GEOPHYSICS, VOL. 66, NO. 1 (JANUARY-FEBRUARY 2001); P. 174–187, 7 FIGS., 4 TABLES.

# Nonlinear conjugate gradients algorithm for 2-D magnetotelluric inversion

William Rodi* and Randall L. Mackie‡

**ABSTRACT**

We investigate a new algorithm for computing regularized solutions of the 2-D magnetotelluric inverse problem. The algorithm employs a nonlinear conjugate gradients (NLCG) scheme to minimize an objective function that penalizes data residuals and second spatial derivatives of resistivity. We compare this algorithm theoretically and numerically to two previous algorithms for constructing such "minimum-structure" models: the Gauss-Newton method, which solves a sequence of linearized inverse problems and has been the standard approach to nonlinear inversion in geophysics, and an algorithm due to Mackie and Madden, which solves a sequence of linearized inverse problems incompletely using a (linear) conjugate gradients technique. Numerical experiments involving synthetic and field data indicate that the two algorithms based on conjugate gradients (NLCG and Mackie-Madden) are more efficient than the Gauss-Newton algorithm in terms of both computer memory requirements and CPU time needed to find accurate solutions to problems of realistic size. This owes largely to the fact that the conjugate gradients-based algorithms avoid two computationally intensive tasks that are performed at each step of a Gauss-Newton iteration: calculation of the full Jacobian matrix of the forward modeling operator, and complete solution of a linear system on the model space. The numerical tests also show that the Mackie-Madden algorithm reduces the objective function more quickly than our new NLCG algorithm in the early stages of minimization, but NLCG is more effective in the later computations. To help understand these results, we describe the Mackie-Madden and new NLCG algorithms in detail and couch each as a special case of a more general conjugate gradients scheme for nonlinear inversion.

## INTRODUCTION

The standard approach to solving nonlinear inverse problems in geophysics has been iterated, linearized inversion. That is, the forward function (for predicting error-free data) is approximated with its first-order Taylor expansion about some reference model; a solution of the resulting linear inverse problem is computed; the solution is then taken as a new reference model, and the process is repeated. Such schemes are generally some form of Newton's method (typically Gauss-Newton or Levenberg-Marquardt). When run to convergence, they minimize an objective function over the space of models and, in this sense, produce an optimal solution of the nonlinear inverse problem. Most inversion algorithms for magnetotelluric (MT) data have been iterated, linearized methods. For 1-D earth models, these include the algorithms of Wu (1968) and Jupp and Vozoff (1975), which obtain nonlinear least-squares solutions, and those of Smith and Booker (1988) and Constable et al. (1987), which find nonlinear least-squares solutions subject to a smoothness constraint ("regularized" solutions). Jupp and Vozoff extended their algorithm to the case of 2-D models (Jupp and Vozoff, 1977), and algorithms for finding regularized solutions of the 2-D MT problem have been presented by Jiracek et al. (1987), Madden and Mackie (1989), Rodi (1989), deGroot-Hedlin and Constable (1990), and Smith and Booker (1991). Mackie and Madden (1993) implemented an iterated, linearized inversion algorithm for 3-D MT data, as did Newman (1995) and Newman and Alumbaugh (1997) for the related problem of crosswell electromagnetic data. However, the usefulness of such algorithms in 3-D electromagnetic inverse problems has been hampered by severe computational difficulties, which we now discuss.

Manuscript received by the Editor August 11, 1998; revised manuscript received June 6, 2000.
*Massachusetts Institute of Technology, Earth Resources Laboratory, E34-458, Cambridge, Massachusetts 02139. E-mail: rodi@ mit.edu.
‡GSY-USA, Inc., PMB #643, 2261 Market Street, San Francisco, California 94114-1600. E-mail: randy@gsy-usa.com.

Compared to global optimization methods like grid search, Monte-Carlo search, and genetic algorithms, inversion methods that make use of the Jacobian (first-order derivative) of the forward function, like those cited in the previous paragraph, generally require the testing of many fewer models to obtain an optimal solution of an inverse problem. This fact is of critical importance in 2-D and 3-D electromagnetic inverse problems where the forward function entails the numerical solution of Maxwell's equations, and is the reason that iterated, linearized methods have occupied center stage in electromagnetic inversion despite their greater susceptibility to finding locally rather than globally optimal solutions. On the other hand, generation of the Jacobian in these same problems multiplies the computational burden many times over that of evaluating the forward function alone, even when efficient reciprocity techniques (Madden, 1972; Rodi, 1976; McGillivray and Oldenburg, 1990) are exploited. Moreover, iterated, linearized inversion methods, done to prescription, have the additional computational chore of solving a linear system on the model space at each iteration step. These two tasks—generating the Jacobian and linear inversion—dominate the computations in 2-D and 3-D MT inversion, where the number of data and model parameters are typically in the hundreds or thousands. The computation of optimal solutions to the 2-D MT inverse problem can require several hours of CPU time on a modern workstation, whereas computing optimal solutions of the 3-D problem is impractical on the computers widely available today.

This computational challenge has motivated various algorithmic shortcuts in 2-D and 3-D MT inversion. One approach has been to approximate the Jacobian based on electromagnetic fields computed for homogeneous or 1-D earth models, which has been used in 2-D MT inversion by Smith and Booker (1991) in their "rapid relaxation inverse" (RRI) and by Farquharson and Oldenburg (1996) for more general 2-D and 3-D electromagnetic problems. Other workers have sought approximate solutions of the linearized inverse problem. In this category is the method of Mackie and Madden (1993), which solves each step of a Gauss-Newton iteration incompletely using a truncated conjugate gradients technique. In addition to bypassing the complete solution of a large linear system, the algorithm avoids computation of the full Jacobian matrix in favor of computing only its action on specific vectors. Although not as fast as RRI, the Mackie-Madden algorithm does not employ approximations to the Jacobian and requires much less computer time and memory than the traditional iterated, linearized inversion methods (as we will demonstrate in this paper). Also in this category is the "subspace method," applied by Oldenburg et al. (1993) to dc resistivity inversion and by others to various other geophysical inverse problems. This method reduces the computational burden by solving each linearized inverse problem on a small set of judiciously calculated "search directions" in the model space.

In their use of incomplete solutions of the linearized inverse problem, the subspace and Mackie-Madden inversion methods depart from the strict schema of iterated, linearized inversion, with an accompanying reduction in the computer resources needed to solve large, nonlinear inverse problems. In this paper we investigate an approach to electromagnetic inversion that is a further departure from the geophysical tradition: *nonlinear* conjugate gradients (NLCG), or conjugate gradients applied directly to the minimization of the objective function prescribed for the nonlinear inverse problem. The use of conjugate gradients for function minimization is a well-established optimization technique (Fletcher and Reeves, 1959; Polak, 1971) and was suggested for nonlinear geophysical inverse problems by Tarantola (1987). It has been applied to varied geophysical problems, including crosswell traveltime tomography (Matarese and Rodi, 1991; Matarese, 1993), crosswell waveform tomography (Thompson, 1993; Reiter and Rodi, 1996), and dc resistivity (Ellis and Oldenburg, 1994; Shi et al., 1996).

Our investigation compares the numerical performance of three algorithms for 2-D magnetotelluric inversion: a Gauss-Newton algorithm, the Mackie-Madden algorithm, and a new NLCG algorithm. In tests involving synthetic and real data, the algorithms are applied to the minimization of a common objective function so that algorithm efficiency and accuracy can be compared directly. Rather than implement a published NLCG algorithm (e.g., Press et al., 1992), we designed our NLCG algorithm to avoid excessive evaluations of the forward problem and to fully exploit the computational techniques for Jacobian operations used in the Mackie-Madden algorithm. Conversely, we modified the original Mackie-Madden algorithm to include a preconditioner that we developed for NLCG. Given this, we can state two objectives of our study: to demonstrate quantitatively the computational advantages of the two algorithms that use conjugate gradients (Mackie-Madden and NLCG) over a traditional iterated, linearized inversion scheme (Gauss-Newton); and to determine whether the NLCG framework offers improvements over the Mackie-Madden approach as a conjugate gradients technique. Towards the latter end and as a prelude to future research on the conjugate-gradients approach to nonlinear inversion, we describe the Mackie-Madden and our new NLCG algorithms in common terms and in detail in an attempt to isolate the precise differences between them.

## PROBLEM FORMULATION

### Forward model for 2-D magnetotellurics

As is customary in 2-D magnetotellurics, we model the solid earth as a conductive halfspace, $z \geq 0$, underlying a perfectly resistive atmosphere. The electromagnetic source is modeled as a plane current sheet at some height $z = -h$. Given that the physical parameters of the earth are independent of one cartesian coordinate ($x$), Maxwell's equations decouple into transverse electric (TE) and transverse magnetic (TM) polarizations. For the purpose of calculating MT data at low frequency, it suffices to solve (see, for example, Swift, 1971)

$$\frac{\partial^2 E_x}{\partial y^2} + \frac{\partial^2 E_x}{\partial z^2} = -i\omega\mu\sigma E_x \tag{1}$$

$$\left.\frac{\partial E_x}{\partial z}\right|_{z=-h} = i\omega\mu \tag{2}$$

for the TE polarization, and

$$\frac{\partial}{\partial y}\left(\rho\frac{\partial H_x}{\partial y}\right) + \frac{\partial}{\partial z}\left(\rho\frac{\partial H_x}{\partial z}\right) = -i\omega\mu H_x \tag{3}$$

$$H_x|_{z=0} = 1 \tag{4}$$

for the TM polarization, where $E_x$ ($H_x$) is the $x$ component of the electric (magnetic induction) field, $\omega$ is angular frequency, $\mu$ is the magnetic permeability (assumed to be that of free space), $\sigma$ is the electrical conductivity, and $\rho$ is the inverse of conductivity, or resistivity.

MT data are electric-to-magnetic-field ratios in the frequency domain, which can be expressed as *complex* apparent resistivities. For the TE polarization, the complex apparent resistivity is defined as

$$\rho_{app} = \frac{i}{\omega\mu}\left(\frac{\langle E_x\rangle}{\langle H_y\rangle}\right)^2. \tag{5}$$

$\langle E_x\rangle$ denotes the value of $E_x$ at an observation site, which is usually taken to be $E_x$ at a point but, more generally, can be a spatial average of the $E_x$ field. $\langle H_y\rangle$ is an analogous functional of the $H_y$ field. We note that Maxwell's equations imply

$$H_y = \frac{1}{i\omega\mu}\frac{\partial E_x}{\partial z}. \tag{6}$$

For the TM polarization, we have

$$\rho_{app} = \frac{i}{\omega\mu}\left(\frac{\langle E_y\rangle}{\langle H_x\rangle}\right)^2 \tag{7}$$

and

$$E_y = \rho\frac{\partial H_x}{\partial z}. \tag{8}$$

We point out that the traditional *real* apparent resistivity is the modulus of $\rho_{app}$.

### Numerical modeling

To solve equations (1)–(8) approximately for a broad class of resistivity functions, the inversion algorithms in this paper employ the numerical forward modeling algorithm described by Mackie et al. (1988). In this algorithm, the halfspace $z \geq 0$ is segmented into 2-D rectangular blocks of varying dimensions, each having a constant resistivity. Spatially heterogeneous resistivity models ensue from varying the resistivities among the blocks. The blocks abutting and outside a finite region are semi-infinite. Maxwell's equations are approximated by finite-difference equations derived using the transmission-network analog of Madden (1972).

For each polarization and frequency, the finite-difference equations can be expressed as a complex system of linear equations,

$$\mathbf{Kv} = \mathbf{s}. \tag{9}$$

In the case of the TE polarization, this linear system represents equations (1) and (2) with the vector **v** comprising samples of the $E_x$ field on a grid. The complex symmetric matrix **K** and right-hand-side vector **s** are functions of frequency and the dimensions and resistivities of the model blocks. For a given observation site, the quantity $\langle E_x\rangle$ in equation (5) is calculated as a linear combination of the elements of **v**, representing some sort of linear interpolation and/or averaging of the $E_x$ field. Likewise, $\langle H_y\rangle$ is calculated as a (different) linear function of **v**, in this case representing also numerical differentiation in accordance with equation (6). Thus, the complex apparent resistivity for one site is given by the formula

$$\rho_{app} = \frac{i}{\omega\mu}\left(\frac{\mathbf{a}^{\mathrm{T}}\mathbf{v}}{\mathbf{b}^{\mathrm{T}}\mathbf{v}}\right)^2 \tag{10}$$

where **a** and **b** are given vectors. An analogous discussion applies to the TM polarization, with **v** being a discretization of the $H_x$ field and with different choices of **K**, **s**, **a**, and **b**.

