Monte Carlo techniques: an overview

Daniel Rothman

INTRODUCTION

The statics algorithm I discuss elsewhere in this report (Rothman, 1984; hereafter referred to as paper 1) is a straightforward implementation of a technique originally introduced by Metropolis et al. (1953). While the algorithm is simple, its implications and justifications can be subtle. The Metropolis algorithm has been popular during the last 30 years among physicists attempting numerical calculations with large systems, and has accordingly been the subject of extensive review. This paper presents a further review of the technique, with emphasis given to geophysical applications. More efficient versions of the basic Metropolis technique are discussed; one method appears particularly promising for residual statics estimation.

AN HISTORICAL PERSPECTIVE

Much of the following discussion is based on material in Hammersley and Handscomb (1964), which appears to be the most readable account of the relevant Monte Carlo theory.

Historically, Metropolis' Monte Carlo method was designed to "solve" the following general problem. A physical system is in *thermal equilibrium* if its probability of being in some state σ with energy $E(\sigma)$ is proportional to the *Boltzmann factor*

$$e^{-\beta E(\sigma)} , \tag{1}$$

where $\beta = (kT)^{-1}$, T is the absolute temperature of the system, and k is Boltzmann's constant. The usual applications in physics are concerned with evaluations of the ergodic averages

$$< f > \; = \; \frac{\int f(\sigma) e^{-\beta E(\sigma)} d\sigma}{\int e^{-\beta E(\sigma)} d\sigma} \tag{2}$$

where f is some function of the system's states, and $< f >$ is the expectation of f. In a discrete phase-space the integrations would of course be summations.

In statistical mechanics, the number of degrees of freedom in phase-space is large, so σ is an n-vector where n is typically of the order of Avogadro's number (10^{23}). Usually (2) cannot be evaluated analytically; moreover, numerical evaluation is non-trivial due to the high dimensionality. Evaluation of (2) is thus performed by Monte Carlo techniques.

The simplest technique of Monte Carlo integration would be the following: randomly generate σ within some interval, evaluate the integrands in (2), and save the result. Repeat this some large number of times and then sum the results to obtain an estimate of $< f >$. Unfortunately, this technique is poor because the exponential factor in (2) causes the greatest contributions to the integrals to occur in a small region of phase-space [where $E(\sigma)$ is minimum] that is likely to be missed by the random number generator. A more appropriate technique is *importance sampling* (Fosdick, 1963; Hammersley and Handscomb, 1964). One randomly generates states with a probability density of

$$P(\sigma) = \frac{e^{-\beta E(\sigma)}}{\int e^{-\beta E(\sigma)} d\sigma} \quad . \tag{3}$$

Estimates of $< f >$ are then simple, uniformly weighted averages of the f's evaluated at these randomly generated states.

Numerical integration of (2) is thus reduced to devising a technique for randomly sampling from (3). This, too, appears difficult because of the nasty multidimensional integral in the denominator (in its discrete form, the denominator is the *partition function* of statistical mechanics). Metropolis et al. overcame this problem by generating a *Markov chain* that asymptotically approaches the probability density (3).

Before getting more deeply into Markov chains, let's assume that we have performed a Monte Carlo simulation that generates states with the Gibbs measure (3). What then? If you're a physicist you'll probably want to perform a uniformly weighted average of the algorithm's output at each time step, which is equation (2). If you're a geophysicist trying to estimate residual statics, however, things are somewhat more complicated. For this task, we need not only run the algorithm to equilibrium, in which states are generated with probability (3), but we must attain equilibrium at a temperature that is sufficiently low so that equilibrium closely approximates the state in which stack power is maximum [$E(\sigma)$ is minimum]. Usually, then, we will simply choose our estimate of statics to be the configuration with the greatest stack power.

MARKOV CHAINS

I will review only the basic essentials of Markov chain theory necessary for the ensuing discussion. The interested reader should refer to chapter 15 of Feller (1968) or any other of the myriad of texts that go into greater detail.

