JOURNAL OF COMPUTATIONAL PHYSICS 75, 314–333 (1988)

Fast l_p Solution of Large, Sparse, Linear Systems: Application to Seismic Travel Time Tomography

JOHN A. SCALES, ADAM GERSZTENKORN, AND SVEN TREITEL

Amoco Production Company, Research Center, P.O. Box 3385, Tulsa, Oklahoma 74102

Received August 22, 1986; revised May 15, 1987

Least-squares inversion tends to be sensitive to small changes in the assumptions; in the statistical lexicon, it is not "robust." Least-squares solutions of noisy, ill-conditioned systems often exhibit unphysical, high-frequency oscillations which have to be damped or smoothed in some fashion. The iteratively reweighted least squares (IRLS) algorithm provides a means of computing approximate l_p solutions ($1 \leqslant p$). We will show that IRLS can be combined with the preconditioned conjugate gradient algorithm in order to solve large, sparse, rectangular systems of linear, algebraic equations very efficiently. Further, for $p \cong 1$, the resulting algorithm is extremely stable; it seems to give reasonably smooth solutions without minute adjustment of the parameters and does not become unstable if an overly optimistic error estimate is used. It also has the ability to reject, or at least significantly diminish, the influence of outliers in the data and does not require the use of damping. In addition, we will show how to efficiently include diagonal weighting of the observations and parameters in order to incorporate *a priori* information into the inversion. This combined inversion algorithm has wide applicability in "real-world" inverse theory, parameter estimation, filtering, etc., where data are noisy and not uniformly reliable. Numerical examples of the l_1 inversion of noisy seismic travel time data will be shown. © 1988 Academic Press, Inc.

INTRODUCTION

The potential advantages of computing l_1 solutions to real-world inverse problems have long been recognized. Eisenhart [1] describes the work of the protean scholar Roger Joseph Boscovich, S.J., who in the 1750s used a smallest absolute value criterion to reconcile geodetic measurements for the determination of the figure of the earth. Boscovich developed a simple geometrical method for computing the best l_1 line through any number of points. Gauss was aware of this work and in his *Theoria Motus* [2] describes an elegant algebraic method for computing the l_1 solution to an overdetermined system of linear, algebraic equations. Gauss' method is based upon the fact that the l_1 solution exactly satisfies a square subsystem whose size equals the number of columns in the original matrix (assuming full column rank). If one computes all of the solutions to these subsystems, it suffices to see which has the smallest l_1 residual. That the l_1 solution actually satisfies one of these subsystems exactly is, according to Gauss, easily shown; for a proof of this adumbration of the fundamental theorem of linear programming see [3].

0021-9991/88 $3.00

Claerbout and Muir [4] give numerous examples of the advantages of l_1 inversion vis-à-vis the more familiar l_2 (least-squares) inversion. For the most part, these advantages are based upon the fact that l_2 solutions tend to overstate the influence of outliers in the data. For highly ill-conditioned problems, as we shall see, even small amounts of Gaussian noise can seriously degrade the quality of least-squares solutions. The disadvantage of l_p optimization is that, except for $p = 2$, the problem is nonlinear. Until the mid 1970s most algorithms used for l_1 inversion were based upon relatively slow linear programming extensions of Gauss' method (e.g., [5]). An iterative approach to l_p inversion, $1 \leqslant p$, which is based upon solving a sequence of weighted least-squares problems, is called IRLS (for iteratively reweighted least squares). IRLS is usually attributed to Beaton and Tukey [6] although Schlossmacher [24] had previously published the l_1 version. Byrd and Pyne [7] give numerous references to the use of IRLS and provide convergence results. The IRLS algorithm has recently been successfully applied to the computation of digital filters by Yarlagadda *et al.* [8], to 1D acoustic inversion by Gersztenkorn *et al.* [9], and to the reduction of astronomical data by Branham [25]. These works are not, however, directly applicable to large, sparse problems. We will show how to combine IRLS with conjugate gradient (CG) least-squares inversion [10, 21]; the resulting algorithm, which is in fact a nonlinearly preconditioned conjugate gradient, is exceptionally efficient for the large, sparse problems encountered in geophysical tomography. Further, numerical studies show that the iterative reweighting ($p \cong 1$) stabilizes the inversion procedure in the presence of noise and ill-conditioning.

We will begin with a brief discussion of the IRLS algorithm. This will be followed by an extension of the sparse, conjugate gradient least-squares algorithm to include iterative reweighting. Finally, this combined algorithm will be applied to various synthetic examples of travel time tomography.

Iteratively Reweighted Least Squares

Linearized travel time tomography gives rise to large, sparse, rectangular systems of linear, algebraic equations

$$Ax = y, \tag{1}$$

where y is a vector containing travel time observations, x is a vector containing model parameters (slowness or inverse-velocity values referred to some discretization), and the matrix A is computed by ray tracing. For a complete discussion of seismic travel time tomography see [11, 12] and the references cited therein. For purposes of this paper, the salient features of Eq. (1) are these: The matrix A is large (perhaps 10^5 rows and 10^4 columns), sparse (less than 1% nonzero), and ill-conditioned (condition numbers as large as 10^5 are not uncommon). Further, due to undersampling and the presence of noise in the data, A may be numerically singular.

The standard approach to solving Eq. (1) is to compute a least-squares solution, usually damped, and to apply some sort of additional filtering or smoothing, either during the solution or after. Smoothing during iterative solution is sometimes used to stabilize the inversion. On the other hand, instead of seeking a vector x which minimizes the sum of the squared residuals of Eq. (1), one can consider the l_p optimization problem:

$$\min_x \sum_i \left| \sum_j A_{ij} x_j - y_i \right|^p, \qquad 1 \leqslant p. \tag{2}$$

The lower bound on p is necessary since $(\sum |x_i|^p)^{1/p}$ is not a norm for p less than one. By setting the x derivative of Eq. (2) equal to zero one arrives at the generalized normal equation

$$A^{\mathrm{T}} R A x = A^{\mathrm{T}} R y \tag{3}$$

for $r_i \equiv (Ax - y)_i \neq 0$, where R is the diagonal matrix with elements $|r_i|^{p-2}$. (For a derivation of this result see the Appendix.) The problem of zero or very small residuals will be discussed directly. Notice that for $p = 2$ (i.e., least-squares) the usual normal equations are recovered. On the other hand, for p less than 2 the weighting matrix R tends to diminish the influence of large residuals; this has the effect of eliminating the influence of outliers in the data, and, as we shall see, stabilizing the inversion. Note also that if the matrix R were to have constant coefficients, then Eq. (3) would be equivalent to an ordinary weighted least-squares problem.

