Quasi-Newton methods for large-scale distributed parameter estimation

E. Haber

Dept of Mathematics and Computer Science Emory University, Atlanta, GA

April 11, 2003

Abstract

We develop Quasi-Newton methods for distributed parameter estimation problems, where the forward problem is governed by a set of partial differential equations. A Tikhonov style regularization approach yields an optimization problem with a special structure, where the gradients are calculated using the adjoint method. In many cases standard Quasi-Newton methods (such as L-BFGS) are not very effective and tend to converge slowly.

Taking advantage of the special structure of the problem and the quantities that are calculated in typical gradient descent methods we develop a class of highly effective methods for the solution of the problem.

We demonstrate the merits and effectiveness of our algorithm on two realistic model problems.

1 Introduction

In this paper we develop Quasi-Newton (QN) methods for the solution of nonlinear inverse problems that arise from elliptic Partial Differential Equations (PDEs). We consider the optimization problem that stems from a Tikhonov-style regularized (discretized) inverse problem of the form

$$\min \quad \phi = \frac{1}{2}\|Q\,u(m) - b^{\text{obs}}\|^2 + \beta R(m, m^{\text{ref}}) \;=\; \phi_d(m) + \beta R(m, m^{\text{ref}}), \quad (1)$$

where b^{obs} is some observed data and m^{ref} is a reference model. The functional R is a regularization function which is a discretization of a differential operator and can be typically expressed as a sparse matrix. The vector $u(m)$ is the field which depends on the model m, and Q is an interpolation matrix. Solving for $u(m)$ is the forward problem, and the goal of the inverse problem is to recover m by solving the optimization problem (1). In this paper we consider the case where the forward problem is a set of discretized PDEs of the form

$$\mathcal{A}(m)u = F \qquad \leftrightarrow \qquad u = \mathcal{A}(m)^{-1}F \ , \tag{2}$$

where $\mathcal{A}(m)$ is a discretized differential operator that depends on m, and the right-hand side is a matrix F. That is, we have a multiple right-hand side problem here.

Equation (2) forms a constraint under which ϕ (see equation (1)) needs to be minimized. A natural question that arises at this point is how easy would it be to solve problem (1)-(2) in its given, unconstrained form. In this paper we are particulary interested in problems where the number of right-hand sides is very large and one cannot store u. It is however possible to evaluate and store $Qu = Q\mathcal{A}(m)^{-1}F$. Therefore, an all-at-once (or full-space) type method [8, 10] which does not eliminate the PDE constraint is difficult to apply, and resorting to an unconstrained optimization formulation may be advantageous. Therefore, we will turn to solve (1)-(2) by means of eliminating (2) and incorporating the resulting expression for u into ϕ in (1). For notational convenience, below we will continue to express some of the quantities with dependence in u, but with the understanding that in the actual computation u will be replaced as explained above.

The gradient of (1) is given by

$$g \ = \ J(m, u(m))^T(Q\,u(m) - b^{\text{obs}}\ ) + \beta R_m(m, m^{\text{ref}}) = g_d + \beta g_m, \tag{3}$$

where the (usually dense) matrix $J(m, u(m))$ is the so-called sensitivity matrix of the form

$$J = -Q\mathcal{A}(m)^{-1}\frac{\partial[\mathcal{A}(m)u]}{\partial m},$$

(see [8] for derivation) and all the matrices but $\mathcal{A}^{-1}$ are sparse. It is important to note that although J is dense it need not be computed explicitly and matrix-vector products $J^T v$ can be computed by solving the adjoint problem

$$J^T v = -\left(\frac{\partial[\mathcal{A}(m)u]}{\partial m}\right)^T \left(\mathcal{A}(m)^{-T}(Q^T v)\right).$$

Such a calculation may be computationally intense but is not prohibitively expensive. The Hessian of (1) is given by

$$H \;=\; J(m,u(m))^T J(m,u(m)) + A_s(m) + \beta R_{mm}, \tag{4}$$

where $A_s(m)$ is a matrix of second derivatives $(Qu(m))_{mm}$. It is possible to use the Conjugate Gradient (CG) method for the solution of a linear system of the form $Hv = y$. It should be noted, though, that a single CG iteration will involve two matrix-vector products involving J and J^T, (the forward problem and the adjoint problem respectively) and thus the solution of the linear iteration within the nonlinear solver is computationally expensive. It is thus the computational cost that motivates the use of Quasi-Newton methods for the solution of the problem.

Problem (1) has special characteristics that are different from general unconstrained optimization problems, and taking advantage of these characteristics may result in a highly effective QN method. The following considerations form the basis for the derivation of our method:

- The gradient and the Hessian are comprised of two parts: A part that relates to the regularization and a part that relates to the data misfit. Of the two, the former is relatively easy to compute (for the class of problems we are considering here) whereas the latter is difficult to evaluate. Thus an effective QN method should be able to produce a good approximation to the data functional in particular.

- Unlike many other nonlinear least-squares problems, the gradient g and the gradient with respect to the data, g_d, are calculated without explicitly computing J. This is a very important difference between our problem and other least-squares problems. Indeed, in most other cases discussed in the literature (see [4, 14] and references therein) the matrix J is available when the gradient is computed.

- The matrix J can be thought of as a discretization of a compact integral operator with eigenvalues which cluster at zero. Therefore it can be approximated by a low rank matrix. This is a key point in our derivation and is crucial in assessing the savings that are gained throughout our implementation.

Based on the above considerations, we derive an algorithm which does not rely on an approximation of the full Hessian of the objective function.

