Stanford Exploration Project, Report 124, April 4, 2006, pages 19–24

Short Note

Flattening with cosine transforms

Jesse Lomask and Sergey Fomel

INTRODUCTION

In the Fourier-domain flattening methods presented previously (Lomask and Claerbout, 2002; Lomask, 2003; Lomask et al., 2005) the data has to be mirrored in order to eliminate Fourier artifacts. This means that the data is replicated and reversed so that the boundaries are periodic. This requires four times the memory in 2D and eight times in 3D. In a world where post-stack data cubes can easily be tens of gigabytes in size, efficient memory usage is extremely important.

Here we apply a discrete cosine transform (DCT) approach developed for 2D phase unwrapping (Ghiglia and Romero, 1994; Ghiglia and Pritt, 1998) to our flattening method. As a result, we the reduce memory requirements of the transforms in 2D by a factor of four and in 3D by a factor of eight. We reap an additional factor of two savings from using real instead of complex numbers. Furthermore, the reduction in size significantly reduces computation time as well. We demonstrate its use on a simple 3D synthetic model.

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

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

Note that because the estimated dip $\mathbf{p}(\mathrm{x},\mathrm{y},\boldsymbol{\tau})$ field is a function of the unknown $\boldsymbol{\tau}(x,y,t)$ field, this problem is non-linear and, therefore, difficult to solve directly. We solve this using a Gauss-Newton approach by iterating over equations (2)-(4), i.e.,

iterate {

$$\begin{aligned} \mathbf{r} &= [\nabla \boldsymbol{\tau}_k - \mathbf{p}(\boldsymbol{\tau}_k)] && (2)\\ \Delta\boldsymbol{\tau} &= (\nabla^T\nabla)^{-1}\nabla^T\mathbf{r} && (3)\\ \boldsymbol{\tau}_{k+1} &= \boldsymbol{\tau}_k + \Delta\boldsymbol{\tau} && (4) \end{aligned}$$

} ,

where the subscript k denotes the iteration number.

To greatly improve efficiency, we solve equation (3) in the Fourier domain. We apply the divergence to the estimated dips and divide by the Z-transform of the discretized Laplacian in the Fourier domain as

$$\Delta\boldsymbol{\tau} \approx \mathrm{FFT_{2D}}^{-1}\left[\frac{\mathrm{FFT_{2D}}\left[\nabla^T\mathbf{r}\right]}{-Z_x^{-1}-Z_y^{-1}+4-Z_x-Z_y}\right] \quad (5)$$

where $Z_x = e^{iw\Delta x}$ and $Z_y = e^{iw\Delta y}$. This is extremely efficient compared to iterative methods for solving equation (3).

Unfortunately, mirroring is required to avoid boundary affects in the Fourier domain. Mirroring is concatenating reversed copies of the data along each dimension so that the input to the tranform has periodic boundaries. In 1D the input is increased by a factor of 2, in 2D the input is increased by a factor of four, and so on. In this case, we mirror the divergence of the residual ($\nabla^T\mathbf{r}$). Mirroring is described in more detail by Ghiglia and Pritt (1998) for application to 2D phase unwrapping.

Ghiglia and Pritt (1998) also describe an alternative method using discrete cosine transforms that does not need mirroring. This method does not require mirroring because the cosine transform assumes that image is even (periodic) implicitly. Since it makes this assumption, then only the unmirrored input itself must be transformed. The alternative formulation is

$$\Delta\boldsymbol{\tau} \approx \mathrm{DCT_{2D}}^{-1}\left[\frac{\mathrm{DCT_{2D}}\left[\nabla^T\mathbf{r}\right]}{-2\cos(w\Delta x)-2\cos(w\Delta y)+4}\right]. \quad (6)$$

Computational cost savings

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, in 2D the number of operations per iteration is about $8n(1 + \log(4n_2n_3))$ for the FFT method. For the DCT method, the number of operations per iteration is about $8n + 2n\log(n_2n_3)$. In 3D the FFT method requires about $8n(1+\log(8n))$ operations where as the DCT only requires $8n + 2n\log(n)$. Because there is a certain about of overhead in each iteration common to both algorithms, the actually cost saving is typically a factor of two to four in 2D and three to five in 3D depending on the size of the problem.