It follows from the Minkowski inequality that any convex combination of solutions to Eq. (2) is also a solution. To see this, suppose that x and z are two solutions of Eq. (2) whose error is $\|Ax - y\| = \|Az - y\| = \gamma$, and that $0 \leqslant t \leqslant 1$. Then, adding and subtracting ty, one has

$$\begin{aligned} \|A\{tx + (1-t)z\} - y\| &= \|tAx + Az - tAz + ty - ty - y\| \\ &= \|t(Ax - y) + (1-t)(Az - y)\| \\ &\leqslant t\,\|Ax - y\| + (1-t)\,\|Az - y\| \\ &= t\gamma + (1-t)\gamma \\ &= \gamma. \end{aligned}$$

Using this one can show that for $p > 1$, Eq. (2) is uniquely soluble, provided A has full column rank, whereas for $p = 1$ it is not [14].

One obvious difficulty with choosing the weights to be the inverses of the residuals is that very small residuals might cause the algorithm to become unstable. But zero residuals just mean that we place infinite confidence in these observations. Huber [13] suggests replacing r_i with some lower cutoff, say ε, when $|r_i|$ is less than ε. This eliminates the problem of zero residuals and, as Byrd and Pyne [7] show, is sufficient to guarantee convergence of the IRLS algorithm. In practice this just means that beyond a certain point, all high-confidence observations are weighted the same. In addition, Gersztenkorn *et al.* [9] found that the results of IRLS were improved by normalizing the elements of R to lie between ε/r_{max} and 1. Now in fact we know from Gauss' method [2] that, in the absence of roundoff error, there will always be some exactly zero residuals, the number of which is equal to the column rank of the matrix. Therefore, solving Eq. (3) with the Huber taper will necessarily give somewhat different results than those obtained using linear programming [14].

The IRLS approach to solving Eq. (3) is to solve a sequence of weighted least-squares problems with recursively computed weighting matrices, thereby approximating the solution to the l_p minimization problem. Thus, the solution at the kth step of the iteration is

$$x_k = (A^T R_{k-1} A)^{-1} A^T R_{k-1} y, \tag{4}$$

provided A has full column rank. The rank deficient case is easily handled provided one uses a solution method which gives the pseudo-inverse solution at each step, such as conjugate gradient, SVD, or the row-action methods [10, 20]. The initial approximation used for R is simply the identity matrix; the initial step therefore is to compute an approximate least-squares solution. Convergence results are given in [7].

Implementation

There are several approaches available for implementing IRLS. Coleman *et al.* [16] use standard orthogonalization methods (Householder transformations or SVD) to compute a direct solution to each weighted l_2 problem. For large, sparse problems orthogonalization by Householder transformation causes unacceptable fill in the matrix. Although there are a number of direct methods available for sparse least-squares problems, such as orthogonalization by sparse Givens rotations, we prefer to use iterative methods, in particular, the conjugate gradient method, because:

(a) The matrix itself is never directly accessed, so there is no fill, and the matrix $A^T A$ is not computed.

(b) Sparse CG software is fairly simple; the modifications necessary for iterative reweighting take just a few dozen lines of code.

(c) CG has proved to be accurate and efficient for a wide range of problems, including the singular problems encountered in tomography and inverse scattering (e.g., [9–12, 15]).

It is important to note however that, in principle, any iterative least-squares solver could be used within IRLS; we have chosen CG primarily for its simplicity.

We begin IRLS by computing a "small" number of CG iterations; if a good estimate of the noise level is known, the unweighted iterations should be terminated well before the fractional reduction in the residual $\|r_n\|/\|r_0\|$ has reached this level [18]. Keeping the number of initial, unweighted CG iterations small, at the possible expense of a greater number of weighting steps, prevents the deleterious effects of undamped least-squares from cropping up in the first IRLS step. Once the iterative weighting has begun the algorithm is very stable and relatively insensitive to changes in the parameters. At this point, the residual vector, which is automatically computed by CG, is used to compute the first weighting matrix R. We then apply CG to the weighted normal equation, Eq. (3) (see below), iterating until an intermediate stopping criterion is satisfied; this is usually just a fixed number of iterations. A new weighting matrix is computed and the procedure is repeated until convergence to the desired degree of accuracy is achieved. For a more complete discussion of convergence criteria see [18].

To apply the sparse CG algorithm to the (iteratively) weighted normal equations, they are written in factored form

$$A^{\mathrm{T}}R(Ax-y)=0. \tag{5}$$

A least-squares form of CG, developed by Hestenes and Stiefel [21] and referred to by Paige and Saunders as CGLS [18], uses the factored form of the normal equations, with $R=I$, and successively applies A and A^{T} to an initial approximation in order to generate the solution. This implicit approach is significantly more stable than application of the symmetric CG algorithm to the normal equations [26]. One observes that the weighting matrix R only appears as a right-multiplier of A^{T} in Eq. (5). Now, since CG requires A^{T} only in the form of sparse A^{T}-times-vector multiplies, it is a straightforward matter to rewrite the A^{T}-times-vector subroutine to include the diagonal weighting matrix R. Since R is diagonal, there is no loss of sparsity and the increase in the number of floating-point operations is fairly small. Further, it is a simple matter to turn reweighting on or off as desired.

