CHAPTER 3

Residual statics estimation by simulated annealing: theory

The estimation of large statics poses several diverse and interesting problems. From a geophysical point of view, the problem is to accurately correlate related seismograms. As I indicated in the previous chapter, the constraint of surface-consistency leads to a computational problem that extends beyond the simple crosscorrelation of one-dimensional seismograms. The actual problem is a multidimensional optimization problem, in which each shot and receiver static is a parameter to be estimated.

Because the number of parameters to be estimated is always large, no statics estimation problem is trivial. However, when statics are small (or a good first guess can be made), conventional techniques of optimization can be employed with considerable success. The case in which there are large statics (coupled with the lack of a good first guess) deserves special attention. The large statics problem not only appears nontrivial, but it also appears computationally *intractable.* Thus the estimation of large statics is one of the more subtle problems in geophysical data processing.

In this chapter, I introduce a mathematical model for surface-consistent statics and then describe how and why the estimation of large statics poses an apparently intractable optimization problem. Using a heuristic argument, I then introduce the technique of simulated annealing, describe how it can overcome the apparently intractable nature of the problem, and show two ways in which it can be adapted for statics estimation. The statics algorithm I introduce is a Monte Carlo procedure that searches for a solution in a random, yet controlled, manner. The chapter is concluded with a formulation of the algorithm as a Markov chain.

3.1 A MODEL FOR SURFACE-CONSISTENT STATICS

In this section, I present a mathematical model for surface-consistent statics. This model shows that statics estimation is inherently a nonlinear, rather than a linear, inverse problem. Two previous solutions to the statics problem are then reviewed. The first method is based on linearization and least-squares inversion. The second technique incorporates a more direct approach to optimization, and serves as the basis for the algorithm I develop in the remainder of the chapter.

3.1.1 The underlying nonlinearity

The surface-consistent model, described in §2.3, states that the time delay of each trace is the sum of a shot static and a receiver static. Denote by $d_{ij}(t)$ the trace recorded at receiver j after shot i is fired at time $t = 0$. Designate the shot static by s_i and the receiver static by r_j. Let $h_{ij}(t)$ be the trace that would have been recorded if the near-surface were homogeneous; that is, if $s_i = r_j = 0$. Define $G_{ij} = G(s_i, r_j)$ to be an operator that delays the unknown trace $h_{ij}(t)$ by the amount $s_i + r_j$. Then, for each seismogram recorded in a seismic survey, the equation

$$d_{ij}(t) = G_{ij}[\, h_{ij}(t)\,] \tag{3.1}$$

defines the relationship between the unknown, statics-corrected data $h_{ij}(t)$ and the observed data $d_{ij}(t)$. In a typical residual statics estimation problem, there are thousands of simultaneous equations of the form (3.1). Beginning from this underlying model, the solution of a statics problem either provides $h_{ij}(t)$ directly, or it provides a solution for s_i and r_j, from which the $h_{ij}(t)$ can then be obtained via the simple relation

$$h_{ij}(t) = d_{ij}(t + s_i + r_j) \; .$$

Because s_i and r_j are unknown, G_{ij} is also unknown. Thus, although G_{ij} is a linear operator, its inverse is unknown; therefore equation (3.1) cannot be solved for $h_{ij}(t)$ by linear inversion. Equation (3.1) thus poses a *nonlinear* inverse problem.

3.1.2 Two previous approaches

A popular method for the solution of the system of equations given by (3.1) is to derive a similar set of equations from (3.1) that can be solved by linear inversion (Taner et al., 1974; Wiggins et al., 1976). This linearization is performed by first estimating a time delay t_{ij} of each trace $d_{ij}(t)$ relative to a reference trace $f_{ij}(t)$; t_{ij} is obtained by choosing the lag that yields the maximum of the crosscorrelation between $d_{ij}(t)$ and $f_{ij}(t)$. Then, by invoking surface-consistency, one can write

$$t_{ij} \approx s_i + r_j \quad . \tag{3.2}$$

[Taner et al. (1974) and Wiggins et al. (1976) also include *subsurface-consistent* terms in this equation—see §3.4.3]. The system of equations given by (3.2) is then solved for each s_i and r_j using the method of least squares.

Another technique for the solution of equation (3.1) has been recently demonstrated by Ronen and Claerbout (1985). Their technique is based on a fundamental observation: the best estimate of shot and receiver statics is that estimate which maximizes the power (sum of the squared amplitudes) in the CMP-stacked section. To express this mathematically, designate the unknown shot statics by $\mathbf{s} = \{s_i\}$ and the unknown receiver statics by $\mathbf{r} = \{r_j\}$. Denote the negative stack power by $E(\mathbf{s},\mathbf{r})$. Let $d_{yh}(t)$ signify the seismic traces after (approximate) NMO correction has been made and sorting to midpoint (y) and offset (h) coordinates has been done. Then, to estimate the best set of shot and receiver statics, seek the solution of the optimization problem

$$\min_{[\mathbf{s},\mathbf{r}]} E(\mathbf{s},\mathbf{r})$$

where

$$E(\mathbf{s},\mathbf{r}) = -\sum_y \sum_t \left[\sum_h d_{yh}(t + s_{i(y,h)} + r_{j(y,h)}) \right]^2 \quad . \tag{3.3}$$

The subscripts i and j are uniquely determined by y and h. To perform the minimization, Ronen and Claerbout (1985) iteratively update a statics model. They estimate each shot and receiver static in sequence by choosing the lag that yields the maximum of a surface-consistently averaged crosscorrelation function.

