Perfect simulation for sample-based inference

Jesper Møller[1] and Geoff K. Nicholls[2]

August 4, 1999

[1] *Department of Mathematical Sciences, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark.* Email: jm@math.auc.dk.
[2] *Department of Mathematics, Auckland University, Private Bag 92019 Auckland, New Zealand.* Email: nicholls@math.auckland.ac.nz.

Abstract: Perfect simulation algorithms based on Propp and Wilson (1996) have so far been of limited use for sampling problems of interest in statistics. We specify a new family of perfect sampling algorithms obtained by combining MCMC tempering algorithms with dominated coupling from the past, and demonstrate that our algorithms will be useful for sample based inference. Perfect tempering algorithms are less efficient than the MCMC algorithms on which they typically depend. However, samples returned by perfect tempering are distributed according to the intended distribution, so that these new sampling algorithms do not suffer from the convergence problems of MCMC. Perfect tempering is related to rejection sampling. When rejection sampling has been tried, but has proved impractical, it may be possible to convert the rejection algorithm into a perfect tempering algorithm, with a significant gain in algorithm efficiency.

Keywords: Bayesian inference; Dominated coupling from the past; Exact sampling; Logistic Regression; Markov chain Monte Carlo; Metropolis-Hastings algorithm; Reversible jump MCMC; Simulated sintering; Simulated tempering; Strauss process.

1 Introduction

Reliable general purpose simulation algorithms play an important role in the automation of sample based Bayesian inference. Standard Markov chain Monte Carlo (MCMC) algorithms are very often adequate for this purpose (Gilks et al., 1996). However a sufficient condition, guaranteeing the distributional properties of the output, is not usually available.

Propp and Wilson (1996) give an algorithm for exactly sampling the target distribution of certain MCMC algorithms. Their ideas have been generalised and extended in a number of ways. The resulting algorithms have proven useful in spatial statistics, stochastic geometry and statistical physics (Propp and Wilson, 1996; Häggström et al., 1999; Häggström and Nelander, 1997; Kendall, 1997a; Kendall and Thönnes, 1997; Kendall, 1998; Propp and Wilson, 1998; Kendall and Møller, 1999; Mira et al., 1999; Møller, 1999b; Møller and Schladitz, 1999; Thönnes, 1999). However, they have so far been of limited use for sampling problems of general interest in statistics (Fismen, 1997; Hobert et al., 1998; Murdoch and Green, 1998; Møller, 1999b; Green and Murdoch, 1999). In particular, perfect samplers are least likely to be available for just those sampling problems in which their high cost, in ingenuity and computing, might be overlooked.

The following situation is typical. A MCMC algorithm is given. It is believed to converge efficiently to its intended target distribution, even though that distribution is, say, multi-modal. The sampling problem is difficult, and so some independent check of the MCMC might be called for. However, in order to get verifiably perfect samples, perfect-sampler updates must satisfy a number of conditions (often related to monotonicity). As a result one cannot, in the perfect sampler, use the powerful updates one can use in regular MCMC, and so it does not terminate in any reasonable amount of time.

The family of perfect tempering algorithms we describe may be helpful. They are built around the simulated tempering algorithms of Marinari and Parisi (1992), Geyer and Thompson (1995) and Liu and Sabatti (1998), though in Section 2.1 we take a more general setting for the tempering. Also, one may add to a finished perfect sampler any regular MCMC update, as a fixed-level tempering update, without disturbing the perfect sampler. It is possible to build good quality forward sampling processes within this setting.

The essentials are as follows. The tempering has $N^\star + 1$ levels, labelled $0, 1, \ldots, N^\star$, corresponding to a sequence of artificial distributions which interpolate between the target distribution at level $N^\star$ and a distribution on an atom at level zero. In this way we place the target distribution in a mixture distribution containing an atom. The tempering process is coupled to a second process, a simple random walk on the $0, 1, \ldots, N^\star$ tempering level labels. The random walk is constructed and coupled so that it dominates the tempering process: if the random walk is initialised at a tempering level above the tempering process, then its future level is at or above the level of the tempering process. When the dominating random walk, initialised in level $N^\star$, visits level zero, all coupled tempering processes coalesce in the atom at level zero. Besides determining coalescence, domination is used in Section 2.2 to study the ergodicity properties of the tempering process. These ideas are developed in Section 2.3 where a perfect simulation algorithm is given. Kendall and Møller (1999) use domination in a different fashion toward a similar end.

The property of domination is related to the envelope property of the covering function in rejection sampling. Indeed, one generic family of perfect tempering algorithms is derived using this connection. Such perfect tempering algorithms can be far more efficient than the rejection algorithm to which they are closely related. This relation is made explicit in the final paragraph of Section 3.2.4. The overhead in programme complexity, compared to rejection, need not be large. The efficiency of the algorithms we devise are very sensitive to the relative weighting (the pseudo-prior) of the component distributions in the sequence of tempering distributions. We have an empirically determined recipe for choosing these weights.

Efficiency comparisons of our perfect sampler, rejection and regular MCMC are discussed in Section 3.1.1 and later on in Section 2.3. Since our perfect sampler produces groups of samples, correlated within groups, but independent between groups, these comparisons are a little more complicated than they might otherwise be.

Our three examples include two Bayesian sampling problems. In Section 3.2 we study the Flour Beetle Mortality dataset of Bliss (1935), while Section 3.3 treats a radiocarbon calibration problem. In Section 3.4 we simulate a spatial point process and thereby demonstrate that our perfect simulation algorithm covers cases with a randomly variable dimension under the target distribution. Our perfect simulated tempering algorithm has worked for all the problems on which we have tried it, though in some (unpublished) examples, a fair amount of ingenuity was needed to get a reasonably efficient algorithm. In Section 3.4.2 we study an alternative perfect simulation algorithm, summarise our experience, and mention some improvements which can be made to the basic algorithm.

2 Perfect Simulated Tempering

All densities and distributions referred to below are unnormalised. Let $G(dx)$ be the distribution of a random variable X^G, taking values in a space Ω_G. Here X^G may be of fixed or variable dimension, since we may wish to sample a posterior distribution developed from a fixed model, from a finite mixture of models, or indeed from some general point process. We refer to G as the *target distribution*: it is the distribution we wish to sample.

2.1 Tempering process and coupling construction

We begin by describing a tempering distribution and forward sampling process, following Geyer and Thompson (1995) and Liu and Sabatti (1998). Suppose we choose to have $N^\star \geq 1$ tempering levels. For each $n \in \{0, 1, \ldots, N^\star\}$ we should choose a distribution $H_n(dx)$ and a corresponding level-n space, Ω_n say. We impose a couple of conditions on this sequence of distributions. We assume that $H_{N^\star} = G$, and H_0 is a distribution on a single atom, $\mathbf{0}$ say, so that $\Omega_0 = \{\mathbf{0}\}$. This setting can be extended as explained in Remark 2 at the end of Section 2.3. As a practical consideration, H_1 might be a distribution which can be sampled perfectly by standard means (such as rejection sampling). The distributions

H_n interpolate between a distribution H_1 from which we can obtain samples, and a distribution $H_{N^\star}$ from which we wish to obtain samples. Marinari and Parisi (1992) and Geyer and Thompson (1995) call H_n at small n the "hot" distributions and H_n at large n the "cold" distributions. A few of the many potentially useful tempering transitions are listed in Section 4.2.

We now form a mixture distribution $H(dx, n)$ and a corresponding pair-variable $Z = (X, N)$, where $N \in \{0, 1, \ldots, N^\star\}$ and $X \in \Omega_N$. Let $\pi = (\pi_0, \pi_1, \ldots, \pi_{N^\star})$ where each $\pi_n > 0$ is a constant. Distribution H is defined over a space

$$\Omega = \bigcup_{n=0}^{N^\star} (\Omega_n \times \{n\}),$$

and is given as the π-weighted mixture

$$H(dx, n) = \pi_n H_n(dx)$$

for each $(x, n) \in \Omega$. The constants π_n control the marginal distribution of Z in its tempering level variable N, and are referred to as the *pseudo-prior*. Notice that

$$(X|N = N^\star) \sim G,$$

ie if we sample according to H and then thin so that only those samples with $N = N^\star$ are kept, the resulting samples are distributed according to G, our target distribution.

We next specify a reversible MCMC jump algorithm (Green, 1995) which generates the Markov chain Z_t,

$$Z_t \equiv (X_t, N_t), \qquad t \in \mathbb{Z}$$

with invariant distribution H, which we refer to below as the *tempering process*. For ease of exposition we suppose that just three types of update are applied. When the current state is (x, n), we may propose a move "up" into $(x', n+1)$ with $x' \in \Omega_{n+1}$, or "down" into $(x', n-1)$ with $x' \in \Omega_{n-1}$, or we may propose a "fixed-level" update into (x', n) with $x' \in \Omega_n$. For each state $(x, n) \in \Omega$, let $p \in (0, 1)$ be the probability a move up is proposed, and let $q \in (0, 1-p]$ be the probability a move down is proposed, so that $1 - p - q$ is the probability for a fixed-level update to be proposed (we interpret "move up" as "do nothing" if $n = N^\star$; and similarly "move down" as "do nothing" if $n = 0$). We suppose that $p = p(x, n)$ and $q = q(x, n)$, which might be state functions, are constants and that $p = q$. These assumptions may be removed without difficulty.

The reversible jump MCMC algorithm is a Metropolis-Hastings algorithm specified by a proposal distribution and an acceptance probability as follows. When the current state is (x, n), and a transition to level n' is proposed as described above, let $f_{n,n'}(x, dx')$ be the proposal distribution for candidate states $x' \in \Omega_{n'}$. The proposal distribution $f_{n,n'}(x, dx')$ must satisfy Green's dimension matching condition (Green, 1995), (C1) The distribution $H(dx,n)f_{n,n'}(x,dx')$ has a density

$$\mathcal{F}((x,n),(x',n')) = \frac{H(dx,n)f_{n,n'}(x,dx')}{\xi(dx,n),(dx',n'))},$$

with respect to a symmetric measure $\xi((dx,n),(dx',n'))$ on $\Omega\times\Omega$.

The acceptance probability is now

$$\alpha((x,n),(x',n')) = \min\{1, r((x,n),(x',n'))\}$$

where

$$r((x,n),(x',n')) = \frac{\mathcal{F}((x',n'),(x,n))}{\mathcal{F}((x,n),(x',n'))} \tag{1}$$

is Hastings' ratio. We take $\frac{0}{0}=0$ in Eq. (1). By Green's condition (C1), Z_t will be reversible with respect to H. When we design algorithms, we substitute r into the detailed balance relations, and check reversibility under H without explicitly constructing ξ. A simulation algorithm for Z_t is given in Figure 1.

