THE ASTROPHYSICAL JOURNAL, 310:473–481, 1986 November 1

OPTIMIZATION ALGORITHMS: SIMULATED ANNEALING AND NEURAL NETWORK PROCESSING

W. JEFFREY AND R. ROSNER
Harvard-Smithsonian Center for Astrophysics
Received 1985 December 9; accepted 1986 March 19

ABSTRACT

We develop and compare two algorithms which have been previously used for discrete optimization: simulated annealing and neural network processing. We demonstrate how these algorithms can be used to find global extrema of functions, while avoiding trapping in local extrema. In the standard treatment of neural network processors, only quadratic and linear terms in the function variables are included in the objective function. We extend this traditional approach to show how constraints not expressible in quadratic and linear terms (e.g., entropy) can be incorporated into the function to be minimized. We also demonstrate the efficiency of our implementation of neural net processing and show how its speed advantage over more traditional optimization techniques (even when implemented on serial processors) is related to its convergence properties. An important application of our results is in the interpretation of remote sensing data, since typical indirect sensing problems can be readily cast into the language of optimization theory; the methods presented here have the particular ability to solve severely ill posed inversion problems. The algorithms described here have been implemented on a serial processor but are cast in a form which is ideally suited for parallel processing.

Subject heading: numerical methods

I. INTRODUCTION

Optimization problems commonly encountered in disciplines like astronomy have traditionally been plagued by two major concerns: (1) finding the global minimum for functions with multiple minima (e.g., nonconvex functions), and (2) solving the problem in minimal computation time. Traditional gradient search methods are computationally quick but are of little use since they are easily trapped by the first minimum they encounter. Exhaustive enumerative searches can usually find the global minimum, but only at the expense of massive computational requirements. The ideal algorithm would instead quickly find the optimal solution. In this paper, we describe two different techniques that have been proposed to solve optimization problems and address one or both of the above concerns. The first, termed simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983), mimics the way a thermalized system with a large number of degrees of freedom attains its ground state as the temperature slowly decreases. The second technique, referred to as neural network processing (Hopfield 1982, 1984; see also Kohonen 1977, 1984), models the collective computational behavior of neurons in the human brain. A major aim of this paper is to demonstrate that these techniques can be reformulated in order to find extrema of functions and, in particular, to apply them to the inversion of remote sensing data. We also demonstrate that both techniques have the ability to locate the *global* extremum of a function; the neural network technique is shown to be particularly efficient in finding the optimal solution.

In § II, we provide a brief overview of the formal mathematics specific to the inversion problem as recast into the optimization framework. Section III describes in some detail both the simulated annealing and neural network processing algorithms, supplemented by specific implementation examples. The final section is devoted to a summary of our results and conclusions.

II. MATHEMATICAL BACKGROUND: INVERSION AS AN OPTIMIZATION PROBLEM

The difficulty of reliably inverting indirect sensing observations has been long recognized and extensively treated from the mathematical perspective (see Menke 1984; Jansson 1984). Nevertheless, it remains the case that the optimal inversion techniques which have developed have not seen wide application, especially in areas such as astronomy. There are several reasons for this, but perhaps the most prominent are the perceived uncertainties related to error propagation and sensitivity to noise; and the consequent perception has been that inversion is more of an art form than a science. In the accompanying paper (Jeffrey and Rosner 1986), we explored these issues systematically for the principal available methods by conducting extensive simulations. The purpose of the present paper is instead to develop new analytical and computational tools for carrying out such inversions practically. In this section, we briefly lay out the basic mathematical issues confronted in the typical remote sensing problem and provide the framework for formulating the inversion as an optimization problem.

a) The General Issues

In the general remote sensing problem, one is required to solve an equation of the generic form

$$g = Kq\,, \tag{2.1}$$

where g represents the data, K represents some integral operator, and q is the result to be determined. In explicit integral form, equation (2.1) is just a Fredholm equation of the first kind,

$$g(y) = \int_D dx k(x, y) q(x)\,, \tag{2.2a}$$

where $k(x, y)$ represents a known kernel (which might, for example, be the instrument point response function) and D is the support of k; in the discrete form, equation (2.1) instead reduces to the inversion problem

$$g_i = \sum_{j=1}^{N} k_{ij} q_j , \tag{2.2b}$$

where k_{ij} is a known matrix. The inversion of equations (2.1)–(2.2) for perfect data (i.e., if the signal $\boldsymbol{q}$ is uncontaminated by noise, and $\boldsymbol{g}$ is known with perfect precision) is a classic mathematics problem.

Among the known attributes of equation (2.1) are (Colton and Kress 1983) that it is both illposed and illconditioned: (1) solutions may not necessarily exist and the solution—if it exists at all—is likely to be nonunique; and (2) such solutions—if they do exist—in general depend discontinuously on the data $g(y)$ and hence are likely to be unstable to small errors in the data. The reason for the first difficulty can be readily appreciated by glancing at Figure 1, which sketches the general transformation $K: q \rightarrow g$ schematically: Inevitable measurement errors or signal noise can lead to data which do not lie in the range of K, and hence not in the domain of the inverse operator K^{-1}; thus, no solution will exist. Similarly, the transformation $K: q \rightarrow g$ may not be one-to-one (viz., if K is a smoothing operator), so that the inverse transformation does not give a unique result. Finally, the unstable behavior of the solution can be appreciated by examining the matrix form of equation (2.2b):

$$\boldsymbol{g} = K\boldsymbol{q} . \tag{2.3}$$

If $\boldsymbol{g}$ contains (observational) errors $\Delta\boldsymbol{g}$, then a direct inverse in the least-squares sense, i.e., a generalized inverse (Moore 1935; Bjerhammar 1951; Penrose 1955), would yield the elementary solution

$$\boldsymbol{q} = (K^T K)^{-1} K^T [\boldsymbol{g} + \Delta\boldsymbol{g}] \tag{2.4}$$

(where we have not assumed that K need be a square matrix). If the row vectors of $K^T K$ are not fully linearly independent, then $K^T K$ will be nearly singular, and the above inversion will greatly magnify the error component, possibly driving it to dominate the solution. This is the basis for the ill-conditioned (unstable) nature of the problem.