The DCT method also has a significant advantage in memory usage as compared to the DCT. Without requiring mirroring, the DCT saves a factor of four in 2D and eight in 3D, however, again there is a certain amount of overhead common to both algorithms. In 2D, the DCT requires about $9n$ memory allocation as opposed to the $23n$ memory of the FFT algorithm. In 3D, the DCT requires still about $9n$ where as the FFT jumps up to $39n$. In summary, the DCT saves greater than a factor of two in 2D and greater than a factor of four in 3D in memory requirements.

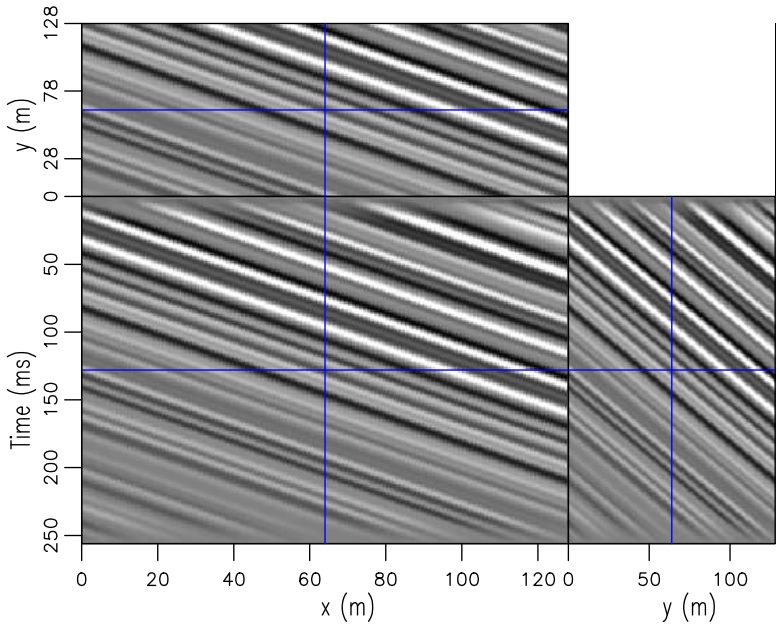

Figure 1: The dipping planes synthetic model. Although a trivial flattening test case, the boundaries of the divergence of the dip are not periodic. jesse3-plane3D [ER]

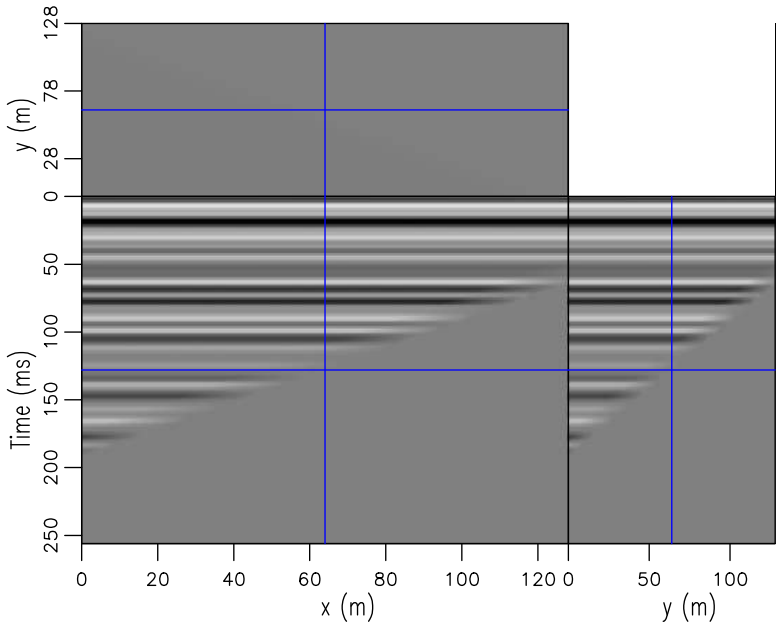

Figure 2: The data in Figure 1 flattened using the FFT method with mirrors. jesse3-plane3D.fft_mirr.flat_cos [ER]

RESULTS

We compared the cosine transform algorithm to the FFT algorithm using the synthetic with simple dipping planes displayed in Figure 1. Although it is just a simple model, it still requires mirroring boundary conditions for the FFT method because the divergence of the dip ($\nabla^T \mathbf{r}$) in equation (5) is not periodic. The result of one iteration of FFT flattening with mirrors is displayed in Figure 2. Notice it is perfectly flat. If we apply the same FFT algorithm without mirrors, we get the result displayed in Figure 3. It is clearly not flat. If we apply the DCT method as shown in Figure 4, it is flattened in one iteration using only a fraction of the memory and computations.

