Due Date: 17:00, Thursday, Dec 5, 2011
TA: Yunyue (Elita) Li (yunyue.li@sep.stanford.edu)

Lab 8 - Vesuvio and inversion

Dehuai Peng[1]

ABSTRACT

The Vesuvio satellite data set provides two measurements for the mountain: amplitude and phase. A straightforward computation of phase gives results with jumps of 2π. In this exercise, you will unwrap the phase and experiment with different inversion schemes.

VESUVIUS PHASE UNWRAPPING

Figure 1 shows radar images of Mt. Vesuvius[2] in Italy. These images are made from backscatter signals $s_1(t)$ and $s_2(t)$, recorded along two satellite orbits 800 km high and 54 m apart. The signals are very high frequency (the radar wavelength being 2.7 cm). They were Fourier transformed and one multiplied by the complex conjugate of the other, getting the product $Z = S_1(\omega)\bar{S}_2(\omega)$. The product's amplitude and phase are shown in Figure 1. Examining the data, you can notice that where the signals are strongest (darkest on the left), the phase (on the right) is the most spatially consistent.

To reduce the time needed for analysis and printing, we reduced the data size two different ways, by decimation and by local averaging, as shown in Figure 2. The decimation was to about 1 part in 9 on each axis, and the local averaging was done in 9×9 windows giving the same spatial resolution in each case. The local averaging was done independently in the plane of the real part and the plane of the imaginary part. Putting them back together again showed that the phase angle of the averaged data behaves much more consistently.

Phase is a troublesome measurement because we generally see it modulo 2π. Marching up the mountain we see the phase getting lighter and lighter until it suddenly jumps to black which then continues to lighten as we continue up the mountain to the next jump. Let us undertake to compute the phase including all of its jumps of

[1]**e-mail:** dhpeng@prc.cn

[2]A web search engine quickly finds you other views.

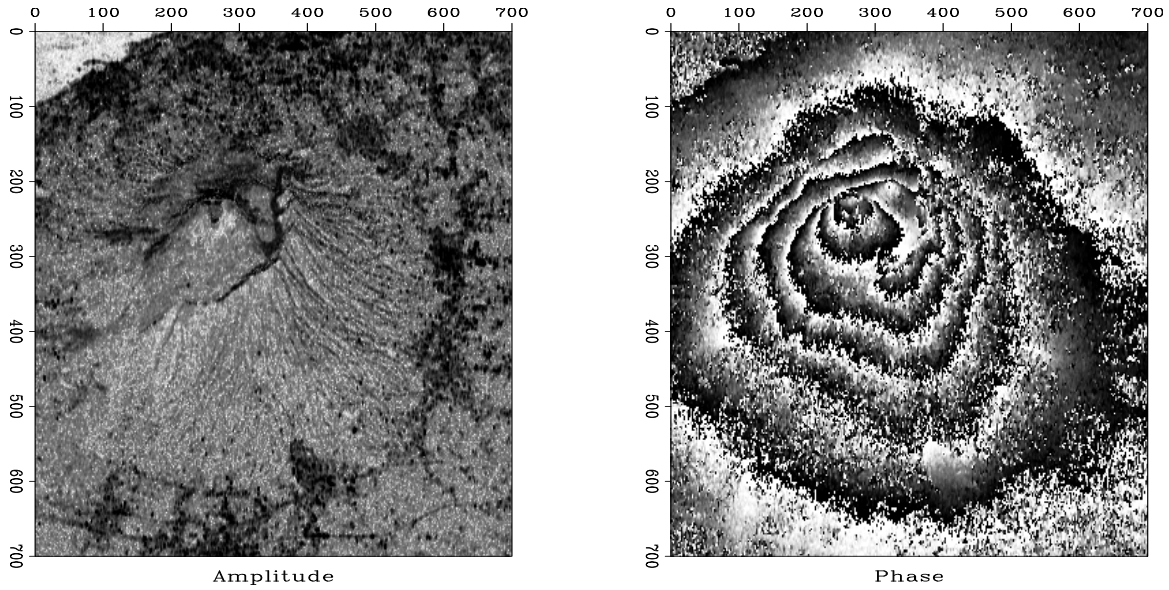


Figure 1: Radar image of Mt. Vesuvius. Left is the amplitude $|Z(x,y)|$. Non-reflecting ocean in upper left corner. Right is the phase $\arctan(\Re Z(x,y), \Im Z(x,y))$. (Umberto Spagnolini)

