GEOPHYSICS, VOL. 71, NO. 4 (JULY-AUGUST 2006); P. P13–P20, 4 FIGS.
10.1190/1.2210848

Flattening without picking

Jesse Lomask[1], Antoine Guitton[1], Sergey Fomel[2], Jon Claerbout[1], and Alejandro A. Valenciano[1]

ABSTRACT

We present an efficient full-volume automatic dense-picking method for flattening seismic data. First local dips (stepouts) are calculated over the entire seismic volume. The dips are then resolved into time shifts (or depth shifts) using a nonlinear Gauss-Newton iterative approach that exploits fast Fourier transforms to minimize computation time. To handle faults (discontinuous reflections), we apply a weighted inversion scheme. The weight identifies locations of faults, allowing dips to be summed around the faults to reduce the influence of erroneous dip estimates near the fault. If a fault model is not provided, we can estimate a suitable weight (essentially a fault indicator) within our inversion using an iteratively reweighted least squares (IRLS) method. The method is tested successfully on both synthetic and field data sets of varying degrees of complexity, including salt piercements, angular unconformities, and laterally limited faults.

INTRODUCTION

Despite numerous advances in computational power in recent years, seismic interpretation still requires significant manual picking. One of the main goals of interpretation is to extract geologic and reservoir features from seismic data. One commonly used interpretation technique that helps with this effort is flattening data on horizons (e.g., Lee, 2001), also known as stratal slicing (Zeng et al., 1998). This procedure removes structure and allows the interpreter to see geologic features as they were emplaced. For instance, after flattening seismic data, an interpreter can see in one image an entire floodplain complete with meandering channels. However, to flatten seismic data, a horizon needs to be identified and tracked throughout the data volume. If the structure changes often with depth, then many horizons need to be identified and tracked. This picking process is time consuming and expensive.

Certain visualization products and autopickers seek to make picking and flattening processes as efficient as possible. However, they often suffer from weaknesses that prevent them from being truly practical. For example, 3D volume interpretation packages allow interpreters to view their data with depth perception using stereo glasses. These products have an opacity ability (James et al., 2002) that allows interpreters to make unwanted data transparent. Unfortunately, unless the zone of interest has a known unique range of attribute values, interpreters resort to picking on 2D slices. Additionally, traditional amplitude-based autopickers can fail if the horizon being tracked has significant amplitude variation or, worse, polarity reversal. Other tracking techniques such as artificial neural networks are less sensitive to amplitude variations but are still prone to error if the seismic wavelet character varies significantly from the training data (Leggett et al., 1996).

In this document, we propose a method for automatically flattening entire 3D seismic cubes without manual picking. This concept was previously presented by Lomask and Claerbout (2002) and Lomask (2003a, b). It is an efficient algorithm that intrinsically performs automatic dense picking on entire 3D cubes at once. Our method involves first calculating local dips (stepouts) everywhere in the data using a dip estimation technique (Claerbout, 1992; Fomel, 2002). These dips are resolved into time shifts (or depth shifts) via a nonlinear least-squares problem. We use an iterative Gauss-Newton approach that integrates dips quickly using fast Fourier transforms (FFTs). Subsequently, the data are shifted vertically according to the summed time shifts to output a flattened volume. Bienati and Spagnolini (1998, 2001) and Bienati et al. (1999a, 1999b) use a similar approach to resolve the dips numerically into time shifts for autopicking horizons and flattening gathers but solve a linear version of the problem. Stark (2004) takes a full-volume approach to achieve the same goal but unwraps instantaneous phase instead of dips. Blinov and Petrou (2005) use dynamic programming to track horizons by summing local dips.

Manuscript received by the Editor May 26, 2005; revised manuscript received December 7, 2005; published online July 11, 2006.
[1]Stanford Exploration Project, Mitchell Building, Department of Geophysics, Stanford University, Stanford, California 94305. E-mail: lomask@sep.stanford.edu; antoine@sep.stanford.edu; clearbout@sep.stanford.edu; valencia@sep.stanford.edu.
[2]Bureau of Economic Geology, University of Texas at Austin, Austin, Texas 78713. E-mail: sergey.fomel@beg.utexas.edu.

As with amplitude-based autopickers, amplitude variation also affects the quality of dip estimation and, in turn, impacts the quality of our flattening method. However, the effect will be less significant because our method can flatten the entire data cube at once — globally — in a least-squares sense, minimizing the effect of poor dip information. Discontinuities in the data from faults tend to corrupt the local dip estimates at the faults. In this case, weights identifying the faults are applied within the iterative scheme, allowing reconstruction of horizons for certain fault geometries. Such weights could be obtained from a previously determined fault model. If a fault model is not provided, we automatically generate suitable weights using iteratively reweighted least squares (IRLS). Once a seismic volume is flattened, automatic horizon tracking becomes a trivial matter of reversing the flattening process to unflatten flat surfaces. The prestack applications for this method are numerous. For example, time shifts can be incorporated easily into an automatic velocity-picking scheme (Guitton et al., 2004).

In the following sections, we present the flattening methodology and a series of real-world geologic challenges for this method. The first is a 3D synthetic data set generated from a model with a dipping fault and thinning beds. Then we present a structurally simple, 3D salt piercement field data set from the Gulf of Mexico. We consider it structurally simple because the geologic dips do not change significantly with depth. Increasing complexity, we flatten a 3D field data set from the North Sea that contains an angular unconformity and has significant folding. Last, we present a 3D faulted field data set from the Gulf of Mexico.

