Stanford Exploration Project, Report 117, October 23, 2004, pages 51–63

Stanford Exploration Project, Report 117, October 23, 2004, pages 51–63

Bound constrained optimization: Application to the dip estimation problem

Antoine Guitton[1]

ABSTRACT

A bound constrained optimization algorithm called L-BFGS-B is presented. It combines a trust region method with a quasi-Newton update of the Hessian and a line-search. This algorithm is tested on the non-linear dip estimation problem. Results show that the optimization algorithm converges effectively toward a model with bounds. Furthermore, bounds improve the estimated dips where plane-waves with different slopes overlap (e.g., with aliased data). When no constraints are applied, the algorithm is of comparable speed to a conjugate gradient solver.

INTRODUCTION

Most geophysical inverse problems require, explicitly or implicitly, that the final model is within a range of acceptable values. For instance, densities should always be positives, P-waves velocity should be greater than S-waves velocity, etc. These bounds, or constraints, on the final model are called simple bounds. Many different techniques have been developed by the optimization community to solve inverse problems with simple bounds. What makes these methods attractive is their cost and ease of use.

Among the variety of bound constrained methods, the L-BFGS-B technique (Zhu et al., 1997) is very appealing. The L-BFGS-B method is an extension of the well-known, quasi-Newton, limited-memory BFGS technique (Broyden, 1969; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970; Nocedal, 1980; Liu and Nocedal, 1989) that can impose simple bounds on the model parameters. L-BFGS-B (Limited-memory BFGS with Bounds) incorporates a trust region method with update of the Hessian (or second derivative) and a line search. There are three main reasons why L-BFGS-B is chosen for this task. First, the L-BFGS method has been already successfully applied on different geophysical optimization problems such as minimization of the Huber norm (Guitton and Symes, 2003) or multiple attenuation with sparse radon transforms (Sava and Guitton, 2003). Second, the memory requirements and the computation costs for L-BFGS-B are limited. Finally, the user interface is very simple, making its implementation within existing code very simple.

In this paper, the L-BFGS-B method is presented with a few algorithmic details. These details include the gradient projection technique and the trust region method. Both elements

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

are blended inside the L-BFGS-B algorithm to insure that the estimated models are within a desired range of values. As an illustration, L-BFGS-B is tested on the dip estimation problem as formulated by Fomel (2002). L-BFGS-B is an improvement over the conventional conjugate gradient approach because it allows the incorporation of simple bound constraints on the dips. Several examples illustrate the dip estimation results.

OPTIMIZATION

In this section, the basic ideas behind bound constrained optimization algorithms are presented. Then the L-BFGS-B method is succinctly described.

The problem

The goal of the proposed algorithm is to find a vector of model parameters $\mathbf{x}$ such that we minimize (Kelley, 1999)

$$\min f(\mathbf{x}) \text{ subject to } \mathbf{x} \in \Omega, \tag{1}$$

where

$$\mathbf{x} \in \Omega = \{\mathbf{x} \in \Re^N \mid l_i \leq x_i \leq u_i\}, \tag{2}$$

with l_i and u_i being the lower and upper bounds for the model x_i, respectively. In this case, l_i and u_i are called simple bounds. They can be different for each point of the model space. The model vector that obeys equation (1) is called $\mathbf{x}^*$.

The sets of indices i for which the ith constraint are active/inactive are called the active/inactive sets $A(x)/I(x)$. Most of the algorithms used to solve bound constrained problems first identify $A(x)$ and then solve the minimization problem for the free variables of $I(x)$.

Gradient Projection Algorithm

Identifying $A(x)$ is a minimization problem in itself. An effective way to find $A(x)$ is by using the gradient projection method (Kelley, 1999). First, define P as the projection onto Ω of $\mathbf{x}$ such that for each x_i we have

$$P(x_i) = \begin{cases} l_i & if \quad x_i \leq l_i, \\ x_i & if \quad l_i \leq x_i \leq u_i, \\ u_i & if \quad x_i \geq u_i. \end{cases} \tag{3}$$

From this definition, one can modify the classical steepest descent algorithm by projecting it onto the feasible region as follows:

$$\mathbf{x_{k+1}} = P(\mathbf{x_k} - \lambda \mathbf{g_k}), \tag{4}$$

where k is the iteration number, $\mathbf{g_k}$ is the gradient of $f(\mathbf{x})$ at iteration k and λ is the step-length given by a line search scheme. Given the current position $\mathbf{x_k}$, finding the local minimizer is relatively straightforward. The important property of the projection algorithm is that the active set after many iterations is the same as the active set of the solution vector $\mathbf{x}^*$. Then, the exact solution of the projected gradient is not needed and only an approximate one is. Then, the inactive set can be optimized for the unconstrained variables, the bounded variables being held fixed. The unconstrained problem can be solved by any method for unconstrained optimization. The method in this paper is based on a trust region method that incorporates two important variations: first, line searches are used; second, the Hessian of the objective function (or second derivative) is approximated with BFGS matrices.

