Stanford Exploration Project, Report 124, April 4, 2006, pages 11–18

Flattening with geological constraints

Jesse Lomask and Antoine Guitton

ABSTRACT

In areas with faults and poor signal/noise ratio, where reflectors can be discontinuous from place to place, a dip-based flattening technique might not be able to appropriately track sedimentary layers. To aid the flattening algorithms, one or few reflectors can be picked. This information can be then incorporated in our algorithms as geological constraints. In a first method, we add a model mask to a time domain solution using a Gauss-Newton approach that incorporates an initial solution. In a second method, we set the lower and upper bounds of a constrained optimization algorithm called limited memory BFGS with bounds (L-BFGS-B). Having incorporated the geological information, the flattening algorithms can accurately pick reflectors in 2D and 3D for noisy field data examples. In addition, preliminary results seem to indicate that the L-BFGS-B method converges faster than the Gauss-Newton method.

INTRODUCTION

In all of the flattening methods presented thus far (Lomask and Claerbout, 2002; Lomask, 2003a,b; Guitton et al., 2005b; Lomask et al., 2005, In press; Guitton et al., 2005a) a key selling point is that they require no picking. This would be fine if all data sets had perfectly estimated dips, but in the real world flattening without picking can produce results that are not perfect. Noise, both coherent and otherwise, can overwhelm the dip estimation causing reflectors in those areas to not be flat. Consequently, it would be useful to have the ability to add some geological constraints to restrict the flattening result in areas of poor data quality while allowing it to efficiently tackle other areas where the dips are accurate.

Here, we present two flattening methods with hard constraints. The hard constraints can be manually picked horizons or individual picks. Both methods require regularization in time (or depth) which carries along with it certain disadvantages as compared to unregularized methods. The largest disadvantage being that increasing regularization reduces local accuracy. In one method the hard constraints are implemented as model mask within the inversion. In the other method, the hard constraints are set as lower and upper bounds of a Limited-memory BFGS with Bounds (L-BFGS-B) algorithm (Zhu et al., 1997). We envision a tool that interpretors can run once completely unconstrained, then quality control the results. The interpreter can then adjust some horizons and then run the flattening method again honoring their changes. The algorithm is fast enough so that this process can be repeated several times.

In the future, computational and algorithmic improvements can result in a flattening method that is so efficient that the flattening process can be run between picks. This method also has the potential of combining other information into the flattening such as well log picks.

METHOD

The flattening method described in Lomask et al. (2005) creates a time-shift (or depth-shift) field $\boldsymbol{\tau}(x,y,t)$ such that its gradient approximates the dip $\mathbf{p}(\mathrm{x},\mathrm{y},\boldsymbol{\tau})$. The dip is a function of $\boldsymbol{\tau}$ because for any given horizon, the appropriate dips to be summed are the dips along the horizon itself. Using the gradient operator ($\boldsymbol{\nabla} = [\frac{\partial}{\partial x} \;\; \frac{\partial}{\partial y}]^T$) and the estimated dip ($\mathbf{p} = [\mathbf{p}_x \;\; \mathbf{p}_y]^T$), our regression is

$$\boldsymbol{\nabla}\boldsymbol{\tau}(x,y,t) \quad = \quad \mathbf{p}(x,y,\boldsymbol{\tau}). \tag{1}$$

To add regularization in the time direction, we apply a 3D gradient operator with a residual weight $\mathbf{W}_\epsilon$ that controls the amount of vertical regularization defined as

$$\mathbf{W}_\epsilon \boldsymbol{\nabla} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\epsilon} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial t} \end{bmatrix} \quad , \tag{2}$$

where $\mathbf{W}_\epsilon$ is a large block diagonal matrix consisting of two identity matrices $\mathbf{I}$ and a diagonal matrix $\boldsymbol{\epsilon}$=$\epsilon\mathbf{I}$. For simplicity, we implicitly chain this weight operator to the gradient operator to create a new operator now defined as a 3D gradient with an weighting parameter ϵ as

$$\boldsymbol{\nabla}_\epsilon \quad = \quad \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \epsilon\frac{\partial}{\partial t} \end{bmatrix} \quad . \tag{3}$$