b) Resolving the Inversion Problem: Regularization and Optimization

From the above we conclude that equation (2.2) may *formally* possess no solution or possibly an infinite set of solutions. However, the physical situation guarantees that a solution must exist, and the question is then how to arrive at a (hopefully unique) "answer." Evidently, in order to find that one solution some new *a priori* constraint must be imposed on the inversion. This additional condition (or conditions) will often have no other justification than it (they) select a unique solution, so that the inversion problem becomes well posed. What we shall mean by "solving" equation (2.2) is, first, that the solution both minimize the usual norm constraint

$$h(q) = \int_{\Omega} dy [g^d(y) - g(y)]^2 \tag{2.5a}$$

and satisfy additional constraints, which we shall write in the form

$$\xi(q) = 0 , \tag{2.5b}$$

Here $g^d(y)$ is the given data, $g(y)$ is the result of folding a trial solution $q(x)$ through equation (2.2a),

$$g(y) \equiv \int_D dx k(x, y) q(x) , \tag{2.6}$$

and Ω is the range of y. For the discrete problem (2.2b), for which the number of points at which q is to be evaluated is (say) N, h is a function on R^N, $h: R^N \rightarrow R$. Such use of *a posteriori* constraints to find a solution can be viewed as a kind of regularization (Tikhonov and Arsenin 1977) of the inversion problem (2.2) and is the basis for virtually all the standard inversion techniques in the presence of noise, including the Phillips-Twomey method (Twomey 1977), the modified Chahine nonlinear iteration method (Chahine 1970), the Backus-Gilbert inversion (Backus and Gilbert 1970), the spectral synthesis method (Parker 1977), and the collection of methods referred to generically as the maximum-entropy

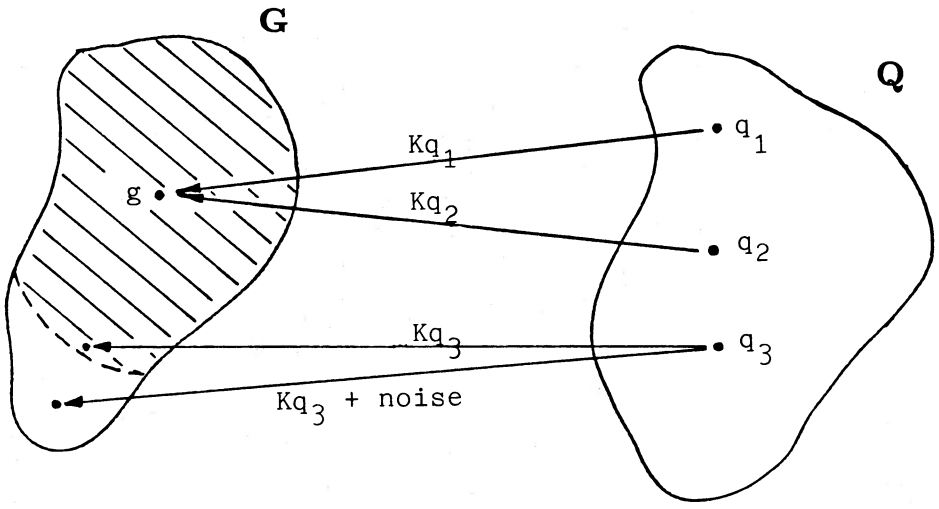

FIG. 1.—Schematic representation of the inversion problem posed by eq. (2.1), and the origin of its ill-posed and ill-conditioned nature (the details are discussed in the text). The shaded region in the data space G indicates the range of K.

inversions (Jaynes 1957; Frieden 1972). A comparison of the efficacy of these various methods for inversions of practical interest in astronomy is given by Jeffrey and Rosner (1986).

One general approach to solving this constrained minimization problem is thus to construct a "penalty" (or "cost," or "objective") function of the form

$$H(q, \lambda) = h(q) + \lambda[\xi(q)]^2 \tag{2.7}$$

and then to minimize $H(q, \lambda)$, i.e., solve an unconstrained minimization problem.

The optimization approach to solving the inversion problem raises two central questions:

1. Is there a *best* recipe for defining the objective function?
2. Is there some guarantee that the optimization algorithm in fact can find the true (global) minimum?

We shall not discuss the first of these two concerns further, and defer to Bertsekas (1982) and Poggio, Torre, and Koch (1985) for recent cogent discussions. As for the second issue, it is well known that as long as the objective function H is convex, then standard gradient search algorithms function well (provided one always recalls the pitfalls of naive gradient descent; see Acton 1970). If the cost function is not convex, then global minimization becomes very difficult, and stochastic techniques are typically appealed to; the resulting great cost in computational effort then raises the third central question of inversion by optimization:

3. Is the optimization algorithm efficient?

In the next section, we address this specific problem by providing a translation of the above minimization problem to algorithms which are especially adept at locating the global minimum of functions. These algorithms are also well suited to parallel processing and hence promise to vastly speed the minimization problem.

III. SOLUTION OF INVERSE PROBLEMS BY SIMULATED ANNEALING AND NEURAL NET PROCESSING

Since the seminal paper by Metropolis *et al.* (1953), it has been recognized that the efficiency of solving optimization problems with many constraints is vastly improved if the minimization algorithm allows both increases and decreases in the cost (or likelihood) function as the trial solution proceeds. This property strongly distinguishes the method discussed here from approaches such as gradient search methods, which invariably follow the path of decreasing cost only. Indeed, the ability to "climb out" of local minima is essential for problems in which the cost function is not convex, but yet where the global minimum is sought. In this section, we shall discuss two distinct methods for accomplishing global optimization in the presence of local minima in the objective function: the direct adaptation of the "simulated annealing" algorithm developed by Kirkpatrick, Gelatt, and Vecchi (1983), and the representation of the optimization problem (2.7) in the context of neural network processing (Hopfield 1982, 1984; Hopfield and Tank 1985). Of necessity, our discussion will be illustrative and based on inversion or optimization problems which are typical of those faced in astronomy; we therefore do not claim to be exhaustive in our study of optimization problems.

