NORGES TEKNISK-NATURVITENSKAPELIGE
UNIVERSITET

Using all Metropolis–Hastings proposals to estimate mean values

by

Håkon Tjelmeland

PREPRINT
STATISTICS NO. 4/2004

NORWEGIAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY
TRONDHEIM, NORWAY

This report has URL http://www.math.ntnu.no/preprint/statistics/2004/S4-2004.ps
Håkon Tjelmeland has homepage: http://www.math.ntnu.no/∼Haakon.Tjelmeland
E-mail: Haakon.Tjelmeland@stat.ntnu.no
Address: Department of Mathematical Sciences, Norwegian University of Science and Technology,
N-7491 Trondheim, Norway.

Using all Metropolis–Hastings proposals to estimate mean values

Håkon Tjelmeland
Department of Mathematical Sciences,
Norwegian University of Science and Technology,
NO-7491 Trondheim, Norway.

E-mail: Haakon.Tjelmeland@stat.ntnu.no

Abstract

We suggest to use all proposed states in a Metropolis–Hastings algorithm to estimate mean values. The traditional approach is to use only the accepted states. We propose to estimate the mean with a weighted mean of both accepted and rejected states. The amount of computation necessary to obtain the weights is proportional to the number of Metropolis–Hastings realisations. We identify sufficient conditions for the weighted mean to form an unbiased estimator. In simulation experiments the new estimator gives up to 33% lower estimation variance compared to the traditional estimator.

When both accepted and rejected states can be used to estimate mean values it becomes more attractive to run Metropolis–Hastings algorithms with several proposals in each iteration. Even if at most one of the proposals can be accepted, they can now all be used to estimate mean values. We generalise the Metropolis–Hastings scheme to this situation by defining a Markov chain for the joint distribution for the state vector and the potential new states. This framework allows moves that are not time reversible and the use of antithetic variables in the proposal distribution. Moreover, when the number of potential new states in each iteration is large, the resulting algorithm is ideally suited for parallel computation. In a simulation example with 128 proposals in each iteration, the variance of the new estimator is up to 76% lower than the variance of the traditional estimator.

Key words: Markov chain Monte Carlo, Metropolis–Hastings algorithm, multiple proposals, rejected states, weighted mean.

1 Introduction

Suppose we want to estimate the mean, μ, of a function $f(x)$ when x is distributed according to a target distribution $\pi(\cdot)$. If x is of high dimension and $\pi(\cdot)$ is sufficiently complex, the only viable alternative for this is to use Markov chain Monte Carlo (MCMC). The Metropolis–Hastings (Metropolis et al., 1953; Hastings, 1970) scheme is the most commonly used MCMC algorithm. Letting x denote the current state of the Markov chain, each iteration of the Metropolis–Hastings algorithm consists of two steps. First, a potential new state y is proposed from a proposal distribution. Second, y is accepted with a certain probability, otherwise the old state x is retained. The traditional estimator for μ is the empirical mean of $f(x)$, after having discarded a "burn-in" period. Thus, only accepted proposals are used in the estimation. Intuitively this appears as a waste of information and one would expect smaller estimation variance by using all proposed states. In Casella and Robert (1996) and Robert and Casella (1999) this is the background for developing an improved Rao-Blackwellised version of the traditional estimator. Simulation examples in Casella and Robert (1996) demonstrate that significant reduction in estimation variance can be obtained. However, the amount of computation necessary to compute the Rao-Blackwellised estimator is of order N^2, where N is the number of Metropolis–Hastings iterations. Measured in computation time the Rao-Blackwellised estimator is therefore inferior to the traditional estimator when N is large.

In this paper we define an estimator for μ by taking a weighted mean of both accepted and rejected states. We identify sufficient conditions on the weights for the estimator to be unbiased. In contrast to the Rao-Blackwellised estimator, computation time for our new estimator is or order N. In particular, the traditional estimator is one element in our new class of unbiased estimators.

When all proposed states can be used to estimate mean values it becomes more interesting to consider also algorithms where several potential new states are proposed in each iteration. These are ideally suited for parallel computation and even if at most one of the proposals can be accepted, they can now all be used to estimate μ. Metropolis–Hastings algorithms with several proposals in each iteration are discussed in Liu et al. (2000) and Qin and Liu (2001). To get the full advantage of the idea we define a Markov chain for the joint distribution for the state vector of interest, x, and the proposals. The resulting scheme allows moves that are not time reversible. Moreover, the use of antithetic variables can easily be included in the proposal distributions.

This paper is organised as follows. In Section 2 we define the joint Markov chain for the state vector of interest and the potential new states. This set-up leaves two quantities to be specified: the proposal distribution and the acceptance probabilities. In Sections 3 and 4 we discuss each of these aspects. In Section 5 we define the new estimator for μ and show it to be unbiased. Section 6 presents simulation examples and, finally, Section 7 provides conclusions.

2 MCMC with multiple proposals in each iteration

In this section we discuss how multiple proposals can be used to define a Markov chain with a given stationary distribution. Our set-up is a generalisation of the standard Metropolis–Hastings algorithm with one proposal in each iteration, see Smith and Roberts (1993) and Dellaportas and Roberts (2003) for nice introductions.

For definiteness, assume the target distribution to be continuous on $\Re^n$ and let $\pi(\cdot)$ denote its density. Let $m \geq 1$ be the number of proposals to be used in each iteration. To define the algorithm we use an integer valued stochastic variable $\kappa \in \{0, 1, \ldots, m\}$ and a set of stochastic vectors $y_0, y_1, \ldots, y_m \in \Re^n$. We also use the notations $\boldsymbol{y} = (y_0, y_1, \ldots, y_m)$ and $\boldsymbol{y}_{-j} = (y_0, \ldots, y_{j-1}, y_{j+1}, \ldots, y_m)$. Let κ have a uniform (marginal) distribution and let $p_\kappa(\boldsymbol{y})$ denote the conditional distribution for $\boldsymbol{y}$ given κ, i.e. the joint distribution for κ and $\boldsymbol{y}$ is

$$p(\boldsymbol{y}, \kappa) = \frac{1}{m+1} p_\kappa(\boldsymbol{y}). \tag{1}$$

Let $p_\kappa(\boldsymbol{y})$ be given from $\pi(\cdot)$ by

$$p_\kappa(\boldsymbol{y}) = \pi(y_\kappa) q_\kappa(\boldsymbol{y}_{-\kappa} | y_\kappa), \tag{2}$$

where $q_k(\cdot|y_k), k = 0, 1, \ldots, m$ is a set of "proposal" densities, i.e. for each $k \in \{0, 1, \ldots, m\}$ and $y_k \in \Re^n$, $q_k(\cdot|y_k)$ is a probability density function for $\boldsymbol{y}_{-k} \in \Re^{nm}$. The proposal densities can be chosen quite freely, but we require sampling from the $q_k(\cdot|y_k)$'s to be easy. Note that, by construction, y_κ is distributed according to our target distribution.