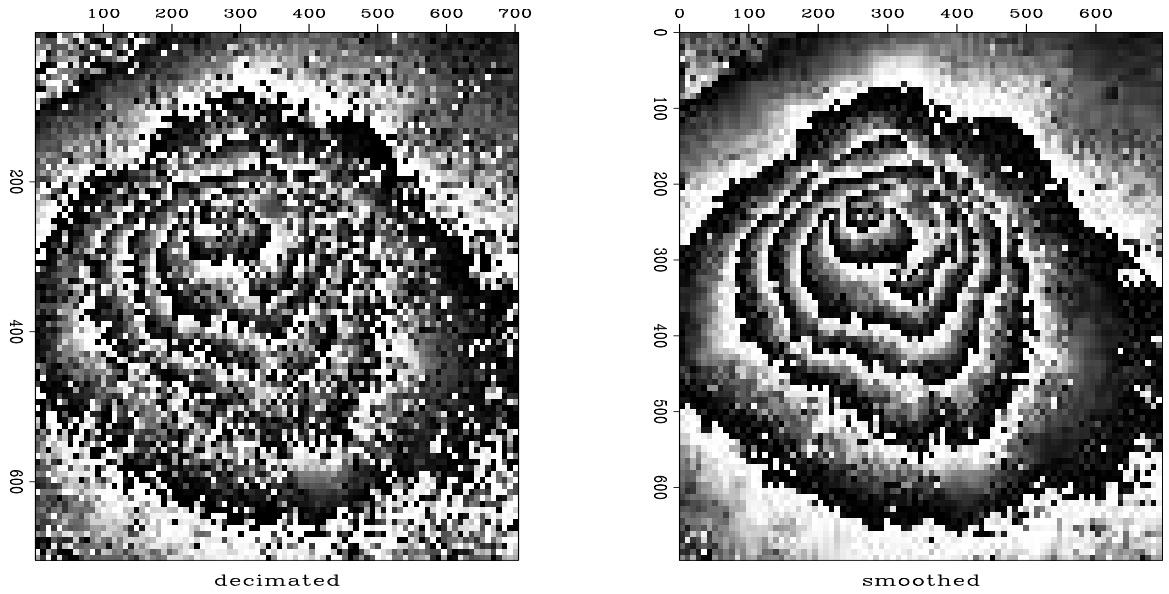


Figure 2: Phase based on decimated data (left) and smoothed data (right).

2π. Begin with a complex number Z representing the complex-valued image at any location in the (x, y)-plane.

$$\begin{aligned} re^{i\phi} &= Z && (1)\\ \ln|r| + i\phi &= \ln Z && (2)\\ \phi(x,y) &= \Im \ln Z(x,y) \;+\; 2\pi N(x,y) && (3) \end{aligned}$$

A computer will find the imaginary part of the logarithm with the arctan function of two arguments, `atan2(y,x)`, which will put the phase in the range $-\pi < \phi \leq \pi$ although any multiple of 2π could be added. We seem to escape the $2\pi N$ phase ambiguity by differentiating:

$$\begin{aligned} \frac{\partial \phi}{\partial x} &= \Im \frac{1}{Z}\frac{\partial Z}{\partial x} && (4)\\ \frac{\partial \phi}{\partial x} &= \frac{\Im \bar{Z}\frac{\partial Z}{\partial x}}{\bar{Z}Z} && (5) \end{aligned}$$