### Inversion method

We can write the inverse problem as

$$\mathbf{d} = F(\mathbf{m}) + \mathbf{e}$$

where **d** is a data vector, **m** is a model vector, **e** is an error vector, and $F$ is a forward modeling function. We take $\mathbf{d} = [d^1\ \ d^2\ \cdots\ d^N]^{\mathrm{T}}$ with each $d^i$ being either the log amplitude or phase of $\rho_{app}$ for a particular polarization (TE or TM), observation site, and frequency ($\omega$). We take $\mathbf{m} = [m^1\ \ m^2\ \cdots\ m^M]^{\mathrm{T}}$ to be a vector of parameters that define the resistivity function. Being consistent with the numerical forward modeling scheme, we let $M$ be the number of model blocks and each $m^j$ be the logarithm of resistivity ($\log\rho$) for a unique block. Given these definitions of **d** and **m**, the function $F$ is defined implicitly by equations (9) and (10).

We solve the inverse problem in the sense of Tikhonov and Arsenin (1977), taking a "regularized solution" to be a model minimizing an objective function, $\Psi$, defined by

$$\Psi(\mathbf{m}) = (\mathbf{d} - F(\mathbf{m}))^{\mathrm{T}}\mathbf{V}^{-1}(\mathbf{d} - F(\mathbf{m})) + \lambda\mathbf{m}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{Lm} \tag{11}$$

for given $\lambda$, **V**, and **L**. The regularization parameter, $\lambda$, is a positive number. The positive-definite matrix **V** plays the role of the variance of the error vector **e**. The second term of $\Psi$ defines a stabilizing functional on the model space. In this study we choose the matrix **L** to be a simple, second-difference operator such that, when the grid of model blocks is uniform, **Lm** approximates the Laplacian of $\log\rho$.

The remainder of this paper deals with numerical algorithms for minimizing $\Psi$.

## MINIMIZATION ALGORITHMS

We will consider three numerical algorithms for minimizing the objective function $\Psi$ with respect to **m**: the Gauss-Newton method, the method of Mackie and Madden (1993), and nonlinear conjugate gradients. For the remainder of this paper, we will label our particular implementation of these algorithms as GN, MM, and NLCG, respectively. Each algorithm generates a sequence of models $\mathbf{m}_0, \mathbf{m}_1, \ldots$, with the hope that $\Psi(\mathbf{m}_\ell) \to \min_{\mathbf{m}}\Psi(\mathbf{m})$ as $\ell \to \infty$.

To describe the three algorithms in detail, we introduce the following notations. The gradient and Hessian of the objective function are the $M$-dimensional vector **g** and $M \times M$ symmetric matrix **H** defined by

$$g^j(\mathbf{m}) = \partial j\Psi(\mathbf{m})$$
$$H^{jk}(\mathbf{m}) = \partial_j\partial_k\Psi(\mathbf{m}), \quad j,k = 1,\ldots,M$$

where $\partial_j$ signifies partial differentiation with respect to the $j$th argument of a function [reading $\Psi(m)$ as $\Psi(m^1, m^2, \ldots, m^M)$].

Let $\mathbf{A}$ denote the Jacobian matrix of the forward function $F$:

$$A^{ij}(\mathbf{m}) = \partial_j F^i(\mathbf{m}), \quad i = 1, \ldots, N; \quad j = 1, \ldots, M.$$

Given equation (11), we have

$$\mathbf{g}(\mathbf{m}) = -2\mathbf{A}(\mathbf{m})^{\mathrm{T}}\mathbf{V}^{-1}(\mathbf{d} - F(\mathbf{m})) + 2\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{m} \tag{12}$$

$$\mathbf{H}(\mathbf{m}) = 2\mathbf{A}(\mathbf{m})^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{A}(\mathbf{m}) + 2\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L} - 2\sum_{i=1}^{N} q^i \mathbf{B}_i(\mathbf{m}) \tag{13}$$

where $\mathbf{B}_i$ is the Hessian of $F^i$ and $\mathbf{q} = \mathbf{V}^{-1}(\mathbf{d} - F(\mathbf{m}))$.

We also define an approximate objective function and its gradient and Hessian based on linearization of $F$. For linearization about a model $\mathbf{m}_{ref}$, define

$$\tilde{F}(\mathbf{m}; \mathbf{m}_{ref}) = F(\mathbf{m}_{ref}) + \mathbf{A}(\mathbf{m}_{ref})(\mathbf{m} - \mathbf{m}_{ref})$$

$$\tilde{\Psi}(\mathbf{m}; \mathbf{m}_{ref}) = (\mathbf{d} - \tilde{F}(\mathbf{m}; \mathbf{m}_{ref}))^{\mathrm{T}}\mathbf{V}^{-1}(\mathbf{d} - \tilde{F}(\mathbf{m}; \mathbf{m}_{ref})) + \lambda\mathbf{m}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{m}.$$

It is easy to show that the gradient and Hessian of $\tilde{\Psi}$ are given by the expressions

$$\tilde{\mathbf{g}}(\mathbf{m}; \mathbf{m}_{ref}) = -2\mathbf{A}(\mathbf{m}_{ref})^{\mathrm{T}}\mathbf{V}^{-1}(\mathbf{d} - \tilde{F}(\mathbf{m}; \mathbf{m}_{ref})) + 2\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{m}$$

$$\tilde{\mathbf{H}}(\mathbf{m}_{ref}) = 2\mathbf{A}(\mathbf{m}_{ref})^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{A}(\mathbf{m}_{ref}) + 2\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}. \tag{14}$$

$\tilde{\Psi}$ is quadratic in $\mathbf{m}$ (its first argument), $\tilde{\mathbf{g}}$ is linear in $\mathbf{m}$, and $\tilde{\mathbf{H}}$ is independent of $\mathbf{m}$. In fact,

$$\tilde{\Psi}(\mathbf{m}; \mathbf{m}_{ref}) = \Psi(\mathbf{m}_{ref}) + \mathbf{g}(\mathbf{m}_{ref})^{\mathrm{T}}(\mathbf{m} - \mathbf{m}_{ref}) + \frac{1}{2}(\mathbf{m} - \mathbf{m}_{ref})^{\mathrm{T}}\tilde{\mathbf{H}}(\mathbf{m}_{ref})(\mathbf{m} - \mathbf{m}_{ref}) \tag{15}$$

$$\tilde{\mathbf{g}}(\mathbf{m}; \mathbf{m}_{ref}) = \mathbf{g}(\mathbf{m}_{ref}) + \tilde{\mathbf{H}}(\mathbf{m}_{ref})(\mathbf{m} - \mathbf{m}_{ref}). \tag{16}$$

Clearly $\tilde{F}(\mathbf{m}_{ref}; \mathbf{m}_{ref}) = F(\mathbf{m}_{ref})$, $\tilde{\Psi}(\mathbf{m}_{ref}; \mathbf{m}_{ref}) = \Psi(\mathbf{m}_{ref})$ and $\tilde{\mathbf{g}}(\mathbf{m}_{ref}; \mathbf{m}_{ref}) = \mathbf{g}(\mathbf{m}_{ref})$, but $\tilde{\mathbf{H}}(\mathbf{m}_{ref})$ is only an approximation to $\mathbf{H}(\mathbf{m}_{ref})$ obtained by dropping the last term in equation (13).

### Gauss-Newton algorithm (GN)

One can describe the Gauss-Newton iteration as recursive minimization of $\tilde{\Psi}$, i.e. the model sequence satisfies

$$\mathbf{m}_0 = \text{given}$$

$$\tilde{\Psi}(\mathbf{m}_{\ell+1}; \mathbf{m}_\ell) = \min_{\mathbf{m}} \tilde{\Psi}(\mathbf{m}; \mathbf{m}_\ell), \quad \ell = 0, 1, 2, \ldots. \tag{17}$$

A consequence of equation (17) is that the gradient vector, $\tilde{\mathbf{g}}(\mathbf{m}_{\ell+1}; \mathbf{m}_\ell)$, is zero. In light of equation (16), $\mathbf{m}_{\ell+1}$ satisfies the linear vector equation

$$\tilde{\mathbf{H}}_\ell(\mathbf{m}_{\ell+1} - \mathbf{m}_\ell) = -\mathbf{g}_\ell, \tag{18}$$

where we make the abbreviations

$$\mathbf{g}_\ell \equiv \mathbf{g}(\mathbf{m}_\ell)$$

$$\tilde{\mathbf{H}}_\ell \equiv \tilde{\mathbf{H}}(\mathbf{m}_\ell).$$

Presuming $\tilde{\mathbf{H}}_\ell$ to be nonsingular, this necessary condition is also sufficient and we can write the Gauss-Newton iteration as $\mathbf{m}_{\ell+1} = \mathbf{m}_\ell - \tilde{\mathbf{H}}_\ell^{-1}\mathbf{g}_\ell$.

Levenberg (1944) and Marquardt (1963) proposed a modification of the Gauss-Newton method in which the model increment at each step is damped. The rationale for damping is to prevent unproductive movements through the solution space caused by the nonquadratic behavior of $\Psi$ or poor conditioning of $\tilde{\mathbf{H}}$. In algorithm GN, we employ a simple version of Levenberg-Marquardt damping and replace equation (18) with

$$\left(\tilde{\mathbf{H}}_\ell + \epsilon_\ell \mathbf{I}\right)(\mathbf{m}_{\ell+1} - \mathbf{m}_\ell) = -\mathbf{g}_\ell. \tag{19}$$

Here, $\mathbf{I}$ is the identity matrix and $\epsilon_\ell$ is a positive damping parameter allowed to vary with iteration step. Since the objective function we are minimizing includes its own damping in the form of the stabilizing (last) term in equation (11), and since this term is a quadratic function of $\mathbf{m}$, a large amount of Levenberg-Marquardt damping is not needed in our problem. Algorithm GN chooses $\epsilon_\ell$ to be quite small after the first few iteration steps and is therefore not a significant departure from the Gauss-Newton method.

Our implementation of the Gauss-Newton algorithm solves equation (19) using a linear, symmetric system solver from the Linpack software library (Dongarra et al., 1979). First, the damped Hessian matrix, $\tilde{\mathbf{H}}_\ell + \epsilon_\ell \mathbf{I}$, is factored using Gaussian elimination with symmetric pivoting. The factored system is then solved with $-\mathbf{g}_\ell$ as the right-hand side vector. The Jacobian matrix, $\mathbf{A}(\mathbf{m}_\ell)$, is needed to compute $\mathbf{g}_\ell$ and $\tilde{\mathbf{H}}_\ell$ in accordance with equations (12) and (14). GN generates the Jacobian using the reciprocity method of Rodi (1976), which translates the task to that of solving a set of "pseudoforward" problems having the same structure as equation (9) (see Appendix). The memory requirements of GN are dominated by storage of the Jacobian ($NM$ real numbers) and the Hessian ($M^2$ real numbers). We note that the memory needed for forward modeling and evaluating $\Psi$ scales linearly with $N$ and $M$.

Convergence of the Gauss-Newton, or Levenberg-Marquardt, iteration implies that the sequence $\mathbf{g}_\ell$ converges to zero and thus that the solution is a stationary point of $\Psi$. Whether the stationary point corresponds to a minimum or otherwise depends on how strongly nonquadratic $\Psi$ is. When the method does find a minimum of $\Psi$, there is no assurance that it is a global minimum.

### Mackie-Madden algorithm (MM)

The second minimization algorithm we study is the algorithm first introduced by Madden and Mackie (1989) and fully implemented and more completely described by Mackie and Madden (1993). As adapted to 3-D dc resistivity inversion, the algorithm is also described by Zhang et al. (1995).

