Journal of Computational Physics **172,** 827–840 (2001)
doi:10.1006/jcph.2001.6860, available online at http://www.idealibrary.com on IDEAL®

Multipoint Metropolis Method with Application to Hybrid Monte Carlo

Zhaohui S. Qin and Jun S. Liu[1]

Department of Statistics, Harvard University, 1 Oxford Street, Cambridge, Massachusetts 02138
E-mail: qin@stat.harvard.edu; jliu@stat.harvard.edu

Received October 11, 2000; revised June 25, 2001

We propose the multipoint Metropolis algorithm as an extension of the orientational-bias Monte Carlo of Frenkel and Smit. A ratio statistics similar to that in the Metropolis algorithm is introduced to maintain the detailed balance. The multipoint idea can be applied to improve the efficiency of a general Markov chain-based Monte Carlo algorithm. To illustrate, we describe two variations of the idea—the random-grid Metropolis and the multipoint Hybrid Monte Carlo—and apply them to a number of examples. © 2001 Academic Press

Key Words: Boltzmann distribution; Gibbs sampler; Hamiltonian; heat-bath algorithm; leap-frog; mixture Gaussian; uncoupled oscillator.

1. INTRODUCTION

Computer simulation and optimization for molecular structures and dynamics have been of great interests to chemists, physicists, and biologists. One of the central tools used for these endeavors is Monte Carlo. Recently, statisticians began to appreciate the importance of Monte Carlo methods and have extended many Monte Carlo techniques developed by physicists and structural chemists to solve a broader array of problems, e.g., those in Bayesian inference, artificial intelligence, genetics, computational biology, and others. A major part of these computational problems can be summarized abstractly into the following mathematical setting: A system is parameterized by a vector $\mathbf{x}$ and characterized by a probability distribution $\pi(\mathbf{x})$, which is known up to a normalizing constant (i.e., function $q(\mathbf{x}) = Z\pi(\mathbf{x})$ is known, but the constant Z, often called the partition function, is unknown). It is of interest to estimate the mathematical expectation of a given function $h(\cdot)$, i.e., the value of

$$\langle h \rangle = \int h(\mathbf{x})\pi(\mathbf{x})\, d\mathbf{x}.$$

[1] Corresponding author. Support in part by the National Science Foundation under grants DMS-9803649 and DMS-0094613.

If random samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ can be drawn from $\pi(\mathbf{x})$, one can approximate the above integral by the average of the $h(\mathbf{x}_i)$.

Among all the methods that enable one to draw random samples from an arbitrarily given distribution, those based on Markov chain theory are perhaps most widely used. The construction proposed by Metropolis *et al.* [13] and modified by Hastings [7], often referred to as the Metropolis–Hastings (M–H) algorithm, is the most fundamental building block for all Markov chain Monte Carlo (MCMC) algorithms. The M–H algorithm can be implemented as follows:

- Suppose at the t-th iteration we have a sample $\mathbf{x}_t$. At the $(t+1)$st iteration, a "perturbation" $\mathbf{y}$ of the current state is proposed. Operationally this can be achieved by drawing from a Markov transition function $T(\mathbf{y} \mid \mathbf{x}_t)$ (also called the proposal function).
- We let $\mathbf{x}_{t+1} = \mathbf{y}$ with probability

$$r = \min\left\{1, \frac{\pi(\mathbf{y})T(\mathbf{x}_t \mid \mathbf{y})}{\pi(\mathbf{x}_t)T(\mathbf{y} \mid \mathbf{x}_t)}\right\},$$

and let $\mathbf{x}_{t+1} = \mathbf{x}_t$ with probability $1 - r$. We call r the M–H ratio.

Theoretically, the M–H algorithm can be applied to an arbitrary unnormalized distribution regardless of its dimension. However, in most applications, the Markov chain generated by the M–H algorithm can be trapped indefinitely in a local mode so that the equilibration time of the sampler can be unacceptably long. One remedy is to enable the sampler to take larger steps. Under the M–H framework, this idea amounts to let the proposal function $T(\cdot \mid \cdot)$ to make more drastic perturbations on the current state. Although this may allow the sampler to escape from certain local modes, unavoidably a more violent change in $\mathbf{x}_t$ is much less likely to be accepted, which also hurts the efficiency of the algorithm. The multipoint method introduced in this article aims at alleviating this conflict of interest.

One of the most effective means for improving equilibration time of a MCMC sampler is parallel tempering [3, 5], in which one runs in parallel several independent Markov chains with different temperature parameters and proposes temperature swapping periodically. However, in cases where one is interested only in the simulation at a fixed temperature, it is not clear whether the improvement justifies the additional complexity and computation of parallel tempering. The multipoint method, on the other hand, is a simple generalization of the Metropolis rule and is perhaps more suitable for a moderate system. Additionally, one can combine the multipoint idea with parallel tempering to design more innovative algorithms such as evolutionary Monte Carlo [9].

The multipoint method is closely related to the "multiple-try Metropolis" (MTM) proposed in [10], both of which can be viewed as generalizations of the "orientational-bias Monte Carlo" (OBMC) method described in Frenkel and Smit [4]. In both MTM and OBMC, one is allowed to propose multiple independent trial samples from a proposal function $T(\cdot \mid \cdot)$. As a consequence, one can afford to employ a proposal transition T that covers a relatively larger region in the state space. Our new method further allows one to propose multiple *dependent* trial points and to reuse some old proposals. These extra features simplify the computation in MTM and extend the applicability of the multipoint idea. In particular, we show that the method can be used to improve a hybrid Monte Carlo (HMC) algorithm.

2. MULTIPOINT METROPOLIS METHOD

In the M–H algorithm, a new trial state is generated from the proposal transition function, and then a decision is made on whether to accept the new state based on a likelihood ratio. In the multipoint algorithm, we allow the algorithm to make multiple proposals and then choose a good one among them.

