Gradient Methods

Rafael E. Banchs

INTRODUCTION

This report discuss one class of the local search algorithms to be used in the inverse modeling of the time harmonic field electric logging problem, the Gradient Methods. First, a brief analysis of Gradient Methods in general is presented. Then a more precise description of each of the specific algorithms that have been implemented is provided.

GENERAL DESCRIPTION OF GRADIENT METHODS

As it was described in [1], the Gradient Methods are local search algorithms. As their name suggests it, Gradient Methods search for local minima over the error function surface by using the information provided by the local value of the gradient. They are linear inversion methods since they only relay in the information provided by the first derivatives. Their success is always based on the assumption that very close to a minimum the error surface is concave; this is called the condition number assumption [2]. For this reason, the definition of the objective function plays an important role in the performance of the algorithm.

The general Gradient Method algorithm can be describe as an iterative procedure in which each iteration is composed by the following three basic steps:

1.- Computation of a direction of search.

In this first step, a vector $\bar{d}$ (direction of search) is computed based on the information provided by the gradient at the current location on the objective function. Basically, the gradient direction and the direction of search must form an angle greater than $\pi/2$ and smaller than $3\pi/2$. Under this

condition the direction of search is guarantied to be a descent direction, so the algorithm will move downwards along the error surface.

2.- Computation of a step size.

Once the direction of search has been computed, a real number s (step size or jump size) is computed. This value is also computed using the information provided by the gradient which some times is complemented by a more detailed exploration of the error surface along the direction of search. The procedures by which the step size can be computed are called Line Search Algorithms. Three different Line Search algorithms will be discussed latter.

3.- Updating of the model parameters.

Once the direction of search and the step size have been determined, the new model can be obtained by 'jumping' in the model parameter space. Basically, the new location in the space is determined by the previous location plus the step size times the direction of search. The new model is then used as the staring point for a new iteration and the procedure continues until a convergence condition is achieved.

THE OBJECTIVE FUNCTION

As it was mentioned before, the definition of the objective function plays a very important role in the performance of the optimization process. In the case of Gradient Methods, it is very clear that continuous first derivatives are to be desired; and, in general, smoothness is also a desired property of the error surface. For these reasons the most common used objective function is the mean square error function. Although any even order error will provide continuous first derivatives (if the function itself has continuous derivatives, of course), the square error will be the smoothest of all. Based on these facts, we will define the objective function as the mean square error between the given data and the model response.

In this way, for a given set of measurements $\{M_1, M_2, M_3 \ldots M_N\}$, the objective function is defined as follows:

$$E(\bar{x}) = \frac{1}{N}\sum_{k=1}^{N}(M_k - f_k(\bar{x}))^2 \tag{1}$$

where $\bar{x}$ represents the model (set of conductivities), which is given by:

$$\bar{x} = \begin{bmatrix} x_1 & x_2 & \ldots & x_M \end{bmatrix}^T = \begin{bmatrix} \ln(\sigma_1) & \ln(\sigma_2) & \ldots & \ln(\sigma_M) \end{bmatrix}^T \tag{2}$$

Also in (1), $f_k(\bar{x})$ represents the equivalent measurement to M_k but computed for the model $\bar{x}$. In other words; if for example, M_k is the real part of the logging tool reading at 100 Hz for the actual formation; then $f_k(\bar{x})$ must be the real part of the tool reading at 100 Hz for model $\bar{x}$. In this way:

$$M_k = f_k(\bar{x}^*) \qquad \text{for } k = 1,2,...N \tag{3}$$

where $\bar{x}^*$ is the actual formation. Notice, however, that (3) will not necessarily hold if the formation's model used for the inversion has a different radii distribution or a different number of zones from the actual formation.