Let $\mathbf{X} = \{X_1, X_2, \cdots, X_t\}$ be a sequence of random variables in which any X_i may assume one of the states in $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_n\}$. A Markov chain is a sequence in which the present depends only on the immediate past; that is, the probability distribution of X_t depends only on X_{t-1}, and nothing previously. Formally,

$$P(X_t = \sigma_j \mid X_{t-1} = \sigma_{i_{t-1}}, \cdots, X_2 = \sigma_{i_2}, X_1 = \sigma_{i_1}) = P(X_t = \sigma_j \mid X_{t-1} = \sigma_{i_{t-1}}) \ . \quad (4)$$

For the Markov chains we shall consider, the probabilities p_{ij} of transition from state i to state j are stationary in time. Equation (4) can then be simplified by writing

$$p_{ij} = P(\sigma_i \rightarrow \sigma_j) = P(X_t = \sigma_j \mid X_{t-1} = \sigma_i) \ . \quad (5)$$

Let $p_{ij}^{(n)}$ be the probability of transition from σ_i to σ_j in n (time) steps:

$$p_{ij}^{(n)} = P(X_t = \sigma_j \mid X_{t-n} = \sigma_i) \ . \quad (6)$$

The n-step transition probabilities may be calculated by summing over all possible paths $\sigma_i \sigma_{i_1} \cdots \sigma_{i_{n-1}} \sigma_j$ starting at σ_i. For example, $p_{ij}^{(1)} = p_{ij}$,

$$p_{ij}^{(2)} = \sum_{\nu} p_{i\nu} p_{\nu j} \quad (7)$$

and, by induction,

$$p_{ij}^{(n)} = \sum_{\nu} p_{i\nu} p_{\nu j}^{(n-1)} \ . \quad (8)$$

The transition probabilities may be assembled in a *matrix of transition probabilities* $\mathbf{P} = \{p_{ij}\}$. $\mathbf{P}$ is called a *stochastic* matrix because each $p_{ij} \geq 0$ and each row sums to unity. Let the probabilities $\{\pi_i(n)\}$ represent the probability of being in state σ_i at time n. Assembling these in a row vector $\boldsymbol{\pi}(n)$, we see that

$$\boldsymbol{\pi}(1) = \boldsymbol{\pi}(0)\mathbf{P} \ ,$$

where $\boldsymbol{\pi}(0)$ contains the initial state probabilities. The n-step transition probabilities in equation (8) may now be concisely written as

$$\boldsymbol{\pi}(n) = \boldsymbol{\pi}(0)\mathbf{P}^n \ .$$

It is easy to show that $\mathbf{P}^n$ is also a stochastic matrix.

Markov chains are generally classified in ways that describe the the passage between all possible states. A Markov chain is *irreducible* if every state can be reached (over time and with some positive probability) from every other state. An individual state σ_i is termed *periodic* if the probabilities of recurrence $p_{ii}^{(n)}$ are non-zero only if n is some integral multiple of t; otherwise σ_i is *aperiodic* (aperiodic states are far more common). We restrict ourselves to discussion of chains that have a finite number of possible states. One of the basic theorems for Markov chains states that *if a chain is irreducible, finite, and aperiodic, then the limits*

$$\Pi_j = \lim_{n \to \infty} p_{ij}^{(n)} \tag{9}$$

exist, are unique, and are independent of the initial state σ_i. *Furthermore,*

$$\Pi_j > 0, \qquad \sum \Pi_j = 1 \ , \tag{10}$$

and

$$\Pi_j = \sum_i \Pi_i p_{ij} \ . \tag{11}$$

Conversely, if Π_j *exist satisfying (10) and (11), and if the chain is irreducible and aperiodic, then the* Π_j *are given by (9).* We designate as *ergodic* those Markov chains that exhibit the limiting probabilities (9).