FLATTENING THEORY

Ordinarily, to flatten a single surface, each sample is shifted vertically to match a chosen reference point. For instance, this reference point can be the intersection of the horizon and a well pick. In three dimensions, this reference point becomes a vertical line (reference trace). To flatten 3D cubes, our objective is to find a mapping field $\tau\,(x,y,t)$ such that each time slice of this field contains the locations of all data points along the horizon that happens to intersect the reference trace at that time slice. We achieve this by summing dips. The basic idea is similar to phase unwrapping (Ghiglia and Romero, 1994); but instead of summing phase differences to get total phase, dips are summed to get total time shifts which are used then to flatten the data.

The first step is to calculate local dips everywhere in the 3D seismic cube. Dips can be calculated efficiently using a local plane-wave destructor filter as described by Claerbout (1992) or with an improved dip estimator described by Fomel (2002). We primarily use the latter. For each point in the data cube, two components of dip, b and q, are estimated in the x- and y-directions, respectively. These can be represented everywhere on the mesh as $b(x,y,t)$ and $q(x,y,t)$. If the data to be flattened are in depth, then dip is dimensionless, but when the data are in time, dip has units of time over distance.

Our goal is to find a time-shift (or depth-shift) field $\tau\,(x,y,t)$ such that its gradient approximates the dip $\mathbf{p}(x,y,\tau)$. The dip is a function of τ because for any given horizon, the appropriate dips to be summed are the dips along the horizon itself. Using the matrix representation of the gradient operator $\left(\boldsymbol{\nabla} = \left[\frac{\partial}{\partial x}\ \frac{\partial}{\partial y}\right]^T\right)$ and the estimated dip ($\mathbf{p} = [b\ q]^T$), our regression (developed in Appendix A) is

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

Because the estimated dip field $\mathbf{p}(x,y,\tau)$ is a function of the unknown field $\tau\,(x,y,t)$, this problem is nonlinear and therefore difficult to solve directly. We solve it using a Gauss-Newton approach by iterating over the following equations:

$$\mathbf{r} = \boldsymbol{\nabla}\tau_k(x,y,t) - \mathbf{p}(x,y,\tau_k), \tag{2}$$

$$\Delta\tau = ([\boldsymbol{\nabla}^T\boldsymbol{\nabla}]^{-1}\boldsymbol{\nabla}^T)\mathbf{r}, \tag{3}$$

$$\tau_{k+1}(x,y,t) = \tau_k(x,y,t) + \Delta\tau, \tag{4}$$

where the subscript k denotes the iteration number.

A more intuitive way to understand this method is to consider the first iteration. If no initial solution is provided, τ_0 will be zero and the residual $\mathbf{r}$ will be the input dips $\mathbf{p}$ along each time slice. These dips are then summed into time shifts using equation 3. At this point, these time shifts can be used to flatten the data. However, because the dips were summed along time slices and not along individual horizons, the data will not be perfectly flat. The degree that the data are not flat is related to the variability of the dip in time. At this point, we could reestimate new dips on the partially flattened data and then repeat the process. Iterating in this way will eventually flatten the data. This is essentially how our method works, but instead of reestimating dips at each iteration, we correct the original dips at each iteration with equation 2. Not only is this more efficient than estimating new dips at each iteration, but it is also more robust because discontinuities introduced by flattening create inaccurate dips.

Convergence is generally reached when the change in normalized residual between consecutive iterations is smaller than a user-specified tolerance μ, i.e.,

$$\frac{\|\mathbf{r}_{k-1}\| - \|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} < \mu. \tag{5}$$

A value $\mu = 0.001$ often gives sufficiently flat results. An appealing alternative stopping criterion would be to consider only the absolute value of the residual $\|\mathbf{r}_k\|$ because it is essentially the sum of the remaining dips. In this case, the stopping tolerance would be the user-specified minimum average dip value. Unfortunately, dip value is very sensitive to the amount of noise and, consequently, would not make a satisfactory stopping criterion. The stopping criterion in equation 5 is less sensitive to the absolute value of the noise because it considers only the change in residual from iteration to iteration. Once convergence is achieved, the resulting time-shift field $\tau\,(x,y,t)$ contains all of the time shifts required to map the original unflattened data to flattened data. This is implemented by applying the time shifts relative to a reference trace. In other words, each trace is shifted to match the reference trace. For simplicity, we assume the reference trace to be a vertical line; however, it could, in principle, be nonvertical or even discontinuous. In general, we operate on one time slice at a time. After iterating until convergence, we then select the next time slice and proceed down through the cube. In this way, each time slice is solved independently. However, the number of iterations can be significantly reduced by passing the solution of the previous time slice as an initial solution to the current time slice.

To improve efficiency greatly, we solve equation 3 in the Fourier domain using the FFT. We apply the divergence to the estimated dips and divide by the z-transform of the discretized Laplacian in the Fourier domain, i.e.,

$$\Delta\tau \approx \mathrm{FFT}_{2D}^{-1}\left[\frac{\mathrm{FFT}_{2D}[\nabla^T\mathbf{r}]}{-Z_x^{-1}-Z_y^{-1}+4-Z_x-Z_y}\right], \quad (6)$$

where $Z_x = e^{iw\Delta x}$ and $Z_y = e^{iw\Delta y}$. This fast Fourier approach is similar to the method of Ghiglia and Romero (1994) for unwrapping phase. This amounts to calling both a forward and inverse FFT at each iteration. The ability to invert the 2D Laplacian in one step is the key to the method's computational efficiency. To avoid artifacts in the Fourier domain, we mirror the divergence of the residual $\nabla^T\mathbf{r}$ before we apply the Fourier transform. Mirroring is described by Ghiglia and Pritt (1998) for application to 2D phase unwrapping. They also describe a modification that uses cosine transforms instead of FFTs to eliminate the need for mirroring.