We will be interested in only those forward simulation processes Z_t which can, in the sense of Section 2.2, be dominated by a coupled random walk, $D_t \in \{0,1,\ldots,N^\star\}$. We impose conditions on Z_t, which will allow Z_t to be dominated:

(C2) Suppose the following conditions are satisfied for all $(x,n)\in\Omega$ with $n<N^\star$ and $f_{n,n+1}(x,\cdot)$-almost all $x'\in\Omega_{n+1}$:
(a) there is symmetry and positivity in Hastings' ratio,

$$r((x,n),(x',n+1)) = 1/r((x',n+1),(x,n)) > 0,$$

(b) there exist upper bounds $K_n \in (0,\infty)$ satisfying

$$K_n \times \frac{\pi_{n+1}}{\pi_n} \geq r((x,n),(x',n+1)).$$

Remark: Taking (C2a) and (C2b) together, we see that acceptance ratios for downward transitions must be bounded away from zero. We impose condition (C2a) in order to avoid giving too much attention to special cases, while condition (C2b) is analogous to local stability for point processes under point addition, used in Geyer (1999) and Kendall and Møller (1999), and related also to the envelope condition of a rejection algorithm. It will sometimes be convenient to relax these conditions. Refer to Remark 1 at the end of Section 2.3.

We can now complete the specification of D_t. We define proposal probabilities, which coincide with those of N_t, and acceptance probabilities

$$\tilde{\alpha}(n,n+1) = \min\left\{1,\ K_n\frac{\pi_{n+1}}{\pi_n}\right\} \tag{2}$$

$$\tilde{\alpha}(n,n-1) = \min\left\{1,\ \frac{1}{K_{n-1}}\frac{\pi_{n-1}}{\pi_n}\right\} \tag{3}$$

$$\tilde{\alpha}(n,n) = 1 \tag{4}$$

for candidate states in the simulation of D_t. The process D_t is a random walk, which moves from n to $n+1$ with probability $p\,\tilde{\alpha}(n, n+1)$ (if $0 \le n < N^\star$) and from n to $n-1$ with probability $q\,\tilde{\alpha}(n, n-1)$ (if $0 < n \le N^\star$); otherwise it stays in n. A simulation algorithm for D_t is given in Figure 2.

2.2 Domination and ergodicity

In this section we couple the two chains Z_t and D_t, define the domination property and show that Z_t is uniformly ergodic. In order to use the general results for Markov chains considered below, we assume, as in (Meyn and Tweedie, 1993), that the σ-field on Ω is separable.

Following Propp and Wilson (1996) and later authors, it is convenient to specify a coupling of Z_t and D_t in terms of a so-called stochastic recursive sequence (SRS). Let U_t^1, U_t^2, $t \in \mathbb{Z}$ be independent $U(0,1)$-variates. We set

$$Z_{t+1} = \mathsf{STupdate}(Z_t; U_t^1, U_t^2), \tag{5}$$

and

$$D_{t+1} = \mathsf{RWupdate}(D_t; U_t^1, U_t^2), \tag{6}$$

where the functions STupdate and RWupdate are given in Figure 1 and Figure 2. Notice that extra random variables are needed to generate candidate states, $x' \sim f_{n,n'}(x, dx')$, in the $Z_t \to Z_{t+1}$-update; these extra random variables play no role in the coupling between STupdate and RWupdate. This is convenient, since it follows that rejection sampling may be used to simulate from $f_{n,n'}(x, dx')$ without otherwise complicating the coupling. Perfect simulation algorithms which simulate Z_t more than once over the same t-values must store and re-use these extra random numbers. This will not be necessary in our case.

Propp and Wilson (1996) give a perfect simulation algorithm in which monotonicity is used to detect coalescence. We give a similar algorithm in which coalescence is determined using a *dominating process*, following Kendall and Møller (1999).

(P1) Using the coupling construction in Eq. (5) and Eq. (6), the D_t process *dominates* the Z_t process in the sense that

$$\Pr\{N_{t+1} \le D_{t+1} | N_t = n, D_t = m\} = 1 \quad \text{whenever } n \le m.$$

This property of algorithms STupdate and RWupdate follows immediately using Eqs. (2) – (4) with condition (C2).

We turn now to the ergodicity properties of the two chains. We consider first D_t, the dominating process. This random walk is ergodic on the finite state space $\{0, 1, \ldots, N^\star\}$. Furthermore, by construction as a Metropolis algorithm, D_t is reversible with invariant distribution $Q^D \equiv (Q_0^D, \ldots, Q_{N^\star}^D)$ where

$$Q_{n+1}^D = Q_n^D\, K_n\, \frac{\pi_{n+1}}{\pi_n}. \tag{7}$$

We refer below to $Q^N \equiv (Q_0^N, \ldots, Q_{N^\star}^N)$ also, the equilibrium distribution for N_t over $\{0, 1, \ldots, N^\star\}$, for which we have in general no closed form.

Consider next the tempering process. Condition (C2) distinguishes Z_t from a general reversible jump MCMC algorithm, for which ergodicity properties cannot in general be established (Green, 1995; Geyer, 1999).

(P2) Tempering process Z_t is uniformly ergodic with equilibrium H.

We now show that property (P2) holds. By construction, Z_t is reversible with respect to H and so H is an invariant distribution for Z_t. By property (P1), for any measurable $A \subseteq \Omega$ and any state $z \in \Omega$,

$$\Pr\{Z_{t+N^\star} \in A | Z_t = z\} \geq \epsilon \mathbb{I}_{(\mathbf{0},0)\in A},$$

where

$$\epsilon = \Pr\{D_{t+N^\star} = 0 | D_t = 0\}.$$

Since D_t is an ergodic random walk, ϵ satisfies $\epsilon > 0$. Hence, in the terminology of Meyn and Tweedie (1993), the state space Ω is a small set for the chain Z_t. This is equivalent to uniform ergodicity, *ie*

$$\sup_{z\in\Omega} || \Pr\{Z_t \in \cdot | Z_0 = z\} - H(\cdot)|| \to 0 \quad \text{as } t \to \infty,$$

where $|| \cdot ||$ denotes the total variation norm (Theorem 16.0.2 in Meyn and Tweedie, 1993).

2.3 Perfect sampling

We now describe how *dominated tempering* may be used to estimate expectations under the target distribution G. First of all, suppose we can initialise Z_t so that $Z_0 \sim H$. It would then be straightforward to estimate expectations under G: Let a fixed integer $L > 0$ be given; simulate Z_t forwards from $t = 0$ to $t = L$ generating a realization $\{(x_t, n_t)\}_{t=0}^L$ of Z_t. Samples x_t in the set $\{x_t; n_t = N^\star, 0 \leq t \leq L\}$ may be used to estimate expectations, since, for real functions f with $\mathsf{E}_G|f| < \infty$,

$$\mathsf{E}_G f = \frac{\mathsf{E}_H\{\sum_{t=0}^L f(X_t)\mathbb{I}_{N_t=N^*}\}}{\mathsf{E}_H\{\sum_{t=0}^L \mathbb{I}_{N_t=N^*}\}}.$$

See Section 3.1.1 and the discussion in Geyer and Thompson (1995) for further information. Note that, if we set $\tau^\star = \inf\{t \geq 0 : N_t = N^\star\}$, the random variable $X_{\tau^\star}$ does not in general follow G. We warn against this curiously attractive error.

The problem then is to generate $Z_0 \sim H$. A perfect simulation algorithm $\mathsf{PWperfect}$ is specified in Figure 3. It works as follows. A realization $\underline{u} = ((u_{-1}^1, u_{-1}^2), (u_{-2}^1, u_{-2}^2), \ldots)$ of the sequence $(U_{-1}^1, U_{-1}^2), (U_{-2}^1, U_{-2}^2), \ldots$ of random number pairs is fixed. Using these fixed random numbers, we make a sequence of simulations of D_t running forwards from negative times. Let $a = 0, 1, 2, \ldots$ label these simulations. The "T" argument of $\mathsf{PWperfect}$ determines how far back the first simulation starts. At each simulation we start one step further back, initialising $D^{(a)}_{-T-a} = N^\star$, and simulating forward using

$$D^{(a)}_{t+1} = \mathsf{RWupdate}(D^{(a)}_t; u^1_t, u^2_t).$$

Simulation "a" runs forward in time until one of three events occurs: (A) the event $D^{(a)}_t = 0$ occurs, or (B) $D^{(a)}_t$ couples with the previous simulation, so that $D^{(a)}_t = D^{(a-1)}_t$, or (C) the simulation reaches $t = 0$ without (A) or (B) occurring. When the simulation is terminated by events (B) or (C), we take a step back in time and repeat the calculation for $D^{(a+1)}_t$ starting from $D^{(a+1)}_{-T-a-1} = N^\star$. When the simulation is terminated by event (A), the algorithm moves to a second phase. Let a_0 be the index of this final sequence, so that a_0 is the smallest $a > 0$ such that (A) occurs. Given a_0, let

$$-\tau_\star = \inf\{-T - a_0 < t \leq 0; D^{(a_0)}_t = 0\}$$

so that $-\tau_\star$ is this first hitting time for the $D^{(a_0)}_t$ simulation. Every tempering path, simulated from times $t \leq -\tau_\star$ using the fixed random numbers $\underline{u}$, is in state $(\mathbf{0}, 0)$ at time $t = -\tau_\star$ (using the domination property (P2), and irrespective of the coupling adopted for the numbers used to simulate candidate states in the Z_t simulation). We therefore set $Z_{-\tau_\star} = (\mathbf{0}, 0)$ and simulate forward from $t = -\tau_\star$ up to $t = -1$, using

$$Z_{t+1} = \mathsf{STupdate}(Z_t; u^1_t, u^2_t).$$

It is the value Z_0 determined by this final forward simulation of Z_t which is the useful sample from H.

> (P3) With probability one the perfect simulated tempering algorithm $\mathsf{PWperfect}$ terminates and returns $\tau_\star < \infty$ and $Z_0 \sim H$.

A proof of the above result follows the proof of Theorem 2 in Propp and Wilson (1996) closely, and is therefore omitted.

Remarks:

1. In some applications it may be useful to relax condition (C2) and observe that the following properties establish (P3): the domination property (P1) holds, 0 is an ergodic atom for D_t, Z_t converges in distribution towards H as $t \to \infty$, and we have the coupling construction of Eqs. (5) and (6). A similar setting is used in Kendall and Møller (1999).

2. The assumption that Ω_0 is a singleton can be relaxed. We can replace Ω_0 by a general measurable set provided that, for example, the tempering process

regenerates whenever it visits $\Omega_0 \times \{0\}$. The simulated tempering chain would then regenerate whenever the dominating random walk has a $0 \to 1$ transition. Since $Z_{-\tau_\star+1}$ does not depend on $Z_{-\tau_\star}$, we can set $Z_{-\tau_\star} = (x, 0)$ with $x \in \Omega_0$ arbitrary. Properties (P1)–(P3) hold as before.

3. A more efficient algorithm is described in Section 4. Our examples use PWperfect as it is more easily implemented from STupdate, the forward tempering update.