The distribution $\{\Pi_j\}$ is called the *equilibrium* distribution. The term equilibrium has physical significance. In statistical mechanics, an isolated system is said to be in equilibrium if its current state is independent of its initial state; that is, if all memory of its initial configuration has been washed away due the multiplicity of events. For a Markov chain, we say that equilibrium has been attained if the current state-probability vector is independent of the initial state-probability vector. We designate this equilibrium vector as $\mathbf{\Pi}$ and observe the limit

$$\mathbf{\Pi} = \lim_{n \to \infty} \boldsymbol{\pi}(0)\mathbf{P}^n \ .$$

Furthermore, a matrix representation of (11) reveals that $\mathbf{\Pi}$ is an eigenvector of $\mathbf{P}$, with eigenvalue 1:

$$\mathbf{\Pi} = \mathbf{\Pi}\mathbf{P} \ .$$

Finding the equilibrium probabilities may thus be cast as an eigenvalue problem. [See Claerbout (1976) or Howard (1960) for an exposition using Z-transforms.]

METROPOLIS' MARKOV CHAIN

To show that the Metropolis algorithm does indeed converge to a Gibbs distribution, we (1) assume a discrete system, (2) construct the transition-probability matrix that contains the acceptance probabilities, and (3) show that

$$\mathbf{\Pi} = \{\Pi_j\} = \left\{ \frac{e^{-\beta E(\sigma_j)}}{\sum_j e^{-\beta E(\sigma_j)}} \right\} \tag{12}$$

is the equilibrium distribution for $\mathbf{P}$.

Metropolis avoids evaluation of the denominator of (12) by deft use of the ratios

$$\Pi_j \,/\, \Pi_i = e^{-\beta[E(\sigma_j)-E(\sigma_i)]} = e^{-\beta\Delta E} \tag{13}$$

to define his transition probabilities. In my residual statics algorithm, ΔE represents the change in stack power due to a change in the value of a given shot or receiver static (hereafter referred to as parameters). The algorithm sequentially "visits" each parameter and performs a random perturbation of its present value. Perturbations that increase stack power are always accepted, and perturbations that decrease power are accepted with probability $\exp(-\beta\Delta E)$. One iteration is completed after each parameter has undergone an attempted transition. To describe the probabilistic change in the value of the m th parameter, we construct a transition-probability matrix $\mathbf{P}(m)$. We assume that there are a total of M parameters, and that each parameter may be assigned one of N values. Thus there are N^M possible states, or configurations, of the system, and $\mathbf{P}(m)$ is an N^M by N^M matrix. Each row contains only N non-zero elements, since only N new states are directly accessible from any given state. There are M distinct transition-probability matrices to describe the transitions of each parameter. The matrix of transition probabilities that describes the changes from iteration to iteration is given by the product of the M matrices:

$$\mathbf{P} = \mathbf{P}(1)\mathbf{P}(2)\cdots\mathbf{P}(M) \ . \tag{14}$$

To show that $\mathbf{\Pi}$ in equation (12) is the equilibrium vector for $\mathbf{P}$, we will first show that $\mathbf{\Pi}$ is an eigenvector with eigenvalue 1 for each $\mathbf{P}(m)$. Then $\mathbf{\Pi}$ is also an eigenvector with eigenvalue 1 for $\mathbf{P}$. Assuming (but not proving) that $\mathbf{P}$ is irreducible and that all states are aperiodic, we can conclude that the Markov chain is ergodic and that $\{\Pi_j\}$ is its equilibrium distribution.

