Stanford Exploration Project, Report 103, April 27, 2000, pages 205–266

Stanford Exploration Project, Report 103, April 27, 2000, pages 205–266

Huber solver versus IRLS algorithm for quasi L1 inversion

Antoine Guitton[1]

ABSTRACT

The l^1 problem can be approximated using either Iteratively Reweighted Least Squares methods (IRLS) with an appropriate weighting function or the Huber misfit function with an appropriate solver. Comparisons of both methods show that they give very similar results as they handle problems suited to l^1 properly. In addition, the nonlinear scheme, using the Huber norm, generally converges better than the linear IRLS algorithms on velocity-stack inversion. Finally, the Huber solver is far easier to operate because it is sensitive to fewer parameters than IRLS.

INTRODUCTION

Robust error measures such as the l^1 norm have a number of uses in geophysics. As measures of data misfit, they show considerably less sensitivity to large measurement errors than does the least-squares (l^2) measure. In a recent work, Guitton and Symes (1999) show that a possible approximation of the l^1 norm is viable using the Huber misfit function (Huber, 1973). In particular, their work on velocity inversion, using the hyperbolic Radon transform, suggests that (1) the Huber function gives far more robust model estimates than does least squares, (2) its minimization using a standard quasi-Newton method is comparable in computational cost to least-squares estimation using conjugate gradient iteration, and (3) the result of Huber data fitting is stable over a wide range of choices for the $l^2 \rightarrow l^1$ threshold and total number of quasi-Newton steps. For the same geophysical problem, Nichols (1994) and Hugonnet (1998) use a different implementation of the hybrid l^2-l^1 norm, using an IRLS algorithm with a thoroughly chosen weighting function. Both strategies (IRLS and the Huber solver) show similar results, but at this stage, no direct comparisons between the two methods have been carried out.

This paper will compare the efficiency of the these two robust solvers with synthetic and real data. First, I will briefly review both methods and describe algorithmic issues. Then, I will compare the Huber solver and IRLS performing velocity inversion as suggested by Thorson and Claerbout (1985), Nichols (1994), and Lumley et al. (1994).

[1]**email:** antoine@sep.stanford.edu

IRLS AND THE HUBER SOLVER, AN OVERVIEW

Minimizing the Huber function

In the work reported here, I use a hybrid l^1-l^2 error measure proposed by Huber (Huber, 1973):

$$M_\epsilon(r) = \begin{cases} \frac{r^2}{2\epsilon}, & 0 \le |r| \le \epsilon \\ |r| - \frac{\epsilon}{2}, & \epsilon < |r| \end{cases}$$

I will call $\sum_{i=1}^{N} M_\epsilon(r_i)$ the *Huber misfit function* H_ϵ, or Huber function for short (Figure 1). Note that the Huber function is smooth near zero residual, and weights small residuals by the mean square. It is reasonable to suppose that the Huber function, while maintaining robustness against large residuals, is easier to minimize than l^1. The parameter ϵ, which controls the limit between l^1 and l^2, is called the Huber threshold.

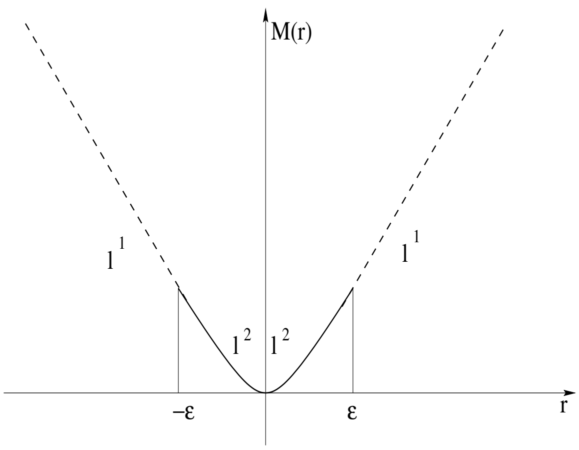

Figure 1: Error measure proposed by Huber (Huber, 1973). The upper part above ϵ is the l^1 norm, while the lower part is the l^2 norm. antoine2-huber [NR]

The problem is this: given some observed data $\mathbf{d_{obs}}$, we want to find the best model $\mathbf{m}$ that explains the data *via* the operator $\mathbf{H}$. This may be posed in terms of an inverse problem leading to the minimization of the function

$$f(\mathbf{m}) = E(\mathbf{Hm} - \mathbf{d_{obs}}), \tag{1}$$

where E is an error measure function. As discussed above, I will use the Huber function and try to minimize

$$f(\mathbf{m}) = H_\epsilon(\mathbf{Hm} - \mathbf{d_{obs}}). \tag{2}$$

We seek to find the stationary point $\mathbf{m}^*$ of the function f. This solution point satisfies $\mathbf{f}'(\mathbf{m}^*) = \mathbf{0}$. This is a nonlinear system of equations, and from Taylor expansion we have

$$\mathbf{f}'(\mathbf{m} + \delta\mathbf{m}) \simeq \mathbf{f}'(\mathbf{m}) + \mathbf{f}''(\mathbf{m})\delta\mathbf{m}$$

if $\|\delta\mathbf{m}\|$ is sufficiently small. The *Newton's method* consists in finding $\delta\mathbf{m}$ such that

$$\mathbf{H}\delta\mathbf{m} = -\mathbf{f}'(\mathbf{m}) \text{ with } \mathbf{H} = \mathbf{f}''(\mathbf{m}),$$

and computing the next iterate by

$$\mathbf{m}_{n+1} = \mathbf{m}_n + \alpha_n \delta \mathbf{m}_n \quad (3)$$

where α_n is a steplength given by a Line Search algorithm. The general process of the program is then

1. compute the gradient $\mathbf{f}'(\mathbf{m})$
2. compute $\delta\mathbf{m}_n = -\mathbf{H}^{-1}\mathbf{f}'(\mathbf{m})$
3. compute α_n using a line search
4. update the solution $\mathbf{m}_{n+1} = \mathbf{m}_n + \alpha_n \delta \mathbf{m}_n$
5. update the Hessian $\mathbf{H}$
6. go to 1.

Because the Huber function is not twice continuously differentiable, the Hessian is not computed directly but approximated using a limited Memory BFGS update (Guitton, 2000), as proposed by Nocedal (1980) and Liu and Nocedal (1989). I have implemented for the line search a More and Thuente (More and Thuente, 1994) algorithm, which ensures sufficient decrease for the function f and obeys curvature conditions (the so-called *Wolfe conditions*, Kelley (1999)), thus guaranteeing that a quasi-Newton update is possible. The steplength $\alpha_n = 1$ is always tried first, saving significant computation. These choices lead to good performances for both convergence rate and computation cost. I call "Huber solver" the algorithm detailed above when used with the Huber norm.