We conclude this section with three further points of explanation related to PWperfect. First, we consider the event (B) for the following reason. From the point at which the path of $D_t^{(a+1)}$ couples to the path of $D_t^{(a)}$, the further evolution of $D_t^{(a+1)}$ is given (*ie*, $D_t^{(a+1)} = D_t^{(a)}$ if $D_s^{(a+1)} = D_s^{(a)}$ and $-T - a \leq s < t \leq 0$) and since, in the algorithm, we know that that future evolution does not contain the interesting event (A), we terminate the simulation and step back once again. Readers may find it instructive to sketch possible paths of the $D_t^{(a)}$ processes and observe the further sandwiching property, $D_t^{(a+1)} \leq D_t^{(a)}$ for $-T - a \leq t \leq 0$. Second, for simplicity, we call PWperfect with $T = 1$, but any integer $T \geq 1$ is correct. Third, notice that when a run is terminated by event (B) or (C), PWperfect calls itself with $T \to T+1$. In practice event (B) terminates all but the first few and the last $D_t^{(a)}$-simulations after just a few steps. If event (B) occurs after b steps on average then the number of D_t-update computations is $2b\tau_\star$, or thereabouts. One may replace the main loop of PWperfect by something like

repeat

  $t \leftarrow t + 1$

  $D \leftarrow$ RWupdate$(D; u_t^1, u_t^2)$

  if $t = [T/2]$

    return PWperfect$(\underline{u}, 2T)$

until D=0

(where $[T/2]$ is the greatest integer less than $T/2$, and we call PWperfect$(\underline{u}, T)$ with $T > 0$). The starting time then doubles at each step back, as in Propp and Wilson (1996). Notice that event (B) no longer terminates a forward simulation. If $\tau'_\star$ is the hitting time returned under this doubling rule then $\tau'_\star \geq \tau_\star$ for $\underline{u}$ given: so one step back gives a shorter hitting time. This is worth something when the simulation of Z_t is expensive. However, the doubling rule has its advantages. The number of D_t-updates computed under the doubling rule is not more than $2\tau_\star$ (so long as the simulated path of $D_t^{(a_0)}$ crosses no times $t = 2^d T$, for d positive integer). Also, the code above is simpler than that given in Figure 3.

3 Examples

3.1 Introduction to the examples

3.1.1 Comparing simulation algorithms

We follow Sokal (1989). Suppose we wish to estimate $\mu_f = \mathsf{E}_G f$ for some real function f satisfying $\mathsf{E}_G|f| < \infty$. Let $\hat{\mu}_f$ be the sample mean of f determined from the output of a simulation taking M updates in total (so M is not the number of sample values in the output, but the number of updates used to generate that output - this is explained in more detail below). Let $\sigma_f^2 = \mathrm{var}_G(f)$ be the variance of f in G and let $\sigma_{\mu_f}^2 = \mathrm{var}(\hat{\mu}_f)$ be the variance of our estimate for μ_f. Then $\sigma_f^2/\sigma_{\mu_f}^2$ (the inverse integrated autocorrelation) estimates the number of "effective independent samples" in the output and $\mathcal{E} = \sigma_f^2/[\sigma_{\mu_f}^2 M]$ may be thought of as the number of effective independent samples generated per update of work done (when M is measured in updates). We will call $\mathcal{E}$ the efficiency of the algorithm generating the output under consideration, and use $\mathcal{E}_P$, $\mathcal{E}_R$, and $\mathcal{E}_M$ to denote the efficiency of our perfect simulation, rejection and regular MCMC algorithms respectively. In the estimation of $\sigma_{\mu_f}^2/\sigma_f^2$, Geyer's monotone sequence estimator (Geyer, 1992) was used.

Simple updates, computed rapidly, may reduce estimator variance at a rate which is greater, in CPU-seconds, than sophisticated updates, requiring more time to compute. In such cases it is sensible to measure M in CPU-seconds (on fixed hardware). This, however, compares computer programmes and not algorithms. If one implementation is either clumsy and inefficient, or highly optimised, the comparison is not useful.

We assume it is understood what we mean by "output" and "update" for regular MCMC. In rejection sampling, an update is one rejection or acceptance operation, and the output is the sequence of accepted samples. Consider a sequence of calls to $\mathsf{PWperfect}$. The result of the i'th such call is a sample $Z_0^{(i)}$ say, and a backwards simulation time, $\tau_\star^{(i)}$ say. The sample $Z_0^{(i)}$ is used to initialise simulation of an ordered set of samples $s^{(i)} = \{x_t^{(i)}; n_t^{(i)} = N^\star, 0 \le t \le L\}$. If B such sets are produced then the output is defined to be the sequence of sequences $s = (s^{(1)}, s^{(2)}, \ldots, s^{(B)})$. Let $M_i = \tau_\star^{(i)} + L$ be the number of tempering updates needed to simulate $s^{(i)}$. Then $M = \sum M_i$ is the number of updates used to simulate s. Note that we are counting, in the the cost M, perfect simulations yielding no samples (clearly, $s^{(i)}$ may be empty). We are not allowing for the cost of simulating D_t in M, as D_t-updates are generally trivial by comparison with Z_t-updates. We compensate by presenting CPU-time based efficiencies, in which M is measured in CPU-seconds, including the time to simulate D_t, alongside the update-based efficiency. We gather together our efficiency estimates in Table 1.

3.1.2 Pseudo-priors and forward simulation

How should we specify the pseudo-prior π? These numbers control the distribution of D_t and N_t over $\{0, 1, \ldots, N^\star\}$. They may be set to any positive values without biasing our perfect samples. However, the efficiency, $\mathcal{E}$, just defined, depends on the pseudo-prior, and it is natural to choose π so that independent samples are returned at the highest possible rate. This calculation is beyond our reach. Consider, however, the following.

For $\pi = \pi_D$ specified by $\pi_{D,n+1} = \pi_{D,n}/K_n$, for $n = 0, \ldots, N-1$, the equilibrium distribution Q^D for D is a uniform distribution on $\{0, 1, \ldots, N^\star\}$, by Eq. (7). In this case, D_t soon hits $D_t = 0$, so $\tau_\star$ is small, and a Z_0^{equi}-value is quickly returned by PWperfect. However, downward proposals with $n' = n-1$ in the Z_t simulation of Figure 1 are always accepted as $r((x,n),(x',n-1)) \geq 1$ by condition (C2), so $Q_0^N \gg Q_{N^\star}^N$ is typical under pseudo-prior π_D. Hence, events $N_t = N^\star$ are rare, and we typically get no samples for reasonable simulation lengths L from $t = 0$.

An alternative pseudo-prior, π_N say, makes Q^N uniform. The weighting is $\pi_{N,n} = c/c_n$ where $c > 0$ is an arbitrary constant and

$$c_n = \int_{\Omega_n} H_n(dx). \tag{8}$$

Geyer and Thompson (1995) discuss how the c_n can be estimated up to proportionality using some preliminary forward simulations of the Z_t process (we used their Robbins-Monro estimation). Since H_n is typically more dispersed than H_{n+1}, we generally find $\pi_{N,N^\star} \gg \pi_{N,0}$ and consequently $Q_0^D \ll Q_{N^\star}^D$ under $\pi = \pi_N$ (refer to the last sentence of Section 3.4.1 for an explicit example). This time tempering events with $N_t = N^\star$ are common, so the simulation of G given $Z_0 \sim H$ is quick. However, D_t rarely visits $D_t = 0$, so that $\tau_\star$ can be very large, and the return time of PWperfect is unacceptable.

We found empirically that a geometric average pseudo-prior $\bar{\pi}$, with $\bar{\pi}_n = \sqrt{\pi_{D,n}\pi_{N,n}}$, is a useful starting point, as it balances these two extremes. Geyer and Thompson (1995) estimate the number of distributions and the spacing of the distributions most efficient for forward sampling of the target. We have not attempted to optimise our schedules in this respect.

The efficiency of perfect simulation clearly depends on L. In principle, $\mathcal{E}$, measured as a rate in CPU-time, gives a basis for adjusting L to increase efficiency. In practice we choose L so that the random set $s^{(i)}$ above is non-empty with some probability around one half.

3.1.3 Density notation

In the examples considered later G is given in terms of an unnormalised density $g(x)$ and a measure ν_G of Ω_G, so that

$$G(dx) = g(x)\nu_G(dx).$$

Moreover, for each $n \in \{0, 1, \ldots, N^\star\}$ the distribution H_n has an unnormalised density $h_n(x)$ with respect to some measure $\nu_n(dx)$ of Ω_n, so

$$H_n(dx) = h_n(x)\nu_n(dx).$$

In particular we set $h_0 = 1$, $\nu_0(dx) = \mathbb{I}_{\mathbf{0}\in dx}$, $\Omega_G = \Omega_{N^\star}$, $h_{N^\star}(x) = g(x)$, and $\nu_G = \nu_{N^\star}$. Hence the distribution H has an unnormalised density $h(x,n) = \pi_n h_n(x)$,

$$H(dx, n) = h(x, n)\nu(dx)$$

with respect to the measure ν, defined for all measurable subsets A of Ω by

$$\nu(A) = \sum_{n=0}^{N^\star} \int \mathbb{I}_{(x,n)\in A}\nu_n(dx).$$

Here 'measurable' means with regard to some suitable σ-field on Ω, which is typically generated by some σ-fields defined for each space Ω_n, $n = 0, 1, \ldots, N^\star$ — these measure theoretical details are rather obvious and left out in the examples considered below.

3.2 Flour Beetle Mortality dataset

3.2.1 Introduction

Carlin and Louis (1996) describe the following sampling problem. It arises from the data set containing $K = 8$ items, shown in Table 2, extracted from Bliss (1935). We abbreviate the motivation. In terms of a three-component parameter vector $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ and data ω, y and a given above, the unnormalised posterior (target) density of x is

$$g(x) = P(x)\prod_{i=1}^{K} \ell(\omega_i, y_i, a_i|x).$$

Here

$$\ell(\omega_i, y_i, a_i|x) = \mathcal{I}(\omega_i, x)^{y_i}[1 + \mathcal{I}(\omega_i, x)]^{a_i - y_i},$$

is a likelihood factor,

$$\mathcal{I}(\omega_i, x) = \left\{ \frac{\exp(\frac{\omega_i - x_1}{\exp(x_2)})}{1 + \exp(\frac{\omega_i - x_1}{\exp(x_2)})} \right\}^{\exp(x_3)}$$

is a threshold function,

$$P(x) = \exp(\tilde{a}_0 x_3 - 2e_0 x_2) \exp\left[-\frac{1}{2} \left(\frac{x_1 - c_0}{d_0} \right)^2 - b_0 \exp(x_3) - f_0 \exp(-2x_2) \right]$$

is an unnormalised prior density and

$$\tilde{a}_0 = 0.25,\ b_0 = 0.25,\ c_0 = 2,\ d_0 = \sqrt{10},\ e_0 = 2.000004,\ f_0 = 0.001$$

are constants. Here $\nu_G(dx) = dx_1 dx_2 dx_3$, with dx_i Lebesgue measure of $\mathbb{R}$.

In this problem x are related to parameters (μ, σ, m) as $x_1 = \mu$, $\exp(x_2) = \sigma$ and $\exp(x_3) = m$, and the form of the prior density $P(x)$ reflects a choice of independent priors $\mu \sim \mathsf{N}(c_0, d_0^2)$, $\sigma^2 \sim \mathrm{I}\Gamma(e_0, f_0)$ (where $\mathrm{I}\Gamma$ is the inverse Gamma distribution) and $m \sim \Gamma(\tilde{a}_0, b_0)$, along with a Jacobian factor.

