi_k\right)\right) \\ &= M^{-2}\sum_{k=1}^{M}\mathrm{Var}_\mu\left(f\left(\Phi_k\right)\right) + 2M^{-2}\sum_{1\le i<j\le M}\mathbb{E}_\mu\left[\bar{f}(\Phi_i)\bar{f}(\Phi_j)\right].\end{aligned}$$

Since the Markov chain is assumed to be at stationarity, we have $\mathrm{Var}_\mu\left(f(\Phi_k)\right) = \sigma_f^2$ for all $k \ge 0$, where $\sigma_f^2 \coloneqq \mu f^2 - (\mu f)^2$. Using this stationarity and denoting by ρ_k the correlation coefficient between the random variables $f(\Phi_0)$ and $f(\Phi_k)$, it follows that

$$\mathrm{Var}_\mu\left(S_M\right) = \frac{\sigma_f^2}{M}\left\{1 + \frac{2}{M}\sum_{i=1}^{M-1}\sum_{j=i+1}^{M}\rho_{j-i}\right\} = \frac{\sigma_f^2}{M}\left\{1 + \frac{2}{M}\sum_{i=1}^{M-1}\sum_{k=1}^{M-i}\rho_k\right\}.$$

Reversing the order of summation yields

$$M\,\mathrm{Var}_\mu\left(S_M\right) = \sigma_f^2\left\{1 + \frac{2}{M}\sum_{k=1}^{M}\sum_{i=1}^{M-k}\rho_k\right\} = \sigma_f^2\left\{1 + 2\sum_{k=1}^{M}\left(1 - \frac{k}{M}\right)\rho_k\right\}.$$

Taking the appropriate limit gives the required

$$v\left(f,K\right) \coloneqq \lim_{M\to\infty} M\,\mathrm{Var}_\mu\left(S_M\right) = \sigma_f^2\left\{1 + 2\sum_{k=1}^{\infty}\rho_k\right\}.$$

A.2 Jensen's Inequality

Denote $(l_i)_{i\in[\![1,n]\!]}$ a set of likelihood estimates produced from running a VRPF independently n times. A non-rigorous argument is now provided. As a consequence of Jensen's inequality and the concavity of $log(\cdot)$:

$$\begin{aligned}\overline{log(l)} &:= \frac{1}{n}\sum_{i=1}^{n} log(l_i) \\ &\leq log(\frac{1}{n}\sum_{i=1}^{n} l_i) \\ &=: log(\bar{l}).\end{aligned}$$

Therefore averaging the log-likelihood estimates (like in Figure 5.5) is guaranteed to be upper bounded by the logarithm of the average of the likelihood estimates. Suppose the true likelihood is $l^* = 500$. In order to model unbiased estimates we will use a gamma distribution. Supose $\tilde{l}_i \sim^{iid} \text{Gamma}(k,\gamma)$ so that the probability density function is given by

$$f\left(\tilde{l}_i;k,\gamma\right) = \frac{\tilde{l}_i^{k-1}e^{-\tilde{l}_i/\gamma}}{\gamma^k\Gamma(k)}\mathbb{1}\{\tilde{l}_i>0\}\mathbb{1}\{k>0\}\mathbb{1}\{\gamma>0\}$$

where $\Gamma(\cdot)$ is the gamma function. Notice that taking $k = 1/\gamma$ ensures $\text{E}(\tilde{l}_i) = 1$, and that $\text{Var}(\tilde{l}_i) = \gamma$. We now consider an experiment where an independent set of likelihood estimates $(\tilde{l}_i)_{i\in[\![1,n]\!]}$ derived using the same particle filter are assumed to have distribution $\tilde{l}_i \sim^{iid} \text{Gamma}(1/\gamma,\gamma)$ for parameter values $\gamma \in \{0,2^{-2},2^{-1},2^0,2^1\} = \{0,0.25,0.5,1,2\}$. Now define $l_i = 500 * \tilde{l}_i$. Where we have written $\gamma = 0$ we really mean $l_i = l^*$ for all i. The results of the experiment are shown in Figure A.1. In the left plot are the artificial 'likelihood estimates' and on the right are the corresponding 'log-likelihood estimates'. In other words, each point in the left plot is given by $\frac{1}{n}\sum_{i=1}^{n} l_i$ where $l_i/500 \sim^{iid} \text{Gamma}(1/\gamma,\gamma)$, whereas the right plot shows $\frac{1}{n}\sum_{i=1}^{n} log(l_i)$ where we have used $n = 1\times10^6$. The whole experiment is repeated four times at each value of γ and lines join the first attempt at each value of γ, the second attempt, and so on. Using lines in the first plot gives a sense of the random fluctuation around the mean. Of course, as the likelihood estimate is unbiased, these fluctuations average to roughly the correct value (shown in dotted red). On the other hand, the average of the log-likelihood estimates displays a concave behaviour, tending towards the true value $\log(l^*)$ as the variance of the estimates decreases. This is analogous to increasing the number of particles used in the particle filter. Moreover the four runs of the experiment have given almost identical results on the log scale.

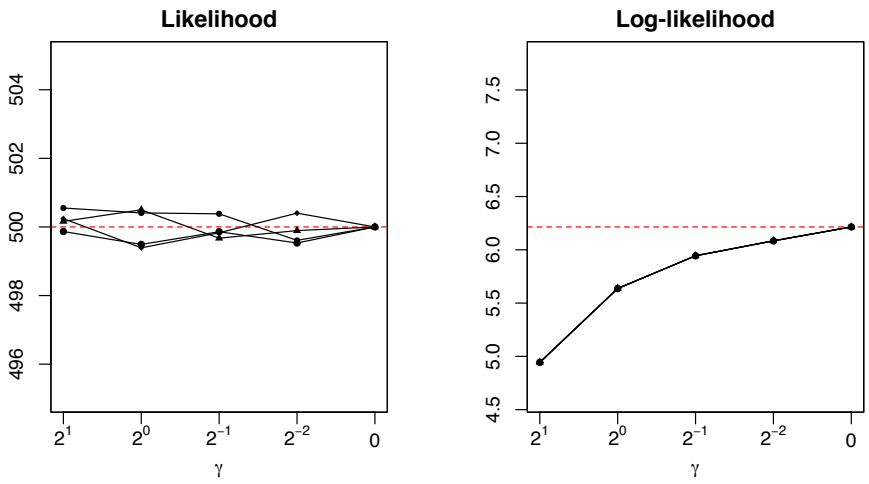


Figure A.1: *A comparison of the likelihood and log-likelihood of an imagined particle filter.*