We now construct $\mathbf{P}(m)$. Assume that the present configuration of values for all parameters is in state σ_i. To update the m th shot or receiver static, the algorithm randomly perturbs this parameter's value to put the system in state σ_j with probability

$p_{ij}^*(m)$. The $p_{ij}^*(m)$ satisfy

$$p_{ij}^*(m) \geq 0\,, \quad \sum_j p_{ij}^*(m) = 1\,, \quad \text{and} \quad p_{ij}^*(m) = p_{ji}^*(m)\,. \tag{15}$$

The $p_{ij}^*(m)$ define a symmetric N^M by N^M transition-probability matrix, with at most N non-zero elements in each row (because the value of only one parameter changes). In the algorithm of paper 1, all non-zero $p_{ij}^*(m)$ are equal, so that random moves are uniformly likely between the minimum and maximum allowable static shifts.

Acceptance or rejection of this random move depends of course on $\exp(-\beta\Delta E) = \Pi_j/\Pi_i$. The probabilities defining the transition of the m th parameter are then

$$p_{ij}(m) = \begin{cases} p_{ij}^*\,\Pi_j/\Pi_i & \Pi_j/\Pi_i < 1 \\ \\ p_{ij}^* & \Pi_j/\Pi_i \geq 1 \end{cases} \tag{16a}$$

for $i \neq j$, and

$$p_{ii}(m) = p_{ii}^*(m) + \sum_j{}' p_{ij}^*(m)\,(1 - \Pi_j/\Pi_i) \tag{16b}$$

for $i = j$, where $\sum'$ is taken only over values of j such that $\Pi_j/\Pi_i < 1$. We see here the familiar acceptance criteria: if energy (negative stack power) decreases, the random move is accepted, otherwise it is accepted only with probability $\exp(-\beta\Delta E)$. After some algebraic manipulation, one may readily show that (Hammersley and Handscomb, 1964)

$$p_{ij} \geq 0\,, \quad \sum_j p_{ij}(m) = 1 \tag{17}$$

and

$$\Pi_j = \sum_i \Pi_i\, p_{ij}(m)\,. \tag{18}$$

By (17), $\mathbf{P}(m) = \{p_{ij}(m)\}$ is a proper transition matrix, and by (18), $\mathbf{\Pi} = \{\Pi_j\}$ is a row eigenvector with eigenvalue 1 for $\mathbf{P}(m)$. By (14), $\mathbf{\Pi}$ is also an eigenvector with eigenvalue 1 for $\mathbf{P}$. If $\mathbf{P}$ is irreducible, then $\mathbf{\Pi}$ is its equilibrium vector. Each $\mathbf{P}(m)$ is not irreducible, because repeated transitions of the same parameter would simply stay among N states. However, since each $\mathbf{P}(m)$ contains a non-zero transition probability for each possible value of the m th parameter, we may readily assume that $\mathbf{P}$ is irreducible. We also assume that all states are aperiodic. Thus, by the limit theorem in the previous section, the chain is ergodic, and its equilibrium distribution is the Gibbs distribution (12).

ALTERNATIVES

Needless to say, the Metropolis algorithm is slow. Thirty years of use have led to a few good ideas in the physics literature, however, and the prospects for computational speedup are fortunately far from dim.

Any thoughts on efficiency should properly focus on obvious weaknesses. This is clearly a problem dependent area, so I shall abandon most attempts at generality to get down to the central issue: residual statics estimation. The basic problem facing the statics algorithm I discussed in paper 1 is its slow convergence upon settling in a minimum (i.e., equilibrium at low temperature). This is due to the way the algorithm chooses its random moves - for each visitation to a shot or receiver static, a random guess is made, which is then accepted or rejected. The random guesses are made with the probabilities p_{ij}^*. As I mentioned earlier, the implementation in paper 1 was such that the random guesses were uniformly likely between some predetermined maximum and minimum value for each static. A typical application might search for, say, statics within ± 200 msec. Coarse discretization at 8 msec. intervals yields 51 possible values from which to draw a random guess. Near a minimum only a very few of these 51 candidates is an acceptable guess, so it is clear that the present set of random moves leaves something to be desired.

