The Annals of Statistics
2003, Vol. 31, No. 3, 705–767

SLICE SAMPLING[1]

BY RADFORD M. NEAL

University of Toronto

Markov chain sampling methods that adapt to characteristics of the distribution being sampled can be constructed using the principle that one can sample from a distribution by sampling uniformly from the region under the plot of its density function. A Markov chain that converges to this uniform distribution can be constructed by alternating uniform sampling in the vertical direction with uniform sampling from the horizontal "slice" defined by the current vertical position, or more generally, with some update that leaves the uniform distribution over this slice invariant. Such "slice sampling" methods are easily implemented for univariate distributions, and can be used to sample from a multivariate distribution by updating each variable in turn. This approach is often easier to implement than Gibbs sampling and more efficient than simple Metropolis updates, due to the ability of slice sampling to adaptively choose the magnitude of changes made. It is therefore attractive for routine and automated use. Slice sampling methods that update all variables simultaneously are also possible. These methods can adaptively choose the magnitudes of changes made to each variable, based on the local properties of the density function. More ambitiously, such methods could potentially adapt to the dependencies between variables by constructing local quadratic approximations. Another approach is to improve sampling efficiency by suppressing random walks. This can be done for univariate slice sampling by "overrelaxation," and for multivariate slice sampling by "reflection" from the edges of the slice.

1. Introduction. Markov chain methods such as Gibbs sampling and the Metropolis algorithm can be used to sample from many of the complex, multivariate distributions encountered in statistics. However, to implement Gibbs sampling, one may need to devise methods for sampling from nonstandard univariate distributions, and to use the Metropolis algorithm, one must find an appropriate "proposal" distribution that will lead to efficient sampling. The need for such special tailoring limits the routine use of these methods and inhibits the development of software that automatically constructs Markov chain samplers from model specifications. Furthermore, many common Markov chain samplers are inefficient, due to a combination of two flaws. First, they may try to make changes that are not well adapted to the local properties of the density function,

Received September 2000; revised July 2001.

[1]Supported in part by the Natural Sciences and Engineering Research Council of Canada and by the Institute for Robotics and Intelligent Systems.

AMS 2000 subject classifications. 65C60, 65C05.

Key words and phrases. Markov chain Monte Carlo, auxiliary variables, adaptive methods, Gibbs sampling, Metropolis algorithm, overrelaxation, dynamical methods.

with the result that changes must be made in small steps. Second, these small steps take the form of a random walk, in which about n^2 such steps are needed in order to move a distance that could be traversed in only n steps if these steps moved consistently in one direction.

In this paper, I describe a class of "slice sampling" methods that can be applied to a wide variety of distributions. Simple forms of univariate slice sampling are an alternative to Gibbs sampling that avoids the need to sample from nonstandard distributions. These slice sampling methods can adaptively change the scale of changes made, which makes them easier to tune than Metropolis methods and also avoids problems that arise when the appropriate scale of changes varies over the distribution. More complex slice sampling methods can adapt to the dependencies between variables, allowing larger changes than would be possible with Gibbs sampling or simple Metropolis methods. Slice sampling methods that improve sampling by suppressing random walks can also be constructed.

Slice sampling originates with the observation that to sample from a univariate distribution, we can sample points uniformly from the region under the curve of its density function and then look only at the horizontal coordinates of the sample points. A Markov chain that converges to this uniform distribution can be constructed by alternately sampling uniformly from the vertical interval defined by the density at the current point and from the union of intervals that constitutes the horizontal "slice" though the plot of the density function that this vertical position defines. If this last step is difficult, we may substitute some other update that leaves the uniform distribution over the slice invariant. To sample from a multivariate distribution, such single-variable slice sampling updates can be applied to each variable in turn. Section 4 presents these single-variable slice sampling methods.

We can also apply the slice sampling approach to a multivariate distribution directly, as described in Section 5, by sampling uniformly under the multidimensional plot of its density function. As for a univariate distribution, this can be done by alternately sampling uniformly from the vertical interval from zero up to the density at the current point and then uniformly from the slice defined by this vertical position. When the slice is high-dimensional, how to sample efficiently from it is less obvious than for single-variable slice sampling, but one gains the possibility of sampling in a way that respects the dependencies between variables. I show how, in the context of slice sampling, the way changes are proposed can be adapted to respect these dependencies, based on local information about the density function. In particular, local quadratic approximations could be constructed, as have been used very successfully for optimization problems. However, further research will be needed to fully exploit the adaptive capabilities of multivariate slice sampling.

One might instead accept that dependencies between variables will lead to the distribution being explored in small steps, but try at least to avoid exploring the distribution by an inefficient random walk, which is what happens when simple forms of the Metropolis algorithm are used. The benefits of random walk suppression are analyzed theoretically in some simple contexts by Diaconis,

Holmes and Neal (2000). Large gains in sampling efficiency can be obtained in practice when random walks are suppressed using the hybrid Monte Carlo or other dynamical methods [Duane, Kennedy, Pendleton and Roweth (1987), Horowitz (1991), Neal (1994, 1996)], or by using an overrelaxation method [Adler (1981), Barone and Frigessi (1990), Green and Han (1992), Neal (1998)]. Dynamical and overrelaxation methods are not always easy to apply, however. Use of Markov chain samplers that avoids random walks would be assisted by the development of methods that require less special programming and parameter tuning.

Two approaches to random walk suppression based on slice sampling are discussed in this paper. In Section 6, I show how one can implement an overrelaxed version of the single-variable slice sampling scheme. This may provide the benefits of Adler's (1981) Gaussian overrelaxation method for more general distributions. In Section 7, I describe slice sampling analogues of dynamical methods, which move around a multivariate slice using a stepping procedure that proceeds consistently in one direction while reflecting off the slice boundaries. These more elaborate slice sampling methods require more tuning than single-variable slice sampling, but they may still be easier to apply than alternative methods that avoid random walks.

I illustrate the benefits of the adaptive nature of slice sampling in Section 8, showing that it avoids disaster when sampling from a distribution typical of priors for hierarchical Bayesian models. Simple Metropolis methods can give the wrong answer for this problem, while providing little indication that anything is amiss.

I conclude with a discussion of the merits of the various slice sampling methods in comparison with other Markov chain methods and of their suitability for routine and automated use. Below, I set the stage by discussing general-purpose Markov chain methods that are currently in wide use.

2. General-purpose Markov chain sampling methods. Applications of Markov chain sampling in statistics often involve sampling from many distributions, such as posterior distributions for parameters of various different models, given various different datasets. For routine use of Markov chain methods, it is important to minimize the amount of effort that the data analyst must spend in order to sample from all these distributions. Ideally, a Markov chain sampler would be constructed automatically for each model and dataset.

The Markov chain method most commonly used in statistics is Gibbs sampling, popularized by Gelfand and Smith (1990). Suppose that we wish to sample from a distribution over n state variables (e.g., model parameters), written as $x = (x_1, \ldots, x_n)$, with probability density $p(x)$. Gibbs sampling proceeds by sampling in succession from the conditional distributions for each x_i given the current values of the other x_j for $j \neq i$, with conditional densities written as $p(x_i|\{x_j\}_{j\neq i})$. Repetition of this procedure defines a Markov chain which leaves the desired distribution invariant, and which in many circumstances is ergodic [e.g., when $p(x) > 0$ for all x]. Running the Gibbs sampler for a sufficiently

long time will then produce a sample of values for x from close to the desired distribution, from which we can estimate the expectations of those functions of state that are of interest.

Gibbs sampling can be done only if we know how to sample from all the required conditional distributions. These sometimes have standard forms for which efficient sampling methods have been developed, but there are many models for which sampling from these conditional distributions requires the development of custom algorithms, or is infeasible in practice (e.g., for multilayer perceptron networks [Neal (1996)]). Note, however, that once methods for sampling from these conditional distributions have been found, no further tuning parameters need be set in order to produce the final Markov chain sampler.

The routine use of Gibbs sampling has been assisted by the development of adaptive rejection sampling (ARS) [Gilks and Wild (1992), Gilks (1992)], which can be used to sample efficiently from any conditional distribution whose density function is log concave, given only the ability to compute some function, $f_i(x_i)$, that is proportional to the conditional density, $p(x_i|\{x_j\}_{j\neq i})$ [the ability to also compute the derivative, $f_i'(x_i)$, is helpful, but not essential]. This method has been used for some time by the BUGS software [Thomas, Spiegelhalter and Gilks (1992)] to automatically generate Markov chain samplers from model specifications. The first step in applying ARS is to find points on each side of the mode of the conditional distribution. This will in general require a search, which will in turn require the choice of some length scale for an initial step. However, the burden of setting this scale parameter is lessened by the fact that a good value for it can be chosen "retrospectively," based on past iterations of the Markov chain, without invalidating the results, since the value chosen affects only the computation time, not the distribution sampled from.

The adaptive rejection Metropolis sampling (ARMS) method [Gilks, Best and Tan (1995)] generalizes ARS to conditional distributions whose density functions may not be log-concave. However, when the density is not log-concave, ARMS does not produce a new point drawn independently from the conditional distribution, but merely updates the current point in a fashion that leaves this distribution invariant. Also, when a conditional distribution is not log-concave, the points used to set up the initial approximation to it must not be chosen with reference to past iterations, as this could result in the wrong distribution being sampled [Gilks, Neal, Best and Tan (1997)]. The initial approximation must be chosen based only on prior knowledge (including any preliminary Markov chain sampling runs), and on the current values of the other variables. Unlike ARS, neither the current value of the variable being updated, nor any statistics collected from previous updates (e.g., the typical scale of changes) can be used. This hinders routine use of the method.

Another general way of constructing a Markov chain sampler is to perform Metropolis updates [Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (1953),

Hastings (1970)], either for each variable in turn, as with Gibbs sampling, or for all variables simultaneously. A Metropolis update starts with the random selection of a "candidate" state, drawn from a "proposal" distribution. The candidate state is then accepted or rejected as the new state of the Markov chain, based on the ratio of the probability densities of the candidate state and the current state. If the candidate state is rejected, the new state is the same as the old state.

A simple "random-walk" Metropolis scheme can be constructed based on a symmetric proposal distribution (e.g., Gaussian) that is centred on the current state. All variables could be updated simultaneously in such a scheme, or alternatively, one variable could be updated at a time. In either case, a scale parameter is required for each variable, in order to fix the width of the proposal distribution in that dimension. For the method to be valid, these scale parameters must not be set on the basis of past iterations, but rather only on the basis of prior knowledge (including preliminary runs), and the current values of too large a scale for the proposal distribution will result in a high rejection rate, while choosing too small a scale will result in inefficient exploration via a random walk with unnecessarily small steps. Furthermore, the appropriate scale for Metropolis proposals may vary from one part of the distribution to another, in which case no single value will produce acceptable results. Selecting a scale at random from some range can sometimes alleviate these problems, but at a large cost in wasted effort whenever the scale selected is inappropriate.

It is tempting to tune the Metropolis proposal distribution based on the rejection rate in past iterations of the Markov chain, but such "retrospective tuning" is not valid, since it can disturb the stationary distribution to which the process converges. Fixing the proposal distribution based on a preliminary run is allowed, but if the original proposal distribution was not good, such a preliminary run may not have sampled from the whole distribution, and hence may be a bad guide for tuning.

We therefore see that although Gibbs sampling and Metropolis methods have been used to do much useful work, there is a need for better methods that can be routinely applied in a wider variety of situations. One aim of this paper is to find variations on slice sampling that can be used to sample from any continuous distribution, given only the ability to evaluate a "black-box" function that is proportional to its density, and in some cases, to also evaluate the gradient of this function. For many distributions, these new methods will not sample more efficiently than Gibbs sampling or a well-designed Metropolis scheme, but the slice sampling methods will often require less effort to implement and tune. For some distributions, however, slice sampling can be much more efficient, because it can adaptively choose a scale for changes appropriate to the region of the distribution currently being sampled. Slice samplers that adapt in more elaborate ways, or that suppress random walks, can potentially be much faster than simple Metropolis methods or Gibbs sampling.

3. The idea of slice sampling. Suppose we wish to sample from a distribution for a variable, x, taking values in some subset of $\Re^n$, whose density is proportional to some function $f(x)$. We can do this by sampling uniformly from the $(n+1)$-dimensional region that lies under the plot of $f(x)$. This idea can be formalized by introducing an auxiliary real variable, y, and defining a joint distribution over x and y that is uniform over the region $U = \{(x, y) : 0 < y < f(x)\}$ below the curve or surface defined by $f(x)$. That is, the joint density for (x, y) is

$$p(x, y) = \begin{cases} 1/Z, & \text{if } 0 < y < f(x), \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $Z = \int f(x)\,dx$. The marginal density for x is then

$$p(x) = \int_0^{f(x)} (1/Z)\,dy = f(x)/Z \tag{2}$$

as desired. To sample for x, we can sample jointly for (x, y), and then ignore y.

Generating independent points drawn uniformly from U may not be easy, so we might instead define a Markov chain that will converge to this uniform distribution. Gibbs sampling is one possibility: We sample alternately from the conditional distribution for y given the current x, which is uniform over the interval $(0, f(x))$, and from the conditional distribution for x given the current y, which is uniform over the region $S = \{x : y < f(x)\}$, which I call the "slice" defined by y. Generating an independent point drawn uniformly from S may still be difficult, in which case we can substitute some update for x that leaves the uniform distribution over S invariant. Higdon (1996) has interpreted the standard Metropolis algorithm in these terms. Beyond this, however, reducing the problem to that of updating x so as to leave a *uniform* distribution invariant allows us to use various tricks that would not be valid for a nonuniform distribution.

Related methods have been used in the past. Chen and Schmeiser (1998) describe a method that samples from the distribution over the region U by moving in random directions. Their work shares with this paper the aim of finding a method that requires little or no tuning, and hence is suitable for use as a "black-box" sampler, when little is known of the distribution being sampled from. Unfortunately, Chen and Schmeiser do not achieve this goal: though they intend that causal users fix the a_r and b_r parameters of their sampler at default values, these parameters in fact play essentially the same role as the proposal width in a simple random-walk Metropolis algorithm, and will sometimes have to be tuned if reasonable performance is to be achieved. This tuning is harder than for simple Metropolis updates, since the optimal setting of these parameters depends not just on the properties of the distribution but also on the normalizing constant, Z. In their main example, this problem is lessened by the way they periodically adjust the normalization constant based on the current point, but this state-

dependent adjustment destroys the reversibility of the transitions, undermining the correctness of the algorithm.

The highly successful Swendsen–Wang algorithm for the Ising model can also be seen as an auxiliary variable method, which led to its generalization by Edwards and Sokal (1988). In their scheme, the density (or probability mass) function is proportional to a product of k functions: $p(x) \propto f_1(x) \cdots f_k(x)$. They introduce k auxiliary variables, $y_1, \ldots, y_k$, and define a joint distribution for $(x, y_1, \ldots, y_k)$ which is uniform over the region in which $0 < y_i < f_i(x)$ for $i = 1, \ldots, k$. Gibbs sampling, or some other Markov chain procedure, can then be used to sample for $(x, y_1, \ldots, y_k)$. The slice sampling procedure described above is a special case of this procedure, when there is a single auxiliary variable (i.e., $k = 1$). Besag and Green (1993) and Higdon (1996) have discussed applications in image analysis of these methods with $k > 1$.

Mira and Tierney (2002) have shown that these auxiliary variable methods, with one or with many auxiliary variables, are uniformly ergodic under certain conditions. Roberts and Rosenthal (1999) have shown that these methods are geometrically ergodic under weaker conditions, and have also found some quantitative convergence bounds. These results all assume that the sampler generates a new value for x that is uniformly drawn from S, independently of the old value, which is often difficult in practice.

Concurrently with the work reported here, Damien, Wakefield and Walker (1999) have viewed methods based on multiple auxiliary variables as a general approach to constructing Markov chain samplers for Bayesian inference problems. They illustrate how one can often decompose $f(x)$ into a product of k factors for which the intersection of the sets $\{x : y_i < f_i(x)\}$ is easy to compute. This leads to an easily implemented sampler, but convergence is slowed by the presence of many auxiliary variables. For example, for a model of k i.i.d. data points, one simple approach (similar to some examples of Damien, Wakefield and Walker) is to have one factor for each data point, whose product is the likelihood. (Suppose that prior is uniform, and so need not be represented in the posterior density.) For many models, $\{x : y_i < f_i(x)\}$ will be easy to compute when f_i is the likelihood from one data point. However, if this approach is applied to n data points that are modeled as coming from a Gaussian distribution with mean μ and variance 1, one can show that after the y_i are chosen, the allowable range for μ will have width of order $1/n$. Since the width of the posterior distribution for μ will be of order $1/\sqrt{n}$, and since the posterior will be explored by a random walk, the convergence time will be of order n. Gibbs sampling would, of course, converge in a single iteration when there is only one parameter, and the slice sampling methods of this paper would also converge very rapidly, for any n. Using a large number of auxiliary variables is a costly way to avoid difficult computations.

I therefore am concerned in this paper with methods based on slice sampling with a single auxiliary variable. So that these methods will be practical for a wide range of problems, they often use updates for x that do not produce a point drawn independently from the slice, S, but merely change x in some fashion that leaves the uniform distribution over S invariant. This allows the methods to be used for any continuous distribution, provided only that we can compute some function, $f(x)$, that is proportional to the density.

4. Single-variable slice sampling methods. Slice sampling is simplest when only one (real-valued) variable is being updated. This will of course be the case when the distribution of interest is univariate, but more typically, the single-variable slice sampling methods of this section will be used to sample from a multivariate distribution for $x = (x_1, \ldots, x_n)$ by sampling repeatedly for each variable in turn. To update x_i, we must be able to compute a function, $f_i(x_i)$, that is proportional to $p(x_i|\{x_j\}_{j\neq i})$, where $\{x_j\}_{j\neq i}$ are the values of the other variables.

To simplify notation, I will here write the single real variable being updated as x (with subscripts denoting different such points, not components of x). I will write $f(x)$ for the function proportional to the probability density of x. The single-variable slice sampling methods discussed here replace the current value, x_0, with a new value, x_1, found by a three-step procedure:

(a) Draw a real value, y, uniformly from $(0, f(x_0))$, thereby defining a horizontal "slice": $S = \{x : y < f(x)\}$. Note that x_0 is always within S.

(b) Find an interval, $I = (L, R)$, around x_0 that contains all, or much, of the slice.

(c) Draw the new point, x_1, from the part of the slice within this interval.

Step (a) picks a value for the auxiliary variable that is characteristic of slice sampling. Note that there is no need to retain this auxiliary variable from one iteration of the Markov chain to the next, since its old value is forgotten at this point anyway. In practice, it is often safer to compute $g(x) = \log(f(x))$ rather than $f(x)$ itself, in order to avoid possible problems with floating-point underflow. One can then use the auxiliary variable $z = \log(y) = g(x_0) - e$, where e is exponentially distributed with mean one, and define the slice by $S = \{x : z < g(x)\}$.

Steps (b) and (c) can potentially be implemented in several ways, which must of course be such that the resulting Markov chain leaves the distribution defined by $f(x)$ invariant. Figure 1 illustrates one generally applicable method, in which the interval is found by "stepping out," and the new point is drawn with a "shrinkage" procedure. Figure 2 illustrates an alternative "doubling" procedure for finding the interval. These and some other variations are described in detail next, followed by a proof that the resulting transitions leave the correct distribution invariant.

