Markov Chain Monte Carlo Algorithms Allowing Parallel Processing –II

Guthrie Miller*

Los Alamos National Laboratory, Group RP-2, USA

Abstract: A variant of the Metropolis-Rosenbluth-Teller algorithm that allows parallel processing has been described in a previous paper ("Markov Chain Monte Carlo Calculations Allowing Parallel Processing Using a Variant of the Metropolis Algorithm") that appeared in this journal in 2010. In this follow-on paper, the new algorithm as well as the Metropolis-Rosenbluth-Teller and Barker algorithms are analyzed for finite, integer-valued Markov Chains, which are easier to understand in detail than continuous variable chains. The new algorithm is shown to approximately satisfy detailed balance when the random walk step is much larger than the support region of the desired steady-state distribution function. Parallelizable versions of the MRT and B algorithms are given. The time to reach a steady state is calculated and compared for these three different algorithms and different number of multiple candidates, potentially offering different degrees of parallel processing.

Keywords: Metropolis algorithm, parallel processing, markov chain monte carlo (MCMC), bayesian statistics, data analysis.

1. INTRODUCTION

Markov Chain Monte Carlo is a very powerful and useful technique in Bayesian data analysis as well as in statistical physics, where it originated with the 1953 paper [1] by Nicolas Metropolis, Adriana Rosenbluth, Marshall Rosenbluth, Agusta Teller, and Edward Teller (MRT algorithm). Another algorithm was proposed by Barker [2] in 1965 (B algorithm). The new algorithm considered here was described and demonstrated in a preceding paper [3]; however, that paper contains a serious oversight in failing to note that detailed balance is no longer satisfied exactly with the new algorithm. This would seem to be a very serious problem and it is addressed in the present paper. For reasons of simplicity and clarity, the analysis is done for finite, integer-valued Markov Chains. This type of analysis is similar to that of Rubenstein *et al.* [4].

The preceding paper [3], which is easily assessable through open access, provides general background and literature references. In addition to the references cited there, references [5-7] may be helpful to the reader.

The motivation for this work is to find Markov Chain Monte Carlo algorithms that allow parallel processing, because increases in computer processing power in the future will more and more come about through having multi-processing systems. To take advantage of such systems for a single problem requires a parallel algorithm, pieces of which can be executed in parallel on many different processors.

2. THE ALGORITHMS

We begin by simply stating the algorithms for the case of continuous variables, because henceforth everything else will be done for integer variables. In this way the motivation for the work with integer variables will be clearer.

The Markov Chain algorithm (the rule telling the computer how to select the next point θ' in parameter space, given that the chain is at a current point θ) is the following. First *l* candidates for the new point, labeled by *i = 1, l*, are generated from a conditional probability distribution $q(\theta' \mid \theta)$ (read as "the probability distribution of θ' given, or conditioned on, θ"). This "candidate" probability distribution is chosen to be a random walk ($\theta' = \theta + \Delta(x - 1/2)$, where *x* is a random number uniformly distributed between 0 and 1, and Δ is a fixed parameter, for all or some subset of the dimensions of θ'. The "energy" calculations for the *l* candidates can be done using parallel processing.

The chain is moved to the next point *i = 0, l* with discrete probability $\alpha(\theta_i', \theta)$, where the point *i* = 0 represents the unchanged current point.

Three different Markov Chain algorithms are considered. These will be stated in terms of the candidate distribution and the desired steady state distribution function *f*.

The Barker (B) algorithm, as generalized to include a general candidate distribution by Hastings [8], has

$$\alpha(\theta_i', \theta) = \frac{1}{l} \frac{\dfrac{f(\theta_i')}{q(\theta_i' \mid \theta)}}{\dfrac{f(\theta_i')}{q(\theta_i' \mid \theta)} + \dfrac{f(\theta)}{q(\theta \mid \theta_i')}}, \quad i > 0 \ , \tag{1}$$

$$\alpha(\theta, \theta) = *$$

where $*$ denotes what is required by normalization, namely

$$\alpha(\theta, \theta) = 1 - \sum_{i=1}^{l} \alpha(\theta_i', \theta)$$

One can verify the detailed balance condition

$$\frac{q(\theta \mid \theta_i')\alpha(\theta \mid \theta_i') f(\theta_i')}{q(\theta_i' \mid \theta)\alpha(\theta_i' \mid \theta) f(\theta)} = 1 \ .$$

The Metropolis-Rosenbluth-Teller algorithm, has

*Address correspondence to this author at the Los Alamos National Laboratory, Group RP-2, USA; Tel: 505 667 5547; Fax: 505 665 6071;
E-mails: guthriemiller@earthlink.net; guthrie@lanl.gov

$$\alpha(\theta_i',\theta) = \frac{1}{l}\min\left(\frac{f(\theta_i')}{q(\theta_i'\,|\,\theta)}\frac{q(\theta\,|\,\theta_i')}{f(\theta)},1\right) \quad i>0 \ , \qquad (2)$$
$$\alpha(\theta,\theta) = *, \quad i=0$$

and one can similarly verify detailed balance.