3.2.2 Tempering sequence

We now specify the tempering sequence. In this problem we use a sequence of densities distinguished by the value of the inverse temperature β_n in a tempering schedule $\beta = (\beta_1, \ldots, \beta_{N^\star})$. Here $\beta_{i+1} > \beta_i$ and $\beta_{N^\star} = 1$. In choosing h_n, an important observation is that $\ell(\omega_i, y_i, a_i|x)$ reaches its maximum value at

$$\ell_i^* \equiv (y_i/a_i)^{y_i} (1 - y_i/a_i)^{a_i - y_i},$$

as would be the case in a regular logistic regression. We therefore take as our sequence of tempering densities

$$h_n(x) = \left[\prod_{i=1}^{K} \ell(\omega_i, y_i, a_i|x)/\ell_i^* \right]^{\beta_n} \times P(x), \qquad \Omega_n = \mathbf{R}^3, \qquad \nu_n = \nu_G$$

for $n = 1, 2, \ldots, N^\star$. When we change the tempering level we do not otherwise change the state, *ie* for $|n - n'| = 1$ we let $f_{n,n'}(x, dx') = \mathbb{I}_{x \in dx'}$. We therefore have

$$r((x, n), (x, n+1)) = \frac{\pi_{n+1}}{\pi_n} \times \left[\prod_{i=1}^{K} \ell(\omega_i, y_i, a_i|x)/\ell_i^* \right]^{\beta_{n+1} - \beta_n}$$

for $1 \le n < N^\star$. If a transition from $n = 0$ to $n = 1$ is proposed then we sample the proposal $x' \sim H_1$ directly by sampling the prior and rejecting with remaining factors. We have $r((\mathbf{0}, 0), (x', 1)) = \pi_1/\pi_0$. Condition (C2a) is thereby satisfied. Because $r((\mathbf{0}, 0), (x', 1))$ is independent of x and x', and the π_n may be set to any positive values without biasing our perfect samples, we are free to set $\pi_0 = \pi_1$ and so obtain $r((\mathbf{0}, 0), (x', 1)) = 1$. Since $\ell(\omega_i, y_i, a_i|x) \le \ell_i^*$ and $\beta_{n+1} > \beta_n$, condition (C2b) is satisfied under the choice $K_n = 1$ for each $n = 0, 1, \ldots, N^\star - 1$.

3.2.3 Regular MCMC

In this section we describe a regular MCMC update for $g(x)$. We use the same type of update at fixed level $n = n' > 0$ in the tempering algorithm $\mathsf{STupdate}$ of Figure 1.

Carlin and Louis (1996) note that there is no closed form available for any of the three posterior conditional distributions needed to implement a Gibbs sampler for g. They go on to show that a Metropolis random walk algorithm, with proposal covariance matrix

$$\Sigma = \begin{bmatrix} 0.000292 & -0.003546 & -0.007856 \\ -0.003546 & 0.074733 & 0.117809 \\ -0.007856 & 0.117809 & 0.241551 \end{bmatrix}$$

gives an effective sampling process for g.

Suppose (x, n) is the current state and that $p < u_t^1 < 1 - q$ in $\mathsf{STupdate}$. Three independent standard normal variates $w = (w^1, w^2, w^3)$, $w^i \sim \mathsf{N}(0, 1)$, are generated and used to determine a candidate state, $x' = x + w\mathtt{chol}(\Sigma)$ where $\mathtt{chol}(\Sigma)$ is the Cholesky matrix of Σ. The candidate is accepted with probability $\min\{1, h_n(x')/h_n(x)\}$ at level n of our tempering. Set $h_n = g(x)$ to obtain the acceptance probability for the regular MCMC update.

3.2.4 Results from experiments

After a little experimentation we found a tempering schedule with $N^\star = 3$, $\beta_1 = 0$, $\beta_2 = 0.06$ and $\beta_3 = 1$ to be reasonably effective. We set $p = q = 1/3$ throughout our simulations as the tempering is less efficient when fixed level updates are excluded. Constants π_N, which give N_t a uniform distribution on $\{0, 1, \ldots, N^\star\}$, were estimated. Sample output from forward tempering is shown in Figure 4. Perfect simulation was then carried out using $\bar{\pi}$, the geometric average pseudo-prior. Sample output from a perfect simulation run is shown in Figure 5.

How does the perfect sampler compare with regular MCMC and rejection sampling? A comparison based on CPU-time is preferred to one based on updates, since the perfect tempering updates are cheaper to compute on average than regular MCMC updates, in any reasonable implementation. Referring to Table 1, our perfect sampling process was about 50 times less efficient than our regular MCMC, and around 100 times more efficient than our rejection sampler. Note that greater ingenuity will uncover better rejection and standard-MCMC algorithms. However, these lead directly to a more efficient dominated tempering algorithm.

We wish to make the connection with rejection explicit here. Consider a tempering schedule with $N^\star = 2$, $\beta_1 = 0$ and $\beta_2 = 1$, and pseudo-prior π_D. In the perfect tempering algorithm, $\alpha((x, 1), (x, 2)) = h_2(x)/h_1(x)$ (since $K_1 = 1$ and $\pi_{D,1} = \pi_{D,2}$, so $r((x, 1), (x, 2)) < 1$). Consider the rejection sampler with envelope function h_1: this algorithm accepts $x \sim h_1$ with probability $h_2(x)/h_1(x)$; in case of acceptance, $x \sim h_2$; otherwise we repeat over until we obtain acceptance. The two algorithms are essentially the same, though the tempering process wastes time on transitions such as $(x, 1) \leftrightarrow (\mathbf{0}, 0)$. Referring to Eq. (8), the expected number of rejection/acceptance operations in the rejection algorithm is $c_1/c_2 \simeq 250000$ (the inverse of the Bayes factor for the prior $P(x)$). The efficiency of rejection in updates is thus $\mathcal{E}_R = 4 \times 10^{-6}$ (as in Table 1). We improve on rejection by replacing pseudo-prior π_D by $\bar{\pi}$, the geometric average pseudo-prior. Under $\bar{\pi}$, the event $r((x, 1), (x, 2)) \geq 1$ may occur. At $\mathcal{E}_P = 2 \times 10^{-4}$, our perfect tempering is fifty times more efficient than the rejection to which it is closely related. It has the efficiency of Metropolised independent sampling (Liu, 1996), but gives perfect samples, as a rejection algorithm would do. Basing the comparison on CPU-time leads to the same factor (the update work is comparable). The implementation overheads in a perfect sampler of this kind, compared to rejection, are slight. However, we do even better with the tempering schedule with $N^\star = 3$, given above.

3.3 Example: Radiocarbon age calibration

This instance of simulation based Bayesian inference has a couple of related features which might be expected to make perfect simulation difficult. The prior density is integrable but unbounded, and it is not easy to get good convergence in the usual sense, with mixtures of Metropolis-Hastings updates.

3.3.1 Definitions

The data comprises $K = 14$ conventional radiocarbon age determinations $y = (y_1, \ldots, y_K)$ and standard errors $\tilde{\sigma} = (\tilde{\sigma}_1, \ldots, \tilde{\sigma}_K)$ taken from Anderson et al. (1996) (the SM/C:Dune terrestrial series). In the format $(y_n, \tilde{\sigma}_n)$, the data are $(580, 47)$, $(630, 82)$, $(600, 50)$, $(537, 44)$, $(600, 50)$, $(509, 72)$, $(570, 45)$, $(670, 47)$, $(624, 58)$, $(560, 45)$, $(646, 47)$, $(630, 35)$, $(660, 46)$, and $(787, 72)$. The observation model for y_n is

$$y_n|\Theta_n \sim N(\mu(\Theta_n), \hat{\sigma}(\Theta_n)^2 + \tilde{\sigma}_n^2), \tag{9}$$

where Θ_n is the unknown true date (measured in calendar years AD) of the n'th dated specimen, and μ and $\hat{\sigma}^2$ are standard, empirically determined (Stuiver et al., 1998) radiocarbon calibration functions. These functions are available from `http://depts.washington.edu/qil/` in decadal tabulation, which we spline to arrive at functions μ and $\hat{\sigma}^2$ piecewise constant by year. A southern hemisphere correction of 27 and standard deviation 5 (McCormac et al., 1998) was subtracted from each y_n-value in the analysis. Let θ_n be a trial value for Θ_n and let $\ell(y_n|\theta_n)$ denote the normal density function as given in Eq. (9).

In the prior model the date parameters $\theta_1, \ldots, \theta_K$ are independently and uniformly distributed between two parameters $\chi = (\chi_1, \chi_2)$ which are themselves known to lie in a (time) interval $[A, B] \subset \mathbb{R}$ of length $R = B - A$, with $\chi_2 > \chi_1$, but are otherwise unknown. Letting $\theta = (\theta_1, \ldots, \theta_K)$ and $x = (\chi_1, \chi_2, \theta)$, the state space of the target distribution is

$$\Omega_G = \{x : A \leq \chi_1 < \chi_2 \leq B,\, \theta \in [\chi_1, \chi_2]^K\}.$$

See Buck and Litton (1996) for further background. The unnormalised posterior density g of x is

$$g(x|y; K) = \frac{1}{R - (\chi_2 - \chi_1)} \times \frac{1}{(\chi_2 - \chi_1)^K} \times \prod_{m=1}^{K} \ell(y_m|\theta_m)$$

with $\nu_G(dx) = d\chi_1 d\chi_2 d\theta_1 \ldots d\theta_K$, the restriction of Lebesgue measure to Ω_G.

3.3.2 Tempering sequence and simulation

We begin by specifying the sequence of tempering distributions $H_n(dx)$. We make a very simple choice: the level-n tempering distribution H_n coincides with the posterior distribution which would be appropriate for the subset $y = (y_1, \ldots, y_{n-1})$, $\tilde{\sigma} = (\tilde{\sigma}_1, \ldots, \tilde{\sigma}_{n-1})$ of the data. Let $N^\star = K + 1$. The sequence of parameter spaces have increasing dimension, starting with the parameter pair $x = (\chi_1, \chi_2)$, for $x \in \Omega_1$, and then adding in turn the parameters θ_1 up to θ_K, so that $x = (\chi_1, \chi_2, \theta_1, \ldots, \theta_{n-1})$ for $x \in \Omega_n$. The parameter vectors and parameter spaces are

$$\Omega_n = \{x : A \leq \chi_1 < \chi_2 \leq B,\, (\theta_1, \ldots, \theta_{n-1}) \in [\chi_1, \chi_2]^{n-1}\}, \qquad 1 \leq n \leq N^\star$$

with $\Omega_1 = \{x : A \leq \chi_1 \leq \chi_2 \leq B\}$. The densities are chosen to be $h_n(x) = g(x|y; n-1)$ so that

$$h_1(x) = \frac{1}{R - (\chi_2 - \chi_1)}$$

and, for $2 \leq n \leq N^\star$,

$$h_n(x) = \frac{1}{R - (\chi_2 - \chi_1)} \times \frac{1}{(\chi_2 - \chi_1)^{n-1}} \times \prod_{m=1}^{n-1} \ell(y_m|\theta_m).$$

Finally $\nu_1(dx) = d\chi_1 d\chi_2$, and for $2 \leq n \leq N^\star$, $\nu_n(dx) = d\chi_1 d\chi_2 d\theta_1 d\theta_2 \ldots d\theta_{n-1}$, in both cases the restriction of Lebesgue measure to Ω_n $(1 \leq n \leq N^\star)$.

For $1 \leq n < N^\star$, the tempering transition from $N_t = n$ to $N_{t+1} = n + 1$ is a "birth" transition for parameter θ_n. Since we have $\chi_1 \leq \theta_n \leq \chi_2$ and $x = (\chi_1, \chi_2, \theta_1, \ldots, \theta_{n-1})$ for $x \in \Omega_n$, the candidate state is $x' = (\chi_1, \chi_2, \theta_1, \ldots, \theta_{n-1}, \theta_n)$, with

$$\theta_n \sim \ell(y_n|\theta_n) \times \mathbb{I}_{\theta_n \in [\chi_1, \chi_2]}$$

The unnormalised densities $\ell(y_n|\cdot)$ are multi-modal but otherwise fairly smooth, and bounded in domain and range. They may be sampled straightforwardly by rejection sampling, or by numerical integration and inversion.