By differentiating (1), the gradient of the error surface can be computed. Its value at location $\bar{x}$ is given by:

$$\nabla E(\bar{x}) = \left[\left.\frac{\partial E}{\partial x_1}\right|_{\bar{x}} \quad \left.\frac{\partial E}{\partial x_2}\right|_{\bar{x}} \quad \ldots \quad \left.\frac{\partial E}{\partial x_M}\right|_{\bar{x}} \right]^T \tag{4.a}$$

$$\text{where: } \left.\frac{\partial E}{\partial x_j}\right|_{\bar{x}} = -\frac{2\,\sigma_j}{N}\sum_{k=1}^{N}\left[(M_k - f_k(\bar{x}))\left.\frac{\partial f_k}{\partial \sigma_j}\right|_{\bar{x}} \right] \qquad \text{for } j = 1,2,...M \tag{4.b}$$

Sometimes, the error function defined in (1) can be inappropriate for the inversion process. That is the case when the data measurement values differ in orders of magnitude. In such a case, those error terms due to small data values are very unlikely to be minimized because of their little influence on the overall error. This situation can be improved by weighting each of the error terms in such a way that their contribution to the total error value would be more equitable. Then, an alternative definition of the objective function could be:

$$E(\bar{x}) = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{M_k - f_k(\bar{x})}{M_k}\right)^2 \tag{5}$$

and its gradient at location $\bar{x}$ would be given by:

$$\nabla\bar{E}(\bar{x}) = \left[\left.\frac{\partial E}{\partial x_1}\right|_{\bar{x}} \quad \left.\frac{\partial E}{\partial x_2}\right|_{\bar{x}} \quad \dots \quad \left.\frac{\partial E}{\partial x_M}\right|_{\bar{x}}\right]^T \tag{6.a}$$

where: $$\left.\frac{\partial E}{\partial x_j}\right|_{\bar{x}} = -\frac{2\,\sigma_j}{N}\sum_{k=1}^{N}\left[\left(\frac{M_k - f_k(\bar{x})}{M_k}\right)\left.\frac{\partial f_k}{\partial \sigma_j}\right|_{\bar{x}}\right] \qquad \text{for} \quad j = 1,2,...M \tag{6.b}$$

LINE SEARCH ALGORITHMS

Once the gradient of the error function at the current location is known, a descent direction of search can be obtained relatively easy. But how far in that direction it is going to be advanced is crucial for the success of the search. The Line Search algorithm is the procedure used to compute the value of the step size. In this section, the three implemented Line Search algorithms are going to be discussed. They are the fixed step size, the inexact line search by modified backtracking and the exact line search.

1.- Fixed Step Size Line Search.

It is the most simple, and most ineffective, of all Linear Search Algorithms. In it, the size of the jump is the same at every iteration. As it will become clear later, this type of Linear Search can only be used with Gradient Methods that do not use normalized directions of search. It also has the disadvantage of producing very slow convergence rates and presents the potential risk of not achieving any convergence at all by getting the optimization process trapped in undesired oscillations (this last situation is illustrated in Figure 1). For all this reasons, fixed step size line search is rarely used in practice.

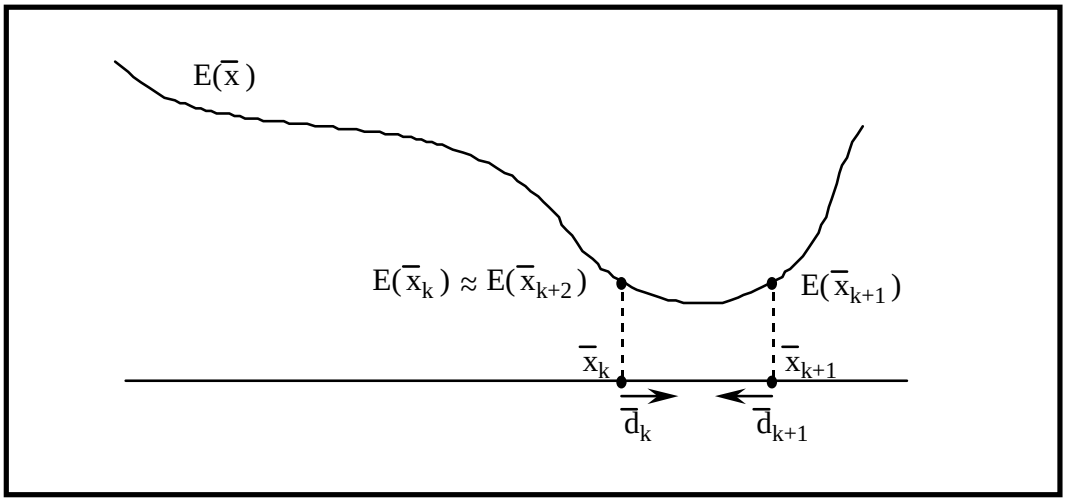

Figure 1: Undesired oscillation under Fixed Step Size Line Search.