Trust region methods

Trust region methods are widely popular for solving problems where the Hessian has regions of negative curvatures (Kelley, 1999). The basic idea behind these methods is to fit the objective function locally with a quadratic model near a point $\mathbf{x_k}$:

$$m(\mathbf{x}) = f(\mathbf{x_k}) + \mathbf{g_k^T}(\mathbf{x}-\mathbf{x_k}) + (\mathbf{x}-\mathbf{x_k})^\mathbf{T}\mathbf{B_k}(\mathbf{x}-\mathbf{x_k})/2, \quad (5)$$

where $()^\mathbf{T}$ is the transpose and $\mathbf{B_k}$ is the Hessian of $f(\mathbf{x})$ at iteration k. The goal is to find an $\mathbf{x}$ that minimizes $m(\mathbf{x})$ such that

$$\|\mathbf{x}-\mathbf{x_k}\| \leq \Delta, \quad (6)$$

where Δ is the trust region radius. Once a local minimizer is found, either the step is accepted, or the radius Δ is changed, or both. Termination criterion end the process. In the most simple case, an update of $\mathbf{x_k}$ is obtained by minimizing the projected steepest descent onto the quadratic area

$$\Phi(\lambda) = m(\mathbf{x_k} - \lambda\mathbf{g_k}), \quad (7)$$

with

$$\|\mathbf{x_k} - \lambda\mathbf{g_k}\| \leq \Delta.$$

This update is also called the Cauchy point. This technique resembles quite closely the projection gradient algorithm to find the active set $A(\mathbf{x})$. It is then not surprising that many algorithms for bound constrained optimization are either completely or partially based on trust region methods.

The algorithm used in this paper does not involve any trust region radii, but line search algorithms instead. Yet, this algorithm keeps the concept of fitting the objective function locally with a quadratic form in equation (5) to find the minimum of $f(\mathbf{x})$ in equation (1).

The L-BFGS-B algorithm

The L-BFGS-B algorithm is an extension of the L-BFGS algorithm to handle simple bounds on the model (Zhu et al., 1997). The L-BFGS algorithm is a very efficient algorithm for solving large scale problems. L-BFGS-B borrows ideas from the trust region methods while keeping the L-BFGS update of the Hessian and line search algorithms. Methods based completely on the trust region techniques exist and are freely available. Among them, the program SBMIN from the LANCELOT package[2] (written in Fortran 77) is very popular (Conn et al., 1992). The original L-BFGS-B is freely available in Fortran 77 from the Northwestern University webpage[3].

The L-BFGS-B works as follows for one iteration:

1. Find an approximation of the Cauchy point for
$$\Phi(\lambda) = m(\mathbf{x}(\lambda)) = m(P(\mathbf{x_k} - \lambda \mathbf{g_k})), \tag{8}$$
with $m(\mathbf{x})$ being the quadratic form in equation (5). Identify $A(\mathbf{x})$ and $I(\mathbf{x})$.
2. Minimize the quadratic form in equation (5) for the unconstrained variables. This step gives a search direction.
3. Perform a line search along the new search direction to find the minimum of $f(\mathbf{x})$.
4. Update the Hessian with the L-BFGS method (Nocedal, 1980) and check if convergence is obtained.

In a "pure" trust region method, step (3) is replaced by tests assessing the success of the update by measuring the difference between the quadratic form and the true objective function at the update. In addition, the radius Δ is also examined.

The L-BFGS-B algorithm is affordable for very large problems. The memory requirement is roughly $(12+2m)N$ where m is the number of BFGS updates kept in memory and N the size of the model space. In practice, $m = 5$ is a typical choice. Per iteration, the number of multiplications range from $4mN + N$ when no constraints are applied to m^2N when all variables are bounded. The program offers the freedom to have different bounds for different points of the model space. In addition, some points can be constrained while others are not.