For the new algorithm,

$$\alpha(\theta_i',\theta) = \frac{\dfrac{f(\theta_i')}{q(\theta_i'\,|\,\theta)}}{\sum\limits_{k=0}^{l}\dfrac{f(\theta_k')}{q(\theta_k'\,|\,\theta)}} \ . \qquad (3)$$

For $i = 0$ in the above, θ_i' is to be replaced by θ. The detailed balance relationship is

$$\frac{q(\theta\,|\,\theta_i')\alpha(\theta\,|\,\theta_i')f(\theta_i')}{q(\theta_i'\,|\,\theta)\alpha(\theta_i'\,|\,\theta)f(\theta)} = \frac{\sum\limits_{k=0}^{l}\dfrac{f(\theta_k')}{q(\theta_k'\,|\,\theta)}}{\sum\limits_{k'=0}^{l}\dfrac{f(\theta_{k'}')}{q(\theta_{k'}'\,|\,\theta_i')}} \ .$$

When the candidate distribution $q(\theta_k'\,|\,\theta)$ is independent of the initial point θ , this is exactly equal to 1 for $l = 1$ (it reduces to the B algorithm) and approximately equal to 1 for any l when the region explored by the candidate distribution is large enough to encompass the support region of the desired steady state distribution function f.

In the random walk situation q is simply either 0 or a constant

$$q(\theta'\,|\,\theta) = \frac{1}{\Delta} \quad , |\theta'-\theta| < \frac{\Delta}{2} \ ,$$

and it drops out of the formulas.

Note that for the MRT and B algorithms with multiple candidates, the acceptance probability is just the average over the candidates of the original, single-candidate expressions.

A single Markov chain is run; however for each iteration l candidates for the next position of the chain are generated. Only a single one of these candidates will be probabilistically selected as the next position of the chain. One of these candidates is the current chain position.

A simple example of the use of these algorithms is given in Appendix **1**.

3. ANALYSIS FOR FINITE, INTEGER-VALUED MARKOV CHAINS

For finite, integer-valued Markov Chains it is possible to fairly simply carry out a comprehensive mathematical analysis.

Markov Chains model the ergodic theorem of statistical mechanics, where time averages are shown to be equal to ensemble averages. In the present application of Markov Chains, ensemble averages will be replaced by time averages.

Consider an integer-valued function of time $i(t)$ representing some dynamical variable. Imagine an ensemble of dynamical systems with different initial conditions. Averaging over the entire ensemble there is some probability that system is in state i at time t, given by

$$f_i(t) \equiv P(i\,|\,t) \ .$$

The definition of a Markov Chain is that

$$f_i(t+1) = \sum_j A_{i,j} f_j(t) = \left(Af(t)\right)_i \ , \qquad (4)$$
$$A_{i,j} \equiv P(i\,|\,j) = (A)_{i,j}$$

where the conditional transition probabilities $P(i\,|\,j)$ are independent of time, and we have used matrix notation to denote the square matrix of transition probabilities A and the column matrix $f(t)$ representing the distribution function at time t. Because probabilities must be normalized,

$$\sum_i f_i(t) = 1$$
$$\sum_i A_{i,j} = 1 \ .$$

Also, it is clear from Eq. (4) that

$$f(t) = \underbrace{A\times A\times A\ldots}_{t\ times} f(0) = A^t f(0) \ , \qquad (5)$$

where A^t denotes the t^{th} power of the matrix A (the matrix multiplied by itself t times).

The ergodic theorem deals with time averages of an arbitrary function g of the dynamical variable. Such time averages are seen to be equal to averages over the steady state distribution function of the system

$$\lim_{T\to\infty}\left(\frac{1}{T}\sum_{t=1}^{T} g\big(i(t)\big)\right) = \sum_i \bar{f}_i g(i) \ ,$$

where,

$$\bar{f}_i \equiv \lim_{t\to\infty} f_i(t) \ .$$

The matrix A can be brought to diagonal form with another square matrix L.

$$A = LDL^{-1} \ ,$$

where D is a diagonal matrix. Therefore

$$A^t = LD^tL^{-1} \ , \qquad (6)$$

which, from Eq. (5), provides a general solution for the distribution function at time t.