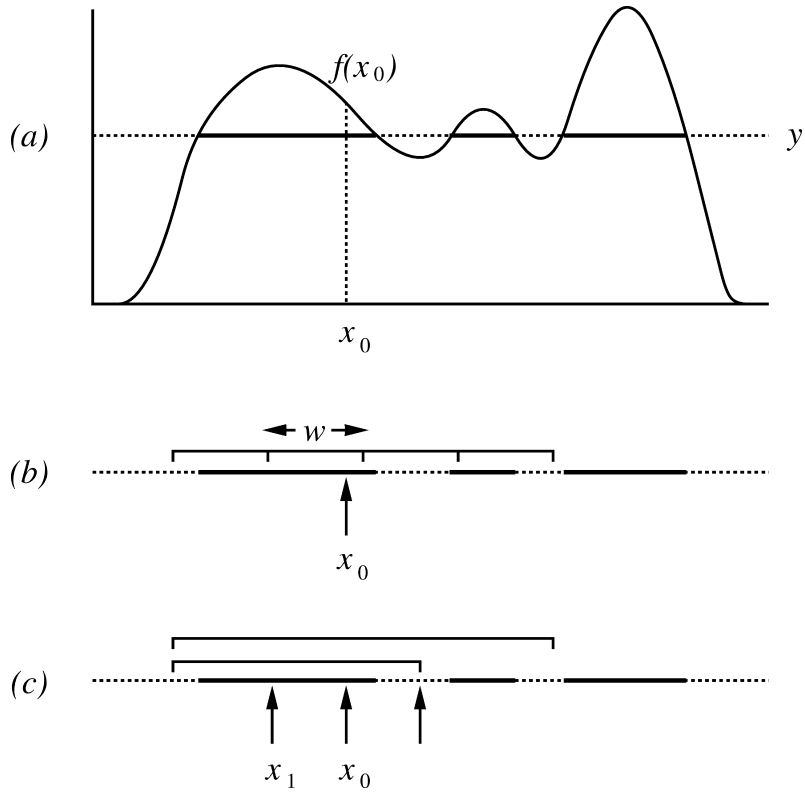


FIG. 1. *A single-variable slice sampling update using the stepping-out and shrinkage procedures. A new point,* x_1, *is selected to follow the current point,* x_0, *in three steps.* (a) *A vertical level,* y, *is drawn uniformly from* $(0, f(x_0))$, *and used to define a horizontal "slice," indicated in bold.* (b) *An interval of width* w *is randomly positioned around* x_0, *and then expanded in steps of size* w *until both ends are outside the slice.* (c) *A new point,* x_1, *is found by picking uniformly from the interval until a point inside the slice is found. Points picked that are outside the slice are used to shrink the interval.*

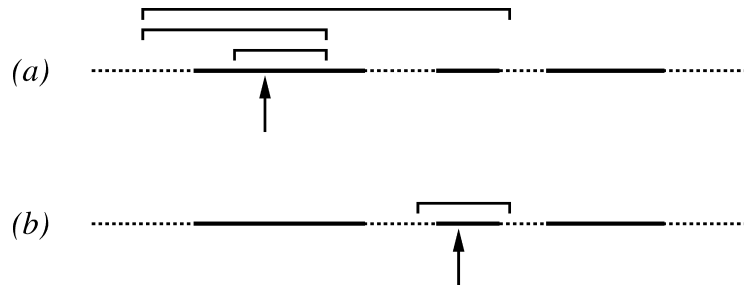


FIG. 2. *The doubling procedure. In* (a), *the initial interval is doubled twice, until both ends are outside the slice. In* (b), *where the start state is different, and the initial interval's ends are already outside the slice, no doubling is done.*

4.1. *Finding an appropriate interval.* After a value for the auxiliary variable has been drawn, defining the slice S, the next task is to find an interval $I = (L, R)$, containing the current point, x_0, from which the new point, x_1, will be drawn. We would like this interval to contain as much of the slice as is feasible, so as to allow the new point to differ as much as possible from the old point, but we would also like to avoid intervals that are much larger than the slice, as this will make the subsequent sampling step less efficient.

Several schemes for finding an interval are possible.

1. Ideally, we would set $L = \inf(S)$ and $R = \sup(S)$. That is, we would set I to the smallest interval that contains all of S. This may not be feasible, however.
2. If the range of possible values of x is bounded, we might let I be that range. However, this may not be good if the slice is typically much smaller than this.
3. Given an estimate, w, for the scale of S, we can randomly pick an initial interval of size w, containing x_0, and then perhaps expand it by a "stepping out" procedure.
4. Similarly, we can randomly pick an initial interval of size w, and then expand it by a "doubling" procedure.

For each scheme, we must also be able to find the set A of acceptable successor states, defined as follows:

$$(3)\quad A = \{x : x \in S \cap I \text{ and } P(\text{Select } I \mid \text{At state } x) = P(\text{Select } I \mid \text{At state } x_0)\}.$$

That is, A is the set of states from which we would be as likely to choose the interval I as we were to choose this I from the current state. When we subsequently sample from within I (see Section 4.2), we will ensure that the state chosen is in A, a fact that will be used in the proof of correctness in Section 4.3. Clearly, for schemes (1) and (2), $A = S$. For scheme (3), we will arrange that $A = S \cap I$. For scheme (4), a special test of whether a state is in A may be necessary.

Scheme (1), in which I is set to the smallest interval containing S, will be feasible when all solutions of $f(x) = y$ can be found analytically, or by an efficient and robust numerical method, but one cannot expect this in general. Often, even the number of disjoint intervals making up S will be hard to determine.

Scheme (2) is certainly easy to implement when the range of x is bounded, and we can always arrange this by applying a suitable transformation. However, if the slice is usually much smaller than the full range, the subsequent sampling (see Section 4.2) will be inefficient. This scheme has been used by Frey (1997).

The "stepping out" procedure [scheme (3) above] is appropriate for any distribution, provided that some rough estimate, w, for the typical width of the slice is available. The manner in which an interval is found by stepping out is illustrated in Figure 1(b) and the procedure is given in detail in Figure 3. The size of the interval found can be limited to mw, for some specified integer m, or the interval can be allowed to grow to any size (i.e., m can be set to infinity), in which case the procedure can be simplified in an obvious way. Simplification is

Input: f = function proportional to the density
x_0 = the current point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
m = integer limiting the size of a slice to mw
Output: (L, R) = the interval found

$U \sim \text{Uniform}(0, 1)$
$L \leftarrow x_0 - w * U$
$R \leftarrow L + w$
$V \sim \text{Uniform}(0, 1)$
$J \leftarrow \text{Floor}(m * V)$
$K \leftarrow (m - 1) - J$

repeat while $J > 0$ and $y < f(L)$:
  $L \leftarrow L - w$
  $J \leftarrow J - 1$

repeat while $K > 0$ and $y < f(R)$:
  $R \leftarrow R + w$
  $K \leftarrow K - 1$

FIG. 3. *The "stepping out" procedure for finding an interval around the current point. The notation* $U \sim \text{Uniform}(0, 1)$ *indicates that* U *is set to a number randomly drawn from the uniform distribution on* $(0, 1)$.

also possible when m is 1, in which case the interval will always be of size w, and there will be no need to evaluate f at its endpoints. Note that the random positioning of the initial interval and the random apportioning of the maximum number of steps m into a limit on going to the left and a limit on going to the right are essential for correctness, as they ensure that the final interval could equally well have been produced from any point within $S \cap I$.

The "doubling" procedure [scheme (4)] can expand the interval faster than the stepping out procedure, and hence may be more efficient when the estimated size of the slice (w) turns out to be too small. This procedure is illustrated in Figure 2, and given in detail in Figure 4. Doubling produces a sequence of intervals, each

Input: f = function proportional to the density
x_0 = the current point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
p = integer limiting the size of a slice to $2^p w$
Output: (L, R) = the interval found

$U \sim \text{Uniform}(0, 1)$
$L \leftarrow x_0 - w * U$
$R \leftarrow L + w$
$K \leftarrow p$

repeat while $K > 0$
  and { $y < f(L)$ or $y < f(R)$ }:
  $V \sim \text{Uniform}(0, 1)$
  if $V < 1/2$ then $L \leftarrow L - (R - L)$
    else $R \leftarrow R + (R - L)$
  $K \leftarrow K - 1$

FIG. 4. *The "doubling" procedure for finding an interval around the current point. Note that it is possible to save some computation in second and later iterations of the loop, since only one of* $f(L)$ *and* $f(R)$ *will have changed from the previous iteration.*

twice the size of the previous one, until an interval is found with both ends outside the slice, or a predetermined limit is reached. Note that when the interval is doubled the two sides are not expanded equally. Instead just one side is expanded, chosen at random (irrespective of whether that side is already outside the slice). This is essential to the correctness of the method, since it produces a final interval that could have been obtained from points other than the current one. The set A of acceptable next states is restricted to those for which the same interval could have been produced, and is in general not all of $S \cap I$. This complicates the subsequent sampling somewhat, as described below.

4.2. *Sampling from the part of the slice within the interval.* Once an interval, $I = (L, R)$, has been found containing the current point, x_0, the final step of the single-variable slice sampling procedure is to randomly draw a new point, x_1, from within this interval. This point must lie within the set A of points acceptable as the next state of the Markov chain, defined in (3).

Two methods could be used to sample from I.

(i) Repeatedly sample uniformly from I until a point is drawn that lies within A.

(ii) Repeatedly sample uniformly from an interval that is initially equal to I, and which shrinks each time a point is drawn that is not in A, until a point within A is found.

Method (i) could be very inefficient, if A turns out to be a tiny portion of I. The shrinkage of the interval in method (ii) ensures that the expected number of points drawn will not be too large, making it a more appropriate method for general use.

The shrinkage procedure is shown in detail in Figure 5. Note that each rejected point is used to shrink the interval in such a way that the current point remains

Input:	f	= function proportional to the density
	x_0	= the current point
	y	= the vertical level defining the slice
	(L, R)	= the interval to sample from
Output:	x_1	= the new point

$\bar{L} \leftarrow L, \quad \bar{R} \leftarrow R$

Repeat:

  $U \sim \text{Uniform}(0, 1)$

  $x_1 \leftarrow \bar{L} + U * (\bar{R} - \bar{L})$

  if $y < f(x_1)$ and Accept(x_1) then

    exit loop

  if $x_1 < x_0$ then $\bar{L} \leftarrow x_1$

    else $\bar{R} \leftarrow x_1$

FIG. 5. *The "shrinkage" procedure for sampling from the interval. Accept* (x_1) *is notation for a test of whether a point,* x_1*, that is, within* $S \cap I$ *is an acceptable next state. If scheme* (1), (2) *or* (3) *was used for constructing the interval, all points within* $S \cap I$ *are acceptable. If the doubling procedure* [*scheme* (4)] *was used, the point must pass the test of Figure* 6.

Input: f = function proportional to the density
x_0 = the current point
x_1 = the possible next point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
(L, R) = the interval found by the doubling procedure, using w

Output: whether or not x_1 is acceptable as the next state

$\hat{L} \leftarrow L, \quad \hat{R} \leftarrow R$
$D \leftarrow$ false

repeat while $\hat{R} - \hat{L} > 1.1 * w$:
  $M \leftarrow (\hat{L} + \hat{R}) / 2$
  if { $x_0 < M$ and $x_1 \geq M$ } or
    { $x_0 \geq M$ and $x_1 < M$ } then
      $D \leftarrow$ true
  if $x_1 < M$ then $\hat{R} \leftarrow M$
    else $\hat{L} \leftarrow M$
  if D and $y \geq f(\hat{L})$ and $y \geq f(\hat{R})$ then
    the new point is not acceptable
The new point is acceptable if it is not rejected in the loop above

FIG. 6. *The test for whether a new point, x_1, that is, within $S \cap I$ is an acceptable next state, when the interval was found by the "doubling" procedure. The multiplication by* 1.1 *in the "while" condition guards against possible round-off error. The variable D tracks whether the intervals that would be generated from the new point differ from those leading to the current point. When they don't, time can be saved by omitting a check.*

within it. Since the current point is always within A, the interval used always contains acceptable points, ensuring that the procedure will terminate.

If the interval was found by scheme (1), (2), or (3), the set A is simply $S \cap I$. However, if the doubling procedure [scheme (4)] was used, A may be a smaller subset of $S \cap I$. This is illustrated in Figure 2. In 2(a), an interval is found by doubling an initial interval until both ends are outside the slice. A different starting point is considered in 2(b), one which might have been drawn from the interval found in 2(a). The doubling procedure terminates earlier starting from here, so this point is not in A. (Note that A is here defined conditional on the alignment of the initial interval.)

The Accept(x_1) predicate in Figure 6 tests whether a point in $S \cap I$ is in A when the doubling procedure [scheme (4)] was used. This procedure works backward through the intervals that the doubling procedure would pass through to arrive at I when starting from the new point, checking that none of them has both ends outside the slice, which would lead to earlier termination of the doubling procedure.

4.3. *Correctness of single-variable slice sampling.* To show that single-variable slice sampling is a correct procedure, we must show that each update leaves the desired distribution invariant. To guarantee convergence to this distribution, the resulting Markov chain must also be ergodic. This is not always true, but it is in those situations [such as when $f(x) > 0$ for all x] for which one can easily

show that Gibbs sampling is ergodic. I will not discuss the more difficult situations here.

To show invariance, we suppose that the initial state, x_0, is distributed according to $f(x)$. In step (a) of single-variable slice sampling, a value for y is drawn uniformly from $(0, f(x))$. The joint distribution for x_0 and y will therefore be as in equation (1). If the subsequent steps update x_0 to x_1 in a manner that leaves this joint distribution invariant, then when we subsequently discard y, the resulting distribution for x_1 will be the marginal of this joint distribution, which is the same as that defined by $f(x)$, as desired.

We therefore need only show that the selection of x_1 to follow x_0 in steps (b) and (c) of the single-variable slice sampling procedure leaves the joint distribution of x and y invariant. Since these steps do not change y, this is the same as leaving the conditional distribution for x given y invariant, and this conditional distribution is uniform over $S = \{x : y < f(x)\}$, the slice defined by y. We can show invariance of this distribution by showing that the updates satisfy detailed balance, which for a uniform distribution reduces to showing that the probability density for x_1 to be selected as the next state, given that the current state is x_0, is the same as the probability density for x_0 to be the next state, given that x_1 is the current state, for any states x_0 and x_1 within S.

In the process of picking a new state, various intermediate choices are made randomly. When the interval is found by the stepping out procedure of Figure 3, the alignment of the initial interval is randomly chosen, as is the division of the maximum number of intervals into those used to extend to the left and those used to extend to the right. For the doubling procedure of Figure 4, the alignment of the initial interval is random and the decisions whether to extend to the right or to the left are also made randomly. When sampling is done using the shrinkage procedure of Figure 5, zero or more rejected points will be chosen before the final point. Let r denote these intermediate random choices. I will prove that detailed balance holds for the entire procedure by showing the following stronger result:

$$\begin{aligned} &P(\text{next state} = x_1,\ \text{intermediate choices} = r \mid \text{current state} = x_0) \\ &\quad = P(\text{next state} = x_0,\ \text{intermediate choices} = \pi(r) \mid \text{current state} = x_1), \end{aligned} \tag{4}$$

where $\pi(r)$ is some one-to-one mapping that has Jacobian one (with regard to the real-valued variables), which may depend on x_0 and x_1. Integrating over all possible values for r then gives the desired result.

The mapping π used is as follows. If the interval I is found by the stepping out or doubling procedure, an intermediate value, U, will be generated by the procedure of Figure 3 or 4, and used to define the initial interval. We define π so that it maps the value U_0 chosen when the state is x_0 to the following U_1 when the state is x_1:

$$U_1 = \text{Frac}\big(U_0 + (x_1 - x_0)/w\big), \tag{5}$$

where $\text{Frac}(x) = x - \text{Floor}(x)$ is the fractional part of x. This mapping associates values that produce the same alignment of the initial interval. Note also that it has Jacobian one. If the stepping out procedure is used, a value for J is also generated, uniformly from the set $\{0, \ldots, m-1\}$. The mapping π associates the J_0 found when the state is x_0 with the following J_1 when the state is x_1:

$$J_1 = J_0 + (x_1/w - U_1) - (x_0/w - U_0). \tag{6}$$

Here, $(x_1/w - U_1) - (x_0/w - U_0)$ is an integer giving the number of steps (of size w) from the left end of the interval containing x_0 to the left end of the interval containing x_1. This is the amount by which we must adjust J_0 in order to ensure that if the interval found starting from x_0 grows to its maximum size, the associated interval found starting from x_1 will be identical. Similarly, if the doubling procedure of Figure 4 is used, the sequence of random decisions as to which side of the interval to expand is mapped by π to the sequence of decisions that would cause the interval expanding from x_1 to become identical to the interval expanding from x_0 when the latter first includes x_1, and to remain identical through further expansions. Note here that there is at most one way to obtain a given final interval by successive doublings from a given initial interval, and that the alignment of the initial intervals by the association of U_0 with U_1 ensures that doubling starting from x_1 can indeed lead to the same interval as found from x_0. Finally, to complete the definition, π maps the sequence of rejected points used to shrink the interval found from x_0 (see Figure 5) to the same sequence of points when x_1 is the start state.

It remains to show that with this definition of π, (4) does indeed hold, for all points x_0 and x_1, and all possible intermediate values r. The equation certainly holds when both sides are zero, so we can ignore situations where movement between x_0 and x_1 is impossible (in conjunction with the given intermediate values).

Consider first the probability (density) for producing the intermediate values that define the interval I. For the stepping out and doubling procedures, the values U_0 and U_1 (related by π) that are generated from x_0 and x_1 will certainly have the same probability density, since U is drawn from a uniform distribution. Similarly, for the stepping out procedure, the values J_0 and J_1 are drawn from a uniform distribution over $\{0, \ldots, m-1\}$, and hence have the same probability as long as J_0 and J_1 are both in this set, which will be true whenever movement between x_0 and x_1 is possible. For the doubling procedure, a sequence of decisions as to which side to extend is made, with all sequences of a given length having the same probability. Here also, the sequences associated by π will have the same probability, *provided* the same number of doublings are done starting from x_0 as from x_1. This need not be true in general, but if the sequence from x_1 is shorter, the test of Figure 6 will eliminate x_1 as a possible successor to x_0, and if the sequence from x_0 is shorter, x_1 will not be a possible successor because it will be outside the interval I found from x_0. Both sides of (4) will therefore be zero in this situation.

Note next that the intervals found by any of the schemes of Section 4.1 will be the same for x_0 as for x_1, when the intermediate values chosen are related by π, assuming a transition from x_0 to x_1 is possible. For the stepping out procedure, the maximum extent of the intervals will be the same because of the relationships between U_0 and U_1 and between J_0 and J_1. Furthermore, the actual intervals found by stepping out (limited by the maximum) must also be the same whenever a transition between x_0 and x_1 is possible, since if the interval starting from x_0 has reached x_1, expansion of both intervals will continue in the same direction until the outside of the slice or the maximum is reached, and likewise in the other direction. Similarly, the mapping π is defined to be such that if the interval found by the doubling procedure starting from x_0 includes x_1, the same interval would be found from x_1, provided the process was not terminated earlier (by both ends being outside the slice), in which case x_1 is not a possible successor (as it would be rejected by the procedure of Figure 6). Note also that since the set A is determined by I (for any start state), it too will be the same for x_0 as for x_1.

If we sample from this I by simple rejection [scheme (i) in Section 4.2], the state chosen will be uniformly distributed over A, so the probability of picking x_0 will be the same as that of picking x_1. If we instead use the shrinkage procedure [scheme (ii) in Section 4.2, detailed in Figure 5], we need to consider as intermediate values the sequence of rejected points that was used to narrow the interval (recall that under π this sequence is the same for x_0 as for x_1). The probability density for the first of these is clearly the same for both starting points, since I is the same. As the interval shrinks, it remains the same for both x_0 and x_1, since the rejection decisions (based on A) are the same, and since we need consider only the case where the same end of the interval is moved to the rejected point (as otherwise a transition between x_0 and x_1 in conjunction with these intermediate values would be impossible). The probability densities for later rejected points, and for the final accepted state, are therefore also the same.

This completes the proof. Various seemingly reasonable modifications, such as changing the doubling procedure of Figure 4 so as not to expand the interval on a side that is already outside the slice, would undermine the argument of the proof and hence cannot be used.

4.4. *Shortcuts for unimodal distributions.* Some shortcuts are, however, allowable when the distribution is unimodal, because the slice, S, is then guaranteed to consist of a single interval. The acceptance test in Figure 6 can be omitted, since one can show that it will always indicate that the new point is acceptable. The interval found by the doubling procedure can also be shrunk at the outset by setting its endpoints to the first point in each direction that was found to lie outside the slice, since for a unimodal distribution, this shrinkage cannot eliminate any points within the slice and hence will not change the distribution of the point selected.