The flattening process stretches or compresses the data in time, altering the spectrum. Even worse, the flattening process can disrupt the continuity of the data. To ensure a monotonic and continuous result, one should first smooth the input dips in time (or depth). In some instances, it may be necessary to enforce smoothness while integrating the dips. This can be accomplished by defining a 3D gradient operator with an adjustable smoothing parameter as

$$\nabla_\epsilon\tau = \begin{bmatrix} \dfrac{\partial\tau}{\partial x} \\ \dfrac{\partial\tau}{\partial y} \\ \epsilon\dfrac{\partial\tau}{\partial t} \end{bmatrix} \approx \begin{bmatrix} b \\ q \\ 0 \end{bmatrix} = \mathbf{p}. \quad (7)$$

Our new operator ∇_ϵ has a scalar parameter ϵ used to control the amount of vertical smoothing. In the case of flattening an image in depth, ϵ is dimensionless; in the case of time, it has units of time over distance. This operator requires integrating the entire 3D volume of dips at once rather than slice by slice. The 2D FFTs in equation 6 are replaced with 3D FFTs. Consequently, each iteration is slowed.

The magnitude of the smoothing parameter ϵ depends on competing requirements of the amount of noise and structural complexity. If the structure is changing considerably with time, then it should be chosen to be as small as possible. However, if there is significant noise resulting in erroneous dips, it should be larger.

There is a subtle difference between smoothing the input dips or smoothing the time-shift field. The time shift is essentially an integration of the input dips, so small errors in the input dips can accumulate into large errors in the cumulative time-shift field. In the same way that traditional amplitude-based autopickers can get off the intended horizon and jump to the wrong horizon by making one error, merely smoothing the input dips can cause this algorithm to flatten the wrong reflector. This is less likely to occur by smoothing the time-shift field.

Weighted solution for faults

Local dips estimated at fault discontinuities can be inaccurate. To prevent such inaccurate dips from adversely affecting the quality of the flattening, we can sum dips around the faults and ignore the spurious dips across the faults to get a flattened result; a weighting is applied to the residual to ignore fitting equations that are affected by the bad dips estimated at the faults. The regression to be solved is now

$$\mathbf{W}\nabla\tau\,(x,y,t) = \mathbf{W}\mathbf{p}(x,y,\tau). \quad (8)$$

Equation 3 should then become

$$\Delta\tau = (\nabla^T\mathbf{W}^T\mathbf{W}\nabla)^{-1}\nabla^T\mathbf{W}^T\mathbf{r}. \quad (9)$$

However, because we cannot apply a nonstationary weight in the Fourier domain, we ignore the weights in equation 9 and iterate over the same equations as before, except equation 2 is now replaced with

$$\mathbf{r} = \mathbf{W}[\nabla\tau_k(x,y,t) - \mathbf{p}(x,y,\tau_k)]. \quad (10)$$

By ignoring the weights in equation 9, we are able to still use the Fourier method in equation 6. Naturally, this means that the Fourier method is not approximating the inverse as well as before, causing the algorithm to need more iterations to converge. This approach is similar to that of Ghiglia and Romero (1994) for phase unwrapping.

IRLS for fault weights

We also have an option of automatically creating the fault weights from the data cube itself within the inversion. It is known that the ℓ_1 norm is less sensitive to spikes in the residual (Claerbout and Muir, 1973). Minimizing the ℓ_1 norm makes the assumption that the residuals are exponentially distributed and have a long-tailed distribution relative to the Gaussian function assumed by the ℓ_2 norm inversion. We can take advantage of this to honor dips away from faults and ignore inaccurate dips at faults. By iteratively recomputing a weight and applying it in equation 10, we tend to ignore outlier dip values that occur at faults. By gradually ignoring more and more of the outliers, the solution is no longer satisfying an ℓ_2 (least-squares) cost function but another more robust cost function. Our weight function forces a Geman-McClure distribution (Geman and McClure, 1987) using

$$\mathbf{W} = \frac{1}{\left(1+\mathbf{r}^2/\bar{r}^2\right)^2}, \quad (11)$$

where $\bar{r}$ is an adjustable damping parameter. We choose the Geman-McClure distribution because it is very robust and tends to create weights that are almost binary, much like a fault indicator. The resulting fault indicator model is unique in that it represents the fault picks that best flatten the data. This IRLS approach adds considerably more iterations to the flattening process. Also, it depends strongly on the quality of the data, precluding its use on very noisy data; with such data it tends to create false faults.

The damping parameter $\bar{r}$ controls the sensitivity to outliers. If $\bar{r}$ is large, then the weight will be closer to one everywhere and, as a result, almost all of the dips will be honored. If $\bar{r}$ is small, more of the outlier dips will be ignored. If $\bar{r}$ is too small, then dips that are not affected by faults and noise can be ignored, creating false faults. Practice has shown that it is better to start out with a relatively large damping parameter value of $\bar{r} > 2$ and then scale it by 0.8 when the weight is no longer changing. This generally means scaling it every five IRLS iterations.