A simple view

One simple alternative is to choose a more intelligent set of p_{ij}^*. We might suppose that the range of candidates for random guessing should be narrowed near a minimum. For example, instead of always drawing guesses from the range $\pm\, l$, where l is the maximum expected static, we might instead choose from $s \pm l'$, where s is the parameter's present value and l' is some dynamically adjustable parameter that gets smaller so as to maintain a stable acceptance/rejection ratio at around, say, 1. There is a pitfall here, however. If we want to be able to escape a local minimum, l' must be at least twice as wide as the dominant period of the data to insure that a skipped cycle can be corrected. Thus l' may still be unacceptably large.

The heat bath method

These ideas do, however, indicate a more promising approach, in which the entire set of the most likely candidate moves for a given parameter are known before any random guessing is made. Consider the following thermodynamic analogy. Imagine all the parameters except x_m to form a constant heat bath around x_m. Over time, x_m would exhibit thermally induced fluctuations and eventually settle into a local equilibrium

within its heat bath. If x_m can assume any of N values, then this one-parameter system can assume only N energy levels, which are distributed in proportion to $\exp[-\beta E(x_m)]$. To simulate this algorithmically, we would choose new values for x_m with probability proportional to $\exp[-\beta E(x_m)]$. This allows us to concentrate on the most likely candidate moves, and forget about the problematic p_{ij}^* entirely.

The above technique is called the *heat bath* method (Rebbi, 1984; Creutz, 1984). Choosing new states with probability proportional to $\exp[-\beta E(x_m)]$ means making transitions with the matrix

$$\mathbf{P}'(m) = \lim_{n\to\infty} \mathbf{P}^n(m) \quad . \tag{19}$$

Recall that $\mathbf{P}(m)$ is N^M by N^M and that it contains N non-zero elements per row. These N elements may be grouped into N^{M-1} submatrices $\mathbf{Q}_i$ that each contain N rows and N columns. Each $\mathbf{Q}_i$ describes the transition from state σ_i to σ_j for changes in which only parameter x_m changes. The $\mathbf{Q}_i$ are all stochastic matrices that may be analyzed independently of the others. Each $\mathbf{Q}_i$ is obviously irreducible because all entries are non-zero. By the analysis of the previous section, each $\mathbf{Q}_i$ has equilibrium probabilities proportional to $\exp[-\beta E(x_m)]$. One might imagine an algorithm that continually updates x_m, randomly perturbing and accepting or rejecting according to the Metropolis rules many times before going on to x_{m+1}. Eventually the initial value of x_m would be forgotten and a local equilibrium would be reached. This is equivalent to choosing and accepting random guesses for x_m with the transition matrix $\mathbf{P}'(m)$. We now construct

$$\mathbf{P}' = \mathbf{P}'(1)\mathbf{P}'(2)\cdots\mathbf{P}'(M) \quad . \tag{20}$$

Since (12) is an eigenvector with eigenvalue 1 for $\mathbf{P}(m)$, the same is true for each $\mathbf{P}'(m)$, and also for $\mathbf{P}'$. Because $\mathbf{P}$ is irreducible and aperiodic, so is $\mathbf{P}'$. Thus the Gibbs distribution is also the equilibrium distribution for $\mathbf{P}'$, and the heat bath algorithm produces results equivalent to the Metropolis algorithm.

A statics algorithm using the heat bath method would be the following. For each shot and receiver static x_m, the algorithm calculates stack power ϕ_m as a function of static shift s_j for each of the N possible shifts. It then chooses a new value for x_m by randomly sampling from the probability distribution

$$P(x_m = s_j) = \frac{e^{-\beta\phi_m(s_j)}}{\sum_{j=1}^{N} e^{-\beta\phi_m(s_j)}} \quad . \tag{21}$$

Equation (21) is essentially an exponentiated crosscorrelation function with unit area. We see this by noting that the crosscorrelation function of infinite time sequences w_t and y_t differs from a stack power function by the total power in w_t and y_t, a constant:

$$\phi(s_j) = \sum_t (w_t + y_{t+s_j})^2 = \sum_t w_t \; y_{t+s_j} + \sum_t {w_t}^2 + {y_t}^2 \; . \tag{22}$$

For finite sequences w_t and y_t, the difference between power and crosscorrelation is only approximately constant, but for computational purposes they may usually be taken to be equal.