Both of the aforementioned techniques are successful for a wide variety of statics problems. Both, however, are limited in one key respect. Because these methods obtain estimates of time delays by picking the maxima of crosscorrelation functions, they risk making gross errors, or cycle-skips, when near-surface anomalies are large. In principle, this deficiency can be accommodated by first guessing the statics solution, shifting the traces accordingly, and then using this new set of data as the input to one of these methods. If the first guess of the shot and receiver statics is close to the "true" solution, then the cycle-skipping problem is considerably diminished. If a good first guess cannot be made, cycle-skipping remains a problem.

In optimization theory, $E(\mathbf{s},\mathbf{r})$ is called an *objective function.* In residual statics estimation (and many other geophysical inverse problems) objective functions can contain many minima. The deepest minimum of the objective function is called the *global minimum.* All minima other than the global minimum are called *local minima.* In this thesis, I use the stack-power criterion of Ronen and Claerbout to find the statics that yield the best stack; these statics determine the global minimum of (3.3). As I discuss in the next section, large statics can produce two problems that inhibit global optimization. First, many local minima can exist. Second, there may be insufficient information concerning the approximate location of the global minimum. The combination of these two problems produces an apparently intractable optimization problem.

3.2 STATICS ESTIMATION AND COMPUTATIONAL INTRACTABILITY

Unfortunately, a good first guess cannot be obtained for some statics problems. This is typically the case just when a good first guess is needed the most: when near-surface anomalies are large. Under that condition, the data may reveal little, if any, structure; thus visual interpretation of the data to obtain a first guess might not be feasible.

What can be done, then, in the absence of a good first guess? One might suggest an exhaustive search for the best estimate of shot and receiver statics. The best estimate would be defined simply as that choice of **s** and **r** which globally minimizes $E(\mathbf{s},\mathbf{r})$.

An exhaustive search is computationally impossible, however, because the number of possible solutions to a statics problem is enormous. Suppose that each shot and receiver static may assume only N discrete values and that there are M shots and receivers. Then there are N^M possible solutions to the problem. M is typically several hundred or more, and N is on the order of 50 when near-surface anomalies are large.

The impossibility of an exhaustive search is no cause for dismay by itself. A related and equally important issue is the shape of the objective function $E(\mathbf{s},\mathbf{r})$. For simplicity, consider the components of **s** and **r** to be the components of an M-vector $\mathbf{x} = [\mathbf{s}, \mathbf{r}]$. A schematic representation of the function $E(\mathbf{x})$ is presented in Figure 3.1. If $E(\mathbf{x})$ were to contain only one minimum, global minimization would be a straightforward task: an algorithm would need only to descend along the gradient of $E(\mathbf{x})$. Because seismic data are bandlimited and statics are large, however, $E(\mathbf{x})$ contains many minima of different depths. Suppose that the expected statics span time delays

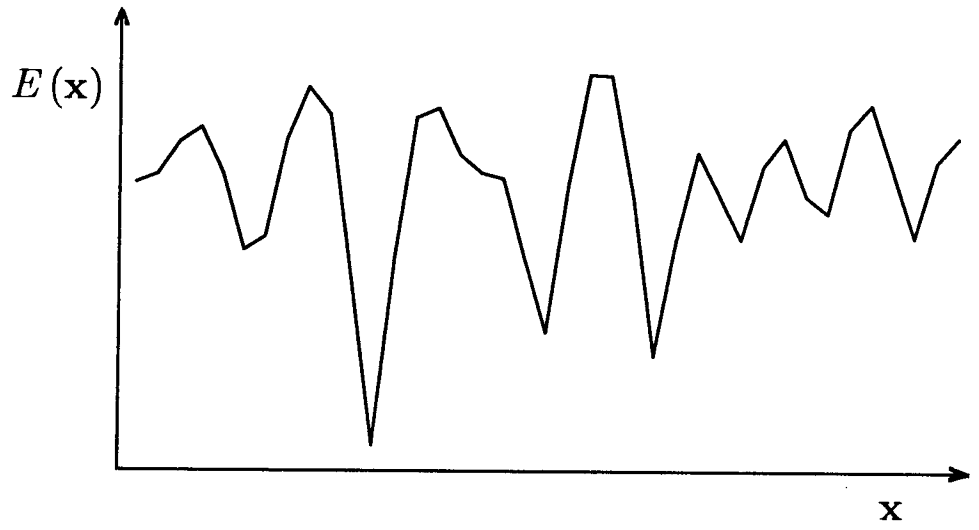

FIG. 3.1. A schematic representation of the objective function $E(\mathbf{x})$, where $\mathbf{x} = [\mathbf{s}, \mathbf{r}]$, a vector of length M. A true representation of $E(\mathbf{x})$ would be plotted in M-dimensional space. The goal of statics estimation is to find the value of $\mathbf{x}$ that globally minimizes $E(\mathbf{x})$.

that are as much as L times the dominant period of the data. Roughly speaking, then, the projection of $E(\mathbf{x})$ along any axis in $\mathbf{x}$-space will contain L minima. The precise number of minima in the M-dimensional space cannot be known, but it is reasonable to conjecture that the number of minima grows exponentially with M, as does the number of possible solutions.