We define a Markov chain with invariant distribution $p(\cdot, \cdot)$ by alternating between two types of updates: (i) substitute current values of $\boldsymbol{y}_{-\kappa}$ by new values sampled from $q_\kappa(\cdot|y_\kappa)$; and (ii) substitute the current value of κ by a new value sampled according to a $(m+1) \times (m+1)$ transition matrix $\mathbf{P}(\boldsymbol{y}) = [\mathrm{P}_{k,l}(\boldsymbol{y})]_{k,l=0}^m$, i.e. if the current value of κ is k, the probability for the new value to be l is $\mathrm{P}_{k,l}(\boldsymbol{y})$. As indicated in the notation, the transition matrix $\mathbf{P}(\boldsymbol{y})$ is a function of $\boldsymbol{y}$. Step (i) above is a Gibbs step (Chib and Greenberg, 1995) for $\boldsymbol{y}_{-\kappa}$ and is clearly invariant with respect to $p(\cdot, \cdot)$. For step (ii) to be invariant with respect to $p(\cdot, \cdot)$ one must have

$$p_l(\boldsymbol{y}) = \sum_{k=0}^m p_k(\boldsymbol{y}) \mathrm{P}_{k,l}(\boldsymbol{y}) \;\text{ for } l \in \{0, 1, \ldots, m\} \text{ and } \boldsymbol{y} \in \Re^{n(m+1)}. \tag{3}$$

In addition one must of course have that $\mathbf{P}(\boldsymbol{y})$ is really a transition matrix for all $\boldsymbol{y}$, i.e.

$$\mathrm{P}_{k,l}(\boldsymbol{y}) \geq 0 \;\text{ for } k, l \in \{0, 1, \ldots, m\} \text{ and } \boldsymbol{y} \in \Re^{n(m+1)}, \text{ and} \tag{4}$$

$$\sum_{l=0}^m \mathrm{P}_{k,l}(\boldsymbol{y}) = 1 \;\text{ for } k \in \{0, 1, \ldots, m\} \text{ and } \boldsymbol{y} \in \Re^{n(m+1)}. \tag{5}$$

When $m = 1$, equation (3) is equivalent to detailed balance,

$$p_k(\boldsymbol{y})\mathrm{P}_{k,l}(\boldsymbol{y}) = p_l(\boldsymbol{y})\mathrm{P}_{l,k}(\boldsymbol{y}) \text{ for } k, l \in \{0, 1, \ldots, m\} \text{ and } \boldsymbol{y} \in \Re^{n(m+1)}, \tag{6}$$

whereas for $m > 1$ detailed balance is more restrictive than (3).

Step (i) above is the proposal step. The components of the new $\boldsymbol{y}_{-\kappa}$ should be considered as potential new states for y_κ. Step (ii) is the acceptance/rejection step. A change in κ corresponds to an acceptance, whereas an unchanged value for κ means that all proposals are rejected.

It remains to specify two important components of the Markov chain. The first is the choice of proposal distribution $q_k(\cdot|y_k)$. This is always an important issue in a Metropolis–Hastings algorithm and many possibilities for $m = 1$ are discussed in the MCMC literature, see discussions and references in Gamerman (1997), Chen et al. (2000) and Liu (2001). The case of several proposals in each iteration has got much less attention, but Liu et al. (2000) give some suggestions. We discuss this topic in Section 3. The second component that remains to be specified is the choice of transition matrix $\mathbf{P}(\boldsymbol{y})$. For $m = 1$, Peskun (1973) shows that one particular choice is optimal in asymptotic variance, so in most of the MCMC literature this choice is essentially a non-existing topic. For $m > 1$ the result in Peskun (1973) is still relevant, but does not produce a single optimal choice for $\mathbf{P}(\boldsymbol{y})$. We discuss this further in Section 4.

MCMC is used as an instrument to estimate the mean of one or several functions with respect to the target distribution. Assume our mean of interest is

$$\mu = \mathrm{E}_\pi\{f(x)\} = \int f(x)\pi(x)\mathrm{d}x, \tag{7}$$

where $f(\cdot)$ is some scalar valued function. Let $\{(y_0^i, y_1^i, \ldots, y_m^i, \kappa^i)\}_{i=1}^N$ denote the simulated values from the above Markov chain, where the superscript specifies iteration number (after having discarded a burn-in period). Then $y_{\kappa^i}^i, i = 1, \ldots, N$ are samples from the target distribution and the traditional MCMC estimator for μ is

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^N f(y_{\kappa^i}^i). \tag{8}$$

Thus, to estimate μ the $y_{\kappa^i}^i$'s are given weight one and all the remaining y_j^i's are assigned weight zero. Considering that the target density at all y_j^i's are already computed in the Markov chain, this appears to be sub-optimal. One would expect improved estimators by assigning non-zero weights to all y_j^i's. The challenge is to choose weights that give an unbiased estimator. We pursue this idea in Section 5.