Mackie and Madden (1993) presented their algorithm as iterated, linearized inversion. Solution of the linear inverse problem at each iteration step was formulated in terms of a maximum-likelihood criterion. It is informative and well serves our purpose to recast the Mackie-Madden algorithm as a modification of the Gauss-Newton method which, like Gauss-Newton, performs a minimization of the nonquadratic objective function $\Psi$.

That is, algorithm MM is a Gauss-Newton iteration in which the linear system (18) is solved incompletely by a conjugate gradients (CG) technique. The incompleteness results from halting the conjugate gradients iteration prematurely after a prescribed number of steps, $K$. Thus, for each $\ell$, the updated model, $\mathbf{m}_{\ell+1}$, is generated as a sequence:

$$\begin{aligned}\mathbf{m}_{\ell,0} &= \mathbf{m}_\ell \\ \mathbf{m}_{\ell,k+1} &= \mathbf{m}_{\ell,k} + \alpha_{\ell,k}\mathbf{p}_{\ell,k}, \quad k = 0, 1, \ldots, K-1 \\ \mathbf{m}_{\ell+1} &= \mathbf{m}_{\ell,K}.\end{aligned}$$

For each $k$, the vector $\mathbf{p}_{\ell,k}$ is a search direction in model space and the scalar $\alpha_{\ell,k}$ is a step size. Let us make the additional abbreviation

$$\tilde{\mathbf{g}}_{\ell,k} \equiv \tilde{\mathbf{g}}(\mathbf{m}_{\ell,k}; \mathbf{m}_\ell).$$

In accordance with the CG algorithm (Hestenes and Stiefel, 1952), the step size is given by the formula

$$\alpha_{\ell,k} = -\frac{\tilde{\mathbf{g}}_{\ell,k}^{\mathrm{T}}\mathbf{p}_{\ell,k}}{\mathbf{p}_{\ell,k}^{\mathrm{T}}\tilde{\mathbf{H}}_\ell\mathbf{p}_{\ell,k}}, \tag{20}$$

which, we point out, solves the univariate minimization problem,

$$\tilde{\Psi}(\mathbf{m}_{\ell,k} + \alpha_{\ell,k}\mathbf{p}_{\ell,k}; \mathbf{m}_\ell) = \min_\alpha \tilde{\Psi}(\mathbf{m}_{\ell,k} + \alpha\mathbf{p}_{\ell,k}; \mathbf{m}_\ell).$$

The search directions are iterated as

$$\begin{aligned}\mathbf{p}_{\ell,0} &= -\mathbf{C}_\ell\mathbf{g}_\ell \\ \mathbf{p}_{\ell,k} &= -\mathbf{C}_\ell\tilde{\mathbf{g}}_{\ell,k} + \beta_{\ell,k}\mathbf{p}_{\ell,k-1}, \quad k = 1, 2, \ldots, K-1\end{aligned} \tag{21}$$

where the $M \times M$ positive-definite matrix $\mathbf{C}_\ell$ is known as a preconditioner, and where the scalars $\beta_{\ell,k}$ are calculated as

$$\beta_{\ell,k} = \frac{\tilde{\mathbf{g}}_{\ell,k}^{\mathrm{T}}\mathbf{C}_\ell\tilde{\mathbf{g}}_{\ell,k}}{\tilde{\mathbf{g}}_{\ell,k-1}^{\mathrm{T}}\mathbf{C}_\ell\tilde{\mathbf{g}}_{\ell,k-1}}.$$

The first term of equation (21) is a preconditioned steepest descent direction, which minimizes $\mathbf{p}^{\mathrm{T}}\tilde{\mathbf{g}}_{\ell,k}$, the directional derivative of $\tilde{\Psi}(\mathbf{m}; \mathbf{m}_\ell)$ at $\mathbf{m} = \mathbf{m}_{\ell,k}$, with $\mathbf{p}^{\mathrm{T}}\mathbf{C}_\ell^{-1}\mathbf{p}$ fixed. The second term modifies the search direction so that it is conjugate to previous search directions, meaning

$$\mathbf{p}_{\ell,k}^{\mathrm{T}}\tilde{\mathbf{H}}_\ell\mathbf{p}_{\ell,k'} = 0, \quad k' < k. \tag{22}$$

The final ingredient of the conjugate gradients algorithm is iteration of the gradient vectors:

$$\begin{aligned}\tilde{\mathbf{g}}_{\ell,0} &= \mathbf{g}_\ell \\ \tilde{\mathbf{g}}_{\ell,k+1} &= \tilde{\mathbf{g}}_{\ell,k} + \alpha_{\ell,k}\tilde{\mathbf{H}}_\ell\mathbf{p}_{\ell,k}, \quad k = 0, 1, \ldots, K-2,\end{aligned}$$

which follows from equation (16).

The main computations entailed in algorithm MM are involved in the evaluation of the forward function, $F(\mathbf{m}_\ell)$, for each $\ell$ [needed to compute $\Psi(\mathbf{m}_\ell)$ and $\mathbf{g}_\ell$], and operation with the Jacobian matrix and its transpose for each $k$ and $\ell$. Regarding the latter, let

$$\mathbf{A}_\ell \equiv \mathbf{A}(\mathbf{m}_\ell)$$

and define

$$\mathbf{f}_{\ell,k} = \mathbf{A}_\ell\mathbf{p}_{\ell,k}, \quad k = 0, 1, \ldots, K-1. \tag{23}$$

Then the denominator of equation (20) can be written

$$\mathbf{p}_{\ell,k}^{\mathrm{T}}\tilde{\mathbf{H}}_\ell\mathbf{p}_{\ell,k} = 2\mathbf{f}_{\ell,k}^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{f}_{\ell,k} + 2\lambda\mathbf{p}_{\ell,k}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{p}_{\ell,k},$$

and the iteration for gradient vectors becomes

$$\tilde{\mathbf{g}}_{\ell,0} = -2\mathbf{A}_\ell^{\mathrm{T}}\mathbf{V}^{-1}(\mathbf{d} - F(\mathbf{m}_\ell)) + 2\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{m}_\ell \tag{24}$$

$$\tilde{\mathbf{g}}_{\ell,k+1} = \tilde{\mathbf{g}}_{\ell,k} + 2\alpha_{\ell,k}\mathbf{A}_\ell^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{f}_{\ell,k} + 2\alpha_{\ell,k}\lambda\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{p}_{\ell,k}, \quad k = 0, 1, \ldots, K-2. \tag{25}$$

From equations (23)–(25), we see that $\mathbf{A}_\ell$ and $\mathbf{A}_\ell^{\mathrm{T}}$ each operate on $K$ vectors, or one each per CG step. Mackie and Madden (1993) showed that operations with the Jacobian and its transpose can be accomplished without computing the Jacobian itself. Instead, the vector resulting from either of these operations can be found as the solution of a single pseudoforward problem requiring the same amount of computation as the actual forward problem, $F$. (We define *one* forward problem to include *all* frequencies and polarizations involved in the data vector.) The algorithms for operating with $\mathbf{A}_\ell$ and $\mathbf{A}_\ell^{\mathrm{T}}$ are detailed in the Appendix. The main memory used by MM comprises several vectors of length $N$ (e.g. $\mathbf{f}_{\ell,k}$) and $M$ (e.g. $\mathbf{p}_{\ell,k}$, $\tilde{\mathbf{g}}_{\ell,k}$, and $\mathbf{C}_\ell\tilde{\mathbf{g}}_{\ell,k}$). Our preconditioner ($\mathbf{C}_\ell$) requires no storage (see the section "Preconditioning" below). Thus, the memory needed by MM scales linearly with the number of data and model parameters, compared to the quadratic scaling for GN.

We apply algorithm MM using relatively few CG steps per Gauss-Newton step. The main purpose in doing so is to keep the computational effort needed for Jacobian operations under that which would be needed to generate the full Jacobian matrix. The Jacobian operations performed in $K$ CG steps of MM require computations equivalent to solving $2K$ forward problems, as indicated above. The computational effort needed to generate the full Jacobian matrix is harder to characterize in general but, in the usual situation where the station set is common for all frequencies and polarizations, amounts to one forward problem per station. Therefore, MM will do less computation (related to the Jacobian) per Gauss-Newton step than GN when $K$ is less than half the number of stations. Additonally, algorithm MM avoids the factorization of $\tilde{\mathbf{H}}$. Truncating the CG iteration also effects a kind of damping of the Gauss-Newton updates, achieving similar goals as Levenberg-Marquardt damping. It is for this reason that algorithm MM solves the undamped system (18), rather than system (19).

### Nonlinear conjugate gradients (NLCG)

In algorithm MM, the method of conjugate gradients was applied inside a Gauss-Newton–style iteration to incompletely solve a linear system or, equivalently, to incompletely minimize a quadratic approximation to the objective function. Nonlinear conjugate gradients (see, for example, Luenberger, 1984) directly solve minimization problems that are not quadratic, abandoning the framework of iterated, linearized inversion. Algorithm NLCG employs the Polak-Ribiere variant of nonlinear conjugate gradients (Polak, 1971) to minimize the objective function $\Psi$ of equation (11).

The model sequence for nonlinear CG is determined by a sequence of univariate minimizations, or line searches, along computed search directions:

$$\begin{aligned} \mathbf{m}_0 &= \text{given} \\ \Psi(\mathbf{m}_\ell + \alpha_\ell \mathbf{p}_\ell) &= \min_\alpha \Psi(\mathbf{m}_\ell + \alpha \mathbf{p}_\ell) \\ \mathbf{m}_{\ell+1} &= \mathbf{m}_\ell + \alpha_\ell \mathbf{p}_\ell, \quad \ell = 0, 1, 2, \ldots. \end{aligned} \tag{26}$$

The search directions are iterated similarly to linear CG:

$$\begin{aligned} \mathbf{p}_0 &= -\mathbf{C}_0 \mathbf{g}_0 \\ \mathbf{p}_\ell &= -\mathbf{C}_\ell \mathbf{g}_\ell + \beta_\ell \mathbf{p}_{\ell-1}, \quad \ell = 1, 2, \ldots \end{aligned} \tag{27}$$

where, in the Polak-Ribiere technique,

$$\beta_\ell = \frac{\mathbf{g}_\ell^{\mathrm{T}} \mathbf{C}_\ell (\mathbf{g}_\ell - \mathbf{g}_{\ell-1})}{\mathbf{g}_{\ell-1}^{\mathrm{T}} \mathbf{C}_{\ell-1} \mathbf{g}_{\ell-1}}.$$

The quantity $-\mathbf{C}_\ell \mathbf{g}_\ell$ is again the (preconditioned) steepest descent direction, minimizing the directional derivative of $\Psi$ evaluated at $\mathbf{m}_\ell$. Unlike linear CG, the search directions are not necessarily conjugate with respect to some fixed matrix, as in equation (22), but they do satisfy the weaker condition

$$\mathbf{p}_\ell^{\mathrm{T}} (\mathbf{g}_\ell - \mathbf{g}_{\ell-1}) = 0, \quad \ell > 0. \tag{28}$$

The minimization problem, equation (26), is not quadratic and requires some iterative technique to solve. Since it involves only a single unknown, it is tempting to attack the problem as one of global optimization, i.e., finding a global minimum of $\Psi$ with respect to $\alpha$. Doing so would gain one advantage over the Gauss-Newton method, which makes no attempt to distinguish local from global minima. However, global optimization potentially leads to many forward problem calculations per NLCG step. Given the computational intensity of the MT forward problem, algorithm NLCG does not attempt global line minimization but approaches equation (26) with computational parsimony as a primary consideration.

