

Multi-Point Metropolis Method with Application to Hybrid Monte Carlo

Zhaohui Qin and Jun S. Liu¹

Department of Statistics, Harvard University, 1 Oxford Street, Cambridge, MA 02138

E-mail: qin@stat.harvard.edu; jliu@stat.harvard.edu

We propose the multi-point Metropolis algorithm as a non-trivial extension of a method by Frenkel and Smit. A new statistics similar to the Metropolis ratio is introduced to maintain the detailed balance. The multi-point idea can be generally applied to improve the efficiency of a Markov chain-based Monte Carlo algorithm. As an illustration, we describe two variations — the random-grid Metropolis and the multi-point Hybrid Monte Carlo — and apply them to a number of examples.

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

¹Support in part by the National Science Foundation under grants DMS-9803649 and DMS-0094613. Part of the work was conducted while JSL was on the faculty of the Statistics Department of Stanford University

tools used for these endeavors is Monte Carlo. Recently, statisticians begin 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 (also called partition function). It is of interest to estimate the expectation of a given function $h(\cdot)$, i.e., the value of

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

If random samples $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ can be drawn from $\pi(\mathbf{x})$, then we 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 Markov chain-based simulation methods are perhaps most widely used in practice. The construction proposed by Metropolis *et al.* [11] and modified by Hastings [6], often referred to as the Metropolis-Hastings (M-H) algorithm, is the most fundamental building block for designing a Markov chain Monte Carlo (MCMC) algorithm. The M-H algorithm can be easily implemented as an iteration of the following steps

- Suppose at the t -th iteration we obtained a sample \mathbf{x}_t . At the next iteration we propose a “perturbation” of the current state, which can be realized by generating a candidate \mathbf{y} from a rather arbitrary Markov transition function $T(\cdot | \mathbf{x}_t)$ (also called the proposal function).

- We let the next state $\mathbf{x}_{t+1} = \mathbf{y}$ with probability

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

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

Theoretically, the M-H algorithm can be applied to an arbitrary distribution π given up to a normalizing constant, regardless of its dimensionality. However, in most serious 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 $T(\cdot | \cdot)$ 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. A low acceptance rate also hurts the efficiency of the algorithm. The multi-point method introduced in this article aims at alleviating this conflict of interests.

***New addition for revision One can imagine that several steps is taken instead of just one from the current state, such that a larger vicinity space of the current state can be explored, which perhaps will increase the opportunity of escaping from a local mode trap. Compare to other more sophisticated techniques, such as parallel tempering [3] and reference there in, multi-point method provide a simple alternative, especially suitable for moderate complex system.

The multi-point method is closely related to the “multiple-try Metropolis” (MTM) method proposed in [8]. Both methods can be viewed as generalizations of the “orientational-biased 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 $T(\cdot | \cdot)$. As a consequence, one can afford to employ a proposal transition T which covers a relatively larger region in the state space. Our new method further allows one to propose multiple *dependent* trial points. This extra feature greatly extends the applicability of the multi-point idea. In particular, we will show that the method can be used to improve a hybrid Monte Carlo (HMC) algorithm.

2. MULTI-POINT 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 multi-point 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 $\mathbf{y}_1 \sim T_1(\cdot | \mathbf{x})$, $\mathbf{y}_2 \sim T_2(\cdot | \mathbf{x}, \mathbf{y}_1)$, \dots , and $\mathbf{y}_n \sim T_n(\cdot | \mathbf{x}, \mathbf{y}_1, \dots, \mathbf{y}_{n-1})$. For briefness, we let $\mathbf{y}_{[1:j]} = (\mathbf{y}_1, \dots, \mathbf{y}_j)$ and let $\mathbf{y}_{[j:1]}$ denote the vector with the reverse order, i.e., $\mathbf{y}_{[j:1]} = (\mathbf{y}_j, \mathbf{y}_{j-1}, \dots, \mathbf{y}_1)$. We suppress all the subscripts for T . That is, the joint sampling distribution of $\mathbf{y}_{[1:j]}$ is written as

$$T(\mathbf{y}_{[1:j]} | \mathbf{x}) \equiv T(\mathbf{y}_1 | \mathbf{x}) \times \dots \times T(\mathbf{y}_j | \mathbf{x}, \mathbf{y}_{[1:(j-1)]}), \quad (1)$$

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

$$\omega_j(\mathbf{x}, \mathbf{y}_{[1:j]}) = \pi(\mathbf{x})T(\mathbf{y}_{[1:j]} | \mathbf{x})\lambda_j(\mathbf{x}, \mathbf{y}_1, \dots, \mathbf{y}_j).$$