As this point it is worthwhile noting that the question whether the full Hessian should be approximated is by no means a new topic for investigation. An extensive discussion of this can be found in the literature. For example, consider the common case where R is a quadratic regularization of the form $(m - m^{\text{ref}})^T W^T W (m - m^{\text{ref}})$ with some smoothing matrix W. The problem (1) can be thought of as a least-square problem of the form

$$\begin{pmatrix} Qu(m) \\ \sqrt{\beta} W m \end{pmatrix} \approx \begin{pmatrix} b^{\text{obs}} \\ \sqrt{\beta} W m^{\text{ref}} \end{pmatrix}.$$

Dennis & Schnabel [4, P. 229] advocate that for least-squares problems approximating the full Hessian does not lead to very good results. Gilles, Vogel and Bardsley show in [6] that for certain problems choosing $\beta W^T W$ as the initial Hessian dramatically improves the results for quadratic regularization for large enough β. Nevertheless, if the regularization parameter is small or if the regularization is nonlinear (such as total-variation), then this approach may be less effective. In the non-quadratic regularization case such an approach does not utilize the explicit knowledge of the part of the Hessian that is associated with the regularization part. This part of the Hessian is usually a discretization of a non-compact operator, and standard QN methods such as L-BFGS [14] may have diffuculties in approximating it by a low rank pertubation to the Hessian. These observations provide further motivation for the algorithm discussed in the present work.

The rest of the paper lays out the construction of the new algorithm by way of exploiting the above properties in order to generate an efficient QN method. In Section 2 we discuss our QN strategy and the QN iteration. In Section 3 we show how to effectivaly store and solve the linear subproblem in each nonlinear iteration, and we summarize our algorithm. Finally, in Section 4 we present numerical experiments that validate the viability of our approach, and draw some conclusions.

2 The Quasi-Newton iteration

As explained above we do not intend to approximate the full Hessian, only the part which relates to the data functional, ϕ_d. Assume that we have done k iterations and we have the following quantities:

1. The models $[m_1, ..., m_k]$,

2. The predicted data residuals $[r_1(m_1), ..., r_k(m_k)] = [Qu(m_1)b^{\text{obs}}, ..., Qu(m_k) - b^{\text{obs}}]$

3. The data gradients $[g_1^d(m_1), ...g_k^d(m_k)] = [J_1^T r_1(m_1), ..., J_k^T r_k(m_k)]$

Note again that we do not calculate the sensitivities J_k and that $g_k^d = J_k^T r_k$ is calculated implicitly. Using this information, we would like to approximate the sensitivity matrix J. It is important that the true J_k does not have to be full rank. In fact, for most inverse problems J_k is either of low rank or contains very small singular values [9]. This is because J_k is usually a discretization of a compact operator with singular values that cluster at zero [5] and this is a key observation for the approximation of J_k because a low rank update generates a reasonable approximation. Similar arguments where presented in [12] to prove convergence of a symmetric rank-one update.

To approximate J_k, we note that the fields $u(m_k)$ and the sensitivities J_k obey

$$Qu(m_{k-1}) \approx Qu(m_k) + J_k(m_{k-1} - m_k)$$

Defining as usual $s = m_k - m_{k-1}$, $y = Q(u(m_k) - u(m_{k-1})) = r(m_k) - r(m_{k-1})$ one obtains the usual secant equation

$$y = \widehat{J}_k s \tag{5}$$

where $\widehat{J}_k$ is the secant approximation to the true matrix J_k. We regard Equation (5) as the "*first secant equation*". We can use this equation to evaluate $\widehat{J}_k$ using the usual secant-type arguments of least-change updates [4] which leads to Broyden's update[1]

$$\widehat{J}_k = \widehat{J}_{k-1} + [s^T s]^{-1}(y - \widehat{J}_{k-1})s^T \tag{6}$$

However, we have more information and we may be able to obtain a better approximation. Our extra information is given by the data gradient vectors. Using the gradients we have

$$g_{k-1}^d \approx g_k^d - (J_k^T J_k + A_s)s$$

and defining $q = g_k^d - g_{k-1}^d$ we obtain

$$q \approx J_k^T J_k s + A_s s = J_k^T y + A_s s \tag{7}$$

[1]Note that although in general $\hat{J}$ is nonsquare the results from [4] p. 169 is unchanged (see also [7])

Most of our problems are low residual problems, indeed, it is well known that for elliptic forward problems even a constant function m gives a reasonable fit to the data. For such problems, the term A_s may be very small and can be dropped obtaining our second secant equation

$$q = \widehat{J}_k^T y = \widehat{J}_k^T \widehat{J}_k s \tag{8}$$

which we refer to as the "*second secant equation*". Note that for (8) to hold we must have the curvature condition

$$s^T q \geq 0 \tag{9}$$

which is the usual secant condition for a symmetric positive semi-definite matrix. We refer to this condition as the "*first secant condition*". Furthermore, multiplying (8) by s^T we obtain

$$s^T q = s^T \widehat{J}_k^T y = y^T y \tag{10}$$

This yields a second condition on the vectors s, y and q which we refer to as the "*second secant condition*". Obviously, this condition does not hold exactly but if it holds approximately, then we can look for a secant update, $\widehat{J}_k$, such that both secant equations (5) and (2) approximately hold. If the second secant condition (10) holds exactly then the two secant equations are not completely independent. We have $n_1 + n_2$ equations (assuming J is $n_1 \times n_2$) but one constraint (10). Therefore we have only $n_1 + n_2 - 1$ independent equations. We will return to this point later.

We now look at two different options for the update by a rank-one and two matrices.