Each random heat bath move will of course require about N times the computation that an equivalent Metropolis move requires, so it may appear that we've gained nothing. For my residual statics algorithm, however, this is not the case. I currently do my energy function calculations in an (FPS 120B) array processor. It turns out that crosscorrelation over a fair number of lags is roughly equivalent in AP time to a simple one parameter stack power calculation. Thus the information in (21) may be gained for little additional cost. I've recently implemented the heat bath method, and my early tests appear to converge considerably faster than equivalent Metropolis runs. (See *Residual statics estimation by simulated annealing: field data results,* this report.)

No repeated moves

Once we go to the trouble to compute (21), it may still be that descent into a minimum is sufficiently deep enough so that repeated visits to parameters produce changes too slowly. Bortz et al. (1975) describe a method that continuously updates a table that indexes each parameter according to its probability of changing. Each step in the Bortz algorithm is a random selection from this table, and always produces a change. Unfortunately, updating this table after each move requires evaluation of (21) not only for the parameter undergoing the change but also for all of its neighbors. For statics, this entails reevaluating (21) for all shots and receivers within a cablelength, typically on the order of 100 times. Thus we can expect that the Bortz algorithm would only be useful when the acceptance/rejection ratio falls below about .01. I foresee little utility for this approach in residual statics estimation, but it may be helpful elsewhere.

CONCLUSIONS

The Monte Carlo techniques that can be used for simulated annealing are methods that produce random samples from a Gibbs probability distribution. Each Monte Carlo algorithm described here may be cast as a Markov chain. The equilibrium distribution of these Markov chains is the Gibbs distribution. For a sufficiently low temperature,

reaching equilibrium is equivalent to locating the energy minimum (i.e., optimization). Thus studies of Markov equilibrium properties are essential to studies of optimization by simulated annealing.

Of the several alternatives to the basic Metropolis algorithm, the heat bath method appears the most promising for residual statics estimation. Because this algorithm chooses random moves from a local equilibrium distribution, its approach to equilibrium, and therefore its convergence to the global minimum, is expected to be faster than the simpler Metropolis technique.

REFERENCES

Bortz, A.B., Kalos, M.H., and Lebowitz, J.L., 1975, A new algorithm for Monte Carlo Simulation of Ising spin systems: Journal of Computational Physics, v. 17, p. 10-18.

Claerbout, J.F., 1976, Fundamentals of geophysical data processing: McGraw-Hill Book Co., N.Y., 274 p.

Creutz, M., 1984, Quarks, gluons, and lattices: Cambridge University Press, N.Y., 169 p.

Feller, W., 1968, An introduction to probability theory and its applications: 3rd ed., v. 1, John Wiley and Sons, New York, 509 p.

Fosdick, L.D., 1963, Monte Carlo computations on the Ising lattice: Methods in Computational Physics, v. 1, p. 245-280.

Hammersley, J.M. and Handscomb, D.C., 1964, Monte Carlo methods: London, Chapman and Hall, 178 p.

Howard, R.A., 1960, Dynamic programming and Markov processes: MIT Press, Cambridge, 136 p.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., 1953, Equation of state calculations by fast computing machines: Journal of Chemical Physics, v. 21, p. 1087-1092.

Rebbi, C., 1984, Monte Carlo calculations in lattice gauge theories: in Applications of the Monte Carlo method, K. Binder, ed., p. 277-298.

Rothman, D.H., 1984, Nonlinear inversion, simulated annealing, and residual statics estimation: SEP Report 41.