Despite the large number of local minima, global minimization of $E(\mathbf{x})$ can still be a straightforward problem if the initial value of $\mathbf{x}$ is in the region of the global minimum. The initial value of $\mathbf{x}$ is "in the region" if no uphill climbing is needed for the global minimum to be found. This condition typically exists when statics are small; then the usual initial value $\mathbf{x} = 0$ is close enough. If $\mathbf{x} = 0$ is not sufficiently close to the global minimum, one can try to guess a value of $\mathbf{x}$ that is. If a close first guess cannot be obtained, however, global minimization of $E(\mathbf{x})$ appears impossible. As I mentioned above, an exhaustive search cannot be performed—there are simply too many possible solutions. Thus, if statics are large and a good first guess cannot be obtained, the problem appears computationally intractable.

Computationally intractable optimization problems are the subject of much research among theorists who study computer algorithms (Garey and Johnson, 1979). Precise definitions of intractability exist; here, a computationally intractable optimization problem is defined broadly as a problem in which the number of possible solutions grows exponentially with the number of parameters, and in which exhaustive search is the only way (in principle) to find the optimal solution.

That a problem is computationally intractable does not mean that it cannot be satisfactorily solved; intractability implies only that there is no way to be sure that the solution achieved is globally optimal. The usual approach to computationally intractable problems, in geophysics and elsewhere, is to design a heuristic algorithm. Heuristic methods are usually problem-specific, but some general methods exist. One general heuristic technique is called *iterative improvement* (Lin, 1975). The method of statics estimation described by Ronen and Claerbout (1985) is one form of iterative improvement: each parameter (a shot or receiver static) is individually updated in sequence until no better value for any parameter exists. Each time a parameter is updated, the value that most decreases the objective function is chosen. In general, when objective functions contain many minima and no good first guess can be made, the result of iterative improvement is usually a local rather than a global minimum. Thus, in these cases it is customary to employ iterative improvement many times beginning with different (often random) initial guesses; the best result is then taken as the final solution. When the number of local minima is large, however, one cannot expect that random initial guesses followed by descent to the nearest minimum would be likely to find (or even approximate) the global minimum; this is the case in the estimation of large statics.

The principal deficiency of iterative improvement is its strong dependence upon an initial guess. This dependence arises from the method's ability to make only downhill moves; it thereby always arrives at a nearby minimum. For the estimation of large statics, a technique with less dependence upon the first guess is needed. To solve the cycle-skipping problem, a statics estimation algorithm must be able to search in *both* the downhill *and* the uphill direction.

3.3 SIMULATED ANNEALING

Recently, a new heuristic algorithm called *simulated annealing* was introduced for the approximate global optimization of objective functions that contain many local minima (Kirkpatrick et al., 1983). Simulated annealing is similar to iterative improvement in that it updates individual parameters in sequence. Unlike iterative improvement, however, simulated annealing can choose new values that either decrease or increase the objective function. Because it has this ability to make uphill moves, simulated annealing is not dependent upon an initial guess.

Simulated annealing is a Monte Carlo optimization procedure that mimics the physical process of annealing. Annealing is the way in which crystals are grown: a substance is first melted, and then cooled very slowly until a crystal is formed. The rate of cooling is important, because a non-crystalline, metastable glass can form if cooling is too rapid. Kirkpatrick et al. viewed the growth of a crystal as analogous to finding the global minimum in optimization, and the development of a glass as the analog to wrongly selecting a local minimum. Their principal application was to the combinatorial optimization problems that arise in the physical design of computers.

In conceiving the idea of simulated annealing, Kirkpatrick et al. made an analogy between the energy in a physical system and the objective function in an optimization problem. Physical systems in thermal equilibrium exhibit fluctuations in energy. By introducing a control parameter analogous to absolute temperature, Kirkpatrick et al. were able to perturb parameters in a way that simulates the fluctuations of a physical system in thermal equilibrium. Controlled uphill are therefore possible; thus local minima can be escaped.

Because simulated annealing is firmly rooted in an analogy with statistical mechanics, I continue my discussion of the algorithm by first introducing some pertinent concepts from statistical mechanics. I then discuss two methods for the application of simulated annealing to residual statics estimation. The first approach, based on the algorithm of Metropolis et al. (1953), closely follows the technique described by Kirkpatrick et al. (1983). The second method is a more efficient version of the first approach.

3.3.1 Statistical mechanics

Statistical mechanics provides methods for the study of the behavior of macroscopic systems (e.g., solids, liquids, or gases) composed of many microscopic constituents (e.g., atoms or molecules)†. The results of statistical mechanics apply only to a system in equilibrium; that is, a system in which enough time has elapsed such that its behavior no longer exhibits the influence of its initial state. Whereas a system of just a few particles can often be adequately described by the equations that govern mechanical interactions, a system of many particles is too complex for this approach to yield any useful predictions of the system's behavior. When the number of particles is large, however, statistical mechanics can be employed to derive important average properties of the system.

A fundamental result of statistical mechanics gives the probability of a system being in a given state if the system is in thermal equilibrium. The state of a system is defined as the configuration of its microscopic components. Designate this configuration as a random variable $\mathbf{X} = \{X_1, X_2, \cdots, X_M\}$, and denote the realization of $\mathbf{X}$ by $\mathbf{x} = \{x_1, x_2, \cdots, x_M\}$. The probability that the system is in state $\mathbf{x}$ is given by the joint probability distribution

$$P(\mathbf{X}{=}\mathbf{x}) = \frac{1}{Z} e^{\frac{-E(\mathbf{x})}{k_B T}} \tag{3.4}$$

where $E(\mathbf{x})$ is the system's energy, k_B is Boltzmann's constant, T is absolute temperature, and Z is the normalizing constant

$$Z = \sum_{\mathbf{x}} e^{\frac{-E(\mathbf{x})}{k_B T}} .$$

Z is called the *partition function.* Equation (3.4) is a *Gibbs distribution,* also referred to as a canonical distribution.