If the distribution is known to be unimodal *and* no limit is imposed on the size of the interval found (i.e., m and p in Figures 3 and 4 are infinite), the estimate, w, for the typical size of a slice can be set on the basis of past iterations. One could, for example, set w to the average distance between the old and new points in past iterations. This is valid because the distribution of the new point does not depend on w in this situation, even though w influences how efficiently this new point is found. Indeed, when the distribution is known to be unimodal, one can use any method at all for finding an interval that contains the current point and has both ends outside the slice, as any such interval will lead to the new point finally chosen being drawn uniformly from the slice.

5. Multivariate slice sampling methods. Rather than sample from a distribution for $x = (x_1, \ldots, x_n)$ by applying one of the single-variable slice sampling procedures described above to each x_i in turn, we might try instead to apply the idea of slice sampling directly to the multivariate distribution. I will start by describing a straightforward generalization of the single-variable methods to multivariate distributions, and then describe a more sophisticated method, which can potentially allow for adaptation to the local dependencies between variables.

5.1. *Multivariate slice sampling with hyperrectangles.* We can generalize the single-variable slice sampling methods of Section 4 to methods for performing multivariate updates by replacing the interval $I = (L, R)$ by an axis-aligned hyperrectangle $H = \{x : L_i < x_i < R_i \text{ for all } i = 1, \ldots, n\}$. Here, L_i and R_i define the extent of the hyperrectangle along the axis for variable x_i.

The procedure for finding the next state, $x_1 = (x_{1,1}, \ldots, x_{1,n})$, from the current state, $x_0 = (x_{0,1}, \ldots, x_{0,n})$, parallels the single-variable procedure:

(a) Draw a real value, y, uniformly from $(0, f(x_0))$, thereby defining the slice $S = \{x : y < f(x)\}$.

(b) Find a hyperrectangle, $H = (L_1, R_1) \times \cdots \times (L_n, R_n)$, around x_0, which preferably contains at least a big part of the slice.

(c) Draw the new point, x_1, from the part of the slice within this hyperrectangle.

It would perhaps be ideal for step (b) to set H to the smallest hyperrectangle containing S, but this is unlikely to be feasible. When all the variables have bounded ranges we might set H to the entire space, but this may be inefficient, since S is likely to be much smaller. We may therefore have to be content with finding an H that contains the current point, x_0, but probably not all of S. We will need estimates, w_i, for the appropriate dimensions of H along each axis, which we might set to a common value, w, if we know nothing about the relative scales of the variables. The simplest way of finding H is then to randomly position a hyperrectangle with these dimensions, uniformly over positions that lead to H containing x_0. This generalizes the random positioning of the initial interval I for

single-variable slice sampling. The stepping out and doubling procedures do not generalize so easily, however. The goal of finding an interval whose endpoints are outside the slice would generalize to finding a hyperrectangle all of whose vertices are outside the slice, but since an n-dimensional hyperrectangle has 2^n vertices, we would certainly not want to test for this when n is large. The stepping out procedure seems to be too time consuming in any case, since one would need to step out in each of the n directions. The doubling procedure does generalize appropriately, and one could decide to stop doubling when a randomly drawn point picked uniformly from the current hyperrectangle is outside the slice. Here, however, I will consider only the simplest scheme, which is to use the randomly positioned hyperrectangle without any expansion, though it is then crucial that the w_i not be much smaller than they should be.

The shrinkage procedure of Figure 5 generalizes easily to multiple dimensions: the hyperrectangle is simply shrunk independently along each axis. Combining this with random positioning of H gives the multivariate slice sampling method shown in Figure 7(a), and given in detail in Figure 8. The validity of this method can be proved in the same way as was done for single-variable slice sampling in Section 4.3.

Although this simple multivariate slice sampling method is easily implemented, in one respect it works less well than applying single-variable slice sampling to each variable in turn. When each variable is updated separately, the interval for that variable will be shrunk only as far as needed to obtain a new value within the slice. The amount of shrinkage can be different for different variables. In contrast, the procedure of Figure 8 shrinks all dimensions of the hyperrectangle until a point inside the slice is found, even though the probability density may not vary rapidly in some of these dimensions, making shrinkage in these directions unnecessary.

One way to try to avoid this problem is illustrated in Figure 7(b). Rather than shrink all dimensions of the hyperrectangle when the last point chosen was outside the slice, we can instead shrink along only one axis, basing the choice on the gradient of $\log f(x)$, evaluated at the last point. Specifically, only the axis corresponding to variable x_i is shrunk, where i maximizes the following product:

$$(R_i - L_i)\,|G_i|, \tag{7}$$

where G is the gradient of $\log f(x)$ at the last point chosen. By multiplying the magnitude of component i of the gradient by the width of the hyperrectangle in this direction, we get an estimate of the amount by which $\log f(x)$ changes along axis i. The axis for which this change is thought to be largest is likely to be the best one to shrink in order to eliminate points outside the slice. Unfortunately, if this decision were based as well on whether the sign of the gradient indicates that $\log f(x)$ is increasing or decreasing as we move toward the current point, x_0, the shrinkage decision might be different if we were to shrink from the final accepted point, x_1, which would invalidate the method.

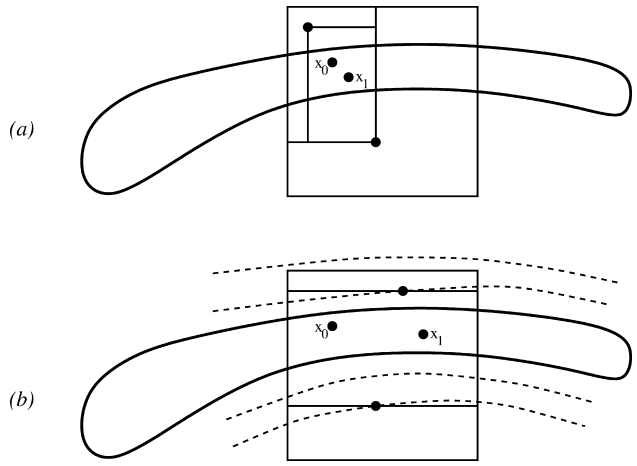


FIG. 7. *Multivariate slice sampling with hyperrectangles. The heavy line outlines the slice, containing the current point, x_0. The large square is the initial hyperrectangle. In* (a), *the hyperrectangle is shrunk in all directions when the point drawn is outside the slice, until a new point, x_1, inside the slice is found. In* (b), *the hyperrectangle is shrunk along only one axis, determined from the gradient and the current dimensions of the hyperrectangle. The dashed lines are contours of the density function, indicating the direction of the gradient.*

Many more elaborate schemes along these lines are possible. For instance, we might shrink along all axes for which the product (7) is greater than some

Input: f = function proportional to the density

x_0 = the current point, of dimension n

w_i = scale estimates for each variable, $i = 1, \ldots, n$

Output: x_1 = the new point

Step (a): *Find value of y that defines the slice*

$y \sim \text{Uniform}(0,\ f(x_0))$

Step (b): *Randomly position the hyperrectangle*

$H = (L_1, R_1) \times \cdots \times (L_n, R_n)$

For $i = 1$ to n:

$U_i \sim \text{Uniform}(0, 1)$
$L_i \leftarrow x_{0,i} - w_i * U_i$
$R_i \leftarrow L_i + w_i$

Step (c): *Sample from H, shrinking when points are rejected.*

Repeat:

For $i = 1$ to n:

$U_i \sim \text{Uniform}(0, 1)$
$x_{1,i} \leftarrow L_i + U_i * (R_i - L_i)$

if $y < f(x_1)$ then exit loop

For $i = 1$ to n:

if $x_{1,i} < x_{0,i}$ then $L_i \leftarrow x_{1,i}$
else $R_i \leftarrow x_{1,i}$

FIG. 8. *A simple multivariate slice sampling procedure, with randomly positioned hyperrectangle and shrinkage in all directions, as in Figure* 7(a).

threshold. A good scheme might preserve the ability of single-variable slice sampling to adapt differently for different variables, while keeping the advantages that simultaneous updates may sometimes have (e.g., in producing an ergodic chain when there are tight dependencies between variables).

More ambitiously, we might hope that a multivariate slice sampler could adapt to the dependencies between variables, not just to their different scales. This requires that we go beyond axis-aligned hyperrectangles, as is done in the next section.

5.2. *A framework for adaptive multivariate slice sampling.* We would like a more general framework by which trial points outside the slice that were previously rejected can be used to guide the selection of future trial points. In contrast to schemes based on hyperrectangles, we would like future trial points to potentially come from distributions that take account of the dependencies between variables. The scheme I present here achieves this by laying down a trail of "crumbs" that guides the selection of future trial points, leading eventually to a point inside the slice. A crumb can be anything, for example, a discrete value, a real number, a vector, a hyperrectangle, or a point in the state space being sampled from.

As with the previous slice sampling schemes, we start by choosing a value y uniformly between zero and $f(x_0)$, where x_0 is the current point. A crumb, c_1, is then drawn at random from some distribution with density (or probability mass) function $g_1(c; x_0, y)$. Note that this distribution may depend on both the current point, x_0, and on the value of y that defines the slice. A first trial point, x_1^*, is then drawn from the distribution with density $h_1(x^*; y, c_1) = g_1(c_1; x^*, y)/Z_1(y, c_1)$, where $Z_1(y, c_1) = \int g_1(c_1; x^*, y)\, dx^*$ is the appropriate normalizing constant. One can view x_1^* as being drawn from a pseudo-posterior distribution, based on a uniform prior, and the "data" that the first crumb was c_1. If x_1^* is inside the slice, it becomes the new state, and we are finished. Otherwise, a second crumb c_2, is drawn from some distribution $g_2(c; x_0, y, c_1, x_1^*)$, which may depend on the previous crumb and the previous trial point, as well as x_0 and y. The second trial point is then drawn from the pseudo-posterior distribution based on the "data" c_1 and c_2; that is, x_2^* is drawn from

$$h_2(x^*; y, c_1, x_1^*, c_2) = g_1(c_1; x^*, y) g_2(c_2; x^*, y, c_1, x_1^*)/Z_2(y, c_1, x_1^*, c_2), \tag{8}$$

where $Z_2(y, c_1, x_1^*, c_2) = \int g_1(c_1; x^*, y) g_2(c_2; x^*, y, c_1, x_1^*)\, dx^*$. If x_2^* is inside the slice, it becomes the new state. Otherwise, we draw a third crumb, from a distribution that may depend on the current state, the value defining the slice, the previous crumbs and the previous trial points, generate a third trial point using this and the previous crumbs, and so forth until a trial point lying within the slice is found.

The distributions of trial points in this sequence will become more and more concentrated, since they are pseudo-posterior distributions based on more and more pseudo-data (the crumbs). Since this pseudo-data is generated from the

current point, the concentration will be around this point, which is of course within the slice. The probability of a trial point being within the slice therefore increases towards one as the procedure progresses.

To show that this procedure leaves the distribution with density $f(x)/Z$ invariant, it suffices to show that it separately satisfies detailed balance with respect to transitions that occur in conjunction with any given number of crumbs being drawn. In the case, for instance, of transitions involving two crumbs, we can show this by showing the stronger property that for any x_1^* that is not in the slice defined by y and any x_2^* that is in this slice, the following will hold:

$$P(x_0)P(y, c_1, x_1^*, c_2, x_2^*|x_0) = P(x_2^*)P(y, c_1, x_1^*, c_2, x_0|x_2^*). \tag{9}$$

Here, $P(x_0)$ and $P(x_2^*)$ are the probability densities for the current point and the point that will become the new point [which are proportional to $f(x)$]. The conditional probabilities above are the densities for the given sequence of values being chosen during the procedure, given that the current point is the one conditioned on. The left-hand side of (9) can be written as follows:

$$\begin{aligned} &P(x_0)P(y|x_0)P(c_1|x_0, y)P(x_1^*|y, c_1)P(c_2|x_0, y, c_1, x_1^*)P(x_2^*|y, c_1, x_1^*, c_2) \\ &\quad = [f(x_0)/Z][1/f(x_0)]g_1(c_1; x_0, y)[g_1(c_1; x_1^*, y)/Z_1(y, c_1)] \\ &\qquad \times g_2(c_2; x_0, y, c_1, x_1^*)[g_1(c_1; x_2^*, y)g_2(c_2; x_2^*, y, c_1, x_1^*)/Z_2(y, c_1, x_1^*, c_2)]. \end{aligned}$$

The right-hand side is

$$\begin{aligned} &P(x_2^*)P(y|x_2^*)P(c_1|x_2^*, y)P(x_1^*|y, c_1)P(c_2|x_2^*, y, c_1, x_1^*)P(x_0|y, c_1, x_1^*, c_2) \\ &\quad = [f(x_2^*)/Z][1/f(x_2^*)]g_1(c_1; x_2^*, y)[g_1(c_1; x_1^*, y)/Z_1(y, c_1)] \\ &\qquad \times g_2(c_2; x_2^*, y, c_1, x_1^*)[g_1(c_1; x_0, y)g_2(c_2; x_0, y, c_1, x_1^*)/Z_2(y, c_1, x_1^*, c_2)]. \end{aligned}$$

These are equal, as is true in general for transitions involving any number of crumbs.

The hyperrectangle methods of Section 5.1 can be viewed in this framework. The randomly placed initial hyperrectangle is the first crumb. The first trial point is chosen from those points that could produce this initial hyperrectangle, which is simply the set of points within the hyperrectangle. The second and later crumbs are the shrunken hyperrectangles. Conditional on the current point, the previous crumb (i.e., the previous hyperrectangle), and the previous trial point, these later crumbs have degenerate distributions, concentrated on a single hyperrectangle. The possible corresponding trial points are the points within the shrunken hyperrectangle.

By using different sorts of crumbs, and different distributions for them, a huge variety of methods could be constructed within this framework. I discuss here only methods in which the crumbs are points in the state space, and have multivariate Gaussian distributions. The distributions of the trial points given the crumbs will then also be multivariate Gaussians.

In the simplest method of this sort, every g_i is Gaussian with mean x_0 and covariance matrix $\sigma^2 I$, for some fixed σ^2. The distribution, h_i, for x_i^* is then Gaussian with mean $\bar{c}_i = (c_1 + \cdots + c_i)/i$ and covariance matrix $(\sigma^2/i)I$. As more and more trial points are generated, they will come from narrower and narrower distributions, which will be concentrated closer and closer to the current point (since $\bar{c}_i$ will approach x_0). This is analogous to shrinkage in the hyperrectangle method. In practice, it is probably desirable to let σ_i^2 decrease with i (perhaps exponentially), so that the trial points would be forced closer to x_0 more quickly. Alternatively, we might look at $f(x_{i-1}^*)/y$ in order to estimate what value for σ_i would produce a distribution for the next trial point, x_i^*, that is likely to lie within the slice.

More generally, g_i could be a multivariate Gaussian with mean x_0 and some covariance matrix Σ_i, which may depend on the value of y, the previous crumbs, and the previous trial points. In particular, Σ_i could depend on the gradients of $f(x_j^*)$ for $j < i$, which provide information on what Gaussian distribution would be a good local approximation to $f(x)$. The distribution, h_i, for trial point x_i^* will then have covariance $\Sigma_i^* = [\Sigma_1^{-1} + \cdots + \Sigma_i^{-1}]^{-1}$ and mean $\bar{c}_i = \Sigma_i^* [\Sigma_1^{-1} c_1 + \cdots + \Sigma_i^{-1} c_i]$.

When x is of only moderate dimensionality, operations on covariance matrices would be fairly fast, and a wide variety of ways for producing Σ_i would be feasible. For higher-dimensional problems, such operations would need to be avoided, as is done in an optimization context with the conjugate gradient and related methods. Further research is therefore needed in order to fully exploit the potential of this promising framework for adaptation, and to compare it with methods based on the "delayed rejection" (also called "splitting rejection") framework of Tierney and Mira [Mira (1998), Chapter 5; Tierney and Mira (1999); Green and Mira (2001)].

6. Overrelaxed slice sampling. When variables are updated in ways that do not take account of their dependencies, changes must be small, and many updates will be needed to move from one part of the distribution to another. Sampling efficiency can be improved in this context by suppressing the random walk behavior characteristic of simple schemes such as Gibbs sampling. One way of achieving this is by using "overrelaxed" updates. Like Gibbs sampling, overrelaxation methods update each variable in turn, but rather than drawing a new value for a variable from its conditional distribution independently of the current value, the new value is instead chosen to be on the opposite side of the mode from the current value. In Adler's (1981) scheme, applicable when the conditional distributions are Gaussian, the new value for variable i is

$$x_i' = \mu_i + \alpha(x_i - \mu_i) + \sigma_i(1 - \alpha^2)^{1/2} n, \tag{10}$$

where μ_i and σ_i are the conditional mean and standard deviation of variable i, n is a Gaussian with mean zero and variance one, and α is a parameter slightly

greater than -1. This method is analyzed and discussed by Barone and Frigessi (1990) and by Green and Han (1992), though these discussions fail in some respects to elucidate the true benefits and limitations of overrelaxation: the crucial advantage being that it sometimes (though not always) suppresses random walks [Neal (1998)].

Various attempts have been made to produce overrelaxation schemes that can be used when the conditional distributions are not Gaussian. I have reviewed several such schemes and introduced one of my own [Neal (1998)]. The concept of overrelaxation seems to apply only when the conditional distributions are unimodal, so we may assume that this is usually the case, though we would like the method to at least remain valid (i.e., leave the desired distribution invariant) even if this assumption turns out to be false. To obtain the full benefits of overrelaxation, it is important that almost every update be overrelaxed, with few or no "rejections" that leave the state unchanged, as such rejections reintroduce an undesirable random walk aspect to the motion through state space [Neal (1998)].

In this section, I will show how overrelaxation can be done using slice sampling. Of the many possible schemes, I will describe only one in detail, based on stepping out and bisection, which is illustrated in Figure 9, and detailed in Figure 10.

To begin, we apply the stepping out procedure of Figure 3 to find an interval around the current point. Normally, we would set the maximum size of the interval (m) to infinity, since a proper overrelaxation operation requires that the entire slice be found, but the scheme remains valid for any m.

If the stepping out procedure finds an interval around the slice that is bigger than the initial interval, the two outermost steps will serve to locate the endpoints

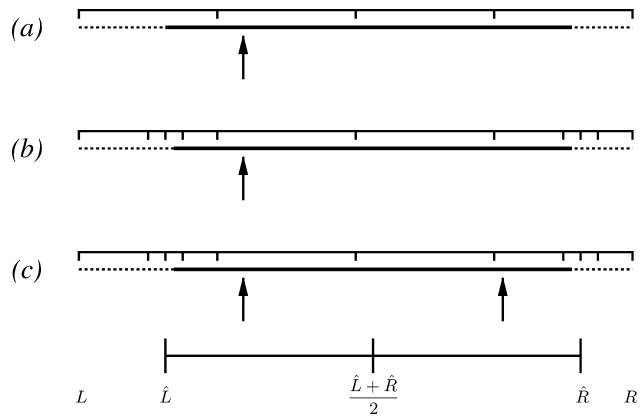


FIG. 9. *Overrelaxation using the stepping out procedure and bisection. In* (a), *an interval*, (L, R), *with both ends outside the slice is found by stepping out from the current point. In* (b), *the endpoints of the slice are located more accurately using bisection. In* (c), *a candidate point is found by flipping through the point half-way between the approximations to the endpoints. In this case*, *the candidate point is accepted*, *since it is within the slice*, *and within the interval prior to bisection.*

Input: f = function proportional to the density
x_0 = the current point
y = the vertical level defining the slice
w = estimate of the typical size of a slice
a = integer limiting endpoint accuracy to $2^{-a}\,w$
(L, R) = interval found by the stepping out procedure using stepsize w

Output: x_1 = the new point

$\bar{L} \leftarrow L, \quad \bar{R} \leftarrow R$
$\bar{w} \leftarrow w, \quad \bar{a} \leftarrow a$

When the interval is only of size w, narrow it until the mid-point is inside the slice (or accuracy limit is reached).

if $R - L < 1.1 * w$ then

repeat:

$M \leftarrow (\bar{L} + \bar{R})\,/\,2$
if $\bar{a} = 0$ or $y < f(M)$ then
exit loop
if $x_0 > M$ then $\bar{L} \leftarrow M$
else $\bar{R} \leftarrow M$
$\bar{a} \leftarrow \bar{a} - 1$
$\bar{w} \leftarrow \bar{w}\,/\,2$

Refine endpoint locations by bisection, to the specified accuracy.