There are many data structures available for the matrix-vector multiplies. For this paper we have chosen an especially simple one, one which nevertheless works quite well on a scalar machine with sufficient memory to keep the whole matrix in core; Pissanetsky [27], for example, gives many alternatives. We will use a data structure in which the NZERO nonzero elements of the matrix are stored row by row in the array ELEM, and the row and column indices are stored in integer arrays IROW and ICOL. Then an algorithm for doing $y=A^{\mathrm{T}}\cdot x$, where y and x are arbitrary vectors of the appropriate dimensions, is simply

```
do k = 1, NZERO
y(ICOL(k)) = y(ICOL(k)) + ELEM(k) * x(IROW(k))
continue.
```

At each reweighting step the diagonal elements of the R matrix are put into a 1-D array. The weighted matrix multiply $y = A^T R \cdot x$ is then

```
do k = 1, NZERO
y(ICOL(k)) = y(ICOL(k)) + ELEM(k) * x(IROW(k)) * R(IROW(k))
continue.
```

Since at each reweighting step the array R is constant, this expression can be used to do the weighted least-squares problem, where the weights are applied to the rows or observations. Thus if one has a priori knowledge about the accuracy of the observations, that knowledge can be incorporated into the inversion at this stage. Suppose, for example, that the travel time data are not uniformly reliable. Perhaps a weight factor could be computed for each travel time based, e.g., on the signal to noise ratio or trace-to-trace correlation. These weights could be stored and used later when the inversion is performed. Similarly, column or parameter weighting can be incorporated. If one has a priori information about the parameters, or simply wishes to equilibrate the columns of the matrix, one can include a diagonal weighting matrix H and solve the equivalent (for H diagonal and nonzero) problem $AHH^{-1}x = AHx' = y$ for x' and then recover x. This involves changing the $y = A \cdot x$ routine from

```
do k = 1, NZERO
y(IROW(k)) = y(IROW(k)) + ELEM(k) * x(ICOL(k))
continue
```

to

```
do k = 1, NZERO
y(IROW(k)) = y(IROW(k)) + ELEM(k) * x(ICOL(k)) * H(ICOL(k))
continue.
```

Note that if a good diagonal preconditioner H is known, one can replace ELEM(k) by ELEM(k) * H(ICOL(k)) once and for all and considerably reduce the number of floating-point operations. Now, the bulk of the floating-point operations in a conjugate gradient solution of the least-squares problem are accounted for by matrix-vector inner products and transposed matrix-vector inner products, one each per iteration. There are NZERO multiplies and NZERO adds per matrix-vector inner product. So the total number of floating-point operations is 2 NZERO adds and 2 NZERO multiplies per iteration. If either row or column weighting is included this is increased to 2 NZERO adds and 3 NZERO multiplies per iteration. Thus, if the total number of iterations is held fixed, the additional work required to do the weighting is modest.

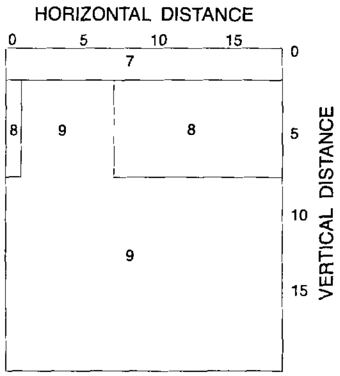

FIG. 1. Geometry for VSP tomography example. The middle layer contains a velocity anomaly. Starting with a three layer initial guess, travel time perturbations are inverted in order to recover the anomaly.

NUMERICAL EXAMPLES

We will consider two examples of tomographic inversion. The tomographic aspects of both of these problems have been described elsewhere. The first example, a synthetic vertical seismic profile (VSP) in which a square velocity anomaly is to be reconstructed, is described in [10]. The second example, the tomographic inversion of a complicated synthetic seismic reflection survey, is described in [11]. In both examples the IRLS minimization norm is taken to be $p = 1$. In order to mitigate the ill-conditioning all calculations are done using 64-bit arithmetic (on an IBM 3090 scalar computer).

Figure 1 shows a VSP geometry with a square velocity anomaly in the middle layer. Figure 2 shows the ray illumination from 18 sources on the surface to 18

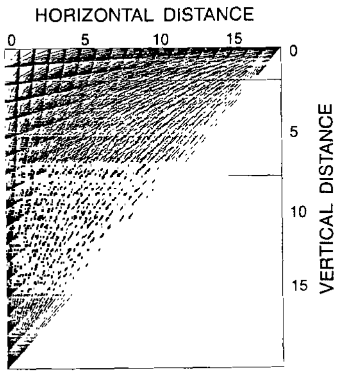

FIG. 2. 18 sources are positioned along the top surface and 18 receivers are positioned along the side of the model.

receivers on the left side of the model. In this example, straight rays are used; in the reflection tomography example curved rays were traced by numerically solving two-point boundary value problems for the ray equation (i.e., the Euler–Lagrange equation for Fermat's principle). The upper left triangular region was covered with 136 square cells of constant slowness; the matrix is therefore 324 rows by 136 columns. The initial guess consisted of the three layers without the square anomaly. The idea of linearized travel time inversion is to trace rays through an initial slowness (inverse–velocity) model and use the differences between the computed travel times and the observed travel times to invert for the perturbations to the slowness model. These slowness perturbations are then added to the initial model in order to produce an updated model. To graphically display the solutions, the slowness perturbations are plotted as a function of cell number. The resulting plots give the perturbations to the slowness vector required to reconstruct the anomaly. Since the anomaly was covered by 6 rows of 6 cells each, the exact solution corresponds to 6 perturbation steps of height 0.0138 ($=\frac{1}{8}-\frac{1}{9}$), each of which is 6 cells wide. Figure 3 shows the conjugate gradient inversion of the travel time residuals for various numbers of iterations and the exact solution. Since the travel times were produced with straight ray modeling and straight rays were used to invert them, the data were noise free and the reconstruction after 500 CG iterations was nearly perfect. Interestingly, even in this case all of the singular values are not accurately determined. The SVD solutions given in [10] are highly inaccurate if all 136 singular values are used; the conjugate gradient solution shows no such difficulty. Next we tried adding noise to the right-hand side (the travel times). First, an isolated spike, 0.9 times the maximum absolute value of the travel time residuals, was added to one of the 18^2 travel times. Figure 4 shows the exact solution (no spike), 25 and 250 iterations of conjugate gradient (no reweighting), and 25 iterations of CG followed by 9 reweighting steps of 25 CG iterations each (denoted 10×25 in the figure).

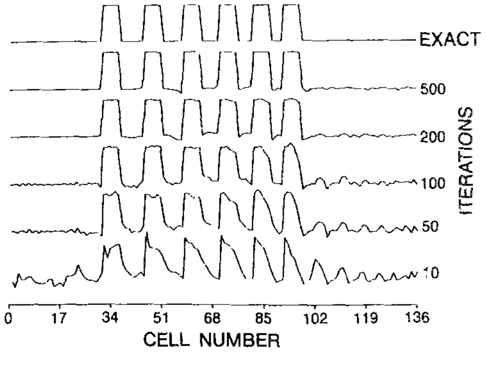

FIG. 3. Conjugate gradient solution for the slowness perturbations required to reconstruct the anomaly in the noise free case. The exact solution and the solutions for increasing numbers of iterations are shown.