There are three different stopping criteria for the L-BFGS-B algorithm. First the program stops when the maximum number of iterations is reached. Or, the program stops when the decrease of the objective function becomes small enough. Or, the program stops when the norm of the projected gradient (in a ℓ^∞ sense) is small enough.

Now for the bells and whistles, tests indicate that the L-BGFS-B algorithm ran in single precision with no constraints is not quite twice as slow as a conjugate gradient solver per

[2]http://www.cse.clrc.ac.uk/nag/lancelot/lancelot.shtml
[3]http://www.ece.northwestern.edu/ nocedal/lbfgsb.html

iteration. This result is quite remarkable when considering that L-BFGS-B works for any type of non-linear (or linear) problem with line searches. In addition, the number of iterations needed to convergence is almost identical for both L-BFGS-B and the conjugate gradient solver. In the next section, the L-BFGS-B algorithm is utilized to estimate local dips from seismic data.

APPLICATION TO THE DIP ESTIMATION PROBLEM

The goal of dip estimation is to find a local stepout, σ, that destroys the local plane wave such that,

$$0 \approx \frac{\partial u}{\partial h} + \sigma \frac{\partial u}{\partial \tau}, \tag{9}$$

where u is the wavefield at time τ and offset h. For all gathers, we evaluate the slope σ with a method based on high-order plane-wave destructor filters (Fomel, 2002). With the Z transform notation, Fomel (2002) shows that there is a 2-D filter

$$C_n(Z_t, Z_h) = B_n(Z_t^{-1}) - Z_h B_n(Z_t), \tag{10}$$

with

$$B_n(Z_t) = \sum_{k=-n/2}^{n/2} a_k(\sigma^{n-1}) Z_t^k, \tag{11}$$

that annihilates the local plane wave. The number of coefficients for the filter B_n is n. The filter coefficients $a_k(\sigma^{n-1})$ are functions of σ^{n-1} as detailed in equations (9) and (10) of Fomel (2002). For instance, if $n = 3$, we have

$$\begin{aligned} a_{-1}(\sigma^2) &= \frac{(1-\sigma)(2-\sigma)}{12}, \\ a_0(\sigma^2) &= \frac{(2+\sigma)(2-\sigma)}{6}, \\ a_1(\sigma^2) &= \frac{(1+\sigma)(2+\sigma)}{12}. \end{aligned} \tag{12}$$

In equation (10), σ is unknown and is estimated with a non-linear solver.

Inversion

Fomel (2002) shows that a possible solution to the slope estimation problem is obtained by minimizing the non-linear function

$$f(\sigma) = \|\mathbf{C}(\sigma)\mathbf{d}\|^2, \tag{13}$$

where $\mathbf{C}(\sigma)$ is the operator convolving the data with the 2-D filter $C_n(Z_t, Z_h)$ and $\mathbf{d}$ is the known data. Fomel (2002) proposes iterating with a Gauss-Newton algorithm:

$$\sigma_{k+1} = \sigma_k - (\mathbf{C}(\sigma_k)'\mathbf{d}\mathbf{C}(\sigma_k)'\mathbf{d})^{-1}\mathbf{C}(\sigma_k)'\mathbf{d}\mathbf{C}(\sigma_k)\mathbf{d}, \tag{14}$$

where the step is estimated with a conjugate gradient method. One problem with this approach is that it converges well only if we are close enough to the true solution. Another problem stems from the fact that no line search in the gradient direction is implemented. Therefore, this method might oscillate close to the solution.

To make the dip estimation more robust and also to incorporate the possibility to bound-constrain the dips, the L-BFGS-B method is used instead of the Gauss-Newton approach. For the L-BFGS-B method, similar to BFGS, we need to estimate the function and its gradient. Incorporating a regularization operator $\mathbf{R}$ that penalizes differences between adjacent dip values, the objective function becomes

$$f(\sigma) = \frac{1}{2}\left(\|\mathbf{C}(\sigma)\mathbf{d}\|^2 + \epsilon^2\|\mathbf{R}\sigma\|^2\right) \tag{15}$$

and the gradient is

$$\mathbf{g}(\sigma) = \mathbf{C}(\sigma)'\mathbf{d}\mathbf{C}(\sigma)\mathbf{d} + \epsilon^2\mathbf{R}'\mathbf{R}\sigma. \tag{16}$$

The dips can be then estimated reliably. In the next section, the L-BFGS-B method is illustrated on different examples. These examples illustrate that the bound constrained optimization improves the dip values when events with different local slopes overlap.