$\hat{L} \leftarrow \bar{L}, \quad \hat{R} \leftarrow \bar{R}$

repeat while $\bar{a} > 0$:

$\bar{a} \leftarrow \bar{a} - 1$
$\bar{w} \leftarrow \bar{w}\,/\,2$
if $y \geq f(\hat{L} + \bar{w})$ then $\hat{L} \leftarrow \hat{L} + \bar{w}$
if $y \geq f(\hat{R} - \bar{w})$ then $\hat{R} \leftarrow \hat{R} - \bar{w}$

Find a candidate point by flipping from the current point to the opposite side of $(\hat{L}, \hat{R})$, then test it for acceptability.

$x_1 \leftarrow \hat{L} + \hat{R} - x_0$
if $x_1 < \bar{L}$ or $x_1 > \bar{R}$ or $y \geq f(x_1)$ then
$x_1 \leftarrow x_0$

FIG. 10. *The overrelaxation procedure using bisection.*

of the slice to within an interval of size w (assuming the slice consists of a single interval, as it will if the distribution is unimodal). We then locate the endpoints more precisely using a bisection procedure. For each endpoint, we test whether the midpoint of the interval in which it is located is inside or outside the slice and shrink this interval appropriately to narrow the location of the endpoint. After this is done a times, each endpoint will be known to lie in an interval of size $2^{-a}\,w$.

If the stepping out procedure finds that the initial interval (of size w) already has both ends outside the slice, then before doing any bisection, we narrow this interval, by shrinking it in half repeatedly until its midpoint is within the slice. We then use bisection as above to locate the endpoints to within an interval of size $2^{-a}\,w$.

After the locations of the endpoints have been narrowed down, we approximate the entire slice by the interval $(\hat{L}, \hat{R})$, formed from the outer bounds on the endpoint locations. To do an overrelaxed update, we flip from the current point, x_0, to a new point, x_1, that is the same distance as the current point from the middle of

this interval, but on the opposite side. That is, we let

$$x_1 = \frac{\hat{L} + \hat{R}}{2} - \left(x_0 - \frac{\hat{L} + \hat{R}}{2} \right) = \hat{L} + \hat{R} - x_0. \tag{11}$$

We must sometimes reject this candidate point, in which case the new point is the same as the current point. First, we must reject x_1 if it lies outside the interval, $(\bar{L}, \bar{R})$, that was found prior to bisection, since the interval found from x_1 would then be different, and detailed balance would not hold. However, this cannot happen when the distribution is unimodal. Secondly, we must reject x_1 if it lies outside the slice. This can happen even for a unimodal distribution, when the endpoints of the slice have not been located exactly. However, the probability of rejection for a unimodal distribution can be reduced to as low a level as desired, at moderate cost, by locating the endpoints more precisely using more iterations of bisection.

The correctness of this procedure can be seen using arguments similar to those of Section 4.3. The interval before bisection can be found by the doubling procedure instead of stepping out, provided the point found is rejected if it fails the acceptance test of Figure 6. However, rejection for this reason will not occur in the case of a unimodal distribution, which is presumably typical, since overrelaxation is likely inappropriate for a multimodal distribution.

Many methods other than bisection could be used to locate the endpoints before overrelaxing. If the derivative of $f(x)$ can easily be calculated, we could use Newton iteration, whose rapid convergence would often allow the endpoints to be calculated to machine precision in a few iterations. For unimodal distributions, such exact calculations would eliminate the possibility of rejection, and make the final result be independent of the way the interval containing the slice was found, thereby allowing use of retrospective methods for tuning the procedure for finding this interval.

To obtain a full sampling scheme, overrelaxed updates of this sort would be applied to each variable in turn, in a fixed order, for a number of cycles, after which a cycle of normal single-variable slice sampling updates would be done. Alternatively, each update could be done normally with some small probability. A Markov chain consisting solely of overrelaxed updates might not be ergodic (perhaps staying on one contour of the probability density), and might in any case suppress random walks for too long. The frequency of normal updates is a tuning parameter, analogous to the choice of α in Adler's overrelaxation method, and would ideally be set so that the Markov chain moves systematically, rather than in a random walk, for long enough that it traverses a distance comparable to the largest dimension of the multivariate distribution, but for no longer than this. To keep from doing a random walk for around k steps, one would do every kth update normally and also arrange for the rejection rate for the overrelaxed updates to be less than $1/k$.

7. Reflective slice sampling. Multivariate slice sampling methods can also be designed to suppress random walks. In this section I describe methods that "reflect" off the boundaries of the slice. Such movement with reflection can be seen as a specialization to uniform distributions of the Hamiltonian dynamics on which the hybrid Monte Carlo method of Duane, Kennedy, Pendleton and Roweth (1987) is based.

As before, suppose we wish to sample from a distribution over $\Re^n$, defined by a function $f(x)$ that is proportional to the probability density. We assume here that we can compute both $f(x)$ and its gradient. In each iteration of the Markov chain, we will draw a value for an auxiliary variable, y, uniformly from $(0, f(x))$, thereby defining an n-dimensional slice $S = \{x : y < f(x)\}$. We also introduce n additional "momentum" variables, written as a vector, p, which serve to indicate the current direction and speed of motion through state space. At the start of each iteration, we pick a value for p, independently of x, from some rotationally symmetric distribution, typically Gaussian with mean zero and identity covariance.

Once y and p have been drawn, we repeatedly update x by stepping in the direction of p. After some predetermined number of steps, we take the final value of x as our new state (provided it is acceptable). In each step, we try to set $x' = x + wp$, for some scale parameter w that determines the average step size. However, if the resulting x' is outside the slice S [i.e., $y \geq f(x')$], we must somehow try to bring it back inside. The schemes considered here all do this by some form of reflection, but differ in the exact procedure used.

Ideally, we would reflect from the exact point at which movement in the direction of p first takes us outside the slice. This reflection operation modifies p, after which motion continues in the new direction, until we again encounter the boundary of the slice. When we hit the boundary at a point where the gradient of $f(x)$ is h, reflection will change p as follows:

$$p' = p - 2h\frac{p \cdot h}{|h|^2}. \tag{12}$$

This ideal reflection scheme is illustrated for a two-dimensional slice in Figure 11. Using the fact that the reflection transformation above has Jacobian one and is its own inverse, one can show that movement with reflection for some predetermined duration leaves invariant the joint distribution of x (uniform within the slice) and p (rotationally symmetric, independent of x), so this way of sampling is valid, with no need for an acceptance test. One can also see from the figure how such motion can proceed consistently in one direction (until the end of the slice is reached), rather than in a random walk.

Ideal reflection may be difficult to implement, however, as it requires precise calculation of where the current path intersects the boundary of the slice. Finding this point analytically is sometimes possible, as in the application of reflective slice sampling by Downs, MacKay and Lee (2000). We might instead try to solve for the intersection point numerically, but if the slice is not known to be convex, it may be

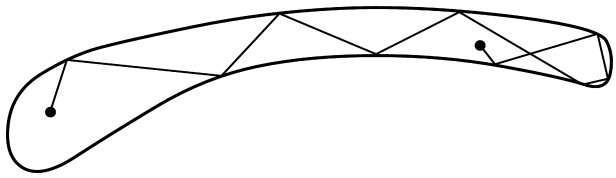


FIG. 11. *Moving around a two-dimensional slice by reflection from the exact boundaries.*

difficult even to determine with certainty that any intersection point that has been found is in fact the first one that would be encountered. Rather than attempt such exact calculations, we can instead employ one of two approximate schemes, based on "inside" or "outside" reflection, although the trajectories these schemes produce must sometimes be rejected in order to produce the exactly correct equilibrium distribution (in which case, the chain remains in the state from which the trajectory began).

When stepping from x to $x' = x + wp$ takes us outside the slice, we can try to reflect from the last inside point, x, instead of from the exact point where the path intersects the boundary, using the gradient of $f(x)$ at this inside point. The process is illustrated in Figure 12. However, for this method to be valid, we must check that the reverse trajectory would also reflect at this point, by verifying that a step in the direction opposite to our new heading would take us outside the slice. If this is not so, we must either reject the entire trajectory of which this reflection step forms a part, or alternatively, set p and x so that we retrace the path taken to this point (starting at the inside point where the reflection failed).

Alternatively, when we step outside the slice, we can try to reflect from the outside point, x', based on the gradient at that point. A trajectory with several such reflections is shown in Figure 13. After performing a predetermined number of

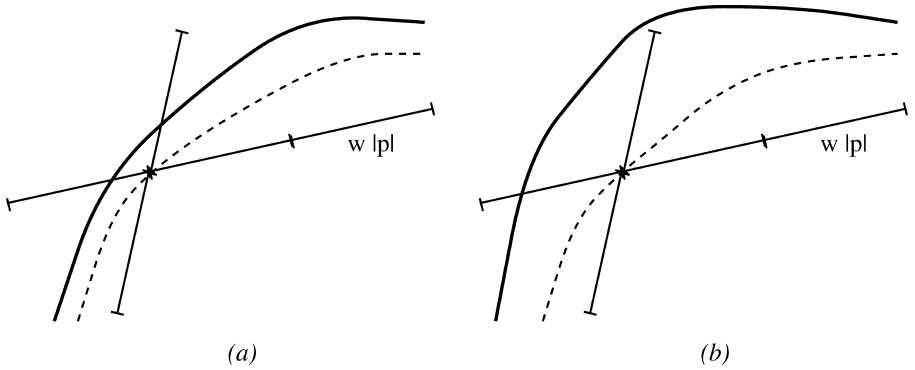


FIG. 12. *Reflection from an inside point. The trajectories here go in steps of size* $w|p|$, *starting from the top right, until a point outside the slice is reached, when a reflection is attempted based on the inner contour shown. In* (a), *the reflection is successful*; *in* (b), *it must be rejected, since the reverse trajectory would not reflect at this point.*

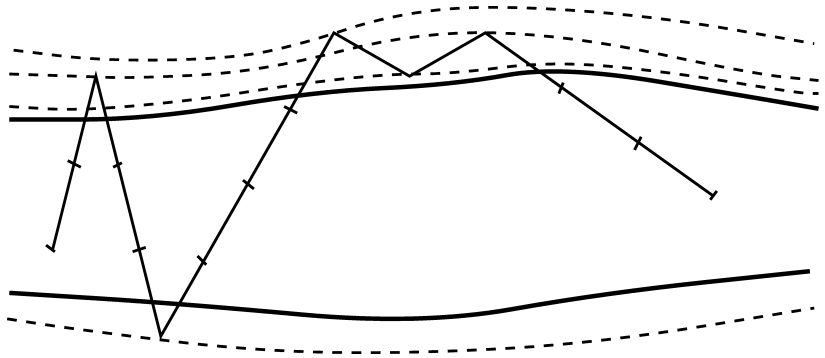


FIG. 13. *Reflection from outside points. Starting from the left, two reflections based on outside contours lead back inside the slice after the next step. The step after the third reflection is still outside the slice, so further reflections must be done. In this case, the trajectory eventually returns to the slice, and its endpoint would therefore be accepted.*

steps, we accept the trajectory if the final point is inside the slice. Note that for this method to be valid, one must reflect *whenever* the current point is outside the slice, even if this leads one away from the slice rather than toward it. This will sometimes result in the trajectory never returning to the slice, and hence being rejected, but other times, as in the figure, it does return eventually.

Many variations on these procedures are possible. Previously, it was assumed that values for y and p are randomly drawn at the beginning of a trajectory, and then kept the same for many steps (apart from changes to p due to reflections). When using inside reflection, we might instead pick a new value for y more often, perhaps before every step, and we might also partially update p, as is done in Horowitz's (1991) variation on hybrid Monte Carlo. When using outside reflection, the acceptance rate can be increased by terminating the trajectory when either some preset maximum number of steps have been taken, *or* some preset number of steps have ended inside the slice. When termination occurs for the latter reason, the final point will necessarily be inside the slice, and the trajectory will therefore be accepted.

8. A demonstration. To illustrate the benefits stemming from the adaptive nature of slice sampling, I show here how it can help avoid a disastrous scenario, in which a seriously wrong answer is obtained without any obvious indication that something is amiss.

The task is to sample from a distribution for ten real-valued variables, v and x_1 to x_9. The marginal distribution of v is Gaussian with mean zero and standard deviation 3. Conditional on a given value of v, the variables x_1 to x_9 are independent, with the conditional distribution for each being Gaussian with mean zero and variance e^v. The resulting shape resembles a ten-dimensional funnel, with small values for v at its narrow end, and large values for v at its wide end. Such

a distribution is typical of priors for components of Bayesian hierarchical models: x_1 to x_9 might, for example, be random effects for nine subjects, with v being the log of the variance of these random effects. If the data happens to be largely uninformative, the problem of sampling from the posterior will be similar to that of sampling from the prior, so this test is relevant to actual Bayesian inference problems.

It is of course possible to sample from this distribution directly, by simply sampling for v, and then sampling for each of x_1 to x_9 given this value for v, thereby obtaining independent points from exactly the correct distribution. And in any case, we already know the correct marginal distribution for v, which will be the main focus of this test. We will pretend, however, that we don't already know the answer and compare what we would then conclude using various Markov chain methods to what we know is actually correct.

Figure 14 shows the results of trying to sample from this distribution using Metropolis methods and single-variable slice sampling. The upper plot shows 2,000 iterations of a run in which each iteration consists of 10,000 multivariate Metropolis updates (i.e., 20 million Metropolis updates altogether). The proposal distribution was a spherical Gaussian centered on the current state, with standard deviation one for each of the ten variables. The initial state had $v = 0$ and all $x_i = 1$. The points plotted are the value of v at each iteration, with dotted lines shown at $v = \pm 7.5$.

The results of this run are grossly incorrect. We know that the marginal distribution for v is Gaussian with mean zero and standard deviation 3. One would expect that out of 2,000 points from this distribution, on average 95.6 (4.8%) should be less than -5, but none of the points sampled by the multivariate Metropolis method is in this region. Moreover, there is little in the plot to indicate that anything is wrong. In an actual application, the results of a run such as this could easily be accepted as being correct, with serious consequences.

The source of the problem is the low probability of accepting a proposal when in a state where v is small. When v is -4, for example, the standard deviation of the x_i conditional on this value for v is 0.135. The chances that a multivariate Metropolis proposal in which each x_i has standard deviation one will produce values for all the x_i that are within this range of zero are about $0.135^9 \approx 1.5 \times 10^{-8}$. The proposal will include a change to v as well as the x_i, so this calculation does not give the exact acceptance probability, but it does indicate that when v is small, the acceptance probability can become very small, and the chain will remain in the same state for a very long time. Since the Markov chain leaves the correct distribution invariant, it follows that the chain will only very rarely move from a large value of v (which happens to be where this run was started) to a small value for v; indeed, this never occurred in the actual run.

Once one suspects a problem of this sort, signs of it can be seen in the plot. In particular, starting at iteration 1,198, the value of v stays at around -3.3 for 25 iterations (i.e., for 250,000 Metropolis updates). However, there are no obvious

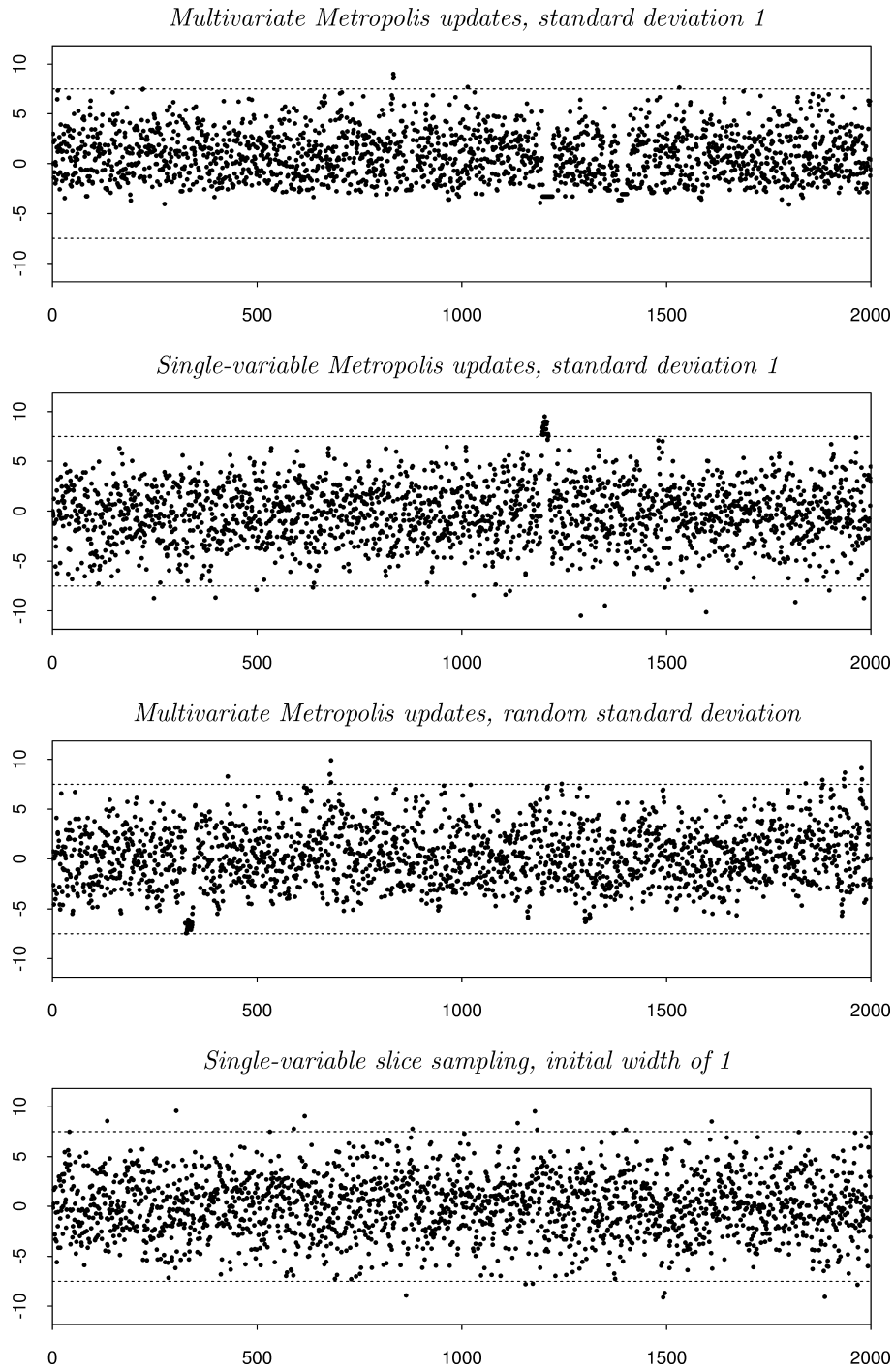


FIG. 14. *Sampling from the funnel distribution using Metropolis and slice-sampling methods.*

occurrences of this sort in the first 1,000 iterations, so the problem would not be apparent even to a suspicious user if only half as many iterations had been done. Running several chains from different starting states might have revealed the problem, but when sampling from more complex distributions, it is difficult to be sure that an appropriate variety of starting states has been tried.

The second plot in Figure 14 shows the results of sampling from the funnel distribution using single-variable Metropolis updates, applied to each variable in sequence. The proposal distribution was a Gaussian centered on the current value, with standard deviation one. Each iteration for this run consisted of 1,300 updates for each variable in turn, which take approximately as long as 10,000 multivariate Metropolis updates (with the program and machine used). As before, the plot shows the value of v after each of 2,000 such iterations.

The results using single-variable Metropolis updates are not as grossly wrong as those obtained using multivariate Metropolis updates. Small values for v are obtained in the expected proportion. The previous problem of very low acceptance rates when v is small is avoided because even when the standard deviation for one of the x_i given v is much smaller than the proposal standard deviation, proposals to change a single x_i are still accepted occasionally (e.g., when $v = -9$, the standard deviation of the x_i is 0.011, and about one proposal in 100 is accepted).