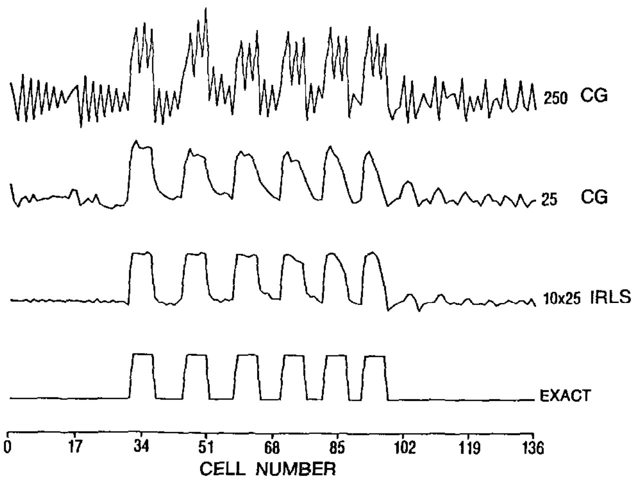

FIG. 4. An isolated spike is added to the right-hand side. Shown are the exact solution (with no spike) and the solutions obtained with 25 and 250 iterations of CG and 10 IRLS steps of 25 CG iterations each. The figures labeled 250 and 10×25 therefore represent the same number of CG iterations.

With experience we have found that IRLS is relatively insensitive to the number of CG iterations per reweighting step. After 250 iterations, the least-squares algorithm became unstable, whereas the l_1 solution via IRLS was relatively unaffected by the noise. The fact that l_1 inversion is more stable than l_2 inversion in the presence of isolated bursts of noise is well known (for example, [4, 8, 9]).

In order to see to what extent, if any, the beneficial effects of IRLS are due merely to restarting the CG, we carried out a 10×25 IRLS experiment as in Fig. 4, but with trivial weighting (minimization norm equal to 2). These results are shown in Fig. 4a, along with the l_1 result. Comparing this with Fig. 4 one can see that indeed a small improvement is due to restarting, but that the lion's share of the improvement is due to the iterative reweighting. For more details on restarted CG-type methods see, e.g., [28].

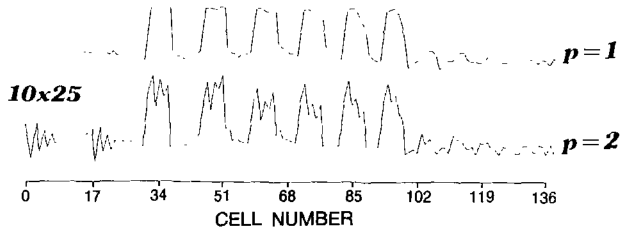

FIG. 4a. The same problem as in Fig. 4. To see to what extent the improvements shown in Fig. 4 are due to simply restarting the CG we tried a (10×25) IRLS sequence with the minimization norm equal to 2, i.e., using unit weights. By comparison with Fig. 4 we see some improvement over the unrestarted CG (1×250), but the bulk of the improvement is clearly due to the iterative reweighting.

Having looked at the effects of a short burst of noise in the data, suppose instead that one adds Gaussian noise to all of the travel times. Figures 5 and 6 show two such inversions. Each figure shows 130 CG iterations (no reweighting) and a reweighting sequence which consists of 25 CG iterations followed by 3 reweighting steps of, respectively, 30, 35, and 40 iterations, for a total of 130 iterations. In Fig. 5 the mean of the noise is 0 and the standard deviation is 0.01 times the max of $|y_i|$, the right-hand side (r.h.s.). In Fig. 6 the mean of the noise is 0 and the standard deviation (SD) is 0.005 times the max of $|y_i|$.

The condition number of the matrix A for this problem is roughly 10^5. This is sufficiently large that even the relatively small perturbations involved can cause drastic effects. In Fig. 6 the noise level is so small that standard least-squares inversion works well, provided that the number of iterations is limited. Figure 7 shows the effects of doing too many least-squares iterations; the top trace is 400 iterations of CG, the middle trace is IRLS (consisting of a weighting sequence of 80, 90, 100, and 110 iterations), and the bottom trace is 100 iterations of CG. In Fig. 5, the effects of the ill-conditioning of the matrix are obvious even with a limited number of iterations. This points to a basic property of the CG algorithm: one observes that in the early stages of iteration, changes in the solution are primarily due to the largest eigenvalues (in this case of A^TRA), whereas the effects of the smallest eigenvalues become more pronounced as iteration proceeds. (For a discussion of the connection between CG and the Lanczos algorithm, from which CG inherits this property, see [17].) Therefore, noise which is small compared to the large eigenvalues but comparable to the small eigenvalues will have a comparatively large effect on the small eigenvalues, and hence on the solution, after sufficiently many iterations. A key advantage of IRLS is that one need not be quite so careful about terminating the iteration or estimating the level of noise in the data.

To see the effects of the reweighting procedure on the eigenvalue spectrum of A^TRA, we did singular value decompositions of this matrix at various stages of the IRLS procedure for the Gaussian perturbations used for Fig. 5. Figure 8 shows the singular values of A^TRA for $N=0$, 1, 2, and 5 reweighting steps; each reweighting step involves 50 CG iterations. For $N=0$, R is simply the identity matrix. The condition number of A^TRA was essentially unchanged from step to step. Next we computed winnowed SVD solutions to $A^TRAx=A^TRy$, where R was computed with 50 CG iterations. Figures 9 and 10 show the solutions computed with singular value lower cutoffs of, respectively, 0.001 and 0.1. The solutions labeled RWT were com- [illegible] reweighting. Again, one can see that in the presence of noise iterative reweighting improves the stability of the inversion.