It is seen that $\pi(\mathbf{x})T(\mathbf{y}_{[1:j]} | \mathbf{x})$ 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

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

Again, in the remaining part of the article, we suppress all the subscripts for functions λ_j and ω_j . The details of the multi-point algorithm are as follows:

Multi-Point Metropolis Algorithm

- Draw n trial points, $\mathbf{y}_1, \dots, \mathbf{y}_n$ according to the joint proposal distribution T , as defined in (1); compute $\omega(\mathbf{y}_{[j:1]}, \mathbf{x})$ for $j = 1, \dots, n$.

- Select \mathbf{y} among the trial set $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ with probability proportional to $\omega(\mathbf{y}_{[j:1]}, \mathbf{x})$, $j = 1, \dots, n$. Suppose $\mathbf{y} = \mathbf{y}_k$ has been selected.

- Construct a “reference set” $\{\mathbf{x}_{k+1}^*, \dots, \mathbf{x}_n^*\}$ by sampling \mathbf{x}_{j+1}^* from

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

for $j = k, \dots, n-1$. For notational simplicity, we name $\mathbf{x}_j^* = \mathbf{y}_{k-j}$ for $j = 1, \dots, k-1$ and $\mathbf{x}_k^* = \mathbf{x}$. Thus, the reference path $\{\mathbf{x}_1^*, \dots, \mathbf{x}_n^*\}$ is like a reversal of the forward path $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ 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

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

and reject it with probability $1 - r_{mp}$. The quantity r_{mp} is referred to as the multi-point Metropolis ratio.

Figure 1 gives a cartoon illustration of the algorithm. Compared with OBMC and MTM, a unique feature of our new algorithm is that the middle $k+1$ points are reused in the reference set.

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 counter-balanced 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 multi-point method satisfies the detailed balance condition.

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

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

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

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

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

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

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

The multi-point 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. At the other extreme, all the multiple trial points can be generated independently as in OBMC [4] and MTM [8]. A specially useful scenario is that the multiple candidates are generated by a Markov chain. For example, one may have a favorite transition function $K(\mathbf{y} | \mathbf{x})$, which is a “cheap” approximation of the desirable transition kernel $A(\mathbf{y} | \mathbf{x})$ (whose equilibrium distribution is π). Then one can generate a few candidates from composing K and use the multi-point method to select among them. In many applications, however, we may favor those points that are “farther” away from \mathbf{x} in the set of multiple trial points. The following *weighted multi-point* method can be used to achieve this goal.

Weighted Multi-Point 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 an additional weight to each generated candidate. 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

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

and use $\tilde{\omega}$ in the place of ω in the multi-point algorithm previously defined. Typical choices can be $u_j = j^\alpha$ or $u_j = \log j$, both give increasing preference to points “farther” away. In fact, this flexibility has been reflected in the original multi-point algorithm for that we can choose λ_j freely. Expression (2) just 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 hybrid Monte Carlo method (Section 4).

3. RANDOM-GRID METROPOLIS

As a demonstration of how one can use the multi-point 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

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

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

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

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

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

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

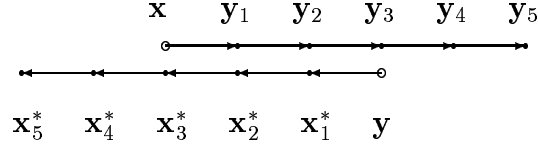


FIG. 1. Graphical illustration of a random-grid Metropolis step. In this case, $n = 5$, $k = 3$.

$y = y_3$ was selected as candidate. Then we name $x_1^* = y_2$, $x_2^* = y_1$, $x_3^* = x$.

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{e} , we can construct $2n$ candidates as

$$y_{-j} = x - j \cdot r \cdot \vec{e}, \quad j = 1, \dots, n,$$

$$y_j = x + j \cdot r \cdot \vec{e}, \quad j = 1, \dots, n.$$

Then, after selecting a candidate y from the candidate set, we construct the reference set as

$$x_j^* = y + j \cdot r \cdot \vec{e},$$

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

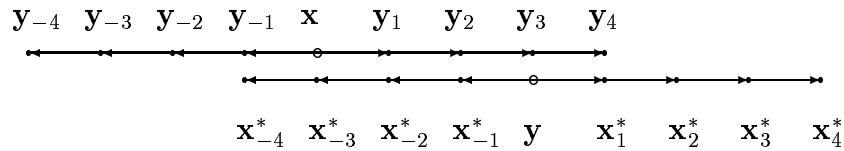


FIG. 2. Illustration of two-sided random-grid Metropolis chain. In this case, $n = 4$, $k = 3$.