However, *large* values for v are sampled poorly in this run. About 0.6% of the values should be greater than 7.5 (which is marked by a dotted line), but no such values are seen in the first half of the run (1,000 iterations, 1.3 million updates for each variable). Around iteration 1,200, the chain moves to large values of v and stays there for 17 iterations (22,100 updates for each variable). This number of points above 7.5 is not too far from the expected number in 2,000 iterations, which is 12.4, so in this sense the run produced approximately the right answer. However, it is clear that this was largely a matter of luck. Movement to large values of v is rare, because once such a value for v is reached, the chain is likely to stay at a large value for v for a long time. In this case, the problem is not a high rejection rate, but rather slow exploration of the space in small steps. For example, the standard deviation of the x_i when v is 7.5 is 42.5. Exploring a range of plus or minus twice this by a random walk with steps of size around 1 takes about $(4 \times 42.5)^2 = 28{,}900$ updates of each variable. While exploring this range, substantial amounts of time will be spent with values for the x_i that are not compatible with smaller values of v. (This problem is not as severe in the previous run, because the multivariate proposals take larger steps, since they change all variables at once.)

We might try to avoid the problems with sampling for both large and small values of v by picking the proposal standard deviation at random, from a wide range. The third plot in Figure 14 shows the results when using multivariate Metropolis proposals in which the log base 10 of the proposal standard deviation is chosen uniformly from the interval $(-3, 3)$. Large values for v are sampled fairly well, but small values for v are still a problem, though the results are not as bad as for multivariate Metropolis with the proposal standard deviation fixed

at 1. Increasing the range of proposal standard deviations to even more than six orders of magnitude might fix the problem, but at an even greater cost in wasted computation when the random choice is inappropriate.

The bottom plot in Figure 14 shows the results of trying to sample from the funnel distribution using single-variable slice sampling. The initial interval was of size 1, and was expanded by the stepping-out procedure (Figure 3) until both ends are outside the slice, and then sampled from with the shrinkage procedure (Figure 5). Each of the 2,000 iterations done consisted of 120 such updates for each variable in turn, which takes approximately the same amount of time as the Metropolis methods. The average number of evaluations of f for these slice sampling updates was 12.7, but a few updates required more than a hundred evaluations.

The results with single-variable slice sampling are quite good. Small values of v are perhaps sampled slightly less well than with single-variable Metropolis updates, but the difference is not large. Large values of v are sampled better than with any of the Metropolis methods. This good performance is due to the way the stepping out and shrinkage procedures can adapt to the local characteristics of the distribution.

9. Discussion. The table in Figure 15 summarizes the characteristics of the slice sampling methods discussed in this paper and of some competing approaches for sampling from general distributions on continuous state spaces. I list methods that update a single variable at a time separately from multivariate methods. Single-variable methods may be preferred when the coordinate system used is such that one expects many of the variables to be almost independent. Furthermore, for some distributions, recomputing the probability density after a change to one variable may be much faster than recomputing it after a change to all variables. When there are strong dependencies between variables, however, single-variable updates may converge slowly, or even be nonergodic, though simple-minded multivariate methods will not necessarily be better in such a situation.

The first column in the table indicates whether the method requires that derivatives of the log probability density be computable. Derivatives are needed by dynamical methods and reflective slice sampling, which limits their applicability. Adaptive rejection sampling [Gilks and Wild (1992), Gilks (1992)] and overrelaxed slice sampling can take advantage of derivatives, but can operate without them with only a moderate loss of efficiency; for example, with no derivatives available, overrelaxed slice sampling can find endpoints using bisection rather than Newton iteration.

The second and third columns indicate how critical it is that tuning parameters be set to good values, and whether or under what conditions "retrospective tuning" is allowed; that is, whether parameters of the method can be set based on information from past iterations. Adaptive rejection sampling (ARS) for log concave distributions is very good in these respects; one must specify a stepsize

	Derivatives needed?	How critical is tuning?	Retrospective tuning allowed?	Can suppress random walks?
Single-variable methods				
ARS/ARMS	No (but helpful)	Low–Medium	If log concave	No
Single-variable Metropolis	No	Medium	No	No
Single-variable slice sampling	No	Low	If unimodal	No
Overrelaxed slice sampling	No (but helpful)	Low	If unimodal and endpoints exact	Yes
Multivariate methods				
Multivariate Metropolis	No	Medium–High	No	No
Dynamical methods	Yes	High	No	Yes
Slice sampling with hyperrectangles	No	Low–Medium	No	No
Slice sampling with Gaussian crumbs	Possibly helpful	Low–Medium	No	No
Reflective slice sampling	Yes	Medium–High	No	Yes

FIG. 15. *Characteristics of some general-purpose Markov chain sampling methods.*

to use in searching for a point on the other side of the mode, but this parameter can be tuned retrospectively, and if it is too small, it can be rapidly increased by doubling. Parameter tuning is more of a problem when ARMS [Gilks, Best and Tan (1995)] is used for distributions not known to be log concave. A poor choice of parameters may have worse effects, and retrospective tuning is not allowed [Gilks, Neal, Best and Tan (1997)]. Tuning is also a problem for single-variable and multivariate Metropolis methods: proposing changes that are too small leads to an inefficient random walk, while proposing changes that are too large leads to frequent rejections. Metropolis methods must not be tuned retrospectively.

Single-variable slice sampling and overrelaxed slice sampling offer advantages over other methods in these respects. Whereas ARS/ARMS allows retrospective tuning only for log concave distributions, it is allowed for these slice sampling methods when they are applied to any unimodal distribution (provided the interval is expanded to the whole slice, and endpoints for overrelaxation are computed exactly). Furthermore, the tuning is less critical for slice sampling than for the other methods (apart from ARS), as discussed further below.

The final column indicates whether the method can potentially suppress random walk behavior. This is important when sampling from a distribution with high dependencies between variables, which may have to be explored in small steps, since the difference in efficiency between diffusive and systematic exploration of the distribution can then be very large. Hybrid Monte Carlo works very well as a way of suppressing random walks, provided it is tuned properly. However, using a stepsize for hybrid Monte Carlo that is too large is disastrous, since the dynamical simulation becomes unstable, and very few changes are accepted. Reflective slice sampling offers an alternative that may sometimes be easier to tune.

We can also explore the differences between these methods by seeing how well they work in various circumstances. The most favourable situation is when our prior knowledge lets us choose good tuning parameters for all methods. A Metropolis algorithm with a simple proposal distribution will then explore the distribution fairly efficiently (although in a random walk), and will have low overhead, since it requires evaluation of $f(x)$ at only a single new point in each iteration. Single-variable slice sampling will be comparably efficient, however, provided we stick with the interval chosen initially (i.e., set $m = 1$). There will then be no need to evaluate $f(x)$ at the boundaries of the interval, and if the first point chosen from this interval is within the slice, only a single evaluation of $f(x)$ will be done. If this point is outside the slice, further evaluations will be required, but this inefficiency corresponds to the possibility of rejection with the Metropolis algorithm. Multivariate slice sampling with an initial hyperrectangle that is not expanded behaves analogously. Metropolis and slice sampling methods should therefore have similar performance when both are tuned well. However, slice sampling will work better if it turns out that we mistakenly chose too large a width for the Metropolis proposal distribution and initial slice sampling interval. This will lead to a high rejection rate for the Metropolis algorithm, but the sampling procedures of Figures 5 and 8 efficiently use rejected points to shrink the interval, lessening the impact of such a bad choice.

As seen in the demonstration of Section 8, the advantage of slice sampling over Metropolis methods can be quite dramatic if we do not know enough to choose a good tuning parameter, or if no single value of the tuning parameter is appropriate.

Another possibility is that we know that the conditional distributions are log concave, but we do not know how wide they are. Adaptive rejection sampling (ARS) with retrospective tuning will then work quite well. Single-variable slice sampling will also work well, since in this situation it too can be tuned retrospectively (provided no limit is set on the size of the interval). However, ARS does true Gibbs sampling, whereas the slice sampling updates do not produce points that are independent of the previous point. This dependency probably slows convergence, so ARS may be better than single-variable slice sampling in this context (though this will depend also on how many function evaluations each method requires).

Suppose, however, that we know only that the conditional distributions are unimodal, but not necessarily log concave. We would then need to use ARMS rather than ARS, and would not be able to tune it retrospectively, whereas we can still use single-variable slice sampling with retrospective tuning. This will likely not be as good as true Gibbs sampling, however, which we should prefer if the conditional distribution happens to be one that can be efficiently sampled from. In particular, if slice sampling is used to sample from a heavy-tailed distribution, it may move only infrequently between the tails and the central region, since this transition can occur only when we move to a point under the curve of $f(x)$ that is as low as the region under the tails. However, there appears to be no general purpose scheme that avoids problems in this situation.

Finally, consider a situation where we do not know that the conditional distributions are unimodal, and have only a rough idea of an appropriate width for a proposal distribution or initial slice sampling interval. Single-variable slice sampling copes fairly well with this uncertainty. If the initial interval is too small it can be expanded as needed, either by stepping out or by doubling—which is better depends on whether the faster expansion of doubling is worth the extra overhead from the acceptance test of Figure 6. If instead the initial interval is too big, it will be shrunk efficiently by the procedure of Figure 5. We might try to achieve similar robustness with the Metropolis algorithm by doing several updates for each variable, using proposal distributions with a range of widths. For example, if w is our best guess at an appropriate width, we might do updates with widths of $w/4$, $w/2$, w, $2w$ and $4w$. This may ensure that an appropriate proposal distribution is used some of the time, but it is unattractive for two reasons. First, the limits of the range (e.g., from $w/4$ to $4w$) must be set *a priori*. Second, for this approach to be valid, we must continue through the original sequence of widths even after it is clear that we have gone past the appropriate one. These problems are not present with slice sampling.

Multivariate slice sampling using hyperrectangles will usually not offer much advantage over single-variable slice sampling (as is also the case with multivariate versus single-variable Metropolis methods). However, the more general framework for multivariate slice sampling based on "crumbs" that was outlined in Section 5.2 offers the possibility of adapting not just to the scales of the variables, but also to the dependencies between them. The benefits of such methods can only be determined after further research, but huge increases in efficiency would seem conceivable, if one is to judge from the analogous comparison of minimization by simple steepest descent versus more sophisticated quasi-Newton or conjugate gradient methods.

The practical utility of the slice sampling methods described here will ultimately be determined by experience in a variety of applications. Some applications will involve tailor-made sampling schemes for particular models—for instance, Frey (1997) used single-variable slice sampling to sample for latent variables

in a slice sampling for another neural network model. The adaptivity of slice sampling should prove particularly useful when using tempering or annealing methods [Geyer and Thompson (1995), Neal (2001)] in order to avoid problems with multimodality, since these methods require sampling from a whole sequence of distributions, and we would rather not have to manually tune a sampler for each one. Slice sampling is also particularly suitable for use in automatically generated samplers, and is now used in some situations by the WinBUGS system [Lunn, Thomas, Best and Spiegelhalter (2000)]. Readers can try out slice sampling methods for themselves, on a variety of Bayesian models, using the "software for flexible Bayesian modeling" that is available from my web page. This software (version of 2000-08-21) implements most of the methods discussed in this paper.

Acknowledgments. I thank Brendan Frey, Gareth Roberts, Jeffrey Rosenthal and David MacKay for helpful discussions.

REFERENCES

ADLER, S. L. (1981). Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions. *Phys. Rev. D* **23** 2901–2904.

BARONE, P. and FRIGESSI, A. (1990). Improving stochastic relaxation for Gaussian random fields. *Probab. Engrg. Inform. Sci.* **4** 369–389.

BESAG, J. and GREEN, P. J. (1993). Spatial statistics and Bayesian computation (with discussion). *J. Roy. Statist. Soc. Ser. B* **55** 25–37, 53–102.

CHEN, M.-H. and SCHMEISER, B. W. (1998). Toward black-box sampling: A random-direction interior-point Markov chain approach. *J. Comput. Graph. Statist.* **7** 1–22.

DAMIEN, P., WAKEFIELD, J. C. and WALKER, S. G. (1999). Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **61** 331–344.

DIACONIS, P., HOLMES, S. and NEAL, R. M. (2000). Analysis of a non-reversible Markov chain sampler. *Ann. Appl. Probab.* **10** 726–752.

DOWNS, O. B., MACKAY, D. J. C. and LEE, D. D. (2000). The nonnegative Boltzmann machine. In *Advances in Neural Information Processing Systems* (S. A. Solla, T. K. Leen and K.-R. Muller, eds.) 428–434. MIT Press, Cambridge, MA.

DUANE, S., KENNEDY, A. D., PENDLETON, B. J. and ROWETH, D. (1987). Hybrid Monte Carlo. *Phys. Lett. B* **195** 216–222.

EDWARDS, R. G. and SOKAL, A. D. (1988). Generalization of the Fortuin–Kasteleyn–Swendsen–Wang representation and Monte Carlo algorithm. *Phys. Rev. D* **38** 2009–2012.

FREY, B. J. (1997). Continuous sigmoidal belief networks trained using slice sampling. In *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan and T. Petsche, eds.). MIT Press, Cambridge, MA.

GELFAND, A. E. and SMITH, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *J. Amer. Statist. Assoc.* **85** 398–409.

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. R. (1992). Derivative-free adaptive rejection sampling for Gibbs sampling. In *Bayesian Statistics* (J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, eds.) 641–649. Oxford Univ. Press.

GILKS, W. R., BEST, N. G. and TAN, K. K. C. (1995). Adaptive rejection Metropolis sampling within Gibbs sampling. *Appl. Statist.* **44** 455–472.

GILKS, W. R., NEAL, R. M., BEST, N. G. and TAN, K. K. C. (1997). Corrigendum: Adaptive rejection Metropolis sampling. *Appl. Statist.* **46** 541–542.

GILKS, W. R. and WILD, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Appl. Statist.* **41** 337–348.

GREEN, P. J. and HAN, X. (1992). Metropolis methods, Gaussian proposals and antithetic variables. *Stochastic Models, Statistical Methods, and Algorithms in Image Analysis* (P. Barone et al., eds.). *Lecture Notes in Statist.* **74** 142–164. Springer, New York.

GREEN, P. J. and MIRA, A. (2001). Delayed rejection in reversible jump Metropolis–Hastings. *Biometrika* **88** 1035–1053.

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

HIGDON, D. M. (1996). Auxiliary variable methods for Markov chain Monte Carlo with applications. ISDS discussion paper 96-17, Duke Univ.

HOROWITZ, A. M. (1991). A generalized guided Monte Carlo algorithm. *Phys. Lett. B* **268** 247–252.

LUNN, D. J., THOMAS, A., BEST, N. and SPIEGELHALTER, D. (2000). WinBUGS—a Bayesian modelling framework: Concepts, structure, and extensibility. *Statist. Comput.* **10** 325–337.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H. and TELLER, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21** 1087–1092.

MIRA, A. (1998). Ordering, splicing and splitting Monte Carlo Markov chains. Ph.D. dissertation, School of Statistics, Univ. Minnesota.

MIRA, A. and TIERNEY, L. (2002). Efficiency and convergence properties of slice samplers. *Scand. J. Statist.* **29** 1–12.

NEAL, R. M. (1994). An improved acceptance procedure for the hybrid Monte Carlo algorithm. *J. Comput. Phys.* **111** 194–203.

NEAL, R. M. (1996). *Bayesian Learning for Neural Networks*. *Lecture Notes in Statist.* **118**. Springer, New York.

NEAL, R. M. (1998). Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. In *Learning in Graphical Models* (M. I. Jordan, ed.) 205–228. Kluwer, Dordrecht.

NEAL, R. M. (2001). Annealed importance sampling. *Statist. Comput.* **11** 125–139.

ROBERTS, G. O. and ROSENTHAL, J. S. (1999). Convergence of slice sampler Markov chains. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **61** 643–660.

THOMAS, A., SPIEGELHALTER, D. J. and GILKS, W. R. (1992). BUGS: A program to perform Bayesian inference using Gibbs sampling. In *Bayesian Statistics* (J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, eds.) 837–842. Oxford Univ. Press.

TIERNEY, L. and MIRA, A. (1999). Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine* **18** 2507–2515.

DEPARTMENT OF STATISTICS
UNIVERSITY OF TORONTO
100 ST. GEORGE STREET, 6TH FLOOR
TORONTO, ONTARIO
CANADA M5S 3G3
E-MAIL: radford@stat.utoronto.ca
WEB: http://www.cs.utoronto.ca/∼radford/

DISCUSSION

BY MING-HUI CHEN AND BRUCE W. SCHMEISER

University of Connecticut and Purdue University

We congratulate the author for a well-written paper. Slice sampling is an alternative to Gibbs sampling that avoids the need to sample from nonstandard distributions. The paper provides a detailed overview of the recent development of Markov chain Monte Carlo sampling methods in general and the related slice sampling algorithms in particular. One aim of the paper is to find variations on slice sampling that can be used to sample from any continuous distribution and that require little or no tuning. Thus, this is an important step in developing "black-box" samplers.

The main idea of slice sampling is formalized by introducing an auxiliary real variable y, and defining a joint distribution over x and y that is uniform over the region $U = \{(x, y) : 0 < y < f(x)\}$. This is the idea proposed in Chen and Schmeiser (1998). For single-variable slice sampling, the variation of slice sampling proposed by Neal operates analogously to Gibbs sampling in the sense that to obtain the next point x_1, y is generated from the conditional distribution $[y|x_0]$ given the current point x_0 and then x_1 is drawn from $[x|y]$. Both $[y|x_0]$ and $[x|y]$ are uniform distributions. Since the closed form of the support of $[x|y]$ is not available, sampling directly from $[x|y]$ is not possible. A clever development is Neal's sophisticated (but relatively expensive) sampling procedure to generate x_1 from the "slice" $S = \{x : y < f(x)\}$.

In Chen and Schmeiser (1998), we proposed random-direction interior point (RDIP), a general sampler designed to be "black box" in the sense that the user need not tune the sampler to the problem. RDIP samples from the uniform distribution defined over the region U below the curve of the surface defined by $f(x)$. Both slice sampling and RDIP require evaluations of $f(x)$. Slice sampling, however, can be more expensive than RDIP because slice sampling requires evaluating $f(x)$ more than once per iteration. The intention of RDIP's design is to use as much free information as possible. For the high-dimensional case, the hyperrectangle idea in slice sampling could be inefficient. For example, suppose $f(x)$ is the bivariate normal density with a high correlation. Then, the hyperrectangle idea essentially mimics the Gibbs sampler, which suffers slow convergence; see Chen and Schmeiser (1993) for a detailed discussion. Aligning the hyperrectangle (or ellipses) to the shape of $f(x)$, along the lines of Kaufman and Smith (1998), seems like a good idea.

As Neal mentions, the computational efficiency of our "black-box" sampler RDIP depends on the normalization constant. Our goal was to be automatic and reasonably efficient, rather than to tune the sampler to the problem. If, however,

the user wants the algorithm to tune itself, then renormalizing can be done automatically in the initial stage of sampling.

Neal is concerned that, in Example 4.3 of Chen and Schmeiser (1998), the state-dependent adjustment destroys the reversibility of the transitions. Unfortunately, we were not clear in describing implementation details, which has lead to confusion. When the same normalization constant is used, the version of our algorithm presented in that example is simply a Metropolis-within-Gibbs sampler, which is well known to be reversible. If the normalization is based on the current value, the reversibility is maintained if we appropriately choose an acceptance probability within RDIP, which we did without describing the details. In addition, we empirically checked convergence using several diagnostic criteria and (later) our model parameter estimates matched those of Nandram and Chen (1996).

REFERENCES

CHEN, M.-H. and SCHMEISER, B. W. (1993). Performance of the Gibbs, hit-and-run, and Metropolis samplers. *J. Comput. Graph. Statist.* **2** 251–272.

CHEN, M.-H. and SCHMEISER, B. W. (1998). Toward black-box sampling: A random-direction interior-point Markov chain approach. *J. Comput. Graph. Statist.* **7** 1–22.

NANDRAM, B. and CHEN, M.-H. (1996). Reparameterizing the generalized linear model to accelerate Gibbs sampler convergence. *J. Statist. Comput. Simulation* **54** 129–144.