For purposes of comparison we tried another widely used least-squares solver on this model with the isolated spike in the data. The results of the LSQR algorithm of Paige and Saunders [18–19] are shown in Fig. 11. LSQR incorporates damping (adding a small positive term to reduce singularity and thereby improve stability) at essentially no extra cost. The choice of damping parameter is somewhat problematical, so we show results for various values of damping and maximum

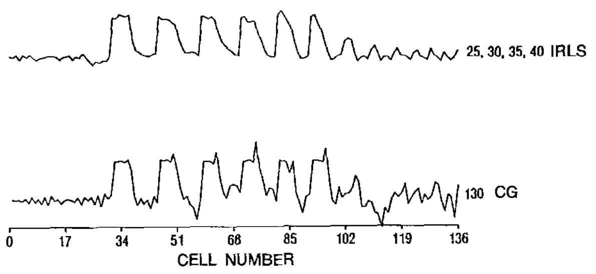

FIG. 5. Gaussian noise is added to all of the elements of the r.h.s. In the first example, the mean of the noise is 0 and the standard deviation is 0.01 times the maximum absolute value of the r.h.s. Shown are 130 iterations of CG and 4 steps of IRLS totaling the same number of CG iterations.

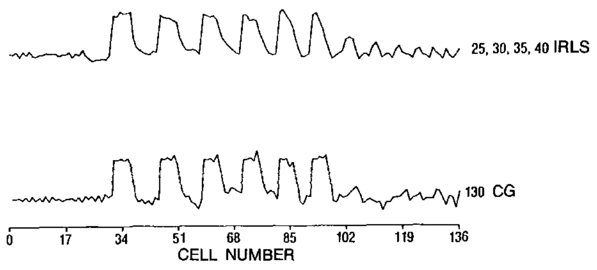

FIG. 6. As in Fig. 5, Gaussian noise is added to the r.h.s. In this case the mean of the noise is 0 and the standard deviation is 0.005 times the maximum absolute value of the r.h.s. Shown are 130 iterations of CG and 4 steps of IRLS totaling the same number of CG iterations.

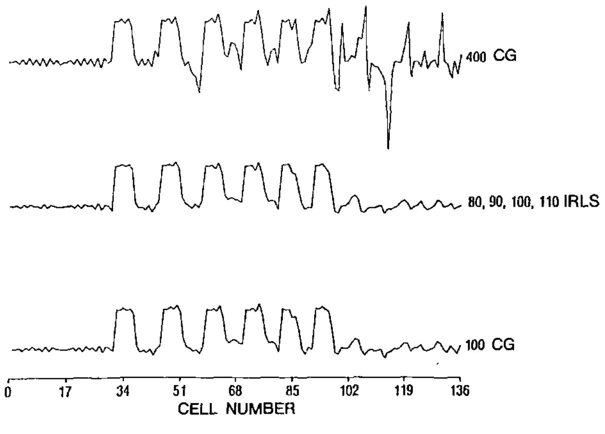

FIG. 7. The same problem as in Fig. 6; showing the effects of too many iterations in unweighted CG. The top and bottom traces are unweighted CG. The middle trace was computed with 4 reweighting steps.

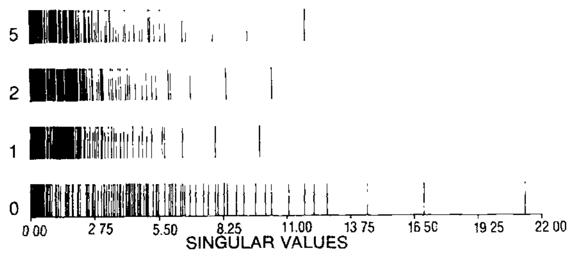

FIG. 8. Eigenvalue spectrum (computed with SVD) of A^TRA for 0, 1, 2, and 5 reweighting steps. 50 CG iterations are done for each reweighting step. The condition number of A^TRA is essentially unchanged between steps.

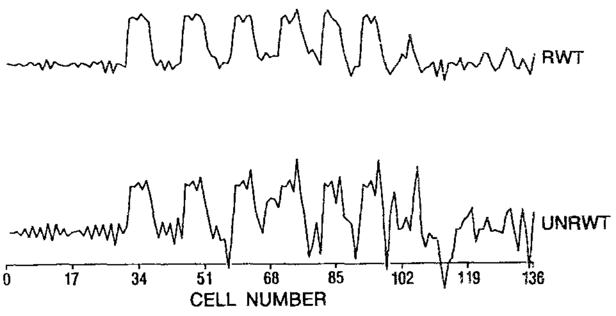

FIG. 9. Winnowed SVD solutions to $A^TRAx = A^TRy$ (labeled RWT) and $A^TAx = A^Ty$, using a singular value cutoff of 0.001. R is computed using 50 CG iterations.

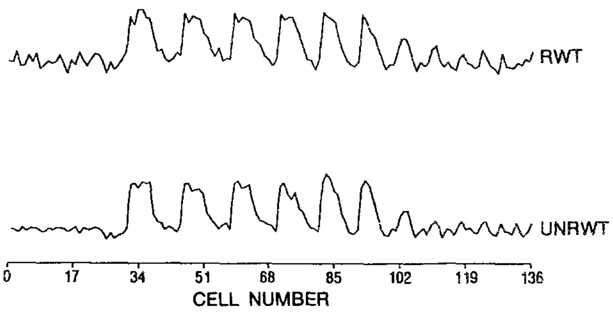

FIG. 10. Winnowed SVD solutions to $A^TRAx = A^TRy$ (labeled RWT) and $A^TAx = A^Ty$, using a singular value cutoff of 0.1. R is computed using 50 CG iterations.

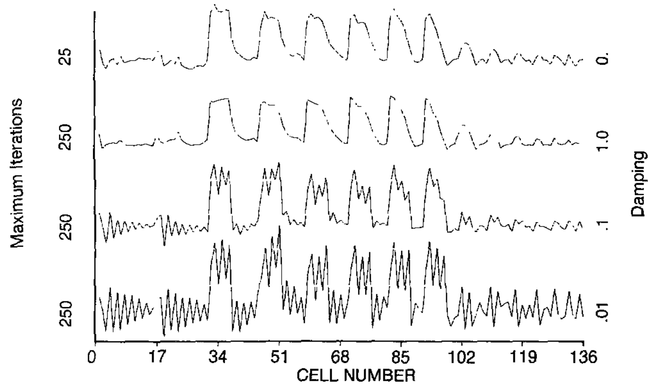

FIG. 11. An isolated spike (as in Fig. 4) is added to the r.h.s. The top trace is 25 undamped iterations of LSQR. The next three traces show various levels of damping applied to LSQR with a maximum number of iterations equal to 250. The comparable IRLS results are in Fig. 4.

iteration number. Not surprisingly, the effects of explicit damping are similar to the effects of reducing the total number of iterations. On the unperturbed model LSQR gives results which are virtually identical to least-squares CG. Figure 11 should be compared with the IRLS results in Fig. 4. If we stop after only 25 iterations the LSQR solution and IRLS solution are similar. And if we over-damp the problem, the LSQR inversion remains stable. But no matter what we do we are unable to recover the solution to the extent that we can with the l_1 solver.