When $1 < n \leq N^\star$, the tempering transition from $N_t = n$ to $N_{t+1} = n - 1$ is a "death" transition for parameter θ_{n-1}. If $x = (\chi_1, \chi_2, \theta_1, \ldots, \theta_{n-1}) \in \Omega_n$, then the candidate state is $x' = (\chi_1, \chi_2, \theta_1, \ldots, \theta_{n-2})$.

The tempering transition from $N_t = 0$ to $N_{t+1} = 1$ is a birth transition for the pair (χ_1, χ_2). Since we can sample according to H_1 directly, we simply set $f_{0,1}(x, dx') = H_1(dx')$. This direct sampling is accomplished by taking $r \sim U(0, R)$, $\chi_1 \sim U(A, A + (R - r))$ and $\chi_2 = \chi_1 + r$. The transition from $N_t = 1$ to $N_t = 0$ is a death transition for the pair (χ_1, χ_2).

With these operations generating candidate states in the MCMC, it follows that

$$r((x, n), (x', n + 1)) = \frac{\int_{\chi_1}^{\chi_2} \ell(y_n|z)\, dz}{\chi_2 - \chi_1} \times \frac{\pi_{n+1}}{\pi_n},$$

for $1 \leq n < N^\star$, and for $n = 0$,

$$r((\mathbf{0}, 0), (x', 1)) = R\pi_1/\pi_0.$$

Condition (C2a) is seen to hold.

We now turn to the dominating process and condition (C2b). As in Section 3.2.2 we are free to set $\pi_0 = R\pi_1$ so that $r((\mathbf{0}, 0), (x', 1)) = 1$. Letting $\ell^*(y_n)$ equal the largest value taken by $\ell(y_n|z)$ for $z \in [A, B]$, we satisfy condition (C2b) taking $K_0 = R$ and $K_n = \ell^*(y_n)$ for $1 \leq n < N^\star$. Note that $\tilde{\alpha}(0, 1) = \alpha((\mathbf{0}, 0), (x', 1)) = 1$ independent of x' in Figure 1. Such perfect coupling of N_t and D_t in the neighbourhood of $n = 0$ will often be possible when a variable dimension tempering sequence is used, as it is often possible to generate IID samples from fixed univariate and bivariate distributions.

3.3.3 Regular MCMC

As in Section 3.2.3 we use the same updates in our regular MCMC algorithm and at fixed-level updates in our tempering algorithm.

Suppose the current state is (x, n) and $n \geq 1$; set $n = N^\star$ in the case of the regular MCMC algorithm. We have:

1. a Metropolis adjusted random walk update acting on a randomly chosen parameter; this gives ergodicity in principle, but is insufficient in practice;

2. a Metropolis adjusted random walk update of fixed scale Δ applied to all dates; a number $\delta \sim U(-\Delta, \Delta)$ is generated; the candidate state is $x'_i = x_i + \delta$ for each $i = 1, 2 \ldots n + 1$;

3. a Metropolis-Hastings adjusted centred random scaling; two random numbers $\eta \sim U(0,1)$ and $\rho \sim U(\eta, 1/\eta)$ are generated; the candidate state is $x'_i = \rho x_i - \frac{\rho-1}{n+1}\sum_j x_j$ for each $i = 1, 2 \ldots n+1$, and Hastings' ratio is

$$r((x,n),(x',n)) = \frac{1}{\rho} \times \frac{R - (\chi_2 - \chi_1)}{R - (\chi'_2 - \chi'_1)} \times \prod_{m=1}^{n-1} \left(\frac{\ell(y_m|\theta'_m)}{\ell(y_m|\theta_m)} \right).$$

At each update in the regular MCMC, with probability 1/3 each, one of these three moves is chosen. In our tempering, we take $p = q = 1/3$: when $p < u_t^1 < 1 - q$ in STupdate, Figure 1, these same moves are used as fixed-level updates, in the same proportions.

3.3.4 Results from experiments

Sample output from a forward run is plotted in Figure 6, and from a perfect simulation in Figure 7. Posterior distributions for radiocarbon calibration with up to about ten dates are readily sampled by rejection. However, referring to Table 1, the efficiency of our best rejection algorithm for the posterior density of the full set of fourteen dates was around a tenth (per update) or a fiftieth (per CPU-second) that of our perfect simulation. We regard the CPU-time comparison as more representative, as our rejection update was intrinsicly more expensive to compute than a tempering update. Our rejection algorithm took advantage of the conditional independence of the θ_n given χ_1 and χ_2.

Referring to Table 1, our perfect simulation is competitive with regular MCMC in this problem (regular MCMC is about fifty times more efficient per update, but only nine times more efficient per CPU-second, in comparable implementations). Our regular MCMC suffers here from correlation in equilibrium, despite having three carefully tailored move types. Radiocarbon calibration is a hard sampling problem for regular MCMC: tempering might be appropriate, independent of its role in perfect sampling.

The rejection algorithm suggested a number of improvements to the perfect tempering algorithm. These were not exploited for the following reason. The perfect simulation algorithm presented here is intended to illustrate variable dimension style tempering, or "sintering" (Liu and Sabatti, 1998). More efficient rejection or regular MCMC algorithms might be given, however, as discussed in Section 3.2.4, such improvements will in general lead directly to more efficient dominated tempering algorithms.

3.4 The Strauss process

3.4.1 Distribution and tempering

We wish to emphasise that target distributions G of random variables X^G of randomly variable dimension may be sampled perfectly within the family of algorithms we have outlined. As an example we consider a Strauss point process (Strauss, 1975; Kelly and Ripley, 1976) below. Perfect simulation (Kendall, 1997b; Kendall, 1998; Häggström et al., 1999; Thönnes, 1999; Møller and Schladitz, 1999; Kendall and Møller, 1999) and regular MCMC algorithms (Geyer and Møller, 1994; Geyer, 1999; Møller, 1999a) have been given for a wide range of spatial point processes.

Let Ω_G denote the set of all finite subsets of the unit square $S = [0,1]^2$; thus an element $x \in \Omega_G$ is a point configuration of the form $x = \{x_1, \ldots, x_k\} \subset S$ with $0 \leq k < \infty$; if $k = 0$ then $x = \emptyset$ is the empty point configuration. Let ν_G be the Poisson process of rate $\beta > 0$ on S. In other words, if $X^P \sim \nu_G$ then, firstly, the number of points in X^P (the count) follows a Poisson distribution with mean β and, secondly, conditional on this count, points in X^P are IID and uniformly distributed on S. We specify a Strauss process by down-weighting the density for point configurations of X^P by a factor $\gamma^{s_R(x)}$, where $s_R(x)$ is the number of pairs of points in $x \in \Omega_G$ within a distance R of one another ("R-close"), and $R > 0$ and $0 \leq \gamma \leq 1$ are parameters of the process. Then our target Strauss density becomes

$$g(x) = \gamma^{s_R(x)}.$$

In this problem we use a sequence of densities distinguished by the value of the interaction parameter γ_n in a tempering schedule using $1 = \gamma_1 > \gamma_2 > \cdots > \gamma_{N^\star} = \gamma$. Our tempering sequence interpolates between the distributions of X^P and X^G (varying R instead of γ, from $R_1 = 0$ to $R_{N^\star} = R$ gives an awkward implementation). We choose

$$h_n(x) = \gamma_n^{s_R(x)}, \qquad \Omega_n = \Omega_G, \qquad \nu_n = \nu_G$$

for $n = 1, \ldots, N^\star$, and let $p = q = 1/3$. When a fixed level tempering transition is selected (*ie* $p < u_t^1 < 1 - q$ in $\mathsf{STupdate}$, Figure 1), we update the state using Metropolis-Hastings updates of the kind described in Geyer and Møller (1994). We propose a change to the tempering level whenever $u_t^1 < p$ or $u_t^1 > 1 - q$ in $\mathsf{STupdate}$, Figure 1, and when we do this, we do not otherwise change the state, (a point configuration which could contain any number of points). We therefore have

$$r((x,n),(x',n+1)) = \frac{\pi_{n+1}}{\pi_n} \times (\gamma_{n+1}/\gamma_n)^{s_R(x)}$$

for $1 \leq n < N^\star$. At the transition from $n = 0$ to $n = 1$, we sample $x \sim H_1$ directly, since $(X|N = 1)$ is a Poisson point process. It follows that $r((\mathbf{0},0),(x',1)) = \pi_1/\pi_0$ for this transition. Condition (C2a) is satisfied. We set $\pi_0 = \pi_1$, to obtain $r((\mathbf{0},0),(x',1)) = 1$, as in Section 3.2.2. Finally $(\gamma_n/\gamma_{n+1})^{s_R(x)} \leq 1$ so condition (C2b) is satisfied with $K_n = 1$ for each $n = 0, 1, \ldots N^\star - 1$.

For what parameter values does the proposed simulation scheme fail? Let

$$\rho = \mathsf{E}_G s_R / \mathsf{E}_P s_R\,.$$

This is a strictly increasing function of $\gamma \in [0,1]$ for fixed β, R and S; the limiting case $\gamma = \rho = 0$ is a hard core point process, while $\gamma = \rho = 1$ is the Poisson process ν_G. The particular method described here fails when ρ is small, so that the average number of R-close pairs in the Strauss state is significantly smaller than the number in the Poisson state. However, it is possible that other sequences of tempering distributions, interpolating distributions from a lattice process, or some other spatial hierarchy, will be more effective; see, for example, Mase et al., (1999).

Finally, for the tempering distributions as specified above, the normalising constants c_n of Eq. (8) satisfy $1 = c_0 = c_1 < c_2 < \cdots < c_{N^\star}$, and so $\pi_{N,0} = \pi_{N,1} > \pi_{N,2} > \cdots > \pi_{N,N^\star}$ as discussed in Section 3.1.2.

3.4.2 Results from experiments

A perfect sampler for the Strauss process was implemented, by modifying an earlier implementation of a rejection sampler. The existence of this code motivated a tempering schedule with $N^\star = 2$, $\gamma_1 = 1$ and $\gamma_2 = \gamma$. The return time of our program is sensitive to ρ. Perfect simulation and rejection approach regular MCMC in efficiency as ρ grows. We took parameter values $\gamma = 0.75$, $\beta = 100$ and $R = 0.1$ in a unit square with periodic boundary conditions, giving $\rho \simeq 0.3$. This is about the smallest ρ-value we can treat with this particular tempering schedule, in a reasonable amount of time, on available hardware. A simulated state is shown in Figure 8.