EXAMPLES

This section presents several examples of the dip estimation problem with bound constraints. All of these examples were computed with double precision arithmetic. For most of these examples the program stopped because the objective function was not decreasing enough, thus indicating that a possible minimum was found.

Figures 1 to 6 show in (a) the input data, in (b) the estimated dips with L-BFGS-B and no constraints, in (c) the estimated positive dips and in (d), the estimated negative dips. The same clip is applied for all the color plots within each Figure. Warm colors represent positive dips whereas cold colors represent negative dips. Figures 1 and 2 illustrate that the dip estimation program with bounds work as expected, but they do not represent real challenges where, for example, multiple events with different slopes overlap.

Figure 3 shows how the bounds can improve the local dip values. In Figure 3a, when no bounds are used, a clear cut difference between positive and negative dips is visible. Applying bounds in Figures 3b and 3c, the dip estimation program is able to locate the dips of interest when aliasing is present.

On a CMP gather in Figure 4a, two lines cross at a location where the dips should be positive. If no bounds are applied, Figure 4b shows that negative dips are found instead.

Applying bounds in Figure 4c, strong positive dips are now recovered. We are able to estimate positive dips beyond aliasing, thus improving on the existing program. Similar conclusions can be made on Figure 5.

It might happen that we cannot separate dips as easily as we could in Figures 3, 4 and 5. For example, Figure 6 displays some earthquake data from Professor Peter Shearer for which positive and negative dips do not clearly separate (see, for instance, where the two black lines cross). The problem here stems from the fact that the event with negative dips (in blue) is much stronger than the overlapping event with positive dips at this location.

Finally, Figure 7 shows a dip decomposition of the data in Figure 5a. This illustrates the ability to select a small range of dips that goes beyond the simple positive/negative constraints of the preceding examples.

CONCLUSION

L-BFGS-B is an algorithm that solves non-linear problems by imposing some constraints on the model. This program incorporates concepts from trust region methods plus BFGS matrices and line searches. Per iteration, this program requires roughly $O(N)$ computations, N being the size of the model space. Used with single precision arithmetic and no bounds, this program is not quite twice as slow as conjugate gradient. Therefore, L-BFGS-B can be used for solving many non-linear geophysical problems.

As an illustration, the bound constrained optimization code was employed to estimate local dips from seismic data. These examples show that the bounds were effectively working and that this method was converging toward acceptable solutions. In addition, this technique clearly improves on the existing method without bounds by discarding non-physical dip values at locations where aliasing is present. Flattening (Lomask, 2003) could greatly benefit from this improvement.

In geophysics, the number of applications for this type of solver could be quite large. An obvious choice is velocity estimation. For instance, Dix inversion might benefit from the possibility of constraining the estimated interval velocities to a reasonable range.

ACKNOWLEDGMENTS

I would like to thank Jesse Lomask for his suggestions on this paper.

REFERENCES

Broyden, C. G., 1969, A new double-rank minimization algorithm: AMS Notices, **16**, 670.

Conn, A. R., Gould, N. I. M., and Toint, P. L., 1992, LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A): Springer Series in Computational Mathematics 17.

Fletcher, R., 1970, A new approach to variable metric methods: Comput. J., **13**, 317–322.

Fomel, S., 2002, Applications of plane-wave destruction filters: Geophysics, **67**, no. 06, 1946–1960.

Goldfarb, D., 1970, A family of variable metric methods derived by variational means: Math. Comp., **24**, 23–26.

Guitton, A., and Symes, W., 2003, Robust inversion of seismic data using the Huber norm: Geophysics, **68**, no. 4, 1310–1319.

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.

Lomask, J., 2003, Flattening 3D seismic cubes without picking: Soc. of Expl. Geophys., 73rd Ann. Internat. Mtg., 1402–1405.

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

Sava, P., and Guitton, A., 2003, Multiple attenuation in the image space: Soc. of Expl. Geophys., 73rd Ann. Internat. Mtg., 1933–1936.

Shanno, D. F., 1970, Conditioning of quasi-Newton methods for function minimization: Math. Comp., **24**, 647–657.

Zhu, H., Byrd, R. H., and Nocedal, J., 1997, Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization: ACM Transactions on Mathematical Software, **23**, 550–560.