Our line search algorithm is a univariate version of the Gauss-Newton method, with certain modifications. To describe it efficiently, we denote the univariate function to be minimized as $\Phi_\ell$ and its Gauss-Newton approximation as $\tilde{\Phi}_\ell$:

$$\begin{aligned} \Phi_\ell(\alpha) &\equiv \Psi(\mathbf{m}_\ell + \alpha \mathbf{p}_\ell) \\ \tilde{\Phi}_\ell(\alpha; \mathbf{m}_{ref}) &\equiv \tilde{\Psi}(\mathbf{m}_\ell + \alpha \mathbf{p}_\ell; \mathbf{m}_{ref}). \end{aligned}$$

Our line search generates a sequence of models

$$\mathbf{m}_{\ell,k} = \mathbf{m}_\ell + \alpha_{\ell,k} \mathbf{p}_\ell, \quad k = 0, 1, 2, \ldots,$$

where

$$\begin{aligned} \alpha_{\ell,0} &= 0 \\ \tilde{\Phi}_\ell(\alpha_{\ell,k+1}; \mathbf{m}_{\ell,k}) &= \min_\alpha \tilde{\Phi}_\ell(\alpha; \mathbf{m}_{\ell,k}), \quad k = 0, 1, 2, \ldots. \end{aligned} \tag{29}$$

Since $\tilde{\Psi}(\mathbf{m}; \mathbf{m}_{\ell,k})$ is quadratic in $\mathbf{m}$, $\tilde{\Phi}_\ell(\alpha; \mathbf{m}_{\ell,k})$ is quadratic in $\alpha$ and it is easy to show that the minimization in equation (29) is solved by

$$\alpha_{\ell,k+1} = \alpha_{\ell,k} - \frac{\mathbf{g}_{\ell,k}^{\mathrm{T}} \mathbf{p}_\ell}{\mathbf{p}_\ell^{\mathrm{T}} \tilde{\mathbf{H}}_{\ell,k} \mathbf{p}_\ell}. \tag{30}$$

Here we define

$$\begin{aligned} \mathbf{g}_{\ell,k} &\equiv \mathbf{g}(\mathbf{m}_{\ell,k}) \\ \tilde{\mathbf{H}}_{\ell,k} &\equiv \tilde{\mathbf{H}}(\mathbf{m}_{\ell,k}). \end{aligned}$$

Our modifications of this Gauss-Newton scheme are

1) We keep track of the best (smallest $\Psi$) model encountered in the line search. Let us denote this as $\mathbf{m}_{\ell,best} \equiv \mathbf{m}_\ell + \alpha_{\ell,best} \mathbf{p}_\ell$.
2) If $\Phi_\ell$ increases during the iteration ($\Phi_\ell(\alpha_{\ell,k}) > \Phi_\ell(\alpha_{\ell,k-1})$), we calculate the next step-size by bisection:
$$\alpha_{\ell,k+1} = \frac{1}{2}(\alpha_{\ell,k} + \alpha_{\ell,best}). \tag{31}$$
3) On the second or later steps of a line search, if the current and previous best models bracket a minimum, in the sense that (prime denotes derivative)
$$\Phi'_\ell(\alpha_{\ell,best}) \Phi'_\ell(\alpha_{\ell,k}) < 0,$$
then, instead of equation (30), $\alpha_{\ell,k+1}$ is calculated so as to yield the local minimum of a cubic approximation to $\Phi_\ell(\alpha)$. The cubic approximation matches $\Phi_\ell$ and $\Phi'_\ell$ at $\alpha = \alpha_{\ell,k}$ and $\alpha = \alpha_{\ell,best}$.

The line search is deemed to *converge* when the estimated value of the objective function for $\alpha_{\ell,k+1}$, predicted by the quadratic or cubic approximation as appropriate, agrees with $\Phi_\ell(\alpha_{\ell,k+1})$ within some prescribed tolerance. In the usual case of a Gauss-Newton update, the convergence condition is

$$|\Phi_\ell(\alpha_{\ell,k+1}) - \tilde{\Phi}_\ell(\alpha_{\ell,k+1}; \mathbf{m}_{\ell,k})| \le \tau \Phi_\ell(\alpha_{\ell,k+1})$$

where $\tau \ll 1$ is the tolerance. The line search is deemed to *fail* if it does not converge within a prescribed maximum number of steps, or if $\Phi_\ell(\alpha_{\ell,k+1}) > 1.5 \Phi_\ell(\alpha_{\ell,best})$ occurs. In any case, the final result of the $\ell$th line search is taken as the best model found:

$$\mathbf{m}_{\ell+1} = \mathbf{m}_{\ell,best}.$$

If the line search converged, the new search direction, $\mathbf{p}_{\ell+1}$, is computed with equation (27). If it failed, $\mathbf{p}_{\ell+1}$ is taken as the steepest descent direction [first term of equation (27)], breaking the conjugacy with previous search directions.

The main computations of algorithm NLCG are similar to those of MM. To evaluate $\mathbf{g}_{\ell,k}$ and $\mathbf{p}_\ell^{\mathrm{T}} \tilde{\mathbf{H}}_{\ell,k} \mathbf{p}_\ell$ in equation (30) entails the computation of vectors $\mathbf{A}_{\ell,k}^{\mathrm{T}} \mathbf{V}^{-1} (\mathbf{d} - F(\mathbf{m}_{\ell,k}))$ and $\mathbf{A}_{\ell,k} \mathbf{p}_\ell$ [where $\mathbf{A}_{\ell,k} \equiv \mathbf{A}(\mathbf{m}_{\ell,k})$]. Computing $\alpha_{\ell,k+1}$ by cubic interpolation, however, does not require the second derivative of $\tilde{\Phi}_\ell$, in which case $\mathbf{A}_{\ell,k} \mathbf{p}_\ell$ is not done. The same pseudoforward algorithms as in MM are used in NLCG to perform Jacobian operations (see Appendix). NLCG, unlike MM, evaluates the forward function for each model update. Therefore, each line search step in NLCG solves the equivalent of two or three forward problems. The memory requirements of NLCG are also similar to MM, scaling linearly with $N$ and $M$.

We close our description of NLCG by pointing out a potential pitfall and related computational benefit of the line search stopping condition. Our condition compares $\Psi$ at the newest model, $\mathbf{m}_{\ell,k+1}$, to the quadratic or cubic approximation extrapolated from the previous model, $\mathbf{m}_{\ell,k}$. The pitfall is that agreement between these does not guarantee that $\Psi$ is near

a minimum with respect to $\alpha$, so the line search might stop prematurely. The benefit ensues when $F$ is approximately linear between $\mathbf{m}_{\ell,k}$ and the minimizing model. In this case, the stopping condition will be met and $\mathbf{m}_{\ell,k+1}$ will be an accurate result of the line search, even though $\Psi$ and its gradient may have changed greatly from their values at $\mathbf{m}_{\ell,k}$. The search stops without additional, unnecessary computations such as an additional update ($\mathbf{m}_{\ell,k+2}$) or second derivative information at the new model (requiring $\mathbf{A}_{\ell,k+1}\mathbf{p}_\ell$). Consequently, when the nonlinear CG iteration has progressed to the point where $F$ behaves linearly in all search directions, each line minimization will require only one step ($\mathbf{m}_{\ell+1} = \mathbf{m}_{\ell,1}$) and the remaining computations will be essentially the same as the linear CG computations in MM, with the exception that the forward function $F$ is evaluated each time the model is updated.

### Preconditioning

We recall that algorithms MM and NLCG each provide for the use of a preconditioner, $\mathbf{C}_\ell$, in their respective implementations of conjugate gradients. The preconditioner can have a big impact on efficiency in conjugate gradients. Two competing considerations in its choice are the computational cost of applying the preconditioner, and its effectiveness in "steering" the gradient vector into a productive search direction.

This study compares two versions of each of algorithms MM and NLCG: one without preconditioning ($\mathbf{C}_\ell = \mathbf{I}$) and one using

$$\mathbf{C}_\ell = \left(\gamma_\ell \mathbf{I} + \lambda \mathbf{L}^T\mathbf{L}\right)^{-1}, \tag{32}$$

where $\gamma_\ell$ is a specified scalar. In the latter case, we apply the preconditioner to a vector $\mathbf{g}$ by solving the linear system for $\mathbf{h}$,

$$\left(\gamma_\ell \mathbf{I} + \lambda \mathbf{L}^T\mathbf{L}\right)\mathbf{h} = \mathbf{g}.$$

We solve this system using a (linear) conjugate gradients technique.

The rationale for equation (32) is to have an operator that can be applied efficiently and that in some sense acts like the inverse of $\tilde{\mathbf{H}}_\ell$, the approximate Hessian matrix. The efficiency of applying $\mathbf{C}_\ell$ stems from the simplicity and sparseness of the above linear system for $\mathbf{h}$. The amount of computation needed to solve the system is less than one forward function evaluation and, thus, adds little overhead to either algorithm MM or NLCG. The approximation to the inverse Hessian arises from the second term of $\mathbf{C}_\ell^{-1}$, but we also attempt to choose $\gamma_\ell$ so that the first term is of comparable size to the matrix $\mathbf{A}_\ell^T\mathbf{V}^{-1}\mathbf{A}_\ell$. In our later examples, we took $\gamma_\ell$ to be a constant (independent of $\ell$) based on the Jacobian matrix of a homogeneous medium.

### Theoretical comparison of MM and NLCG

In the three main applications of NLCG presented below ("Numerical Experiments"), updating of the step size, $\alpha_\ell$, by cubic interpolation occurred nine times, updating by bisection [formula (31)] occurred zero times, and Gauss-Newton updating [formula (30)] occurred 211 times (for a total of 220 line search steps among the three examples). Moreover, none of the line searches failed to converge within the tolerance given. The line search algorithm in NLCG is thus primarily a univariate Gauss-Newton algorithm, and it is informative to compare a simplified NLCG, in which the line search enhancements (cubic interpolation and bisection) are ignored, to MM.

Algorithms MM and NLCG both generate a doubly indexed sequence of models, $\mathbf{m}_{\ell,k}$. In MM, the slower index ($\ell$) indexes a Gauss-Newton iteration while the faster index ($k$) indexes a conjugate gradients loop. In our simplified NLCG, the opposite is the case, with $\ell$ a conjugate gradients counter and $k$ a Gauss-Newton counter. However, the algorithms perform similar calculations at each step of their respective inner loops. The difference between the algorithms can be identified with the frequency with which the following events occur: calculating the forward function ($F$); changing the search direction ($\mathbf{p}$) used in conjugate gradients; and resetting the search direction to be the steepest descent direction.

To demonstrate this, we sketch a simple algorithm having a single loop that subsumes MM and NLCG with the restricted line search. The input is a starting model, $\mathbf{m}_0$:

*Algorithm CGI* ($\mathbf{m}_0$)
  $\mathbf{m} := \mathbf{m}_0$;
  for $\ell = 0, 1, 2, \ldots$
    if *new_ref*
      $\mathbf{m}_{ref} := \mathbf{m}$;
      $\mathbf{e} := \mathbf{d} - F(\mathbf{m}_{ref})$;
    else
      $\mathbf{e} := \mathbf{e} - \alpha\mathbf{f}$;
    end
    $\mathbf{g} := -2\mathbf{A}(\mathbf{m}_{ref})^T\mathbf{V}^{-1}\mathbf{e} + 2\lambda\mathbf{L}^T\mathbf{L}\mathbf{m}$;
    $\Psi := \mathbf{e}^T\mathbf{V}^{-1}\mathbf{e} + \lambda\mathbf{m}^T\mathbf{L}^T\mathbf{L}\mathbf{m}$;
    if *new_dir*
      $\mathbf{h} := \mathbf{C}(\mathbf{m}_{ref})\mathbf{g}$;
      if *steep*
        $\beta := 0$;
      else
        $\beta := \mathbf{h}^T(\mathbf{g} - \mathbf{g}_{last})/\gamma_{last}$;
      end
      $\mathbf{p} := -\mathbf{h} + \beta\mathbf{p}$;
      $\mathbf{g}_{last} := \mathbf{g}$;
      $\gamma_{last} := \mathbf{h}^T\mathbf{g}$;
    end
    $\mathbf{f} := \mathbf{A}(\mathbf{m}_{ref})\mathbf{p}$;
    $\alpha := -\mathbf{p}^T\mathbf{g}/(\mathbf{f}^T\mathbf{V}^{-1}\mathbf{f} + \lambda\mathbf{p}^T\mathbf{L}^T\mathbf{L}\mathbf{p})$;
    $\mathbf{m} := \mathbf{m} + \alpha\mathbf{p}$;
  next $\ell$

The reader can verify that this algorithm corresponds to our mathematical descriptions of MM and NLCG. To help, we point out that the formula for $\alpha$ above corresponds to that for $\alpha_{\ell,k}$ in equation (20) (used in MM) but with that for $\alpha_{\ell,k+1} - \alpha_{\ell,k}$ in equation (30) (used in the NLCG line search). Further, CGI replaces iteration of the gradient vector, in equation (25), with iteration of an error vector, $\mathbf{e}$.