For the next example we considered a synthetic seismic reflection survey. Finite difference synthetic seismograms were computed for the geologic model shown in Fig. 12. Grey-scale velocities in feet/second are shown on the left. The model is 32000 ft across and 16000 ft deep. It was covered with approximately 2000 square cells of constant slowness; and about 4500 travel times were extracted from the unstacked, common-source seismograms, thus giving a linear system (Eq. (1)) of 4500 rows and 2000 columns.

It should be emphasized that this problem is well suited to solution via damped least-squares since the data are exceptionally clean and the seismograms easily interpreted; there are no outliers, for example. When inverting real data, outliers are an ever-present problem. Thus one would expect that the real advantages of robust methods, such as IRLS, would be seen more clearly in application to real data.

Figure 13 shows the first approximation to the model used in the tomographic inversion. (See [11] for details.) Even though the initial approximation consists of flat layers, the model is not uniformly illuminated. This is due to the inability to get travel times associated with the deeper or more complex portions of the model. The relative cell illumination is shown in Fig. 14. The cell illumination consists of the total ray length per cell divided by a characteristic cell length; the definition of this

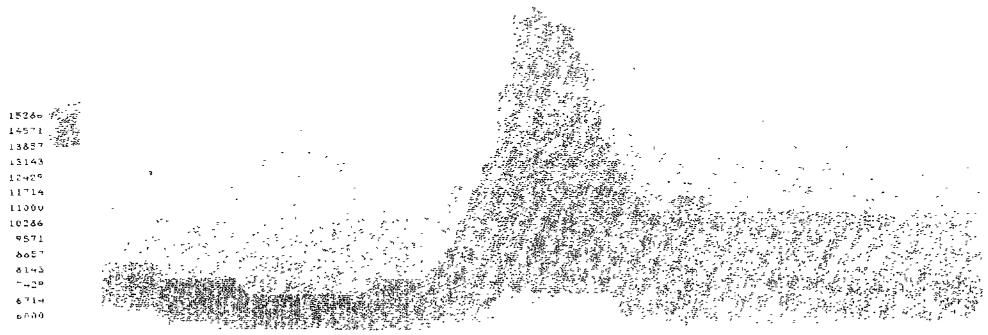

FIG. 12. Finite difference synthetic seismograms were produced for this velocity model. Velocities are shown in grey-scale code. Travel times (for the reflected rays) were then extracted from the unstacked data and compared to travel times computed by numerically tracing rays through the initial model shown in Fig. 13.

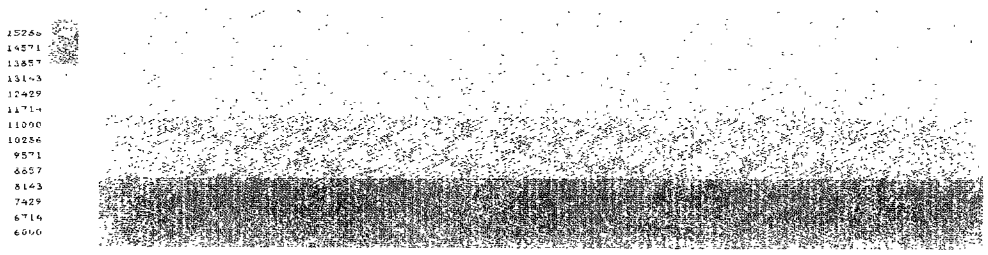

FIG. 13. Initial velocity model used in the tomographic inversion.

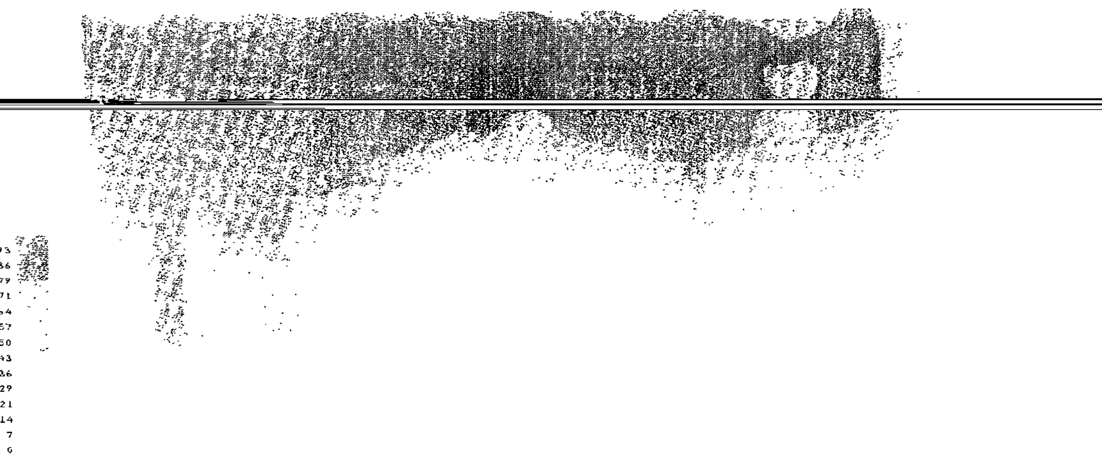

FIG. 14. The relative cell illumination. This is the total length of the rays passing through a given cell normalized by a characteristic cell length.