Computational cost

For a data cube with dimensions $n = n_1 \times n_2 \times n_3$, each iteration requires n_1 forward and reverse 2D FFTs. Therefore, the number of

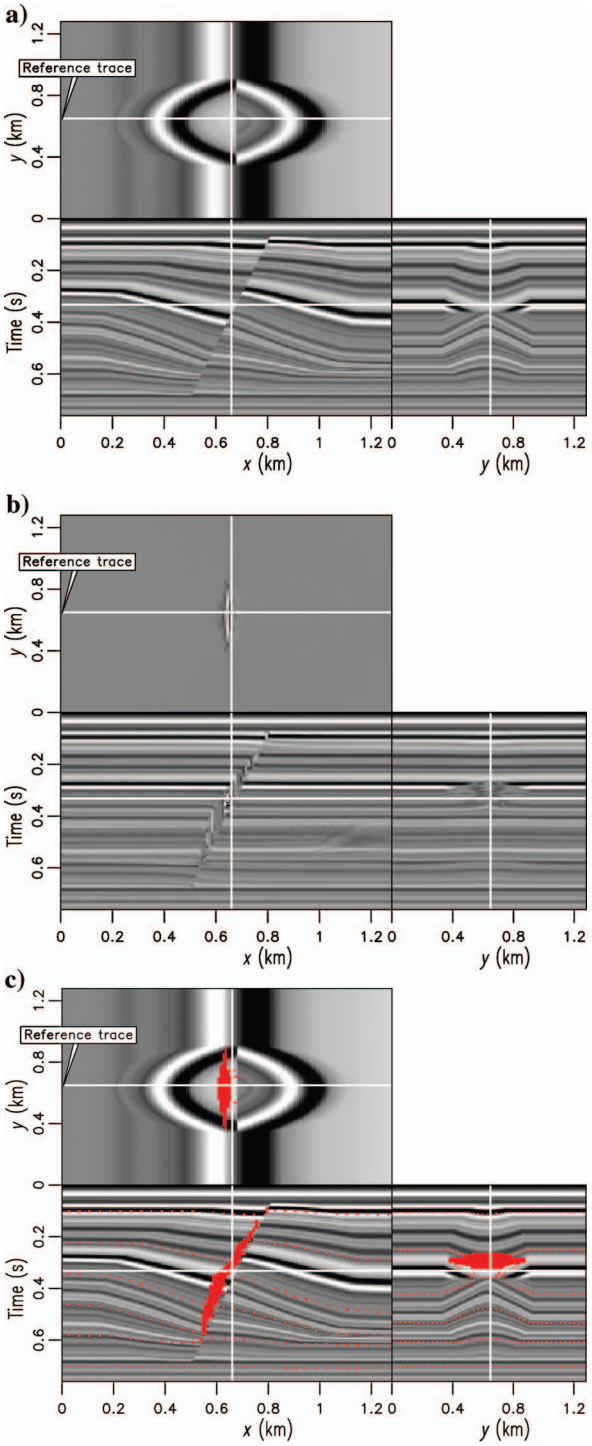

Figure 1. A synthetic model with structure varying with depth as well as a dipping discontinuity representing a fault. (a) The white lines superimposed onto the orthogonal sections identify the location of these sections: a time slice at 0.332 s, an inline section at y = 0.65 km, and a crossline section at x = 0.66 km. The horizontal scale for the right face is the same as the vertical scale for the upper face. The reference trace is located at x = 0.0 km and y = 0.65 km. (b) As Figure 1a only after flattening using the IRLS method. Notice how the image is flattened accurately. (c) Result of overlaying tracked horizons on the image in Figure 1a. The fault weight that was generated automatically by the IRLS approach is displayed in solid red. The method successfully tracks the horizons (dashed red).

operations per iteration is about $8n(1 + \log(4n_2n_3))$. The number of iterations is a function of the variability of the structure and the degree of weighting. For instance, if the structure is constant with time (or depth), it will be flattened in one iteration (whereas in this case $\mathbf{p}$ is independent of τ, causing the problem to be linear). On the other hand, if a weight is applied and the structure changes much with depth, it may take as many as 100 iterations. The IRLS approach requires many more iterations because the problem is solved again each time a new weight is estimated. A typical problem may take ten IRLS iterations, causing a tenfold increase in computation time. However, initializing each slice with the solution of the previous slice can greatly reduce the total number of iterations required to converge.

EXAMPLES OF 3D FLATTENING

We demonstrate this flattening method's efficacy on a synthetic data set and field 3D data sets. We start with a synthetic data set to illustrate how this method can handle data with faults and folds, and then we test the method on several 3D field data sets with varying degrees of structural complexity.

Synthetic data

Figure 1a is a 3D synthetic data set that presents two geologic challenges. First, the structure is changing with depth, requiring multiple nonlinear iterations. Second, a significant fault is present in the middle of the cube. As can be seen on the time slice, the fault is limited laterally and terminates within the data cube, i.e., the tip line of the fault is contained within the data. The tip line is a boundary of a fault that delineates the limit of slip. Because dips computed at fault discontinuities are, in general, inaccurate and will compromise the quality of the flattening result, we use the IRLS method to estimate iteratively a weight that ignores the inaccurate dips estimated at the fault and honors the dips away from the fault.

The flattening result is shown in Figure 1b. The location of the reference trace is displayed on the horizon slice. This trace is held constant while the rest of the cube is shifted vertically to match it. Notice how the cube is flattened accurately except in the area of the fault itself. The method is able to flatten this cube without prior information of the fault location because the fault is laterally limited and the S/N ratio is high. It is important that the fault be limited laterally so that dips can be summed around the fault. Furthermore, the IRLS approach requires a good S/N ratio to prevent it from creating false faults. The damping parameter $\bar{r} = 0.2$ is found by testing on one time slice in the center of the cube. Initially, it was set at a higher value of $\bar{r} = 2$ and was then gradually lowered until the results improved. If we had already had a fault model or automatic fault detector, such as a coherency cube (Bahorich and Farmer, 1995), we could have passed that to the inversion as a weight instead of implementing the less robust and more computationally expensive IRLS method. The automatically estimated fault weight is shown in solid red in Figure 1c. The fault weight is slightly shifted from the discontinuity because the weight identifies poor dips in the flattened cube.