Algorithm CGI has three flags: *new_ref*, *new_dir*, and *steep*. The flag *new_ref* is set to 1 (true) if the current model is to be used as a reference model for linearization. The flag *new_dir* is 1 if the search direction is to be changed. Flag *steep* is 1 if the newly computed search direction is to be reset to the steepest descent direction, thus breaking the conjugacy condition [equation (28)]. All three flags are initialized to 1. We can characterize algorithms MM and NLCG by how these flags are changed thereafter, as shown in Table 1. Algorithm CGI above does not show tests for line search convergence or failure, but these could be the same as in NLCG.

The main computations in CGI are taken up in the evaluation of the three quantities $F(\mathbf{m}_{ref})$, $\mathbf{A}(\mathbf{m}_{ref})\mathbf{p}$ and $\mathbf{A}(\mathbf{m}_{ref})^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{e}$. Each of these quantities requires the same computational effort (see Appendix). The latter two quantities (operations with $\mathbf{A}$ and $\mathbf{A}^{\mathrm{T}}$) are done on each pass through the loop unconditionally, while the forward function is done only when *new_ref* is 1. Therefore, each model update in CGI requires computations equal to two or three forward function evaluations, depending on how *new_ref* is determined.

## NUMERICAL EXPERIMENTS

This section presents results of testing the three MT inversion algorithms described above on synthetic and field data. In each test, algorithms GN, MM and NLCG were applied to the minimization of a common objective function $\Psi$ [equation (11)] with a given data vector $\mathbf{d}$, variance matrix $\mathbf{V}$, regularization parameter $\lambda$, and regularization operator $\mathbf{L}$. The data vector and error variance matrix are described below with each example. The regularization operator for each example was the second-order finite-difference operator described earlier. To choose the regularization parameter, we ran preliminary inversions with a few values of $\lambda$ and then subjectively chose one that gave reasonable data residuals and model smoothness. We point out that none of the three inversion algorithms being tested determines $\lambda$ as an output. Various other parameters specific to the inversion algorithms were selected as follows:

1) In GN, the Levenberg-Marquardt damping parameter was set to 0.001 times the current value of the objective function: $\epsilon_\ell = 0.001\Psi(\mathbf{m}_\ell)$.
2) In NLCG, the tolerance for deciding convergence of the line minimization, $\tau$, was set to 0.003.
3) In MM and NLCG, the preconditioner was either that defined by equation (32) or, in one experiment, the identity (no preconditioning).
4) In MM, the number of conjugate gradient steps per Gauss-Newton step, $K$, was set to 3.

All results were computed on a 400-MHz Pentium II PC running the Linux operating system. The CPU times stated below are intended to reflect only the *relative* performance of the algorithms. We emphasize that the intent of these tests was to compare the speed and accuracy of GN, MM and NLCG as minimization algorithms, not the quality of the inversion models in a geophysical sense.

### Examples with synthetic data

We generated synthetic data by applying a 2-D MT forward modeling algorithm to specified models of the earth's resistivity and perturbing the results with random noise. The forward modeling algorithm we used for this purpose was intentionally different from that used in our inversion algorithms. Synthetic data were calculated using the finite-element algorithm of Wannamaker et al. (1986), whereas our inversion algorithms employ the transmission-network algorithm of Mackie et al. (1988). Each synthetic data set comprises complex apparent resistivities at multiple station locations, frequencies, and polarizations. Noise was included by adding an error to the complex logarithm of each apparent resistivity: $\log \rho_{app} + e_r + ie_i$, where $e_r$ and $e_i$ are uncorrelated samples from a Gaussian distribution having zero mean and 0.05 standard deviation (5% noise). The noise was uncorrelated between frequencies, stations, and polarizations. For comparison, the accuracy of our forward modeling algorithm is approximately 1–3% for the range of parameters (grid dimensions, frequencies, and resistivities) involved in the test problems below (Madden and Mackie, 1989).

**Table 1. How flags are set in algorithms MM and NLCG.**

| Event | MM | NLCG |
|---|---|---|
| *new_ref*=1 | Every $K$th update | Every update |
| *new_dir*=1 | Every update | When line search converges or fails |
| *steep*=1 | Every $K$th update | When line search fails |

**Model 1**.—Our first tests employ a simple resistivity model consisting of a 10 ohm-m rectangular body embedded in a 100 ohm-m background. The anomalous body has dimensions of $10 \times 10$ km and its top is 2 km below the earth's surface. The tests use synthetic data for the TM and TE polarizations at seven sites and five frequencies, yielding a total of 140 real-valued data. The frequencies range from 0.01 to 100 Hz and are evenly spaced on a logarithmic scale. The model parameterization for inversion divides the earth into a grid of blocks numbering 29 in the horizontal ($y$) direction and 27 in the vertical ($z$) direction, implying a total of 783 model parameters. The variance matrix ($\mathbf{V}$) was set to 0.0025 times the identity matrix, and the regularization parameter ($\lambda$) was chosen as 30. The starting model for each inversion was a uniform halfspace with $\rho = 30$ ohm-m.

We applied inversion algorithm GN and two versions each of MM and NLCG (with and without preconditioning) to the synthetic data from model 1. Figure 1 shows the performance of each algorithm in terms of the value of the objective function ($\Psi$) it achieves as a function of CPU time expended. CPU time used to compute the objective function for the starting model is ignored, so the first symbol plotted for each algorithm is at zero CPU time. Following this, a symbol is plotted for each iteration step of an algorithm: a Gauss-Newton step for GN and MM, a conjugate gradients step for NLCG. It is immediately evident from Figure 1 that, in both MM and NLCG, the preconditioner enhances performance significantly, especially in the case of MM. With preconditioning, MM and NLCG effectively converge to a final result in less than one minute of CPU time, while without preconditioning, they are far from convergence after a minute. We also infer from the spacing between symbols that preconditioning does not add significantly to the amount of computation in either algorithm. Henceforth, we will consider MM and NLCG only with preconditioning.

Next, we compare algorithms MM, NLCG and GN. We see from Figure 1 that GN, like MM and NLCG, effectively converges in less than one minute of CPU time. However, the rates of convergence differ amongst the algorithms. MM and NLCG reduce the objective function in the early stages of minimization at a noticeably faster rate than GN. This is quantified in Table 2, which gives the amount of CPU time expended by each algorithm to achieve various values of the objective function, determined by interpolating between iteration steps. Values of $\Psi$ are referenced to the smallest value achieved by any of the

algorithms (in this case GN), which is denoted $\Psi_{min}$ in the table. It is clear that MM and NLCG achieve each level of the objective function, down to $1.05\,\Psi_{min}$, much faster than GN, with MM being slightly faster than NLCG. In the later stages of minimization ($\Psi < 1.05\,\Psi_{min}$), NLCG becomes the most efficient, reaching within 1% of the minimum in about 20% less CPU time than GN and 40% less than MM.

Figure 2 displays one model from the model sequence generated by each of the three algorithms, i.e., the model yielding the objective function value closest to $1.01\,\Psi_{min}$. The images are truncated spatially to display the best resolved parameters; deeper blocks and those laterally away from the station array are not shown. The models from the different algorithms are clearly very similar. Each model differs (block by block over the portion shown) from the best model generated (that yielding $\Psi = \Psi_{min}$) by less than a factor of 1.3 in resistivity, or difference of 0.1 in $\log_{10}\rho$. Models later in each inversion sequence are even closer to each other and to the best model. This confirms numerically the premise of our formulation that it is the minimization criterion, and not the minimization algorithm, that determines the solution of the inverse problem.

**Table 2. CPU times versus objective function: first synthetic data set.**

| | $\Psi/\Psi_{min}$ ($\Psi_{min} = 443.82$) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2.0 | 1.5 | 1.2 | 1.1 | 1.05 | 1.02 | 1.01 |
| GN | 23 | 28 | 33 | 36 | 41 | 45 | 46 |
| MM | 9 | 12 | 14 | 21 | 31 | 48 | 61 |
| NLCG | 11 | 13 | 18 | 23 | 27 | 32 | 36 |

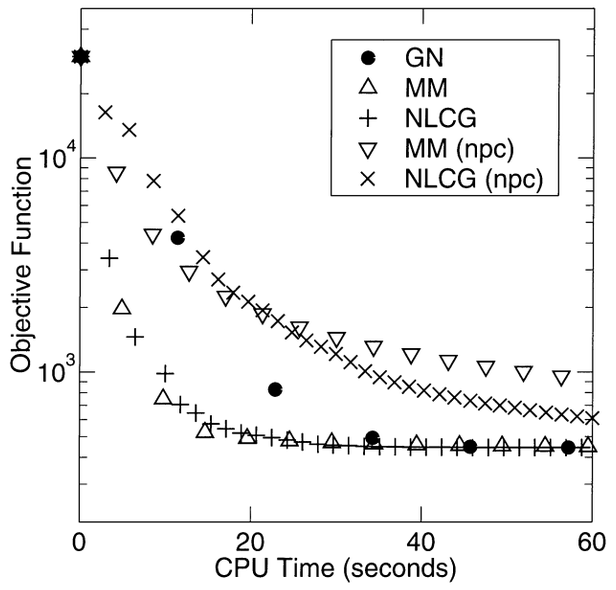


FIG. 1. Objective function versus CPU time resulting from the application of the following inversion algorithms to the first synthetic data set: the Gauss-Newton algorithm (GN, filled circles), the Mackie-Madden algorithm (MM) with and without preconditioning (up and down triangles), and nonlinear conjugate gradients (NLCG) with and without preconditioning (pluses and crosses). (The label "npc" denotes "no preconditioning.")

We note that the number of steps until convergence and the CPU time used per step differ markedly among the algorithms (Figure 1). GN requires the fewest number of steps and takes the longest for each step, whereas NLCG requires the most steps and is fastest per step. In MM and NLCG, the time per iteration step reflects largely the number of forward problems (and pseudoforward problems) invoked. Given our input parameters, algorithm MM solves seven (i.e. $1+2K$) forward problems per Gauss-Newton step (six devoted to operations with the Jacobian matrix). NLCG solves three forward problems per line search step (two for Jacobian operations). Since the stopping criterion for the line search was rather liberal ($\tau = 0.003$), all but the first three line minimizations converged in one step. (The first three each required two steps.) GN solves eight forward problems per Gauss-Newton step (seven to compute the Jacobian matrix), which is only one greater than MM. However, GN spends significant CPU time creating and factoring the Hessian matrix, which explains why its CPU time per Gauss-Newton step is so much larger than that of MM.

Also of interest in Figure 1 is the observation that MM had a larger initial reduction in the objective function than GN. This difference must be due to the difference between using Levenberg-Marquardt damping and truncated iteration for modifying the Gauss-Newton model update. Since we did not attempt to optimize the choice of $\epsilon_\ell$ in GN or $K$ in MM, we note this difference without drawing a general conclusion about the merits of the two damping techniques.