2.- Inexact Line Search by Modified Backtracking.

Inexact line search algorithms just look, along the direction of search, for a new set of model parameters such that the new value of the objective function is substantially reduced with respect to the previous one. In other words, the size of the jump is selected in order to approximately minimize the objective function along the direction of search.

One common inexact line search algorithm is the backtracking procedure [2]. In it, the objective function is evaluated, first at a point far from the current location and then closer and closer until the backtracking condition is met; then the backtracking is completed. The backtracking condition is given by:

$$E(\bar{x}+s\bar{d}) \le E(\bar{x})+\alpha\, s\, \nabla\bar{E}(\bar{x})^{T}\,\bar{d} \qquad \text{with } 0<\alpha<0.5 \tag{7}$$

where $\bar{x}$ represents the current model parameters, $\bar{d}$ is the direction of search, s is the size of the jump and α is a pre-defined constant.

The first evaluation of the backtracking condition (7) is performed with s = 1, and then successive evaluations are performed by updating s according to (8) until (7) holds.

$$s := \beta\, s \qquad \text{with } 0<\beta<1.0 \tag{8}$$

where β is another pre-defined constant.

Notice that what the backtracking procedure is actually doing is to compare the objective function to a linear approximation of it at some points along the direction of search; and stops when it has found a point at which the objective function is guarantied to be 'substantially' smaller than at the current model location. Here the significance of 'substantially' is totally determined by the value of the constant α. Notice also that if the direction of search is a descent direction ($\nabla\bar{E}(\bar{x})^T\,\bar{d} < 0$), then the backtracking procedure is guarantied to stop because for s small enough:

$$E(\bar{x}+s\bar{d}) \approx E(\bar{x}) + s\,\nabla\bar{E}(\bar{x})^T\,\bar{d} < E(\bar{x}) + \alpha\, s\,\nabla\bar{E}(\bar{x})^T\,\bar{d} \qquad (9)$$

which is backtracking condition itself.

As it can be seen from (8), one of the main advantages of backtracking is that only evaluations of the objective function, and not its gradient, are required during the line search procedure. This is really important when the computation of the derivatives is computationally expensive. However, pure backtracking as it was defined above presents one disadvantage. The problem occurs when the local search algorithm is reaching a minimum. As in each iteration backtracking starts with a relatively large jump (s=1), many evaluations are required to get to the step size that satisfies the backtracking condition. In fact the closer is ß to 1.0, the worse gets the problem. Figure 2.a illustrates such a situation. In our modified version of backtracking, this problem is solved by adding 'memory' to the size of the jump s. In this way, in every new iteration the backtracking procedure starts with the step size of the previous iteration. So, when the local search algorithm is approaching a solution, the speed of convergence is improved.

Nevertheless, this new feature leads to another kind of problem, which is illustrated in Figure 2.b. The new situation occurs when the local search algorithm is approximating a saddle point and the value of the step size has been substantially reduced. Under such an scenario and given the modification above introduced, two different things can happen. Either the algorithm will continue a very slow search after passing the saddle point; or ,what is worse, it can converge to the saddle point 'thinking' it is a minimum. To avoid both of those situations there must be a

mechanism that eventually allows the step size to increase its value. In our modified version of backtracking, this is accomplished by using a forward-tracking procedure. Under this scheme, if the backtracking condition is met during the first evaluation of the objective function; instead of stopping the line search, then the size of the jump is updated as in (8) but by using 1/ß. This is done until the backtracking condition ceases to hold. In this way, the local search algorithm will speed up after passing the saddle point. Notice, however, that the forward-tracking procedure will slow down a little bit the speed of convergence at minima; but it will be still faster than pure backtracking.