For a system in thermal equilibrium, the Gibbs distribution describes the expected fluctuations of the system's configuration. Of central importance here is the fact that fluctuations can either decrease or increase energy. The low energy states are favored, however, especially when the system is at low temperature: as T becomes small, the Gibbs distribution assigns progressively greater probability to low-energy configurations. In the limit as $T \rightarrow 0$, a system in equilibrium assumes its minimal energy configuration, or ground state. This ground state often corresponds to a

†For an introduction to statistical mechanics, see Reif (1965).

perfectly ordered, crystalline structure. To reach this ordered state, the system must be cooled slowly, so that equilibrium can be attained at each new temperature. Although the final state is highly ordered, a physical system passes through many disordered states at non-zero temperature before it reaches its ground state.

3.3.2 Statics estimation using the Metropolis algorithm

The average behavior of a physical system in thermal equilibrium can be simulated using the Metropolis algorithm (Metropolis et al., 1953). The simulation is performed by randomly generating states with probabilities given by the Gibbs distribution. Low-energy states are the most probable configurations to be generated by a Metropolis simulation. High-energy states are the least probable configurations, but they are always possible.

The Metropolis algorithm proceeds as follows. A system is parameterized by M model parameters. In physical simulations, a model parameter can be, for example, a magnetic spin or molecular position. In statics estimation the model parameters are the shot and receiver statics. For each individual model parameter X_m, a random perturbation of its current value is performed, and the change in energy, ΔE, is computed. If $\Delta E \leq 0$ (i.e., if energy decreases), the perturbation is accepted. If ΔE is positive then the perturbation is accepted with probability

$$P(\Delta E) = e^{\frac{-\Delta E}{T}} .$$

(Here and in the remainder of the thesis k_B is taken to be 1.) This conditional acceptance is easily implemented by choosing a random number α uniformly distributed between 0 and 1. If $\alpha \leq P(\Delta E)$ then the perturbation is accepted; otherwise the existing value for the parameter is retained. Each time a perturbation of a particular parameter is accepted, that parameter's value is updated immediately, and all future energy calculations are based on this value until it is updated again. Random perturbations according to these rules eventually cause the system to reach equilibrium, in which configurations $\mathbf{x} = \{x_m\}$ are generated with a Gibbs probability distribution. See Hammersley and Handscomb (1964) for an elementary proof that the equilibrium distribution is a Gibbs distribution.

The optimization technique of Kirkpatrick et al. slowly lowers the temperature T during execution of the Metropolis algorithm. At high T, uphill moves are as likely as downhill moves; as T decreases, however, uphill moves become increasingly unlikely. If the system is cooled sufficiently slowly and equilibrium conditions are maintained, the model parameters eventually converge to a (ground) state of minimum energy. For a detailed derivation of convergence properties with respect to changes in T, see Geman and Geman (1984).

When the Metropolis algorithm is employed to estimate statics, the algorithm accepts not only random guesses of statics that lead to increased stack power, but also some random guesses that decrease stack power. The computation of the change in stack power is a local calculation that does not require restacking all CMP gathers; only those CMP gathers containing a trace associated with that shot or receiver static need be restacked. Thus each shot static s_i directly influences the stack power of only a subset Y_{s_i} of all midpoints y. Likewise each receiver static directly affects only a subset Y_{r_j}. The contribution to stack power due to s_i is given by

$$\phi_{s_i}(s_i) = \sum_{y \in Y_{s_i}} \sum_t \left[\sum_h d_{yh}(t + s_{i(y,h)} + r_{j(y,h)}) \right]^2 , \qquad (3.5)$$

where all other shot statics s_k, $k \neq i$, and all r_j are fixed. Similarly, the contribution to stack power due to r_j is given by

$$\phi_{r_j}(r_j) = \sum_{y \in Y_{r_j}} \sum_t \left[\sum_h d_{yh}(t + s_{i(y,h)} + r_{j(y,h)}) \right]^2 , \qquad (3.6)$$

where now r_k, $k \neq j$, and all s_i are fixed. The computations required to obtain $\phi_{s_i}(s_i)$ or $\phi_{r_j}(r_j)$ are considerably simpler than the computations that would be required if all shot and receiver statics were updated simultaneously. The ease with which these local calculations are performed is central to the success of the Metropolis algorithm.

In summary, the Metropolis algorithm sequentially cycles through a two-step scheme for each parameter: (1) a random guess for the static shift is made; then (2) a decision is made to either accept or reject this new guess. In Chapter 4, I show a result of using the Metropolis algorithm to solve for statics in synthetic data. The results are positive, but in my tests it became clear that many computations are wasted in the evaluation of random guesses that have a high probability of being rejected. My implementation of the Metropolis algorithm makes random guesses that are uniformly distributed between some predetermined maximum and minimum value for each static. (Other schemes are possible.) For the synthetic data example, this range spanned 21 values. The field data example in Chapter 4, however, required a wider range which proved to be too large: too many computations were wasted in the evaluation of unacceptable guesses. I therefore developed an alternative technique, described below.

3.3.3 Transforming correlation to probability: random moves in one step

To avoid low acceptance-to-rejection ratios, the second method computes the relative probabilities of acceptance for each possible move before any random guesses are made. Instead of making random guesses that are then accepted or rejected, the second method simply produces weighted guesses that are always accepted. In contrast to the two-step Metropolis method, this is a *one-step* technique.