The Iteratively Reweighted Least Square method

The IRLS implementation of the hybrid l^2-l^1 norm differs greatly from the Huber solver. In this section, I follow quite closely what Nichols (1994) and Darche (1989) suggested in previous reports. The main idea is that instead of minimizing the simple l^2 norm, we can choose to minimize a weighted residual. The objective function becomes

$$f(\mathbf{m}) = \|\mathbf{WHm} - \mathbf{Wd}_{obs}\|_2. \quad (4)$$

A difference between the Huber solver and IRLS clearly arises here since we still solve a least-square problem with IRLS and not with the Huber solver. The IRLS method weights residuals within a linear l^2 framework and Huber uses either l^2 or l^1 following the residual with a nonlinear update. A particular choice for $\mathbf{W}$ will lead to the minimization of the l^2-l^1 norm, however. In this paper, I choose

$$w_{ii} = \frac{1}{[1 + (r_{ii}/\epsilon)^2]^{1/4}}, \quad (5)$$

with $r_i = \mathbf{Hm}_i - \mathbf{d}_i$ and ϵ a damping parameter. With this particular choice of $\mathbf{W}$, minimizing f is equivalent to minimizing (Bube and Langan, 1997)

$$J(\mathbf{m}) = \sum_{i=1}^{n} j(r_i) = \sum_{i=1}^{n} \sqrt{1+(r_i/\epsilon)^2} - 1. \tag{6}$$

For any given residual r_i, we notice that

$$j(r) \approx \begin{cases} \frac{1}{2}(r/\epsilon)^2 & \text{for |r| small} \\ |r|/\epsilon & \text{for |r| large.} \end{cases}$$

Hence, we obtain a l^1 treatment of large residuals and a l^2 treatment of small residuals. Darche (1989) gives a practical value for ϵ that can be also used for the Huber threshold:

$$\epsilon = \frac{max|\mathbf{d}|}{100}.$$

Two different IRLS algorithms are suggested in the literature (Scales and Gersztenkorn, 1988). The first method consists of solving successive l^2 problems with a constant weight and taking the last solution obtained to recompute the weight for the next l^2 problem. The second method consists of recomputing the weights at each iteration (in fact, we need only a few iterations of the conjugate gradient scheme before calculating the new weight), solving small piecewise linear problems. The IRLS algorithms converge if each minimization reaches a minimum for a constant weight (Bube and Langan, 1997). This is the case for the first method described above, but it is also true for the second method since each starting point of the next iteration is the ending point of the previous solving piecewise linear problems. For practical reasons, we utilize the second method. Indeed, for most geophysical problems, the complete solution of the l^2 problem with a constant weight would lead to prohibitive costs. The general process in the program is as follows:

1. compute the current residual $\mathbf{r} = \mathbf{Hm} - \mathbf{d}$
2. compute the weighting operator $\mathbf{W}$ using $\mathbf{r}$
3. solve the weighted least-squares problem (equation 4) using a Conjugate Gradient algorithm
4. go to first step

We do not detail the Conjugate Gradient step here. For more details, the reader may refer to Paige and Saunders (1982). In our implementation, we control the restarting schedule of the iterative scheme with one parameter that governs the periodic computing of the weight. The Conjugate Gradient is reset for each new weighting function, meaning that the first iteration of each new least-squares problem (for each new weight) is a steepest descent step. In addition, the last solution $\mathbf{m}$ of the previous least-squares problem is used to compute the new residual and the new weighting matrix (steps 1 and 2 in the IRLS algorithm). In the following examples, except when indicated, we recompute the weight every five iterations. In the following sections, I call "IRLS" the IRLS algorithm described above with the weighting function in equation (5).

COMPARING IRLS AND HUBER ON A GEOPHYSICAL PROBLEM

We now compare IRLS and the Huber solver doing velocity-stack inversion. This method was first introduced by Thorson and Claerbout (1985) and completed with an IRLS algorithm by Nichols (1994). Guitton and Symes (1999) introduced the Huber misfit function with a specific non-linear solver as an alternative to IRLS. The velocity estimation is made using inverse methods to minimize the function

$$f(\mathbf{m}) = E(\mathbf{Hm} - \mathbf{d}),$$

where E is an error measure function, $\mathbf{H}$ is the modeling operator that transforms the model space $\mathbf{m}$ (velocity domain) into the data space $\mathbf{d}$ (CMP gathers). The adjoint operator ($\mathbf{H}^{\dagger}$) is the well-known hyperbolic Radon transform operator. The forward operation is

$$d(t,x) = \sum_{s=s_{min}}^{s_{max}} w_o m(\tau = \sqrt{t^2 - s^2 x^2}, s), \tag{7}$$

and the adjoint transformation becomes

$$m(\tau,s) = \sum_{x=x_{min}}^{x_{max}} w_o d(t = \sqrt{\tau^2 + s^2 x^2}, x), \tag{8}$$

where w_o is a weighting function (Claerbout and Black, 1997). We now solve the inverse problem using both the Huber solver and IRLS on synthetic and real data. Note that in the following sections, I assume that when $\epsilon = max|\mathbf{d}|/100$, we nearly solve the l^1 problem for both the Huber solver and IRLS.

Tests on synthetic data: spiky events

Figure 2 shows the input data for the first synthetic example plus the l^2 inversion result; it demonstrates the sensitivity of least-squares to outliers: some ellipses appear in the model space and the data space is infested with noise. Figures 3 and 4 show a comparison between IRLS and the Huber solver (same clip applied on the data). The damping parameter for IRLS and the Huber threshold were chosen as suggested above ($\epsilon = max|\mathbf{d}|/100$). Both results are very comparable since we get rid of the spikes in the data space (Figure 3). In addition, the five expected velocity events, associated with the five input hyperbolas, are clearly identifiable in the model space (Figure 4). Both results are very conclusive since the spikes have been correctly "l^1 handled".

Tests on synthetic data: groundroll

In this example, we contaminated the data with a low frequency/low velocity linear event that simulates, to some extent, groundroll (Figure 5, left). Because the operator $\mathbf{A}$ models only hyperbolas, the simulated groundroll introduces some inconsistent data. It is therefore