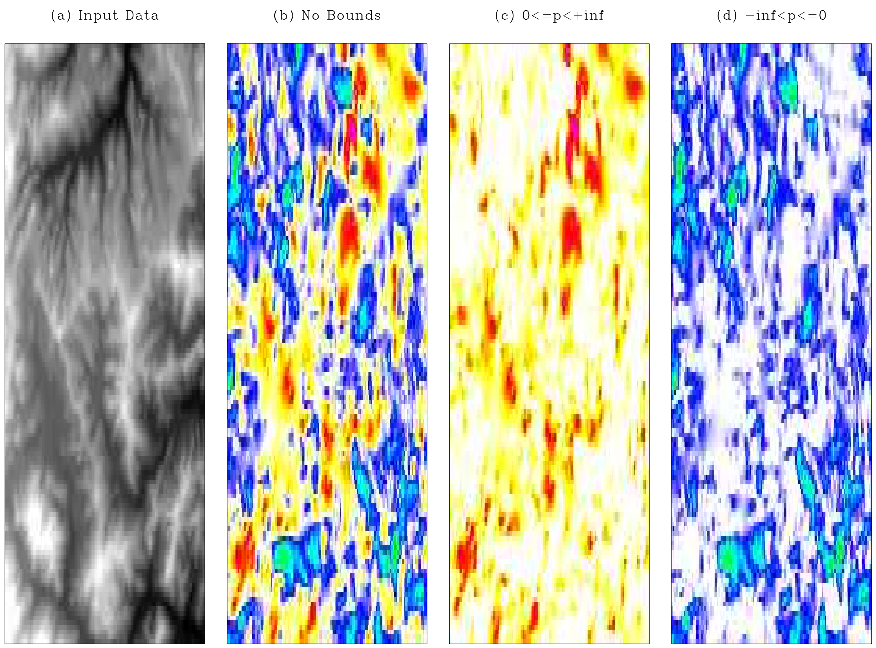

Figure 1: (a) Input data. (b) Estimated dips with L-BFGS-B without bound constraints. (c) Estimated positive dips only. (d) Estimated negative dips only. antoine1-fabricposneg [ER]

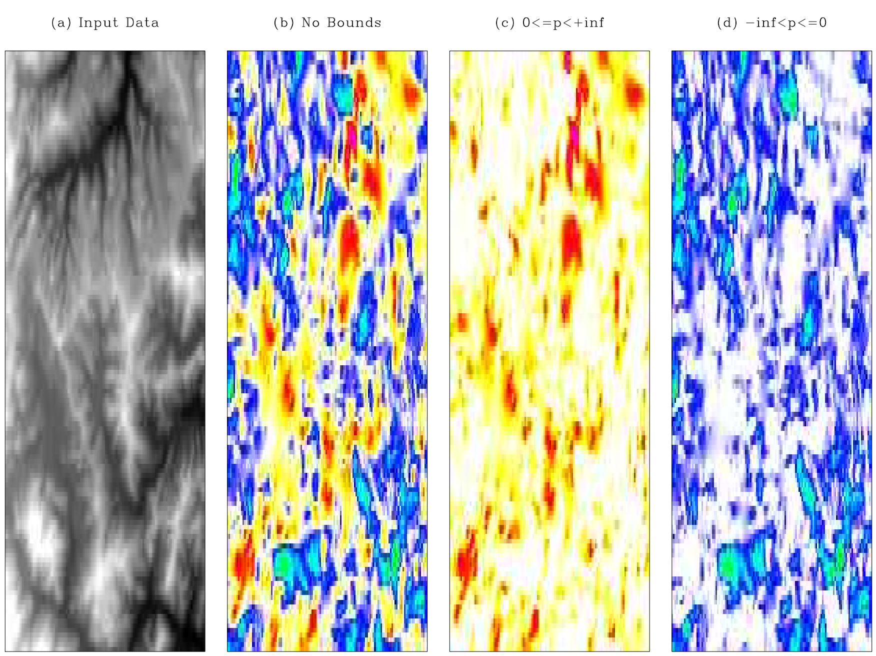

Figure 2: (a) Input data. This topographic map comes from the bay area. (b) Estimated dips with L-BFGS-B without bound constraints. (c) Estimated positive dips only. (d) Estimated negative dips only. antoine1-sfbayposneg [ER]