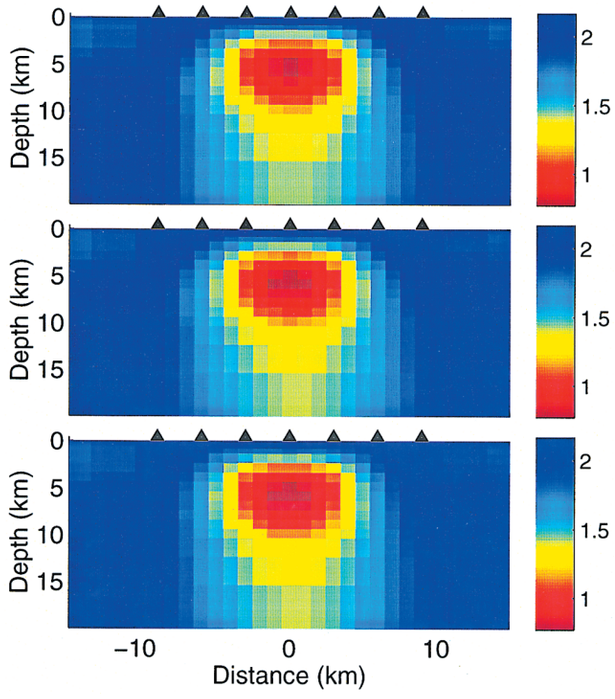


FIG. 2. Inversion models from the first synthetic data set, computed with algorithms GN (top), MM with preconditioning (middle), and NLCG with preconditioning (bottom). Resistivity scales (right) have units $\log_{10}$ ohm-meters. Station locations are marked with triangles. Each model yields $\Psi = 1.01\Psi_{min}$ (see Table 2).

**Model 2**.—The next experiment with synthetic data uses a more complicated model and larger data set. The model represents a block-faulted structure with a resistive unit exposed at the surface of the up-thrown block. The down-thrown block has the resistive unit being overlaid by a more conductive surface layer. The data set comprises complex TM and TE apparent resistivities for 12 sites and ten frequencies between 0.0032 and 100 Hz, giving a total of 480 data. The inversion model has 660 parameters corresponding to a $33 \times 20$ grid of blocks. The initial model for each algorithm was a homogeneous halfspace of 10 ohm-m. The variance matrix was the same as in the previous example, and the regularization parameter was set to 20.

The performance of the three inversion algorithms is presented in Figure 3 and Table 3. The algorithms differ in a similar manner as in the previous example. In the beginning, the conjugate gradients-based algorithms (MM and NLCG) reduce the objective function much faster than the Gauss-Newton algorithm, with MM noticeably faster than NLCG. In the later stages of minimization, MM exhibits a slow convergence rate and is overtaken first by NLCG and then by GN in reducing the objective function. MM was halted after about 1000 s, at which point $\Psi$ was 2.6% larger than $\Psi_{min}$ (which again was achieved by GN); hence the dashes in the last two columns of Table 3. We note that only six of the iterative line searches performed by NLCG took more than a single step, five taking two steps and one taking three.

**Table 3. CPU times versus objective function: second synthetic data set.**

| | $\Psi/\Psi_{min}$ ($\Psi_{min} = 1890.7$) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2.0 | 1.5 | 1.2 | 1.1 | 1.05 | 1.02 | 1.01 |
| GN | 125 | 139 | 162 | 180 | 222 | 353 | 531 |
| MM | 47 | 67 | 114 | 201 | 404 | — | — |
| NLCG | 51 | 61 | 82 | 109 | 150 | 229 | 296 |

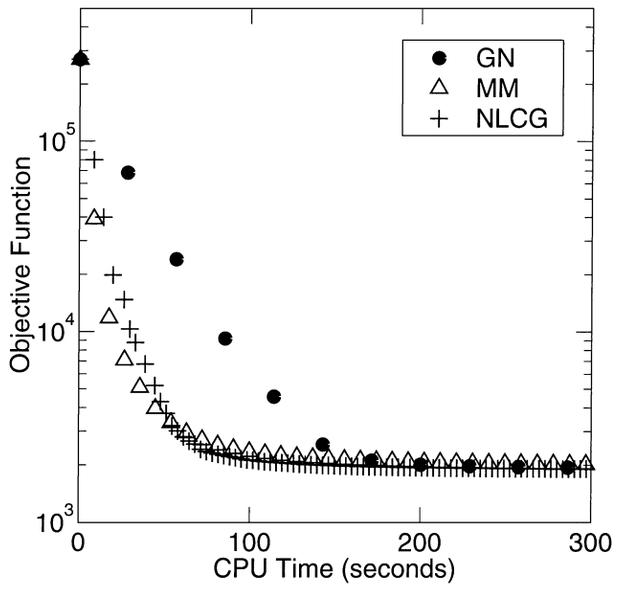


FIG. 3. Objective function versus CPU time resulting from the application of inversion algorithms GN, MM and NLCG to the second synthetic data set. Conventions are as in Figure 1.

Inversion models resulting from the second data set are shown in Figure 4. In the case of GN and NLCG, the models are for $\Psi = 1.01\ \Psi_{min}$; for MM, it is the last model generated ($\Psi = 1.026\ \Psi_{min}$). As in the previous example, there is great similarity among the models, although noticeable differences occur in the conductive overburden as well as beneath its right edge ($x \approx 5, z > 10$ km). In the distance and depth range shown, the maximum departure of the displayed GN and NLCG models from the best model computed is a factor of 2 in resistivity, whereas for MM it is a factor of 5. For both GN and NLCG, the departure drops to about 1.5 when $\Psi$ reaches $1.005\ \Psi_{min}$.

### Example with field data

Lastly, we demonstrate the various inversion algorithms on real MT data collected by P. Wannamaker in the Basin and Range (Wannamaker et al., 1997). The data set comprises TM complex apparent resistivities at 58 sites and 17 frequencies per site, for a total of 1972 real-valued data. The inversion model was parameterized with a $118 \times 25$ grid of blocks, yielding 2950 model parameters. Each algorithm was applied with a homogeneous initial model with resistivity 100 ohm-m. The diagonal elements of the variance matrix ($\mathbf{V}$) were set equal to the squares of the reported standard errors and the off-diagonal ones were set to zero. The regularization parameter was chosen as 8. The results are presented in Figures 5–7 and Table 4.

Looking at Figure 5, it is clear that NLCG and MM perform vastly better than GN on this real data set. NLCG achieved the

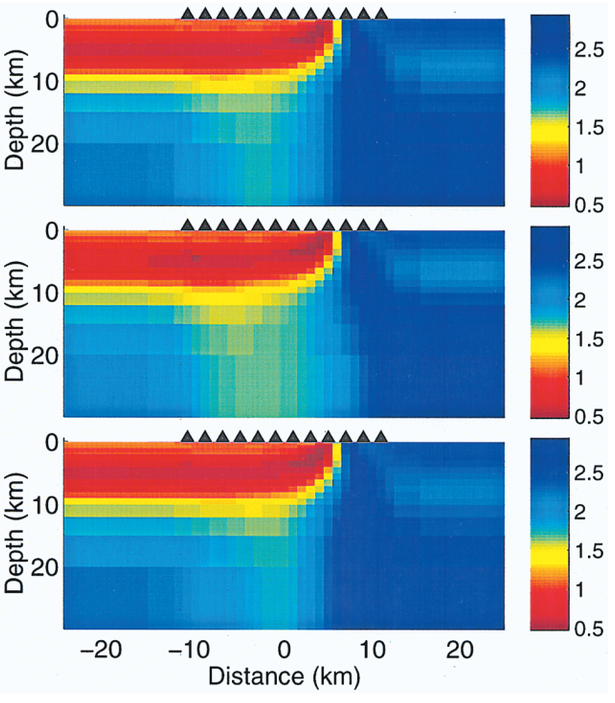


FIG. 4. Inversion models from the second synthetic data set, computed with algorithms GN (top), MM (middle), and NLCG (bottom). The resistivity models are displayed with the same conventions as Figure 2. The GN and NLCG models yield $\Psi = 1.01\ \Psi_{min}$ and the MM model $\Psi = 1.026\ \Psi_{min}$ (see Table 3).

smallest $\Psi$ among the algorithms in roughly the same amount of time needed for one step of GN. GN took over 3 CPU hours to reach within 10% of this value (Table 4), and reached only within 4.4% of $\Psi_{min}$ when it was halted after about 7 hours. These results demonstrate the poor scalability of algorithm GN with problem size. In this problem, GN solves 59 forward problems per Gauss-Newton step (compared to seven for MM) and must factor a $2950 \times 2950$ matrix (the damped Hessian). The computer memory requirements are also extensive as the Jacobian matrix contains 5.8 million (real) elements and the Hessian 8.7 million elements. MM and NLCG, on the other hand, require only several vectors of length 2950.

Figure 6 replots the MM and NLCG results on an expanded time scale so that the performance of these conjugate gradients-based algorithms can be compared. We see the same pattern as in the synthetic data examples, only this time MM performs even more favorably than NLCG in the early stages of minimization. NLCG shows faster convergence at the later stages, overtaking MM when $\Psi$ is between 1.2 and 1.1 of the minimum (Table 4). All but seven of the line searches in NLCG converged in a single step, and only the first took as many as three steps.

The MM and NLCG inversion models in Figure 7 yield $\Psi = 1.01\ \Psi_{min}$, whereas the GN model yields $\Psi = 1.044\ \Psi_{min}$. There are some significant differences between the GN model and the others in a vertical band near the rightmost station ($x \approx 60$ km), which are difficult to see since the color scale covers almost a factor of 10 000 in resistivity. Otherwise the models are very similar. The maximum discrepancy from the model yielding $\Psi = \Psi_{min}$ is about a factor of 4 for the GN model and a factor of 2 for the others.

**Table 4. CPU times versus objective function: Basin and Range data set.**

| | $\Psi/\Psi_{min}$ ($\Psi_{min} = 9408.9$) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2.0 | 1.5 | 1.2 | 1.1 | 1.05 | 1.02 | 1.01 |
| GN | 5143 | 6216 | 8245 | 11343 | 21608 | — | — |
| MM | 65 | 111 | 288 | 501 | 731 | 1232 | 1751 |
| NLCG | 158 | 224 | 342 | 425 | 536 | 712 | 827 |

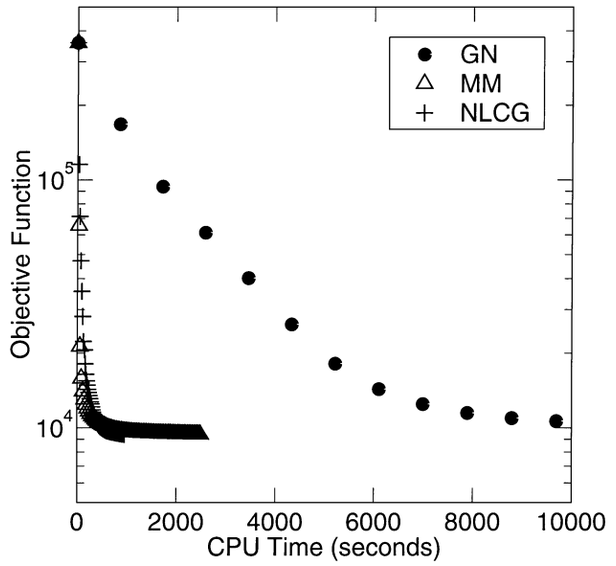


FIG. 5. Objective function versus CPU time resulting from the application of GN, MM, and NLCG to real MT data from the Basin and Range (Wannamaker et al., 1997). Conventions are as in Figure 1.

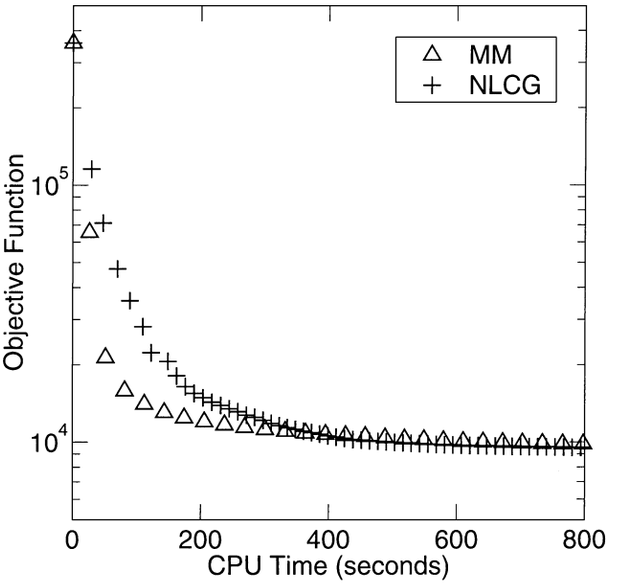


FIG. 6. The results of Figure 5 for algorithms MM and NLCG shown on an expanded time scale.