Suppose the current state is $\mathbf{x}$, we propose n candidates by sampling $\mathbf{y}_1 \sim T_1(\cdot \mid \mathbf{x})$, $\mathbf{y}_2 \sim T_2(\cdot \mid \mathbf{x}, \mathbf{y}_1), \ldots,$ and $\mathbf{y}_n \sim T_n(\cdot \mid \mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_{n-1})$. For briefness, we let $\mathbf{y}_{[1:j]}=(\mathbf{y}_1, \ldots, \mathbf{y}_j)$ and let $\mathbf{y}_{[j:1]}$ denote the vector with the reverse order, i.e., $\mathbf{y}_{[j:1]} = (\mathbf{y}_j, \mathbf{y}_{j-1}, \ldots, \mathbf{y}_1)$. We suppress all the subscripts for T. That is, the joint sampling distribution of $\mathbf{y}_{[1:j]}$ is written as

$$T\left(\mathbf{y}_{[1:j]} \,\middle|\, \mathbf{x}\right) \equiv T(\mathbf{y}_1 \mid \mathbf{x}) \times \cdots \times T\left(\mathbf{y}_j \,\middle|\, \mathbf{x}, \mathbf{y}_{[1:(j-1)]}\right), \tag{1}$$

with the understanding that the $T(\mathbf{y}_j \mid \mathbf{x}, \mathbf{y}_{[1:j-1]})$ are in fact different functions for different j. Finally, we define a weight function

$$\omega_j\left(\mathbf{x}, \mathbf{y}_{[1:j]}\right) = \pi(\mathbf{x}) T\left(\mathbf{y}_{[1:j]} \,\middle|\, \mathbf{x}\right) \lambda_j\left(\mathbf{x}, \mathbf{y}_{[1:j]}\right).$$

It is seen that $\pi(\mathbf{x}) T(\mathbf{y}_{[1:j]} \mid \mathbf{x})$ is a joint distribution of $\mathbf{x}$ and $\mathbf{y}_{[1:j]}$ (but not the stationary one). Here, function λ_j can be any bounded, positive, and *sequentially symmetric* function, where "sequentially symmetric" means that

$$\lambda_j\left(\mathbf{x}, \mathbf{y}_{[1:j]}\right) = \lambda_j\left(\mathbf{y}_{[j:1]}, \mathbf{x}\right).$$

Again, in the remaining part of the article, we suppress all the subscripts for functions λ_j and ω_j. The details of the multipoint algorithm follow.

Multipoint Metropolis Algorithm

- Draw n trial points, $\mathbf{y}_1, \ldots, \mathbf{y}_n$ according to the joint proposal distribution T, as defined in (1); compute $\omega(\mathbf{y}_{[j:1]}, \mathbf{x})$ for $j = 1, \ldots, n$.
- Select $\mathbf{y}$ among the trial set $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ with probability proportional to $\omega(\mathbf{y}_{[j:1]}, \mathbf{x})$, $j = 1, \ldots, n$. Suppose $\mathbf{y} = \mathbf{y}_k$ has been selected.
- Construct a "reference set" $\{\mathbf{x}^*_{k+1}, \ldots, \mathbf{x}^*_n\}$ by sampling $\mathbf{x}^*_{j+1}$ from

$$T\left(\cdot \,\middle|\, \mathbf{y}_{[k:1]}, \mathbf{x}, \mathbf{x}^*_{[k+1:j]}\right),$$

for $j = k, \ldots, n-1$. For notational simplicity, we name $\mathbf{x}^*_j = \mathbf{y}_{k-j}$ for $j = 1, \ldots, k-1$ and $\mathbf{x}^*_k = \mathbf{x}$. Thus, the reference path $\{\mathbf{x}^*_1, \ldots, \mathbf{x}^*_n\}$ is like a reversal of the forward path $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ and the two are "annealed" at the middle $k+1$ positions (like a double stranded DNA sequence with two "sticky" ends).
- Accept $\mathbf{y}$ with probability

$$r_{\mathrm{mp}} = \min\left\{1, \sum_{j=1}^{n} \omega\left(\mathbf{y}_{[j:1]}, \mathbf{x}\right) \Big/ \sum_{j=1}^{n} \omega\left(\mathbf{x}^*_{[j:1]}, \mathbf{y}\right)\right\}$$

and reject it with probability $1 - r_{\mathrm{mp}}$. The quantity r_{mp} is called the multipoint Metropolis ratio.

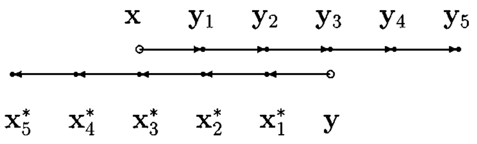
$\mathbf{x}$ $\mathbf{y}_1$ $\mathbf{y}_2$ $\mathbf{y}_3$ $\mathbf{y}_4$ $\mathbf{y}_5$
$\mathbf{x}_5^*$ $\mathbf{x}_4^*$ $\mathbf{x}_3^*$ $\mathbf{x}_2^*$ $\mathbf{x}_1^*$ $\mathbf{y}$

FIG. 1. Graphical illustration of a random-grid Metropolis step. In this case, $n = 5$, $k = 3$. $\mathbf{y} = \mathbf{y}_3$ was selected as candidate. Then we name $\mathbf{x}_1^* = \mathbf{y}_2$, $\mathbf{x}_2^* = \mathbf{y}_1$, $\mathbf{x}_3^* = \mathbf{x}$.

Figure 1 gives a cartoon illustration of the algorithm. Compared with OBMC and MTM, a new feature of our algorithm is that the middle $k+1$ points are reused in the reference set. The idea of reusing old configurations in computing Metropolis-like ratios has been proposed earlier in [2] in a different setting.

The simplest choice of the symmetric function is $\lambda(\mathbf{x}, \mathbf{y}_{[1:j]}) \equiv 1$. Intuitively, the detailed balance condition is maintained because the backward "reference path" compensates the forward path, and the favorable choice of $\mathbf{y}$ is counterbalanced by requiring that the reference path $\mathbf{x}_j^*$ has to pass through the starting point $\mathbf{x}$. In the Appendix, we give a rigorous proof showing that the above multipoint method satisfies the "super" detailed balance condition.