KAUFMAN, D. E. and SMITH, R. L. (1998). Direction choice for accelerated convergence in hit-and-run sampling. *Oper. Res.* **46** 84–95.

DEPARTMENT OF STATISTICS
UNIVERSITY OF CONNECTICUT
215 GLENBROOK ROAD, U-4120
STORRS, CONNECTICUT 06269-4120
E-MAIL: mhchen@stat.uconn.edu

SCHOOL OF INDUSTRIAL ENGINEERING
PURDUE UNIVERSITY
1287 GRISSOM HALL
WEST LAFAYETTE, INDIANA 47907-1287
E-MAIL: bruce@purdue.edu

DISCUSSION

BY OLIVER B. DOWNS[1]

Microsoft Research

The nonnegative Boltzmann machine (NNBM) is a recurrent neural network model that can describe multimodal nonnegative data. Application of maximum likelihood estimation to this model gives a learning rule that is analogous to that of the binary Boltzmann machine. While the model itself is analytically intractable an efficient stochastic version of the learning rule

[1]Supported in part by Microsoft Research Bell Laboratories (Lucent Technologies) and Princeton University Procter Fellowship.

can be obtained using *reflective slice sampling*, since the slice boundaries can be determined analytically from the model. We compare this with the use of advanced mean field theory to learn a generative model for face image data.

1. Introduction. The multivariate Gaussian is the most elementary distribution used to model generic data. It represents the *maximum entropy* distribution under the constraint that the mean and covariance matrix of the distribution match that of the data. For the case of binary data, the maximum entropy distribution that matches the first- and second-order statistics of the data is given by the Boltzmann machine [Hinton and Sejnowski (1983)].

The Boltzmann machine can be generalized to continuous and nonnegative variables [Downs, MacKay and Lee (2000)]. In this case, the maximum entropy distribution for nonnegative data with known first- and second-order statistics is described by the nonnegative Boltzmann distribution (NNBD),

$$P(x) = \begin{cases} \dfrac{1}{Z}\exp[-E(x)], & \text{if } x_i \geq 0 \ \forall i, \\ 0, & \text{if any } x_i < 0, \end{cases} \tag{1}$$

where the energy function $E(x)$ and normalization constant Z are

$$E(x) = \beta x^T A x - b^T x, \tag{2}$$

$$Z = \int_{x \geq 0} dx \, \exp[-E(x)]. \tag{3}$$

The properties of the NNBD differ quite substantially from the Gaussian distribution which would arise for continuous, unbounded data. In particular, the presence of the nonnegativity constraints allows the distribution to have multiple modes, confined to the rectifying axes, since A can be nonpositive definite. Such a distribution would be poorly modeled by a single Gaussian. Here, we describe how a multimodal NNBD can be learned from nonnegative data.

2. Maximum likelihood. The learning rule for the NNBM can be derived by maximizing the log probability of the observed data under (1). Given a set of nonnegative vectors $\{x^\mu\}$, where $\mu = 1, \ldots, M$ indexes the different examples, the log probability is

$$L = \frac{1}{M}\sum_{\mu=1}^{M} \log P(x^\mu) = -\frac{1}{M}\sum_{\mu=1}^{M} E(x^\mu) - \log Z. \tag{4}$$

Taking the derivatives of (4) with respect to the parameters A and b gives

$$\frac{\partial L}{\partial A_{ij}} = \langle x_i x_j \rangle_{\mathrm{f}} - \langle x_i x_j \rangle_{\mathrm{c}}, \tag{5}$$

$$\frac{\partial L}{\partial b_i} = \langle x_i \rangle_{\mathrm{c}} - \langle x_i \rangle_{\mathrm{f}}, \tag{6}$$

where the subscript "c" denotes a "clamped" average over the data, and the subscript "f" denotes a "free" average over the NNBM distribution,

$$\langle f(x)\rangle_{\mathrm{c}} = \frac{1}{M}\sum_{\mu=1}^{M} f(x^{\mu}), \qquad \langle f(x)\rangle_{\mathrm{f}} = \int_{x\geq 0} dx\, P(x) f(x). \tag{7}$$

These derivatives are used to define a gradient ascent learning rule for the NNBM. The contrast between the clamped and free covariance matrices is used to update the interactions A, while the difference between the clamped and free means is used to update the local biases b.

3. Mean-field theories. It is possible to obtain approximations to the statistics of the model and to learn approximate parameters from data using concepts from mean-field theory. In (2) the (inverse) temperature parameter, β, controls the influence of correlations in the model. It is possible to approximate expectations under the model distribution in the limit that these correlations are assumed weak, the "high temperature" limit, at which $\beta = 0$. Since $E(x)$ is linear in this limit, integrals over the NNBD are tractable; we then accommodate small nonzero β by Taylor expansion about $\beta = 0$ [Downs (2001)].

To first order in β this approach returns the "naive" mean-field approximation, equivalent to approximating the NNBD with a factorized product of one-dimensional exponential distributions, with means matching that of the data. This replaces the "free" correlations in (5) with

$$\langle x_i x_j\rangle_{\mathrm{f}} = (1+\delta_{ij})\langle x_i\rangle_{\mathrm{c}}\langle x_j\rangle_{\mathrm{c}}. \tag{8}$$

Then expanding second-order in β we obtain a TAP-like [Kappen and Rodriguez (1998)] correction to this approximation,

$$\Delta\langle x_i x_j\rangle_{\mathrm{f}} = -\frac{\beta}{2}\sum_{kl}\alpha_{ijkl}A_{ij}A_{kl}\langle x_i\rangle_{\mathrm{c}}\langle x_j\rangle_{\mathrm{c}}\langle x_k\rangle_{\mathrm{c}}\langle x_l\rangle_{\mathrm{c}}. \tag{9}$$

4. Monte Carlo sampling. A direct approach to calculating the "free" averages in (5) and (6) is to numerically approximate them. This can be accomplished by generating samples from the NNBD using a Markov chain Monte Carlo method. Such methods employ an iterative stochastic dynamics whose equilibrium distribution converges to that of the desired distribution [MacKay (1998)].

Gibbs sampling from such a distribution requires repeated evaluation of the error function $\mathrm{erf}(z)$, and hence is prone to numerical error or high computational cost. The method of reflective slice sampling [Neal (1997)] circumvents this.

The basic idea of the reflective slice sampling algorithm is shown in Figure 1. Given a sample point x_i, a random $y \in [0, P^*(x_i)]$ [where $P^*(x)$ is the unnormalized density] is first chosen uniformly. Then a slice S is defined as

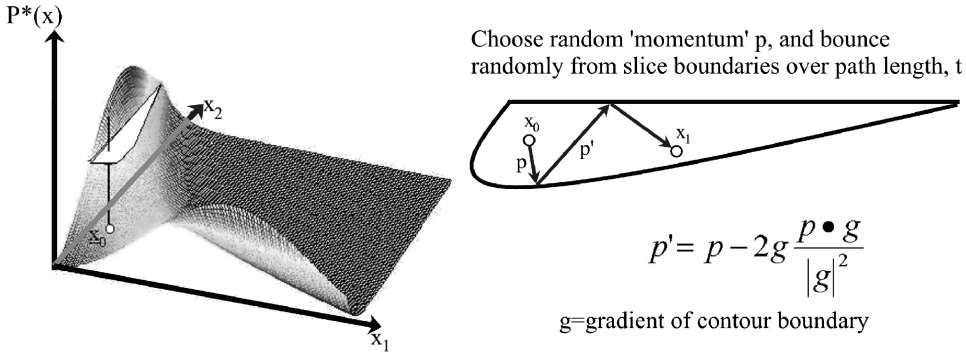


FIG. 1. *Reflective slice sampling in two dimensions. Given the current sample point, x_0, a height $y \in [0, P^*(x_0)]$ is randomly chosen. This defines a slice $(x \in S \mid P^*(x) \geq y)$ in which a new x_1 is chosen, using billiard-ball dynamics with specular reflections from the interior boundaries of the slice.*

the connected set of points $(x \in S \mid P^*(x) \geq y)$ including x_i, and the new point $x_{i+1} \in S$ is chosen randomly from this slice.

In order to efficiently choose a new point within a particular multidimensional slice, reflective "billiard ball" dynamics are used. A random initial direction in the slice, p, is chosen, and the new point is evolved by traveling a predefined distance from the current point along the path specularly reflecting from the boundaries of the slice.

For the NNBM, solving the boundary points along a particular direction in a given slice is quite simple, since it only involves solving the roots of a quadratic equation,

$$\beta x^T A x - b^T x - \log y = 0, \tag{10}$$

along the line defined by the momentum vector, p, bounding the points by the rectifying axes,

$$x_{\text{boundary}} = \left[\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \right]^+, \tag{11}$$

where $^+$ denotes rectification. The reflected momentum direction is a simple function of the incident momentum and the gradient of the slice boundary, g,

$$p' = p - 2g \frac{p \cdot g}{|g|^2} \qquad \text{with } g = 2\beta A x - b. \tag{12}$$

The distribution of x_n for large n can be shown to converge to the desired density $P(x)$. Figure 2(a) demonstrates this for a two-dimensional NNBD.

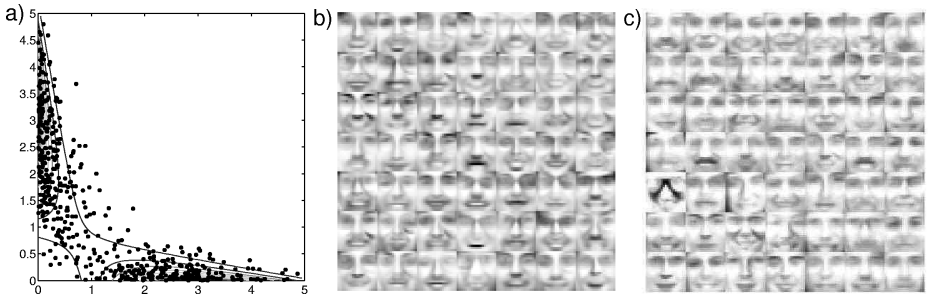


FIG. 2. (a) *Contours of the two-dimensional* "*competitive*" *NNBD overlaid by* 500 *reflective slice samples from the distribution.* (b) *Prototype face images generated from a mean-field NNBM.* (c) *Prototype face images generated from an NNBM learned via reflective slice sampling.*

5. Generative model for faces. We have used the NNBM to learn a generative model for images of human faces. The NNBM is used to model the correlations in the coefficients of the nonnegative matrix factorization (NMF) of the face images [Lee and Seung (1999)]. NMF reduces the dimensionality of nonnegative data by decomposing the face images into parts correponding to eyes, noses, ears, etc. Since the different parts are coactivated in reconstructing a face, the activations of these parts contain significant correlations that need to be captured by a generative model. Here we briefly demonstrate how the NNBM is able to learn these correlations, comparing the mean-field and reflective slice sampling approaches described above.

Sampling from the learned NNBD stochastically generates coefficients which can graphically be displayed as face images. Figures 2(b) and 2(c) compare prototype face images from NNBDs learned using advanced mean-field theory and reflective slice sampling, respectively. We suggest that the latter produces the more plausible prototypes.

6. Discussion. Here we have demonstrated the application of reflective slice sampling to NNBM learning. Since the NNBM learning rule is generally intractable, approximations are required to enable efficient learning. The use of reflective slice sampling is seen to be a reasonable alternative to mean-field approaches for learning the model, and can be implemented efficiently since slice boundaries and reflections can be determined analytically.

Extensions to the present work include incorporating hidden units into the recurrent network, which would imply modeling higher-order statistics of the data.

Acknowledgments. The author is especially grateful to David MacKay and Daniel Lee for their collaboration on the original ideas for this work, to Hagai Attias for comments on this manuscript and to John Hopfield and David Heckerman for their on-going support.

REFERENCES

DOWNS, O. B. (2001). High-temperature expansions for learning models of nonnegative data. In *Advances in Neural Information Processing Systems* **13** (T. K. Leen, T. G. Dietterich and V. Tresp, eds.) 465–471. MIT Press, Cambridge, MA.

DOWNS, O. B., MACKAY, D. J. C. and LEE, D. D. (2000). The nonnegative Boltzmann machine. In *Advances in Neural Information Processing Systems* **12** (S. A. Solla, T. K. Leen and K.-R. Muller, eds.) 428–434. MIT Press, Cambridge, MA.

HINTON, G. E. and SEJNOWSKI, T. J. (1983). Optimal perceptual learning. In *IEEE Conference on Computer Vision and Pattern Recognition* 448–453. Washington.

KAPPEN, H. J. and RODRIGUEZ, F. B. (1998). Efficient learning in Boltzmann machines using linear response theory. *Neural Computation* **10** 1137–1156.

LEE, D. D. and SEUNG, H. S. (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature* **401** 788–791.

MACKAY, D. J. C. (1998). Introduction to Monte Carlo methods. In *Learning in Graphical Models* (M. I. Jordan, ed.) 175–204. Kluwer, Dordrecht.

NEAL, R. M. (1997). Markov chain Monte Carlo methods based on "slicing" the density function. Technical Report 9722, Dept. Statistics, Univ. Toronto.

MICROSOFT RESEARCH
ONE MICROSOFT WAY
REDMOND, WASHINGTON 98052
E-MAIL: t-odowns@microsoft.com

DISCUSSION

BY ANTONIETTA MIRA AND GARETH O. ROBERTS

University of Insubria, Italy and Lancaster University

It is a pleasure to have the opportunity to contribute towards the discussion of this paper. Many much simpler MCMC algorithms have wide applicability in statistics and other fields. However, for us, one important motivation to the development of practically useful slice sampler methods comes from their extremely appealing and robust theoretical properties.

In the first part of our discussion we would like to focus on the theory and methodology related to slice sampling that has appeared in the past few years in the MCMC literature. Secondly, we will give more specific comments on the methodology developed in the paper and expand on some connections between the adaptive slice sampling and the delaying rejection strategy [Mira (1998) and Tierney and Mira (1999)].

Theoretical properties of slice sampling. One aspect of slice samplers that makes their (sometimes rather complex) implementation worthwhile is their extremely robust theoretical properties. Single-variable slice samplers in particular

have been thoroughly investigated by Mira and Tierney (2002) and Roberts and Rosenthal (1999). Generalizing just a little from the setting described in Section 4 of the paper, suppose that our target density can be written as

$$f(x) = f_0(x) f_1(x) \tag{1}$$

and consider the slice sampler which iterates the following procedure assuming that the current state is x:

1. Sample $U \sim \text{Uniform}(0, f_1(x))$.
2. Sample X from the density proportional to $f_0(x)\mathbf{1}_{\{z; f_1(z) \geq U\}}$.

The algorithm is therefore just a two-dimensional Gibbs sampler on an appropriate target distribution. Now two-dimensional Gibbs samplers can have very poor convergence properties, and can frequently fail to be geometrically ergodic, for instance. However, it turns out that under extremely weak regularity conditions [Roberts and Rosenthal (1999)] this simple slice sampler is geometrically ergodic. The weakest form of these regularity conditions is hard to describe easily here, but they imply, for instance, that densities which can be bounded by $|x|^{-(d+\varepsilon)}$ for some $\varepsilon > 0$ in d dimensions all produce geometrically ergodic slice samplers where f_0 is constant. For even stronger results, the slice sampler is uniformly ergodic in the case where f_0 is chosen to be a finite measure [Mira and Tierney (2002)].

A key to the provability of quite general results for slice sampler algorithms is the following observation. It turns out that all the algorithm's convergence properties can be calculated from the function

$$Q(y) = \int_{f_1(z) \geq y} f_0(x)\, dx.$$

Irrespective of the state space on which x itself lives, the Markov chain can be shown to be stochastically monotone under the ordering defined by the function f_1 [see Roberts and Rosenthal (1999, 2001)]. As a result of this, quite reasonable explicit bounds for convergence times (defined as the first time the total variation distance from stationarity decreases below 0.01) can be obtained using the methodology of Roberts and Tweedie (2000). As an illustration of these results, it can be shown that, for the case where f_0 is constant (i.e., in the single-variable slice sampler), and f_1 is a real-valued log-concave function, 525 iterations suffice for convergence from all starting points x with $f_1(x) \geq 0.01 \sup_y f_1(y)$.

Similar results can be deduced for multidimensional log-concave distributions but the bounds worsen as dimension increases reflecting a genuine curse of dimensionality in this problem (despite the fact that this is inherently a two-dimensional Gibbs sampler). To counteract this issue, Roberts and Rosenthal (2002) introduces the *polar slice sampler* in d dimensions, where $f_0(\mathbf{x})$ is chosen

to be $|\mathbf{x}|^{d-1}$. Remarkably, this algorithm can be shown to have convergence times which are stable as dimension increases.

Of course (as with many of these slice sampler methods) the polar slice sampler in its pure form is likely to be difficult to implement in many problems, but its extremely strong convergence properties [as demonstrated in Roberts and Rosenthal (2002)] show great promise. For high-dimensional problems, it might be worthwhile to translate some of Professor Neal's ideas for implementation of slice sampler to the polar slice context.

The slice sampler can also be shown to have theoretical properties which are uniformly superior to independence sampler algorithms. In a Peskun (1973) ordering sense, Mira and Tierney (2002) show that any independence Metropolis–Hastings sampler (with proposal density given by f_0) can be readily turned into a slice sampling scheme of the form (1), and this resulting algorithm is uniformly more efficient. In fact, if f_1 is bounded, then both the slice sampling and the independence Metropolis–Hastings scheme give rise to uniformly ergodic Markov chains. On the other hand, if the ratio is not bounded then the slice sampling can still be geometrically ergodic while the independence Metropolis–Hastings is never geometrically ergodic [Mira and Tierney (2002)].

The perfect slice sampler. A further consequence of the monotonicity property mentioned earlier is that the single-variable slice sampler can be turned into a perfect simulation algorithm [Mira, Møller and Roberts (2001)] following the original coupling from the past (CFTP) [Propp and Wilson (1996)]. If a maximal and minimal point relative to the ordering defined by f can be identified, these are used to initialize the maximal and minimal chains. Otherwise, if the slice sampler is uniformly ergodic a maximal and/or a minimal bounding process can be constructed in the spirit of Kendall and Møller (2000). Reversibility of the slice sampler allows easy simulation of these processes backwards in time to identify the starting point of the maximal and minimal chains. The beauty of the perfect slice sampling construction relies on the possibility of coupling the maximal and minimal chains even on a continuous state space. This is achieved because, thanks to monotonicity, the minimal horizontal slice (i.e., the one defined by the minimal chain) is always a superset of the maximal horizontal slice. If, when sampling over the minimal horizontal slice, a point is selected that belongs to the intersection of the minimal and the maximal horizontal slices, instantaneous coupling happens.

Examples of applications given in Mira, Møller and Roberts (2001) include the Ising model on a two-dimensional grid at the critical temperature and various other automodels. In Casella, Mengersen, Robert and Titterington (2002) a further application of the perfect slice sampler construction to mixture of distributions is studied.

Various modifications of the basic perfect slice algorithm are possible. An *interruptible* perfect simulation algorithm exists, which uses the construction due to Fill (1998). A further improvement of the algorithm in Mira, Møller and Roberts (2001) allows the use of *read once random numbers* as introduced in Wilson (2000).

Irreducibility of stepping out procedure. In choosing w, the slice width in the stepping out mechanism, it is worth commenting that irreducibility of the resulting sampler can be lost. For instance, consider the following target: uniform on the disjoint intervals $[0, a]$ and $[b, b + a]$ for same $a, b > 0$. If we set $w < (b - a)$ and start the sampler in one of the two intervals, the sampler will never reach the other interval when adopting a stepping out procedure. This reducibility issue is avoided when adopting a doubling strategy because, when a decision of expanding is made, the doubling is performed on either direction with equal probability irrespective of whether that side is already outside the slice.

Delaying rejection and adaptive slice sampling. There is a strong connection between the adaptive slice sampler introduced in the paper in Section 5.2 and the delaying rejection strategy [Mira (1998) and Tierney and Mira (1999)].