The one-step method sequentially "visits" each shot and receiver static X_m, as does the Metropolis algorithm. Then, instead of calculating the stack power for a single random guess, the one-step method calculates the stack power for each of N possible shifts. To determine the new value for X_m, the method then draws a random number from a probability distribution derived from these stack-power calculations.

Denote the shot and receiver statics by a single vector-valued random variable $\mathbf{X} = \{X_m\}$, $m = 1, \cdots, M$. Any shot or receiver static X_m can assume any of N values τ_p, $p = 1, \cdots, N$. The stack-power function for the mth shot or receiver static is denoted by $q_m(\tau_p)$ [which is defined precisely in equation (3.8) below.] To choose a new value for X_m, the one-step statics algorithm draws a random number from the probability distribution

$$P(X_m = \tau_p) = \frac{\exp\{\, q_m(\tau_p)/T \,\}}{\sum\limits_{p=1}^{N} \exp\{\, q_m(\tau_p)/T \,\}} \,. \tag{3.7}$$

This new value for X_m is always retained and the stack is always updated, *regardless* of the change in stack power. As the power $q_m(\tau_p)$ increases, the probability that the algorithm chooses $X_m = \tau_p$ also increases. Choices that decrease power, however, are also possible; they are just less likely.

The results in the Appendix (discussed in §3.5) show that the one-step technique simulates thermal equilibrium, as does the Metropolis algorithm; thus the one-step technique is fundamentally equivalent to the Metropolis algorithm. The similarity of the two techniques can be seen by considering the effects of each method on a single parameter. If one were to employ the two-step Metropolis rules many times to repeatedly update X_m, then the initial value of X_m would eventually be "forgotten," and the frequency of occurrence of each possible τ_p would be proportional to the probability distribution (3.7), which is also the distribution from which the one-step technique draws random moves. Repeated application of the Metropolis rules to X_m simulates the physical effects of a heat bath surrounding X_m. In this heat bath, the "states" of X_m (e.g., magnetic spins) would be distributed with equilibrium probabilities proportional to $\exp\{-E(\tau_p)/T\}$, where E is energy and τ_p denotes some physical parameter of interest. Thus, sampling from the probability distribution (3.7) is equivalent to sampling from a *local equilibrium* distribution. Sampling from these local equilibrium distributions for each X_m considerably aids the approach to *global equilibrium,* which must be approximated at a low temperature if simulated annealing is to yield a reasonable result.

In the literature of Monte Carlo simulations in statistical physics, the one-step technique is known as the *heat-bath method* (Rebbi, 1984; Creutz, 1984). Recently, Geman and Geman (1984) adapted this method in their application of simulated annealing to image restoration.

The calculation of the probability distribution (3.7) is not as difficult as it may seem. As Ronen and Claerbout (1985) have shown, crosscorrelation functions can be substituted for explicit power computations when statics corrections are estimated by stack-power maximization. Therefore, for computational convenience, I compute $q_m(\tau_p)$ by performing crosscorrelations as follows. Designate the stacked trace formed by the sum of each trace in CMP gather k by $y_k(t)$. Reference traces are the partially stacked traces

$$y_k^{ij} \equiv y_k(t) - d_{ij}(t) , \qquad k = (i + j)/2 ,$$

where $d_{ij}(t)$ is now the NMO-corrected trace associated with the ith shot and jth receiver. The trace $y_k^{ij}(t)$ is the sum of all traces in CMP gather k except trace $d_{ij}(t)$. Suppose that the mth element of the parameter vector is the shot static corresponding to shot i (or the receiver static corresponding to receiver j). To compute $q_m(\tau_p)$, crosscorrelate each trace in the shot (or receiver) gather against the partial stack of the CMP gather in which that trace is a member. Then sum each of these crosscorrelation functions to obtain a surface-consistent average, and normalize the result to compensate for any spatial variations in amplitude. Written formally, this surface-consistently averaged, normalized crosscorrelation function is

$$q_m(\tau_p) = A^{-1/2} \sum_{i,j \in C_m} R_{ij}(\tau_p) , \tag{3.8}$$

where $R_{ij}(\tau_p)$ is the crosscorrelation function given by

$$R_{ij}(\tau_p) = \sum_t y_k^{ij}(t)\, d_{ij}(t + \tau_p) , \tag{3.9}$$

A is the normalizing constant

$$A = \left(\sum_{i,j \in C_m} \sum_t [d_{ij}(t)]^2 \right) \left(\sum_{i,j \in C_m} \sum_t [y_k^{ij}(t)]^2 \right) ,$$