a) Inversions via Simulated Annealing

The method proposed by Kirkpatrick, Gelatt, and Vecchi (1983), which is an improvement over the algorithm first proposed by Metropolis *et al.* (1953), simulates the slow annealing of physical systems characterized by cost functions with both global and local minima (such as, for example, the free energy function associated with spin glasses). Central to this analogy is the ability of a thermal system to locally increase its free energy if such an increase allows the system as a whole to attain a lower free energy state. Allowing the free energy to increase is therefore crucial for avoiding trapping in local, nonoptimal minima. The probability for such an increase in free energy is just given by the Boltzmann probability $\exp(-\Delta E/KT)$, where ΔE is the energy change and KT is a measure of the thermal energy. As the temperature decreases, the number of changes accepted that increase the system's energy quickly diminishes, and the system is considered "frozen." The extension of this technique to solving optimization problems is straightforward: the energy of the system is given by the objective function, and the ground state configuration corresponds to the global minimum of the problem. The essence of the method lies in defining (1) an algorithm for local adjustments of the trial solution; (2) a stochastic decision algorithm (which uses the method of Metropolis *et al.* 1953); and (3) an "annealing schedule" described by Kirkpatrick, Gelett and Vecchi (1983), which adjusts the decision algorithm probability function as the solution proceeds. Simulated annealing has been largely applied to optimization problems in which the decision tree for modifying the cost function is binary: for example, in the classic "traveling salesman" problem, the cost function (which is just the length of the total path traversed in visiting all target cities) is modified by exchanging the order of cities traveled. In the following, we show by illustration that the type of problem discussed in § II, e.g., the solution of equation (2.2), can be readily translated to an optimization problem solvable by simulated annealing via a modification of the trial solution adjustment algorithm (see also Geman and Geman 1984; Vanderbilt and Louie 1984).

Consider, for example, the optimization problem (2.7): It is immediately obvious that the decision tree for progressively modifying the trial solution q is not binary in the above sense, but is in general ternary; that is, for each value of x, we must decide whether to increase or decrease $q(x)$, or to keep it the same. This seeming difficulty can be trivially circumvented as follows: We force the tertiary decision of whether to change $q(x)$ into a sequence of two binary decision problems: the first choice is either to (1) decrease $q(x)$ or retain its value, or to (2) increase $q(x)$ or retain its value; the first binary choice between (1) and (2) is then made with equal probability, whereas the second binary choice [between changing $q(x)$ or retaining its value] depends on the effect of the change on the objective function [with the probability for changing $q(x)$ if H is increased again varying like a Boltzmann factor].

Vanderbilt and Louie (1984) explored the efficiency of simulated annealing as compared to several other standard optimization algorithms. Their results indicate that when comparing the number of function evaluations and computer time, simulated annealing is roughly equivalent to techniques that systematically explore the entire state space. When one considers, however, the success rate in finding the global minimum, then for simple low-dimensional problems it is more effective to use a systematic search routine. Similarly, for problems for which the cost function is clearly convex and simple in geometric structure, one is far better off using standard gradient descent methods. However, for problems such as the fitting of complex emission and absorption line spectra and image deconvolution, the state space dimensionality can be

enormous, and an exhaustive search for a global minimum is entirely prohibitive. These are the circumstances suited for the simulated annealing method; and a demonstration of the relative effectiveness of this algorithm for solving this class of problems is given in § IIIc.

b) Optimization via Neural Network Processing

We now consider the optimization problem solved in the previous section from the point of of view of dynamics. This allows us to embed the optimization problem in a neural network processor[1] (Hopfield 1982, 1984) and thereby gain computational speed by virtue of both the efficiency of the algorithm and the inherently parallel processing capabilities of neural network processors. This advantage has already been taken advantage of in discrete optimization problems by Hopfield and Tank (1985).

In order to retain the essential thrust of the discussion without overburdening details, let us suppose that equation (2.2) has led us to minimize the cost functional

$$H(\boldsymbol{q}) = \tfrac{1}{2}\sum_{i=1}^{M} (g_i^{\,d} - g_i)^2 , \tag{3.1}$$

where, as before, $\{g_i^{\,d}, i = 1, \ldots, M\}$ are the data, and the $\{g_i\}$ are the result of folding the trial solution $\boldsymbol{q} = (q_1, \ldots, q_N)$ through the transform

$$g_i = \sum_{j=1}^{N} k_{ij} q_j . \tag{3.2}$$

If we insert equation (3.2) into equation (3.1), rearrange the order of summations, and define the new quantities

$$I_j = \sum_{i=1}^{M} k_{ij} g_i^{\,d} , \tag{3.3a}$$

$$T_{ij} = -\sum_{l=1}^{M} k_{li} k_{lj} \tag{3.3b}$$

(the relevance of these new functions, which are constructed from known quantities, will soon become apparent), then we can rewrite the objective function (3.1) for the optimization problem in the form

$$H(\boldsymbol{q}) = -\tfrac{1}{2}\sum_{i,j=1}^{N} T_{ij} q_i q_j - \sum_{j=1}^{N} I_j q_j + \tfrac{1}{2}\sum_{i=1}^{M} (g_i^{\,d})^2 . \tag{3.4}$$

This form is reminiscent of the Hamiltonian of a dynamical system but, as we shall see in a moment, the actual situation is substantially more complex.

The problem of minimizing equation (3.4) (or [3.1]) boils down to finding the global minimum of the surface $H = H(\boldsymbol{q})$ in the $(N + 1)$–dimensional space $R^N \times R$ {with elements $[\boldsymbol{q}, H(\boldsymbol{q})]$}. Note that while the illustrative $H(\boldsymbol{q})$ defined above is convex, it in general need not be, and nothing in the following depends on the convexity properties of $H(\boldsymbol{q})$.

i) The Basic Algorithm

In the following, we (1) derive the basic iteration scheme for minimizing H, (2) demonstrate that in a discrete formulation the sign of changes in H following perturbations in $\boldsymbol{q}$ can be directly controlled, (3) account for the convergence efficiency of the algorithm, and (4) prove the strict convergence property of the contraction operator associated with our algorithm. Now, consider the effect of an affine transformation on the space Q of N-tuples $\boldsymbol{q}$. L: $Q \rightarrow Q$, which induces a motion in Q-space. For definiteness, use t as the parameter for this induced dynamics, and consider a continuous change of the objective function along this trajectory:

$$\frac{dH}{dt} = \sum_{i=1}^{N} \frac{\partial H}{\partial q_i}\frac{dq_i}{dt} . \tag{3.5}$$

From equation (3.4), we can calculate $\partial H/\partial q_i$:

$$\frac{\partial H}{\partial q_i} = -\sum_{j=1}^{N} T_{ij} q_j + I_i . \tag{3.6}$$

Hence we have

$$\frac{dH}{dt} = -\sum_{i=1}^{N} \frac{dq_i}{dt}\left(\sum_{j=1}^{N} T_{ij} q_j + I_i\right) . \tag{3.7}$$