The columns of L are the eigenvectors u of A satisfying

$$Au = \lambda u \ .$$

Let us consider a specific example.

For two dimensions the most general form of the transition matrix is the following

$$A = \begin{bmatrix} p_1 & p_2 \\ 1-p_1 & 1-p_2 \end{bmatrix} \ . \qquad (7)$$

The eigenvalues λ are the roots of the equation

$$\det(A-\lambda I) = 0 \ , \qquad (8)$$

where I is the identity matrix, expressing the fact for the equation $(A-\lambda I)u=0$ to admit a nonzero solution u, the matrix $A - \lambda I$ must be singular.

In this case Eq. (8) is quadratic, with the roots,

$$\lambda = \frac{p_1+1-p_2 \pm \sqrt{(p_1+1-p_2)^2 - 4(p_1(1-p_2)-p_2(1-p_1))}}{2}.$$
$$=1,\ p1-p2$$

One can see that eigenvectors (without a particular normalization) are

$$u_1 = \begin{bmatrix} p_2 \\ 1-p_1 \end{bmatrix}.$$
$$u_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Therefore the matrices L and L^{-1} are given by

$$L = \begin{bmatrix} p_2 & 1 \\ 1-p_1 & -1 \end{bmatrix}.$$
$$L^{-1} = \frac{\begin{bmatrix} 1 & 1 \\ 1-p_1 & -p_2 \end{bmatrix}}{p_2+1-p_1}$$

Let us consider an arbitrary initial distribution given by

$$F(0) = \begin{bmatrix} f_1 \\ 1-f_1 \end{bmatrix}$$

and ask what is the distribution at time t. The answer is provided by Eqs. (5) and (6) and is given in this case by

$$F(t) = \frac{\begin{bmatrix} p_2+T(t) \\ 1-p_1-T(t) \end{bmatrix}}{1+p_2-p_1}. \tag{9}$$
$$T(t) = (p_2-p_1)^t\left(p_2 - f_1(1+p_2-p_1)\right)$$

Notice the following properties of Eq. (9). The transient power term decays exponentially with time constant

$$\tau = -\frac{1}{\log(|\lambda_2|)} \tag{10}$$

where $\lambda_2 = p_1 - p_2$ is the second largest eigenvalue (the largest is always 1), which is a long time if $|\lambda_2|$ is close to 1. After this transient period, the distribution becomes a unique steady state $\bar{f}$, independent of the starting distribution. The chain steady state satisfies the detailed balance condition

$$A_{i,j}\bar{f}_j = A_{j,i}\bar{f}_i, \tag{11}$$

which, in words, means that in steady state, the number of transitions from state j to i is balanced by the number of transitions from i to j.

In general, the distribution function satisfies the equation

$$\frac{\Delta f_i}{\Delta t} = \sum_j \left(A_{i,j} f_j - A_{j,i} f_i\right),$$

so that detailed balance implies the steady state solution $f = \bar{f}$.

Now we consider the uniqueness of the steady state solution more carefully.

Imagine that instead of Eq. (7), the transition matrix was given by

$$A = \begin{bmatrix} p_1 & p_2 & & \\ 1-p_1 & 1-p_2 & & \\ & & p_3 & p_4 \\ & & 1-p_3 & 1-p_4 \end{bmatrix}. \tag{12}$$

Clearly in this case there are two steady states, one involving indices 1 and 2 and another involving indices 3 and 4, and the final steady state is not unique. However, in this case the transition matrix is not ergodic, ergodic meaning that every state can be reached by every other state. We see from Eq. (12) that the state with index 3 mixes only with state 4 and not with 1 and 2. The matrix given by Eq. (12) is not ergodic. In general, if the matrix is ergodic, there is a single unique steady state with eigenvalue 1. If not, there will be more than one unity eigenvalue.

The actual problem we will be concerned with is to determine the transition matrix A that gives a specified steady state $\bar{f}$. Then, the average of a function over $\bar{f}$ can be evaluated as a time average of the function of the dynamical variable.

There are many ways to come up with a transition matrix that gives a specified steady state. Such transition matrices are constructed using the detailed balance condition, Eq. (11), together with the requirement that the columns of A contain positive numbers normalized to unity. Detailed balance implies that elements of $A_{i,j}$ reflected through the diagonal (the transposed elements) are in the inverse ratio of the desired steady state distribution function

$$\frac{A_{i,j}}{A_{j,i}} = \frac{\bar{f}_i}{\bar{f}_j}.$$

In fact, the most basic solution of $Af = f$ contains just two nonzero off-diagonal elements in corresponding transposed positions satisfying the detailed balance relation with neither exceeding unity. The diagonal element in both columns is then what is required by normalization, the complement of the corresponding off-diagonal element. Other solutions can be constructed as a probabilistic average of these basic transition matrices using the fact that a probabilistic mixture of solutions is itself a solution.