2.1 A rank one update

For a rank-one update of $\widehat{J}_k$ we assume that

$$\widehat{J}_k = \widehat{J}_{k-1} + \delta J = \widehat{J}_{k-1} + uv^T \tag{11}$$

Substituting in (5) and (8) we obtain

$$y - \widehat{J}_{k-1} s = uv^T s \tag{12a}$$
$$q - \widehat{J}_{k-1}^T y = vu^T y \tag{12b}$$

Since $v^T s$ is just a scalar, u can be written as

$$u = \alpha(y - \widehat{J}_{k-1}s) \tag{13}$$

Using (12b) we obtain

$$v = \alpha^{-1}[(y - \widehat{J}_{k-1}s)^T y]^{-1}(q - \widehat{J}_{k-1}^T y) \tag{14}$$

Using u and v we have found the secant update

$$\widehat{J}_k = \widehat{J}_{k-1} + [(y - \widehat{J}_{k-1}s)^T y]^{-1}(y - \widehat{J}_{k-1}s)(q - \widehat{J}_{k-1}^T y)^T \tag{15}$$

Note that unless the second secant condition (10) holds exactly, the update does not exactly solve both secant equations (5) and (8). Since we regard the first equation (5) as more accurate, we modify the update using $q^T s \approx y^T y$ to

$$\widehat{J}_k = \widehat{J}_{k-1} + [(q - \widehat{J}_{k-1}^T y)^T s]^{-1}(y - \widehat{J}_{k-1}s)(q - \widehat{J}_{k-1}^T y)^T \tag{16}$$

This equation solves the first secant equation (5) exactly but solves the second secant equation (8) only if the second secant condition holds. To make the update stable we use it only if

$$(q - \widehat{J}_{k-1}^T y)^T s > \text{tol}\, \|s\|\, \|q - \widehat{J}_{k-1}^T y\| \tag{17}$$

Equation (6) and (16) are the secant updates we use. If Equation (17) does not hold we use (6) otherwise we use the update (16).

We comment that if the second secant equation (8) does not hold one may still want to use the second secant equation in order to evaluate the matrix of second derivatives A_s and use the first secant equation (5) in order to evaluate $\widehat{J}_k$. We have tried this but this approach did not do as well as the simple rank one secant update to $\widehat{J}_k$.

2.2 A rank-two update

The disadvantage of a rank one update is that it may not be used due to the condition (17) but furthermore, the Jacobian information builds slowly. A different approach which leads to a more stable update is to use a rank-two update. Assuming a least-change update we want to find $\delta\widehat{J}$ such that

$$\begin{aligned} \min \quad & \frac{1}{2}\|\delta\widehat{J}\|_F^2 \\ \text{s.t} \quad & \delta\widehat{J}s + (\widehat{J}s - y) = 0 \\ & \delta\widehat{J}^T y + (\widehat{J}^T y - q) = 0 \end{aligned} \tag{18}$$

Before we discuss the solution of this optimization problem we recall that the two secant equations are not completely independent and thus we have one degree of freedom.

To solve the problem we form the Lagrangian

$$\mathcal{L} = \frac{1}{2}\|\delta\widehat{J}\|_F^2 + \lambda^T(\delta\widehat{J}s + (\widehat{J}s - y)) + \mu^T(\delta\widehat{J}^T y + (\widehat{J}^T y - q)) \tag{19}$$

where λ and μ are Lagrange multipliers.

Differentiating the Lagrangian we obtain

$$\begin{aligned} \delta\widehat{J} + \lambda s^T + y\mu^T &= 0 && (20a)\\ \delta\widehat{J}s &= y - \widehat{J}s && (20b)\\ \delta\widehat{J}^T y &= q - \widehat{J}^T y && (20c) \end{aligned}$$

To solve the equations we multiply (20a) by s from the right and then y^T from the left and use (20b) and (20c) to obtain

$$\begin{aligned} y - \widehat{J}s + (s^T s)\lambda + (\mu^T s)y &= 0\\ q - \widehat{J}^T y + (\lambda^T y)s + (y^T y)\mu &= 0 \end{aligned}$$

which implies that

$$\begin{aligned} \lambda &= (s^T s)^{-1}(\widehat{J}s - y - (\mu^T s)y)\\ \mu &= (y^T y)^{-1}(\widehat{J}^T y - q - (\lambda^T y)s) \end{aligned}$$

Setting new scalar unknowns $\alpha = \mu^T s$ and $\beta = \lambda^T y$ we can write

$$\delta\widehat{J} = -(s^T s)^{-1}(\widehat{J}s - y - \alpha y)s^T \; - \; (y^T y)^{-1}y(\widehat{J}^T y - q - \beta s)^T \tag{21}$$

This update has two scalar unknowns α and β. We now substitute (21) into the first secant equation (5) and obtain

$$-\left((s^T s)^{-1}(\widehat{J}s - y - \alpha y)s^T \; + \; (y^T y)^{-1}y(\widehat{J}^T y - q - \beta s)^T\right)s \;\; = \;\; y - \widehat{J}s$$

Collecting terms we obtain

$$\left((y^T y)\alpha - s^T\widehat{J}^T y + q^T s + \beta(s^T s)\right)y = 0$$

This is a scalar equation for α and β. In principle, we could substitute $\delta\widehat{J}$ into the second secant equation (8). However, if $q^T s = y^T y$ then we obtain the same scalar equation. This should not come as a surprise because the joint secant equations are rank-deficient. We thus have the freedom to choose α or β as we please. Choosing $\alpha = 0$ the first term in the secant update is equivalent to a correction to Broyden's update (5) (making our update a rank one update to the Broyden update), obtaining

$$\beta = (s^T s)^{-1}(s^T \widehat{J}^T y - q^T s)$$

and the rank two secant update

$$\delta\widehat{J} = (s^T s)^{-1}(y - \widehat{J}s)s^T \;+\; (y^T y)^{-1} y(q - \widehat{J}^T y + \beta s)^T \tag{22}$$

Note that the rank-two update is made of two terms. The first corresponds to the usual Broyden update while the second term in our update can be thought of as a correction to the rank-one update given the second secant equation (8). If the condition $q^T s = y^T y$ holds then this secant update solves the two secant equations (5) and (8) exactly. However in most cases, where this condition does not hold, the second secant equation (8) holds only approximately.