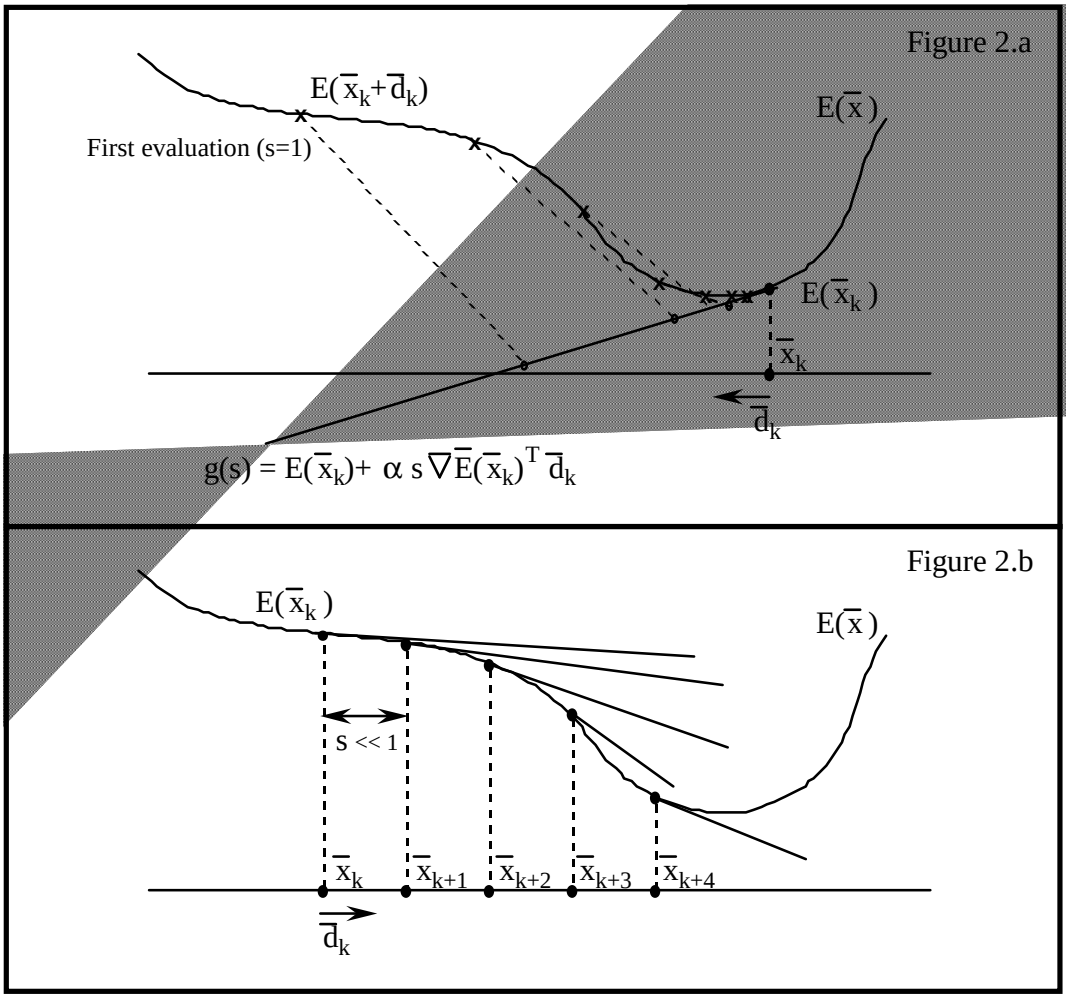

Figure 2: Undesired situations under Inexact Line Search with Backtracking.

3.- Exact Line Search.

In the exact line search algorithms the size of the jump is chosen in order to minimize the objective function along the direction of search. In practice, exact line search algorithms are generally used when the computational cost of evaluating the gradient is greater than the cost of minimizing the problem along the direction of search. Although this is not the case in our particular kind of problem, an exact line search algorithm has been implemented because, as it will be seen later, the conjugate gradients method requires an exact line search procedure.

The implemented algorithm consists of a loop of inexact line searches as the one described before. Although all of the inexact line searches are performed along the same ray (direction of search), the gradient has to be computed at the beginning of each iteration of the loop in order to determine the sign of the step size. The exact line search algorithm stops when the gradient is perpendicular to the direction of search.