For every point on the y-axis, equation (5) is a differential equation on the x-axis, and we could integrate them all to find $\phi(x, y)$. That sounds easy. On the other hand, the same equations are valid when x and y are interchanged, so we get twice as many equations as unknowns. For ideal data, either of these sets of equations should be equivalent to the other, but for real data we expect to be fitting the fitting goal

$$\nabla\phi \quad \approx \quad \frac{\Im \bar{Z}\nabla Z}{\bar{Z}Z} \tag{6}$$

where $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$. This is essentially the same application we solved flattening seismic data with the regression $\nabla\tau \approx \mathbf{d}$. Taking measurements to be phase differences between neighboring mesh points, it is more correct to interpret equation 6 as a difference equation than a differential equation. Since we measure phase differences only over tiny distances (one pixel) we hope not to worry about phases greater than 2π. But if such jumps do occur, they will contribute to overall error.

Let us consider a typical location in the (x, y) plane where the complex numbers $Z_{i,j}$ are given. Define a shorthand a, b, c, and d as follows:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} Z_{i,j} & Z_{i,j+1} \\ Z_{i+1,j} & Z_{i+1,j+1} \end{bmatrix} \tag{7}$$

With this shorthand, the difference equation representation of the fitting goal (6) is:

$$\begin{aligned} \phi_{i+1,j} - \phi_{i,j} &\approx \Delta\phi_{ac} \\ \phi_{i,j+1} - \phi_{i,j} &\approx \Delta\phi_{ab} \end{aligned} \tag{8}$$

Now let us find the phase jumps between the various locations. Complex numbers a and b may be expressed in polar form, say $a = r_a e^{i\phi_a}$ and $b = r_b e^{i\phi_b}$. The complex

number $\bar{a}b = r_a r_b e^{i(\phi_b - \phi_a)}$ has the desired phase $\Delta\phi_{ab}$. To obtain it we take the imaginary part of the complex logarithm $\ln|r_a r_b| + i\Delta\phi_{ab}$.

$$\begin{array}{rcccl} \phi_b - \phi_a & = & \Delta\phi_{ab} & = & \Im \ln \bar{a}b \\ \phi_d - \phi_c & = & \Delta\phi_{cd} & = & \Im \ln \bar{c}d \\ \phi_c - \phi_a & = & \Delta\phi_{ac} & = & \Im \ln \bar{a}c \\ \phi_d - \phi_b & = & \Delta\phi_{bd} & = & \Im \ln \bar{b}d \end{array} \tag{9}$$

which gives the information needed to fill in the right-hand side of (8), as done by subroutine `igrad2init()` from module `unwrap`.

ESTIMATING THE INVERSE GRADIENT

To optimize the fitting goal (8), module `unwrap()` uses three different methods: conjugate gradient, steepest descent and a new cheapo conjugate gradient method. You are all familar with conjugate gradient, so let's introduce cheapo conjugate gradient method here.

Recall that in steepest descent, we do a line search to choose the step length α, by minimizing the data residual in the data space: $\mathbf{r}_{\text{new}} \cdot \mathbf{r}_{\text{new}}$. Then the steepest descent step length α is:

$$\alpha = -\frac{(\mathbf{r} \cdot \Delta\mathbf{r})}{(\Delta\mathbf{r} \cdot \Delta\mathbf{r})} \tag{10}$$

Instead of minimizing the residual in the data space, we can also minimize it in the model space with $\mathbf{q}_{\text{new}} = \mathbf{F^T}\mathbf{r}_{\text{new}}$: $\mathbf{q}_{\text{new}} \cdot \mathbf{q}_{\text{new}}$.

$$\begin{array}{rcl} \mathbf{F^T}(\mathbf{r} + \alpha\Delta\mathbf{r}) \cdot \mathbf{F^T}(\mathbf{r} + \alpha\Delta\mathbf{r}) & = & (\mathbf{F^T}\mathbf{r} + \alpha\mathbf{F^T}\Delta\mathbf{r}) \cdot (\mathbf{F^T}\mathbf{r} + \alpha\mathbf{F^T}\Delta\mathbf{r}) \\ & = & (\Delta\mathbf{x} + \alpha\Delta\mathbf{q}) \cdot (\Delta\mathbf{x} + \alpha\Delta\mathbf{q}). \end{array} \tag{11}$$