In contrast, consider a discrete form of the transformation L: define L_k: $[q_1^{(n)}, \ldots, q_k^{(n)}, \ldots, q_N^{(n)}] \rightarrow [q_1^{(n)}, \ldots, q_k^{(n+1)}, \ldots, q_N^{(n)}]$, and $L() \equiv L_1\{L_2[\ldots L_N()\ldots]\}$; then we can equivalently determine the change in the objective function upon iteration:

$$H + \Delta H_k = -\tfrac{1}{2}\sum_{i,j}^{N} T_{ij}(q_i + \Delta q_i \delta_{ik})(q_j + \Delta q_j \delta_{jk}) - \sum_{i=1}^{N} I_i(q_i + \Delta q_i \delta_{ik}) + \tfrac{1}{2}\sum_{i=1}^{M} (g_i^{\,d})^2 , \tag{3.8}$$

or

$$\Delta H_k = -\tfrac{1}{2}\sum_{i,j}^{N} T_{ij}(q_j \Delta q_i \delta_{ik} + q_i \Delta q_j \delta_{jk} + \Delta q_i \Delta q_j \delta_{ik}\delta_{jk}\} - \sum_{i=1}^{N} I_i \Delta q_i \delta_{ik} . \tag{3.9}$$

Note that equations (3.7) and (3.9) differ by the term nonlinear in the Δq_i; this point will become essential in a moment (see eq. [3.12*b*]).

Now, in either case, we must specify the transformation L_k. By regarding $\boldsymbol{q}$ as the output vector of a network of N neurons, T as the "synaptic connectivity" matrix, and I as the "bias" input term (see Hopfield and Tank 1985 for a detailed discussion), then we are motivated to construct the natural update transformation for the neural network, namely

$$q_i^{(n+1)} = q_i^{(n)} + \lambda_i\left[\sum_{j=1}^{N} T_{ij} q_j^{(n)} + I_i\right]\delta t , \tag{3.10}$$

where λ_i is the "gain" for the ith neuron (note that this definition of the "gain" differs from that of Hopfield 1984), and both T_{ij} and I_i are, by construction, identical to the definitions (3.3). Note that, comparing equations (3.6) and (3.10), that the resulting iteration scheme is thus one of the class of gradient descent methods.

Two limits of (3.10) are of interest. If $\delta t \rightarrow 1$, we have the discrete transformation

$$\Delta q_i^{(n)} = \lambda_i\left[\sum_{j=1}^{N} T_{ij} q_j^{(n)} + I_i\right] \tag{3.11a}$$

(where $\Delta q_i^{(n)} \equiv q_i^{(n+1)} - q_i^{(n)}$), whereas for $\delta t \rightarrow 0$, we have the

[1] The neural network owes its name to modeling the dynamics of systems that attain an energy minimum with the way (it is believed) the brain's neurons reach a stable state (energy minimum) when converging on a memory.

continuous transformation

$$\frac{d}{dt}\boldsymbol{q} = \boldsymbol{\lambda} \cdot L\,, \tag{3.11b}$$

with $[L_i(\boldsymbol{q})]_i = \sum_{j=1}^{N} T_{ij} q_j + I_i$, and $(\boldsymbol{\lambda})_i = \lambda_i$. Note that as long as max $\{\lambda_i\}$ is bounded, then the continuous operator defined in equation (3.11b) is bounded as well.

ii) The Sign of dH/dt

Now, note that with equations (3.10) and (3.11), we have

$$\frac{dH}{dt} = -\sum_{i=1}^{N}\left(\sum_{j=1}^{N} T_{ij} q_j + I_i\right)\frac{dq_i}{dt} = -\sum_{i=1}^{N}\frac{1}{\lambda_i}\left(\frac{dq_i}{dt}\right)^2 \le 0 \tag{3.12a}$$

for the continuous case, and

$$\Delta H_k = -\left(\sum_{j=1}^{N} T_{kj} q_j + I_k\right)\Delta q_k - \frac{1}{2} T_{kk}\Delta q_k \Delta q_k$$

$$= -(\Delta q_k)^2\left(\frac{1}{\lambda_k} + \frac{T_{kk}}{2}\right) \tag{3.12b}$$

for the discrete case. This result is the essence of the difference between the continuous and the discrete cases: whereas $dH/dt \le 0$ for the former case (unless $\lambda < 0$), in the latter case we have the distinct possibilities

i) $\Delta H_k \ge 0$ if $\lambda_k \le 0\,, \qquad T_{kk} \le \dfrac{2}{|\lambda_k|}\,,$

ii) $\Delta H_k \ge 0$ if $\lambda_k \ge \dfrac{2}{|T_{kk}|}\,, \quad T_{kk} < 0\,,$

otherwise $\Delta H_k \le 0$.

To conclude: The differential form unavoidably leads to a decrease in the objective function along any traectory in Q-space generated by the linear operator L with positive λ (and similarly for any discretized version of the related equation of motion [3.11b]); however, the discrete transformation (3.11a) does allow for an increase in the objective function for positive λ. Thus it is essential to distinguish how the discretization is carried out: the quadratic terms of equation (3.9), which lead to the term $T_{kk}/2$ in equation (3.12b), appear only if the discretization is accomplished *before* the limit $\Delta\boldsymbol{q} \to 0$ is taken. Note also that although $T_{kk} \le 0$ for the class of kernels discussed above, it can be positive [for example, for complex kernels such as the Fourier kernels $\exp(i\omega t)$]; in that case, condition (ii) above becomes irrelevant.

iii) Convergence Efficiency

Now, rather than focusing on the variation of the objective function H along the trajectory generated by L, let us consider the trajectory itself and ask how efficiently it takes us to the optimal solution. We begin by writing equation (3.11a) in matrix notation, using the representations defined in equations (3.2) and (3.3): $T \equiv -K^T K$, $\boldsymbol{I} \equiv K^T \boldsymbol{g}^d$, and $\boldsymbol{g} \equiv K\boldsymbol{q}$. Thus,

$$\Delta\boldsymbol{q}^{(n)} = L[\boldsymbol{q}^{(n)}] = \lambda(-K^T K\boldsymbol{q}^{(n)} + K^T\boldsymbol{g}^d)\,,$$

or

$$\Delta\boldsymbol{q}^{(n)} = -\lambda K^T(\boldsymbol{g}^{(n)} - \boldsymbol{g}^d)\,, \tag{3.13}$$

where we have assumed for simplicity that λ is a constant. This is the basic neural network iteration scheme.