In a standard Metropolis–Hastings algorithm, if the proposed candidate move is rejected, instead of retaining the same position and advancing time, a second stage proposal distribution (which is allowed to depend both on the current position of the chain and the candidate move that has just been rejected) can be constructed. The delaying rejection strategy consists in using this second stage proposal to generate a new candidate move. The acceptance probability of this second stage candidate is computed so that reversibility relative to the target is preserved. This procedure can be iterated and higher stage proposals are constructed based on previously rejected candidates (within the same sweep). The advantage of this strategy relies on the fact that the resulting MCMC estimates have, uniformly, a smaller asymptotic variance relative to the standard Metropolis–Hasting type of algorithm [Tierney and Mira (1999)]. Furthermore, the delaying rejection strategy can be adopted also in a varying dimensional setting [Green and Mira (2001)] thus combining efficiency and partial adaptation of the proposal with the flexibility of reversible jumps [Green (1995)].

The idea of reducing the variance at higher stages (later trials), using the values of f (or other information) from previous trial points, is a strategy proposed within the delaying rejection framework [Mira (1998), Chapter 5, and Mira (2002)] which also works within the adaptive slice sampling framework.

On the other hand, the crumbs idea introduced in Professor Neal's paper can be turned into a strategy to define how the proposal distribution at later stages (within the delaying rejection mechanism) depends on previously rejected candidates.

Furthermore, notice that, in light of the similarities between the delaying rejection and the adaptive slice sampling framework, the process of generating crumbs and trial points can be interrupted at any fixed time by allowing the chain to retain the current x position and drawing a new y sample (i.e., a new "vertical slice").

In conclusion. We congratulate Professor Neal for producing a stimulating paper which will greatly advance the applicability of slice sampler methods. These methods have excellent theoretical properties and can be flexibly implemented in conjunction with adaptive techniques, and even sometimes within a perfect simulation context. As is inevitable with MCMC algorithms, there is a gap between applicable methods and algorithms which admit full theoretical analysis. However, this gap is particularly narrow for slice samplers, so that the available theory is highly relevant to practical application.

REFERENCES

CASELLA, G., MENGERSEN, K. L., ROBERT, C. P. and TITTERINGTON, D. M. (2002). Perfect samplers for mixtures of distributions. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **64** 777–790.

FILL, J. A. (1998). An interruptible algorithm for perfect sampling via Markov chains. *Ann. Appl. Probab.* **8** 131–162.

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

GREEN, P. J. and MIRA, A. (2001). Delayed rejection in reversible jump Metropolis–Hastings. *Biometrika* **88** 1035–1053.

KENDALL, W. S. and MØLLER, J. (2000). Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Adv. in Appl. Probab.* **32** 844–865.

MIRA, A. (1998). Ordering, slicing and splitting Monte Carlo Markov chains. Ph.D. dessertation, School of Statistics, Univ. of Minnesota.

MIRA, A. (2002). On Metropolis–Hastings algorithms with delayed rejection. *Metron* **59** 231–241.

MIRA, A., MØLLER, J. and ROBERTS, G. O. (2001). Perfect slice samplers. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **63** 593–606.

MIRA, A. and TIERNEY, L. (2002). Efficiency and convergence properties of slice samplers. *Scand. J. Statist.* **29** 1–12.

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

PROPP, J. and WILSON, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms* **9** 223–252.

ROBERTS, G. O. and ROSENTHAL, J. S. (1999). Convergence of slice sampler Markov chains. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **61** 643–660.

ROBERTS, G. O. and ROSENTHAL, J. S. (2001). Markov chains and deinitialising processes. *Scand. J. Statist.* **28** 489–505.

ROBERTS, G. O. and ROSENTHAL, J. S. (2002). The polar slice sampler. *Stoch. Models* **18** 257–280.

ROBERTS, G. O. and TWEEDIE, R. L. (2000). Rates of convergence of stochastically monotone and continuous time Markov models. *J. Appl. Probab.* **37** 359–373.

TIERNEY, L. and MIRA, A. (1999). Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine* **18** 2507–2515.

WILSON, D. B. (2000). How to couple from the past using a read-once source of randomness. *Random Structures Algorithms* **16** 85–113.

DEPARTMENT OF ECONOMICS
UNIVERSITY OF INSUBRIA
VIA RAVASI 2
21100 VARESE
ITALY
E-MAIL: antonietta.mira@uninsubria.it

DEPARTMENT OF MATHEMATICS
AND STATISTICS
LANCASTER UNIVERSITY
LANCASTER LA1 4AF
UNITED KINGDOM
E-MAIL: G.O.Roberts@lancaster.ac.uk

DISCUSSION

BY JOHN SKILLING AND DAVID J. C. MACKAY

Cambridge University

The real variables of a probabilistic model will always be represented in a computer using a finite number of bits. We describe an implementation of slice sampling in which the stepping-out, randomization and shrinking operations, described by Neal (in this paper) in terms of floating-point operations, are replaced by binary and integer operations.

We assume that the variable x that is being slice sampled is represented by a b-bit integer X taking on one of $B = 2^b$ values, $0, 1, 2, \ldots, B - 1$, many or all of which correspond to valid values of x. We often take these points to have equal prior measure, so that the prior becomes flat over X and all points are automatically a priori equivalent. Floating-point numbers, by contrast, are not equivalent, because of their variable rounding. Using an integer grid eliminates any errors in detailed balance that might thus ensue. We denote by $F(X)$ the appropriately transformed version of the unnormalized density $f(x(X))$.

We assume the following operators on b-bit integers are available:

$X + N$	arithmetic sum, modulo B, of X and N,
$X - N$	difference, modulo B, of X and N,
$X \oplus N$	bitwise exclusive or of X and N,
$N \leftarrow$ `randbits`(l)	sets N to a random l-bit integer.

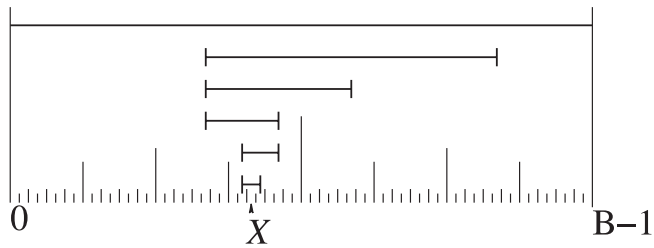
A slice-sampling procedure for integers is then as follows:

Given: a current point X and a height $Y = F(X) \times \text{Uniform}(0, 1) < F(X)$

1	$U \leftarrow$ `randbits`(b)	Define a random translation U of the binary coordinate system.
2	set l to a value $l \leq b$	Set initial l-bit sampling range.
3	do {	
4	$N \leftarrow$ `randbits`(l)	Define a random move within the current interval of width 2^l.
5	$X' \leftarrow ((X - U) \oplus N) + U$	Randomize the lowest l bits of X (in the translated coordinate system).
6	$l \leftarrow l - 1$	If X' is not acceptable, decrease l and try again with a smaller perturbation
7	} until $(X' = X)$ or $(F(X) > Y)$	of X; termination at or before $l = 0$ is assured.

The translation U is introduced to avoid permanent sharp edges, where for example the adjacent binary integers `0111111111` and `1000000000` would otherwise be permanently in different sectors, making it difficult for X to move from one to the other.

Pictorially, the sequence of intervals from which the new candidate points are drawn is like the sequence of intervals in Neal's doubling procedure, Figure 2.



If preliminary stepping-out from the initial range is required, step 2 above can be replaced by the following similar procedure:

2a	set l to a value $l < b$	(l sets the initial width)
2b	do {	
2c	$N \leftarrow$ `randbits`(l)	
2d	$X' \leftarrow ((X - U) \oplus N) + U$	
2e	$l \leftarrow l + 1$	
2f	} until $(l = b)$ or $(F(X) < Y)$	

These shrinking and stepping out methods shrink and expand by a factor of two per evaluation. A variant is to shrink or expand by more than one bit each

time, setting $l \leftarrow l \pm \Delta l$ with $\Delta l > 1$. Provided the initial sampling range is well chosen (i.e., of the same order of magnitude as the acceptable range), we found experimentally that the mean diffusion rate of X per evaluation when $\Delta l = 1$ is at most 25% slower than for Neal's method of shrinking to the rejected point. If the initial sampling range is not well chosen, the faster shrinking allowed here by setting $\Delta l > 1$ enables more rapid diffusion because an admittedly poorer acceptable jump is found more quickly.

A feature of using the integer representation is that, with a suitably extended number of bits, the single integer X can represent two or more real parameters, for example, by mapping X to (x_1, x_2, x_3) through a space-filling curve. Thus multidimensional slice sampling can be performed using the same software as for one dimension. Peano curves are useful here because they relate conveniently to a rectangular grid and they have the best possible locality properties: nearby points on the curve are close in space (though not the converse, which is unattainable). In this case, each successive bit of X represents a factor of 2 in volume. Because we are likely to be uncertain about the optimal sampling volume in several dimensions, it may be helpful to set Δl to the dimensionality. Taking Δl at each step from any preassigned distribution (which may include $\Delta l = 0$) allows extra flexibility.

DEPARTMENT OF APPLIED MATHS
AND THEORETICAL PHYSICS
WILBERFORCE ROAD
CAMBRIDGE CB3 0WA
UNITED KINGDOM

CAVENDISH LABORATORY
MADINGLEY ROAD
CAMBRIDGE CB3 0HE
UNITED KINGDOM
E-MAIL: mackay@mrao.cam.ac.uk

DISCUSSION

BY S. G. WALKER

University of Bath

Suppose that $f(x)$ is a target density to be sampled. One well-known idea is to introduce a latent variable and the conditional density $f(u|x)$ and construct a Gibbs sampler with transition density given by

$$K(x, x') = \int f(x'|u) f(u|x)\, du,$$

which can be written as

$$Q(x, x') = f(x) K(x, x') = \int f(x'|u) f(x|u) f(u)\, du.$$

Clearly $Q(x, x')$ is symmetric; that is, $K(x, x')$ satisfies detailed balance with respect to $f(x)$. To assist the Gibbs sampler it might be desirable to change $f(x'|u)$ to some alternative density function $f^*(x'|u, x)$, so that the new x value depends on u and the old x value. To maintain symmetry, it is sufficient that

$$f_u^*(x'|x) f_u(x) = f_u^*(x|x') f_u(x'),$$

where, for example, $f_u(x) = f(x|u)$. That is, $f_u^*(x'|x)$ satisfies detailed balance with respect to $f_u(x)$, for all u. One idea is to use a Metropolis–Hastings transition density with symmetric proposal density $q_u(x, x')$; that is,

$$\begin{aligned} &f_u(x) f_u^*(x'|x) \\ &\quad = \min\{f_u(x), f_u(x')\} q_u(x, x') + \{1 - r_u(x)\} f_u(x) \mathbf{1}(x = x'), \end{aligned}$$

where

$$r_u(x) = \int \min\{1, f_u(x')/f_u(x)\} q_u(x, x')\, dx'.$$

When $f_u(x)$ is uniform on some interval, as it would be in the case when $f(u|x) = \mathbf{1}\{u < f(x)\}/f(x)$, then it is sufficient that

$$f_u^*(x'|x) = f_u^*(x|x')$$

and this is the basis for Neal's research.

The essence of Neal's idea is to generate random intervals about the current x point which, with probability 1, do not have end points belonging to the set

$$A_u = \{x : u < f(x)\}.$$

A new x' is sampled from this interval. If f is unimodal then

$$f_u^*(x'|x) = 1/|A_u|.$$

The idea is ingeneous and, as demonstrated by Neal in Section 8 of his paper, can do better than alternatives, such as the Metropolis–Hastings algorithm.

However, I have always felt that single-variable slice samplers have severe limitations. For complex models a single-variable slice sampler is not going to work in the majority, if not all, of the cases. Although if it can be made to work, then there are some attractive mathematical properties associated with the single variable slice sampler [Roberts and Rosenthal (1999)].

Many-variable slice samplers have a good chance of working, though as with all algorithms, not always. If there is a latent variable for each data point then problems may arise if there is a lot of data, obviously. Severe autocorrelation can occur, though not always. A complicated model is dealt with in Damien, Wakefield and Walker (1999) based on many variable slice samplers; it would be interesting

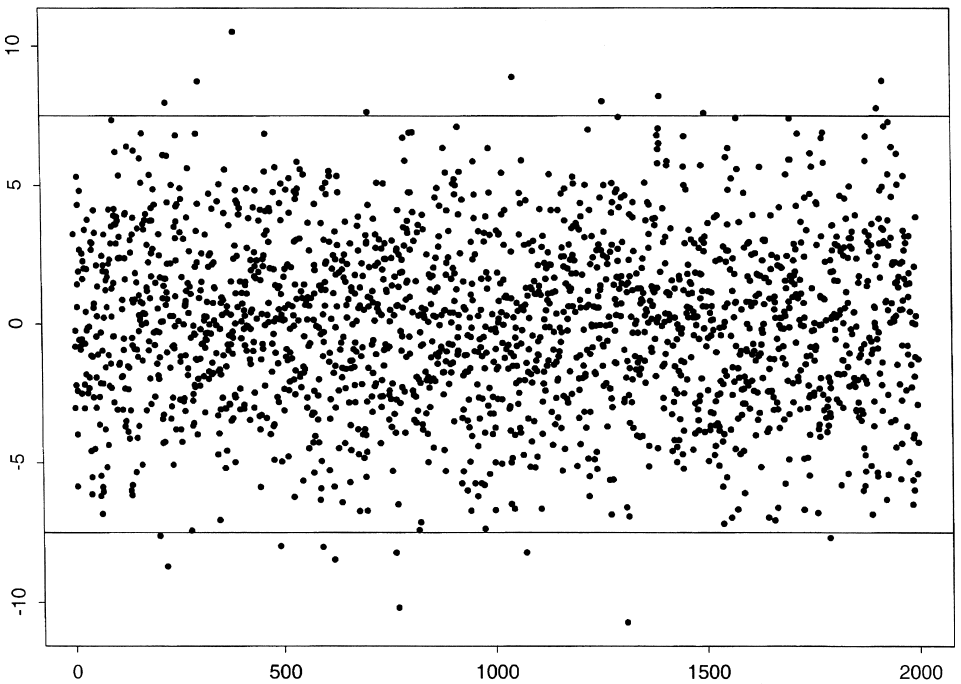
to see if Neal's idea works in any of the nonconjugate and/or hierarchical models which appear in Damien, Wakefield and Walker (1999).

I took Neal's example in Section 8 and used a many-variable slice sampler on it. In fact I took a latent variable for each data point, the idea criticized by Neal. The overall joint density is given by

$$f(x_1,\ldots,x_9,v,z,u_1,\ldots,u_9) \propto \mathbf{1}\{z < \exp[-(v+81/2)^2/18]\} \times \prod_{i=1}^{9} \exp(-u_i/2)\mathbf{1}\{u_i > \exp(-v)x_i^2\}.$$

The extra z variable is more than is required, but it saves having to sample a truncated normal distribution within each sweep of the sampler. In fact, with this many-variable slice sampler, all variables can be sampled via the use of uniform random variables. The full conditionals are all easy to find.

Comparing statistics discussed in Neal's paper, based on a run of size 1,000,000, the percentage of points below -5 was 4.87% and the percentage above 7 was 0.5%, which are both healthy outcomes. High autocorrelation was not really a problem: I took 2,000 from a run of 1,000,000; that is, every 500th sample, and the output is presented in the figure. For comparison with Neal's figures the horizontal lines are at $+7.5$ and -7.5.



Sampling from the funnel distribution using many variable slice sampler

REFERENCES

DAMIEN, P., WAKEFIELD, J. C. and WALKER, S. G. (1999). Gibbs sampling for Bayesian nonconjugate and hierarchical models by using auxiliary variables. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **61** 331–344.

ROBERTS, G. O. and ROSENTHAL, J. S. (1999). Convergence of slice sampler Markov chains. *J. R. Statist. Soc. Ser. B* **61** 643–660.

DEPARTMENT OF MATHEMATICAL SCIENCES
UNIVERSITY OF BATH
BATH, BA2 7AY
UNITED KINGDOM
E-MAIL: S.G.Walker@bath.ac.uk

REJOINDER

BY RADFORD M. NEAL

University of Toronto

I thank the discussants for raising a number of interesting issues. Since many of these issues are related, I will address them by topic. I will first make some general comments on slice sampling, in its ideal and in its practical forms. I will then discuss the three alternatives to slice sampling that were mentioned by Walker, by Chen and Schmeiser, and by Mira and Roberts. Finally, I will investigate whether faster shrinkage of the sampling interval can improve slice sampling, as suggested by Skilling and MacKay.

1. The nature and properties of slice sampling. As I discuss in Section 3 of the paper, the general auxiliary variable technique of which slice sampling is a particular case goes back to Edwards and Sokal's (1988) generalization of the Swendsen–Wang algorithm for Ising systems. In this original context, multiple auxiliary variables were used, one for each "bond" between "spins" in the system. For the Ising system, this works very well. One theme of my paper, however, is that methods based on a *single* auxiliary variable work well for many distributions, partly because practical implementations of these methods can *adapt* to the characteristics of the distribution being sampled.

Readers should note that since introducing the term "slice sampling" [Neal (1997)], I have always used it to refer to such methods based on a single auxiliary variable. Thus, the reference in the title of Section 4 of this paper to "single-variable slice sampling" is *not* meant to distinguish the methods of that section from those based on multiple auxiliary variables. Rather, the methods described in that section update a single real variable with the help of a single auxiliary variable, in contrast to the multivariate methods of Section 5, which simultaneously update

several real variables with the help of a single auxiliary variable. The usage of the phrase "single-variable slice sampler" in the discussions by Walker and by Mira and Roberts is not the same as my usage of the phrase, in the paper and in this rejoinder.

Note also that, contrary to the impression one might get from the description given by Walker, the slice sampling methods in my paper can generally *not* be interpreted as using the Metropolis algorithm to update x in a manner that leaves the uniform distribution over the current slice invariant. Updating x in this way would be valid, but unless the Metropolis proposal depends on the slice level (called y in Section 4), the result would be no different from a standard Metropolis update with the same proposal distribution. Slice sampling methods such as the one illustrated in Figure 1 of the paper have two features that clearly distinguish them from Metropolis updates. First, they potentially evaluate $f(x)$ at *many* points, when expanding the interval and when shrinking it. Second, the point finally accepted will with probability one be different from the current point, in contrast to Metropolis updates, which (apart from degenerate cases) have a positive probability of rejecting the proposed new point, and remaining at the current point. These two features are linked—by using the information from rejected points, slice sampling methods can adapt the distribution of later points so as to efficiently locate an acceptable new point that differs from the current point.

Such an adaptive approach would not be necessary if one could implement the "ideal" slice sampler—in which we alternately sample y uniformly from $(0, f(x))$, and x uniformly from $\{x : y < f(x)\}$, independently of the previous x. As discussed by Mira and Roberts, strong convergence guarantees can sometimes be given for ideal slice samplers. In practice, however, implementing the ideal slice sampler will, for many multivariate problems, be difficult, inefficient, or impossible. (Ideal single-variable slice sampling may be more feasible, but when a multivariate distribution is sampled using single-variable slice sampling for each variable in turn, the attractive convergence guarantees do not apply.) It is presumably the ideal slice sampler that Walker has in mind when he states that the slice sampler "is not going to work in the majority, if not all, of the cases." This is of course why almost all of my paper is devoted to non-ideal slice samplers.

Although the positive theoretical results for ideal slice samplers are of limited practical interest, negative results for the ideal slice sampler are likely to apply to practical slice samplers as well, since it is doubtful that sampling x independently will be much worse than updating x in a way that is dependent on its previous value. Mira and Roberts mention the worsening performance of multivariate slice samplers when the dimensionality increases, even for log-concave distributions. One way of understanding this is to look at how the log probability density of x changes as a result of a slice sampling update. As mentioned in Section 4, one can define the slice in terms of $g(x) = \log(f(x))$, as $S = \{x : z < g(x)\}$, where $z = g(x_0) - e$, with e being drawn from the exponential distribution with mean one. The new state, x_1, drawn from S will have $g(x_1) > g(x_0) - e$, so any decrease

in the log probability density is limited by the value of e chosen. Since e is typically of order one, we will explore different log probability densities by steps of that magnitude. Moreover, this exploration will take the form of a random walk. For many multivariate distributions, such as the Gaussian, the distribution of $g(x)$ will have a width proportional to $\sqrt{d}$, where d is the dimensionality. Exploring the range of values for $g(x)$ by a random walk with steps of order one will then take order d updates. This is unfortunate, but one should note that the Metropolis algorithm is no better in this respect [Caracciolo, Pelissetto and Sokal (1994)]. Note also that this limit is relevant only for multivariate slice sampling. For single-variable slice sampling, we would expect in any case that at least d updates (one for each variable) will be needed to move to a largely independent point, but these updates may be faster than multivariate updates, if they can recompute $f(x)$ by an incremental method.