When the proposal transition is sequentially symmetric, i.e.,

$$T\left(\mathbf{y}_{[1:j+1]} \,\middle|\, \mathbf{y}_0\right) = T\left(\mathbf{y}_{[j:0]} \,\middle|\, \mathbf{y}_{j+1}\right),$$

we can choose $\lambda(\mathbf{x}, \mathbf{y}_{[1:j]}) = 1/T(\mathbf{y}_{[1:j]} \mid \mathbf{x})$, then the algorithm can be simplified as $\omega(\mathbf{y}_{[j:1]}, \mathbf{x}) \propto \pi(y_j)$, a form similar to the one in [4]. That is, we can select $\mathbf{y}$ among the trial set $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ with probability proportional to $\pi(\mathbf{y}_j)$, $j = 1, \dots, n$ and then accept $\mathbf{y}$ with probability

$$\mathrm{r_{mp}} = \min\left\{1, \sum_{j=1}^{n} \pi(\mathbf{y}_j) \middle/ \sum_{j=1}^{n} \pi(\mathbf{x}_j^*)\right\}.$$

Note that a sequentially symmetric proposal can be derived by composing a number of symmetric Markov transition steps, i.e.,

$$T\left(\mathbf{y}_{[1:n]} \,\middle|\, \mathbf{x}\right) = K_1(\mathbf{y}_1 \mid \mathbf{x}) K_2(\mathbf{y}_2 \mid \mathbf{y}_1) \cdots K_n(\mathbf{y}_n \mid \mathbf{y}_{n-1}),$$

where $K_j(\mathbf{y} \mid \mathbf{x}) = K_j(\mathbf{x} \mid \mathbf{y})$ and is a conditional probability function.

The multipoint method is quite general. At one extreme, it is entirely possible that the transition proposal is a semi-deterministic function (see Sections 3 and 4) in which all the generated variables are correlated. An especially useful scenario is that the multiple candidates are generated by a Markov chain. For example, one may have a favorite transition function $K(\mathbf{y} \mid \mathbf{x})$ which is a "cheap" approximation of the desirable transition kernel $A(\mathbf{y} \mid \mathbf{x})$ (whose equilibrium distribution is π). Then one can generate a few candidates from composing K and use the multipoint method to select among them.

At the other extreme, all the multiple trial points can be generated independently, conditional on $\mathbf{x}$ and possibly another random variable, as in OBMC [4] and MTM [10]. For example, the proposal set $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ can be independent and uniform draws from the radius-γ sphere centered at $\mathbf{x}$, where γ is generated from a distribution independent of $\mathbf{x}$. Then, after selecting the candidate $\mathbf{y}_j$, one cannot reuse the old draws. But one can still implement the multipoint method by generating $n-1$ independent and uniformly distributed samples from the sphere centered at $\mathbf{y}_j$ with the same radius γ. Mathematically, the generation of $\mathbf{y}_{[1:n]}$ in the above procedure can be described as

$$T\left(\mathbf{y}_{[1:n]} \,\middle|\, \mathbf{x}, \gamma\right) = T(\mathbf{y}_1 \mid \mathbf{x}, \gamma) \cdots T(\mathbf{y}_n \mid \mathbf{x}, \gamma),$$

and the acceptance–rejection decision is made conditional on γ. The proposal process described by (1) can also be generalized to allow for an additional "ancillary" variable γ, i.e., $\mathbf{y}_{[1:n]}$ can be generated from

$$T\left(\mathbf{y}_{[1:n]} \,\middle|\, \mathbf{x}, \gamma\right) = T(\mathbf{y}_1 \mid \mathbf{x}, \gamma) \cdots T\left(\mathbf{y}_n \,\middle|\, \mathbf{x}, \mathbf{y}_{[1:(n-1)]}, \gamma\right),$$

and the multipoint ratio be computed conditional on γ. The proof of its correctness is similar to that in Section A.2.

Weighted Multipoint Metropolis

Generating more candidates to choose from does not necessarily mean that we will end up "walking farther" at each iteration. Intuitively, those trial points that are close to the starting position $\mathbf{x}$ tend to have a higher acceptance probability than those points that are farther away from $\mathbf{x}$. In order to force the chain to explore a greater area, we can add weights to the generated candidates. Those candidates farther from the starting point $\mathbf{x}$ receive greater weight than those that are closer to $\mathbf{x}$. That is, we can define

$$\tilde{\omega}\left(\mathbf{x}, \mathbf{y}_{[1:j]}\right) = u_j \omega\left(\mathbf{x}, \mathbf{y}_{[1:j]}\right), \tag{2}$$

and use $\tilde{\omega}$ in the place of ω in the multipoint algorithm previously defined. Typical choices can be $u_j = j^\alpha$ or $u_j = \log j$, both giving increasing preference to points "farther" away. In fact, this flexibility has been reflected in the original multipoint algorithm for that we can choose λ_j freely. Expression (2) makes it explicit that we put artificially unequal weights to the multiple candidates. Our experiences show that this weighting strategy is very useful for improving a HMC method (Section 4).

3. RANDOM-GRID METROPOLIS

As a demonstration of how one can use the multipoint strategy, we describe a method, the random-grid Metropolis, which is useful when the state space is a Euclidean space.

Random-Grid Metropolis

- Sample a distance r and a direction (unit vector) $\vec{\mathbf{e}}$ from their proposal distributions, respectively. Construct the n candidates as

$$\mathbf{y}_j = \mathbf{x} + j \cdot r \cdot \vec{\mathbf{e}}, \quad j = 1, \ldots, n.$$

- Select $\mathbf{y}$ among the trial set $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ with probability proportional to $\pi(\mathbf{y}_j)$, $j = 1, \ldots, n$. Suppose the final selection is $\mathbf{y} = \mathbf{y}_k$. Then we let

$$\mathbf{x}_j^* = \mathbf{y} - j \cdot r \cdot \vec{\mathbf{e}} \quad \text{for } j = 1, \ldots, n.$$

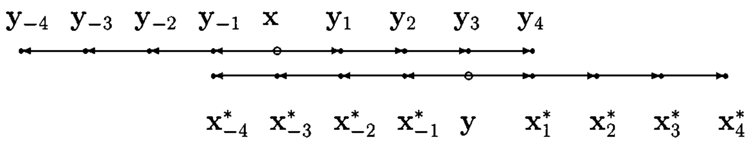

FIG. 2. Illustration of two-sided random-grid Metropolis chain. In this case, $n = 4$, $k = 3$. $\mathbf{y} = \mathbf{y}_3$ was selected as candidate. Then we name $\mathbf{x}^*_1 = \mathbf{y}_4$, $\mathbf{x}^*_{-1} = \mathbf{y}_2$, $\mathbf{x}^*_{-2} = \mathbf{y}_1$, $\mathbf{x}^*_{-3} = \mathbf{x}$, $\mathbf{x}^*_{-4} = \mathbf{y}_{-1}$.