3 Updating and solving

If our method is to be used for large scale problems, we cannot store $\widehat{J}_k$ and only a few vectors can be stored. Furthermore, we need to be able to quickly solve a system of equations of the form

$$(\widehat{J}_k^T \widehat{J}_k + \beta R'')\delta m = -g(m) \tag{23}$$

In this Section we discuss these issues.

3.1 Updating

To make the storage efficient we express $\widehat{J}_k$ as a linear combinations of $\rho_i, u_i, v_i, i = 1, \ldots, k$, where for the rank-one updates

$$\begin{aligned}
\widehat{J}_k &= \sum_{i=1}^{k} \rho_i u_i v_i^T \\
\rho_i &= [(q - \widehat{J}_{k-1}^T y)^T s]^{-1} \quad \text{or} \quad (s_i^T s_i)^{-1} \\
u_i &= y - \widehat{J}_i s_i \\
v_i &= q - \widehat{J}_i^T y_i \quad \text{or} \quad s_i
\end{aligned}$$

and similar for the rank two updates.

At each iteration we use the previous u's and v's to generate the next pair. For the rank one update, this can be simply done as follows. Assume we are at iteration k where previous $k-1$ vectors have been calculated. It is easy to calculate the new vectors u_k and v_k by

$$\begin{aligned}
u_k &= y_k - \sum_{i=1}^{k-1} \rho_i (v_i^T s_k) u_i \\
v_k &= q_k - \sum_{i=1}^{k-1} \rho_i (u_i^T y_k) v_i \quad \text{or} \quad s_k \\
\rho_k &= (u_k^T y_k)^{-1} \quad \text{or} \quad (s_k^T s^k)^{-1}
\end{aligned}$$

If $k > k_{\max}$ we drop the first pair of vectors in the sum truncating the decomposition. This is similar to the L-BFGS method. As stated before, J_k is usually of low rank anyway and therefore only a small number of vector may be needed to approximate it; we will show that in our numerical examples. For the rank two update we need to generate four vectors and we do this in a similar way utilizing the same quantities needed for the rank one update.

3.2 Solving the linear subproblem

We now shortly discuss the solution of the QN-system.

$$(J^T J + \beta R'')\delta m = g \tag{24}$$

(where we drop the iteration number k for ease of notation). We assume that the size of the problem is large and therefore one cannot calculate and

store the QR factorization of J and we turn to iterative methods. Also, we assume that R'' is invertible; this is true for virtually all of our applications. If R'' has a trivial null space (as is the case when $R'' = -\nabla_h^2$ with Neumann boundary conditions) then we further regularize the linear iteration (24) and solve instead

$$(J^T J + \beta(R'' + h^2 I))\delta m = g \tag{25}$$

where h is the grid size. Since R'' has $\mathcal{O}(h^2)$ discretization error, such regularization maintains the overall numerical accuracy (see [16]).[2] We denote the modified regularization matrix as R''_h.

Before we solve the system, recall that we can write the sensitivity matrix J as

$$J = \sum_{j=1}^{k} \rho_j u_j v_j^T$$

This implies that a matrix vector product of the form Jv or $J^T u$ is only $\mathcal{O}(n)$ operations and that the rank of J is at most k. Since k is typically small (say 5-25), this calculation is done very fast and thus although J is a dense matrix its application to a vector is equivalent to a sparse matrix-vector product. In our application R'' is a discretization of a differential operator and therefore, the solution of a system of the form $R''_h u = q$ can be typically done in $\mathcal{O}(n)$ operations using a multigrid method. Here we have implemented a nodal based multigrid method [16] with bilinear interpolation, full weight restriction and used symmetric Gauss-Seidel as a smoother for the solution of the problem.

In order to solve the system we now note that the preconditioned system $(R''_h)^{-1} J^T J + \beta I$ has at most $k+1$ different eigenvalues. This implies that the preconditioned conjugate gradient (PCG) method applied to the system will converge in $k+1$ steps at most even if the condition number of the problem is large. One can terminate the iteration early, which corresponds to an Inexact-Quasi-Newton method and save a few iterations. In our numerical experiments we have observed that the PCG method does not converge to a very small tolerance within k iterations but it usually yields a small residual (say 10^{-5}), which is a sufficient dicrease for the linear iteration within a nonlinear solver.

[2] It can also be viewed as a Gauss-Newton step with a small Levenberg parameter h^2.

3.3 Summary of the algorithm

To complete the optimization algorithm we need to further discuss a globalization method which guarantees convergence to a local minimum. Here we use a line search strategy and since we use a QN method we employ the standard strong Wolfe conditions for the line search [14]. This not only ensures sufficient decrease but also enforces the first secant condition ($s^T q > 0$) explicitly. We summarize our algorithm as follows

Algorithm I - QN method for nonlinear inverse problems

- Initialize m_0, and calculate

$$\begin{aligned} b_0 &= Q\mathcal{A}(m_0)^{-1}F \\ \phi &= \frac{1}{2}\|b_0 - b^{\text{obs}}\|^2 + \beta R(m_0) \\ g_0 &= g_0^d + \beta g_0^m \end{aligned}$$