3 Proposal distribution $q_k(\cdot|y_k)$

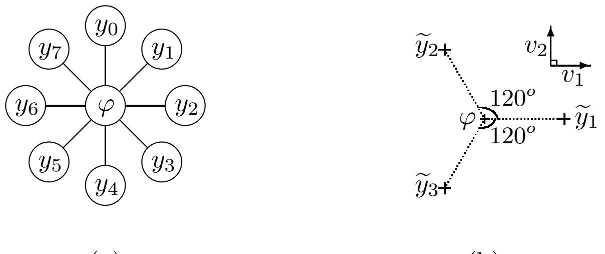
In this section we discuss two proposal distributions for $m \geq 1$. Both of our suggestions simplify to a random walk proposal for $m = 1$. Moreover, both alternatives are Metropolis-type proposals in that for any $\boldsymbol{y} \in \Re^{n(m+1)}$, $q_k(\boldsymbol{y}_{-k}|y_k) = q_0(\boldsymbol{y}_{-0}|y_0)$ for all $k = 1, \ldots, m$.

We specify $q_k(\cdot|y_k)$ by giving a corresponding algorithm for simulating $\boldsymbol{y}_{-k}$ for given k and y_k.

Proposal alternative 1 *(P1) Generate $\boldsymbol{y}_{-k}$ by (i) sampling $\varphi \sim N(y_k, \frac{1}{2}\sigma^2 I)$, and thereafter (ii) sampling $y_j \sim N(\varphi, \frac{1}{2}\sigma^2 I)$ independently for $j = 0, \ldots, k-1, k+1, \ldots, m$.*

Here $\mathrm{N}(\mu, \Sigma)$ denotes a multivariate Gaussian distribution with mean vector μ and covariance matrix Σ, I is the identity matrix and σ^2 is a parameter to be specified. The resulting dependence structure between the y_j's is illustrated in Figure 1(a). Note that for $m = 1$ this reduces to the common Gaussian random walk proposal.

Our second proposal distribution is a minor modification of the first. In *P1* the different $x_j - \varphi$'s are independent. Now we in stead assume them all to have equal lengths and let their directions be maximally spread. We give the proposal for $m = 2$ and $n \geq 2$, but the idea can easily be generalised to other values of $m \leq n$. Letting $\boldsymbol{y} = (y_0, y_1, y_2)$ and $\widetilde{\boldsymbol{y}} = (\widetilde{y}_0, \widetilde{y}_1, \widetilde{y}_2)$ denote the old and new values, respectively, Figure 1(b) illustrates the following choice of $q_k(\cdot|y_k)$.



(a) (b)

Figure 1: (a) Conditional independence structures in *P1* for $m = 7$. (b) Illustration of $q_k(\cdot|y_k)$ for Proposal alternative *P2*.

Proposal alternative 2 *(P2) Generate $\boldsymbol{y}_{-k}$ by first (i) sampling $u_1, u_2 \sim$ Unif(S^{n-1}) independently, and set $v_1 = u_1$ and $v_2 = \{u_2 - u_1(u_1^T u_2)\}/ \parallel u_2 - u_1(u_1^T u_2) \parallel$. Thereby v_1 and v_2 are orthogonal and of unit lengths. Next, (ii) draw $L \sim N(0, \frac{\sigma^2}{3})$, (iii) set $\widetilde{y}_k = y_k$ and (iv) let $\widetilde{\boldsymbol{y}}_{-k}$ and $\varphi \in \Re^n$ be defined from the three relations $\widetilde{y}_1 = \varphi + Lv_1$, $\widetilde{y}_2 = \varphi - \frac{L}{2}v_1 + \frac{\sqrt{3}L}{2}v_2$ and $\widetilde{y}_3 = \varphi - \frac{L}{2}v_1 - \frac{\sqrt{3}L}{2}v_2$.*

Here Unif(S^{n-1}) denotes a uniform density on the unit hyper-sphere. We use the factor $\frac{1}{3}$ in the variance for L to obtain $\widetilde{y}_i - \widetilde{y}_j \sim \text{N}(0, \sigma^2)$ for $i \neq j$.

4 Transition matrix $\mathbf{P}(\boldsymbol{y})$

As discussed in Section 2, the choice of transition matrix $\mathbf{P}(\boldsymbol{y})$ is typically taken as granted when $m = 1$. For $m > 1$ the situation is less clear. In this section we discuss three strategies for its choice. The simplest alternative is the following.

Transition alternative 1 *(T1) Set*

$$P_{k,l}(\boldsymbol{y}) = \frac{p_l(\boldsymbol{y})}{\sum_{j=0}^{m} p_j(\boldsymbol{y})}. \tag{9}$$

It is easily verified that this choice fulfils requirements (3), (4) and (5). It also fulfils detailed balance, equation (6). For $m = 1$ this corresponds to Barker's (1965) acceptance probability, which is known to be sub-optimal in asymptotic variance (Peskun, 1973). This turns out to be true also for $m > 1$, as the above $\mathbf{P}(\boldsymbol{y})$ can be Peskun improved by the following property.

Theorem 1 *For an $(m+1) \times (m+1)$ matrix $\mathbf{Q} = [Q_{k,l}]_{k,l=0}^{m}$ and an $(m+1)$ vector $\boldsymbol{\pi} = [\pi_0, \pi_1, \ldots, \pi_m]^T$ assume*

$$\pi_k Q_{k,l} = \pi_l Q_{l,k} \quad \textit{for } k, l \in \{0, 1, \ldots, m\}. \tag{10}$$

For some $u \in \Re$ and a set $A \subseteq \{0, 1, \ldots, m\}$, define the $(m+1) \times (m+1)$ matrix $\widetilde{\mathbf{Q}} = [\widetilde{Q}_{k,l}]_{k,l=0}^{m}$ from $\mathbf{Q}$ by

$$\widetilde{Q}_{k,l} = Q_{k,l} \quad \textit{if } k \notin A \textit{ or } l \notin A, \tag{11}$$

$$\widetilde{Q}_{k,l} = uQ_{k,l} \quad \textit{if } k, l \in A \textit{ and } k \neq l, \textit{ and} \tag{12}$$

$$\widetilde{Q}_{k,k} = 1 - \sum_{l \neq k} \widetilde{Q}_{k,l} \quad \textit{for } k \in A. \tag{13}$$

Then (10) holds also when $\mathbf{Q}$ is replaced by $\widetilde{\mathbf{Q}}$.