STOPPING CRITERION

The stopping criterion represents a set of conditions under which the local search algorithm is considered to have reached a solution. There is a wide variety of stopping criteria, and the most commonly used is to stop the search when the Euclidean norm of the gradient has reached certain small pre-defined value, which is called the tolerance. Other commonly used stopping criterion evaluates the objective function at each iteration and stops the search when the reduction with respect to the previous iteration is below the tolerance value. Finally, another common criterion, and the one that has been implemented, compares the obtained model parameters at each iteration and stops the search when the amount of variation in the model parameters is below the tolerance.

Notice that none of the criteria described above is failure-free. If it is true that reaching a minimum is a sufficient condition for these criteria to stop the search, the opposite is generally false. For example, the gradient criterion will stop the search at any point the gradient is almost zero, which is not necessarily a minimum; in a similar way, the objective function criterion can mistakenly stop the search if the algorithm is moving along a level set of the error surface; and the parameter variation condition will stop at any iteration in which the resulting step size is very small.

It is possible to improve the assertiveness of the stopping criterion by using a counter. Under this scheme, the algorithm will continue the search until the stopping criterion has been satisfied during the last K consecutive iterations, where K is a pre-defined integer. Another way of improve the assertiveness is, of course, by combining different stopping criteria.

The reasons why the model parameter variation criterion was chosen was that it is the cheapest criterion from a computational point of view and it represents, in our opinion, the most logical criterion. This is since it will stop when no significant improvement is been made to the model. So, it does not matter if the reached model represents a good solution or not, the search algorithm is just stuck at that location.

GRADIENT METHODS

Five Gradient Methods, belonging to three different categories, have been implemented for the inverse modeling of the time harmonic field electric logging problem. Their specific characteristics are described in detail in the present section.

1.- Negative Gradient Method.

This is the most simple of all. In this method, the direction of search is defined as minus the gradient and the size of the jump can be computed by using any of the three line search algorithms discussed before.

2.- Steepest Descent Methods.

In this type of methods, the direction of search is chosen to be the steepest descent direction with respect to certain norm [2]. The steepest descent direction with respect to the norm u is defined as:

$$\bar{d} = \arg\min \left\{ \nabla \bar{E}(\bar{x})^T \, \bar{v} \;\middle|\; \|\bar{v}\|_u = 1 \right\} \tag{10}$$

Notice that what (10) does is to find a vector $\bar{v}$ into the norm u such that the inner product between the gradient and $\bar{v}$ is minimum. In other words, the steepest descent direction with respect to the norm u is the smallest directional derivative according to the norm u.

Depending on the used norm, different algorithms can be defined. Three steepest descent methods have been implemented by using the L1, L2 and L∞ norms. Figure 3 illustrates in a 2-dimensional space the principal differences between the resulting search directions for these three steepest descent algorithms.

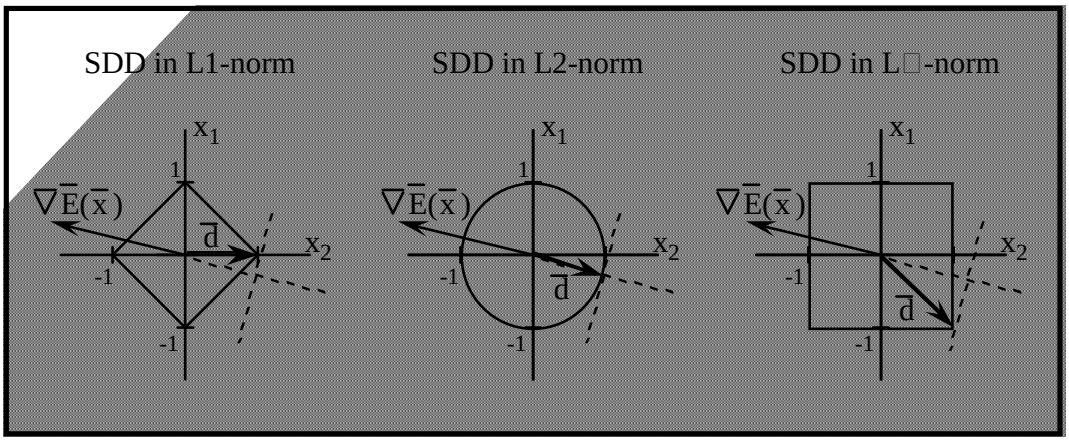

Figure 3: Steepest Descent Direction (SDD) in the L1, L2 and L∞ norms.