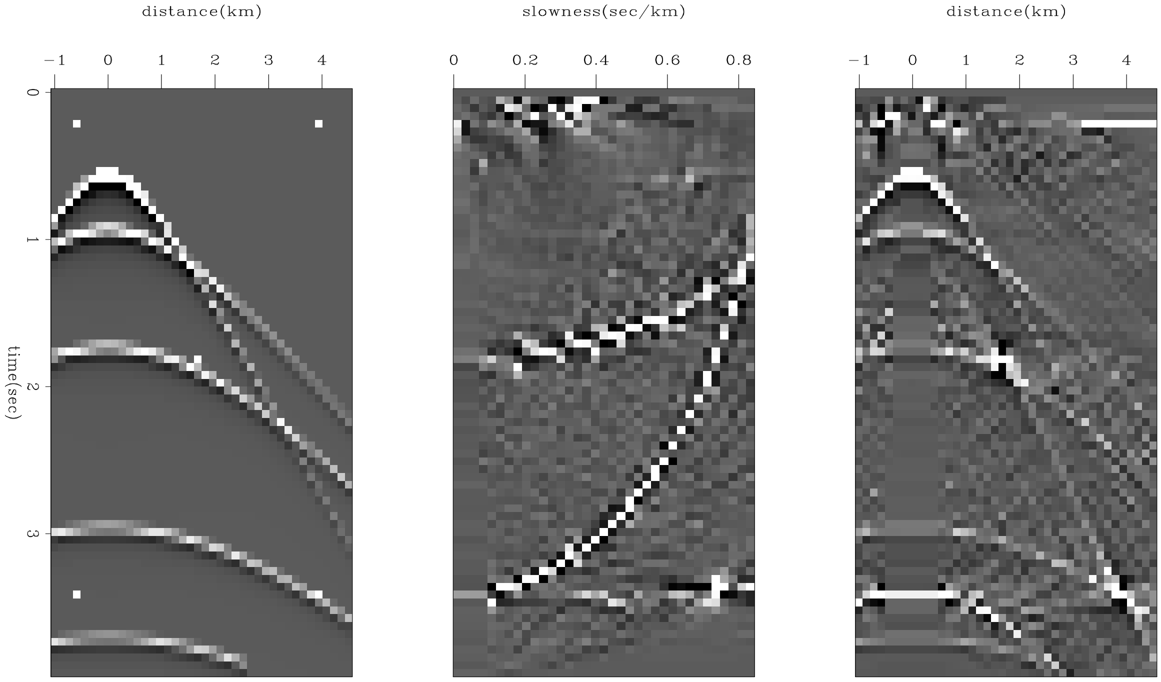

Figure 2: From left to right: 1) Input data. 2) velocity domain, l^2 inversion result. 3) remodeled data from the l^2 result antoine2-gr-spiky-L2-HUBER [ER]

Figure 3: From left to right: 1) Remodeled data with l^2. 2) Remodeled data with the Huber solver. 3) Remodeled data using IRLS antoine2-comp-data-spiky-0.04 [ER]

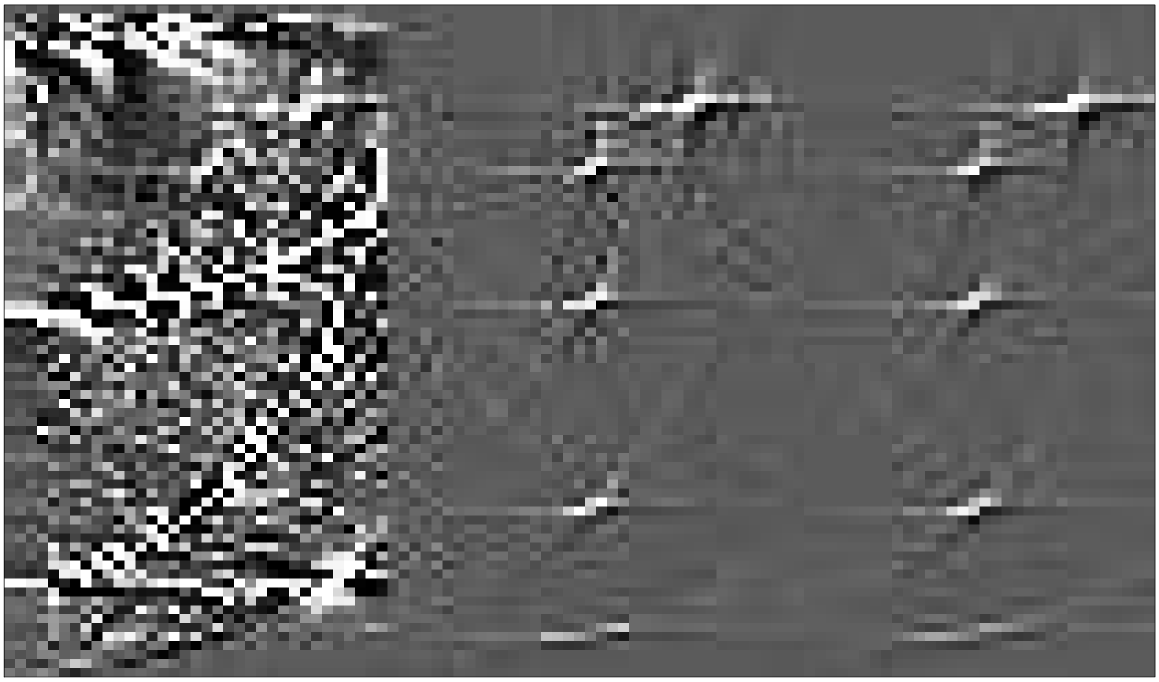

Figure 4: From left to right: 1) l^2 velocity space. 2) Huber velocity space. 3) IRLS velocity space antoine2-comp-mod-spiky-0.04 [ER]

interesting to test the sensitivity of our inversion schemes (l^2 and l^1) for this new problem. The l^2 inversion of the data leads to a fairly noisy model where the five expected events are difficult to pick (Figure 5). However, without groundroll and only a few artifacts, the remodeled l^2 data does not display so dramatic effects. With a "l^1" inversion, the model space is more accurately resolved, as shown in Figure 7. Again, there are few differences between IRLS and the Huber solver, and it looks like the "l^1" norm copes more easily with inconsistent events than l^2 (groundroll in this case).

Tests on real data

Figure 8 shows the input for the real data example. These data are contaminated with a low velocity/low frequency noise crossing the main signal. The Huber threshold and the damping paramater for IRLS are equal and obey the empirical choice that we gave above. Figure 8 details the least-squares result. The velocity panel (middle) shows some horizontal stripes that severely blur the main velocity fairway; in addition, because no strong event appears below $1s.$, a poor resolution is achieved. In regard to the l^2 result, we might note that the remodeled data (right panel of Figure 8) exhibit some artifacts in the top part that shows the limitations of the least-squares inversion in this case. Figure 10 displays the "l^1" inversion results with the Huber solver and IRLS. Again, because we are able to pick up a velocity fairway all the way down, both results are very comparable. Moreover, the remodeled data (Figure 9) show fewer artifacts than they do in the l^2 case.