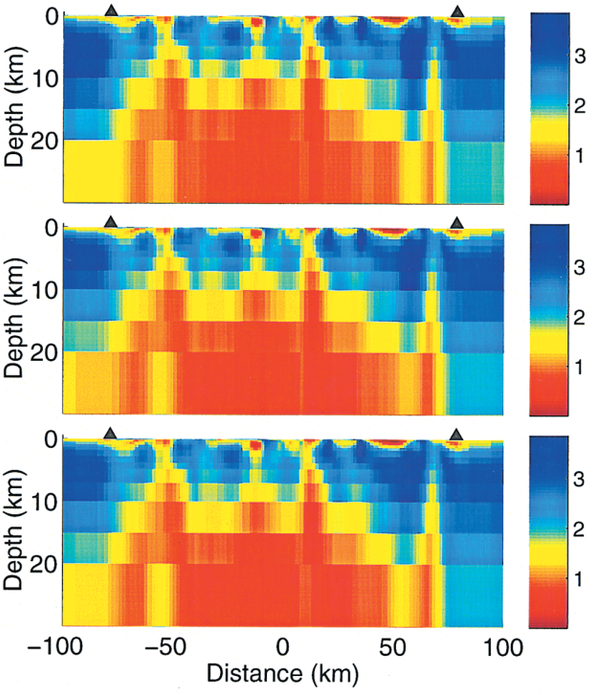


FIG. 7. Inversion models from real MT data from the Basin and Range, computed with algorithms GN (top), MM (middle), and NLCG (bottom). The models are displayed with the same conventions as Figure 2, except that only the first and last of 58 stations are marked. The MM and NLCG models yield $\Psi = 1.01\Psi_{min}$ and the GN model $\Psi = 1.044\Psi_{min}$ (see Table 4).

## DISCUSSION AND CONCLUSIONS

We have compared three minimization algorithms for computing regularized solutions of the 2-D magnetotelluric inverse problem, both theoretically and with numerical experiments involving synthetic and real data. We conclude that the conjugate gradients-based algorithms, MM and NLCG, are superior to a conventional Gauss-Newton algorithm (GN) with regard to the computational resources needed to compute accurate solutions to problems of realistic size. The explanation is that the Gauss-Newton method entails the generation of a full Jacobian matrix and the complete solution of a linearized inverse problem at each step of an iteration. MM and NLCG replace these computations with ones that scale much more favorably with problem size in both CPU and memory usage. Moreover, we enhanced performance by employing a good preconditioner in both CG-based algorithms and a very simple line minimization scheme in NLCG.

Between the Mackie-Madden algorithm and nonlinear conjugate gradients, our numerical tests do not indicate that either algorithm is clearly superior to the other. In all three tests, and especially the largest one with real data, MM reduced the objective function at a faster rate (versus CPU time) than NLCG in the early stages of minimization, whereas NLCG performed more efficiently in the later computations. The early model updates account for most of the reduction of the objective function, suggesting MM is preferable, but in our examples we found that some model parameters, well sensed by the data, change significantly in the last stages of minimization, a fact favoring NLCG. In the real data experiment, these changes amounted to as much as a factor of 30 in resistivity from the point where NLCG overtook MM in the CPU time race. (The objective function was about 1.14 times the minimum at this crossover point.) In the larger synthetic data test, MM took longer than both NLCG and GN to reach within a factor of 10 of the solution model.

We attribute the slower convergence rate of MM to the fact that it interrupts the conjugacy relation among search directions periodically, which is unnecessary near convergence when the forward function is presumably well-approximated as linear. On the other hand, NLCG is probably wasteful in the same situation by computing the nonlinear forward function after every model update. The net effect, however, is faster convergence for NLCG. It is less obvious why MM is better than NLCG in the early computations. One possibility is that the second and third steps of the line search in NLCG, when they occurred, did not reduce the objective function sufficiently to warrant doubling or tripling the CPU time of the search. Perhaps more would have been gained by changing search direction every model update, as in MM. One motivation for doing accurate line minimizations in the NLCG method is to enable the conjugacy of search directions, but conjugacy amongst the earliest search directions is not as important as for the later ones. For this same reason, interrupting conjugacy probably does not hinder MM significantly in the early stages. Lastly, it might be possible for NLCG to skip some nonlinear forward calculations even for the earlier model updates.

We recommend two topics for continued research on these CG-based algorithms for electromagnetic inversion. For both MM and NLCG, we showed that performance is enhanced significantly when a preconditioner is used. In developing these algorithms for this study, we did not put great effort into finding an optimal preconditioner. Our first recommendation is additional work on the development of an effective preconditioner for conjugate gradients-based inversion. Second, since we have seen advantages to both MM and NLCG, we recommend research on hybrid algorithms that combine elements of each. In our theoretical comparison of these algorithms, we pointed out their similarity in structure and sketched a more general algorithm (CGI) that is a template for both. In light of the discussion above, avenues for an improved CGI are more sophisticated tests for when to compute the forward function, when to change search directions, and when to revert to the steepest descent search direction.

We close by remarking that the algorithms of the type presented and tested here, while not optimal, are a clear and needed improvement over the iterated, linearized inversion algorithms in standard use. With some refinement at least, they will allow MT practitioners to use larger model grids and data sets (more frequencies and stations) in their studies, which in the past have often been reduced to accommodate the limitations of the computer. Further, it is quite obvious to us that the standard methods, like Gauss-Newton, are not practical for realistic 3-D electromagnetic problems and, even allowing for improvements in computing hardware, will not be for some time. Our results with 2-D MT suggest that conjugate gradients algorithms would be a much more feasible approach to 3-D electromagnetic inversion.

## ACKNOWLEDGMENTS

We thank Joe Matarese for steering us, and taking the first steps, in the direction of nonlinear conjugate gradients as a technique for fast inversion algorithms. We also thank Phil Wannamaker for providing us with his Basin and Range dataset for use in our comparisons of inversion algorithms. We are grateful to Greg Newman and anonymous reviewers of the paper who provided helpful and insightful comments.

## REFERENCES

Constable, S. C., Parker, R. L., and Constable, C. G., 1987, Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data: Geophysics, **52**, 289–300.

deGroot-Hedlin, C., and Constable, S., 1990, Occam's inversion to generate smooth, two-dimensional models from magnetotelluric data: Geophysics, **55**, 1613–1624.

Dongarra, J. J., Bunch, J. R., Moler, C. B., and Stewart, G. W., 1979, LINPACK: Users' guide: Soc. Ind. Appl. Math.

Ellis, R. G., and Oldenburg, D. W., 1994, The pole-pole 3-D dc-resistivity inverse problem: A conjugate gradient approach: Geophys. J. Internat., **119**, 187–194.

Farquharson, C. G., and Oldenburg, D. W., 1996, Approximate sensitivities for the electromagnetic inverse problem: Geophys. J. Internat., **126**, 235–252.

Fletcher, R., and Reeves, C. M., 1959, Function minimization by conjugate gradients: Computer J., **7**, 149–154.

Hestenes, M. R., and Stiefel, E., 1952, Methods of conjugate gradients for solving linear systems: J. Res. Nat. Bureau Stand., **49**, 409–436.

Jiracek, G. R., Rodi, W. L., and Vanyan, L. L., 1987, Implications of magnetotelluric modeling for the deep crustal environment in the Rio Grande rift: Phys. Earth Plan. Int., **45**, 179–192.

Jupp, D. L. B., and Vozoff, K., 1975, Stable iterative methods for the inversion of geophysical data: Geophys. J. Roy. Astr. Soc., **42**, 957–976.

——— 1977, Two-dimensional magnetotelluric inversion: Geophys. J. Roy. Astr. Soc., **50**, 333–352.

Levenberg, K., 1944, A method for the solution of certain non-linear problems in least squares: Quart. Appl. Math., **2**, 164–168.

Luenberger, D. G., 1984, Linear and nonlinear programming, 2nd ed.: Addison-Wesley Publ. Co.

Mackie, R. L., Bennett, B. R., and Madden, T. R., 1988, Long-period magnetotelluric measurements near the central California coast:

A land-locked view of the conductivity structure under the Pacific Ocean: Geophys. J., **95**, 181–194.
Mackie, R. L., and Madden, T. R., 1993, Three-dimensional magnetotelluric inversion using conjugate gradients: Geophys. J. Internat., **115**, 215–229.
Madden, T. R., 1972, Transmission systems and network analogies to geophysical forward and inverse problems: ONR Technical Report 72-3.
Madden, T. R., and Mackie, R. L., 1989, Three-dimensional magnetotelluric modeling and inversion, Proc. IEEE, **77**, 318–333.
Marquardt, D. W., 1963, An algorithm for least-squares estimation of nonlinear parameters, J. Soc. Indust. Appl. Math, **11**, 431–441.
Matarese, J. R., 1993, Nonlinear traveltime tomography: Ph.D. thesis, Massachusetts Institute of Technology.
Matarese, J. R., and Rodi, W. L., 1991, Nonlinear traveltime inversion of cross-well seismics: a minimum structure approach: 61st Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 917–921.
McGillivray, P. R., and Oldenburg, D. W., 1990, Methods for calculating Frechet derivatives and sensitivities for the non-linear inverse problem: A comparative study, Geophys. Prosp., **38**, 499–524.
Newman, G., 1995, Crosswell electromagnetic inversion using integral and differential equations: Geophysics, **60**, 899–911.
Newman, G. A., and Alumbaugh, D. L., 1997, Three-dimensional massively parallel electromagnetic inversion—I. Theory: Geophys. J. Internat., **128**, 345–354.
Oldenburg, D. W., McGillivray, P. R., and Ellis, R. G., 1993, Generalized subspace methods for large scale inverse problems: Geophys. J. Internat., **114**, 12–20.
Polak, E., 1971, Computational methods in optimization: A unified approach: Academic Press.
Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 1992, Numerical recipes in FORTRAN: The art of scientific computing, 2nd ed.: Cambridge Univ. Press.
Reiter, D. T., and Rodi, W., 1996, Nonlinear waveform tomography applied to crosshole seismic data: Geophysics, **61**, 902–913.
Rodi, W. L., 1976, A technique for improving the accuracy of finite element solutions for magnetotelluric data: Geophys. J. Roy. Astr. Soc., **44**, 483–506.
——— 1989, Regularization and Backus-Gilbert estimation in nonlinear inverse problems: Application to magnetotellurics and surface waves: Ph.D. thesis, Pennsylvania State Univ.
Shi, W., Rodi, W., Mackie, R. L., and Zhang, J., 1996, 3-D d.c. electrical resistivity inversion with application to a contamination site in the Aberjona watershed: Proceedings from Symposium on the Application of Geophysics to Environmental and Engineering Problems (SAGEEP): Environmental and Engineering Geophys. Soc., 1257–1267.
Smith, J. T., and Booker, J. R., 1988, Magnetotelluric inversion for minimum structure: Geophysics, **53**, 1565–1576.
——— 1991, Rapid inversion of two- and three-dimensional magnetotelluric data: J. Geophys. Res., **96**, 3905–3922.
Swift, C. M., Jr., 1971, Theoretical magnetotelluric and Turam response from two-dimensional inhomogeneities: Geophysics, **36**, 38–52.
Tarantola, A., 1987, Inverse problem theory: Elsevier.
Tikhonov, A. N., and Arsenin, V. Y., 1977, Solutions of ill-posed problems: V. H. Winston and Sons.
Thompson, D. R., 1993, Nonlinear waveform tomography: Theory and application to crosshole seismic data: Ph.D. thesis, Massachusetts Institute of Technology.
Wannamaker, P. E., Johnston, J. J., Stodt, J. A., and Booker, J. R., 1997, Anatomy of the southern Cordilleran hingeline, Utah and Nevada, from deep electrical resistivity profiling: Geophysics, **62**, 1069–1086.
Wannamaker, P. E., Stodt, J. A., and Rijo, L., 1986, Two-dimensional topographic responses in magnetotellurics modeled using finite elements: Geophysics, **51**, 2131–2144.
Wu, F. T., 1968, The inverse problem of magnetotelluric sounding: Geophysics, **33**, 972–979.
Zhang, J., Mackie, R. L., and Madden, T. R., 1995, 3-D resistivity forward modeling and inversion using conjugate gradients: Geophysics, **60**, 1313–1325.