As it can be seen from Figure 3, the steepest descent direction in L2 (Euclidean norm) is simply the normalized negative gradient. The one in L1 is related to the largest component of the gradient; so, updating is performed to one variable at a time. And the steepest descent direction in L∞ is given by the corner of the norm that is closest to the direction of the negative gradient. Notice that the steepest descent directions in L1 and L∞ are not always unique.

2.a.- Steepest Descent in L1-norm.

For any index i such that $\left\|\nabla\bar{E}(\bar{x})\right\|_\infty = \left|\left(\nabla\bar{E}(\bar{x})\right)_i\right|$, the steepest descent direction in L1 is given by:

$$\bar{d} = \{ d_1 \quad d_2 \quad \dots \quad d_M \}$$

$$\text{with} \quad d_i = -\text{sign}\left(\nabla\bar{E}(\bar{x})\right)_i \quad \text{and} \quad d_k = 0 \quad \text{for} \quad k \neq i \tag{11}$$

2.b.- Steepest Descent in L2-norm.

The steepest descent direction in L2 is given by:

$$\bar{d} = -\frac{\nabla\bar{E}(\bar{x})}{\left\|\nabla\bar{E}(\bar{x})\right\|_2} \tag{12}$$

2.c.- Steepest Descent in L∞-norm.

The steepest descent direction in L∞ is given by:

$$\bar{d} = \{ d_1 \quad d_2 \quad \dots \quad d_M \}$$

$$\text{with} \quad d_i = -\text{sign}\left(\nabla\bar{E}(\bar{x})\right)_i \quad \text{for} \quad i = 1,2,...M \tag{13}$$

In all the steepest descent methods described, the step size s can be computed by using either the inexact line search algorithm or the exact line search one. Notice that a fixed step size cannot be used with this type of methods since they compute normalized directions of search and the situation illustrated in Figure 1 would always occur.

3.- Quasi-Newton Methods.

In this type of methods, the direction of search is computed as the product between a positive definite matrix **A** and the negative gradient. This can be interpreted as a steepest descent direction with respect to the norm defined by the matrix **A**. The term Quasi-Newton comes from the fact that the matrix **A** intends to be an approximation to the inverse of the Hessian. Two of this methods have been implemented. They are described next.

3.a.- BFGS Method (Broyden-Fletcher-Golfarb-Shanno Method) [2].

In this method, the matrix **A** is initially defined as the inverse of any approximation of the Hessian; or, when this is not possible, it is just defined as the identity matrix. The direction of search is given by the product between **A** and the negative gradient, and the step size is computed by using an inexact line search algorithm. Then, the matrix **A** is updated according to the following formula:

$$\mathbf{A} := \mathbf{A} + \left(1 + \frac{\Delta\bar{g}^T \mathbf{A}\, \Delta\bar{g}}{\Delta\bar{x}^T \Delta\bar{g}}\right) \frac{\Delta\bar{x}\, \Delta\bar{x}^T}{\Delta\bar{x}^T \Delta\bar{g}} - \frac{\mathbf{A}\, \Delta\bar{g}\, \Delta\bar{x}^T + \Delta\bar{x}\, \Delta\bar{g}^T \mathbf{A}}{\Delta\bar{x}^T \Delta\bar{g}} \tag{14.a}$$

where $\Delta\bar{g} = \nabla\bar{E}(\bar{x} + s\,\bar{d}) - \nabla\bar{E}(\bar{x})$ and $\Delta\bar{x} = \bar{x} + s\,\bar{d} - \bar{x} = s\,\bar{d}$ (14.b)

3.b.- Conjugate Gradients [2].