The estimated $\tau\,(x,y,t)$ field applied to flatten the data can also be exploited to reverse the process. That is, we can use the time shifts to unflatten data that is already flat. By unflattening flat surfaces and overlaying them on the data, we are essentially picking any or all of the horizons in the data cube. Figure 1c displays as dashes every fifteenth horizon overlain on the synthetic data shown in Figure 1a. We could just as easily have displayed every horizon, but the image

would appear too cluttered. Notice that the horizons are well tracked, even across a fault. Additionally, the slight shifts in fault weight can be removed by unflattening the fault weight itself.

Gulf of Mexico salt piercement data

Figure 2a is a field 3D data cube from the Gulf of Mexico provided by Chevron. It consists of structurally simple horizons that have been warped up around a salt piercement. Several channels can be seen in the time slice at the top of Figure 2a, but they are largely obscured by the gradual amplitude overprint of a nearly flat horizon that is cut by the time slice.

Figure 2b shows the flattened output of the data in Figure 2a. We flattened this data set using the method without weights. The seismic cube has been converted from a stack of time slices to a stack of horizon slices. Notice that the gradual amplitude overprint present in the unflattened data is no longer present in the flattened data. This is because horizons are no longer cutting across the image. Several channels are now easily visible on the horizon slice. Also, the beds adjacent to the salt dome have been partially reconstructed, causing the salt to occupy a smaller area in Figure 2b.

Figure 2c displays three horizons overlain on the original data in Figure 2a. The horizons track the reflectors on the flanks of the salt well. Within the salt, the horizons gradually diverge from their respective reflector events as the estimated dip becomes less accurate. The time slice at the top displays the swath of a tracked horizon.

North Sea unconformity data

Figure 3a shows a 3D North Sea data set provided by Total. Marked by considerable folding and a sharp angular unconformity, this data set presents a real-world flattening challenge. The flattening result shown in Figure 3b was made using the method without weights. To preserve the continuity of the data, we used a smoothing parameter of $\epsilon = 1.0$ in equation 7. We estimated ϵ through trial and error and aimed to make it as small as possible while preserving the image quality. As a result, the flattened data are not completely flat. Had we used $\epsilon = 0.0$, the data would be flatter but would lose continuity. Consequently, the trade-off between continuity and flatness emerges in cases of pinch-outs and unconformities.

To achieve good agreement between the tracked horizons and the data, we found $\epsilon = 0.0$ to be preferable, i.e., no imposed continuity. The result is shown in Figure 3c. The time slice at the top shows the swaths of a few horizons. Overall, the tracked horizons track up to and along the unconformity, although some significant errors occur where the data quality is poor and, as a result, the estimated dips are inaccurate.

Gulf of Mexico faulted data

Figure 4a is an image of a Gulf of Mexico data set containing a fault. We use the IRLS method to flatten this data cube and estimate a fault weight simultaneously. This IRLS method uses the residual to identify areas to ignore incorrect dips. The damping parameter value $\bar{r} = 0.2$ was found by trial and error on a particular time slice, and subsequently applied to the entire cube. The flattening results are shown in Figure 4b. Notice that the horizons are flat, even directly across from the fault. Figure 4c shows the original data with two unflattened horizons (dashed) overlying it. It successfully tracks the horizons even across the fault. The automatically generated fault weight is shown in solid red.