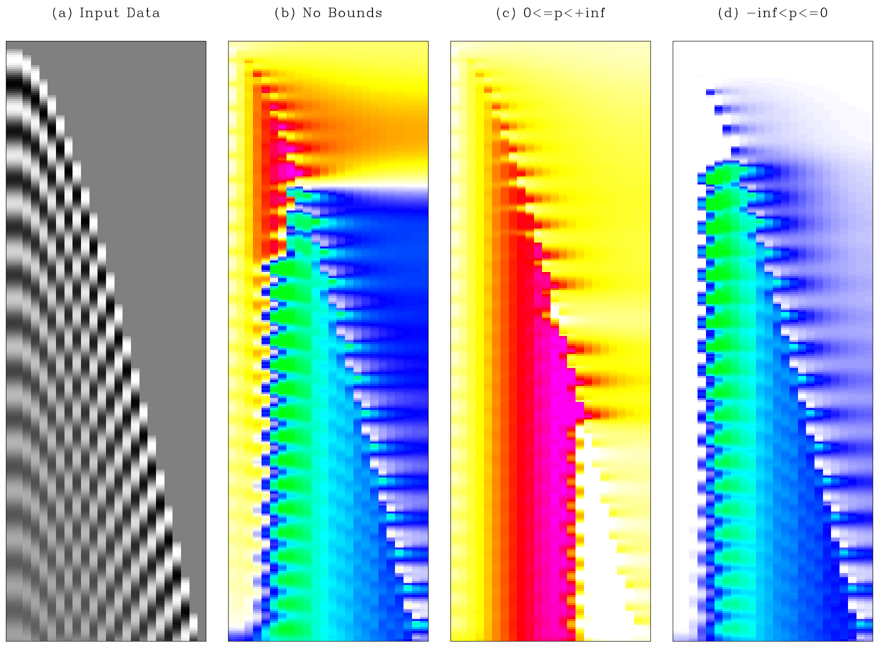

Figure 3: (a) Input data. (b) Estimated dips with L-BFGS-B without bound constraints. (c) Estimated positive dips only. (d) Estimated negative dips only. Positive and negative dips are found beyond aliasing. antoine1-aliasposneg [ER]

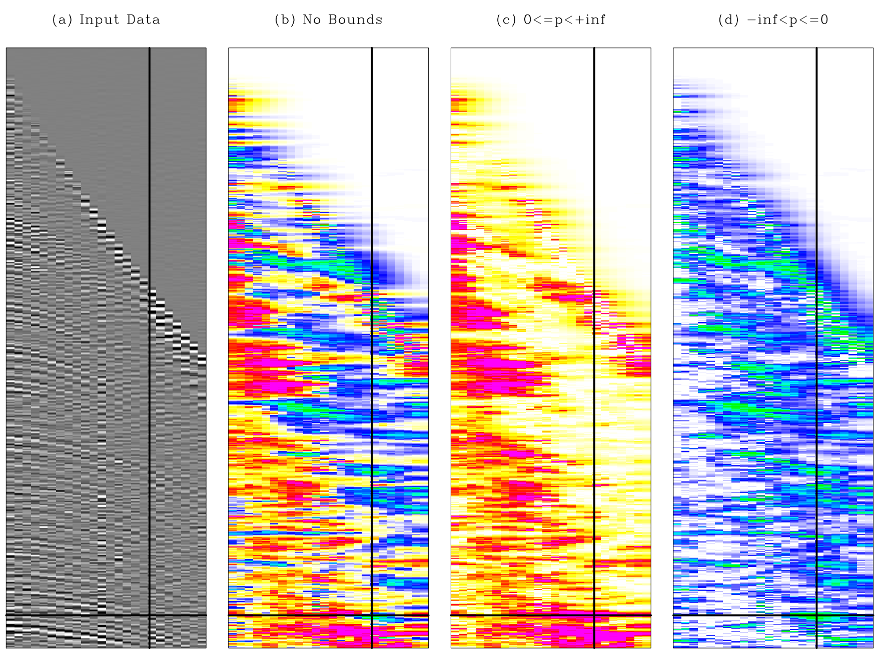

Figure 4: (a) Input data. The lines cross at a location where the dips should be positive. (b) Estimated dips with L-BFGS-B without bound constraints. Negative dips are actually found at the crossing of the lines. (c) Estimated positive dips only. The bound constrained optimization is capable of identifying the true positive dips. (d) Estimated negative dips only. antoine1-dipposneg [ER]