The connection between this algorithm and the related rejection algorithm (rejecting from a Poisson point process) has been covered in Section 3.2.4. In particular, c_1/c_2 is the average number of trial states the rejection sampler generates per returned sample. We measured $c_1/c_2 \simeq 2.4 \times 10^{10}$ so we calculate that $\mathcal{E}_R \simeq 4 \times 10^{-11}$; we measured $\mathcal{E}_P \simeq 5 \times 10^{-8}$ so perfect sampling is around 2000 times more efficient than rejection, per update. Rejections cost more than tempering updates since a proportion of tempering updates are single point operations, and null operations. However, we factorise the rejection in the following way: we generate a Poisson candidate x and put the points x_i into place one at a time; at the insertion of the $i+1$'th point $s_R(x_{i+1}|x_1 \ldots x_i)$ is the number of new R-close pairs; accept the new point with probability $\gamma^{s_R(x_{i+1}|x_1 \ldots x_i)}$; if the new point is rejected, reject the whole state x and start over; if we get to the last point in x and it is accepted, accept x. The probability to accept x by this process is $\gamma^{s_R(x)}$, as required.

In the perfect tempering run which we report, two out of three forward simulations generated zero samples. This suggests we might obtain larger efficiencies, for our perfect sampler, with a little further experimentation with L, π and $N^\star$ parameters. As a result of these features of our implementations, rejection catches up a little: our perfect simulation is 1000 times more efficient than our rejection when we measure efficiency in CPU-time units.

On the other hand, regular MCMC is around $100,000$ times more efficient than our perfect simulation, measuring in updates. Perfect simulation might be used as a check on regular MCMC here, but it would not replace it.

4 Discussion

4.1 Another perfect simulated tempering algorithm

A perfect tempering algorithm, simulating $Z_0 \sim H$ more efficiently than $\mathsf{PWperfect}$, may be given. Let $V_t^2, t \in \mathbb{Z}$ be U(0,1)-variates independent of each other and all other details of the random walk and tempering chains. We modify the joint simulation of D_t and Z_t, coupling

$$D_{t+1} = \mathsf{RWupdate}(D_t; U_t^1, U_t^2)$$

with

$$Z_{t+1} = \mathsf{STtight}(Z_t; D_t, D_{t+1}, U_t^1, V_t^2).$$

The function $\mathsf{STtight}$ is given in Figure 9.

Figure 9 should be compared with Figure 1. The random number $U_t^2 = u_2$ is replaced by $V_t^2 = v_2$, but the Z_t-process is updated using $\mathsf{STupdate}$ as before both when D_t strictly dominates Z_t and when a fixed-level update is proposed at the $D_t \to D_{t+1}$-transition. Call this Case I. Suppose in contrast that $D_t = N_t$, and a move up or down is proposed at the $D_t \to D_{t+1}$-transition. This is Case II. We use the random numbers in a different way in Case II, so that domination is retained. The Z_t-acceptance is, in Case II, determined by two events, the D_t accept-reject event, which uses U_t^2, and the Z_t accept-reject event, which uses V_t^2 and is conditional on the outcome of the D_t accept-reject event. The Markov chain (Z_t, D_t) has the same distribution and domination properties as before (see Section 2.2).

Our new tempering update $\mathsf{STtight}$ allows us to use a more efficient perfect simulation procedure. We simulate the random walk in reverse from its equilibrium at time zero until it hits the atom; the tempering is simulated forwards from that time up to time zero, conditional on events in the first simulation. More precisely, we begin by drawing D_0 from its equilibrium distribution. We then simulate the random walk backwards in time $t \leq 0$. By reversibility, we can use

$$D_{-t} = \mathsf{RWupdate}(D_{-t+1}; 1 - U_{-t}^1, U_{-t}^2), \quad t = 1, 2 \ldots$$

We stop this generation at time $-\tau = \inf\{t \geq 0 : D_{-t} = 0\}$ (by ergodicity, $\tau < \infty$ almost surely). By property (P1), we know that $Z_{-\tau} = \mathbf{0}$. Conditional on the path $D_{-\tau}, \ldots, D_0$ together with the "marks" $U_{-\tau}^1, \ldots, U_0^1$, we now simulate Z_t forwards in time from $Z_{-\tau} = (\mathbf{0}, 0)$:

$$Z_{-t+1} = \mathsf{STtight}(Z_{-t}; D_{-t}, D_{-t+1}, U_{-t}^1, V_{-t}^2), \qquad t = -\tau, \ldots, 1.$$

The return state Z_0 follows the distribution H (this is easily verified modifying the proof of Theorem 3.1 in Kendall and Møller (1999)).

This perfect simulation algorithm looks like rejection sampling as in Fill's perfect simulation algorithm (Fill, 1998). However, it is not, as τ is random. The algorithm may be extended to the case where $N^\star = \infty$ without difficulty. It is not clear whether this is possible for PWperfect.

4.2 Useful tempering transitions

Consider some target distribution G given in terms of a product measure and a density which is a product over interacting cliques. How should we modify G in order to determine the tempering sequence $H_0, H_1, \ldots, H_{N^\star}$? There is also the question of how we might choose the candidate generation distributions, $f_{n,n'}(x, dx')$. Below we present a list of tempering transitions which we have found useful.

Without loss of generality, let $K_n = 1$ for each $0 \leq n < N^\star$. We are aiming to find $f_{n,n'}(x, dx')$ so that $\pi_{N,n} \simeq \pi_{N,n+1}$ for each $0 \leq n < N^\star$, so we are looking for a transition rule $(x, n) \to (x', n')$ which leads to an acceptance ratio α which is close to one for all the pairs of states the rule connects. We may

1. modify the clique interactions in H_n without altering the space of states, so that $\Omega_1 = \ldots = \Omega_{N^\star}$; as in temperature-indexed tempering, we flatten G towards a simple tractable distribution H_1; we might sample the affected variables according to their new conditional distribution, or if this is not convenient (as in Section 3.2 and Section 3.4), we have the option to determine the candidate state by setting $x' = x$;

2. add (or remove) a variable and some or all of its clique interactions; in a variable birth operation we are obliged to sample a value for the new variable; $f_{n,n'}$ must be chosen so that acceptance ratios are bounded well away from zero;

3. add (remove or move) atoms in the state space of one of the variables; in this way continuously distributed variables may be discretised, or assigned a value at an atom;

4. jump from H_0 to a distribution H_1 which has more random variables than G (but simpler interactions), and proceed to *remove* the extra variables as the tempering level n increases towards $N^\star$.

4.3 Conclusions

As always, hard sampling problems are solved using good MCMC update strategies, in which case-specific details make a difference. We have given a framework within which such effective updates can be embedded, to obtain provably perfect simulation. Because we do not require the existence of a partial ordering on the state space Ω (rather, our domination is based on a simpler complete ordering in $\{0, 1, \ldots, N^\star\}$) there exist families of perfect tempering algorithms, based on standard tempering schemes, which generalise fairly easily. Thus the modified logistic regression of Section 3.2 and the Strauss spatial point process of Section 3.4 are both sampled perfectly using temperature-based tempering of the kind described in Marinari and Parisi (1992). Existing MCMC and rejection programmes may be used to suggest good tempering schedules, as well as contributing components to a perfect tempering computer programme. Algorithm components RWupdate, STupdate and PWperfect can be implemented once and reused for different problems.

We expect that, for difficult sampling problems in which regular MCMC algorithms are unreliable, perfect tempering will approach regular MCMC in efficiency (all things considered). We see evidence of this in the example of Section 3.3. However, perfect tempering does not need to be as efficient as regular MCMC in order to be useful.

Perfect tempering might be used as a quality test for a given tempering scheme. If coalescence is obtained in the perfect tempering, in a reasonable number of updates, forward tempering may be used to gather samples, without reinitialising the tempering using dominated coupling from the past. We expect that samples obtained in this way will, for all practical purposes, have the same quality as our perfect samples, in many problems.

Acknowledgement: JM was supported by the European Union's research network "Statistical and Computational Methods for the Analysis of Spatial Data, ERB-FMRX-CT96-0096", by the Centre for Mathematical Physics and Stochastics (MaPhySto), funded by a grant from the Danish National Research Foundation. GKN was supported by the the Stochastic Centre of Chalmers University, Sweden and by the Center for Archaeological Research at the University of Auckland. We acknowledge advice and encouragement from Martin B. Hansen of MaPhySto and the Department of Mathematics, Aalborg University.

References

Anderson, A. J., Smith, I. W. G., and Higham, T. F. G. (1996). Shag river mouth: the archaeology of an early Southern Maori village. In Anderson, A. J., Allingham, B., and Smith, I. W. G., editors, *Shag River Mouth*, volume 27, pages 61–69. ANH Publications, RSPAS, ANU, Canberra.

Bliss, C. I. (1935). The calculation of the dosage-mortality curve. *Annals of Applied Biology* **22**, 134–167.

Carlin, B. P. and Louis, T. A. (1996). *Bayes and Empirical Bayes methods for Data analysis.* Chapman and Hall, London.

Fill, J. (1998). An interruptible algorithm for exact sampling via Markov Chains. *Annals of Applied Probability* **8**, 131–162.

Fismen, M. (1997). Exact sampling using Markov chains. Diploma thesis, Department of Mathematical Sciences, Norwegian University of Technology and Science, Trondheim.

Geyer, C. J. (1992). Practical Markov chain Monte Carlo (with discussion). *Statist. Sci.* **7**, 473–511.

Geyer, C. J. (1999). Likelihood inference for spatial point processes. In Barndorff-Nielsen, O., Kendall, W., and van Lieshout, M., editors, *Stochastic Geometry: Likelihood and Computation*, pages 79–140, Boca Raton. Chapman and Hall/CRC.

Geyer, C. J. and Møller, J. (1994). Simulation and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics* **21**, 359–373.

Geyer, C. J. and Thompson, E. A. (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *J. Amer. Statist. Assoc.* **90**, 909–920.

Gilks, W., Richardson, S., and Spiegelhalter, D., editors (1996). *Markov Chain Monte Carlo in Practice.* Chapman and Hall.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **82**, 711–732.

Green, P. J. and Murdoch, D. J. (1999). Exact sampling for Bayesian inference: towards general purpose algorithms. In Bernardo, J., Berger, J., Dawid, A., and Smith, A., editors, *Bayesian Statistics 6.* Oxford University Press. Presented as an invited paper at the 6th Valencia International Meeting on Bayesian Statistics, Alcossebre, Spain, June 1998.

Häggström, O. and Nelander, K. (1997). Exact sampling from anti-monotone systems. Research Report 1997-03, Department of Mathematics, Chalmers University. *Statistica Neerlandica* (to appear).

Häggström, O., van Lieshout, M. N. M., and Møller, J. (1999). Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. *Bernoulli* **5**, 641–659 (to appear).

Hobert, J. P., Robert, C. P., and Titterington, D. M. (1998). On perfect simulation for some mixtures of distributions. *Statistics and Computing* (to appear).

Kelly, F. P. and Ripley, B. D. (1976). A note on Strauss's model for clustering. *Biometrika* **63**, 357–360.

Kendall, W. S. (1997a). On some weighted Boolean models. In Jeulin, D., editor, *Advances in Theory and Applications of Random Sets*, pages 105–120, Singapore. World Scientific Publishing Company.

Kendall, W. S. (1997b). Perfect simulation for spatial point processes. In *Proc. ISI* 51^{st} *session, Istanbul (August 1997)*, volume 3, pages 163–166.

Kendall, W. S. (1998). Perfect simulation for the area-interaction point process.

In Accardi, L. and Heyde, C., editors, *Probability Towards 2000*, pages 218–234, New York. Springer.

Kendall, W. S. and Møller, J. (1999). Perfect Metropolis-Hastings simulation of locally stable point processes. Technical Report R-99-2001, Department of Mathematics, Aalborg University.