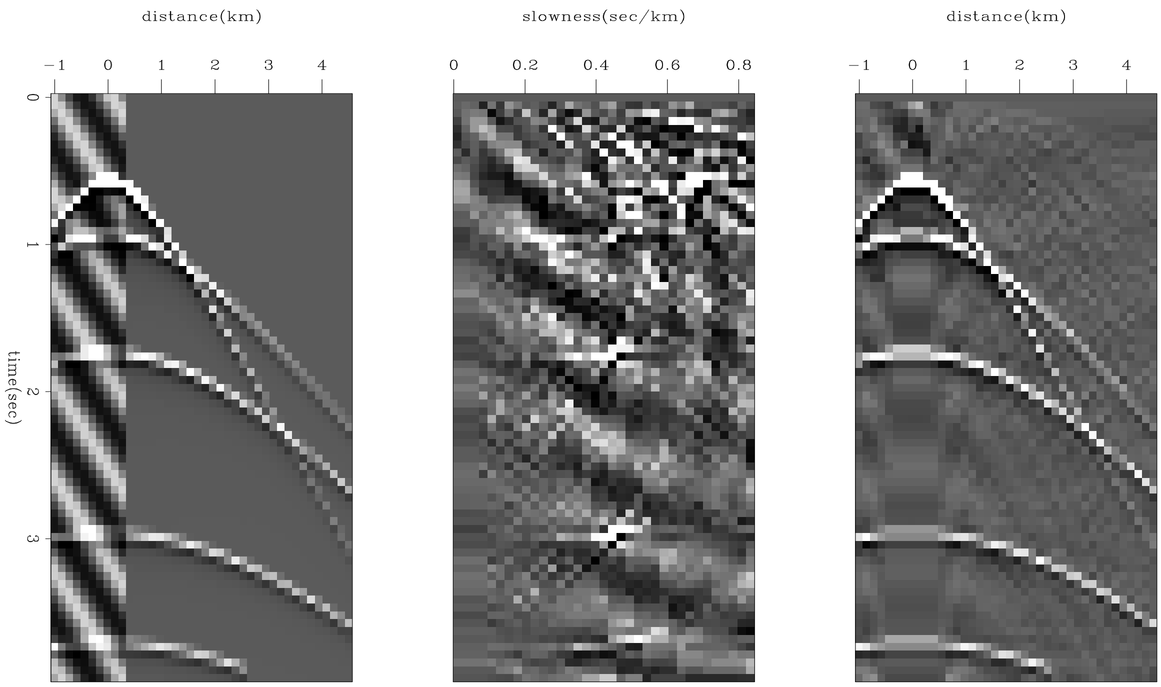

Figure 5: From left to right: 1) Input data. 2) velocity domain, l^2 inversion result. 3) remodeled data from the l^2 result antoine2-gr-freq30-L2-HUBER [ER]

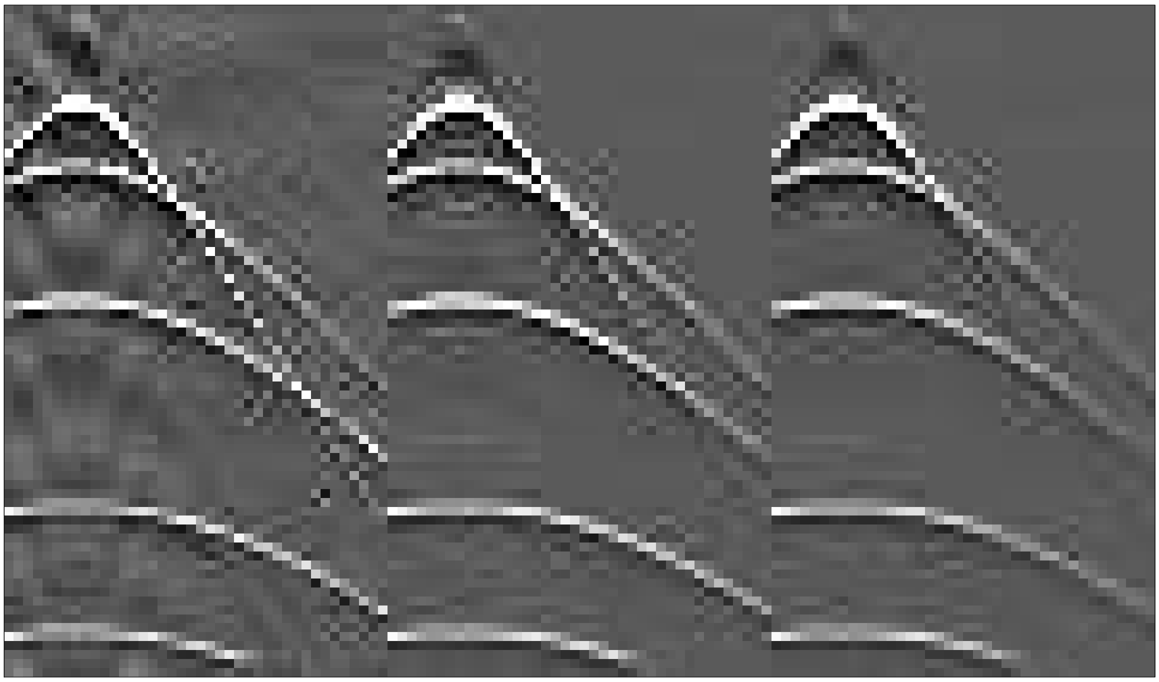

Figure 6: From left to right: 1) Remodeled data with l^2. 2) Remodeled data with the Huber solver. 3) Remodeled data using IRLS antoine2-comp-data-freq30 [ER]

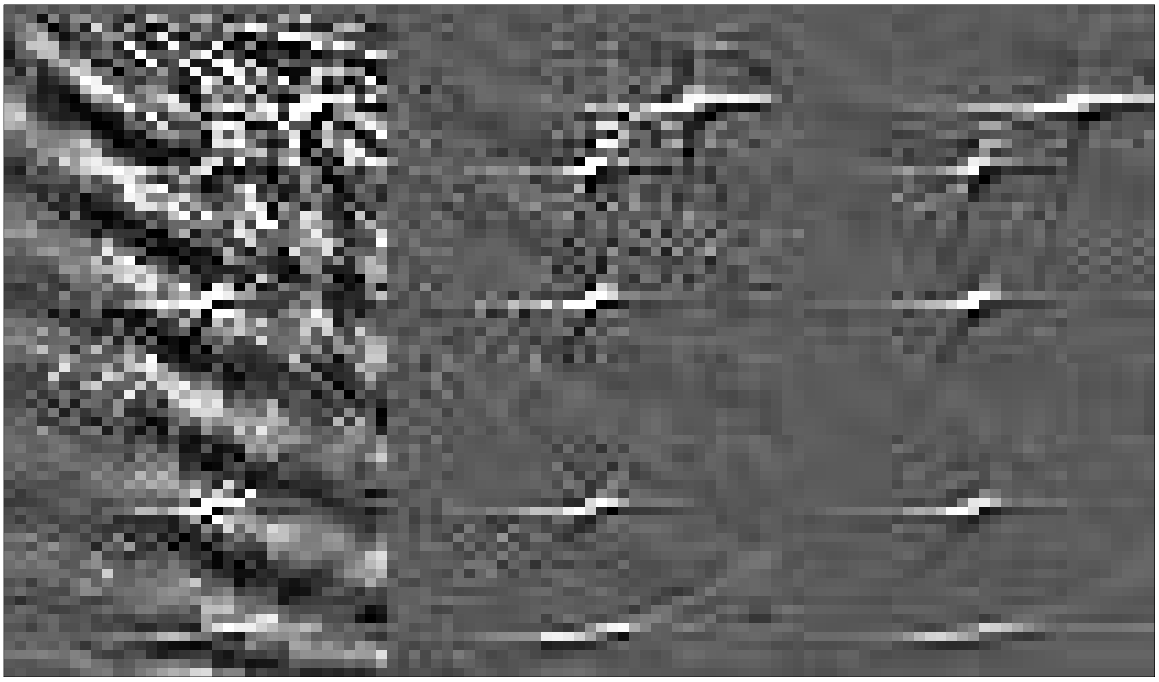

Figure 7: From left to right: 1) l^2 velocity space. 2) Huber velocity space. 3) IRLS velocity space antoine2-comp-mod-freq30 [ER]