The proof is immediate. Our second transition matrix alternative is obtained be starting with the $\mathbf{P}(\boldsymbol{y})$ defined in *T1* and then use the above theorem repeatedly until at most one diagonal element is non-zero. In the following, $|A|$ denotes the number of elements in the set A.

Transition alternative 2 *(T2) Let $\mathbf{P}(\boldsymbol{y})$ by defined via the following process.*

1. *Set $\pi_k = p_k(\boldsymbol{y})/\sum_{j=0}^m p_j(\boldsymbol{y})$ for $k = 0, 1, \ldots, m$.*
2. *Set $t = 0$ and let $\mathbf{P}^0(\boldsymbol{y})$ be the transition matrix defined in (9).*
3. *Set $A^t = \{k : P^t_{k,k}(\boldsymbol{y}) > 0, k = 0, 1, \ldots, m\}$.*
4. *If $|A^t| \leq 1$, set $\mathbf{P}(\boldsymbol{y}) = \mathbf{P}^t(\boldsymbol{y})$ and stop the process.*
5. *Set*
$$u^t = \min_{k \in A^t} \left(\frac{1 - \sum_{l \notin A^t} P^t_{k,l}(\boldsymbol{y})}{\sum_{l \in A^t \setminus \{k\}} P^t_{k,l}(\boldsymbol{y})} \right). \tag{14}$$
6. *Let $\mathbf{P}^{t+1}(\boldsymbol{y})$ be defined from (11), (12) and (13) by substituting in these equations $\mathbf{P}^t(\boldsymbol{y})$ for $\mathbf{Q}$, $\mathbf{P}^{t+1}(\boldsymbol{y})$ for $\widetilde{\mathbf{Q}}$, u^t for u, and A^t for A.*
7. *Assign $t = t + 1$ and goto 3.*

One should note that the choice of u^t in (14) ensures that all elements in $\mathbf{P}(\boldsymbol{y})$ are non-negative. For $m = 1$ the above process gives

$$\mathrm{P}_{k,l}(\boldsymbol{y}) = \min \left\{ 1, \frac{p_l(\boldsymbol{y})}{p_k(\boldsymbol{y})} \right\} \quad \text{for } k \neq l, \tag{15}$$

which we can recognise as the Peskun optimal Metropolis–Hastings acceptance probability. The above process defines all elements in $\mathbf{P}(\boldsymbol{y})$. When simulating the Markov chain one of course only needs the elements in row κ. These can easily be computed without computing the whole matrix $\mathbf{P}(\boldsymbol{y})$. This is computationally important if m is large.

In both of the above strategies for choosing $\mathbf{P}(\boldsymbol{y})$, the resulting transition matrices depend on $\boldsymbol{y}$ only through the $p_k(\boldsymbol{y})$'s. This does not necessarily need to be the case. One may for example want to assign high probabilities to transitions that correspond to large changes in y_κ. Let $d(k, l, y_k, y_l) \geq 0$ be a function that somehow measure the difference between y_k and y_l. The $d(\cdot, \cdot, \cdot, \cdot)$ may, but need not, define a metric on $\Re^n$. One may then let $\mathbf{P}(\boldsymbol{y})$ be the solution of the following linear programming problem.

Transition alternative 3 *(T3) Let $\mathbf{P}(\boldsymbol{y})$ be the matrix that minimises*

$$\sum_{k=0}^m \sum_{l=0}^m d(k, l, y_k, y_l) P_{k,l}(\boldsymbol{y}), \tag{16}$$

under constraints (3), (4) and (5).

This is of course only a viable alternative when m is not too large, as otherwise the computational cost of solving the minimisation problem will dominate the computation time. When $m = 1$ and $d(k, l, y_k, y_l)$ is strictly positive for $k \neq l$ and equal to zero for $k = l$, this strategy also produces the Peskun optimal Metropolis–Hastings acceptance probability. In contrast to *T1* and *T2*, this last approach may, for $m > 1$, generate transition matrices that do not fulfil detail balance condition.

5 Using all proposed states to estimate mean values

Assume we want to estimate μ, defined in (7). As in Section 2, let $\{\boldsymbol{y}^i, \kappa^i\}_{i=1}^N$, where $\boldsymbol{y}^i = (y_0^i, y_1^i, \ldots, y_m^i)$, denote the simulated Markov chain after having discarded a burn-in period. As an alternative estimator to (8), consider

$$\mu^\star = \frac{1}{N}\sum_{i=1}^N \left\{ \sum_{l=0}^m w_{\kappa^i,l}(\boldsymbol{y}^i) f(y_l^i) \right\}, \tag{17}$$

where $\boldsymbol{w}(\boldsymbol{y}) = [w_{k,l}(\boldsymbol{y})]_{k,l=0}^m$ is a weight matrix function. The following theorem gives sufficient conditions for $\mu^\star$ to be an unbiased estimator for μ.

Theorem 2 *Let μ and $\mu^\star$ be given by equations (7) and (17) respectively and let $\{\boldsymbol{y}^i, \kappa^i\}_{i=1}^N$ be a series of (possibly dependent) samples from $p(\cdot,\cdot)$, where $p(\cdot,\cdot)$ is defined by equations (1) and (2). For all $\boldsymbol{y} \in \Re^{n(m+1)}$, assume*

$$p_l(\boldsymbol{y}) = \sum_{k=0}^m p_k(\boldsymbol{y}) w_{k,l}(\boldsymbol{y}) \quad \text{for } l \in \{0,1,\ldots,m\}, \text{ and} \tag{18}$$

$$\sum_{l=0}^m w_{k,l}(\boldsymbol{y}) = 1 \quad \text{for } k \in \{0,1,\ldots,m\}. \tag{19}$$

Then $E\{\mu^\star\} = \mu$.