Therefore, $\mathbf{x}^*_j$ is equal to $\mathbf{y}_{k-j}$ for $j < k$ and to $\mathbf{x} - (k - j) \cdot r \cdot \vec{\mathbf{e}}$ for $j \geq k$.

- Accept $\mathbf{y}$ with probability

$$r_{rg} = \min\left\{1, \sum_{j=1}^{n} \pi(\mathbf{y}_j) \bigg/ \sum_{j=1}^{n} \pi(\mathbf{x}^*_j)\right\},$$

and reject it with probability $1 - r_{rg}$.

Figure 1 shows how a random-grid step is implemented. As a small variation of the random-grid method, we can also take candidates from both sides of the random direction, which can sometimes be more efficient. More precisely, after generating r and $\vec{\mathbf{e}}$, we can construct $2n$ candidates as

$$\begin{aligned} \mathbf{y}_{-j} &= \mathbf{x} - j \cdot r \cdot \vec{\mathbf{e}}, \qquad j = 1, \ldots, n, \\ \mathbf{y}_j &= \mathbf{x} + j \cdot r \cdot \vec{\mathbf{e}}, \qquad j = 1, \ldots, n. \end{aligned}$$

Then, after selecting a candidate $\mathbf{y}$ from the candidate set, we construct the reference set as

$$\mathbf{x}^*_j = \mathbf{y} + j \cdot r \cdot \vec{\mathbf{e}},$$

for $j = \pm 1, \ldots, \pm n$. A graphical illustration is given in Fig. 2.

The proposal function for r can be any proper density function supported on the interval $[0, A]$ (where A can be infinite). From our experience, a uniform distribution, an exponential distribution, or another form of the Gamma distribution are convenient choices. In contrast, the most appropriate choice of the proposal function for $\vec{\mathbf{e}}$ is perhaps the uniform distribution because in most applications we do not have a strong reason to prefer one direction to another.

Since only the step size and the direction are random, it is clear that the resulting candidates from the random-grid method are not mutually independent. For a better exploration of local landscape of the target distribution, one may want to insert a few standard metropolis steps between multipoint iterations.

4. MULTIPOINT HYBRID MONTE CARLO

4.1. Basic Hybrid Monte Carlo Algorithm

Hybrid Monte Carlo (HMC) was first proposed in [1] to deal with numerical simulation problems in lattice field theory. Unlike standard molecular dynamics algorithms, the goal in HMC is to sample from the Boltzmann distribution

$$\pi(\mathbf{q}) \propto \exp(-E(\mathbf{q})).$$

Here, the N-dimensional vector $\mathbf{q}$ describes the system's (positional) configuration, and $E(\mathbf{q})$ is the potential energy function.

The HMC introduces a fictitious N-dimensional momentum vector $\mathbf{p}$ and the Hamiltonian $H(\mathbf{p}, \mathbf{q}) = E(\mathbf{q}) + \frac{1}{2}|\mathbf{p}|^2$. If we can sample $(\mathbf{p}, \mathbf{q})$ jointly from $\pi(\mathbf{p}, \mathbf{q}) \propto \exp(-H(\mathbf{p}, \mathbf{q}))$, then the resulting $\mathbf{q}$ follows the desired Boltzmann distribution. Because a dynamical move of the system based on the Hamiltonian equations,

$$\frac{d\mathbf{q}}{d\tau} = \frac{\partial H}{\partial \mathbf{p}} = \mathbf{p}, \quad \text{and} \quad \frac{d\mathbf{p}}{d\tau} = \frac{\partial H}{\partial \mathbf{q}} = -\nabla E(\mathbf{q}),$$

is time-reversible, volume-preserving, and maintains the constant total energy, it leaves the joint distribution $\pi(\mathbf{p}, \mathbf{q})$ invariant. Ideally, one can simulate $(\mathbf{p}, \mathbf{q})$ by (a) drawing $\mathbf{p}$ from its marginal distribution (which is Gaussian), and (b) evolving the system according to the Hamiltonian equations for a long period of time.

A popular method for simulating the Hamiltonian dynamics is the *leap-frog algorithm*, in which each leap-frog step is time-reversible and volume-preserving [1]. Since each leap-frog step is a discretization of the continuous-time Hamiltonian equations, it does not maintain a constant total energy, resulting in "nonphysical" moves. The following HMC method uses the Metropolis rejection rule to correct for the biases resulting from discretization.

- Generate the momentum vector $\mathbf{p} = (p_1, \ldots, p_N)$ from the standard Gaussian distribution, i.e., draw

$$p_j \sim N(0, 1), \quad \text{for } j = 1, \ldots, N.$$

- A proposal, $(\mathbf{p}', \mathbf{q}')$, is generated from applying n iterations of the leap-frog algorithm to the current state $(\mathbf{p}, \mathbf{q})$.
- Accept $(\mathbf{p}', \mathbf{q}')$ with $r = \min[1, \exp(-\Delta H)]$, where $\Delta H = H(\mathbf{p}', \mathbf{q}') - H(\mathbf{p}, \mathbf{q})$.

Note that this algorithm samples $\mathbf{p}$ and $\mathbf{q}$ jointly although we are only interested in $\mathbf{q}$. Vector $\mathbf{p}$ can be viewed as an auxiliary variable.

4.2. Using Multipoint Rule in Hybrid Monte Carlo

As described in the previous section, there are n dynamical steps implicated by a deterministic scheme, e.g., a leap-frog algorithm, between any two stochastic updates of a HMC algorithm. The multipoint technique can be directly applied here. Since the n dynamical steps are "time-reversible," it is easy to generate a "reverse" path as in the general multipoint method. Since the number of dynamical steps (n) may be a large number, treating these n candidates equally as in the standard multipoint method may not be a good idea. Additionally, computing all the ω functions can be a drag to the overall computational efficiency. A simple modification we apply here is to consider only the last m of these n candidates (this is equivalent to giving 0 weights to the beginning $n - m$ candidates). For example, we may take $m = n/4$.

Suppose we have the configuration $(\mathbf{p}_t, \mathbf{q}_t)$ at present. Define $\mathbf{x} = (\mathbf{p}_t, \mathbf{q}_t)$. The multipoint HMC update for step $(t + 1)$ is as follows.