Kendall, W. S. and Thönnes, E. (1997). Perfect simulation in stochastic geometry. Research report 323, Department of Statistics, University of Warwick. *J. Pattern Recognition* (to appear).

Litton, C. and Buck, C. (1996). An archaeological example: radiocarbon dating. In Gilks, W., Richardson, S., and Spiegelhalter, D., editors, *Markov Chain Monte Carlo in Practice*, pages 466–486. Chapman and Hall, London.

Liu, J. S. (1996). Metropolized independent sampling. *Statistics and Computing* **6**, 113–119.

Liu, J. S. and Sabatti, C. (1998). Simulated sintering: Markov chain Monte Carlo with spaces of varying dimension. In Bernado, J., Berger, J., Dawid, A., and Smith, A., editors, *Bayesian Statistics 6*. OUP.

Marinari, E. and Parisi, G. (1992). Simulated tempering: a new Monte Carlo scheme. *Europhys. Lett.* **19**, 451–458.

Mase, S., Møller, J., Stoyan, D., Waagepetersen, R., and Döge, G. (1999). Packing densities and simulated tempering for hard core Gibbs point processes. Technical Report R-99-2002, Department of Mathematical Sciences, Aalborg University.

McCormac, F. G., Hogg, A. C., Higham, T. F. G., Baillie, M. G. L., Palmer, J. G., Xiong, L., Pilcher, J. R., Brown, D., and Hoper, S. T. (1998). Variations of radiocarbon in tree-rings: Southern hemisphere offset preliminary results. *Radiocarbon* **40**, 1153–1162.

Meyn, S. P. and Tweedie, R. L. (1993). *Markov Chains and Stochastic Stability.* Springer Verlag, New York.

Mira, A., Møller, J., and Roberts., G. O. (1999). Perfect slice samplers. In preparation.

Møller, J. (1999a). Markov chain Monte Carlo and spatial point processes. In Barndorff-Nielsen, O., Kendall, W., and van Lieshout, M., editors, *Stochastic Geometry: Likelihood and Computation*, number 80 in Monographs on Statistics and Applied Probability, pages 141–172, Boca Raton. Chapman and Hall/CRC.

Møller, J. (1999b). Perfect simulation of conditionally specified models. *Journal of Royal Statistical Society* **B 6**, 251–264.

Møller, J. and Schladitz, K. (1999). Extensions of Fill's algorithm for perfect simulation. *Journal of the Royal Statistical Society* **B 61**, (to appear).

Murdoch, D. J. and Green, P. J. (1998). Exact sampling from a continuous state space. *Scandinavian Journal of Statistics* **25**, 483–502.

Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures and Al-*

gorithms **9**, 223–252.

Propp, J. G. and Wilson, D. B. (1998). How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms* **27**, 170–217.

Sokal, A. (1989). Monte Carlo methods in Statistical Mechanics. In *Cours de Troisième Cycle de la Physique en Suisse Romande, Lausanne*.

Strauss, D. J. (1975). A model for clustering. *Biometrika* **63**, 467–475.

Stuiver, M., Reimer, P. J., Bard, E., Beck, J. W., Burr, G. S., Hughen, K. A., Kromer, B., McCormac, F. G., d. Plicht, J., and Spurk, M. (1998). Intcal98 Radiocarbon Age Calibration, 24,000-0 cal BP. *Radiocarbon* **40**, 1041–1083.

Thönnes, E. (1999). Perfect simulation of some point processes for the impatient user. *Advances in Applied Probability* **31**, 69–87.

Experiment	Rejection $\mathcal{E}_R$		Perfect Tempering $\mathcal{E}_P$		Regular MCMC $\mathcal{E}_M$	
	UPD^{-1}	SEC^{-1}	UPD^{-1}	SEC^{-1}	UPD^{-1}	SEC^{-1}
Flour Beetle	4×10^{-6}	8×10^{-4}	1×10^{-4}	1×10^{-1}	9×10^{-2}	5.3
Radiocarbon	2×10^{-6}	2×10^{-4}	2×10^{-5}	1×10^{-2}	1×10^{-3}	9×10^{-2}
Strauss	4×10^{-11}	2×10^{-6}	5×10^{-8}	2×10^{-3}	7×10^{-3}	4×10^{3}

Table 1: Efficiency results, estimated for the examples of Section 2.3, according to the definition for $\mathcal{E}$ given in Section 3.1.1. Subscripts R, P and M denote rejection, perfect tempering and regular MCMC respectively. Columns UPD^{-1} and SEC^{-1} give independent samples per update and per CPU-second respectively. Rows correspond to the sampling problems of Sections 3.2, 3.3 and 3.4.

Dosage ω_i	1.6907	1.7242	1.7552	1.7842	1.8113	1.8369	1.8610	1.8839
# killed y_i	6	13	18	28	52	53	61	60
# exposed a_i	59	60	62	56	63	59	62	60

Table 2: Flour Beetle Mortality data, extracted from Bliss (1935), used in the example of Section 3.2.

$z = \mathsf{STupdate}((x,n); u^1, u^2)$

```
z ← (x, n)
if u¹ < p
        n' ← n + 1
else if u¹ > 1 − q
        n' ← n − 1
else
        n' ← n
if 0 ≤ n' ≤ N*
        draw x' ~ f_{n,n'}(x, dx')
        if u² ≤ α((x, n), (x', n'))
                z ← (x', n')
return z
```

Figure 1: Algorithm evaluating $\mathsf{STupdate}((x,n); u_1, u_2)$, where $(x,n) \in \Omega$ is the current state of the simulated tempering chain and z is the next state, while u^1 and u^2 are $\mathrm{U}(0,1)$, independently realised, random numbers. The notation $x' \sim f_{n,n'}(x, dx')$ indicates a simulation of x' according to $f_{n,n'}(x, dx')$ using an independent stream of random numbers.

$d =$ RWupdate$(m; u^1, u^2)$

$d \leftarrow m$

if $u^1 < p$

$\quad m' \leftarrow m + 1$

else if $u^1 > 1 - q$

$\quad m' \leftarrow m - 1$

else

$\quad m' \leftarrow m$

if $0 \leq m' \leq N^\star$

$\quad$ if $u^2 < \tilde{\alpha}(m, m')$

$\quad\quad d \leftarrow m'$

return d

Figure 2: Algorithm evaluating the function RWupdate$(m; u^1, u^2)$, where $m \in \{0, 1, \ldots, N^\star\}$ is the current state of the random walk and d is the next state, while the random numbers u^1 and u^2 are the same as in Figure 1.

$[Z^{\mathrm{equi}}, \tau_\star] = \mathsf{PWperfect}(\underline{u}, T)$:

```
t ← −T
D ← N*
D' ← N*
repeat
        D ← RWupdate(D; u_t^1, u_t^2)
        t ← t + 1
        if D = D' or ( t = 0 and D ≠ 0 )
                return PWperfect(u, T + 1)
        D' ← RWupdate(D'; u_t^1, u_t^2)
until D=0
τ⋆ ← −t
Z^equi ← (0, 0)
for t = −τ⋆ to −1
        Z^equi ← STupdate(Z^equi; u_t^1, u_t^2)
return [Z^equi, τ⋆]
```

Figure 3: Algorithm evaluating $\mathsf{PWperfect}(\underline{u}, T)$ using a stream of pairs of IID $\mathrm{U}(0,1)$ random numbers $\underline{u} = ((u^1_{-1}, u^2_{-1}), (u^1_{-2}, u^2_{-2}), \ldots)$ associated with times $-1, -2, -3 \ldots$, and $T > 0$ a given integer. The return value Z^{equi} is distributed according to H.

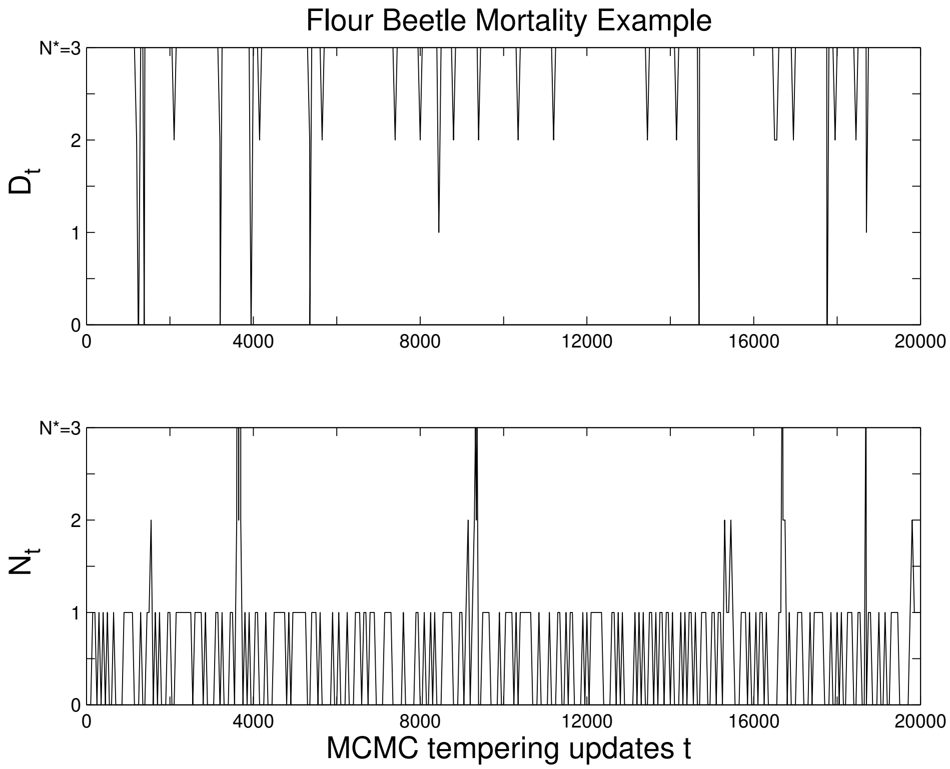

Figure 4: Flour Beetle Mortality example of Section 3.2. Forward simulation of the dominating process D_t (upper graph) with tempering process $Z_t = (X_t, N_t)$, (N_t, lower graph) for geometric average pseudo-prior, $\bar{\pi} = \sqrt{\pi_N \pi_D}$. Notice that $D_t \geq N_t$ at each MCMC update. At top tempering level $N^\star = 3$, the tempering distribution coincides with that of the target distribution G.