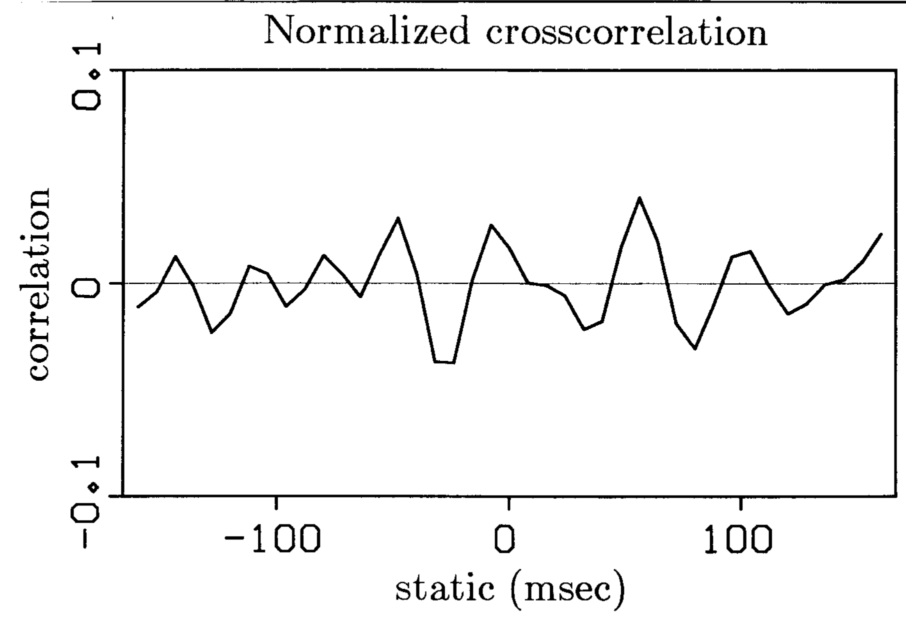

FIG. 3.2a. A normalized crosscorrelation function.

and C_m denotes all shot-receiver (i, j) pairs associated with the traces in shot gather i (or receiver gather j). In my implementation, I perform the summation in equation (3.8) implicitly by using a method described by Ronen and Claerbout (1985): a single unstacked trace formed by the concatenation of each $d_{ij}(t)$ is crosscorrelated against a single partially stacked trace formed by the concatenation of each $y_k^{ij}(t)$. The resulting crosscorrelation function $q_m(\tau_p)$ is then inserted into equation (3.7). The probability distribution (3.7) can then be viewed as simply a normalized crosscorrelation function that is scaled by T, exponentiated, and then scaled again so that it has unit area.

The temperature, T, determines the relative probabilities of each possible static shift; this relationship is illustrated by the sequence of functions in Figures 3.2a-c. Figure 3.2a is a normalized crosscorrelation function computed from the Wyoming data introduced in Chapter 2. Conventional statics algorithms would usually choose the lag (56 ms) that yields the maximum correlation for use in some estimation scheme. Because this lag might correspond to a skipped cycle, however, the one-step method considers all possible lags, and favors each lag by the weights given in equation (3.7). Figure 3.2b shows the transformation of the crosscorrelation function of Figure 3.2a into the probability distribution (3.7), with $T = .044$. The modes (peaks) of the resulting probability distribution correspond to the peaks of the crosscorrelation function. Figure 3.2c depicts the effects of a lower value for T, now .013. The greatest

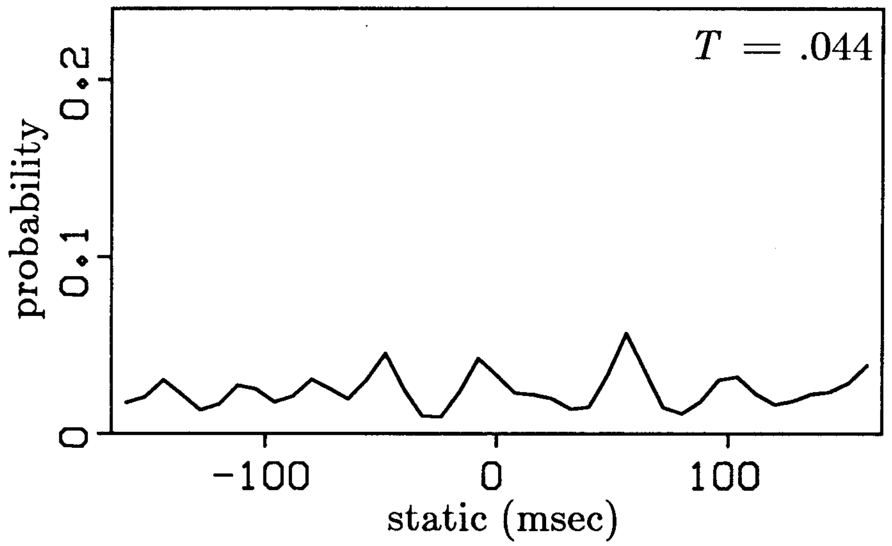

FIG. 3.2b. Transformation of the crosscorrelation function in Figure 3.2a into a probability distribution using equation (3.7), with q_m taken to be crosscorrelation, and $T = .044$. Note the general correspondence of the peaks and troughs in Figures 3.2a and 3.2b.

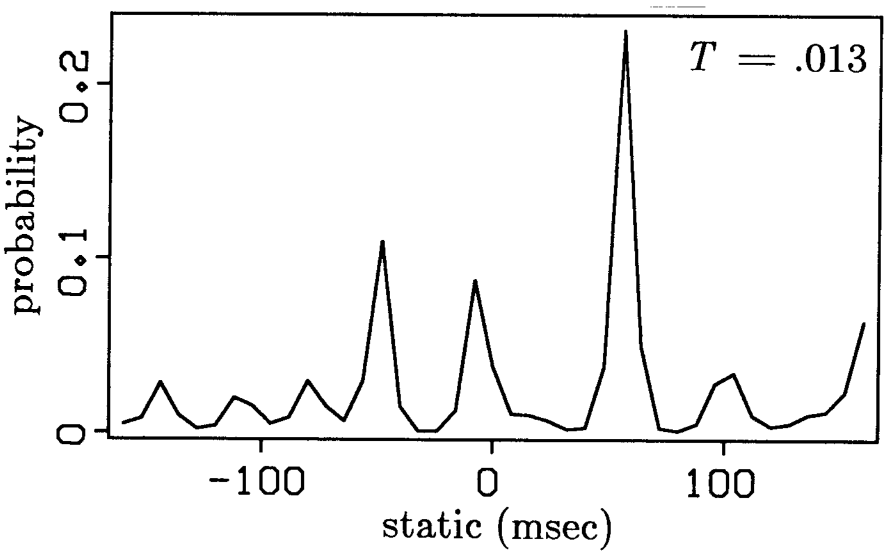

FIG. 3.2c. Transformation of the crosscorrelation function in Figure 3.2a into a probability distribution using equation (3.7), with q_m taken to be crosscorrelation, and $T = .013$. Compared to Figure 3.2b, the tallest peak is now strongly accentuated relative to the other peaks; thus the lag (56 ms) corresponding to that peak is strongly favored when random samples are drawn from this distribution.