$y = y_3$ was selected as candidate. Then we name $x_1^* = y_4$, $x_{-1}^* = y_2$, $x_{-2}^* = y_1$, $x_{-3}^* = x$, $x_{-4}^* = y_{-1}$.

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 $\vec{\epsilon}$ 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 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 multi-point iterations.

4. MULTI-POINT MODIFICATION OF HYBRID MONTE CARLO

4.1. Introduction to Hybrid Monte Carlo

Hybrid Monte Carlo (HMC) is first proposed in [1] to deal with numerical simulation problems in lattice field theory. The reason why the authors name it “hybrid” is that this algorithm, unlike other Monte Carlo methods for lattice systems, combines molecular dynamics steps with a Metropolis acceptance/rejection rule. 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 configuration (positional) and $E(\mathbf{q})$ is the potential energy function.

In order to implement HMC, a fictitious N -dimensional momentum vector \mathbf{p} , is introduced. The total energy function $H(\mathbf{p}, \mathbf{q})$ (the Hamiltonian) incorporates both the potential and the kinetic energy: $H(\mathbf{p}, \mathbf{q}) = E(\mathbf{q}) + \frac{1}{2}|\mathbf{p}|^2$. If we can sample from the joint distribution $\pi(\mathbf{p}, \mathbf{q}) \propto \exp(-H(\mathbf{p}, \mathbf{q}))$ of the position vector and the momentum vector, then the resulting marginal distribution of \mathbf{q} follows

the desired Boltzmann distribution. (Note that one can sample \mathbf{p} from its marginal distribution easily.) Because a dynamical move of the system based on 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 constant total energy, these Hamiltonian moves leave the joint distribution $\pi(\mathbf{p}, \mathbf{q})$ invariant. Thus, an ideal approach to simulate (\mathbf{p}, \mathbf{q}) from π is to (a) draw \mathbf{p} from its marginal distribution (which is Gaussian), and (b) evolve the system according to the Hamiltonian equations for a period of time.

In practice, however, the Hamiltonian dynamics can only be simulated approximately by discretizing the corresponding differential equations, which will result in “non-physical” moves. A popular method is the “leap-frog algorithm”. Each leapfrog step is time-reversible and volume-preserving [1], but it does not maintain a constant total energy. For this reason, the method introduced in [1] used a Metropolis rejection rule to correct for the bias resulting from discretization. Intuitively, leap-frog steps serve as good proposals for implementing a Markov-chain based Monte Carlo method. The HMC algorithm can be summarized as follows:

Hybrid Monte Carlo Algorithm

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

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

- 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 the new position and momenta 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} together although we are only interested in \mathbf{q} . Vector \mathbf{p} can be viewed as an auxiliary variable.

4.2. Using Multi-Point 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 multi-point 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 multi-point method. Since the number of dynamical steps (n) may be a large number, treating these n candidates equally as in the standard multi-point 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 early $n - m$ candidates). For example, we may take $m = n/4$. The algorithm can be described as follows.

Assume at iteration t , after stochastic update, the current energy and momentum are $(\mathbf{p}_t, \mathbf{q}_t)$. Define $\mathbf{x} = (\mathbf{p}_t, \mathbf{q}_t)$.

Multi-Point HMC Algorithm

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

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

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

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 multi-point HMC. That is, we can assign weight u_j to states \mathbf{y}_{n-m+j} and \mathbf{y}_{-k+m-j} , for $j = 1, \dots, m$. 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 Equation 2.

An algorithm that considers multiple trial points in HMC has been proposed earlier by Neal [12] 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.

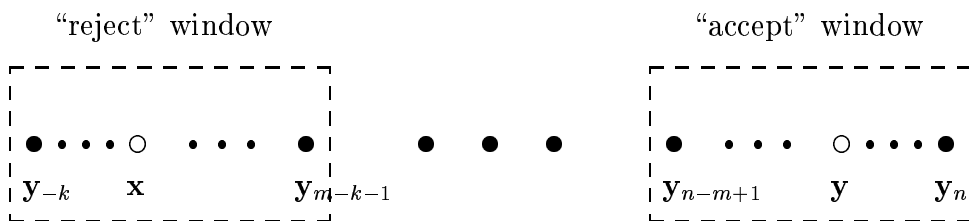


FIG. 3. Graphical illustration of Neal’s “window” HMC.