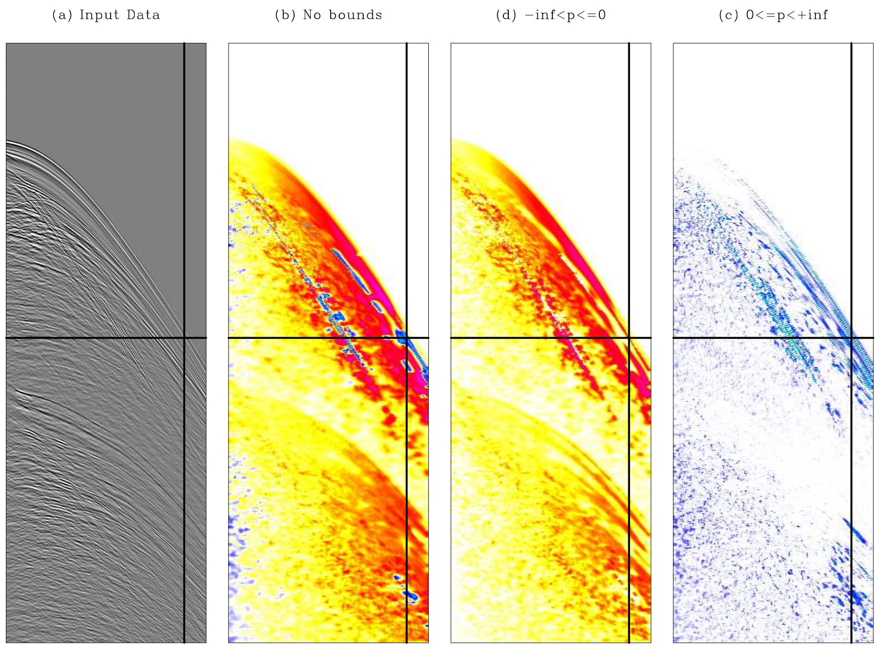

Figure 5: (a) Input data. (b) Estimated dips with L-BFGS-B without bound constraints. At the cross, negative dips are given although only positive dips should be found. (c) Estimated positive dips only. Positive dips are recovered everywhere. (d) Estimated negative dips only. antoine1-gmdip [ER]

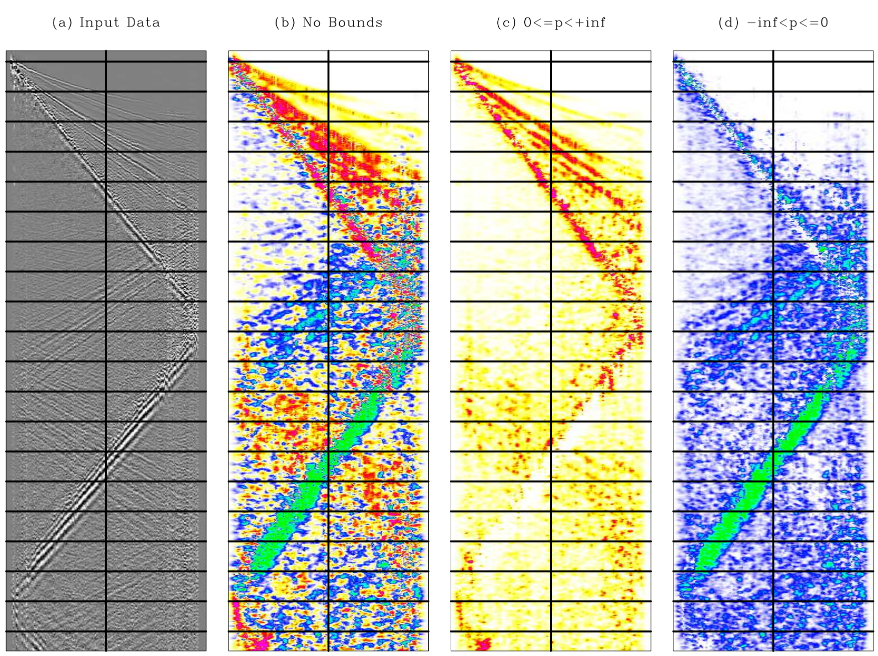

Figure 6: (a) Input data. The cross shows a location where two events overlap. (b) Estimated dips with L-BFGS-B without bound constraints. (c) Estimated positive dips only. At the cross, no positive dips are estimated because the overlapping event that goes in the other direction is too strong. (d) Estimated negative dips only. antoine1-idaposneg [ER]