- Initialize storage of L vectors for U, V and ρ (the vectors for the approximation of $\widehat{J}$)
- for $k = 1, 2, ..$ (QN loop)
  1. Use U, V and ρ to solve for the QN step
  $$(\widehat{J}_k^T \widehat{J}_k + \beta R_h'')p = -g$$
  using PCG with R_h'' as a preconditioner.
  2. Set $m_k = m_{k-1} + \alpha p$ where α is chosen by a line search
  3. calculate
  $$\begin{aligned} b_k &= Q\mathcal{A}(m_k)^{-1}F \\ \phi_k &= \frac{1}{2}\|b_k - b^{\text{obs}}\|^2 + \beta R(m_k) \\ g_k &= g_k^d + \beta g_k^m \end{aligned}$$
  4. Set $s = m_k - m_{k-1}$, $q = g_k^d - g_{k-1}^d$, $y = b_k - b_{k-1}$
  5. Update U, V and ρ by using Equation (16) for the rank-one update, or by using Equation (22) for the rank-two update.

6. If $k > L$ drop the u_1, v_1 and ρ_1,
   set $u_i \leftarrow u_{i+1} \quad v_i \leftarrow v_{i+1} \quad \rho_i \leftarrow \rho_{i+1} \quad i = 2...k-1$.
7. Check for convergence

In the above the approximation to the sensitivity $\widehat{J}$ is used in order to replace the linear solver within the Gauss-Newton step. This will usually deteriorate the convergence rate of the outer iteration and therefore it is possible to combine the QN approximation to the Hessian with the Gauss-Newton iteration by using the QN matrix as a preconditioner in a Gauss-Newton-CG process. In this case we still perform one forward and one adjoint solves for each CG iteration but we use the QN system as a preconditioner. The hope in this case is that the number of CG iterations within each Gauss-Newton iteration will be dramatically reduced. To use this algorithm we follow the same algorithm as above replacing the solution of the QN system with the preconditioned Gauss-Newton system

$$(\widehat{J}_k^T \widehat{J}_k + \beta R'')^{-1}(J_k^T J_k + \beta R'')p = -(\widehat{J}_k^T \widehat{J}_k + \beta R'')^{-1} g$$

Therefore, at every CG iteration we not only solve the forward and adjoint problems, but also solve a system of the form $(\widehat{J}_k^T \widehat{J}_k + \beta R'')v = w$

4 Numerical experiments

In this section we experiment with two 3D model problems. These problems demonstrate the computational advantage of our approach.

The first problem we experiment with is the 3D DC resistivity or impedance tomography where the PDE is given by

$$\begin{aligned} \nabla \cdot \exp(m)\nabla u &= F; \qquad && u \in \Omega; \\ \nabla u \cdot \mathbf{n} &= 0; && u \in \partial\Omega \end{aligned} \tag{26}$$

This problem is routinely solved in geophysics and medical physics (for a survey see [2]).

In our second problem we consider the approximation to the imaginary part of the magnetic fields in Maxwell's equations at low frequencies which is used in magnetic induction tomography [1, 17]

$$\begin{aligned} \nabla \times \exp(m)\nabla \times \mathbf{H} - \nabla \exp(m)\nabla \cdot \mathbf{H} &= \omega \mathbf{F}; \qquad && \mathbf{H} \in \Omega; \\ \mathbf{H} \times \mathbf{n} &= 0; && \mathbf{H} \in \partial\Omega \end{aligned} \tag{27}$$

In order to be able to compare optimization algorithmss we use four methods to solve the problems

- Preconditioned Inexact Gauss-Newton-CG (IGN) with R'' as a preconditioner with an inner termination tolerance of 10^{-2}.
- Our QN(20) method, that is we use 20 vector pairs to approximate J.
- Preconditioned Inexact Gauss-Newton-CG (PIGN) method with QN(20) preconditioner and an inner termination tolerance of 10^{-2}.
- The L-BFGS method for comparison.

For the applications we consider here, the cost of the algorithms is dominated by the number of forward and adjoint PDEs which are needed for the solution. All the methods require one forward and one adjoint solve in order to evaluate the gradients. The Inexact Gauss-Newton-CG method further requires one forward and one adjoint solve for each CG iteration. The total number of PDE solves for the QN method is $2k_{\text{outer}}$ (where k_{outer} is the number of iterations). The number of PDE solves for both versions of the Inexact Gauss-Newton-CG (with and without QN preconditioning) is $2k_{\text{outer}} + 2k_{\text{inner}}$ where k_{inner} is the total number of CG iterations needed in all the combined iterations. Both methods may require extra forward solves if a line search is invoked.

4.1 Application to Resistivity inversion I

In the first problem we try to recover the true model function which is a 3D version of the peaks function in matlab

$$\begin{aligned} m(x,y,z) &= \frac{1}{4}[\ 3(1-x)^2 \exp(-(x^2) - (y+1)^2 - 3(z+1)^2) - \\ &\quad 10(x/5 - x^3 - y^5 - z^5) \exp(-x^2 - y^2 - 3z^2) - \\ &\quad 1/3 \exp(-(x+1)^2 - y^2 - 3z^2) - 2\ ] \\ &\quad -3 \leq (x,y,z) \leq 3 \end{aligned}$$

We have 8×8 measurements equally spaced on the surface (i.e. $z = 3$). We assume we have only 16 sources which are also equally spaced on the surface. The combination of 16 sources and 64 receivers amounts to 1024 data points, which we pollute with 1% Gaussian errors. For these experiments we use two types of regularizations