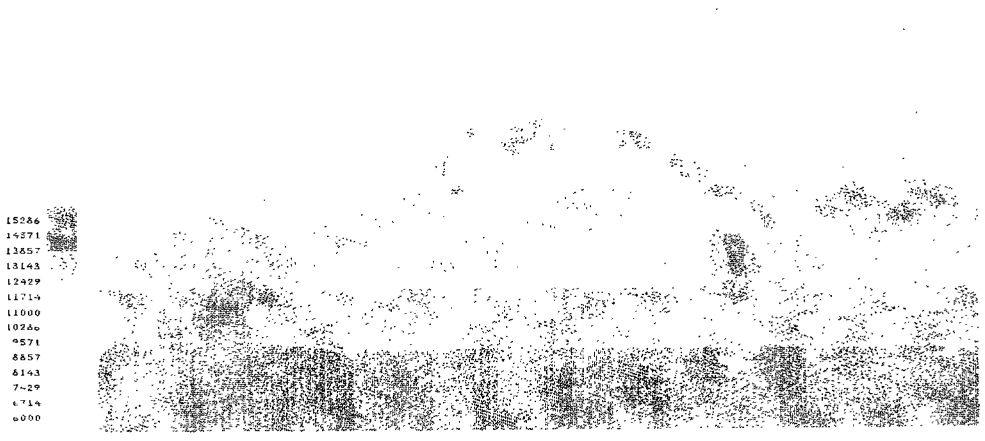

FIG. 15. Least-squares velocity tomogram computed with damped LSQR. Medium damping.

cell length is somewhat arbitrary and is simply a reflection of the fact that a small cell and a large cell might be equally well illuminated with very different total ray lengths. The portions of the model which are poorly illuminated are left unperturbed. Figures 15 and 16 show velocity tomograms that were computed with damped LSQR. The damping parameters were chosen by trial and error. The dimension of the matrix elements is length (i.e., of ray path) and the scale is determined by the size of the tomography cells. Since the tomography cells are 475 ft^2, the damping is, very roughly speaking, large or small relative to this number. The damping used in Fig. 15 was 2500, while for Fig. 16 it was 7500; increasing the damping beyond this value did not significantly change the results. Figure 17 shows the l_1 solution computed with IRLS; no explicit damping is used in the l_1 code. For IRLS and LSQR the travel times were assumed to be accurate to about 1%; in both cases convergence was achieved in less than 50 iterations. The solution times for both methods are comparable, less than 10 s CPU on the IBM 3090 scalar computer. The IRLS tomogram is very similar to the more heavily damped of the two LSQR solutions.

From the standpoint of tomography, Figs. 15–17 are first iterations in an iterative nonlinear inversion procedure. For more details about how this iteration procedure is continued see [11]. As a final step we usually apply a fine-grain median filter to the the output velocity model in order to smooth the solution for the next stage of the iterative inversion. Median filters are extremely useful smoothers for computed tomograms [22]. They ignore outliers instead of averaging them into the solution. Further, median filters preserve edges, ubiquitous features in exploration seismologic models. The median filtered version of the IRLS solution is shown in Fig. 18. This can be compared with the initial model shown in Fig. 12.

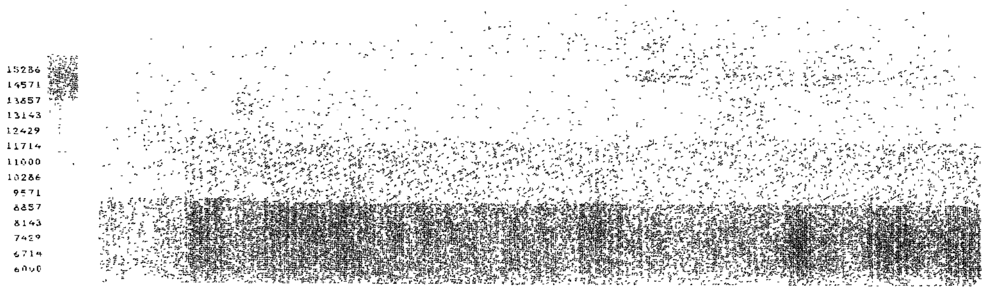

FIG. 16. Least-squares velocity tomogram computed with damped LSQR. High damping.

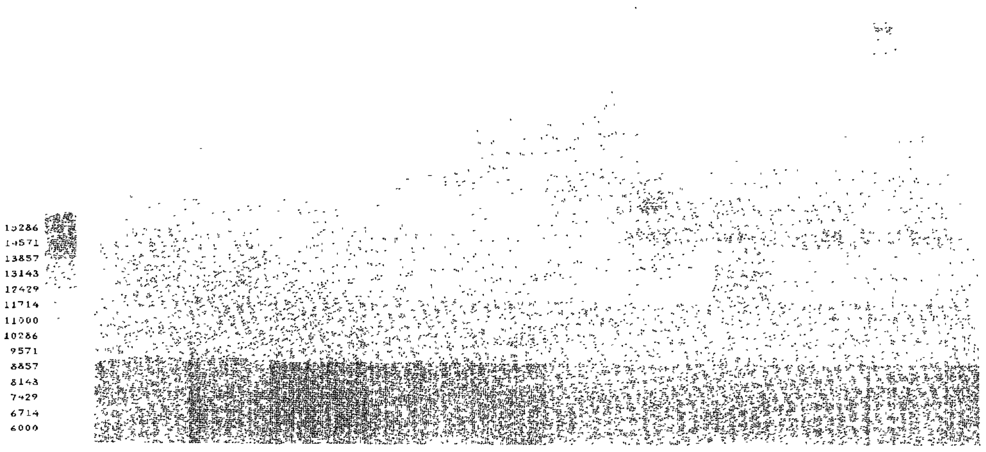

FIG. 17. l_1 velocity tomogram computed with IRLS. No damping is required.

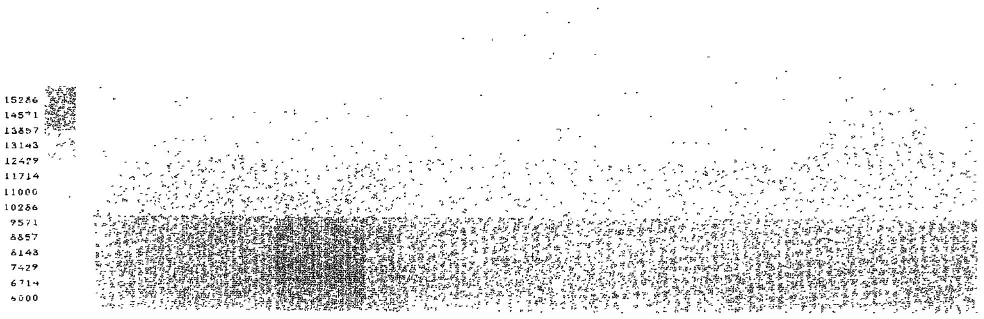

FIG. 18. Same as Fig. 17 after a fine-grain median filter has been applied to smooth the result.