Follow the same procedure, we can solve the required value of α:

$$\alpha = -\frac{(\Delta\mathbf{x} \cdot \Delta\mathbf{q})}{(\Delta\mathbf{q} \cdot \Delta\mathbf{q})} \tag{12}$$

YOUR ASSIGNMENT

If you have not already, log into one of the SEP workstations using your class account. Type netscape at the command prompt to bring up a web browser. Visit the GEE class webpage (http://sepwww.stanford.edu/class/211/), then find and download the source code for this lab, Lab6.tar.gz. Save this file in your home directory, then type tar xzvf Lab8.tar.gz; then type chmod go-rwx Lab8 to protect your work.

1. We can compare the convergence of steepest descent, cheapo cg, and conjugate gradient on a simple 2 variable problem. Figure 3 shows the results. You can reproduce this figure by running the MATLAB file: teststep.m in your Src folder. Comment on the search direction and the convergence of three methods.

Your answer:

2. Comment on the convergence when you change the initial postion $a0$ and the operator A. Include new figures if necessary.

Your answer:

3. We are going to finish the rest of this Lab using Fortran 90. Below is the new template of pseudocode for the new iterative method. Fill in your pseudocode for cheapocg.

$\mathbf{r} \leftarrow \mathbf{Fx} - \mathbf{d}$
iterate {
$\Delta\mathbf{x} \leftarrow \mathbf{F^T}\ \mathbf{r}$
$\Delta\mathbf{r} \leftarrow \mathbf{F}\ \Delta\mathbf{x}$

!!! Fill in your pseudocode for cheapo cg
!!! Hint: use the intermediate variable q

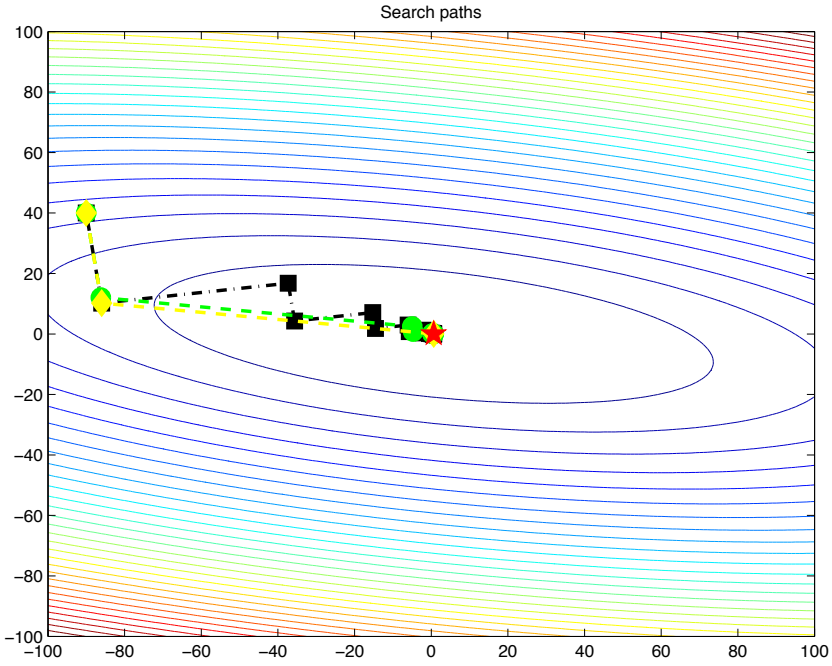
$\mathbf{x} \leftarrow \mathbf{x} + \alpha\Delta\mathbf{x}$
$\mathbf{r} \leftarrow \mathbf{r} + \alpha\Delta\mathbf{r}$
}

4. In your Src folder, you will see a new solver: solver_cheap_smp.f90, which implements the steepest descent and cheapo cg method. Open and read it. Make sure you understand how steepest descent is realized. Make comments where helpful.