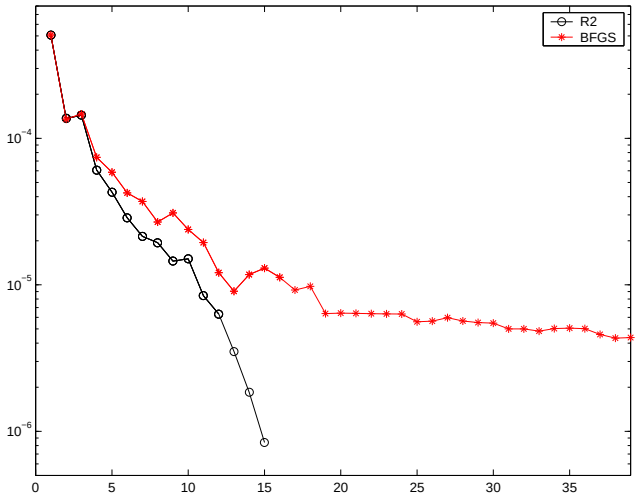

Figure 1: Convergence of our rank two update and the L-BFGS method for the small scale problem with nonquadratic regularization

- A quadratic regularization, $R(m) = \nabla_h^T W^T W \nabla_h m$ where W is a diagonal weighting matrix which penalizes structure close to the sources (see [3] for details) and ∇_h is the discrete gradient.
- Nonquadratic regularization based on the Huber function [11] $R(m) = e^T W h(\nabla_h m; \gamma)$ where e is a vector of ones and γ is the Huber parameter which we fixed as 0.1.

We use a rather coarse grid of 17^3 cells to approximate the PDE which leads to 4913 model unknowns. Solving each PDE (for individual source) can be done in roughly 1sec on a 1100MHz PC and therefore the computation of the forward and adjoint problems require roughly 16sec each. The matrix J is 1024×4913 but we can calculate and store it for further analysis.

In our first experiment we compare our rank two update to the L-BFGS method for Huber regularization. We choose the regularization parameter to be 10^{-7}. While our QN method converged in 16 iteration, the L-BFGS method broke down in the line search process after 39 iterations. This demonstrates the advantage of our method over L-BFGS for problems where the regularization is nonlinear. The convergence curves are plotted in Figure 1.

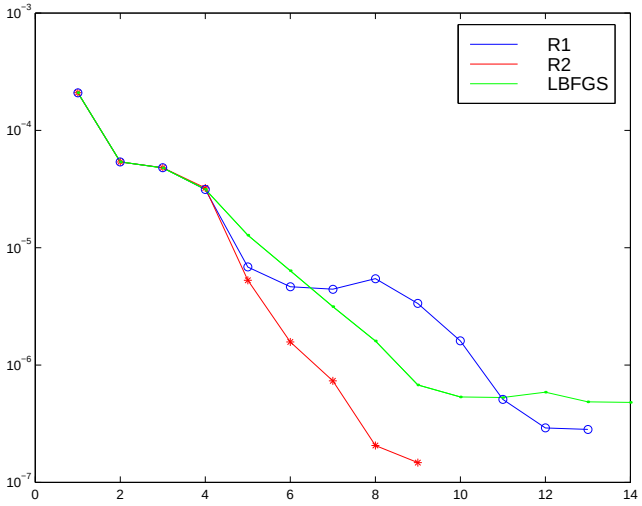

Figure 2: Convergence of our rank one and two updates and the L-BFGS method for the small scale problem with quadratic regularization

In our next experiment we use quadratic regularization with regularization parameter $\beta = 10^{-5}$ and initialize the Hessian in the L-BFGS method to $\beta R''(m)$. The results of the L-BFGS and our method are somewhat similar although we have noticed that our approach require significantly less backtracking steps in line search. This is due to the bad scaling that the β^{-1} factor imposes in the initial guess of the Hessian in the L-BFGS method. Trying to use only $R''(m)$ as an initial guess (hoping that the scaling will be determined by the line search) leads to very slow convergence.

The convergence results for this experiment for our rank-one, two and L-BFGS updates are plotted in Figure 2

The advantage of the L-BFGS update over our approach is that R'' needs to be inverted only once per iteration while in our method the number of R'' inversions is equal to the number of PCG iterations (or to the number of vectors used for the update). Nevertheless, for quadratic regularization, using a multigrid solver our approach was faster for most small regularization parameters and it is only slightly slower when the regularization parameters are large. The reason being that for this problem, the cost of solving $R''u = v$ is roughly 1/16 of the cost of the forward problem. Thus reducing the number of forward and adjoint problems has a profound effect on the cost of the

method.

In our third set of experiments we compare our rank-two update to the inexact Gauss-Newton method with quadratic regularization. To compare we count the number of forward and adjoint PDE solves and summarize the results in Table 1. It is evident that for large regularization parameters the convergence properties of the QN method are almost as good as the Gauss-Newton method. For smaller regularization parameters, although the convergence of the QN method is slower than those of the Gauss-Newton the method, it requires less PDE solves and therefore can be significantly cheaper if the computation of the forward problem is expensive. If the regularization parameter is very small then the QN method does not approximate the GN step well enough and the number of iterations increases dramatically. Even then, the overall cost of the QN method is better but the combination of the QN matrices as a preconditioner to the GN step seems to gain both from fast convergence of the linear system and a fast convergence of the outer iteration.

β	noise	Total		iterations	Total CG		iterations	forward		sol's
		IGN	QN	PIGN	IGN	QN	PIGN	IGN	QN	PIGN
10^{-5}	$6E-2$	11	11	11	31	26	11	89	28	46
10^{-6}	$4E-2$	16	25	16	35	32	17	112	52	68
10^{-7}	$2E-2$	17	36	17	38	42	18	131	78	79
10^{-8}	$8E-3$	21	59	21	52	140	28	158	131	108
10^{-9}	$2E-3$	39	124	39	133	321	88	387	278	259

Table 1: Comparisons between the methods for impedance tomography I

The experiments here also allow us to compare the true Jacobian J to its approximation. Instead of comparing the sensitivity matrix directly, we calculate the SVD of the last iteration and compare its first 3 singular vectors to the singular vectors of our secant update for the experiment with $\beta = 10^{-7}$. The comparison is plotted in Figure 3 and we see that overall, the structure and magnitude of the leading vectors are kept.