peak is now strongly accentuated relative to the others. As T goes to zero, this peak becomes a spike of unit height. Thus the zero temperature, or "quenching" algorithm, always picks the greatest peak, and would always arrive at a nearby local minimum of the objective function.

The spikiness of the probability distribution (3.7) is controlled not only by T, but also by the quality of the stacked traces against which crosscorrelation is performed. Thus, in order for the algorithm to converge to an approximation of the best possible stack, it is important that T be chosen to be high enough to allow the stack to change significantly, yet low enough so that the probability distribution (3.7) is sufficiently spiky when the correct solution is nearly achieved. The greatest advantages of the one-step method are incurred when the distribution (3.7) is composed of only a few spikes and near-zero values elsewhere. The correct value for X_m can then be located after only a few iterations (one iteration includes one attempt to change the value of each parameter). With the two-step Metropolis technique, however, guessing and accepting the correct value can take several hundred iterations or more.

Although it might appear that each calculation of the probability distribution (3.7) would require about N times the computation that an equivalent Metropolis move requires, modern array processors allow the crosscorrelation functions to be calculated rapidly at a cost that is a modest function of the number of lags. Thus the additional information that is contained in the crosscorrelation function more than compensates for the cost of obtaining it.

3.4 GENERAL ISSUES OF PRACTICAL IMPORTANCE

Statics estimation by simulated annealing has some important general properties, regardless of which technique (Metropolis or heat-bath) is chosen. I discuss below three (unrelated) general issues of practical importance: the specification of T, parallel computations, and the absence of subsurface-consistent parameters in the objective function. In the next section I discuss a general issue of theoretical interest: the representation of the algorithm as a Markov chain.

3.4.1 The specification of temperature

Many schemes for decreasing temperature are possible. Geman and Geman (1984) proved that simulated annealing converges to the global minimum for cases in which the temperature T_k in the kth iteration is no lower than $T_0/\log k$, where the initial temperature T_0 depends on the difference between the greatest possible energy and least possible energy. As Geman and Geman point out, this rate of decrease is too slow to be practical (because the required T_0 is too high). Regardless of the temperature-reduction scheme used, one concept is common to all implementations of simulated annealing: experimentation is necessary. Moreover, because experimentation is needed, it is unlikely that a temperature-reduction scheme used for one problem is applicable to another. I make several comments on experimentation and the choice of T in Chapter 4. My experimental results indicate that a single, carefully chosen value for T can be sufficient for convergence in a statics estimation problem. This value of T is set just below a temperature that is roughly analogous to a critical, or freezing temperature.

3.4.2 Parallel computations

Hinton and Sejnowski (1983), Geman and Geman (1984), and others point out that simulated annealing can be implemented in parallel. Parallel computers are collections of processors that operate simultaneously and independently while maintaining a continuous flow of information from each processor to the others [or a subset of the others; see Hockney and Jesshope (1981) for a survey]. In theory, if there were N processors (say, one for each shot and receiver static), run time could be reduced by a factor of N. This is an important point. Kirkpatrick et al. (1983) report that the run time of simulated annealing is nearly linearly related to the number of parameters being estimated. Parallel computations then not only reduce computation time by a considerable factor, but they can also allow large problems to be solved in virtually the same length of time that small problems require.

3.4.3 Subsurface-consistent terms

The linear equations used by Wiggins et al. (1976) and Taner et al. (1974) include a term that compensates for subsurface structural variations. These authors use a structural term because they measure time delays between traces associated with different CMP gathers. Because the power computations in equations (3.5) and (3.6) and the crosscorrelation calculation in equation (3.9) are performed within CMP gathers, however, no structural variation need be compensated. This is true for stack-power maximization schemes in general, such as the one presented by Ronen and Claerbout (1985). Power maximization schemes do *not* need to decompose structural and near-surface variations. Long wavelength statics (statics that vary slowly relative to the length of the seismic cable), however, are poorly resolved by the data and are generally suspect, as they are in all statics estimation methods [see Wiggins et al. (1976) or Ronen and Claerbout (1985) for more details].

A second subsurface-consistent term used by Wiggins et al. (1976) and Taner et al. (1974) concerns residual normal moveout. Although ignored here, residual normal moveout is still an important parameter. Because data must be NMO-corrected prior to statics estimation, the NMO velocities will undoubtedly contain some error. Residual normal moveout can be estimated using simulated annealing, but the computational burden would probably far outweigh any benefits.

3.5 THE ALGORITHM AS A MARKOV CHAIN

In the remainder of the thesis I refer to the statics algorithm as "Monte Carlo statics estimation" or "statics estimation by simulated annealing" when I discuss concepts applicable to both the one-step and two-step techniques. In this section, I formulate the Monte Carlo statics estimation algorithm as a Markov chain. The key issue in the formulation is the steady state of the Markov chain.