See Appendix A for a proof. One should note that requirements (18) and (19) are identical to (3) and (5) for $\mathbf{P}(\boldsymbol{y})$. Thus, the three strategies for choosing $\mathbf{P}(\boldsymbol{y})$ discussed in Section 4 can just as well be used for $\boldsymbol{w}(\boldsymbol{y})$. However, the elements of $\boldsymbol{w}(\boldsymbol{y})$ are not probabilities and thereby do not need to be non-negative. This gives extra freedom in the choice of $\boldsymbol{w}(\boldsymbol{y})$. For example, it is easily verified that

$$w_{k,l}(\boldsymbol{y}) = \frac{c\, p_l(\boldsymbol{y})}{\sum_{j=0}^m p_j(\boldsymbol{y})} \quad \text{for } k \neq l, \text{ and} \tag{20}$$

$$w_{k,k}(\boldsymbol{y}) = 1 - \sum_{l \neq k} w_{k,l}(\boldsymbol{y}) \tag{21}$$

fulfil the requirements of the above theorem for any $c \in \Re^n$. In simulation experiments we have found this to produce good results. In the rest of the paper we therefore restrict the attention to this choice of $\boldsymbol{w}(\boldsymbol{y})$. However, we do not claim it to be optimal in any sense. One should note that $\mu^\star$ is a generalisation of $\widehat{\mu}$, as $\mu^\star = \widehat{\mu}$ for $c = 0$. As $\mu^\star$ is a linear function of c, one can easily express $\text{Var}(\mu^\star)$ as a function of c and the covariance structure of the simulated Markov chain. Inserting (20) and (21) in (17) we get

$$\mu^\star = \frac{1}{N}\sum_{i=1}^N \left\{ g_1(\boldsymbol{y}^i, \kappa^i) + c g_2(\boldsymbol{y}^i, \kappa^i) \right\}, \tag{22}$$

where

$$g_1(\boldsymbol{y}, \kappa) = f(y_\kappa) \qquad \text{and} \qquad g_2(\boldsymbol{y}, \kappa) = \frac{\sum_{l \neq \kappa} p_l(\boldsymbol{y}) \{f(y_l) - f(y_\kappa)\}}{\sum_{l=0}^m p_l(\boldsymbol{y})}. \tag{23}$$

Defining the covariance functions

$$\gamma_{ij}(h) = \text{Cov}\left\{ g_i(\boldsymbol{y}^i, \kappa^i), g_j(\boldsymbol{y}^{i+h}, \kappa^{i+h}) \right\} \quad \text{for } i,j = 1,2 \tag{24}$$

we get, for large N,

$$\text{Var}(\mu^\star) \approx \frac{1}{N}\left\{ \sum_{h=-\infty}^{\infty} \gamma_{11}(h) + 2c \sum_{h=-\infty}^{\infty} \gamma_{12}(h) + c^2 \sum_{h=-\infty}^{\infty} \gamma_{22}(h) \right\}. \tag{25}$$

An optimal value for c can thus be found by minimising this expression. This gives

$$c_{\text{opt}} = -\frac{\sum_{h=-\infty}^{\infty} \gamma_{12}(h)}{\sum_{h=-\infty}^{\infty} \gamma_{22}(h)}. \tag{26}$$

The relative reduction in estimation variance by using $c = c_{\text{opt}}$ in stead of $c = 0$ is

$$\frac{\text{Var}(\mu^\star | c = 0) - \text{Var}(\mu^\star | c = c_{\text{opt}})}{\text{Var}(\mu^\star | c = 0)} = \frac{\left\{\sum_{h=-\infty}^{\infty} \gamma_{12}(h)\right\}^2}{\left\{\sum_{h=-\infty}^{\infty} \gamma_{11}(h)\right\} \cdot \left\{\sum_{h=-\infty}^{\infty} \gamma_{22}(h)\right\}}. \tag{27}$$

By estimating the covariance functions one gets estimates for c_{opt} and corresponding variance reduction. However, if the same Markov chain run is used first to estimate c_{opt} and thereafter to compute $\mu^\star$ for $c = \widehat{c}_{\text{opt}}$, the resulting $\mu^\star$ is not unbiased. A better alternative is to do two independent Markov chain runs, runs A and B say. One may then estimate c_{opt} from each of the runs, getting $\widehat{c}^A_{\text{opt}}$ and $\widehat{c}^B_{\text{opt}}$, and thereafter computing $\mu^\star_A$ from run A with $c = \widehat{c}^B_{\text{opt}}$ and $\mu^\star_B$ from run B with $c = \widehat{c}^A_{\text{opt}}$. Then both $\mu^\star_A$ and $\mu^\star_B$ are unbiased estimators for μ, and so is $\frac{1}{2}(\mu^\star_A + \mu^\star_B)$.

6 Simulation examples

To quantify the possible gain by using the strategies discussed above we present two simulation exercises. For simplicity we use multivariate Gaussian densities for the target density $\pi(\cdot)$. We vary the number of proposals m, the proposal distribution $q_k(\cdot|y_k)$, and the transition kernel $\mathbf{P}(\boldsymbol{y})$. For each case we simulate the Markov chain and use this to estimate c_{opt}, the corresponding estimation variance, $\text{Var}(\mu^\star | c = c_{\text{opt}})$, and the relative variance reduction by using $c = c_{\text{opt}}$ in stead of $c = 0$. To estimate these quantities we substitute the infinite sums in (25), (26) and (27) by corresponding finite sums over empirical covariance functions. For each covariance function we choose a lag cut-off value equal to the first lag where the empirical covariance is less than 0.005 times the lag zero covariance. Even if other and better procedures for estimating these infinite sums exist (Priestley, 1981; Geyer, 1992; Green and Han, 1992), this crude estimation procedure gives satisfactory results in our examples.