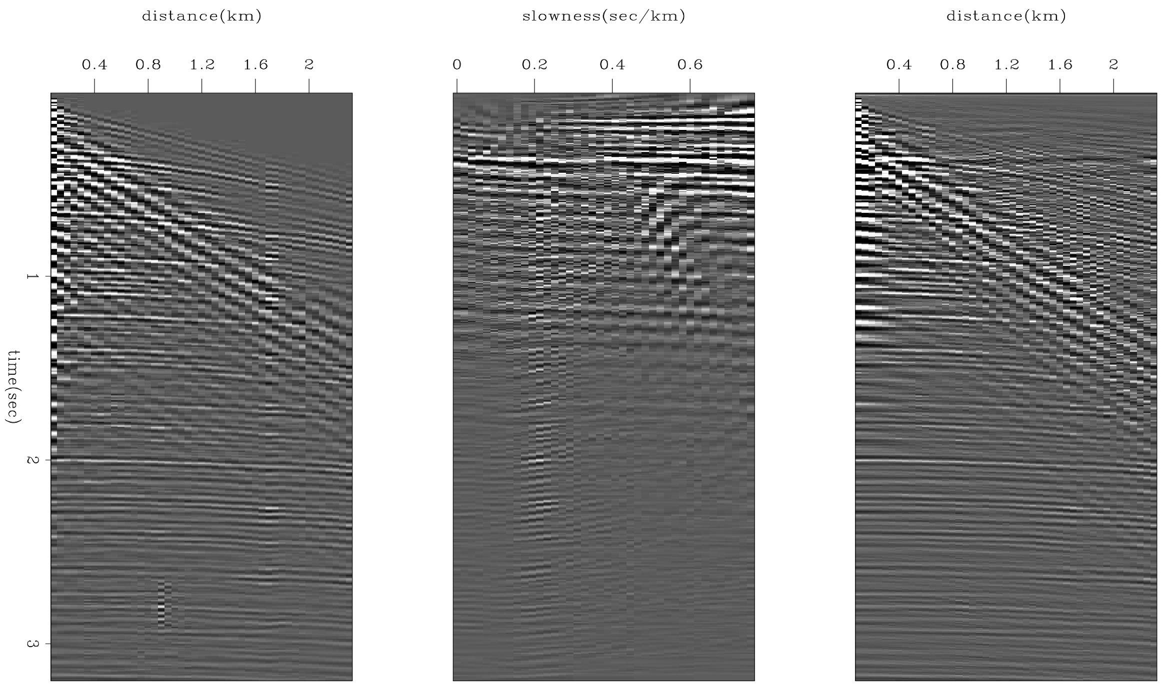

Figure 8: From left to right: 1) Input data. 2) velocity domain, l^2 inversion result. 3) remodeled data from the l^2 result antoine2-gr-wz08-L2-HUBER [ER]

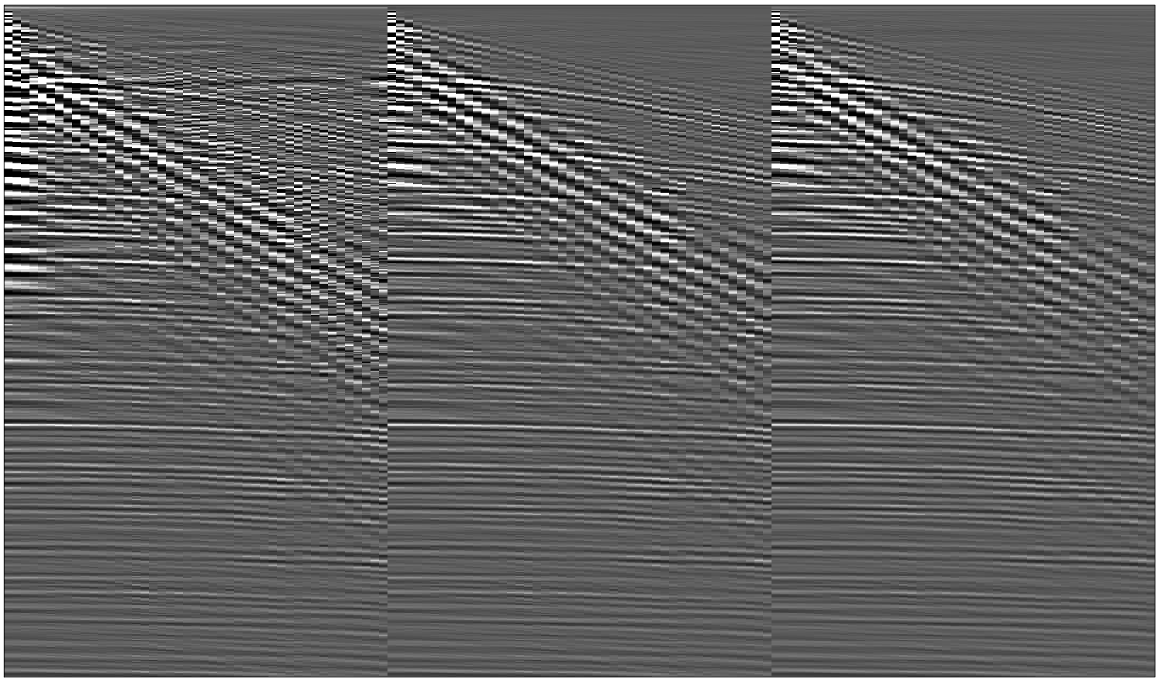

Figure 9: From left to right: 1) Remodeled data with l^2. 2) Remodeled data with the Huber solver. 3) Remodeled data using IRLS antoine2-comp-data-wz08-0.082 [ER]