As Mira and Roberts suggest, it would nevertheless be interesting to look for a generally-usable version of the "polar slice sampler" of Roberts and Rosenthal (2002), which avoids this problem. One obstacle to this is the need to fix an appropriate origin for the polar coordinates, which may be difficult if the user has no detailed knowledge of the distribution.

Mira and Roberts also note that single-variable slice sampling implemented using the "stepping-out" procedure can fail to be irreducible if there are regions in which the probability density is zero. This is true. Unfortunately, the alternative "doubling" strategy does not necessarily avoid this problem, since although it will then always be possible for a point anywhere else in the distribution to be proposed, it is possible that some points will never pass the acceptance test of Figure 6. (This will be the case, for instance, if the distribution is uniform over $[0, 0.2] \cup [1.5, 1.6]$ and $w = 1$.) Irreducibility of a single-variable slice sampler can be ensured by selecting w from a distribution with an infinite tail. However, when a multivariate probability density is zero in some places, sampling by successively applying a single-variable slice sampler to each variable may fail to be irreducible even if each single-variable slice sampler is irreducible, just as Gibbs sampling can fail to be irreducible. Irreducibility of single-variable updating methods in such circumstances can be determined only by considering the specifics of the distribution.

Although I focus in the paper on slice sampling methods that do not rely on $f(x)$ having any convenient form, problems where $f(x)$ is particularly tractable do arise. The contribution by Downs is an interesting example, in which reflective slice sampling can be implemented exactly, without the need to approximate the reflection points. One future challenge is to recognize such situations when the sampler is generated automatically from a model specification.

2. How does slice sampling compare to alternative methods? Three alternatives to slice sampling are advocated by the discussants, which I will discuss in turn.

2.1. *Methods using multiple auxiliary variables.* Damien, Wakefield and Walker (1999) propose samplers based on multiple auxiliary variables. In some cases, their method is essentially that of Edwards and Sokal (1988), which as discussed in Section 3 of the paper, is based on factoring the probability density as $p(x) \propto f_1(x) \cdots f_k(x)$, and introducing k auxiliary variables, $y_1, \ldots, y_k$, one for each factor. Defining the joint density for x and the y_i as $f(x, y_1, \ldots, y_k) \propto \prod_i I\{0 < y_i < f_i(x)\}$ results in the desired marginal distribution for x. The strategy of Damien, Wakefield and Walker is to choose such a factorization of the density that makes all the conditional distributions for the y_i and the components of x easy to sample from, as will be the case if the equations $f_i(x) = y_i$ that determine the bounds for x given the current y_i can be solved analytically. (Damien, Wakefield and Walker, henceforth DWW, also sometimes employ auxiliary variables in a somewhat different way, and for reasons that are unclear to me, they often do not introduce an auxiliary variable for the factor in the posterior density that corresponds to the prior, so the conditional distribution of x is often not uniform within its bounds, but is instead a truncated form of some standard distribution.)

DWW apply this strategy to several Bayesian inference problems. In his discussion, Walker asks whether the slice sampling methods of my paper will work for any of these problems. The answer is that all the methods in my paper can be applied to all the problems treated by DWW. The only requirement for using these methods is the ability to evaluate a function proportional to the posterior density, and for some of the methods, to evaluate the gradient of this function. This is not difficult for the problems of DWW. Two questions remain: "How efficient are the slice sampling methods compared to that of DWW?", and "How do these methods compare with respect to ease of use?"

I argue in Section 3 of the paper that the approach of DWW will be inefficient when there are many data points, if each data point gives rise to a new auxiliary variable, corresponding to a factor in the likelihood for that point, with the number of parameters being fixed. Example 5 of DWW is of this type. This example is a simple logistic regression model, in which n independent Bernoulli data points, w_i, are modeled in terms of corresponding explanatory variables, z_i, as $w_i \sim \text{Bernoulli}[1/(1 + \exp(-xz_i))]$. (Here I have set the constant μ of DWW to zero.) The problem is to sample from the posterior distribution of the single model parameter, x, whose prior is assumed to be $N(0, 1)$.

DWW introduce two auxiliary variables for each data point, corresponding to factors in the likelihood of $1/(1+\exp(-xz_i))^{w_i}$ and $1/(1+\exp(xz_i))^{1-w_i}$ (though for any particular data point, only one of these auxiliary variables will have an effect). Sampling from the conditional distributions of these auxiliary variables given x is easy. A formula for sampling from the conditional distribution of x given the auxiliary variables is provided by DWW, but it is not correct. One problem is that a minus sign is missing before the log in their definition of "a_i." Fixing this produces a method that works as long as all the explanatory variables are non-negative. Negative z_i cause the directions of the relevant inequalities to be

reversed, necessitating a procedure that is somewhat more complex than that given by DWW.

The problems with this example illustrate that the effort required to correctly apply the approach of DWW is not completely trivial, even for a simple model. For more complex problems, the exploration of alternative models that is a desirable part of data analysis may be inhibited if each model must be manually implemented in this manner, even if these implementations are in some sense straightforward. For comparison, slice sampling requires only that functions for evaluating the prior density and the likelihood be provided, which is about the minimum that can be expected, along with rough estimates of the widths of the conditional distributions. How sensitive the slice sampler is to the width estimate for this example will be explored below.

To compare the efficiency of slice sampling with the method of DWW, I randomly generated data sets with $n = 20$, $n = 100$ and $n = 500$ observations, with the parameter x set to 2 and with $z_i \sim N(0, 1)$. For each data set, I simulated 60,000 iterations of several slice sampling methods and of a corrected version of the multiple auxiliary variable method of DWW. The first 10,000 iterations were discarded as "burn-in" (more than enough), and the remaining 50,000 iterations were used to estimate autocorrelation times (one plus twice the sum of autocorrelations at lags one and up), which measure the degree of inefficiency in estimation. Autocorrelation times were estimated both for x and for the log of the posterior density at x.

For the method of DWW, the autocorrelation time for x increased from 6.7, to 29, to 187 as the size of the data set increased from 20, to 100, to 500. The autocorrelation time for the log posterior density showed a similar pattern, increasing from 3.5, to 14, to 86. The autocorrelation times for both functions are approximately proportional to the sample size, as would be expected from my discussion in Section 3. The computation time per iteration is also proportional to n, so the total cost of obtaining estimates of a given accuracy grows in proportion to n^2.

For slice sampling, the autocorrelations for x were almost zero, for all sample sizes, giving autocorrelation times of 1.1 or less. The autocorrelation times for the log posterior density ranged from 1.8 to 2.1. Because the posterior for this model is unimodal, these autocorrelation times do not depend on the method of finding the sampling interval (as long as an interval with endpoints outside the slice is found), but the method used does affect the computation time per iteration. For these tests, I used the doubling and shrinkage procedures of Figures 4 and 5, and I included the acceptability test of Figure 6, even though it is not required for unimodal distributions, so that the results will be indicative of performance more generally.

The computational cost of slice sampling can be assessed by the number of evaluations of the likelihood in each iteration (the dominant cost when n is large). When the width, w, of the initial interval was 1, the average number of likelihood

evaluations per iteration was 9.3, 8.5 and 6.8 when n was 20, 100 and 500. When w was set to 100, the average number of evaluations was 9.8, 10.2 and 11.8, and when w was set to 0.01, the average number of evaluations was 22.6, 21.8 and 19.5. For comparison, one iteration of the method of DWW takes roughly the same time as two evaluations of the likelihood, since the likelihood factors are needed for sampling the auxiliary variables, and corresponding quantities are then calculated when sampling for x. In both cases, we must multiply the computational cost per iteration by the autocorrelation time to get a measure of the overall cost.

Slice sampling works reasonably well in this example even when the tuning parameter, w, is a factor of 100 away from its best value. Its cost grows only in proportion to n. For small n, the method of DWW has a cost comparable to slice sampling. When $n = 500$, however, the method of DWW has a cost of roughly $187 \times 2 = 374$ for estimating $E[x]$, compared to a cost of $1.1 \times 19.5 = 21.5$ for slice sampling, assuming one uses the poorly-chosen value for w of 0.01. The method of DWW is costly in terms of computation time for this problem, and is also harder to implement.

Some of the other examples of DWW have a different character, in which the number of auxiliary variables grows with the number of random effects. In this context, the slowdown as the sample size grows may be less pronounced, since the extra auxiliary variables do not all interact directly. Walker's solution to the problem of sampling from the "funnel" distribution I use as an example in Section 8 of the paper is of this type. Sampling from the conditional distributions for each u_i and x_i in his solution is equivalent to single-variable slice sampling with the endpoints found analytically. Only in the way v is sampled does Walker's solution differ from slice sampling, since the range for v will depend on the u_i, in contrast to single-variable slice sampling, in which the auxiliary variable used to sample for one real variable is forgotten before the next is sampled for.

The point of my "funnel" example was to show that, due to its adaptive nature, slice sampling avoids the difficulties that the Metropolis algorithm is prone to. The big advantage of the approach of DWW is that adaptation is not necessary, since the need for tuning is avoided by converting the problem to one in which the conditional distributions can be handled analytically. It is therefore not surprising that their approach works well in this example, where the distribution was deliberately kept simple, so that the true solution will be known, although the slice sampling and Metropolis methods used were deliberately of a general nature, that did not exploit this simplicity. It is not so clear that the approach of DWW will work well in most practical problems, where such general methods may be required.

2.2. *The random-direction method.* Another proposed alternative to slice sampling is the "black box" sampler of Chen and Schmeiser (1998), which like slice sampling is based on sampling uniformly from under the plot of the density function. Rather than sample vertically and then from a horizontal slice,

however, Chen and Schmeiser propose new points by moving in random directions. As I argue in the paper, Chen and Schmeiser's method is not really a "black box" algorithm—it requires just as much tuning as a random-walk Metropolis algorithm, and this tuning is made harder by its dependence on the normalizing constant of the distribution. I can therefore see no reason why one would ever use this algorithm.

In their discussion, Chen and Schmeiser point to the fact that their method requires only one evaluation of the density per iteration as an advantage over slice sampling. As I discuss above, it is by using information from several evaluations of the density that slice sampling can adapt to the characteristics of the distribution. A method that evaluates the density at only one point must rely on this point being chosen well, using a proposal distribution defined *a priori*. If one chooses to use this strategy, the simplest and most flexible way of proceeding would seem to be to construct an ordinary Metropolis–Hastings algorithm.

2.3. *Delayed rejection methods.* Adaptation based on multiple evaluations of the density is also a characteristic of the "delayed rejection" method [Mira (1998), Tierney and Mira (1999)] mentioned by Mira and Roberts. In this extension of the Metropolis algorithm, rejection of the first point proposed leads to a second point being proposed, whose distribution may be different from that of the first, and perhaps depend on the first point. Acceptance or rejection of this second proposal is done so as to preserve reversibility. A third proposal may be made if the second is rejected, and so on. Mira (1998) and Green and Mira (2001) discuss a number of variations on this technique. One suggestion they make is to first propose a point from a broad distribution centred on the current point, and if that is rejected, to propose a point from a narrower distribution centered on the current point, and so on, with narrower and narrower proposal distributions.

As Mira and Roberts note, this form of delayed rejection is similar in concept to the shrinkage procedure of Figure 5 in the paper, and to the procedure based on Gaussian crumbs described in Section 5.2. I believe that the slice sampling procedures are simpler and more efficient, however. Green and Mira [(2001), Section 3.3] give an example of a two-stage random-walk procedure with different variances, in which computing the second-stage acceptance probability requires the evaluation of the density at a third point, in addition to the evaluations of the density at the two proposed states. They do not attempt to implement third or higher stages, due to the rapid increase in complexity. The symmetric delayed rejection method [Mira (1998), Section 5.3] is simpler, but unfortunately does not allow second and later proposals to be centered on the current point—they must instead be based on the last rejected point, which seems inappropriate. The more general form of the procedure [Mira (1998), Section 5.2] allows all proposals to be centered on the current point, but the acceptance criterion is complex, and it appears that in its higher stages it would usually reject a proposal that came from

an appropriately narrow proposal distribution. A modified acceptance criterion in which the sequence of rejected points is not reversed might be better in this respect.

In contrast, the analogous single-variable slice sampling procedure (with no expansion of the initial interval, but with subsequent shrinkage to rejected points) requires evaluation of the density only at points that are proposed, and can be continued for any number of proposals without any difficulty.

3. Can we improve slice sampling by shrinking the interval faster? Skilling and MacKay describe an interesting version of slice sampling based on integer arithmetic, transformation of parameters to a fixed interval, and shrinkage to binary division points in the interval rather than to the rejected points. The use of integers and of Peano curves to transform a two-dimensional space to a one-dimensional interval may be suitable for some applications, such as in image processing, but they seem to be unnecessary for typical Bayesian inference problems. The shrinkage procedure used, which can shrink the interval faster than my procedure of shrinking to the rejected point, raises an issue of more general interest, since it could be used in any context.

The issue is somewhat subtle. As Skilling and MacKay acknowledge, shrinking to the midpoint of the current interval can cut off part of the slice, with the result that the point finally accepted will on average be closer to the current point than would have been the case if we had shrunk to the rejected point. These smaller movements will increase the autocorrelation time for the chain by some constant factor. On the other hand, if the interval chosen initially is too large, shrinking to the midpoint will cut the interval in half for each rejected point, causing it to shrink exponentially with rate $\log(2) = 0.693$. Shrinking to the rejected point will also result in exponential shrinkage, but with an average rate that can be calculated to be only $-\int_0^1 [\int_0^x \log(1-u)\,du + \int_x^1 \log(u)\,du]\,dx = 1/2$. If the initial interval is much too large, shrinking to the midpoint will therefore require a factor of $0.693/0.5 = 1.386$ fewer density evaluations, which for many problems will be the dominant computational cost. The question is whether or not this constant factor savings in time per iteration outweighs the constant factor increase in autocorrelation time.

I have done experiments addressing this question for a univariate Gaussian distribution and for a bivariate Gaussian distribution with high correlation. In these experiments, I also tested two combined shrinkage methods inspired by Skilling and MacKay's discussion. The first shrinks first to the rejected point, and then to the midpoint of the interval remaining after shrinkage to the rejected point. It shrinks exponentially at a rate of $\log(2) + 1/2 = 1.193$. The second new method is the same except that shrinkage to the midpoint occurs only if the log probability density of the rejected point is less than that of the slice level minus some threshold (I used a threshold of 100). The hope is that this method will be able to shrink rapidly without a substantial increase in autocorrelation time, since

TABLE 1

Performance of various methods of shrinking the slice sampling interval on univariate and bivariate ($\rho = 0.999$) Gaussian distributions. The table shows the average number of density evaluations in one slice sampling update ($\bar{e}$), the autocorrelation times for the (first) variable and for the log probability density, and the products of autocorrelaton times and numbers of evaluations. For the bivariate Gaussian, autocorrelations are for iterations that update each variable 100 times using single-variable slice sampling

	Univariate Gaussian					**Bivariate Gaussian**				
Shrinkage method	$\bar{e}$	τ_x	τ_ℓ	$\bar{e}\tau_x$	$\bar{e}\tau_\ell$	$\bar{e}$	τ_x	τ_ℓ	$\bar{e}\tau_x$	$\bar{e}\tau_\ell$
To rejected point	10.7	1.0	2.0	11	22	7.8	10.0	3.1	78	24
To midpoint	8.1	1.7	2.5	14	20	6.0	17.7	4.8	106	29
Combined rejected/mid	5.7	2.1	2.5	12	14	4.5	22.1	5.9	99	27
Combined w. threshold	6.8	1.2	2.0	8	14	5.5	10.7	3.3	59	18

the extra shrinkage may be disabled before it can lead to part of the slice being cut off.

The univariate Gaussian had standard deviation one. I used an interval of size $w = 1000$ (with no stepping out or doubling) to simulate a situation where the width of the interval was chosen to be much too large. I measured the average number of density evaluations per slice sampling update, which in more complex problems would be a good indicator of time per iteration, and the autocorrelation times for the variable x itself and for the log probability density, ℓ. The results are shown in Table 1. As expected, the number of density evaluations is less for the methods that can shrink faster. The autocorrelation times for x, and to a lesser extent ℓ, are also higher, however. Looking at the product of number of evaluations and autocorrelation time, we see that only the combined method with the threshold for shrinking to the midpoint has a clear advantage over simply shrinking to the rejected point.

The bivariate Gaussian had marginal standard deviations of one and correlation 0.999, which produces conditional standard deviations of 0.0447. I used an interval of size $w = 10$ (with no stepping out or doubling), which is again much too large (224 times the conditional standard deviation). The number of density evaluations was again less for the faster-shrinking methods. The autocorrelation times (measured using points separated by 100 slice sampling scans) were substantially larger for both τ_x and τ_ℓ when shrinkage to the midpoint was done unconditionally, with or without shrinkage to the rejected point. (For comparison, Gibbs sampling produced autocorrelation times of $\tau_x = 9.7$ and $\tau_\ell = 2.7$.) The only scheme that did better than simply shrinking to the rejected point was the combined scheme with a threshold for shrinkage to the midpoint, which was better overall by a factor of about 1.3.

It would be interesting to test these methods in the context described by Skilling and MacKay, in which a two-dimensional parameter is mapped to one dimension

using a Peano curve. The slice for a bivariate Gaussian might then consist of several disjoint sections, and the behavior of the methods might be different.

In the more conventional context I looked at, it appears that only a modest gain can be obtained from faster shrinkage, even when the initial interval is much too wide. However, if a large threshold is set for further shrinkage (after shrinking to the rejected point), there may not be much to lose, so perhaps this could be a standard technique. An indirect benefit of this would be that the strategy of setting the initial interval to the entire range of the variable being updated would become more attractive. By eliminating the need to select w, this would make slice sampling easier to use.

REFERENCES

CARACCIOLO, S., PELISSETTO, A. and SOKAL, A. D. (1994). A general limitation on Monte Carlo algorithms of Metropolis type. *Phys. Rev. Lett.* **72** 179–182.

CHEN, M.-H. and SCHMEISER, B. W. (1998). Toward black-box sampling: A random-direction interior-point Markov chain approach. *J. Comput. Graph. Statistics* **7** 1–22.

DAMIEN, P., WAKEFIELD, J. C. and WALKER, S. G. (1999). Gibbs sampling for Bayesian nonconjugate and hierarchical models by using auxiliary variables. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **61** 331–344.

EDWARDS, R. G. and SOKAL, A. D. (1988). Generalization of the Fortuin–Kasteleyn–Swendsen–Wang representation and Monte Carlo algorithm. *Phys. Rev. D* **38** 2009–2012.

GREEN, P. J. and MIRA, A. (2001). Delayed rejection in reversible jump Metropolis–Hastings. *Biometrika* **88** 1035–1053.

MIRA, A. (1998). Ordering, splicing and splitting Monte Carlo Markov chains. Ph.D. dissertation, School of Statistics, Univ. Minnesota.

NEAL, R. M. (1997). Markov chain Monte Carlo methods based on "slicing" the density function. Technical Report 9722, Dept. Statistics, Univ. Toronto.

ROBERTS, G. O. and ROSENTHAL, J. S. (2002). The polar slice sampler. *Stoch. Models* **18** 257–280.

TIERNEY, L. and MIRA, A. (1999). Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine* **18** 2507–2515.

DEPARTMENT OF STATISTICS
UNIVERSITY OF TORONTO
100 ST. GEORGE STREET, 6TH FLOOR
TORONTO, ONTARIO
CANADA M5S 3G3
E-MAIL: radford@stat.utoronto.ca