The residual is defined as

$$\mathbf{r} \quad = \quad \boldsymbol{\nabla}_\epsilon \boldsymbol{\tau} - \mathbf{p} \quad = \quad \begin{bmatrix} \frac{\partial \boldsymbol{\tau}}{\partial x} \\ \frac{\partial \boldsymbol{\tau}}{\partial y} \\ \epsilon\frac{\partial \boldsymbol{\tau}}{\partial t} \end{bmatrix} \quad - \quad \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \\ \mathbf{0} \end{bmatrix}. \tag{4}$$

We solve this using a Gauss-Newton approach by iterating over equations (5)-(7), i.e.,

iterate {

$$\mathbf{r} \quad = \quad [\boldsymbol{\nabla}_\epsilon \boldsymbol{\tau}_k - \mathbf{p}(x,y,\boldsymbol{\tau}_k)] \tag{5}$$

$$\Delta\boldsymbol{\tau} \quad = \quad (\boldsymbol{\nabla}_\epsilon^T \boldsymbol{\nabla}_\epsilon)^{-1} \boldsymbol{\nabla}_\epsilon^T \mathbf{r} \tag{6}$$

$$\boldsymbol{\tau}_{k+1} \quad = \quad \boldsymbol{\tau}_k + \Delta\boldsymbol{\tau} \tag{7}$$

} ,

where the subscript k denotes the iteration number.

We wish to add a model mask $\mathbf{K}$ to prevent changes to specific areas of an initial $\boldsymbol{\tau}_0$ field. This initial $\boldsymbol{\tau}_0$ field can be picks from any source. In general, they may come from a manually picked horizon or group of horizons. These initial constraints do not have to be a continuous surfaces but instead could be isolated picks, such as well-to-seismic ties. To apply the mask we follow the same the development as the **operator approach to missing data** in (Claerbout, 1999) as

$$\begin{aligned}
\mathbf{0} &\approx \nabla_\epsilon \boldsymbol{\tau} - \mathbf{p} && (8)\\
\mathbf{0} &\approx \nabla_\epsilon (\mathbf{K} + (\mathbf{I}-\mathbf{K}))\boldsymbol{\tau} - \mathbf{p} && (9)\\
\mathbf{0} &\approx \nabla_\epsilon \mathbf{K}\boldsymbol{\tau} + \nabla_\epsilon (\mathbf{I}-\mathbf{K})\boldsymbol{\tau} - \mathbf{p} && (10)\\
\mathbf{0} &\approx \nabla_\epsilon \mathbf{K}\boldsymbol{\tau} + \nabla_\epsilon \boldsymbol{\tau}_0 - \mathbf{p} && (11)\\
\mathbf{0} \approx \mathbf{r} &= \nabla_\epsilon \mathbf{K}\boldsymbol{\tau} + \mathbf{r}_0 - \mathbf{p}. && (12)
\end{aligned}$$

Our resulting equations are now

iterate {

$$\begin{aligned}
\mathbf{r} &= \nabla_\epsilon \mathbf{K}\boldsymbol{\tau}_k - \mathbf{p}(\boldsymbol{\tau}_k) + \mathbf{r}_0 && (13)\\
\Delta\boldsymbol{\tau} &= (\mathbf{K}^T \nabla_\epsilon^T \nabla_\epsilon \mathbf{K})^{-1} \mathbf{K}^T \nabla_\epsilon^T \mathbf{r} && (14)\\
\boldsymbol{\tau}_{k+1} &= \boldsymbol{\tau}_k + \Delta\boldsymbol{\tau} && (15)
\end{aligned}$$

} .

Typically, we solve equation (6) in the Fourier domain, however in equation (14), $\mathbf{K}$ is non-stationary making its application in the Fourier domain difficult if not impossible. Therefore, for now we solve it in the time domain using conjugate gradients.