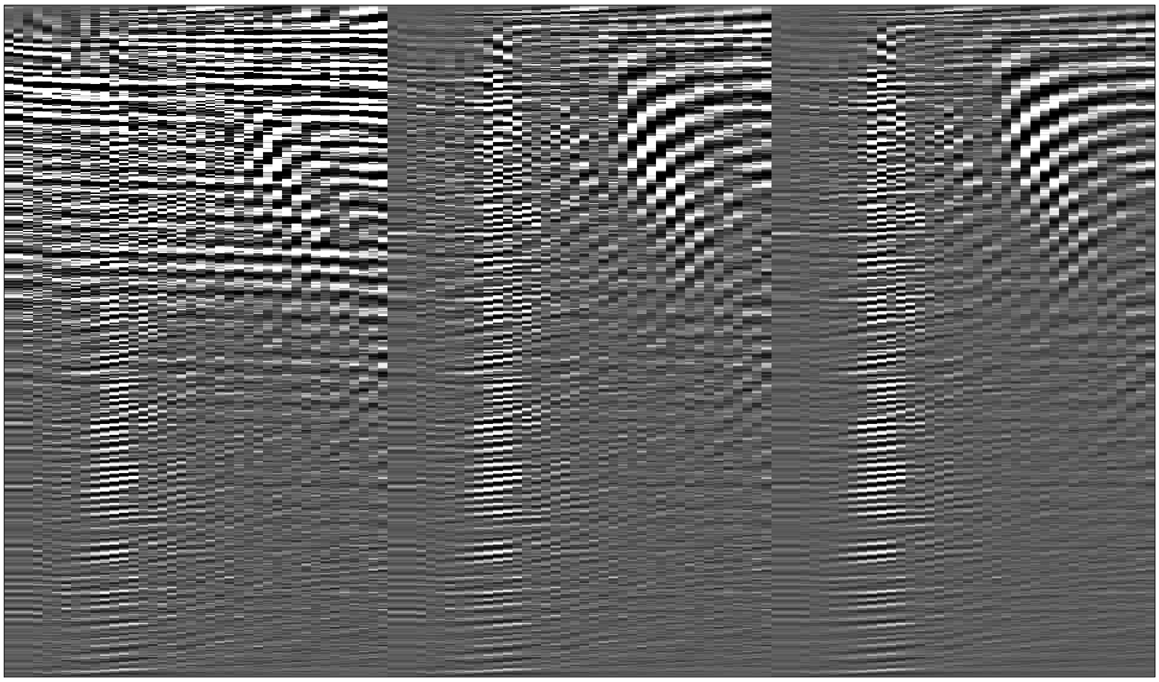

Figure 10: From left to right: 1) l^2 velocity space. 2) Huber velocity space. 3) IRLS velocity space antoine2-comp-mod-wz08-0.082 [ER]

Convergence issues

In this section I compare the convergence of the two solvers. Because the weighting matrix $\mathbf{W}$ is recomputed after a certain number of iterations, the IRLS algorithm consists of a series of different inverse problems. In this case, it may be unrealistic to compare the convergence of the Huber solver with IRLS. Nonetheless, because the final model should be nearly identical for both methods (assuming that they converge to same solution), the comparison appears to be fair. Note that for the IRLS algorithm, the residual is computed using equation 4. Figures 11, 12 and 13 show respectively the convergence of the two inversion schemes for the spiky data, the data with groundroll, and the real data. Except for the spiky data (Figure 11), the Huber solver converges better than the IRLS scheme. Note that small jumps in the residual of the IRLS algorithm usually appear when the weighting matrix $\mathbf{W}$ is recomputed. Figures 14 and 15 display the convergence of the inverse problem with the spiky data for different thresholds. A thorough comparison with Figure 11 (which corresponds to the "optimal" threshold given by Darche (1989)) proves that the convergence is rather sensitive to this parameter: a bigger threshold (Figure 15) leads to better convergence, whereas a smaller value decreases it. In both cases, the Huber solver converges significantly better. Last, Figure 16 displays the convergence of the IRLS algorithm for different restart paramaters: the weighting matrix is recomputed after every iteration (steepest descent), after every five and after every 15 iterations. Recomputing the weight after fifteen iterations increases the convergence, but only slightly. As expected, recomputing the weight after every iteration considerably slows down the convergence.

More about IRLS

All the convergence issues discussed above are dramatically altered if I change the restart parameter, the weighting function, and the damping factor. This means we have to make many decisions about a single inverse problem. To illustrate even more clearly the complexity of IRLS algorithms, I below give a list of weighting functions I found in the literature (ϵ positive constant):

$$w_{ii} = \frac{1}{[1+(r_{ii}/\epsilon)^2]^{1/4}},$$

(Bube and Langan, 1997),

$$w_{ii} = \frac{2\bar{r}}{|r_{ii}|+r_{ii}},$$

(Fomel and Claerbout, 1995),

$$w_{ii} = \begin{cases} |r_{ii}|^{(p-2)/2}, & |r_{ii}| \geq \epsilon \\ |\epsilon|^{(p-2)/2}, & |r_{ii}| < \epsilon \end{cases}$$

with p=1, (Huber, 1981),

$$w_{ii} = \frac{p^{1/2}}{(|r_{ii}|+\epsilon)^{(2-p)/2}},$$

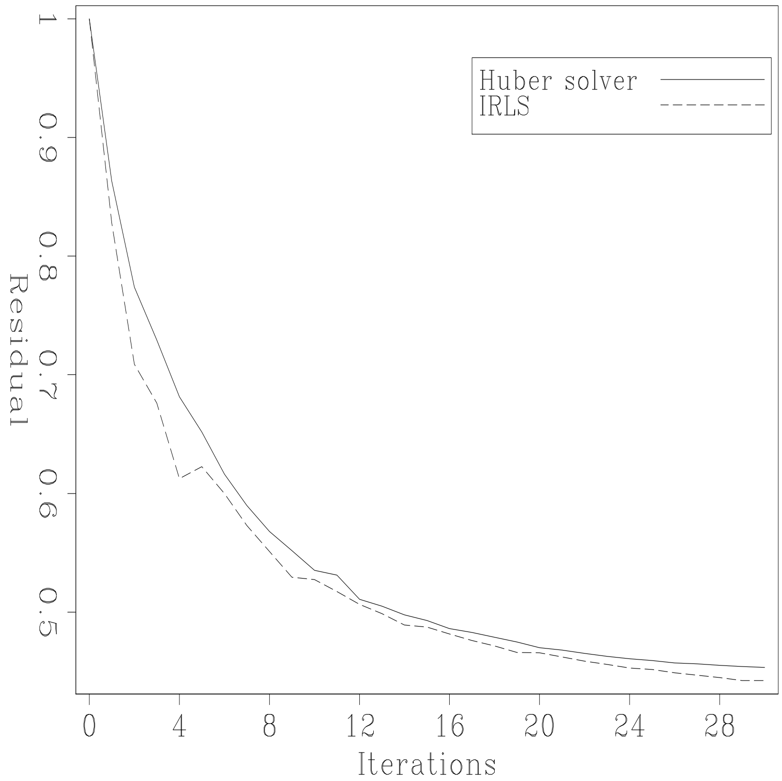

Figure 11: Relative residuals, spiky data antoine2-r-spiky-0.04 [ER]