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 [9] for relevant discussion.

In Neal's window algorithm, the length of the inverse trajectory k is sampled *uniformly* from a discrete set $\{1, 2, \dots, W\}$ where W is window size. Whereas in the multi-point 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 multi-modal 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 2-dimensional mixture Gaussian distribution

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

which is similar to the ones used by Gilks *et al.* [5] and Liu *et al.* [8]. 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. *****revision***** (Other n may also be considered, such as 3 or 5.) The length of each step is generated from an exponential distribution with mean 3. A total of 50,000 and $50,000/8=6250$ iterations were conducted for the classical Metropolis method and the random-grid methods respectively. *****end***** Because multiple candidates are considered in each iteration, the random-grid method consumed twice as much computing time as the standard Metropolis. However, the random-grid method greatly improved the acceptance rate and reduced the autocorrelation among iterations. The acceptance rate is about 27% for standard Metropolis and 40% for random-grid Metropolis. a 50% improvements. A comparison of the two methods in terms of the histograms and autocorrelation plots of their Monte Carlo samples is shown in Figure 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)*, which is defined as the total sum of all the autocorrelations, for the two methods. We approximated the IAT by the sum of first N lag autocorrelations. After taking computation time into account, our simulation showed that the IAT for the Metropolis algorithm was about 33.1 after adjusting for the computational cost (2 to 1 ratio), whereas for the random-grid Metropolis, the IAT was about 5.7, which translates to a 6-fold improvement.

5.2. Uncoupled Oscillators

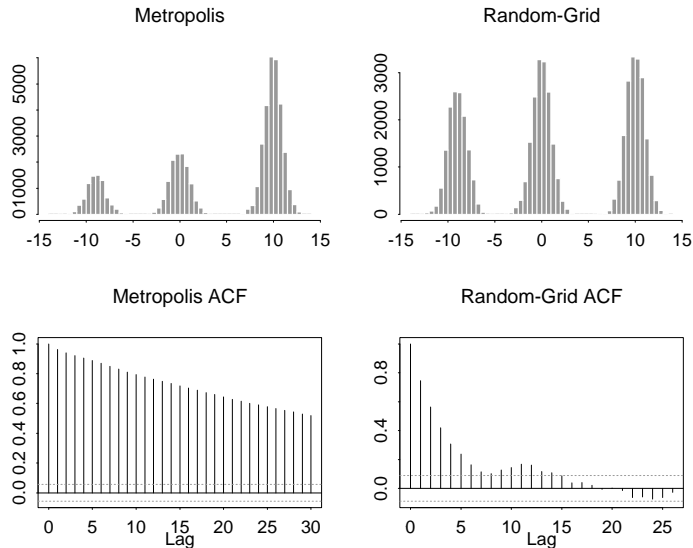


FIG. 4. A comparison of the results obtained by the standard Metropolis method and that by the random-grid method. The autocorrelation functions (ACF) has been adjusted to account for unequal computational costs. Each lag in the ACF represents 40 iterations for the Metropolis algorithm and 20 iterations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

The behavior of the standard HMC algorithm for systems of uncoupled oscillators has been analyzed in detail by Kennedy and Pendleton [7]. Neal [12] used the same example to study the performance of his “window” HMC method. Here we applied our methods to this example for comparison.

In this example, the system contains N uncoupled oscillators with frequencies ν_j for $j = 1, \dots, d$. The potential energy function for such a system is

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

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 sample from by selecting the ν_j to range from 10 to 1000, a much larger range than the one used in [12]. The ratio ν_{\max}/ν_{\min} , where ν_{\max} and ν_{\min} are the largest and smallest of the ν_j , respectively, is regarded as a measurement of the inherent difficulty.

From our simulation result, the multi-point 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 multi-point 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 Figure 5: method **N** refers to Neal’s “window” HMC; method **M** the multi-point HMC proposed in this paper; and method **MW** the multi-point 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 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 leapfrog 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, multi-point HMC, and multi-point 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 sin-