In this method, the direction of search is computed as a linear combination between the negative gradient and the previous direction of search. This is done according to the following formula:

$$\bar{d} = -\nabla\bar{E}(\bar{x}) + \eta\, \bar{d}_{prev} \tag{15}$$

where $\bar{d}_{prev}$ is the direction of search of the previous iteration and η is the linear combination factor, which is zero for the first iteration and is updated according to (16) for the successive ones. Once the search direction has been obtained, the step size must be computed by using an exact line search algorithm. Then, the linear combination factor can be updated according to the following formula:

$$\eta = \frac{\Delta\bar{g}^T\, \nabla\bar{E}(\bar{x} + s\,\bar{d})}{\Delta\bar{g}^T\, \bar{d}} \tag{16.a}$$

where $\Delta\bar{g} = \nabla\bar{E}(\bar{x} + s\,\bar{d}) - \nabla\bar{E}(\bar{x})$ (16.b)

It can be easily verified that the conjugate gradients method is a particular case of the BFGS method. By considering the BFGS method with the identity matrix instead of **A** in (14.a) and using an exact line search, after some algebraic manipulations, equations (15) and (16) follows.

BARRIER FUNCTIONS

Barrier functions are used to prevent search algorithms from getting out of the domain of the problem (the feasible set or some other specified region of operation). They are monotonous smooth functions which domain is defined over the same domain of the problem and they grow without bound as the model parameters approach the domain's boundaries. Although they do modify the objective function, when used carefully, they are expected not to produce significative changes in the location of the minima while certainly improving the performance of the search algorithm.

Two basic types of barrier functions will be described, the inverse barrier function and the logarithmic barrier function [2]. Suppose that the region of operation (domain) of the problem is defined by:

$$\mathrm{dom}\{\mathrm{THFEL}\} = \left\{ \bar{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_M \end{bmatrix}^T \;\middle|\; x_{min} < x_i < x_{max},\ i = 1,2,...M \right\} \tag{17}$$

Then, an inverse barrier function would be given by:

$$\phi(\bar{x}) = -\sum_{i=1}^{M} \left(\frac{1}{x_i - x_{max}} - \frac{1}{x_i - x_{min}} \right) \tag{18}$$

and a logarithmic barrier function by:

$$\phi(\bar{x}) = -\sum_{i=1}^{M} \ln\left[(x_{max} - x_i)\,(x_i - x_{min}) \right] \tag{19}$$

In order to include the barrier function into the optimization process, a new objective function has to be defined as follows:

$$E_B(\bar{x}) = E(\bar{x}) + \frac{1}{t}\phi(\bar{x}) \qquad \text{with} \quad t \geq 1.0 \tag{20}$$

where $E(\bar{x})$ is either the error function defined in (1) or the normalized error function defined in (5) and t is a pre-defined weighting factor that regulates the effect of the barrier function. Notice that when t ->∞, the minimization of $E_B(\bar{x})$ becomes equivalent to the minimization of $E(\bar{x})$.

Finally, the new gradient is given by:

$$\nabla \bar{E}_B(\bar{x}) = \left[\left.\frac{\partial E_B}{\partial x_1}\right|_{\bar{x}} \quad \left.\frac{\partial E_B}{\partial x_2}\right|_{\bar{x}} \quad \cdots \quad \left.\frac{\partial E_B}{\partial x_M}\right|_{\bar{x}} \right]^T \tag{21.a}$$

$$\left.\frac{\partial E_B}{\partial x_j}\right|_{\bar{x}} = \left.\frac{\partial E}{\partial x_j}\right|_{\bar{x}} + \frac{1}{t}\left(\frac{(-1)^n}{(x_j - x_{max})^n} - \frac{1}{(x_j - x_{min})^n}\right) \qquad \text{for} \quad j = 1,2,...M \tag{21.b}$$

where n=2 for the inverse barrier function and n=1 for the logarithmic barrier function.

CONCLUSIONS

The Gradient Methods provide a suitable family of local search algorithms for the inverse modeling of the time harmonic field electric logging problem.

Although these kind of methods are slower than other kind of local search algorithms, as for example the Newton Method, the fact that they only require first order derivatives and the availability of a procedure for computing those derivatives [3], made them a very appropriate choice as local search algorithms.

REFERENCES

[1] Update Report #12: The Inverse Problem.

[2] Boyd, S.; Vandenberghe, L. (1996), Convex Optimization., (Course Reader for EE364) Stanford University.

[3] Update Report #8: Recursive computation of the derivatives.