The smoothed IRLS tomogram shows good agreement with the well-illuminated upper part of the model.

Conclusions

We have shown how to efficiently solve large, sparse, rectangular systems of linear, algebraic equations in the l_p norm through a combination of iteratively reweighted least squares and conjugate gradient. The resulting algorithm is very fast and (for $p \cong 1$) appears to be quite stable in the presence of noise and ill- [redacted] LSQR.

In addition, IRLS is very easy to program and use. We have also shown how to efficiently incorporate a priori weighting of the observations and parameters within the IRLS framework. Finally, the sparse implementation of IRLS via conjugate gradient is virtually as fast as a single application of least-squares CG or LSQR to the l_2 problem.

Although we have confined the numerical examples to synthetic travel time tomography problems, we have successfully applied IRLS to a variety of problems including acoustic inverse scattering [9], the tomographic inversion of real seismic reflection data [11], and the computation of refraction statics corrections. In addition, we have compared the performance of IRLS with that of many least-squares algorithms including iterative back-projection (IBP), many variations of the algebraic reconstruction technique (ART) and the simultaneous iterative reconstruction technique (SIRT), Gauss–Seidel, LSQR, least-squares conjugate gradient, and others. We have found that IRLS consistently performs as well as or better than all of these methods whenever noise and ill-conditioning are significant. As previously mentioned, however, any of these iterative least-squares solvers could, in principle, be adapted to perform IRLS; we have simply looked at one method, the conjugate gradient algorithm of Hestenes and Stiefel.

Appendix

In this Appendix, Eq. (3) will be derived from Eq. (2). Using the notation $\partial_i = \partial/\partial x_i$, it suffices to compute

$$\sum_i \partial_k \left| \sum_j A_{ij} x_j - y_i \right|^p .$$

Assuming for the moment that the argument of the absolute value function is never zero one has, recalling that $\operatorname{sgn}(x) \equiv x/|x|$,

$$\sum_i \partial_k \left|\sum_j A_{ij}x_j - y_i\right|^p \equiv \sum_i \partial_k |r_i|^p = \sum_i \operatorname{sgn}(r_i)\, p\, |r_i|^{p-1} A_{ik}$$
$$= \sum_i r_i p\, |r_i|^{p-2} A_{ik}$$
$$= [A^{\mathrm{T}} R(Ax - y)]_k,$$

where the matrix R is defined to be $\operatorname{diag}(p\, |r_i|^{p-2})$.

Since the last expression is set to zero in order to solve the minimization problem, the multiplicative factor p can be ignored in the weighting matrix; this gives Eq. (3). As has already been mentioned, if p is less than 2 the above expression becomes infinite for zero residuals. This problem is avoided by replacing $|r_i|$ with a lower cutoff when it falls below that cutoff. For more details see [13, 23].

Acknowledgments

The authors are grateful to the two referees for a number of very valuable suggestions.

References

1. C. Eisenhart, "Boscovich and the Combination of Observations," *Roger Joseph Boscovich*, L. L. Whyte, ed. (Allen & Unwin, London, 1961), p. 200.
2. C. F. Gauss, *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium* (Hamburg, 1809). (English translation by C. H. Davis, Little, Brown, Boston, 1857.)
3. P. Bloomfield and W. Steiger, *Least Absolute Deviation* (Birkhäuser, Boston, 1983), Chap. 1.
4. J. Claerbout and F. Muir, *Geophysics* **38**, 826 (1973).
5. I. Barrodale and F. Roberts, *Commun. ACM* **17**, 319 (1974).
6. A. Beaton and J. Tukey, *Technometrics* **16**, 147 (1974).
7. R. Byrd and D. Pyne, Johns Hopkins University Tech. Rep. 313, 1979 (unpublished).
8. R. Yarlagadda, J. Bee Bednar, and T. Watt, *IEEE Trans. Acoust. Speech Signal Process.* **33**, 174 (1985).
9. A. Gersztenkorn, J. Bee Bednar, and L. Lines, *Geophysics* **51**, 357 (1986).
10. J. Scales, *Geophysics* **52**, 179 (1987).
11. R. Bording, A. Gersztenkorn, L. Lines, J. Scales, and S. Treitel, *Geophys. J. R. Astron. Soc.* **90**, 285 (1987).
12. G. Nolet, *J. Comput. Phys.* **61**, 463 (1985).
13. P. Huber, *Robust Statistics* (Wiley, New York, 1981).
14. J. Scales and S. Treitel, *IEEE Trans. Acoust. Speech Signal Process.* **35**, 581 (1987).
15. R. Wang and S. Treitel, *Geophysics* **38**, 310 (1973).
16. D. Coleman, P. Holland, N. Kaden, V. Klema, and S. Peters, *ACM Trans. Math. Software* **6**, 327 (1980).
17. G. Golub and C. Van Loan, *Matrix Computations* (Johns Hopkins, Baltimore, 1983), Chap. 9.
18. C. Paige and M. Saunders, *ACM Trans. Math. Software* **8**, 43 (1982).
19. C. Paige and M. Saunders, *ACM Trans. Math. Software* **8**, 195 (1982).

20. Y. Censor, *SIAM Rev.* **23**, 444 (1981).
21. M. Hestenes and E. Stiefel, *Nat. Bur. Stand. J. Res.* **49**, 409 (1952).
22. A. Gersztenkorn and J. Scales, *Geophys. J. R. Astron. Soc.*, in press.
23. A. Gersztenkorn, M.S. thesis, University of Tulsa, 1984 (unpublished).
24. E. Schlossmacher, *J. Amer. Statist. Assoc.* **68**, 857 (1973).
25. R. Branham, Jr., *Q. J. R. Astron. Soc.* **27**, 182 (1986).
26. R. Chandra, Ph.D. thesis, Yale University, 1978 (unpublished).
27. S. Pissanetsky, *Sparse Matrix Technology* (Academic Press, London, 1984).
28. S. Eisenstat, H. Elman, and M. Schultz, *SIAM J. Numer. Anal.* **20**, 345 (1983).