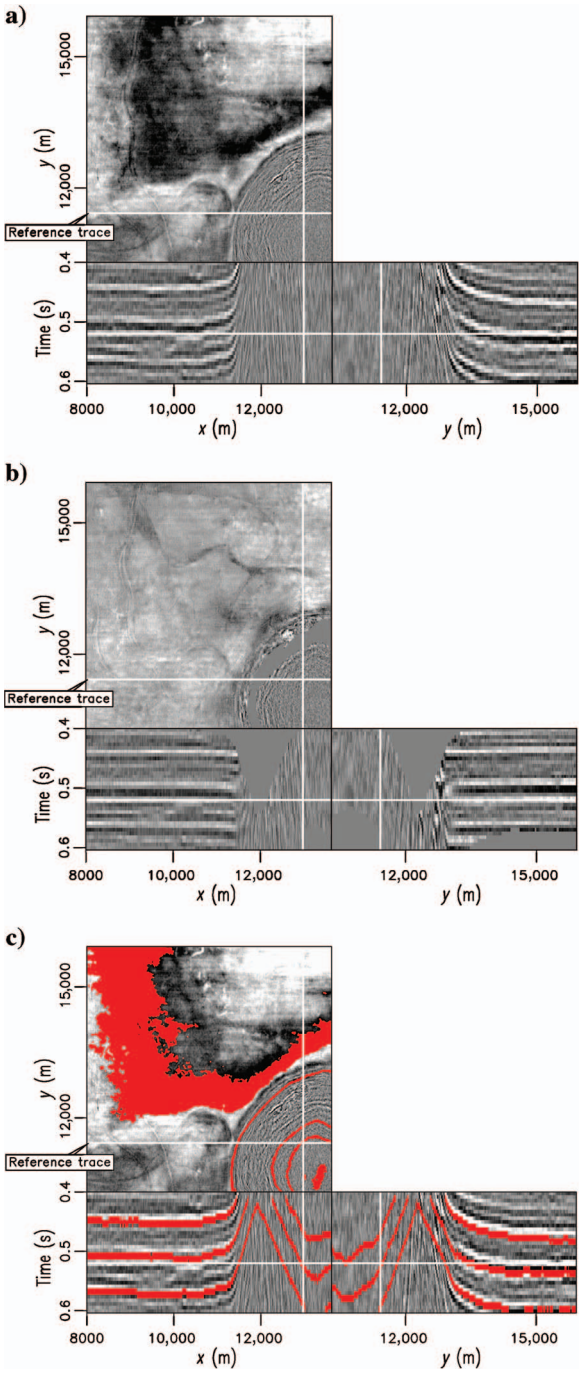

Figure 2. Chevron Gulf of Mexico data. (a) The white lines superimposed onto the orthogonal sections identify the location of these sections: a time slice at 0.522 s, an inline section at $y = 11{,}436$ m, and a crossline section at $x = 12{,}911$ m. The horizontal scale for the right face is the same as the vertical scale for the upper face. The reference trace is located at $x = 8000$ m and $y = 11{,}436$ m. Notice how the beds have been forced up to steep angles by the emplacement of a salt piercement. (b) As Figure 2a only after flattening without any weighting. The top panel is now a horizon slice displaying several clearly visible channels. (c) Result of overlaying tracked horizons on the image in Figure 2a. The horizons have been tracked accurately even up to the considerably steep dips leading into the salt piercement.

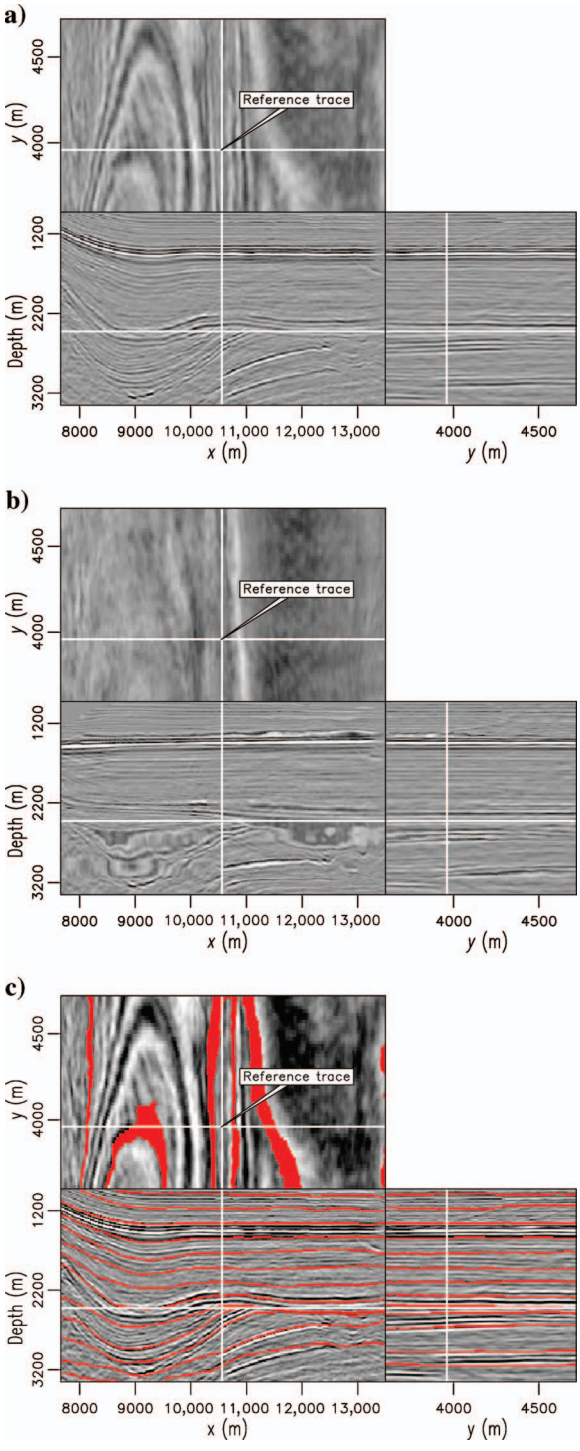

Figure 3. North Sea data. (a) The white lines superimposed onto the orthogonal sections identify the location of these sections: a depth slice at 2425 m, an inline section at y = 3960 m, and a crossline section at x = 10,560 m. The horizontal scale for the right face is the same as the vertical scale for the upper face. The reference trace is located at x = 10,560 m and y = 3960 m. Note the angular unconformity at 2425 m. (b) As Figure 3a only after flattening without weighting. A smoothing parameter $\epsilon = 1.0$ imposes smoothness on the tracking. (c) The result of overlaying tracked horizons on the image in Figure 3a. Here we used a smoothing parameter $\epsilon = 0.0$, causing horizons that lead to the angular unconformity to be tracked.