## APPENDIX

### JACOBIAN COMPUTATIONS

The Gauss-Newton method (algorithm GN) requires the computation of each element of the Jacobian matrix, $\mathbf{A}$. The Mackie-Madden algorithm (MM) and nonlinear conjugate gradients (NLCG), in contrast, employ $\mathbf{A}$ only in the computation of quantities $\mathbf{Ap}$ and $\mathbf{A}^T\mathbf{q}$ for specific vectors $\mathbf{p}$ and $\mathbf{q}$ [e.g., equations (23) and (25)]. This Appendix describes algorithms for the computation of $\mathbf{A}$, $\mathbf{Ap}$, and $\mathbf{A}^T\mathbf{q}$.

To begin, since each datum is the real or imaginary part of a complex quantity, we will convert our problem to one involving complex variables. Let $\hat{\mathbf{d}}$ be a complex vector such that each element of $\mathbf{d}$ is the real or imaginary part of a unique element of $\hat{\mathbf{d}}$:

$$\mathbf{d} = \mathrm{Re}\,\mathbf{E}\hat{\mathbf{d}}$$

where

$$E^{ik} = \begin{cases} 1 & \text{if } d^i \equiv \mathrm{Re}\,\hat{d}^k; \\ -i & \text{if } d^i \equiv \mathrm{Im}\,\hat{d}^k; \\ 0 & \text{else.} \end{cases}$$

We will denote the dimensionality of $\hat{\mathbf{d}}$ as $\hat{N}$, where clearly $\hat{N} \le N$ in general and $N = 2\hat{N}$ just in case amplitude and phase data are included in $\mathbf{d}$ equally. We can now write the forward function $F$ as

$$F(\mathbf{m}) = \mathrm{Re}\,\mathbf{E}\hat{F}(\mathbf{m})$$

where $\hat{F}$ is a complex function. It follows that

$$\mathbf{A} = \mathrm{Re}\,\mathbf{E}\hat{\mathbf{A}}$$

with the complex matrix $\hat{\mathbf{A}}$ being the Jacobian of $\hat{F}$:

$$\hat{A}^{ij}(\mathbf{m}) = \partial_j \hat{F}^i(\mathbf{m}).$$

We also have

$$\mathbf{Ap} = \mathrm{Re}\,\mathbf{E}\hat{\mathbf{A}}\mathbf{p}$$

$$\mathbf{A}^T\mathbf{q} = \mathrm{Re}\,\hat{\mathbf{A}}^T\mathbf{E}^T\mathbf{q}.$$

Our task translates to finding $\hat{\mathbf{A}}$, $\hat{\mathbf{A}}\mathbf{p}$, and $\hat{\mathbf{A}}^T\hat{\mathbf{q}}$ where $\hat{\mathbf{q}} = \mathbf{E}^T\mathbf{q}$.

To specify $\hat{F}$, it is convenient to consider all frequencies and polarizations involved in the data vector $\mathbf{d}$ simultaneously. Let $\mathbf{v}$ be a vector comprising the parameterized $E_x$ and/or $H_x$ fields for all frequencies, and let the linear equation

$$\mathbf{K}(\mathbf{m})\mathbf{v}(\mathbf{m}) = \mathbf{s}(\mathbf{m}) \qquad \text{(A-1)}$$

denote the finite-difference form of Maxwell's equations for all relevant polarizations and frequencies. $\mathbf{K}$ is a block-diagonal matrix (when $\mathbf{v}$ is partitioned by frequencies and polarizations) and $\mathbf{s}$ comprises the right-hand-side vectors for all frequencies and polarizations. We have shown the dependence of $\mathbf{K}$ and $\mathbf{s}$, and hence $\mathbf{v}$, on the model parameter vector $\mathbf{m}$. We can now write

$$\hat{F}^i(\mathbf{m}) = \log \frac{i}{\omega_i \mu} \left( \frac{\mathbf{a}_i(\mathbf{m})^T\mathbf{v}(\mathbf{m})}{\mathbf{b}_i(\mathbf{m})^T\mathbf{v}(\mathbf{m})} \right)^2 \qquad \text{(A-2)}$$

where the vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ are chosen to extract from $\mathbf{v}$ the relevant field averages for the polarization, frequency, and observation site associated with the $i$th complex datum.

### Computation of $\hat{\mathbf{A}}$

We consider the computation of $\hat{\mathbf{A}}$ using two methods described by Rodi (1976). Differentiating equation (A-2),

$$\hat{A}^{ij} = \hat{A}_1^{ij} + \hat{A}_2^{ij}, \tag{A-3}$$

where

$$\hat{A}_1^{ij} = \left( \frac{2}{\mathbf{a}_i^{\mathrm{T}}\mathbf{v}} \partial_j \mathbf{a}_i - \frac{2}{\mathbf{b}_i^{\mathrm{T}}\mathbf{v}} \partial_j \mathbf{b}_i \right)^{\mathrm{T}} \mathbf{v}$$

and

$$\hat{A}_2^{ij} = \mathbf{c}_i^{\mathrm{T}} \partial_j \mathbf{v}, \tag{A-4}$$

where the vector $\mathbf{c}_i$ is defined by

$$\mathbf{c}_i = \frac{2}{\mathbf{a}_i^{\mathrm{T}}\mathbf{v}} \mathbf{a}_i - \frac{2}{\mathbf{b}_i^{\mathrm{T}}\mathbf{v}} \mathbf{b}_i.$$

The matrix $\hat{\mathbf{A}}_1$ accounts for the dependence of $\rho_{app}$ on $\mathbf{m}$ through the vectors $\mathbf{a}_i$ and $\mathbf{b}_i$. The matrix $\hat{\mathbf{A}}_2$ accounts for the dependence of $\mathbf{v}$ on $\mathbf{m}$. We assume the vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ and their partial derivatives can be computed with closed-form expressions so that $\hat{\mathbf{A}}_1$ can also be computed with such. We turn to the more difficult task of computing $\hat{\mathbf{A}}_2$.

From equation (A-1), we can infer

$$\mathbf{K}\partial_j \mathbf{v} = \partial_j \mathbf{s} - (\partial_j \mathbf{K})\mathbf{v}, \quad j = 1, 2, \ldots, M. \tag{A-5}$$

Again, we assume that $\mathbf{K}$, $\mathbf{s}$, and their partial derivatives are known analytically. The first method described by Rodi (1976) is to solve these $M$ "pseudoforward" problems for the vectors $\partial_j \mathbf{v}$ and substitute them into equation (A-4).

The second method of Rodi (1976) exploits the reciprocity property of the forward problem, i.e., the symmetry of $\mathbf{K}$. Solving equation (A-5) and plugging into equation (A-4), we get

$$\hat{A}_2^{ij} = \mathbf{c}_i^{\mathrm{T}} \mathbf{K}^{-1} (\partial_j \mathbf{s} - (\partial_j \mathbf{K})\mathbf{v}). \tag{A-6}$$

Let the vectors $\mathbf{u}_i$ satisfy

$$\mathbf{K}\mathbf{u}_i = \mathbf{c}_i, \quad i = 1, 2, \ldots, \hat{N}. \tag{A-7}$$

Given the symmetry of $\mathbf{K}$, we can then write equation (A-6) as

$$\hat{A}_2^{ij} = \mathbf{u}_i^{\mathrm{T}} (\partial_j \mathbf{s} - (\partial_j \mathbf{K})\mathbf{v}). \tag{A-8}$$

The second method is to solve equations (A-7) and then evaluate equation (A-8).

The matrices $\partial_j \mathbf{K}$ are very sparse since $\mathbf{K}$ is sparse and each of its elements depends on only a few of the $m^j$. The vectors $\partial_j \mathbf{s}$, $\mathbf{a}_i$, and $\mathbf{b}_i$ are likewise sparse, or zero. Therefore, in either method, construction of the right-hand-side vectors for the pseudoforward problems [equations (A-5) or (A-7)] and evaluation of the expression for $\hat{A}_2^{ij}$ [equations (A-4) or (A-8)] take relatively little computation. The major computational effort in either method is in solving the appropriate set of pseudoforward problems: equations (A-5) or (A-7). For this reason, the first method [equations (A-4) and (A-5)] is more efficient when $\hat{N} > M$ (more data than model parameters) while the second, reciprocity method [equations (A-7) and (A-8)] is more efficient when $M > \hat{N}$.

However, this last statement does not take into account the particular structure of the matrix $\mathbf{K}$ and vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ for 2-D magnetotellurics. $\mathbf{K}$ has a block-diagonal structure with each block corresponding to one polarization and frequency combination. Furthermore, the nonzero elements of $\mathbf{a}_i$ and $\mathbf{b}_i$, for any given $i$, are all associated with a common partition of $\mathbf{v}$ (since one 2-D MT datum conventionally involves only a single polarization and frequency). Therefore, only one block of each pseudoforward problem in equation (A-7) needs to be solved and, what is more, we may choose between the first and second methods independently for each polarization/frequency pair in computing its partition of $\hat{\mathbf{A}}_2$. The first (second) method is more efficient when the number of data for that polarization/frequency is larger (smaller) than the number of model parameters.

### Computation of $\hat{\mathbf{A}}\mathbf{p}$ and $\hat{\mathbf{A}}^{\mathrm{T}}\hat{\mathbf{q}}$

From equation (A-3), we have

$$\hat{\mathbf{A}}\mathbf{p} = \hat{\mathbf{A}}_1\mathbf{p} + \hat{\mathbf{A}}_2\mathbf{p}$$
$$\hat{\mathbf{A}}^{\mathrm{T}}\hat{\mathbf{q}} = \hat{\mathbf{A}}_1^{\mathrm{T}}\hat{\mathbf{q}} + \hat{\mathbf{A}}_2^{\mathrm{T}}\hat{\mathbf{q}}.$$

Again, we assume the first term of each expression can be computed explicitly and we turn our attention to the second terms.

The algorithm of Mackie and Madden (1993) for $\hat{\mathbf{A}}_2\mathbf{p}$ may be derived as follows. From equation (A-4), we have

$$\sum_j \hat{A}_2^{ij} p^j = \mathbf{c}_i^{\mathrm{T}} \mathbf{t}, \tag{A-9}$$

where the vector $\mathbf{t}$ is given by

$$\mathbf{t} = \sum_j p^j \partial_j \mathbf{v}.$$

From equation (A-5), it is clear that $\mathbf{t}$ satisfies

$$\mathbf{K}\mathbf{t} = \sum_j p^j (\partial_j \mathbf{s} - (\partial_j \mathbf{K})\mathbf{v}). \tag{A-10}$$

The algorithm for $\hat{\mathbf{A}}_2\mathbf{p}$ is to solve the single forward problem, equation (A-10), for $\mathbf{t}$ and then evaluate equation (A-9).

The Mackie-Madden method for $\hat{\mathbf{A}}_2^{\mathrm{T}}\hat{\mathbf{q}}$ can be derived similarly. From equation (A-8), we have

$$\sum_i \hat{q}^i \hat{A}_2^{ij} = \mathbf{r}^{\mathrm{T}} (\partial_j \mathbf{s} - (\partial_j \mathbf{K})\mathbf{v}), \tag{A-11}$$

where we define the vector $\mathbf{r}$ by

$$\mathbf{r} = \sum_i \hat{q}^i \mathbf{u}_i.$$

From equation (A-7), $\mathbf{r}$ satisfies

$$\mathbf{K}\mathbf{r} = \sum_i \hat{q}^i \mathbf{c}_i. \tag{A-12}$$

The algorithm for $\hat{\mathbf{A}}^{\mathrm{T}}\hat{\mathbf{q}}$ is to solve equation (A-12) and substitute into equation (A-11).

The major computation in each of these algorithms is the solution of one pseudoforward problem: for $\mathbf{r}$ in equation (A-12) or $\mathbf{t}$ in equation (A-10).