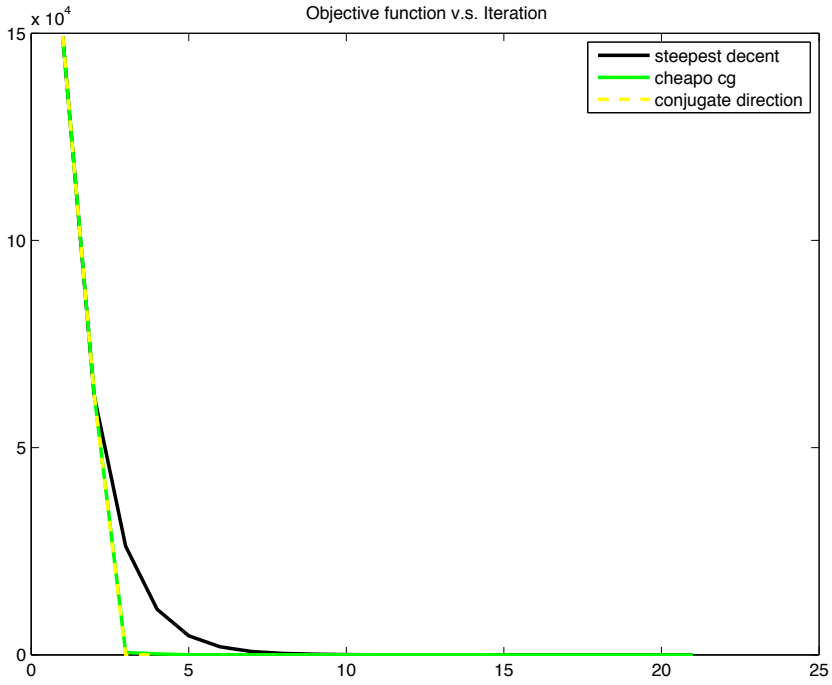
5. Open file cheapocg.f90. This program does one step of line search for steepest descent and cheapo cg. Implement where you compute your step length for cheapo cg.

6. Get back to solver_cheap_smp.f90. Make necessary changes to implement a cheapo cg stepping. The pseudocode you complete above is very helpful.

7. In unwrap.f90, complete the solver call to unwrap the phase.



(a)



(b)

Figure 3: (a) Search paths for three different inversion method. Red star denotes the exact solution; (b) Objective function curve for three different inversion method. In both plots, black line: steepest descent, green line: cheapocg, yellow line: conjugate direction.

8. After you complete all the coding, type make compstep.view to see the unwrap results. You should be seeing similar results as Figure 2.12 in your book. Change the number of iterations if necessary. Comment on your results.

Your answer:

9. **E**XTRA CREDIT Plot the residual as a function of iteration for different inversion scheme. Comment on the convergence.

Your answer:

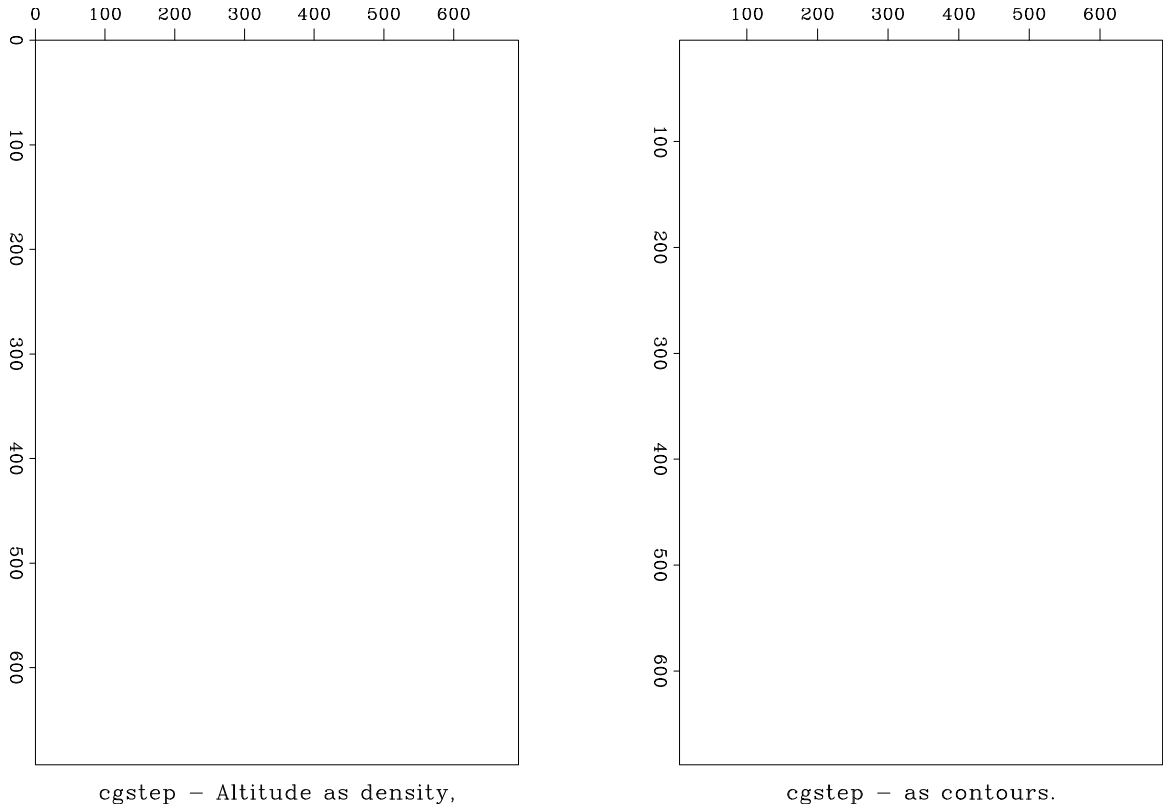


Figure 4: Unwrapped by conjugated grdient.

DONE

When you are all done, type `make build` to produce pdf figures. Then type `scons` to build the paper. Clean up your directory by typing `make clean`.