In our consideration of some explicit solutions, we start with a simple case of a random-walk transition matrix A where, from state j, transitions occur only to one other state $j+1$ or $j-1$.

As a starting point, let the *j*th column of *A* have two nonzero elements given by

$$\begin{aligned} A_{j-1,j} &= \bar{f}_{j-1} \\ A_{j,j} &= * \end{aligned} \quad , \tag{13}$$

where $*$ denotes what is required by column normalization, namely $1-\bar{f}_{j-1}$. One sees immediately that the detailed balance condition is satisfied.

The above remains a solution if the off diagonal matrix elements are multiplied by a positive constant *a* less than 1. This multiplying factor need be constant only with respect to transposed positions within the matrix *A*, and it can be greater than 1 as long as it does not upset the column normalization condition.

The diagonal band can be increased from one adjoining state to *l* = 2, 3 … adjoining states and this form provides a solution when the off diagonal matrix elements are given by Eq. (13) multiplied by *a/l*. One should note that for *l* = 1, even though satisfying the detailed balance requirement, the transition matrix is not ergodic. The transition matrix in that case appears as

$$\begin{array}{ccccc} * & \circ & & & \\ \circ & * & & & \\ & & * & \circ & \\ & & \circ & * & \\ & & & & \cdots \end{array} .$$

The solution given by Eq. (13) has the important drawback that it requires the normalized steady state distribution function. Nonetheless, we will use it to study the effect of the multiplying factor *a*.

Table **1** shows the result of a numerical study using a FORTRAN code available for download as supplementary material from the publisher's web site along with the published article. The discrete space has 100 points and the assumed distribution function is a Gaussian centered on *i* = 50 with a standard deviation of 20. The transition matrix and the 100 eigenvalues and eigenvectors are calculated numerically for a symmetrical random walk where the half width of the random walk step is 1 (*l* = 2) with different values of the constant *a*. The eigenvalues are real and positive having a maximum value of 1 for one eigenvector. That eigenvalue-1 eigenvector, when normalized, returns the assumed steady state. The second largest eigenvalue gives the time to reach steady state from Eq. (10). In Table **1** are shown the times 3τ for three different values of *a*.

Clearly the multiplying factor *a* should be as large as possible without upsetting column normalization in order to have the smallest times to reach steady state.

We can let the quantity *a* be specific to a particular transpose pair of off-diagonal matrix elements of *A* rather than being a constant, and a solution so modified will remain a solution. For the Barker (B) algorithm

$$a_{j-1,j} = \frac{1}{l}\frac{1}{\bar{f}_{j-1}+\bar{f}_j} \ , \tag{14}$$

Table 1. Number of Chain Iterations Required to Reach Steady State (3 τ) with Different Values of the Multiplying Factor a

a	3 t
10	1.05×10^{4}
1	1.05×10^{5}
0.1	1.05×10^{6}

while for the Metropolis, Rosenbluth, Teller algorithm (MRT) it is given by

$$a_{j-1,j} = \frac{1}{l}\frac{1}{\max(\bar{f}_{j-1},\bar{f}_j)} \ . \tag{15}$$

Notice that the MRT factor from Eq. (15) is always greater than the B factor from Eq. (14).

This also corresponds to the relative size of the times to reach steady state given in Table **2**.

Table 2. Number of Chain Iterations Required to Reach Steady State (3 τ) for the B and MRT Algorithms Under the Same Conditions as Table 1

Algorithm	3 τ
B	4.4×10^{3}
MRT	2.3×10^{3}

In this sense the MRT algorithm can be viewed as the maximally efficient member of the family of exact solutions of detailed balance of this type, as has been shown by Peskun [9]. Notice that these algorithms do not require the normalized distribution function.

The new algorithm gives up exact solution of detailed balance in exchange for more rapid convergence and compatibility with parallel processing.

For the new algorithm the columns of the transition matrix are given by

$$A_{i,j} \propto \bar{f}_i \ , \tag{16}$$

including the diagonal element, with the proportionality constant chosen to provide column-by-column normalization. This solution provides a proper Markov Chain transition matrix, but the steady state is now only approximately given by the desired distribution function. When the size of the diagonal band (the size of the random walk step) is many times larger than the size of the support region of the desired steady-state distribution function, we would expect the approximation to be quite good.