6.1 Example 1

Let $\pi(\cdot)$ be a five dimensional Gaussian distribution with zero mean and identity covariance matrix. We estimate the mean of two functions, $f_1(x) = x_1$ and $f_2(x) = x_1^2$, where x_1 denotes the first component of the vector x. Of course, the true mean values are $\mu_1 = \text{E}_\pi\{f_1(x)\} = 0$ and $\mu_2 = \text{E}_\pi\{f_2(x)\} = 1$. To simulate from $\pi(\cdot)$ we combine *P1* from Section 3 and *T2* from Section 4. We run the algorithm for all combinations of $\sigma^2 = (0.1t)^2; t = 1, \ldots, 30$ and $m = 2^t; t = 0, 1, \ldots, 7$. The results of summarised in Figure 2. The left and right columns give estimation results for μ_1 and μ_2, respectively. The plots in the upper row show estimated relative variance reduction, equation (27), as function of proposal standard deviation σ. The eight curves are, from bottom to top, for $m = 2^t; t = 0, 1, \ldots, 7$. Thus, for any fixed σ the gain of using $c = c_{\text{opt}}$ increases uniformly with m. For $m = 1$, the maximum gains are 26% and 33% for estimating μ_1 and μ_2, respectively. For $m = 128$, the improvements get as high as 64% and 76%. The second row in the figure shows estimated values for c_{opt} as function of σ. The curves are, from top to bottom for large σ, for $m = 2^t; t = 0, 1, \ldots, 7$. In the third row we plot, for each value of m, the value of σ that gave the smallest estimated value for $\text{Var}(\mu^\star)$ using $c = 0$ (dashed curve) and $c = c_{\text{opt}}$ (solid curve). We see that the optimal value for σ increases with m. Moreover, the optimal value for σ is somewhat larger for $c = c_{\text{opt}}$ than for $c = 0$. Note that for $m = 1$ the optimal value for σ follows from the results in Roberts et al. (1997). Finally, the two lower rows of Figure 2 show how the estimation variances varies with m. For each value of m we choose the best value for σ^2 and plot estimated values for $N\text{Var}(\mu^\star)$ (fourth row) and $Nm\text{Var}(\mu^\star)$ (fifth row). The upper (dashed) curves are for $c = 0$ and the lower (solid) curves for $c = c_{\text{opt}}$. In an ideal parallel implementation of the algorithm, where computation time per iteration is the same for all values of m, the plots

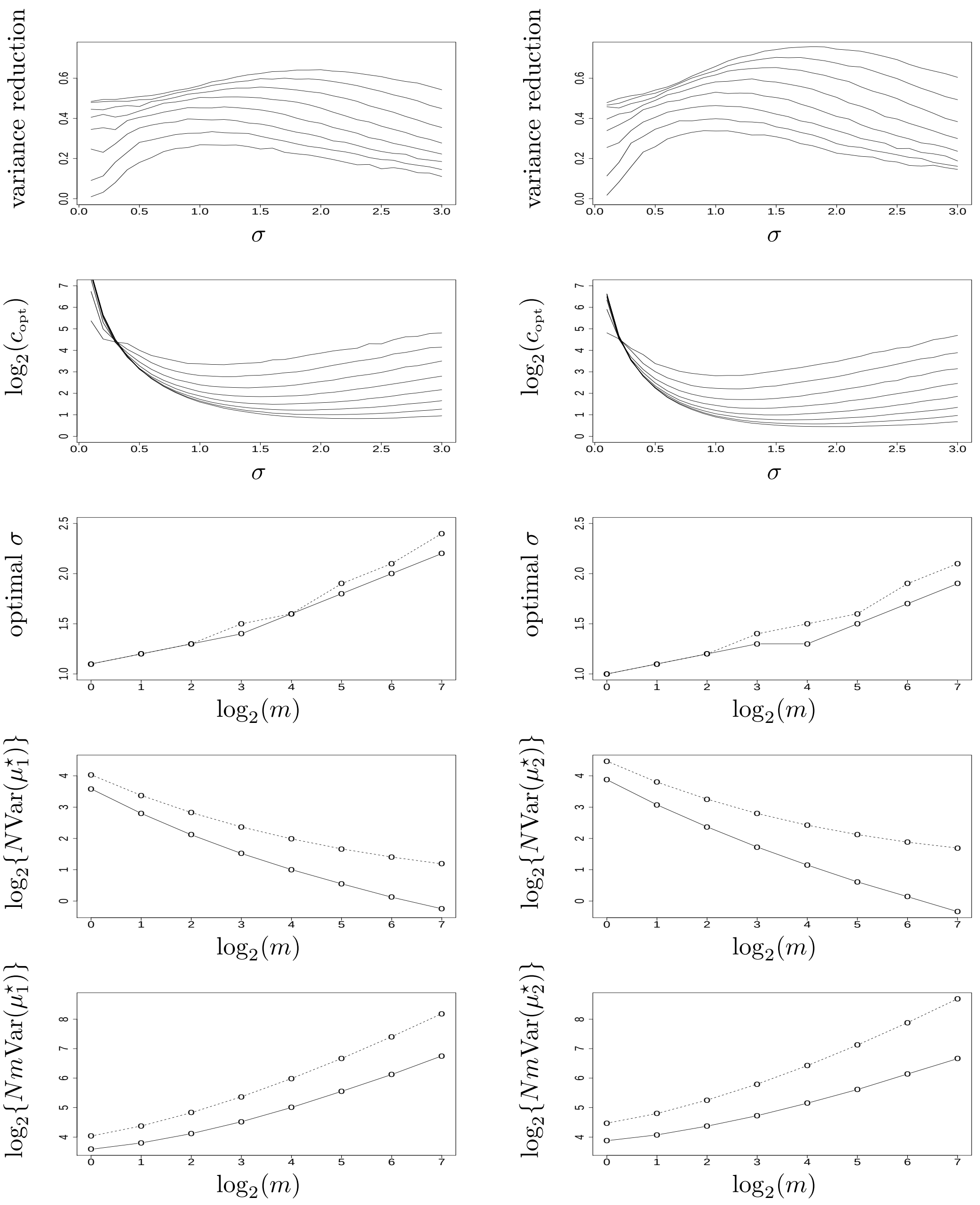


Figure 2: Summary of estimation results for Example 1. The left and right columns contain results for $f_1(x) = x_1$ and $f_2(x) = x_1^2$, respectively. First row: As a function of σ, estimated relative variance reduction by using $c = c_{\text{opt}}$ in stead of $c = 0$. From bottom to top, the eight curves are for $m = 2^t; t = 0, 1, \ldots, 7$. Second row: As a function of σ, estimated value for c_{opt}. From top down (for large values of σ), the eight curves are for $m = 2^t; t = 0, 1, \ldots, 7$. Third row: Estimated optimal values for σ as function of m. The upper (dashed) curves are for $c = 0$, the lower (solid) curves for $c = c_{\text{opt}}$. Forth row: Estimated $N\text{Var}(\mu^\star)$ as function of m. The upper (dashed) curves are for $c = 0$, the lower (solid) curves for $c = c_{\text{opt}}$. Fifth row: Estimated $Nm\text{Var}(\mu^\star)$ as function of m. The upper (dashed) curves are for $c = 0$, the lower (solid) curves for $c = c_{\text{opt}}$.