Multipoint HMC Algorithm

- From the starting state $\mathbf{x}$, we conduct n leap-frog iterations to obtain $\mathbf{y}_1, \ldots, \mathbf{y}_n$.
- Select one candidate $\mathbf{y} = \mathbf{y}_{n-k}$, say, from the candidate set $\mathbf{y}_{n-m+1}, \ldots, \mathbf{y}_n$ according to their Boltzmann probabilities. Here $0 \leq k \leq m-1$.
- From $\mathbf{x}$, run k-steps of inverse leap-frog iteration (i.e., use the negated momentum variable $-\mathbf{p}_t$ to start the leap-frog) to obtain $\mathbf{y}_{-1}, \ldots, \mathbf{y}_{-k}$.
- Accept $\mathbf{y}$ with probability

$$r_{\mathrm{mph}} = \min\left\{1, \frac{\sum_{j=1}^{m} \exp(-H(\mathbf{y}_{n-m+j}))}{\sum_{j=1}^{m} \exp(-H(\mathbf{y}_{-k-1+j}))}\right\},$$

and reject with probability $1 - r_{\mathrm{mph}}$.

The correctness of the algorithm can be seen from the facts that (a) the total energy $H(\mathbf{y})$ is not affected by negating the momentum variable; (b) the leap-frog moves are volume-preserving; and (c) the leap-frog moves are "time-reversible." That is, if we negate the momentum variable for $\mathbf{y}_n$ and apply $n+k$ leap-frog steps, we will get $\mathbf{y}_{-k}$ at the end. Intuitively, using the leap-frog transition is like using a symmetric Markov chain transition.

We can add further weights to the m candidate states considered in the multipoint HMC. That is, we can assign weight u_j to states $\mathbf{y}_{n-m+j}$ and $\mathbf{y}_{-k+m-j}$, for $j = 1, \ldots, m$. The purpose of adding weight is to give increasingly higher preference to states near the end of the leap-frog trajectory. Convenient choices for u_j are $\sqrt{j}$ and $\log j$; u_j is as in Eq. (2).

An algorithm that considers multiple trial points in HMC has been proposed earlier by Neal [14] who named it the "window" HMC. Rather than comparing only the end state of the leap-frog updates, Neal considered a comparison between the total energies of a window of states at the end of the trajectory and that at the beginning of the trajectory. After one of the two windows is chosen, a particular state within the selected window is drawn with probability proportional to its Boltzmann distribution. Figure 3 illustrates this algorithm. In Section 5.2, we compared Neal's method with ours for the simulation of uncoupled oscillators.

Trajectory length n and number of multiple trial points m need to be selected beforehand, and kept to be fixed throughout the whole simulation. However, a new step size ϵ used for leap-frog iterations has to be drawn after each acceptance/rejection decision. If both trajectory length m and step size ϵ are fixed, then the result will show a periodic and circular pattern, it can also delay the convergence. See [11] for relevant discussion.

In Neal's window algorithm, the length of the inverse trajectory k is sampled *uniformly* from a discrete set $\{1, 2, \ldots, W\}$, where W is window size. Whereas in the multipoint

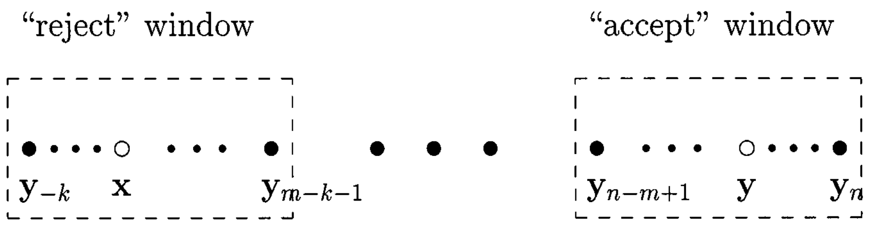

FIG. 3. Graphical illustration of Neal's "window" HMC.

method, the inverse trajectory length k is equal to the number of dynamic steps beyond the selected candidate. The later is intuitively more appealing. Additionally, we can further tune the weighting parameter u_j to bias the sampling distribution of the candidate $\mathbf{y}$ in favor of the end of the trajectory.

5. EXAMPLES

5.1. Simulation from a Mixture Gaussian Distribution

Random-grid Monte Carlo is most useful for sampling from a multimodal distribution defined on a Euclidean space because the method allows one to explore more thoroughly along a randomly chosen direction. In a sense, each of the random-grid moves behaves like the conditional update in a heat-bath algorithm (or, the *Gibbs sampler* in statistics literature). We illustrate here how the random-grid method can be applied to sample from a mixture Gaussian distribution.

Consider simulating from a two-dimensional mixture Gaussian distribution

$$.34 \times N_2(0, I_2) + .33 \times N_2\left\{\begin{pmatrix}-9\\-9\end{pmatrix}, \begin{pmatrix}1 & .9\\ .9 & 1\end{pmatrix}\right\} + .33 \times N_2\left\{\begin{pmatrix}10\\10\end{pmatrix}, \begin{pmatrix}1 & -.9\\ -.9 & 1\end{pmatrix}\right\},$$

which is similar to the ones used by Gilks *et al.* [6] and Liu *et al.* [10]. Compared to theirs, the distribution shown here is more difficult to sample from by a standard Metropolis algorithm because the mean vectors are separated by a larger distance in each dimension.

Both the standard Metropolis and the random-grid Monte Carlo methods were applied to this example. In particular, we used the two-directional random-grid method with $n = 4$ along each direction. The length of each step is generated from an exponential distribution with mean 3. A total of 50,000 iterations were conducted for both the Metropolis and the random-grid methods. Although eight candidates were considered in each iteration, the random-grid method consumed only twice as much computing time as the standard Metropolis. This is mainly because that the evaluation of $\pi(\mathbf{x})$ is easy.

A comparison of the two methods in terms of the histograms and autocorrelation plots of their Monte Carlo samples is shown in Fig. 4. The left panel is for the standard Metropolis and the right panel is for the random-grid method. The histogram on the left panel showed unequal mass among the three modes, suggesting that the equilibration time for the algorithm is very long. The two plots on the bottom panel show the autocorrelations up to lag 30 for the two methods.