In order to study how this iteration scheme converges to the solution, we consider the "motion" of the trial solution in the data space G: operating on both sides of equation (3.13) by K, and defining $\Delta\boldsymbol{g}^{(n)}$ to be just $K\boldsymbol{q}^{(n)}$, gives

$$\Delta\boldsymbol{g}^{(n)} = -\lambda KK^T(\boldsymbol{g}^{(n)} - \boldsymbol{g}^d)\,. \tag{3.14}$$

For real kernels K, KK^T is a real, symmetric matrix and thus can be diagonalized. (In general, we would require a singular value decomposition of KK^T.) We can discard zero and "nearly" zero eigenvalues (or singular values for K not real) and corresponding eigenvectors, so that we will be instead dealing with (in the matrix representation) the resulting reduced rank matrix with dimensions $M' \times M'$ $(M' < M)$. Note further that the diagonalization or singular value decomposition of KK^T now insures a one-to-one mapping of the space Q to the data space G. Performing this transformation on equation (3.14) gives

$$\Delta\boldsymbol{g}^{(n)} = -\lambda U\Lambda U^T(\boldsymbol{g}^{(n)} - \boldsymbol{g}^d)\,, \tag{3.15a}$$

or

$$U^T\Delta\boldsymbol{g}^{(n)} = \lambda\Lambda[U^T\boldsymbol{g}^d - U^T\boldsymbol{g}^{(n)}]\,, \tag{3.15b}$$

where U is the matrix containing the orthonormal eigenvectors of the reduced rank matrix KK^T and Λ is a diagonal matrix containing the corresponding eigenvalues. Geometrically, equation (3.15b) just represents a rotation which leaves the norms of the vectors unchanged. Representing the rotated vectors by a tilde, we have

$$\Delta\tilde{g}^{(n)} = \lambda\Lambda(\tilde{g}^d - \tilde{g}^{(n)})\,, \tag{3.16a}$$

or, in component form,

$$\Lambda\tilde{g}_i^{(n)} = \lambda\Lambda_{ii}(\tilde{g}_i{}^d - \tilde{g}_i^{(n)})\,. \tag{3.16b}$$

Thus we see that on each iteration, the trial vector $[\tilde{g}^{(n)}]$ rotates toward the optimal solution $(\tilde{g}^d)$, weighted by the eigenvalues of the reduced rank KK^T. This weighting is (in some sense) optimal, since for an ill-conditioned problem, the largest eigenvalues correspond to those observations that can be determined with the least uncertainty (in the presence of noise).

iv) Demonstration of Strict Contraction

We now show that equation (3.16) corresponds to a contraction in Q-space, e.g., that L is a contraction operator and that this contraction is monotonic toward the attracting fixed point (i.e., is not oscillatory). First, note that the objective function H induces a metric on the data space G (whose elements are the vectors $\boldsymbol{g}$), but not on the space Q in which the trajectory is actually traced out (recall that L is an affine transformation on the space Q). This raises the problem that although one can trivially impose the Euclidean norm on the space Q, it is not obvious that points in Q close (in the Euclidean sense) to the solution point $\boldsymbol{q}^d$ (defined by $\boldsymbol{g}^d = K\boldsymbol{q}^d$) map via K to points in G close to the data $\boldsymbol{g}^d$. It is true, however, that the discrete transformation (3.11a) has the properties of

a) a contraction map on Q: $d[L(\boldsymbol{q}_1), L(\boldsymbol{q}_2)] < \mathrm{d}(\boldsymbol{q}_1, \boldsymbol{q}_2)$, where $d(,)$ is an appropriately defined distance function on Q; and in addition,

b) a strict contraction on every coordinate line passing through the optimal solution $\{L_i(\boldsymbol{q}_1)\}_i - \{L_i(\boldsymbol{q}^d)\}_i < \{\boldsymbol{q}_1\}_i - \{\boldsymbol{q}^d\}_i$ for all i.

The latter implies in particular that the path generated by L does not "spiral" into the optimal solution, but rather (in a well-defined sense) heads "straight in" to the solution $\boldsymbol{q}^d$.

To prove that L is a contraction mapping on Q, we require that

$$\|L(\boldsymbol{q}^{(n)}) - L(\boldsymbol{q}^d)\| = \alpha\|\boldsymbol{q}^{(n)} - \boldsymbol{q}^d\| \text{ , with } 0 < \alpha < 1 \text{ . (3.17a)}$$

Note that $\boldsymbol{q}^d$ is a fixed point, since by construction (eq. [3.10]), $L(\boldsymbol{q}^d) = \Delta\boldsymbol{q} = 0$. Writing out the mapping L explicitly gives

$$\|T\boldsymbol{q}^{(n)} + \boldsymbol{I} - (T\boldsymbol{q}^d + \boldsymbol{I})\| = \|T(\boldsymbol{q}^{(n)} - \boldsymbol{q}^d)\| \text{ . (3.17b)}$$

Since T is a bounded linear operator, we have the inequality

$$\|T(\boldsymbol{q}^{(n)} - \boldsymbol{q}^d)\| \leq \|T\|\,\|\boldsymbol{q}^{(n)} - \boldsymbol{q}^d\| \text{ , (3.18)}$$

where by $\|T\|$ we mean the induced norm (natural norm)

$$\|T\| = \sup \frac{\|Tx\|}{\|x\|} \; (x \neq 0) \text{ . (3.19)}$$

Identifying $\|T\|$ with α in equation (3.17a) confirms that L is a contraction operator on Q if $\|T\| < 1$, which is in general true for most physical deconvolution kernels. Writing equation (3.18) in the basis in which KK^T is diagonal (and hence T is also diagonal) allows us to examine the effect on L on each component:

$$|\tilde{T}(\tilde{q}^{(n)} - \tilde{q}^d)_i| < |\tilde{T}_{ii}|\,|\tilde{q}_i^{(n)} - \tilde{q}_i{}^d| \text{ . (3.20)}$$

As long as $|\tilde{T}_{ii}| < 1$, we are insured a strict contraction on each coordinate; thus, we have proved condition (*b*) above.

v) *Incorporating Nonquadratic and Nonlinear Terms*

Often when incorporating *a priori* information or smoothness into the objective function, terms other than quadratic or linear in the variables will arise. These terms do not fit readily into the formalism as described above. However, two methods avail themselves for incorporating these expressions into the neural network computational structure:

1. We can Taylor-expand the term to be added and only retain terms up to a certain order in q. But this truncation may not capture some of the essential analytical properties of the original constraint; for example, for terms such as the configurational entropy (q ln q), the Taylor expansion is not guaranteed positive.