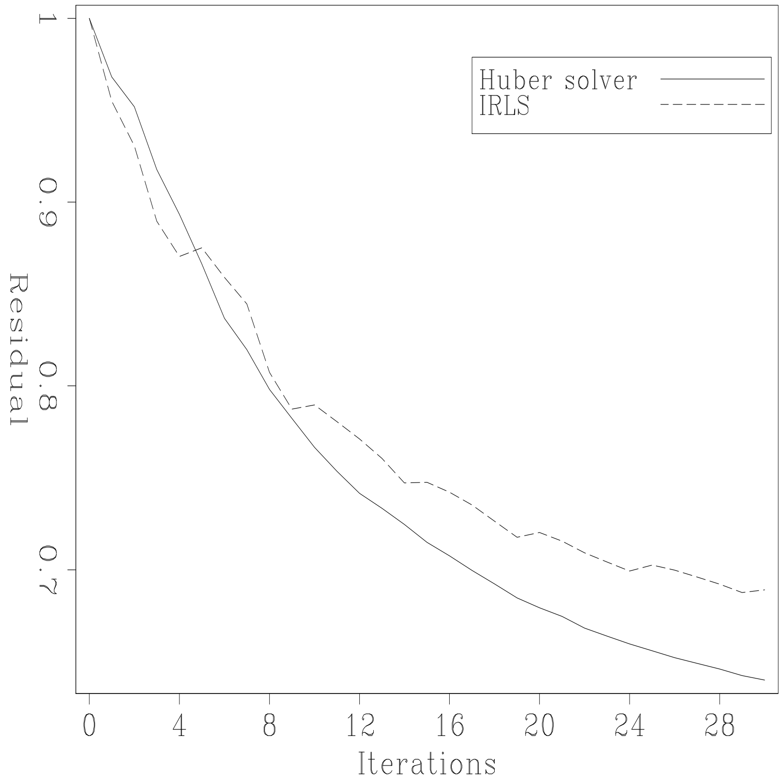

Figure 12: Relative residuals, groundroll data antoine2-r-freq30 [ER]

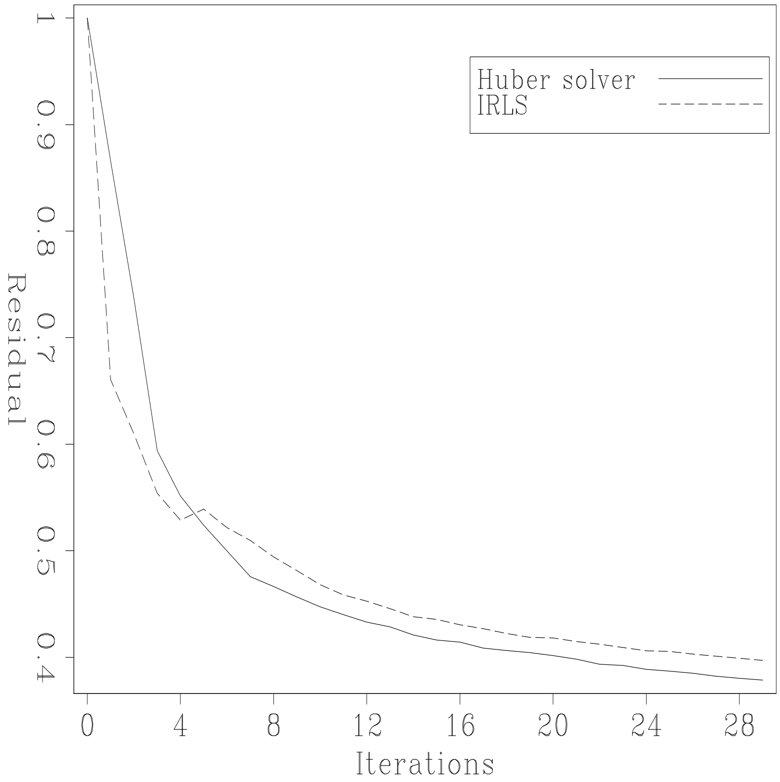

Figure 13: Relative residuals, real data antoine2-r-wz08-0.082 [ER]

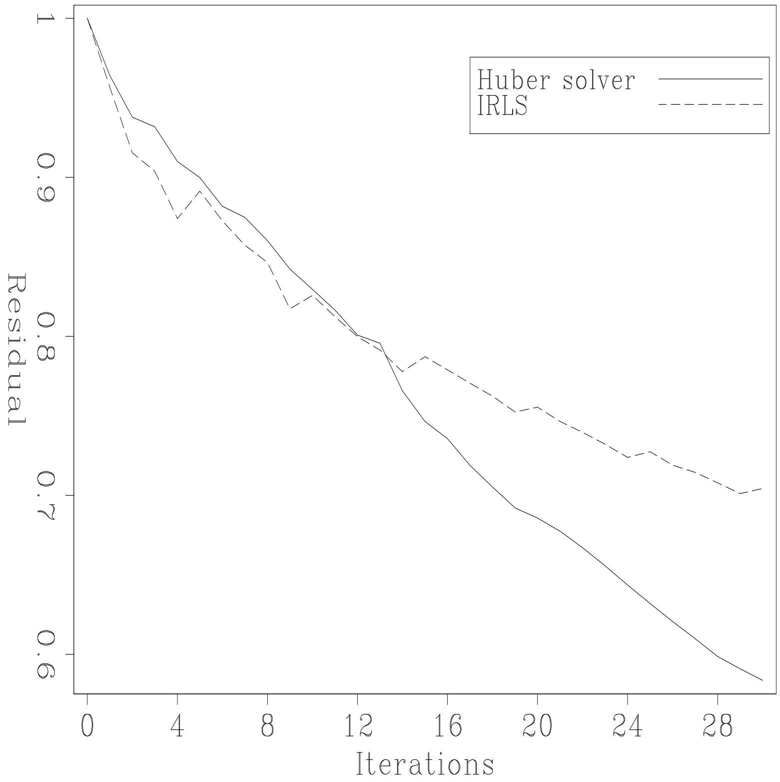

Figure 14: Relative residuals, spiky data: $\epsilon = 0.001$ antoine2-r-spiky-0.001 [ER]

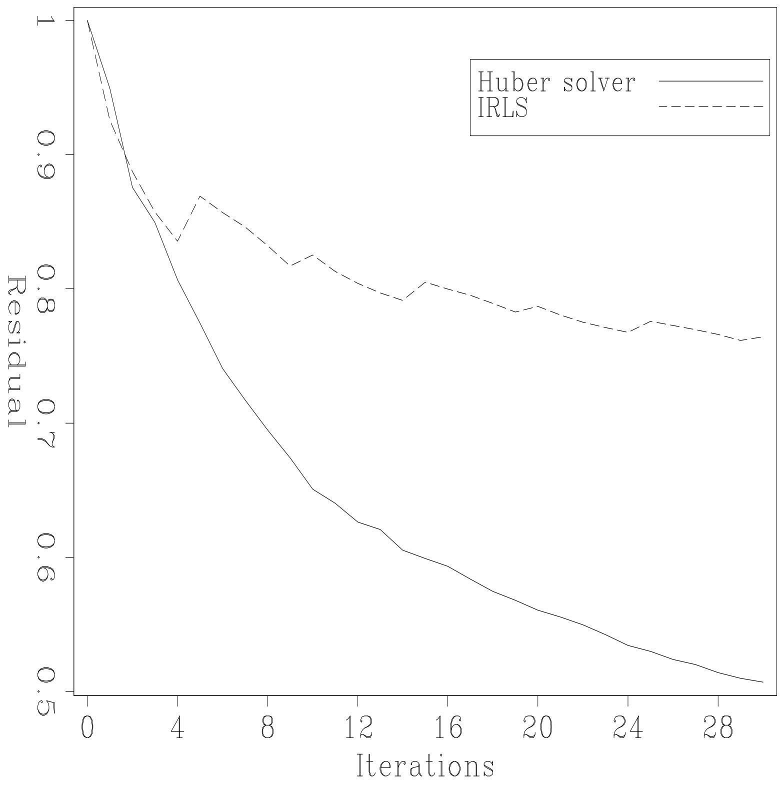

Figure 15: Relative residuals, spiky data: $\epsilon = 4$ antoine2-r-spiky-4 [ER]