We also compared *Integrated Autocorrelation Time (IAT)* of the two methods, where the IAT is defined as $\frac{1}{2} + \sum_{j=1}^{\infty} \rho_j$, with ρ_j being the lag-j autocorrelation of the Markov chain. After taking computation time into account, our simulation showed that the IAT was 32.6 for the Metropolis algorithm, after adjusting for the computational cost (2 to 1 ratio), and 5.2 for the random-grid method. This translates to a six-fold improvement. The random-grid is still more than two-fold better than the Metropolis even if we adjust the extra cost to a 6 to 1 ratio.

5.2. Uncoupled Oscillators

The behavior of the standard HMC algorithm for systems of uncoupled oscillators has been analyzed in detail by Kennedy and Pendleton [8]. Neal [14] used the same example to

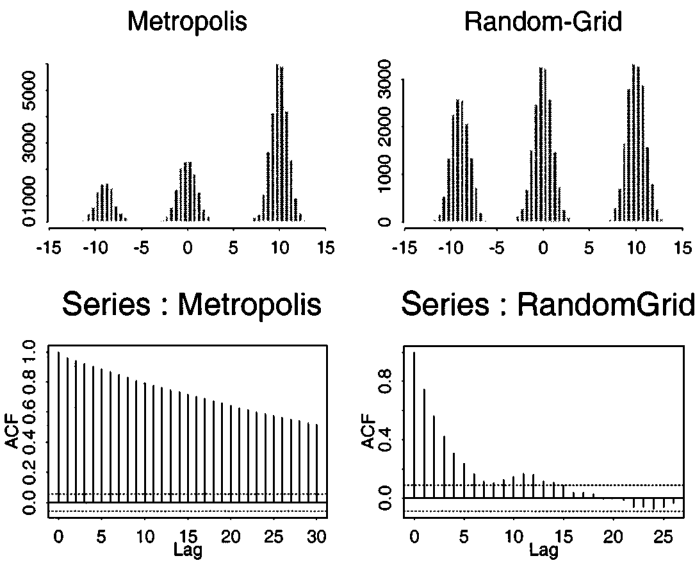

FIG. 4. A comparison of the results obtained by the standard Metropolis method and that by the random-grid method. The autocorrelation functions (ACF) has been adjusted to account for unequal computational costs. Each lag in the ACF represents 40 iterations for the Metropolis algorithm and 20 iterations for the random-grid method, respectively.

study the performance of his "window" HMC method. In this example, the system contains d uncoupled oscillators with frequencies ν_j for $j = 1, \ldots, d$. The potential energy function for such a system is

$$E(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^{d} \nu_j^2 q_j^2. \tag{3}$$

Thus, in the Boltzmann distribution with this energy function, each q_j is independent and distributed as a Gaussian with zero mean and standard deviation $1/\nu_j$. Since the HMC operation is invariant of translation and rotation of the coordinates, the above system can represent any multivariate Gaussian distribution.

In this example, we deliberately make the target distribution difficult to simulate by selecting the ν_j between 10 to 1000, a much larger range than the one used in [14]. The ratio $\nu_{\max}/\nu_{\min}$, where $\nu_{\max}$ and $\nu_{\min}$ are the largest, and the smallest of the ν_j, respectively, is regarded as a measurement of the inherent difficulty.

From our simulation result, the multipoint HMC method proposed in this paper outperformed Neal's "window" HMC method, in terms of both the autocorrelation time and the actual computation time. In most cases, the weighted multipoint method is even better than its unweighted counterpart. The acceptance rate was tuned to be almost the same across all these methods.

Three methods are compared in Fig. 5: method **N** refers to Neal's "window" HMC; method **M** the multipoint HMC proposed in this paper; and method **MW** the multipoint HMC with weight. The left panel shows a side-by-side box plot about the IAT values for 30 cases we simulated. From this plot, we can see that the general performance ranking is **MW** $\succ$ **M** $\succ$ **N**. The right panel shows a scatter plot, in which the x-axis is the IAT values for

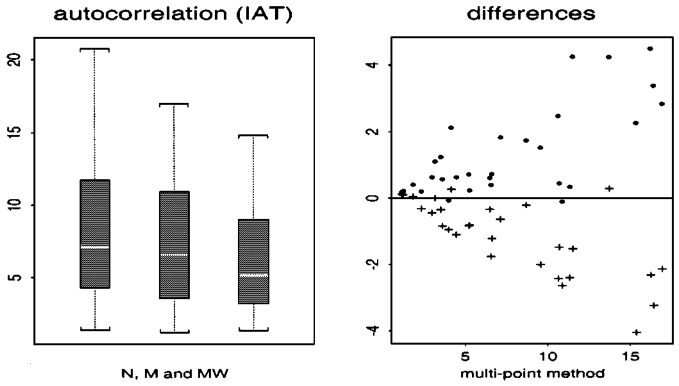

FIG. 5. Comparison of autocorrelation for method **N**. **M** and **MW**.

method **M**. The **y**-axis corresponds to the difference in IAT value, between methods **N** and **M**, **M** and **MW**, respectively. The fan shape of the scatter plot shows that the improvement increases when the IAT score is larger.

In order to make sure that the total leap-frog steps in all three methods are comparable, we set trajectory length to be 45 for method **M** and **MW**, 50 for method **N**. Window size is 20. Step size ϵ is sampled uniformly from interval $(0, 2/\nu_{\max})$. The dimension of the system was taken to be 200.

Another experiment was performed for a system of 1600-dimensional uncoupled oscillators. We compared four different methods: the standard HMC, Neal's "window" HMC, multipoint HMC, and multipoint HMC with weight, corresponding to four panels from top to bottom. We also tested different settings of two adjustable parameters: one is the number of leap-frog steps n, which were set at 100, 50, and 30, respectively; the other parameter controls maximum length of a single leap-frog move, denoted as $\epsilon_{\max}$, which were set at .75, 1.0, 1.5, and 2.0. The leap-frog step length is sampled uniformly from $(0, \epsilon_{\max}/\nu_{\max})$. We summarized the computation times (for 50,000 iterations under each setting), acceptance rates, and the IATs in Table I. It is seen from the table that one can obtain a very efficient setting (IAT = 1.11) by step size, number of steps, and the weights in the multipoint selection.