2. We can add terms such as entropy directly to the update equation (3.11), rather than to the objective function. More precisely, we add the derivative of the entropy term to equation (3.11), normalized in such a way that the added term will vanish if the trial solution does indeed have maximum entropy. This method is the one we prefer, because it generally does capture the essential analytical properties of the constraint. Thus, by following the second prescription, we can add any differentiable constraint to the objective function. A specific example of this implementation is given in § III*c*.

c) Examples

In this section, we illustrate and contrast the performance of simulated annealing, neural network processing, and a standard efficient conjugate-gradient descent algorithm by solving two prototype problems: the inversion of a one-dimensional Fredholm equation of the first kind involving a purely convex objective function; and minimization of a two-dimensional objective function with multiple minima. In addition, we illustrate how an objective function can be modified to include constraint terms (such as entropy) which are themselves not quadratic forms. These examples obviously do not exhaust the range of possibilities of applying the algorithms under study, but only illustrate their power.

i) *A Simple One-dimensional Fredholm Equation of the First Kind with Convex Cost Function*

The problem is defined as follows (from Twomey 1977): We want to invert the Fredholm equation of the first kind

$$g(y) = \int_0^1 x \exp(-xy) q(x) dx \text{ , (3.21)}$$

given the data $g(y)$ at a fixed number of points $\{y_i\}$.

In discretized form, the kernel functions are $k(x, y_i) = x \exp(-xy_i)$, where we use the values $\{y_i{}^{-1}\} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We specify a particular form for $q(x)$, namely $q(x) = 1 + 4(x - \frac{1}{2})^2$, carry out the direct transform in order to synthesize the data at the $\{y_i\}$, and then attempt the inversion in order to recover this known test function.

The problem at hand has an extra degree of difficulty because it is especially illposed and illconditioned (because the kernels for each i greatly overlap each other); hence a direct inversion of equation (3.21) is not possible. Instead, additional constraints such as maximizing entropy, smoothness, or flux conservation, or any other *a priori* knowledge about the solution (such as boundary values or initial conditions) must be used to insure a physically plausible answer. This additional information must be incorporated into the objective function.

We chose the objective function

$$H(q) = \sum_{i=1}^{M} \left[\frac{g_i - g_i{}^d}{g_i{}^d}\right]^2 + \phi_1 \sum_{j=1}^{N} q_j \ln(cq_j) + \phi_2 \left[\sum_{j=1}^{N} q_j - F_{\rm TOT}\right]^2 \text{ , (3.22)}$$

where the first term measures the "goodness of fit" to the data, the second term is a constraint that selects out a plausible solution with maximum configurational entropy, and the last term requires the flux in the solution to match the observations. The ϕ's are Lagrange multipliers that describe the relative weighting of the terms, c is a normalization constant discussed further below, and $F_{\rm TOT}$ is the total observed flux.

Although equation (3.22) can be used directly for simulated annealing, the entropy term does not fit easily into the neural network formalism as described above.

Thus, following the prescription outlined in the previous section, we modify the update equation to include the derivative of the entropy term:

$$\Delta q_i = \lambda\left\{\sum_{j=1}^{N} T_{ij} q_j + I_i + \phi_1 c[1 + \ln(cq_i)]\right\} \text{ , (3.23a)}$$

where

$$T_{ij} = -\sum_{l=1}^{M} \frac{K_{li} K_{lj}}{(g_l{}^d)^2} - \phi_2 \text{ (3.23b)}$$

and

$$I_i = \sum_{l=1}^{M} \frac{K_{li}}{g_l{}^d} + \phi_2 F_{\rm TOT} \text{ ; (3.23c)}$$

the constant c is chosen to be

$$c = N \exp(-1)\left[\sum_{j=1}^{N} q_j\right]^{-1} \text{ . (3.23d)}$$

This choice insures that for a constant solution ($q_i = [1/N] \sum_{j=1}^{N} q_j$ for all i) (i.e., a solution with maximum entropy), the added term will vanish.