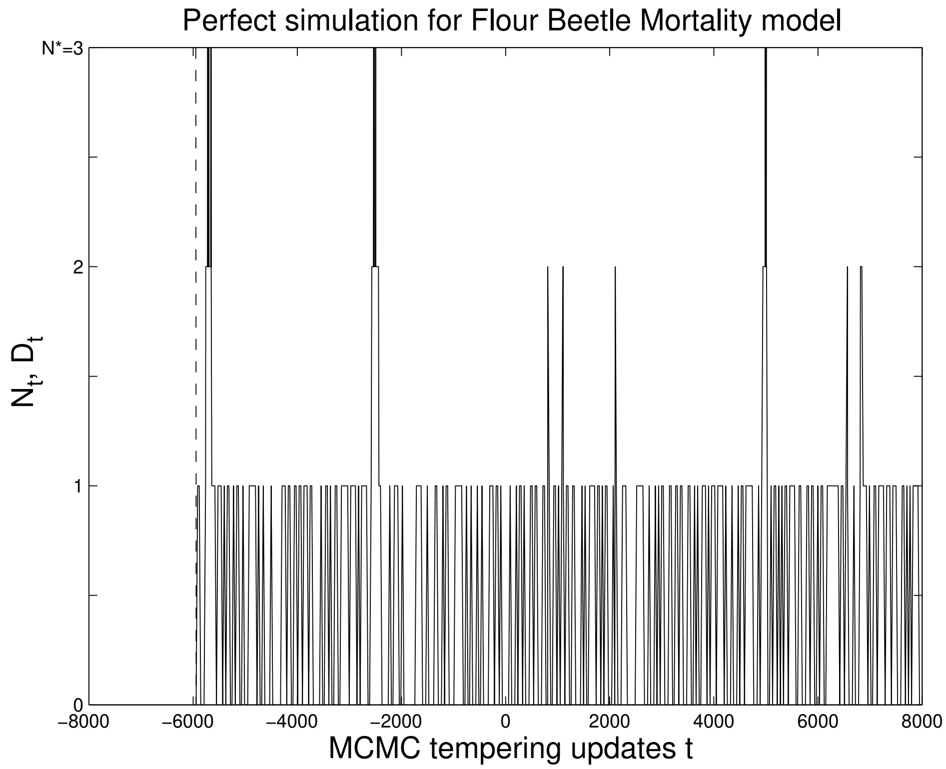

Figure 5: Flour Beetle Mortality example of Section 3.2. Typical perfect simulation, with the dominating process (D_t, - -, at left) and tempering process $Z_t = (X_t, N_t)$, (N_t, solid, at right) for geometric average pseudo-prior, $\bar{\pi} = \sqrt{\pi_N \pi_D}$ with $L = 8000$. The X_t with $N_t = N^\star$ and $t \geq 0$ are distributed according to G.

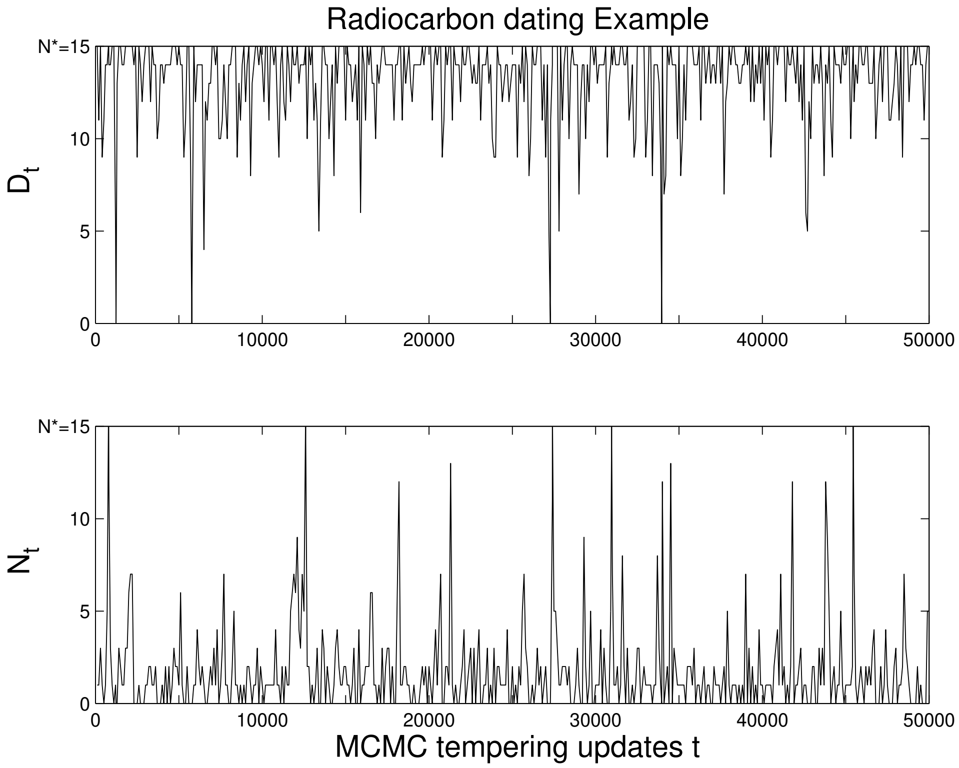

Figure 6: Radiocarbon dating example, Section 3.3. Forward simulation of the dominating process D_t (upper graph) with tempering process $Z_t = (X_t, N_t)$ (N_t, lower graph) for geometric average pseudo-prior, $\bar{\pi} = \sqrt{\pi_N \pi_D}$. At top tempering level $N^\star = 15$, the distribution has the full set of variables of the target distribution, whilst distributions at lower tempering levels have some subset of the full variable set.

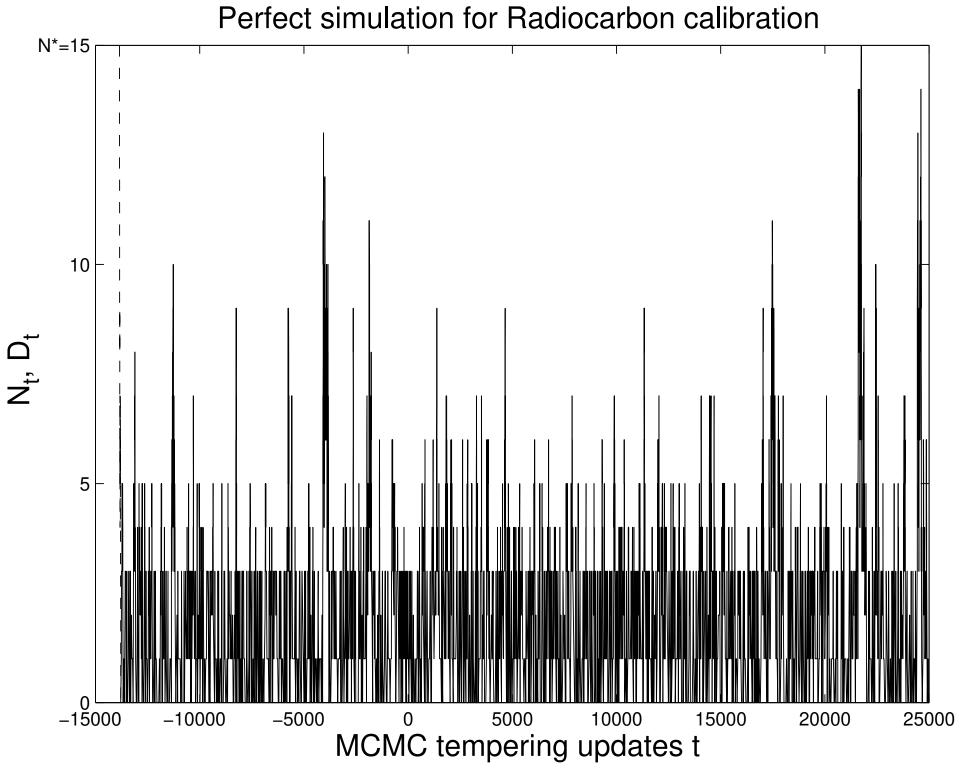

Figure 7: Radiocarbon calibration example, Section 3.3. Typical perfect simulation, with the dominating process (D_t, - -, at left) and tempering process $Z_t = (X_t, N_t)$, (N_t, solid, at right) for geometric average pseudo-prior, $\bar{\pi} = \sqrt{\pi_N \pi_D}$ and $L = 25000$.

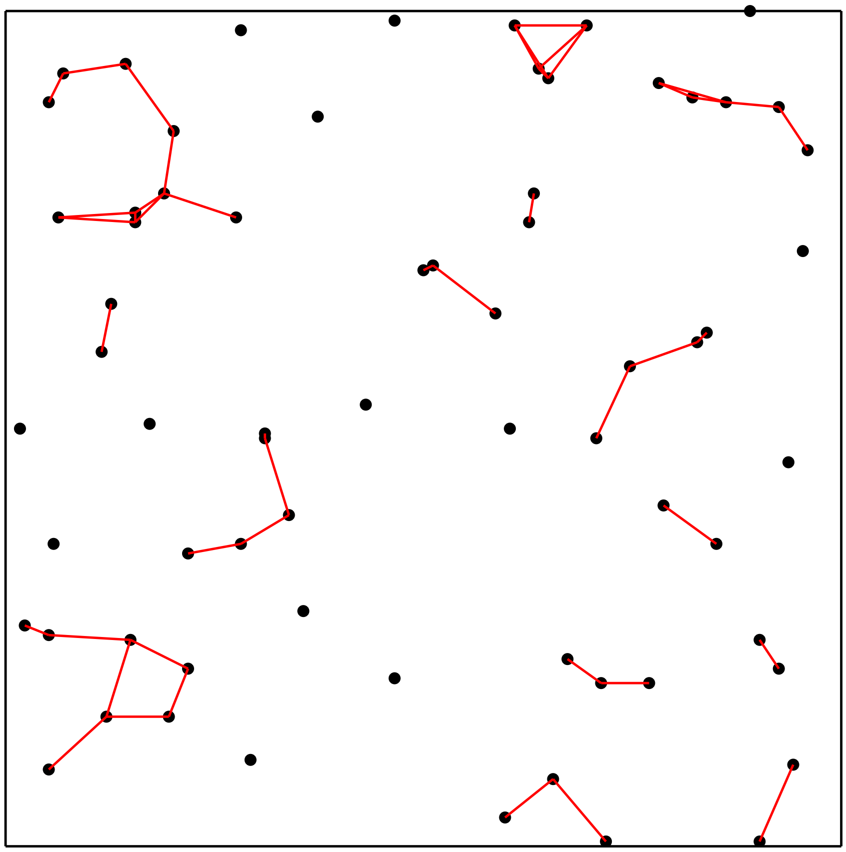

Figure 8: Realisation of the Strauss process, generated by perfect tempering with repulsion parameter $\gamma = 0.75$, Poisson intensity $\beta = 100$ and interaction radius $R = 0.1$ in a unit square with cylindrical boundary conditions. R-close pairs are joined by a line segment.

z =STtight$((x,n);m,d,u^1,v^2)$

```
if m > n or p < u^1 < 1 − q                                    /* Case I */

    z ← STupdate((x,n); u^1, v^2)

else                                                           /* Case II */

    z ← (x,n)
    if u_1 < p and d = n + 1
        draw x' ~ f_{n,n+1}(x, dx')
        n' ← n + 1
        if v^2 < α((x,n),(x',n'))/α̃(n,n')
            z ← (x',n')
    else if u_1 > 1 − q and n > 0
        draw x' ~ f_{n,n−1}(x, dx')
        n' ← n − 1
        if v^2 < (α((x,n),(x',n')) − α̃(n,n'))/(1 − α̃(n,n'))
            z ← (x',n')

return z
```

Figure 9: Algorithm evaluating STtight$((x,n);m,d,u^1,v^2)$, where d =RWupdate$(m;u^1,u^2)$, $(x,n) \in \Omega$, $m \in \{n,n+1,\ldots,N^\star\}$ and u^1, u^2 and v^2 are $\mathrm{U}(0,1)$, independently realised, random numbers. The notation $x' \sim f_{n,n'}(x,dx')$ indicates a simulation of x' according to $f_{n,n'}(x,dx')$ using an independent stream of random numbers.