This is illustrated by the following numerical example shown in Fig. (**1**). The discrete space has 100 points and the assumed distribution function is a Gaussian centered on *i* = 50 with a standard deviation of 4. Two cases are considered, symmetrical random walks where the half width of the random walk step is 1 (l = 2) and 10 (l = 20). The transition matrix and the 100 eigenvalues and eigenvectors for the three algorithms are calculated numerically. The eigenvalue-1 eigenvector gives the steady state, and this eigenvector is compared with the desired steady state distribution function. For the MRT and B algorithms the agreement is exact, as it must be. For the new algorithm, when the random walk step size is small, the actual steady state is significantly narrower than the assumed distribution. On the other hand, when the random walk step size is larger, the agreement is quite good.

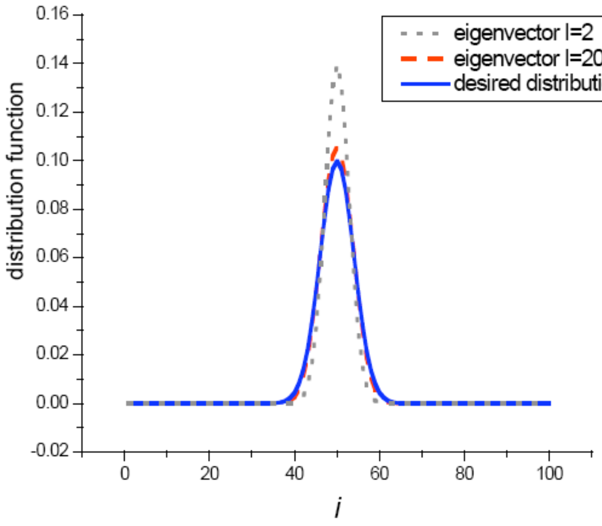

Fig. (1). Steady state distribution functions obtained using the new algorithm.

It is of interest to note that for l = 1, the new algorithm is exactly the same as the B solution and is an exact solution of the detailed balance condition, even though for l = 1 the transition matrix is not ergodic.

When investigating these algorithms numerically, one finds that for a wide diagonal band, the rate of convergence for the new algorithm is much better than for the other two algorithms. This is illustrated in Fig. (**2**), which shows the eigenvalue spectrum for the three algorithms in two cases, narrow and broad random walk step.

Recall that the transient states decay like λ^t, where t is the chain iteration number and λ is the eigenvalue. The initial state is some mixture of the 100 eigenvectors. So, we desire small values of the eigenvalues in order for the system to quickly reach a steady state.

One can also visualize the effect by considering the exploration of a large parameter space where the support region of the distribution function occupies only a small fraction. Using our numerical example, assume the chain is at point j = 50 and the distribution function is peaked at i = 75. The new state after one step is given by $A(i,50)$ and is as shown in Fig. (**3**). The random-walk step size of l = 70 is such that the random walk region includes the distribution peak.

Using the new algorithm, the distribution is very close to the desired steady state distribution in only 1 step!