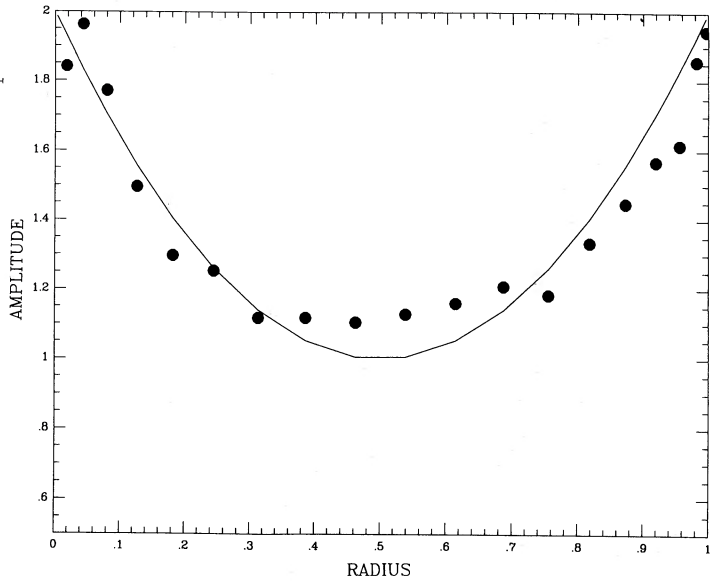

FIG. 2a

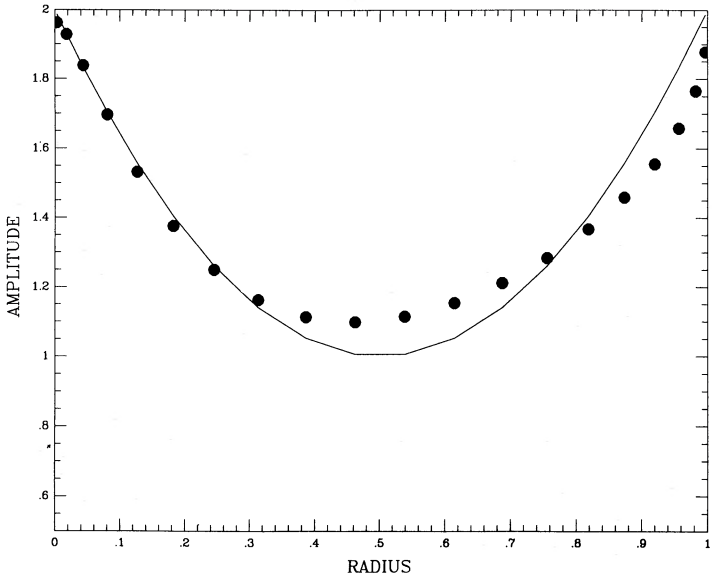

FIG. 2b

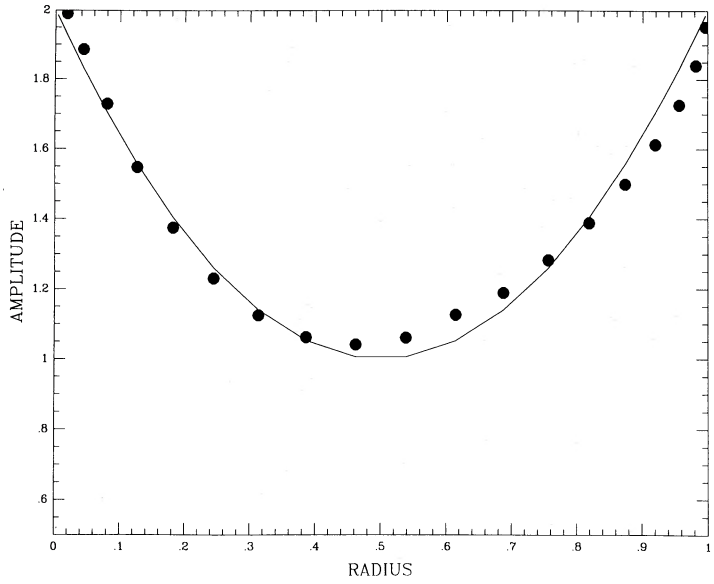

FIG. 2c

FIG. 2.—Comparison of solutions to the inversion problem (2.2*b*) using (*a*) simulated annealing, (*b*) neural network processing, and (*c*) a Polak-Ribiere conjugate gradient method. In all cases, the solid curve is the "true" solution q_i, the kernels are the Poisson kernels given in eq. (3.21), the objective function is convex (eq. [3.22]), and the observations are assumed to be noise-free; the best inversion in each case is given by the locus of circles.

Figure 2 compares the results of applying simulated annealing, neural network processing, and a standard Polak-Ribiere conjugate gradient method (see Press *et al.* 1985) for solving equation (3.22) for noise-free observations. Excellent agreement was obtained in all cases, but the computation time varied substantially. Table 1 gives the relative timing (≡CPU plus system time) for the three algorithms. Note that the neural network algorithm is slightly faster than even the conjugate gradient method, a speed advantage which derives from the way in which the former algorithm uses the global geometry of the objective function to rotate the trial solution vector toward the global minimum. This advantage is most drastically contrasted with traditional gradient search methods, which only use the local geometry (the gradient) to locate the minimum.

ii) *Problems with Nonconvex Cost Functions*

The objective function in equation (3.22) is convex and so possesses only one minimum—the global minimum. Hence, any pure descent method could also have been used to find the optimal solution, and the only advantage of the neural network algorithm resides in its speed. In contrast, the real power of simulated annealing and neural network processing lies in their ability to escape from local minima and to find the global minimum. In order to fully illustrate this capability, we search for the global minimum of the following nonconvex function:

$$f(x, y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (4y^2 - 4)y^2 \,.$$

$$(-3 \le x \le 3,\ -2 \le y \le 2) \,. \qquad (3.24)$$

TABLE 1

RELATIVE COMPUTATION TIME

Method	Integral Equation (first example)	Function Minimum (second example)	Global Minimum Found
Simulated annealing	150	2005	75%
Conjugate gradient	1.2	1	56
Neural network	1	1.8	76
		4.2	100

It is easily verified that this function has six minima, of which two are global minima.

Since we want to locate (one of) the global minima of this function, we can use equation (3.24) directly as the objective function. Only two parameters are then to be determined, namely, the values of x and y at the global minimum. Thus in the framework of the neural network, we only have two "neurons" to consider; and a function with n independent variables would require n "neurons." The potential for parallel processing thus becomes very clear; A system allowing parallel computation for each "neuron" would be able to solve a high-dimensional problem nearly as quickly as for the two-dimensional case.

In order to solve equation (3.24) using the neural network algorithm, we partition the function $f(x, y)$ into quadratic terms and terms higher order in x and y. The quadratic terms fit directly into the formalism as outlined in § III*b*, whereas the higher order terms can be treated in a similar fashion to the entropy term in the previous example. We thus generate the following update equation:

$$\Delta q_1^{(n)} = \lambda\left[\sum_{j=1}^{2} T_{ij} q_j^{(n)} - 2(q_1^{(n)})^5 + 8.4(q_1^{(n)})^3\right], \quad (3.25a)$$

$$\Delta q_2^{(n)} = \lambda\left[\sum_{j=1}^{2} T_{ij} q_j^{(n)} - 16(q_2^{(n)})^3\right], \quad (3.25b)$$

where

$$T_{11} = -8\,, \; T_{12} = T_{21} = -1\,, \quad T_{22} = 8\,,$$

and where $q_1 = x$ and $q_2 = y$. In order for the neural network algorithm to allow the evolving trial solution to escape from local minima, it is necessary to increase the value of the objective function (eq. [3.24]). Thus either inequality i) or ii) in § III*b*(ii) must be satisfied. To satisy either one of them would be sufficient, so that only computational convenience decides which one to choose. In general, we have found that the easiest inequality to satisfy in the numerical implementation is to let $\lambda \rightarrow -\lambda$. This choice ensures that the trial solution will move "uphill" along the cost surface, and hence away from the region of locally lower cost.

When implementing the neural network algorithm to solve equation (3.24), we let the system find a minimum and then reverse the sign of the gain (for a predetermined number of steps) so that the trial solution can climb out of this (possibly local) minimum (note that the trial solution does not simply retrace the previous descent path). The neural network algorithm is then used to find the next minimum (by again reversing the sign of λ); this process is repeated, always keeping track of the lowest minimum found. The search terminates either if the trial solution has "trouble" climbing out of a given minimum (i.e., if the evolving trial solution is trapped after a fixed number of attempts at escaping from the minimum), or until a fixed number of minima has been found. We do not claim that either of these stratagems is optimal in any sense. This procedure for optimizing via neural network processing is thus quite different from the philosophy of simulated annealing and is instead much more closely allied with exhaustive enumerative search. The distinction from exhaustive search is that the neural network algorithm itself picks the succeeding starting points for locating minima by moving directly away from the latest minimum already found. In contrast, exhaustive search methods use either random or pre-selected starting points and hence cannot distinguish between new minima and minima already mapped out.