		$N\text{Var}(\mu_1^\star)$		$N\text{Var}(\mu_2^\star)$	
		T2	*T3*	*T2*	*T3*
$c = 0$	*P1*	10.381	9.871	13.918	13.311
	P2	10.312	9.775	13.941	13.195
$c = c_{\text{opt}}$	*P1*	6.971	6.929	8.421	8.425
	P2	6.881	6.870	8.356	8.333

Table 1: Summary of estimation results for Example 2. The left and right columns contains results for $f_1(x) = x_1$ and $f_2(x) = x_1^2$, respectively. Estimated values for $N\text{Var}(\mu^\star)$ are given for the combinations (*P1*,*T2*), (*P1*,*T3*), (*P2*,*T2*) and (*P2*,*T3*) for $c = 0$ and $c = c_{\text{opt}}$.

in the fourth row give the gain in estimation variance of using $m > 1$. For example, by using $m = 128$ and $c = c_{\text{opt}}$ the estimation variance when estimating μ_2 is approximately reduced with a factor $2^5 = 32$ relative to using $m = 1$ and $c = 0$. Of course, such a gain is not achievable in practice. First, sampling from $\mathbf{P}(\boldsymbol{y})$ is not suited for parallelisation. Second, parallelisation comes with an extra cost associated with communication between processors. With the computationally very cheap target distribution $\pi(\cdot)$ we are using in this example the gain from parallelisation would probably be quite limited, but with a computationally more expensive target density this would change. For a sequential implementation of the algorithm, where the computation time per iteration is approximately proportional to m, one should consider $Nm\text{Var}(\mu^\star)$ as function of m. This is shown in the lower row in Figure 2. As one would anticipate, the best alternative is then to use $m = 1$.

6.2 Example 2

Again we let $\pi(\cdot)$ be a five dimensional Gaussian distribution with zero mean and identity covariance matrix and focus on the two functions $f_1(x) = x_1$ and $f_2(x) = x_1^2$. We use $m = 2$ and get four different simulations algorithms by combining either of *P1* and *P2* with either of *T2* and *T3*. For *T3* we use $d(k, l, y_k, y_l) = 1$ if $k \neq l$ and $d(k, l, y_k, y_l) = 0$ otherwise. For example, with $p_0(\boldsymbol{y}) = 0.4$, $p_1(\boldsymbol{y}) = 0.35$ and $p_2(\boldsymbol{y}) = 0.25$ this gives

$$\mathbf{P}(\boldsymbol{y}) = \begin{bmatrix} 0 & 0.5508 & 0.4492 \\ 0.7990 & 0 & 0.2010 \\ 0.4814 & 0.5186 & 0 \end{bmatrix}. \tag{28}$$

This matrix can be compared with what one gets in this case by using *T2* in stead,

$$\mathbf{P}(\boldsymbol{y}) = \begin{bmatrix} 0.0833 & 0.5833 & 0.3333 \\ 0.6667 & 0 & 0.3333 \\ 0.5333 & 0.4667 & 0 \end{bmatrix}. \tag{29}$$

With the $d(\cdot, \cdot, \cdot, \cdot)$ used here, it follows easily that a diagonal element in $\mathbf{P}(\boldsymbol{y})$ produced from *T3* will always be smaller of equal to the corresponding diagonal element in the matrix generated by *T2*. However, as illustrated in the above example there will not necessarily exist any Peskun ordering (Mira, 2001) of the two transition matrices.

For each of the four combinations (*P1*,*T2*), (*P1*,*T3*), (*P2*,*T2*) and (*P2*,*T3*) we again simulate for $\sigma^2 = (0.1t)^2; t = 1, \ldots, 30$. In each case we estimate $N\text{Var}(\mu^\star)$ for $c = 0$ and for $c = c_{\text{opt}}$. In Table 1 we report the minimum (over different values of the proposal variance σ^2) estimated $N\text{Var}(\mu^\star)$. For $c = 0$ we see that *T3* gives lower estimation variances than *T2*. The same tendency seems to be present for $c = c_{\text{opt}}$, but now the differences are much smaller. For most cases the estimated variance for *P2* is also lower than the corresponding number for *P1*. However, the largest difference in estimation variance is clearly between $c = 0$ and $c = c_{\text{opt}}$.

7 Closing Remarks

In this paper we suggest to use all Metropolis–Hastings proposals to estimate mean values. We give sufficient conditions on the weights for the estimators to be unbiased and demonstrates in simulation examples that significant reduction in estimation variance can be achieved. With the possibility of using also rejected proposals for estimation it becomes more interesting to consider algorithms with many proposals in each iteration. In this paper we therefore also include a thorough discussion of this topic and extend the framework previously given for this.

In Section 3 we define two possible generalisations of the random walk proposal to a situation with multiple proposals in each iteration. An interesting problem is how to generalise other frequently used proposal strategies (for $m = 1$) to the multiple proposal case. For example, what is the natural generalisation of the Langevin–Metropolis–Hastings scheme (Grenander and Miller, 1994; Phillips and Smith, 1994; Roberts and Tweedie, 1996; Roberts and Rosenthal, 1998) to the multiple proposal case? Or what about the independent proposal Metropolis–Hastings algorithm? The "obvious" possibility for the latter is just to produce many independent proposals from the same proposal distribution, but we expect a better alternative is to generate dependent, negatively correlated, proposals. Our Proposal alternative *P2* is one example of how antithetic variables can be used in the proposal generation when $m > 1$. Even if the gain from this was quite limited in our second simulation example, we expect the use of antithetic variables to prove more fruitful in other cases.