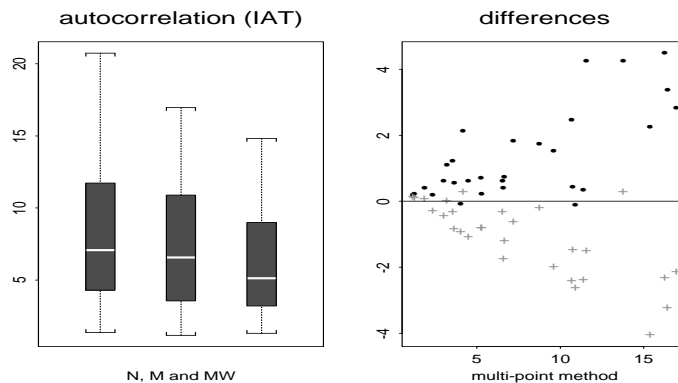


FIG. 5. Comparison of autocorrelation for method **N**, **M** and **MW**

gle leapfrog move, denoted as ϵ_{\max} , which were set at .75, 1.0, 1.5, and 2.0. The leapfrog 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 1. It's 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 multi-point selection.

new for revision

6. DISCUSSION

In this paper, a novel Metropolis type algorithm is proposed to avoid local mode trapping problems encountered in classical Metropolis-Hastings scheme while maintaining a reasonable acceptance rate. to be specific, multiple candidates are sequentially generated by taking several steps away from the current state, and choose the optimal from the candidates. One special case is the so called random-grid Metropolis method, where all candidates are lie on the same or opposite direction and the distance between two adjacent candidates are the same.

Other powerful algorithms such as parallel tempering [3] proves to perform well when dealing with local mode trapping problems. However, our algorithm only

TABLE 1

Comparison of simulation results.

n	100			50			30		
	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
Multi-Point 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 Multi-Point 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

requires one chain, perhaps simpler and more efficient to implement when dealing with moderate complex system. Formal comparison is needed before any conclusion can be reached.

It's also showed in this paper that the multi-point idea can be added into the Hybrid Monte Carlo scheme to improve its efficiency and acceptance rate. This also prompted us to think if the multi-point idea can be used in other Monte Carlo scheme such as the Noose-Hoover chain [10]. It's currently under investigation.

REFERENCES

1. S. Duane, A. D. Kennedy, B. J. Pendleton and D. Roweth. *Phys. Lett. B* **195**. 216 (1987).
2. K. Esselink, L. D. J. C. Loyens, and B. Smit. *Phys. Rev. E* **51** 1560 (1995)
3. M. Falconi, and M. W. Deem. *J. Chem. Phys.* **110**. 1754 (1999)
4. Frenkel, D. and Smit, B. *Understanding Molecular Simulation*. (Academic Press, New York. 1996).
5. W. R. Gilks, G. O. Roberts, and S. K. Sahu. *J. Amer. Statis. Assos.* **93**. 1045 (1998).
6. W. K. Hastings, *Biometrika*. **57**. 97 (1970).
7. A. D. Kennedy and B. Pendleton. *Nucl Phys. B (Proc. Suppl.)* **20**. 118 (1991).
8. J. S. Liu, F. M. Liang, and W. H. Wong. *J. Amer. Statis. Asso.* **95**. 121 (2000).
9. P. B. Mackenzie, *Phys. Lett. B* **226**. 369 (1989).
10. G. J. Martyna, and M. L. Klein. *J. Chem. Phys.* **97** 2635 (1992)
11. N. Metropolis, A. W. Rosenbluth, M. N. Rothenbluth, A. H. Teller, and E. Teller. *J. Chem Phys.* **21**. 1087 (1953).
12. R. M. Neal. *J. Comput. Phys.* **111**. 194 (1994).

APPENDIX: PROOFS OF THE DETAILED BALANCE

A.1. THE GENERAL MULTI-POINT 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 $\{\mathbf{x}_1^*, \dots, \mathbf{x}_n^*\}$, where $\mathbf{x}_j^* = \mathbf{y}_{k-j}$ for $j = 1, \dots, k-1$; $\mathbf{x}_k^* \equiv \mathbf{x}$; and $\mathbf{x}_{l+1}^* \sim T_{l+1}(\cdot \mid \mathbf{y}, \mathbf{x}_{[1:l]}^*)$ for $l = k+1, \dots, n$. With these notations, we want to prove the detailed balance

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

It is noticed, however, the derivation of the actual transition function $A(\mathbf{y} \mid \mathbf{x})$ involves integrating over all the remaining “reference points,” i.e., $\mathbf{y}_{[1:k-1]}$, $\mathbf{y}_{[k+1:n]}$, and $\mathbf{x}_{[k+1:n]}^*$. More precisely, we have $\pi(\mathbf{x})A(\mathbf{y} \mid \mathbf{x}) =$

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

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

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

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

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

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

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

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

A.2. RANDOM-GRID METROPOLIS

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

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

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

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

The above expression is symmetric in \mathbf{x} and \mathbf{y} , the detailed balance is proved.