The following notation is used. The unknown shot and receiver statics are again denoted by $\mathbf{X} = \{X_1, X_2, \cdots, X_M\}$. X_m is taken to be a random variable; x_m is its value. Each parameter may assume only one of N distinct values; thus there are N possible values for each x_i. A *state* $\mathbf{x}_i$ (note boldface) is any combination of values assumed by the entire set of parameters $\mathbf{X}$. Because there are M parameters that may each assume one of N values, there are a total of N^M distinct states of the system.

3.5.1 Markov chains

The basic objective of the Monte Carlo statics algorithm is to generate states $\mathbf{x}_j$ from the probability distribution

$$\Pi_j \equiv P(\mathbf{X}=\mathbf{x}_j) = \frac{\exp\{-E(\mathbf{x}_j) / T\}}{\sum\limits_{j \in A} \exp\{-E(\mathbf{x}_j) / T\}} , \tag{3.10}$$

where E is the negative stack power, and $j \in A = \{1, 2, ..., N^M\}$. If $\mathbf{x}_j$ were the configuration of a physical system, E its energy, and T scaled by Boltzmann's constant, then equation (3.10) would be identical to the Gibbs distribution in equation (3.4).

The problem, then, is to generate states $\mathbf{x}_j$ with probability Π_j. This generation is called *importance sampling* (Fosdick, 1963; Hammersley and Handscomb, 1964). Because there are N^M terms in the denominator, the $\{\Pi_j\}$ cannot be explicitly computed. However, a *Markov chain* that generates random states from the probability distribution (3.10) can be constructed. If this random generation is performed with

$T << E(\mathbf{x})$, then states that yield the greatest stack power can be selectively favored due to the exponential weighting, and approximations of the best stack can be generated. This selective favoring is the essence of importance sampling: the more probable (or important) a state is, the more likely it is to be generated by the Markov chain.

A Markov chain is a sequence of states in which the probability of each newly generated state depends only on the previous state, not on any others. The Monte Carlo statics algorithm is a Markov chain in which each state is a new set of timing delays for each shot and receiver. Consider these states to evolve over a sequence of iterations indexed by t. The notion of immediate dependence is formally expressed by

$$P(\mathbf{X}_t=\mathbf{x}_j \mid \mathbf{X}_{t-1}=\mathbf{x}_{i_{t-1}}, \cdots, \mathbf{X}_2=\mathbf{x}_{i_2}, \mathbf{X}_1=\mathbf{x}_{i_1}) = P(\mathbf{X}_t=\mathbf{x}_j \mid \mathbf{X}_{t-1}=\mathbf{x}_{i_{t-1}}) \ . \quad (3.11)$$

If T is constant, the probability of going from any state $\mathbf{x}_i$ to any state $\mathbf{x}_j$ in one iteration is stationary in time, and is given by the *transition probability*

$$p_{ij} = P(\mathbf{x}_i \rightarrow \mathbf{x}_j) = P(\mathbf{X}_t=\mathbf{x}_j \mid \mathbf{X}_{t-1}=\mathbf{x}_i) \ . \quad (3.12)$$

Most important developments in the theory of Markov chains concern the evolution of a chain over time. Generalizing equation (3.12), let $p_{ij}^{(n)}$ be the probability of transition from $\mathbf{x}_i$ to $\mathbf{x}_j$ in n iterations:

$$p_{ij}^{(n)} = P(\mathbf{X}_t=\mathbf{x}_j \mid \mathbf{X}_{t-n}=\mathbf{x}_i) \ .$$

Binder (1979, 1984), Hammersley and Handscomb (1964), Fosdick (1963), and Metropolis et al. (1953) show that as $n \rightarrow \infty$ the probability of the Metropolis algorithm going from $\mathbf{x}_i$ to $\mathbf{x}_j$ is independent of the initial state $\mathbf{x}_i$. I derive in the Appendix a similar result for the one-step method. For both methods, as $n \rightarrow \infty$ the transition probabilities reach a unique steady state, described below.

3.5.2 The steady state

The Monte Carlo statics algorithm exhibits the limit

$$\Pi_j = \lim_{n \rightarrow \infty} p_{ij}^{(n)} \ ,$$

where $\{\Pi_j\}$ is the Gibbs distribution (3.10). This result is proven for the Metropolis algorithm in the references cited above, and is derived for the one-step statics algorithm in the Appendix. $\{\Pi_j\}$ is called the equilibrium, or *steady-state,* distribution of the Markov chain. The term equilibrium is used here as it is in physics: it describes the behavior of the system after the system has "forgotten" its initial state. Because the low-energy (high-stack-power) states are exponentially favored by the distribution $\{\Pi_j\}$, equilibrium at low T approximates the global minimum. Thus the algorithm is theoretically sound if it is run at low T for a sufficiently large number of iterations. Yet the number of iterations required might be impractically large. How, then, does one reconcile the theory with practice?

Reconciliation begins by experimentation. In principle, it is easy to see why the algorithm should work if T is lowered sufficiently slowly so that equilibrium is attained for every T—this is how Kirkpatrick et al. (1983) originally proposed the concept of simulated annealing. The experimental results illustrated in the next chapter, however, show that the choice of an appropriate low constant temperature is indeed practical: one need only wait for equilibrium at this one temperature. As described in the next chapter, the exact specification of the low T is determined by previous experimentation. When equilibrium is attained at low T, the algorithm locates a deep (if not the deepest) minimum. The success of this approach depends on how rapidly equilibrium can be reached. In general, the approach to equilibrium slows down as either the temperature decreases or the number of deep minima increases.