Preconditioning with the helical derivative

We can reduce the computation time significantly by preconditioning by substituting $\boldsymbol{\tau} = \mathbf{H}^{-1}\mathbf{m}$, where $\mathbf{H}^{-1}$ is the inverse of the 3D helical derivative (Claerbout, 1999). Preconditioning with the helical derivative is a logical choice because its inverse is very close to the inverse of a gradient. However, recall that our 3D gradient operator ∇_ϵ is actually a chain of two matrices $\mathbf{W}_\epsilon$ and ∇. Instead of approximating the inverse of $(\nabla^T\nabla)^{-1}$, we wish to approximate the inverse of $(\nabla^T {\mathbf{W}_\epsilon}^2 \nabla)^{-1}$. Therefore, we factor the finite difference approximation to the Laplacian with an ϵ parameter. In 2D, we factor the Laplacian with an epsilon parameter as:

$$\begin{array}{|c|c|c|}
\hline
 & -\epsilon^2 & \\
\hline
-1 & 2+2\epsilon^2 & -1 \\
\hline
 & -\epsilon^2 & \\
\hline
\end{array} \qquad (16)$$

To extend to 3D, we add another 2 to the center and another set of -1's in the 3rd dimension. Once factored it becomes a 3D helical derivative $\mathbf{H}_\epsilon$ with a scalar weight, ϵ, applied to the time(or depth) axis. When used as a regularization operator, this has the desirable property of having only one output. We choose not to include the mask $\mathbf{K}$ in the preconditioner because it is non-stationary.

With preconditioning, equation (13) becomes this:

$$\mathbf{r} \quad = \quad \nabla_\epsilon \mathbf{K} \mathbf{H}_\epsilon{}^{-1} \mathbf{m} - \mathbf{p}(x, y, \tau_k) + \mathbf{r}_0. \tag{17}$$

We pay a significant price in memory cost for this computational time saving. To precondition this equation requires chaining together several operators thus several temporary arrays are allocated. Also, the computational expense is tied to the number of coefficients used in the filter which in 3D is typically 20.

THE L-BFGS-B ALGORITHM

Alternatively, we can use the L-BFGS-B algorithm for imposing very tight constraints on the picked values of the tau field. The L-BFGS-B algorithm seeks to find a vector of model parameters τ such that we minimize

$$\min f(\tau) \text{ subject to } \tau \in \Omega, \tag{18}$$

where

$$\tau \in \Omega = \{\tau \in \Re^N \mid l_i \leq \tau_i \leq u_i\}, \tag{19}$$

with l_i and u_i being the lower and upper bounds for the model τ_i, respectively. In this case, l_i and u_i are called simple bounds. For the flattening technique, we want to minimize (Lomask, 2003b; Guitton et al., 2005b)

$$f(\tau) = \int\int \left[\left(p_x(x, y, z; \tau) - \frac{\partial \tau}{\partial x} \right)^2 + \left(p_y(x, y, z; \tau) - \frac{\partial \tau}{\partial y} \right)^2 \right] dx\, dy, \tag{20}$$

The L-BFGS-B algorithm combines a quasi-Newton update of the Hessian (second derivative) with a trust-region method. It has been successfully applied for flattening (Guitton et al., 2005b) and dip estimation (Guitton, 2004).

Incorporating the initial τ field is trivial with the L-BFGS-B method: we simply set the bounds where an a-priori value exists:

$$l_i \quad = \quad \tau_{0_i} - \alpha \times \tau_{0_i} \tag{21}$$

$$u_i \quad = \quad \tau_{0_i} + \alpha \times \tau_{0_i}, \tag{22}$$

where α is a small number (≈ 0.001). Note that the L-BFGS-B algorithm allows us to optionally activate the constraints for every point of the model space. Note that in equation 20, the objective function incorporates a smoothing in the vertical direction of the τ field as well.

Although not shown in this paper, the results of the L-BFGS-B algorithm are comparable to the Gauss-Newton approach. However at this stage, the L-BFGS-B algorithm converges faster than the Gauss-Newton technique with preconditioning.

RESULTS

We conducted preliminary tests of these modified flattening methods on both 2D and 3D data sets from the Gulf of Mexico and the North Sea, respectively. We only show the results obtained with the Gauss-Newton approach, the L-BFGS-B results being almost identical.

In the top of Figure 1 is an image of the lower portion of a salt boundary from the Gulf of Mexico. Overlain on this image is a single horizon that results from the unconstrained flattening method. The horizon fails to track the boundary accurately. In the lower image, two picked horizons have been added as hard constraints. Notice that the center horizon now does an overall much better job of tracking the boundary than its unconstrained counterpart. Unfortunately, the price paid is that by applying regularization in time, the constrained horizon does not track the character of data locally as well.