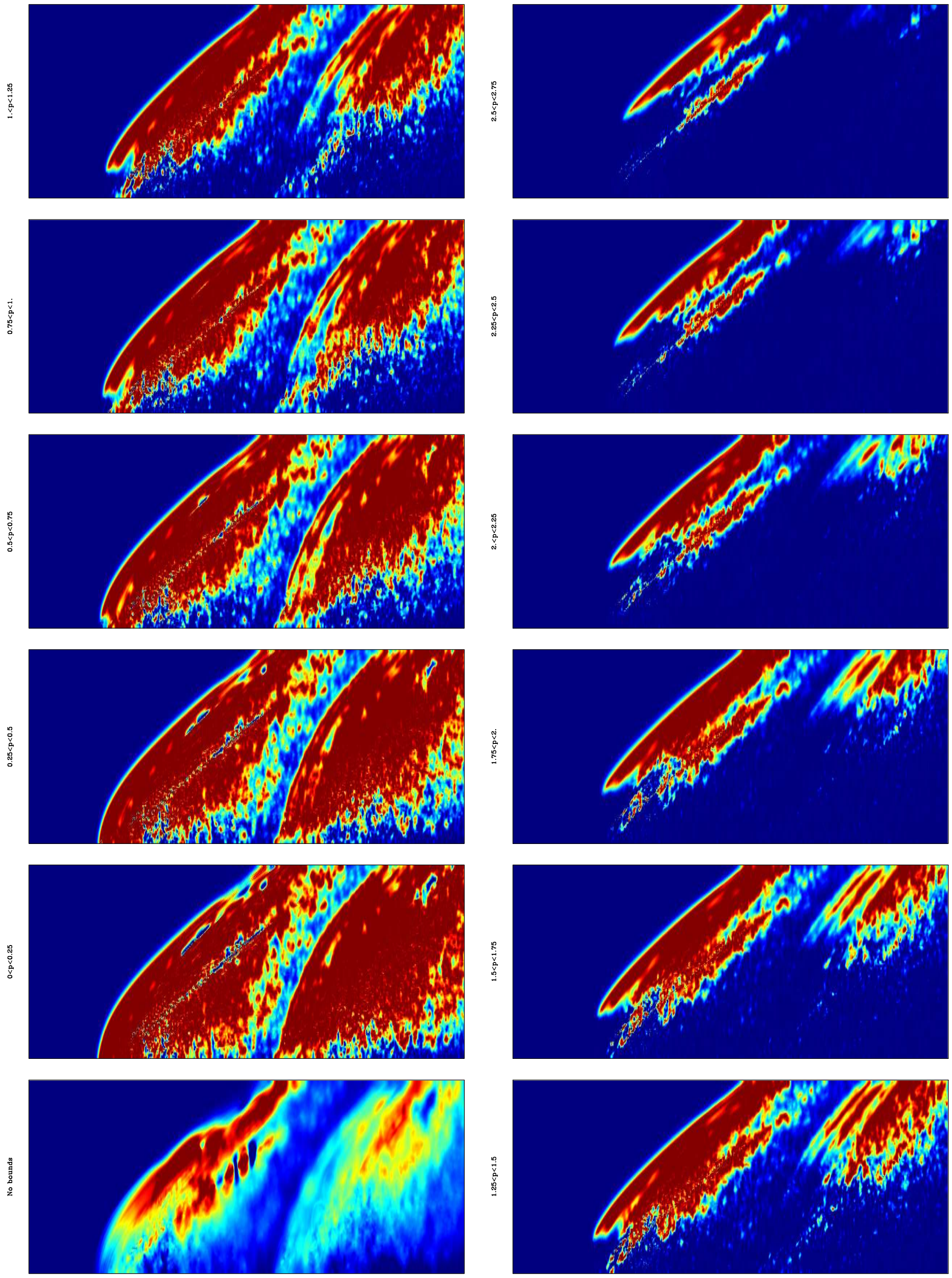

Figure 7: The fist panel on the top left corner shows the dips without the bounds. Then, from left to right, top to bottom, the range of dip increases by 0.25 between each panel, starting at $0 \leq \sigma \leq 0.25$ and finishing at $2.5 \leq \sigma \leq 2.75$. The lower and upper clip values are equal to the bounds for each panel. antoine1-gmdecomp [ER,M]