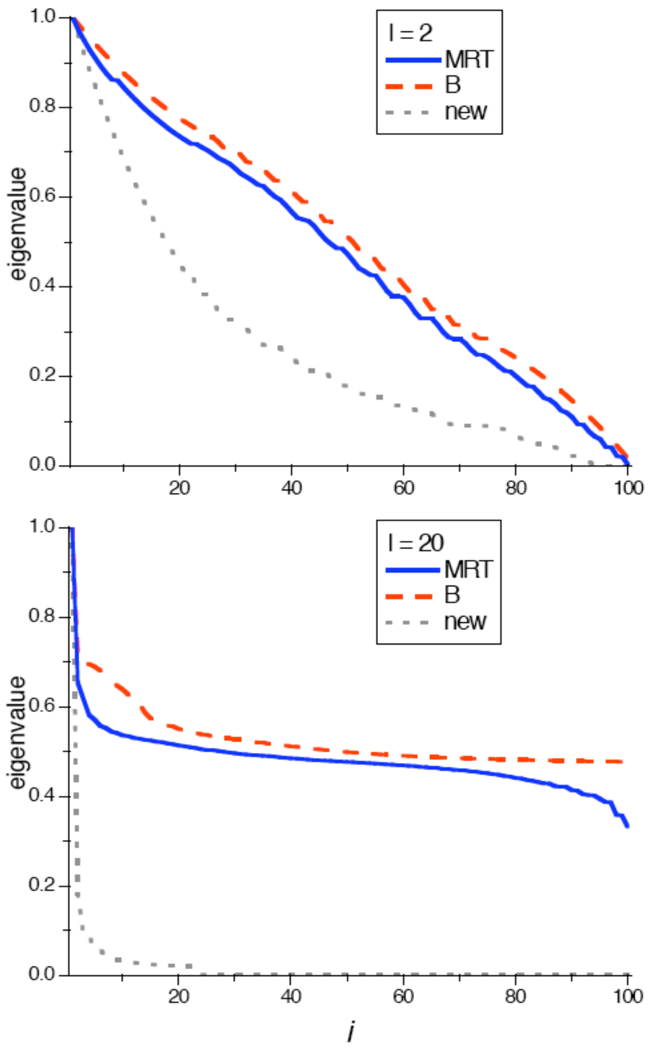

Fig. (2). Eigenvalue spectrum for the three different algorithms and different random walk step size. For a large random walk step, the new algorithm converges much more rapidly than the others.

In practice the transition probabilities are not used directly but represent the ensemble averages of the underlying dynamics of the system. How do we stochastically simulate this dynamics in such a way as to reproduce the desired transition matrix?

Assume that the system is in state j. Let us assume that the new state i is probabilistically generated by first generating a candidate point i with probability $q_{i,j}$ in the diagonal band excluding the center point and then accepting this new state with probability $\alpha_{i,j}$. If the new state is not accepted, the system remains in state j. The ensemble average of this process is

$$A_{i,j} = q_{i,j}\alpha_{i,j} \quad , \tag{17}$$

for off-diagonal matrix elements. The diagonal elements are determined by column normalization. Notice that the columns of q are probability distributions over the diagonal band of width l excluding the actual diagonal, while the columns of A are probability distributions over all l + 1 elements in the diagonal band.

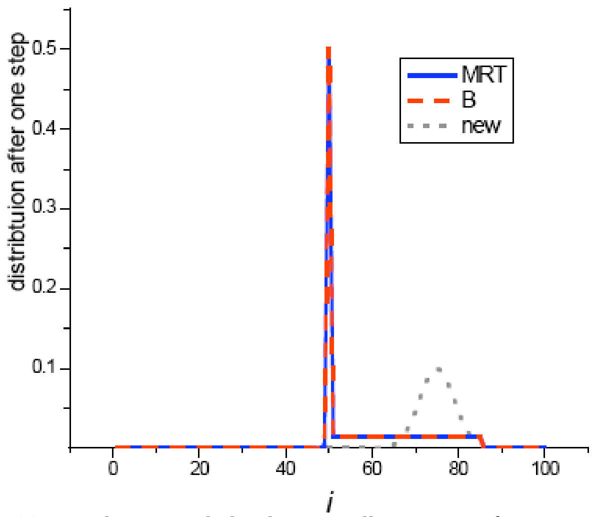

Fig. (3). Distribution, with the chain initially at i = 50, after one step.

So far we have assumed a uniform random walk candidate distribution with

$$q_{i,j} = \frac{1}{l} \quad .$$

The acceptance probability follows from the transition matrix. From Eq. (15) the acceptance probability for the MRT algorithm in this case can be written as

$$\alpha_{i,j} = \frac{\dfrac{\bar{f}_i}{q_{i,j}}}{\max\left(\dfrac{\bar{f}_i}{q_{i,j}}, \dfrac{\bar{f}_j}{q_{j,i}}\right)} \quad . \tag{18}$$

Notice that the acceptance probability is less than or equal to 1, which means that column normalization of *A* goes through, because the column-by-column summation of off-diagonal terms of *A* will be less than 1.

Equation (18) provides a more general form of the MRT algorithm [8].

We can now reconsider the new algorithm. Let us assume a candidate distribution given, within the template of nonzero elements in the diagonal band, by

$$q_{i,j} = \frac{\bar{f}_i}{\sum_{j' \neq j} \bar{f}_{j'}} \quad , \tag{19}$$

where the summation is over *l* indices around, but excluding *j*. In this case, from Eqs. (17) and (18),

$$A_{i,j} = \frac{\bar{f}_i}{\max\left(\sum_{j' \neq j} \bar{f}_{j'}, \sum_{j' \neq i} \bar{f}_{j'}\right)} \quad , \tag{20}$$

instead of

$$A_{i,j} = \frac{\bar{f}_i}{\bar{f}_j + \sum_{j' \neq j} \bar{f}_{j'}} \quad , \tag{21}$$

for the new algorithm. Equation (20) is approximately the same as Eq. (21) in the limit where the diagonal band is large, in which case both summations in Eq. (20) are approximately the same and much larger than the diagonal term included with the summation in the denominator of Eq. (21). The algorithm given by Eq. (20) corrects the detailed balance deficiency of the new algorithm, but numerical studies show that the convergence of this algorithm can be poor.