Figure 1: Images of a 2D Gulf of Mexico data set with flattening picks overlaying a salt boundary: (a) unconstrained, (b) with the upper and lower picked horizons as hard constraints. Notice that the middle horizon in (b) tracks the boundary more accurately than its unconstrained counterpart in (a). jesse2-GoM_combo [ER]

In Figure 2 is a 3D North Sea data set. The top shows several horizons that result from unconstrained flattening. Although some of the horizons are well tracked, several are not. In the lower figure, we picked the brightest horizon and passed it into the flattening method as a hard constraint. Notice that the reflectors above it now tend to more accurately track their respective events.

Also in Figure 2, notice how the reflectors at the top of the cube are tracked more accurately without constraints. This is as expected because the regularization is causing the reflectors to conform to one another. To mitigate this, we could have passed an upper picked horizon from the unconstrained result as an additional hard constraint to the lower result.

Figure 2: Images of a 3D North Sea data set with flattening picks: (a) unconstrained, (b) with a single picked horizon as a hard constraint. Notice that with the hard constraint, reflectors near the picked horizon are tracked more accurately. jesse2-elf_pck3_combo [ER]

CONCLUSIONS AND FUTURE WORK

We have added constraints to two flattening methods so that human interpretation can be incorporated into the solutions. We have successfully demonstrated their effectiveness on both 2D and 3D field data sets. Both methods converge to the same solution, the most important difference is which method converges fastest with the least amount of memory usage. Preliminary tests indicate a similar memory usage (very large, about 40 times the size of the input

image) for both techniques with an advantage to the L-BFGS-B algorithm in terms of speed (3 to 5 times faster for similar convergence).

In order to add constraints, regularization is required to enforce conformity between horizons. In many geological settings this is desirable, however there are some notable exeptions such as angular unconformities. If these unconformities can be identified, it may be necessary to add a residual weight to essentially disable the regularization at those regions of the data.

The ability to incorporate some picking will likely allow the reconstruction of horizons across faults that cut across the entire data cube. As described in (Lomask et al., 2005), in order to automatically flatten a data cube with faults, at least half of the fault's tip line must be encased within the data. That is, the fault must die out. With the ability to add some picks, faults that do not die out can be reconstructed. We envision an interpreter can pick a few points on a 2D line and then flatten the cube. With computational improvements in both the algorithm and hardware, this method could be applied on the fly, as the interpreter adds new picks.

More efficient algorithms may not be far off. It seems plausible to add constraints to the flattening method with the cosine transform described in Lomask and Fomel (2006). Alternatively, we can apply the cosine transform method as either an initial solution or a preconditioner to conjugate gradients in an approach similar to Ghiglia and Romero (1994) for 2D phase unwrapping. Lastly, we maybe able to precondition the full equation with the cosine transform, this method would seem to have the most promise for achieving superior computational efficiency.

ACKNOWLEDGMENT

We would like to thank WesternGeco and Total for supplying the data used in this paper.

REFERENCES

Claerbout, J., 1999, Geophysical estimation by example: Environmental soundings image enhancement: Stanford Exploration Project, `http://sepwww.stanford.edu/sep/prof/`.

Ghiglia, D. C. and L. A. Romero, 1994, Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods: Optical Society of America, **11**, no. 1, 107–117.

Guitton, A., J. M. Lomask, and S. Fomel, 2005a, Non-linear estimation of vertical delayx: 75th SEG Meeting, Extended Abstracts.

Guitton, A., J. Lomask, and S. Fomel, 2005b, Non-linear estimation of vertical delays with a quasi-Newton method: SEP–**120**, 167–178.

Guitton, A., 2004, Bound constrained optimization: application to the dip estimation problem: SEP–**117**, 51–62.

Lomask, J. and J. Claerbout, 2002, Flattening without picking: SEP–**112**, 141–150.

Lomask, J. and S. Fomel, 2006, Flattening with cosine transforms: SEP–**124**.

Lomask, J., A. Guitton, S. Fomel, and J. Claerbout, 2005, Update on flattening without picking: SEP–**120**, 137–158.

Lomask, J., A. Guitton, S. Fomel, and J. Claerbout, In press, Flattening without picking: Geophycics.

Lomask, J., 2003a, Flattening 3D seismic cubes without picking: 73th SEG Meeting, Extended Abstracts, 1402–1405.

Lomask, J., 2003b, Flattening 3-D data cubes in complex geology: SEP–**113**, 247–260.

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