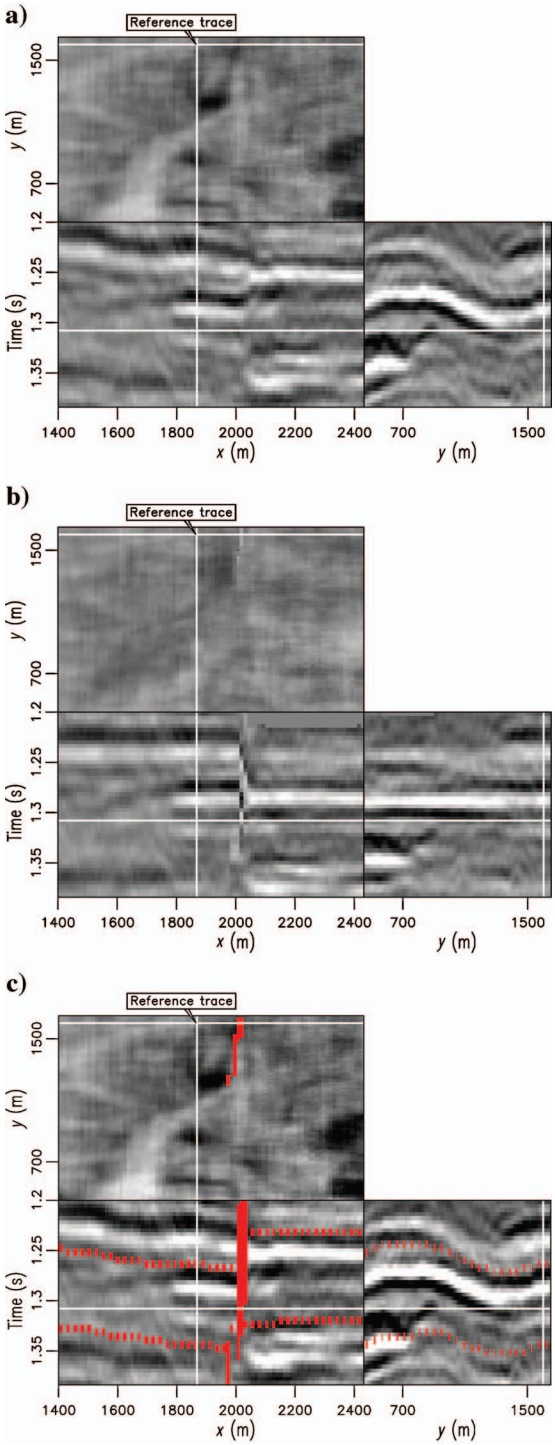

Figure 4. Gulf of Mexico data set displaying a fault. (a) The white lines superimposed onto the orthogonal sections identify the location of these sections: a time slice at 1.292 s, an inline section at y = 1602 m, and a crossline section at x = 1868 m. The horizontal scale for the right face is the same as the vertical scale for the upper face. The reference trace is located at x = 1868 m and y = 1602 m. (b) As Figure 4a only after flattening using IRLS method. (c) The result of overlaying two tracked horizons (dashed red) and the automatically generated fault weight (solid red) on the image in Figure 4a.

CONCLUSIONS

We present a method to flatten 3D seismic cubes efficiently and robustly. This method uses an efficient implementation via FFT within a nonlinear iterative scheme. We demonstrate its effectiveness on both synthetic and field data sets with varying degrees of structural complexity.

Data cubes with vertical, laterally limited faults can be flattened by applying a residual weight. This allows horizons to be tracked around the faults. The weight can be created from a previously determined fault model or coherence attribute. The weight merely identifies inaccurate dip values estimated at fault discontinuities so they will be ignored within the inversion. If not provided by a previous fault model, the residual weight can be estimated automatically by extending the basic flattening method to use IRLS. In this case, the flattening method can be thought of as a fault detector that generates the fault picks which best flatten the data. Unfortunately, this IRLS scheme adds significant noise sensitivity and computation time to our basic flattening method described.

The obtained estimate of $\tau\,(x,y,t)$ has many potential uses; for example, our method can easily be adapted to flatten data cubes on one or any particular combination of horizons. This would assist geologists in analyzing thicknesses for rates of deposition and timing of structural events in growth faults. Because the estimate of $\tau\,(x,y,t)$ captures all of the morphology of all horizons in the entire cube, it has significant attribute-analysis potential, particularly for stress related attributes. Furthermore, for faulted data sets that have been reconstructed successfully by this flattening method, the integrated time shifts contain the slip distributions along the fault planes.

Ultimately, the quality of the flattening result depends on the quality of the input dips. Dip estimates can be affected by noise. Using weights can remove the effect of some inaccurate dips; however, there must still be enough good-quality dips to capture the general trends of the structure. Furthermore, crossing events cannot be flattened easily by this method.

We envision a tool that can be implemented quickly and easily on both poststack and prestack data sets, exploiting its superior computational performance. For poststack data, this method can provide an initial picking that interpreters can then adjust. The potential prestack applications include gather alignment and velocity analysis.

ACKNOWLEDGMENT

We thank Chevron and Total for supplying the data and their permission to publish the results. We also acknowledge the financial support of the sponsors of the Stanford Exploration Project. The final version of this paper benefitted tremendously from the comments of associate editor Joe Dellinger, Huub Douma, Peeter Akerberg, Bob Clapp, and anonymous reviewers.

APPENDIX A

EULER-LAGRANGE SOLUTION FOR FLATTENING

The Euler-Lagrange equation (Farlow, 1993) can be applied to find the function $\tau\,(x,y,t)$ that minimizes

$$J(\tau) = \int\int F(x,y,\tau,\tau_x,\tau_y)dxdy \approx 0, \qquad \text{(A-1)}$$

where τ_x and τ_y are the derivatives of τ in the x- and y-directions, respectively, given by

$$\frac{\partial F}{\partial \tau} - \frac{d}{dx}\left[\frac{\partial F}{\partial \tau_x}\right] - \frac{d}{dy}\left[\frac{\partial F}{\partial \tau_y}\right] = 0. \qquad \text{(A-2)}$$