6. DISCUSSION

In this paper, we propose the multipoint method as a novel extension of the orientational-bias Monte Carlo, which can be used to alleviate the local-mode trapping problems encountered in all the Metropolis-type algorithms. In the new method, multiple correlated candidates are proposed simultaneously, from which one is selected judiciously to compare with the current configuration. To encourage the acceptance of those candidates that are "farther" away from the current state, a weight function can be used to emphasize each candidate's importance.

Two applications of the general multipoint idea, the random-grid method and the multipoint HMC, are described. In the random-grid method, the multiple candidates lie with equal spacing along a randomly chosen direction. In multipoint HMC, the multiple candidates are just the points near the end of a leap-frog trajectory. It has been shown recently

TABLE I
Comparison of Simulation Results

n	100			50			30		
$\epsilon_{\max}$	*t*	*AR*	*IAT*	*t*	*AR*	*IAT*	*t*	*AR*	*IAT*
Standard HMC									
0.75	92.9	88.0	12.35	52.0	86.5	20.97	31.1	88.2	27.58
1.0	89.7	81.0	7.35	52.1	79.4	12.49	31.1	76.5	23.75
1.5	91.7	55.9	11.61	51.4	53.3	19.35	31.0	53.3	22.50
2.0	92.7	40.0	11.19	52.0	41.0	19.66	31.6	42.0	28.07
Neal's "window" HMC									
0.75	108.4	95.6	8.50	57.7	96.2	14.65	32.5	94.7	27.31
1.0	107.0	88.0	8.32	59.2	90.5	13.86	32.6	92.7	20.90
1.5	107.0	72.2	5.73	58.5	74.9	12.07	32.6	68.8	20.90
2.0	108.2	53.7	7.31	56.7	51.4	13.67	32.9	51.4	19.67
Multipoint HMC									
0.75	112.3	96.9	10.57	64.7	96.7	21.46	34.9	95.3	23.50
1.0	113.04	94.6	5.04	63.0	93.7	17.15	33.7	91.7	23.30
1.5	114.7	78.3	4.32	63.4	76.0	16.78	35.4	72.6	22.80
2.0	116.5	61.2	7.61	63.3	57.0	16.86	35.4	54.5	23.38
Weighted Multipoint HMC									
0.75	108.8	96.5	9.20	58.2	94.4	20.63	33.8	94.7	24.83
1.0	107.2	87.4	1.11	57.8	91.2	14.74	33.6	93.3	19.79
1.5	107.2	72.5	6.76	58.1	74.4	16.05	33.6	71.4	23.57
2.0	107.6	53.7	3.25	57.5	52.7	9.86	34.3	55.2	23.73

that the Nośe–Hoover chain method [12] generally outperforms the HMC in simulating Hamiltonian dynamics. It is thus desirable to apply the multipoint idea to this more advanced technique.

The multipoint is a general methodology to improve the local search capability of a MCMC sampler and can be combined with various other powerful methods, such as parallel tempering [5] and adaptive directional sampling [10], to produce more efficient Monte Carlo algorithms [9].

APPENDIX: PROOFS OF THE DETAILED BALANCE

A.1. The General Multipoint Method

Let $A(\mathbf{y} \mid \mathbf{x})$ be the actual transition function and let $\mathbf{y}_k = \mathbf{y}$ be the candidate chosen from the multiple trial points. As described in Section 2, the backward "reference set" is denoted as $\{\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*\}$, where $\mathbf{x}_j^* = \mathbf{y}_{k-j}$ for $j = 1, \ldots, k-1$; $\mathbf{x}_k^* \equiv \mathbf{x}$; and $\mathbf{x}_{l+1}^* \sim T_{l+1}(\cdot \mid \mathbf{y}, \mathbf{x}_{[1:l]}^*)$ for $l = k+1, \ldots, n$. With these notations, we want to prove the "super-detailed balance"

$$\pi(\mathbf{x}) A_k(\mathbf{y} \mid \mathbf{x}) = \pi(\mathbf{y}) A_k(\mathbf{x} \mid \mathbf{y}).$$

The detailed balance follows by summing up all possible values of k in the above equation.

Note that the derivation of the actual transition function $A_k(\mathbf{y} \mid \mathbf{x})$ involves integrating over all the remaining "reference points," $\mathbf{y}_{[1:k-1]}$, $\mathbf{y}_{[k+1:n]}$, and $\mathbf{x}^*_{[k+1:n]}$. More precisely, we have

$$\pi(\mathbf{x})A_k(\mathbf{y} \mid \mathbf{x}) = \int \cdots \int \pi(\mathbf{x}) T\left(\mathbf{y}_{[1:n]} \mid \mathbf{x}\right) \frac{\omega\left(\mathbf{y}_{[k:1]}, \mathbf{x}\right)}{\sum_{j=1}^n \omega\left(\mathbf{y}_{[j:1]}, \mathbf{x}\right)} \min\left\{1, \frac{\sum_{j=1}^n \omega\left(\mathbf{y}_{[j:1]}, \mathbf{x}\right)}{\sum_{j=1}^n \omega\left(\mathbf{x}^*_{[j:1]}, \mathbf{y}\right)}\right\}$$
$$\times T\left(\mathbf{x}^*_{[k+1:n]} \,\middle|\, \mathbf{y}_{[k:1]}, \mathbf{x}\right) d\mathbf{y}_{[k+1:n]}\, d\mathbf{x}^*_{[k+1:n]}\, d\mathbf{y}_{[1:k-1]} \tag{A.1}$$
$$= \int \cdots \int \pi(\mathbf{x}) T\left(\mathbf{y}_{[1:n]} \,\middle|\, \mathbf{x}\right) \pi(\mathbf{y}) T\left(\mathbf{x}^*_{[1:n]} \,\middle|\, \mathbf{y}\right) \lambda\left(\mathbf{y}, \mathbf{x}^*_{[1:k]}\right)$$
$$\times \min\left\{\frac{1}{\sum_{j=1}^n \omega\left(\mathbf{y}_{[j:1]}, \mathbf{x}\right)}, \frac{1}{\sum_{j=1}^n \omega\left(\mathbf{x}^*_{[j:1]}, \mathbf{y}\right)}\right\}$$
$$\times d\mathbf{y}_{[k+1:n]}\, d\mathbf{x}^*_{[k+1:n]}\, d\mathbf{y}_{[1:k-1]}. \tag{A.2}$$