Table 1 also compares simulated annealing, neural network processing, and the conjugate gradient method (which always stops after encountering any minimum) in finding the global minimum. The results are striking: Although in this case the conjugate gradient method was fastest in descending to the global minimum once it entered the basin of attraction for the global minimum, it located the global minimum only 56% of the time (for 1000 trials). The effective rate for annealing is instead almost 2000 times slower (based on 70 trials) than the conjugate gradient procedure but has a success rate of 75% of locating the global minimum. Finally, the neural network algorithm instead ran roughly half as quicky as the conjugate gradient search (as long as the latter once again found itself within the attracting basin) but located the global minimum 76% of the time (for 1000 trials); if instead we are willing to tolerate speeds roughly 4 times slower than the conjugate gradient method, then for this problem we can find the global minimum 100% of the time (again based on 1000 trials). The immediate implication is that the effective efficiency of neural network processing for searching for global minima is far superior to any of the other methods considered.

IV. CONCLUSIONS

We have shown how the simulated annealing algorithm of Kirkpatrick, Gelatt, and Vecchi (1983) and the neural network processor of Hopfield (1982, 1984) and Kohonen (1977, 1984) can be reformulated to solve optimization problems of the type encountered in indirect sensing experiments. Furthermore, we have shown that our development of both algorithms allows us to control directly the "motion" in the solution space, so that it is possible to avoid trapping in local, nonoptimal, minima.

The great advantage in using the simulated annealing algorithm lies in its versatility. Virtually any constraint can be readily incorporated into the energy function. We have shown, however, that the neural network processor can be made nearly as versatile by appropriately modifying the update equation to allow differentiable constraints. The choice between which of these two methods to employ reduces to their efficiency in searching out the global minimum.

In comparing these two algorithms with a standard conjugate gradient approach for two typical moderately sized optimization problems, we find (1) the neural network to be approximately as fast as the conjugate gradient method, but without the liability of always trapping at the first minimum encountered; (2) annealing to be able to locate the optimal solution as well as the neural network, but only at the expense of a far greater computational burden. We thus conclude that neural network processing is the method of choice for optimization problems of the kind we discuss. This is especially true if parallel processors are available: the efficiency comparisons discussed here involve simulations carried out on a serial processor and thus do not take into account the inherent parallel processing structure of simulated annealing or the neural network algorithm.

The neural network's speed was explored in some detail and found to be (at least partially) due to the way the trial solution converges: the algorithm is associated with a contractor in the solution space that "pulls" the solution vector toward the fixed point (the global minimum), while the trial vector (resulting from folding the trial solution through eq. [3.2]) lying in the corresponding data space rotates upon each iter-

ation to point closer to the desired data vector (e.g., the global solution).

The optimization and inversion examples chosen in this paper were in no way meant to exhaust the class of possible problems that these algorithms can be used for. Thus, the relative computation times reported in § III should be regarded as only indicative of the possible gain one obtains by employing the neural network algorithm; for particular problems, it is advisable to first test each method using artificial data for both speed as well as robustness against error magnification. Nevertheless, our simulations indicate that although, in most cases of physical interest, both simulated annealing and the neural network are viable algorithms for searching for the global extrema of functions, neural network processing does have a substantial speed advantage. However, an essential remaining unknown is how neural network processing scales with the dimensionality of the optimization problem and with the number of extrema in the solution domain. Although an exact answer remains to be obtained, we conjecture on the basis of our experience that the efficiency of the implementation presented here will scale similarly to other stochastic optimization methods.

We would like to acknowledge very useful conversations with T. Brown, J. Denker, R. Howard, L. Jackel, S. Jackson, D. Tank, and R. Zimmer.

REFERENCES

Acton, F. S. 1970, *Numerical Methods That Work* (New York: Harper & Row).
Backus, G., and Gilbert, F. 1970, *Phil. Trans. Roy. Soc. London, A*, **266**, 123.
Bertsekas, D. P. 1982, *Constrained Optimization and Lagrange Multiplier Methods* (New York: Academic).
Bjerhammar, A. 1951, *Kungl. Tekn. Hoegsk. Handl.*, No. 49.
Chahine, M. T. 1970, *J. Atm. Sci.*, **73**, 615.
Colton, D., and Kress, R. 1983, *Integral Equation Methods in Scattering Theory* (New York: Wiley).
Frieden, B. R. 1972, *J. Opt. Soc. Am.*, **62**, 511.
Geman, S., and Geman, D. 1984, *IEEE Trans.*, **PAMI-6**, 721.
Hopfield, J. J. 1982, *Proc. Nat. Acad. Sci.*, **79**, 2554.
———. 1984, *Proc. Nat. Acad. Sci.*, **81**, 3088.
Hopfield, J. J., and Tank, D. 1985, *Biol. Cybernetics*, **52**, 141.
Jansson, P. A. 1984, in *Deconvolution, With Applications in Spectroscopy*, ed. P. A. Jansson (Orlando: Academic Press), 67.
Jaynes, E. T. 1957, *Phys. Rev.*, **106**, 620.
Jeffrey, W., and Rosner, R. 1986, *Ap. J.*, **310**, 463.
Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. 1983, *Science*, **220**, 671.
Kohonen, T. 1977, *Associative Memory* (New York: Springer-Verlag).
———. 1984, *Self-Organization and Associative Memory* (New York: Springer-Verlag).
Menke, W. 1984, *Geophysical Data Analysis: Discrete Inverse Theory* (Orlando: Academic Press).
Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. 1953, *J. Chem. Phys.*, **6**, 1087.
Moore, E. H. 1935, *Mem. Am. Phil. Soc.*, **1**, 147.
Parker, R. L. 1977, *Ann. Rev. Earth Planet. Sci.*, **5**, 35.
Penrose, R. 1955, *Proc. Cambridge Phil. Soc.*, **51**, 406.
Poggio, T., Torre, V., and Koch, C. 1985, *Nature*, **317**, 314.
Press, W., Flannery, B., Teukolsky, S., and Vetterling, W. 1985, *Numerical Recipes* (New York: Cambridge University Press).
Tikhonov, A. N., and Arsenin, V. Y. 1977, *Illposed Problems* (Washington: Winston)
Twomey, S. 1977, *Introduction to the Mathematics of Inversion in Remote Sensing and Indirect Measurements* (New York: Elsevier).
Vanderbilt, D., and Louie, S. G. 1984, *J. Comp. Phys.*, **56**, 259.

W. Jeffrey and R. Rosner: Center for Astrophysics, 60 Garden street, Cambridge, MA 02138