4.2 Application to Resistivity inversion II

We now consider a full-scale problem. The same model and regularization functional as in 4.1 is discretized on a $65^3 = 274625$ cells. In this experiment we use the configuration used for environmental studies [15] where sources are put in boreholes and receivers are put on the surface. The experimental configuration is plotted in Figure 4. The number of sources equally spaced

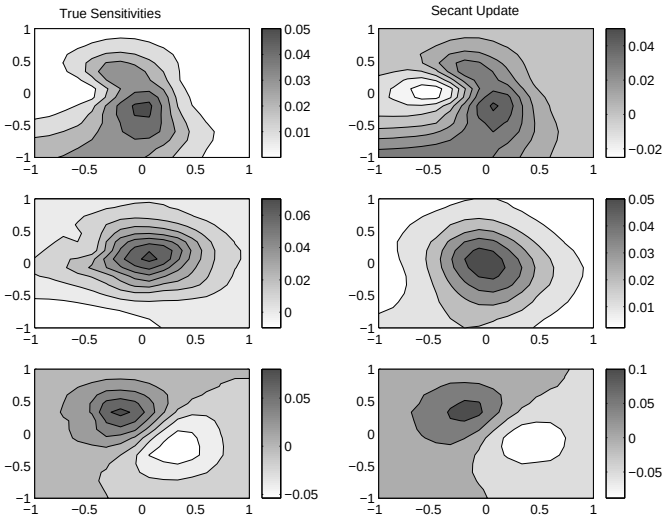

Figure 3: First three singular vectors of the true sensitivity at the solution and its QN approximation

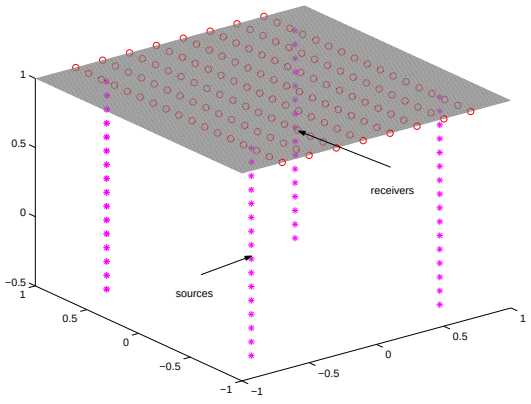

Figure 4: A sketch of the experimental configuration for the large scale 3D problem

in each borehole is $16 \times 4 = 64$ and the number of receivers per source is $32 = 1024$ which amounts to $1024 \times 64 = 65536$ data points and we add 1% noise to the data. This is a large scale DC resistivity survey. In this case each discretized PDE has more than quarter of a million unknowns and although we use multigrid (with operator induced grid transfer [13]) to solve each linear system, our implementation of multigrid takes roughly 2.5min per source and thus one forward problem takes roughly one hour. In this case it is obvious that each forward problem is very expensive and drives the cost of the algorithm. In order to solve this problem we use the rank-two update. In Table 2 we compare the results obtained from the different methods for two noise levels

noise level	Total		iterations	Total	CG	iterations	forward		sol's
	IGN	QN	PIGN	IGN	QN	PIGN	IGN	QN	PIGN
$2E-2$	26	41	26	63	92	32	188	92	124
$1E-2$	37	113	37	123	105	61	421	236	206

Table 2: Comparisons between the methods for large scale impedance tomography

This experiment demonstrate the strength of our QN method. The number of forward problems was dramatically reduced and as a result the wall-clock time for the QN method was less than one half of the Inexact Gauss-Newton method. The preconditioned Inexact Gauss-Newton method was slower than the QN method but still faster than the Inexact Gauss-Newton method. Finally, the results of the inversion are plotted in Figure 2.

4.3 Application to magnetic tomography

In this experiment we use our approach to solve the magnetic induction tomography. This is a borehole experiment where 31 sources are put in one borehole and 31 receivers in the second. The imaginary part of the magnetic field at each receiver due to each transmitter is recorded thus we have 961 data points. The data for this experiment are plotted in Figure 3. This is a field data set[3] and thus we do not have a "true" model.

We now apply the same algorithm for the inversion of the data assuming 5% noise level. We use 20 vectors for the rank two secant update. The results of this experiment are summarized in Table 3. Once again we see