Here function λ is any positive and "sequentially symmetric" function, i.e.,

$$\lambda\left(\mathbf{y}, \mathbf{x}^*_{[1:k]}\right) = \lambda\left(\mathbf{x}^*_{[k:1]}, \mathbf{y}\right).$$

To see that expression (A.2) is symmetric in $\mathbf{x}$ and $\mathbf{y}$, we simply exchange the notations of $\mathbf{x}$ and $\mathbf{y}$, and $\mathbf{x}^*_j$ and $\mathbf{y}_j$, respectively. Because λ is sequentially symmetric, we have

$$\lambda\left(\mathbf{y}, \mathbf{x}^*_{[1:k]}\right) = \lambda\left(\mathbf{x}^*_{[k:1]}, \mathbf{y}\right) \equiv \lambda\left(\mathbf{x}, \mathbf{y}_{[1:k]}\right).$$

Thus, expression (A.2) does not change its value after the above notational exchange. This concludes the proof of the super-detailed balance condition

$$\pi(\mathbf{x})A_k(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y})A_k(\mathbf{y}, \mathbf{x}).$$

A.2. Random-Grid Metropolis

For this algorithm, we have

$$\pi(\mathbf{x})A_k(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{x}) \int_R g(r) \frac{\pi(\mathbf{y})}{\sum_{j=1}^n \pi(\mathbf{y}_j)} \min\left\{1, \frac{\sum_{j=1}^n \pi(\mathbf{y}_j)}{\sum_{j=0}^{n-k} \pi(\mathbf{x}^*_j) + \sum_{j=1}^{k-1} \pi(\mathbf{y}_j)}\right\} dr$$
$$= \pi(\mathbf{x}) \int g(r) \frac{\pi(\mathbf{y})}{\sum_{j=1}^n \pi(\mathbf{x} + jr, \mathbf{x})} \min\left\{1, \frac{\sum_{j=1}^n \pi(\mathbf{x} + jr, \mathbf{x})}{\sum_{j=1}^n \pi(\mathbf{y} - jr, \mathbf{y})}\right\} dr$$
$$= \pi(\mathbf{x}) \int g(r) \pi(\mathbf{y}) \min\left\{\frac{1}{\sum_{j=1}^n \pi(\mathbf{x} + jr, \mathbf{x})}, \frac{1}{\sum_{j=1}^n \pi(\mathbf{y} - jr, \mathbf{y})}\right\} dr.$$

The above expression is apparently symmetric in $\mathbf{x}$ and $\mathbf{y}$, thus we proved that $\pi(\mathbf{x})A_k(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y})A_k(\mathbf{y}, \mathbf{x})$, which is the detailed balance condition.

For two-sided random-grid Metropolis, the proof is very similar,

$$\begin{aligned}\pi(\mathbf{x})A_k(\mathbf{x},\mathbf{y}) &= \pi(\mathbf{x})\int g(r)\frac{\pi(\mathbf{y})}{\sum_j \pi(\mathbf{y}\pm jr)}\min\left\{1, \frac{\sum_j \pi(\mathbf{y}\pm jr)}{\sum_j \pi(\mathbf{x}\pm jr)}\right\}dr\\ &= \pi(\mathbf{x})\int g(r)\frac{\pi(\mathbf{y})}{\sum_j \pi(\mathbf{x}\pm jr)}\min\left\{1, \frac{\sum_j \pi(\mathbf{x}\pm jr)}{\sum_j \pi(\mathbf{y}\pm jr)}\right\}dr\\ &= \pi(\mathbf{x})\int g(r)\pi(\mathbf{y})\min\left\{\frac{1}{\sum_j \pi(\mathbf{x}\pm jr)}, \frac{1}{\sum_j \pi(\mathbf{y}\pm jr)}\right\}dr.\end{aligned}$$

The above expression is symmetric in **x** and **y**; the detailed balance is proved.

ACKNOWLEDGMENTS

The work was partially supported by the National Science Foundation. Part of the work was conducted while JSL was on the faculty of the Statistics Department of Stanford University. The authors thank the two anonymous referees for insightful suggestions.

REFERENCES

1. S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, Hybrid Monte Carlo. *Phys. Lett. B* **195**, 216 (1987).
2. K. Esselink, L. D. J. C. Loyens, and B. Smit, Parallel Monte Carlo simulation, *Phys. Rev. E* **51**, 1560 (1995).
3. M. Falconi and M. W. Deem, A biased Monte Carlo scheme for zeolite structure solution, *J. Chem. Phys.* **110**, 1754 (1999).
4. D. Frenkel and B. Smit, *Understanding Molecular Simulation* (Academic Press, New York, 1996).
5. C. J. Geyer, Markov chain Monte Carlo maximum likelihood, in *Computing Science and Statistics: Proc. 23rd Symp. Interface*, edited by E. Keramigas (Interface Foundation, Fairfax, VA, 1991).
6. W. R. Gilks, G. O. Roberts, and S. K. Sahu, Adaptive Markov chain Monte Carlo through regeneration, *J. Amer. Stat. Assoc.* **93**, 1045 (1998).
7. W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, *Biometrika*, **57**, 97 (1970).
8. A. D. Kennedy and B. Pendleton, Acceptances and autocorrelations in hybrid Monte Carlo, *Nucl. Phys. B* (*Proc. Suppl.*) **20**, 118 (1991).
9. F. Liang and W. H. Wong, Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models, *J. Amer. Stat. Assoc.* **96**, 653 (2001).
10. J. S. Liu, F. M. Liang, and W. H. Wong, The use of multiple-try method and local optimization in Metropolis sampling, *J. Amer. Stat. Asso.* **95**, 121 (2000).
11. P. B. Mackenzie, An improved hybrid Monte Carlo method, *Phys. Lett. B* **226**, 369 (1989).
12. G. J. Martyna and M. L. Klein, Nośe–Hoover chains: the canonical ensemble via continuous dynamics, *J. Chem. Phys.* **97**, 2635 (1992).
13. N. Metropolis, A. W. Rosenbluth, M. N. Rothenbluth, A. H. Teller, and E. Teller, Equations of state calculations by fast computing machines, *J. Chem. Phys.* **21**, 1087 (1953).
14. R. M. Neal, An improved acceptance procedure for the hybrid Monte Carlo algorithm, *J. Comput. Phys.* **111**, 194 (1994).