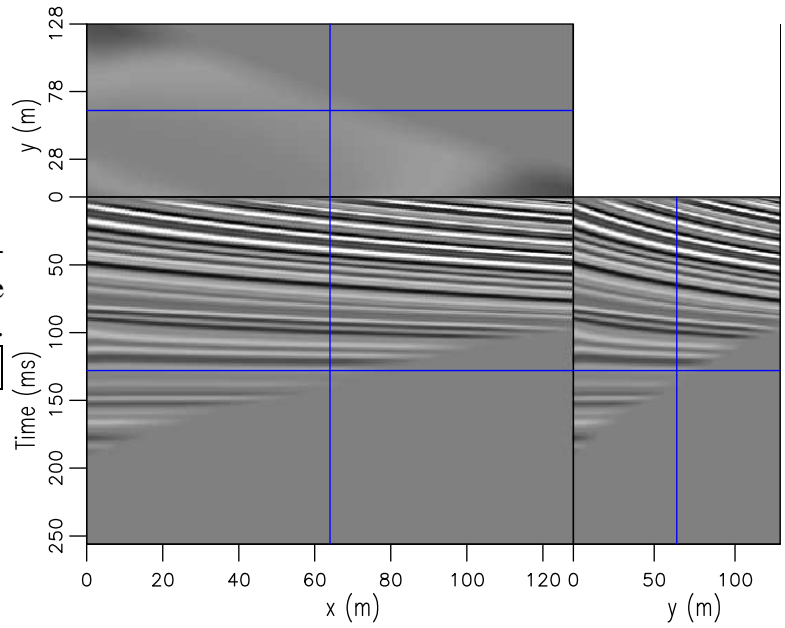

Figure 3: The data in Figure 1 flattened using the FFT method without mirrors. jesse3-plane3D.fft_no_mirr.flat_cos [ER]

CONCLUSION

Here, we have tested a discrete cosine-tranform flattening method. It results in computational cost savings of approximately a factor of two to four in 2D and three to five in 3D as compared to the FFT flattening method. The memory savings are about a factor of two in 2D and four in 3D.

We intend to leverage the computational advantages of this DCT method by incorporating it either as an initial solution or a preconditioner into other flattening methods, possibly including flattening with geological constraints (Lomask and Guitton, 2006).

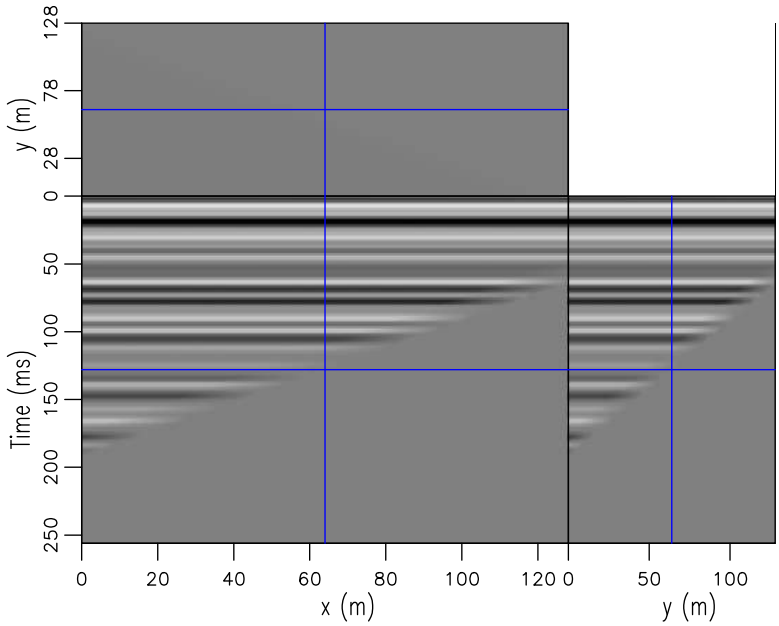

Figure 4: The data in Figure 1 flattened using the DCT method without mirrors. jesse3-plane3D.cos.flat_cos [ER]

ACKNOWLEDGMENT

We would like to thank Daniel Rosales, Gabriel Alvarez, and Bob Clapp for help implementing this. Also would like to thank Peeter Akerberg for useful discussions.

REFERENCES

Ghiglia, D. C. and M. D. Pritt, 1998, Two-dimentional phase unwrapping: Theory, algorithms, and software: John Wiley and 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.

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

Lomask, J. and A. Guitton, 2006, Flattening with geological constraints: SEP–**124**.

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

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