The simulation of the dynamics consists in first generating a candidate point, calculating the acceptance probability for this candidate, and, if the candidate is accepted, moving to the new point. If the transition probability just involves the desired distribution function evaluated at the points *j* and *i*, this process requires one new evaluation of the desired distribution function at the candidate point *i*.

From the standpoint of parallel processing we would like to be able to advantageously use more than one evaluation of the desired distribution function at each chain iteration, because these calculations, which usually involve the bulk of the computer time, could then be performed in parallel. To do this, the *l* probabilities $A_{i,j}$ for all the possible new points can be calculated in parallel (in the continuous variable case, a new distribution function evaluation is needed for each off-diagonal point), and the new chain position is then probabilistically generated from this discrete probability distribution. This parallelization method works for any of the algorithms, but as Appendix **1** demonstrates, is only advantageous for the new algorithm. This is also illustrated in Table 3, which shows the times required to reach a steady state from Eq. (10) for the sample problem we have been considering (assumed steady state distribution is a Gaussian centered at *i* = 50 with standard deviation 4).

Table 3. Number of Chain Iterations Required to Reach Steady State (3 τ) with Different Numbers of Multiple Candidates (l) for the Three Algorithms

Algorithm	*l*	3 *τ*
MRT	2	106
B		193.5
MRT	20	7.1
B		9.2
New		1.8
Eq. (20)		5×10^9

Fig. (**4**) shows the distribution after 7 iterations with an initial state *i* =1 for the new algorithm, the MRT algorithm, and the algorithm defined by Eq. (20). As before, *l* = 20 and the assumed steady state distribution is a Gaussian centered at *i* = 50 with standard deviation 4.

With the intial state concentrated at *i* = 1, the new algorithm moves the distribution towards the steady state as much as possible given the size of the diagonal band, while Eq. (20) leaves it almost unchanged, even more so than for the MRT algorithm, because in the tail of the distribution the second summation term dominates in the denominator, and all the off-diagonal transition probabilities are very small.

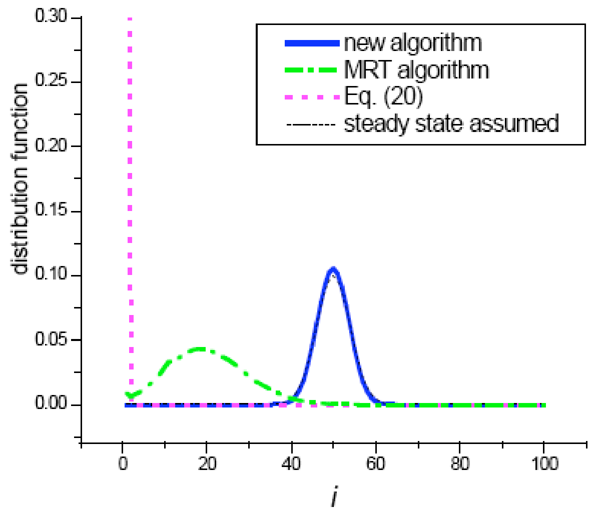

Fig. (4). Distribution, with the chain initially at i = 1, after 7 steps.

4. DISCUSSION AND CONCLUSIONS

In practice, often the issue of overriding importance is the computation time. The column ($3\ \tau$) in Table **3** is the number of chain iterations required to reach a steady state. If parallel processing is used and the evaluations of $\bar{f}$ for multiple candidates are done in parallel, this number is a measure of the computation time. The last column in Table **1** is a measure of the total number of evaluations of $\bar{f}$ required reaching a steady state and thus the computer time.

One should note with caution that these results confuse the effect of the number of multiple candidates with the effect of a larger random walk step size.

In Appendix **1** the continuous version of the new algorithm is compared with the MRT algorithm. The striking good performance of the new algorithm makes this an algorithm of great interest.

ACKNOWLEDGEMENT

The author is grateful to James Gubernatis for helpful discussions of this work.

APPENDIX 1—EXAMPLE OF USE OF CONTINUOUS VARIABLE MCMC ALGORITHMS

The function f is chosen to be a 6-dimensional Gaussian centered at some point ($\theta = 0.33$) for each of the 6 dimensions. The standard deviation is 1×10^{-6} for each of the 6 dimensions. The object is to evaluate the first and second moments of f in order to obtain the average and standard deviation. This would be a very challenging problem for other methods of numerical integration.