Also the choice of transition matrix $\mathbf{P}(\boldsymbol{y})$ is important for the performance of an algorithm. We have discussed some possibilities, but we expect there are other and better choices. Last, but not least, the choice of weight matrix $\boldsymbol{w}(\boldsymbol{y})$ is important. Our simple choice in (20) and (21) give good results in the simulation examples. However, it is not optimal in any sense we suspect better alternatives can be found. One should also remember that the requirements in Theorem 2 are not necessary, just sufficient, conditions for $\mu^\star$ to be unbiased.

References

Barker, A. A. (1965). Monte Carlo calculations of the radial distribution functions for a proton-electron plasma, *Australian Journal of Physics* **18**: 119–133.

Casella, G. and Robert, C. P. (1996). Rao–Blackwellisation of sampling schemes, *Biometrika* **83**: 81–94.

Chen, M.-H., Shao, Q.-M. and Ibrahim, J. G. (2000). *Monte Carlo methods in Bayesian computation*, Springer, Berlin.

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

Dellaportas, P. and Roberts, G. O. (2003). An introduction to MCMC, *in* J. Møller (ed.), *Spatial Statistics and Computational Methods*, number 173 in *Lecture Notes in Statistics*, Springer, Berlin, pp. 1–41.

Gamerman, D. (1997). *Markov chain Monte Carlo: Stochastic simulation for Bayesian inference*, Chapman & Hall, London.

Geyer, C. (1992). Practical Markov chain Monte Carlo (with discussion), *Statistical Science* **7**: 473–511.

Green, P. J. and Han, X. L. (1992). Metropolis methods, Gaussian proposals, and antithetic variables, *in* P. Barone, A. Frigessi and M. Piccioni (eds), *Stochastic Models, Statistical Methods and Algorithms in Image Analysis*, number 74 in *Lecture Notes in Statistics*, Springer, Berlin, pp. 142–164.

Grenander, U. and Miller, M. I. (1994). Representations of knowledge in complex systems (with discussion), *J. R. Statist. Soc. B* **56**(4): 549–603.

Hastings, W. K. (1970). Monte Carlo simulation methods using Markov chains and their applications, *Biometrika* **57**: 97–109.

Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*, Springer, Berlin.

Liu, J. S., Liang, F. M. and Wong, W. H. (2000). The multiple-try method and local optimization in Metropolis sampling, *J. Am. Statist. Ass.* **95**: 121–134.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of state calculations by fast computing machines, *Journal of Chemical Physics* **21**: 1087–1092.

Mira, A. (2001). Efficiency of finite state space Monte Carlo Markov chains, *Statistics and Probability Letters* **54**: 405–411.

Peskun, P. H. (1973). Optimum Monte-Carlo sampling using Markov chains, *Biometrika* **60**: 607–612.

Phillips, D. B. and Smith, A. F. M. (1994). Bayesian model comparison via jump diffusions, *Technical report 94-20*, Imperial College of Science.

Priestley, M. B. (1981). *Spectral analysis and time series*, Academic, London.

Qin, Z. S. and Liu, J. S. (2001). Multi-point Metropolis method with application to hybrid Monte Carlo, *Journal of Computational Physics* **172**: 827–840.

Robert, C. P. and Casella, G. (1999). *Monte Carlo statistical methods*, Springer, Berlin.

Roberts, G. O., Gelman, A. and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms, *Annals of Applied Probability* **7**: 110–120.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions, *J. R. Statist. Soc. B* **60**: 255–268.

Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin diffusions and their discrete approximations, *Bernoulli* **2**: 341–363.

Smith, A. F. M. and Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods (with discussion), *J. R. Statist. Soc. B* **55**: 3–23.

A Proof of Theorem 2

To prove Theorem 2 we need to show

$$\mathrm{E}_p\left\{\sum_{l=0}^{m} w_{\kappa,l}(\boldsymbol{y})f(y_l)\right\} = \sum_{k=0}^{m}\int\sum_{l=0}^{m} w_{k,l}(\boldsymbol{y})f(y_l)\frac{1}{m+1}p_k(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = \mu. \tag{30}$$

Starting with the left hand side of this expression we get

$$\begin{aligned}
\mathrm{E}_p\left\{\sum_{l=0}^{m} w_{\kappa,l}(\boldsymbol{y})f(y_l)\right\} &= \frac{1}{m+1}\sum_{k=0}^{m}\int\sum_{l\neq k} w_{k,l}(\boldsymbol{y})f(y_l)p_k(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \\
&\quad + \frac{1}{m+1}\sum_{k=0}^{m}\int\left\{1-\sum_{l\neq k} w_{k,l}(\boldsymbol{y})\right\}f(y_k)p_k(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \\
= \mu + \frac{1}{m+1}\int\sum_{k=0}^{m}\sum_{l\neq k} w_{k,l}(\boldsymbol{y})f(y_l)p_k(\boldsymbol{y})\mathrm{d}\boldsymbol{y} &- \frac{1}{m+1}\int\sum_{k=0}^{m}\sum_{l\neq k} w_{k,l}(\boldsymbol{y})f(y_k)p_k(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \\
= \mu + \frac{1}{m+1}\int\sum_{k=0}^{m}\sum_{l\neq k} w_{k,l}(\boldsymbol{y})f(y_l)p_k(\boldsymbol{y})\mathrm{d}\boldsymbol{y} &- \frac{1}{m+1}\int\sum_{k=0}^{m}\sum_{l\neq k} w_{l,k}(\boldsymbol{y})f(y_l)p_l(\boldsymbol{y})\mathrm{d}\boldsymbol{y},
\end{aligned}$$

where, to get the last expression, we have interchanged the summation indices k and l in the last double sum. Thus, a sufficient condition for (30) to hold true is

$$\sum_{k\neq l} w_{k,l}(\boldsymbol{y})p_k(\boldsymbol{y}) = \sum_{k\neq l} w_{l,k}(\boldsymbol{y})p_l(\boldsymbol{y}) \tag{31}$$

for all $l = 0, 1, \ldots, m$ and $\boldsymbol{y} \in \Re^{n(m+1)}$. This equation is however, using (19), equivalent to

$$p_l(\boldsymbol{y})\{1 - w_{l,l}(\boldsymbol{y})\} = \sum_{k\neq l} w_{k,l}(\boldsymbol{y})p_k(\boldsymbol{y}) \tag{32}$$

which in turn is equivalent to (18).