[3]Data supplied by Chevron and Schlumberger-EMI

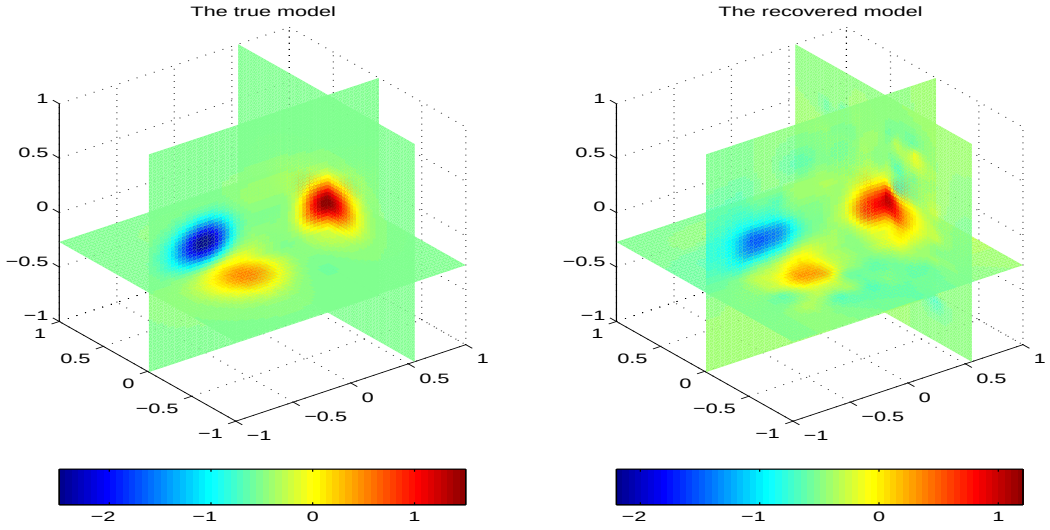

Figure 5: The recovered 3D model for the large scale DC inversion

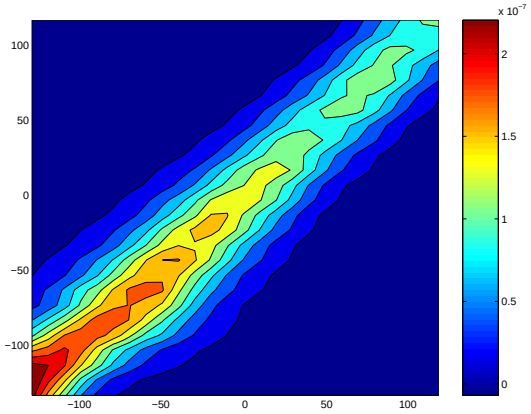

Figure 6: The magnetic data as a function of source-receiver. The x direction is the location of the source and the y is the location of the receiver.

Total		iterations	Total	CG	iterations	forward		sol's
IGN	QN	PIGN	IGN	QN	PIGN	IGN	QN	PIGN
27	61	27	132	134	57	323	129	172

Table 3: Comparisons between the methods for magnetic induction tomography

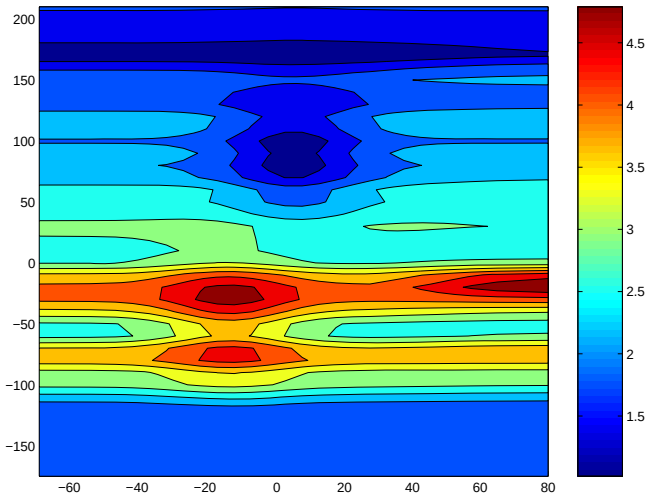

Figure 7: Results of magnetic induction tomography. A slice from the 3D volume

that our QN method is very effective and speeds the nonlinear iteration by either preconditioning the Gauss-Newton step or by replacing it. The result of the inversion is plotted in Figure 7.

References

[1] D. Alumbaugh. Theoretical and practical considerations for crosswell electromagnetic tomography assuming cylindrical geometry. *Geophysics*, 60:846–870, 1995.

[2] L. Borcea. Electrical impedance tomography. *Inverse Problems*, 18:99–136, 2002. n6.

[3] J. Chan, E. Haber, and D. Oldenburg. Three-dimensional numerical modelling and inversion of magnetometric resistivity data. *Geophys. J. Int.*, 149:679–697, 2002.

[4] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.

[5] H.W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer, 1996.

[6] Luc Gilles, C.R. Vogel, and J.M. Bardsley. Computational methods for a large-scale inverse problem arising in atmospheric optics. *Inverse Problems*, 18:237–252, 2002.

[7] E. Haber. *Numerical Strategies for the Solution of Inverse Problems*. PhD thesis, University of British Columbia, 1997.

[8] E. Haber, U. Ascher, and D. Oldenburg. On optimization techniques for solving nonlinear inverse problems. *Inverse problems*, 16:1263–1280, 2000.

[9] P. C. Hansen. *Rank Deficient and Ill-Posed Problems*. SIAM, Philadelphia, 1998.

[10] M. Heinkenschloss and L.N. Vicente. Analysis of inexact trust region SQP algorithms. Technical report, TR 99-18, Rice University, September 1999.

[11] P. J. Huber. Robust estimation of a location parameter. *Ann. Math. Stats.*, 35:73–101, 1964.

[12] C. T. Kelley and E. W. Sachs. Local convergence of the symmetric rank-one iteration. *Computational Optimization and Applications*, 9:43–63, 1998.

[13] S. Knapek. Matrix-dependent multigrid homogenization for diffusion problems. *SIAM J. Sci. Comp.*, 20(2):515–533, 1999.

[14] J. Nocedal and S. Wright. *Numerical Optimization*. New York: Springer, 1999.

[15] A. Pidlisecky, R. Knight, and E. Haber. An assessment of feasibility of cone penetrometer based resistivity imaging system. *Submitted*, 2003.

[16] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2001.

[17] M. Wilt, D. Alumbaugh, H. Morrison, A. Becker, K. Lee, and M. Deszcz-Pan. Crosswell electromagnetic tomography: System design considerations. *Geophysics*, 60:871–885, 1995.