The algorithms use a random walk step size $\Delta = 1 \times 10^{-5}$ however half the time candidates are chosen from the entire space ($\Delta = 1$). At each iteration, only one coordinate is randomly selected for movement. Two runs are made with the chain started at $\theta = 0$ and $\theta = 1$ and with different random number seeds.

Fig. (**5**) shows the history of the log of the likelihood function (the "energy") as a function of iteration for different MCMC algorithms.

In Table **4** are shown the average and standard deviation calculated using 150 samples, one every 10 iterations, after equilibration. These are obtained by making a certain length run with an initial fraction ("burnfrac") ignored in the moment calculation. Only the final 1500 iterations are used for the moment calculation. As seen from Table **4**, the correct answers are being reproduced.

For the MRT algorithm, using more candidates at each iteration does not result in any improvement. As seen from these results, with the new algorithm more candidates give an improvement in performance, with 8 candidates allowing about 5 times fewer iterations.

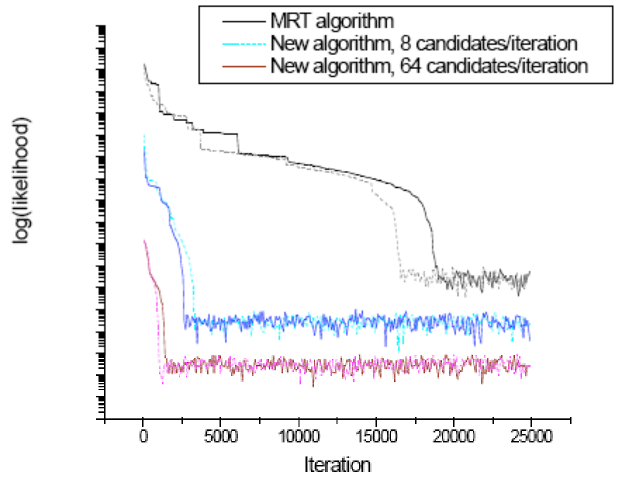

Fig. (5). Log of likelihood function versus iteration (with shifted vertical scale). Iteration number can be thought of as a measure of computation time as long as the energy calculations can be done in parallel.

Table 4. Average and Standard Deviation of a Very Narrow 6-Dimension Gaussian Calculated Using different MCMC Algorithms. The Quantity *l* is the Number of Candidates Considered at Each Iteration

Algorithm	Iterations	Burnfrac	Idim	Average	SD (1.e-6)
MRT(*l* = 1)	21500	0.93	1	0.33	0.940
			2	0.33	1.124
			3	0.33	0.929
			4	0.33	0.832
			5	0.33	1.021
			6	0.33	0.974
New(*l* = 8)	4100	0.634	1	0.33	0.858
			2	0.33	0.930
			3	0.33	0.936
			4	0.33	0.882
			5	0.33	1.084
			6	0.33	0.905

SUPPLEMENTARY MATERIALS

Supplementary material is available on the publishers web site along with the published article.

REFERENCES

[1] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculation by fast computing machines. J Chem Phys 1953; 21 (6):1087-92.

[2] Barker AA. Monte Carlo calculations of the radial distribution functions for proton-electron plasma. Aust J Phys 1965; 18: 119-33.

[3] Miller G. Markov Chain Monte Carlo calculations allowing parallel processing using a variant of the metropolis algorithm**.** Open Numer Meth 2010; 2: 12-7.

[4] Rubenstein BM, Gubernatis JE, Doll JD. Comparative Monte Carlo efficiency by Monte Carlo analysis. Phys Rev E 2010; 82: 036701.

[5] Wilkinson D. Parallel Bayesian computation in Kontoghiorghes EJ, ed. Handbook of parallel computing and statistics. Chapman and Hall 2006; chapter 16: 477-508.

[6] Whiley M, Wilson SP. Parallel algorithms for Markov chain Monte Carlo in latent spatial Gaussian models. Stat Comput 2004; 14 (3): 171-9.

[7] Yan J, Cowles MK, Wang S, Armstrong MP. Parallelizing MCMC for Bayesian spatiotemporal geostatistical models. Stat Comput 2007; 17 (4): 323-35.

[8] Hastings WK. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 1970; 57 (1): 97-109.

[9] Peskun PH. Optimum Monte-Carlo sampling using Markov chains. Biometrika 1973; 60 (3): 607-12.

Received: March 11, 2011 Revised: April 14, 2011 Accepted: April 27, 2011