In our application, we have

$$J(\tau) = \int\int \left[\left(b(x,y,\tau) - \frac{\partial \tau}{\partial x}\right)^2 + \left(q(x,y,\tau) - \frac{\partial \tau}{\partial y}\right)^2\right]dxdy, \qquad \text{(A-3)}$$

where b is the dip in the x-direction and q is the dip in the y-direction. Calculating $\frac{\partial F}{\partial \tau}$, $\frac{\partial F}{\partial \tau_x}$, and $\frac{\partial F}{\partial \tau_y}$, we find

$$\frac{\partial F}{\partial \tau} = 2\left[b - \frac{\partial \tau}{\partial x}\right]\frac{\partial b}{\partial \tau} + 2\left[q - \frac{\partial \tau}{\partial y}\right]\frac{\partial q}{\partial \tau}, \qquad \text{(A-4)}$$

$$\frac{\partial F}{\partial \tau_x} = -2\left[b - \frac{\partial \tau}{\partial x}\right], \qquad \text{(A-5)}$$

and

$$\frac{\partial F}{\partial \tau_y} = -2\left[q - \frac{\partial \tau}{\partial y}\right]. \qquad \text{(A-6)}$$

Substituting equations A-4–A-6 into equation A-2 and simplifying, we get

$$\frac{\partial^2 \tau}{\partial x^2} + \frac{\partial^2 \tau}{\partial y^2} = \frac{\partial b}{\partial x} + \frac{\partial q}{\partial y} + \frac{1}{2}\frac{\partial (b^2 + q^2)}{\partial \tau}. \qquad \text{(A-7)}$$

In our method, we ignore the last term of equation A-7 and iteratively solve

$$\frac{\partial^2 \tau}{\partial x^2} + \frac{\partial^2 \tau}{\partial y^2} = \frac{\partial b}{\partial x} + \frac{\partial q}{\partial y}. \qquad \text{(A-8)}$$

We ignore the last term because the first two terms define a gradient direction that we can implement in the Fourier domain efficiently. Without the third term, the method takes more iterations to reach the same answer because the gradient calculation is less accurate. However, because each iteration can be performed much more quickly if the third term is not included, it is more computationally efficient to neglect it. Equation A-8 can be rewritten using the gradient ($\mathbf{\nabla} = [\frac{\partial}{\partial x}\ \frac{\partial}{\partial y}]^T$) and the estimated dip ($\mathbf{p} = [b\ q]^T$) as

$$\mathbf{\nabla}^T\mathbf{\nabla}\tau = \mathbf{\nabla}^T\mathbf{p}. \qquad \text{(A-9)}$$

Therefore, the regression to be solved is

$$\mathbf{\nabla}\tau = \mathbf{p}. \qquad \text{(A-10)}$$

REFERENCES

Bahorich, M., and S. Farmer, 1995, 3-D seismic discontinuity for faults and stratigraphic features: The coherence cube: The Leading Edge, **14**,

1053–1058.
Bienati, N., and U. Spagnolini, 1998, Traveltime picking in 3D data volumes: 60th Annual Meeting, EAGE, Extended Abstracts, Session 01-12.
———, 2001, Multidimensional wavefront estimation from differential delays: IEEE Transactions on Geoscience and Remote Sensing, **39**, 655–664.
Bienati, N., M. Nicoli, and U. Spagnolini, 1999a, Automatic horizon picking algorithms for multidimensional data: 61st Annual Meeting, EAGE, Extended Abstracts, Session 6030.
———, 1999b, Horizon picking for multidimensional data: An integrated approach: 6th International Congress, Brazilian Geophysical Society, Proceedings, SBGf359.
Blinov, A., and M. Petrou, 2005, Reconstruction of 3-D horizons from 3-D data sets: Institute of Electrical and Electronics Engineers, Geoscience and Remote Sensing, **43**, 1421–1431.
Claerbout, J. F., 1992, Earth soundings analysis: Processing versus inversion: Blackwell Scientific Publications.
Claerbout, J. F., and F. Muir, 1973, Robust modeling with erratic data: Geophysics, **38**, 826–844.
Farlow, S. J., 1993, Partial differential equations for scientists and engineers: Dover Publications.
Fomel, S., 2002, Applications of plane-wave destruction filters: Geophysics, **67**, 1946–1960.
Geman, S., and D. McClure, 1987, Statistical methods for tomographic image reconstruction: Bulletin of the International Statistical Institute, L, no. II, 4–5.
Ghiglia, D. C., and M. D. Pritt, 1998, Two-dimensional phase unwrapping: Theory, algorithms, and software: John Wiley & Sons, Inc.
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. F. Claerbout, and J. M. Lomask, 2004, First order interval velocity estimates without picking: 74th Annual International Meeting, SEG, Expanded Abstracts, 2339–2342.
James, H., A. Peloso, and J. Wang, 2002, Volume interpretation of multi-attribute 3D surveys: First Break, **20**, no. 3, 176–180.
Lee, R., 2001, Pitfalls in seismic data flattening: The Leading Edge, **20**, no. 1, 160–164.
Leggett, M., W. A. Sandham, and T. S. Durrani, 1996, 3D horizon tracking using artificial neural networks: First Break, **14**, 413–418.
Lomask, J., and J. Claerbout, 2002, Flattening without picking: Stanford Exploration Project Report, **112**, 141–149.
Lomask, J., 2003a, Flattening 3D seismic cubes without picking: 73rd Annual International Meeting, SEG Expanded Abstracts, 1402–1405.
———, 2003b, Flattening 3-D data cubes in complex geology: Stanford Exploration Project Report, **113**, 249–259.
Stark, T. J., 2004, Relative geologic time (age) volume — Relating every seismic sample to a geologically reasonable horizon: The Leading Edge, **23**, 928–932.
Zeng, H., M. M. Backus, K. T. Barrow, and N. Tyler, 1998, Stratal slicing, part I: Realistic 3D seismic model: Geophysics, **63**, 502–513.