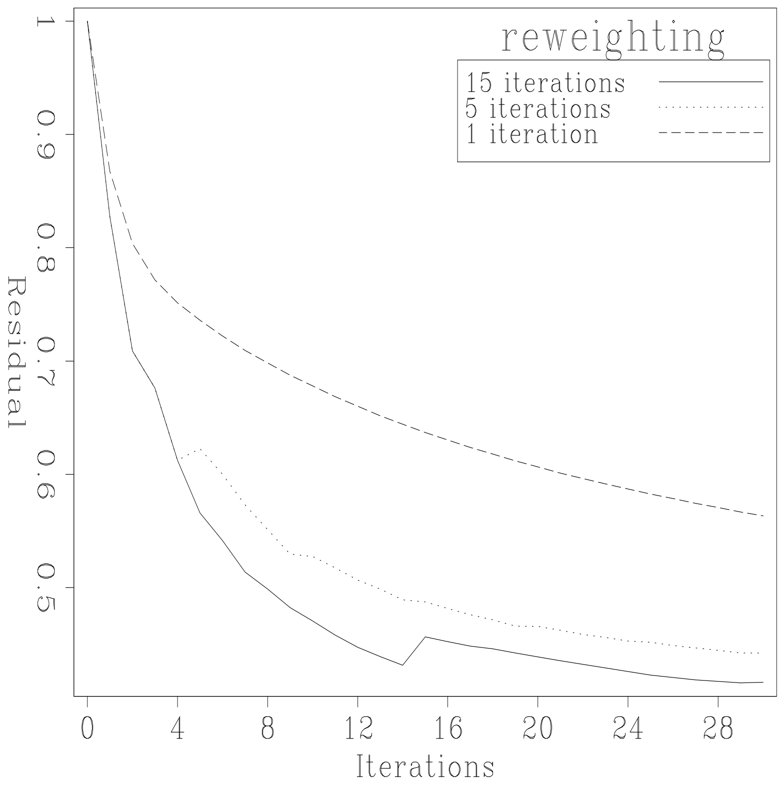

Figure 16: IRLS convergence for different restarts antoine2-cIRLSrt [ER]

with p=1, (Hugonnet, 1998), and finally

$$w_{ii} = \frac{1}{\sqrt{1+(r_{ii}/\epsilon)^2}}.$$

(Claerbout and Fomel, 1999), and there is probably more.

Each weighting function has different pros and cons and should be carefully chosen according to the problem we are trying to solve. Generally speaking, they all aim to weight down outliers in the data. With only the threshold to set up a priori, the Huber solver appears far easier to utilize than IRLS algorithms. Moreover, on a velocity stack inversion problem, for a very common weighting function (equation 5) with reasonable parameters (damping factor and restart parameter), I have shown that the Huber norm fosters better convergence than IRLS.

CONCLUSION

With thoroughly chosen parameters, I have shown that the l^1 problem can be approached using either the Huber along with an appropriate non-linear solver or with IRLS algorithms along with an appropriate weighting function. For the particular problem of velocity stack inversion, however, the Huber solver generally bears better convergence properties. From the practical point of view, the IRLS algorithms pose cumbersome problems for the choice of (1) the weighting function (2) the restart parameter and (3) the damping factor. With only one parameter to set up a priori, the Huber solver appears far easier to operate. This ease of use should encourage more people to use the Huber norm when implementing the hybrid l^2-l^1 norm.

REFERENCES

Bube, K. P., and Langan, R. T., 1997, Hybrid lambda1/lambda2 minimization with applications to tomography: Geophysics, **62**, no. 04, 1183–1195.

Claerbout, J. F., and Black, J. L., 1997, Basic earth imaging: Class notes, http://sepwww.stanford.edu/sep/prof/index.html.

Claerbout, J. F., and Fomel, S., 1999, Geohpysical estimation with example: Class notes, http://sepwww.stanford.edu/sep/prof/index.html.

Darche, G., 1989, Iterative l1 deconvolution: SEP-Report, **61**, 281–301.

Fomel, S., and Claerbout, J., 1995, Searching the sea of Galilee: SEP–**84**, 259–270.

Guitton, A., and Symes, W. W., 1999, Robust and stable velocity analysis using the Huber function: 69th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1166–1169.

Guitton, A., 2000, Implementation of a nonlinear solver for minimizing the Huber norm: SEP–**103**, 281–289.

Huber, P. J., 1973, Robust regression: Asymptotics, conjectures, and Monte Carlo: Ann. Statist., **1**, 799–821.

Huber, P. J., 1981, Robust statistics: Wiley series in Probability and Mathematical statistics.

Hugonnet, P., 1998, La transformee de radon generalisee et ses applications a la sismique: Ph.D. thesis, Ecole doctorale des sciences exactes et de leurs applications, Universite de Pau et des pays de l'Adour.

Kelley, C. T., 1999, Iterative methods for optimization: SIAM in applied mathematics.

Liu, D. C., and Nocedal, J., 1989, On the limited memory bfgs method for large-scale optimization: Mathematical Programming, **45**, 503–528.

Lumley, D., Nichols, D., and Rekdal, T., 1994, Amplitude-preserved multiple suppression: SEP–**82**, 17–24.

More, J. J., and Thuente, J., 1994, Line search algorithms with guaranteed sufficient decrease: ACM Transactions on Mathematical Software, **20**, 286–307.

Nichols, D., 1994, Velocity-stack inversion using lp norms: SEP–**82**, 1–16.

Nocedal, J., 1980, Updating quasi-Newton matrices with limited storage: Mathematics of Computation, **95**, 339–353.

Paige, C. C., and Saunders, M. A., 1982, An algorithm for sparse linear equations and sparse least squares: ACM Transactions on Mathematical Software, **61**, 43–71.

Scales, J. A., and Gersztenkorn, A., 1988, Robust methods in inverse theory: Inverse Problems, **4**, 1071–1091.

Thorson, J. R., and Claerbout, J. F., 1985, Velocity-stack and slant-stack stochastic inversion: Geophysics, **50**, 2727–2741.