REFERENCES

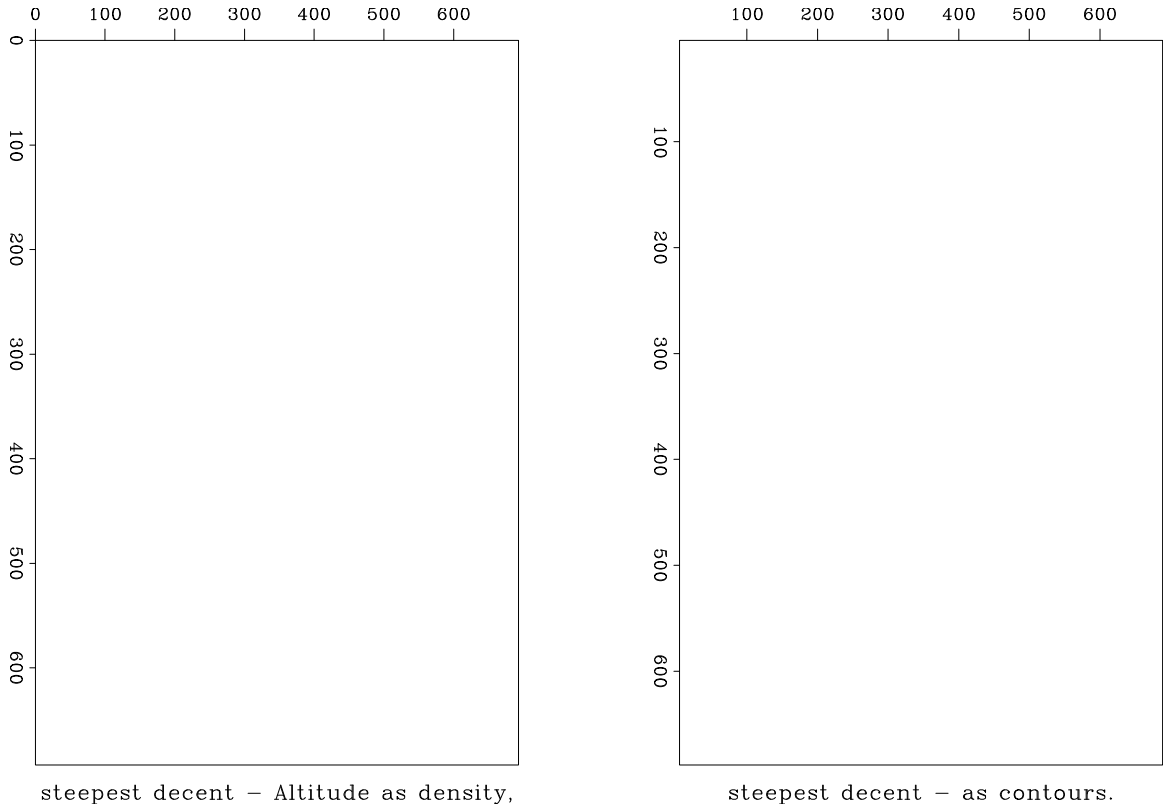


Figure 5: Unwrapped by steepest descent.

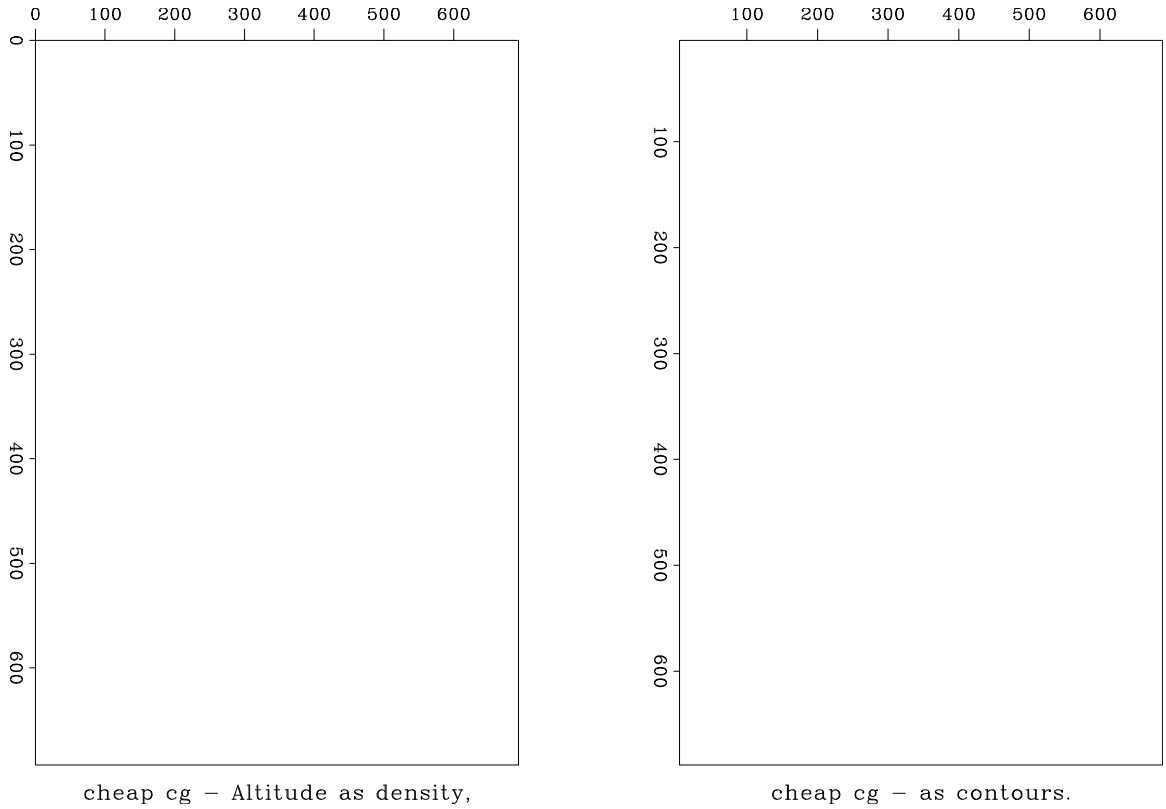


Figure 6: Unwrapped by cheapo cg.