GEO 365N/384S Seismic Data Processing
Computational Assignment 4

Team: Longhorns

ABSTRACT

This assignment consists of four parts:

1. Processing Alaska data through surface-consistent amplitude correction, velocity analysis, and stack.
2. Processing Viking Graben data through deconvolution and surface-consistent amplitude correction.
3. Processing Teapot Dome data through surface-consistent amplitude correction.
4. Processing your own data.

PREREQUISITES

Completing the computational part of this homework assignment requires

- `Madagascar` software environment available from `http://www.ahay.org/`
- LaTeX environment with SEGTeX available from `http://www.ahay.org/wiki/SEGTeX`

To do the assignment on your personal computer, you need to install the required environments.

To setup the Madagascar environment in the JGB 3.216B computer lab, run the following commands:

```
module load madagascar-devel
source $RSFROOT/share/madagascar/etc/env.csh
setenv DATAPATH $HOME/data/
setenv RSFBOOK $HOME/data/book
setenv RSFFIGS $HOME/data/figs
```

You can put these commands in your `$HOME/.cshrc` file to run them automatically at login time.

To setup the LaTeX environment, run the following commands:

```
cd
git clone https://github.com/SEGTeX/texmf.git
texhash
```

You only need to do it once.

The homework code is available from the class repository by running

```
svn co https://github.com/TCCS-BEG/geo384s/trunk/hw4
```

You can also download it from your team's private repository.

GENERATING THIS DOCUMENT

At any point of doing this computational assignment, you can regenerate this document and display it on your screen.

1. Change directory to `hw4`:

   ```
   cd hw4
   ```

2. Run

   ```
   sftour scons lock
   scons read &
   ```

As the first step, open `hw4/paper.tex` file in your favorite editor and edit the first line to enter the name of your team. Then run `scons read` again.

ALASKA DATA

In the first part of the assignment, we will pick up processing of the Alaska line where he last left it, after the groundroll attenuation step. Figure 1 shows the shot gathers after groundroll removal by the *Emc-Hammer* team.

In this assignment, you will process the data further by applying surface-consistent amplitude balancing, velocity analysis, and stack.

1. Change directory to `hw4/alaska`.

2. Run

   ```
   scons -c
   ```

   to remove (clean) previously generated files.

3. Run

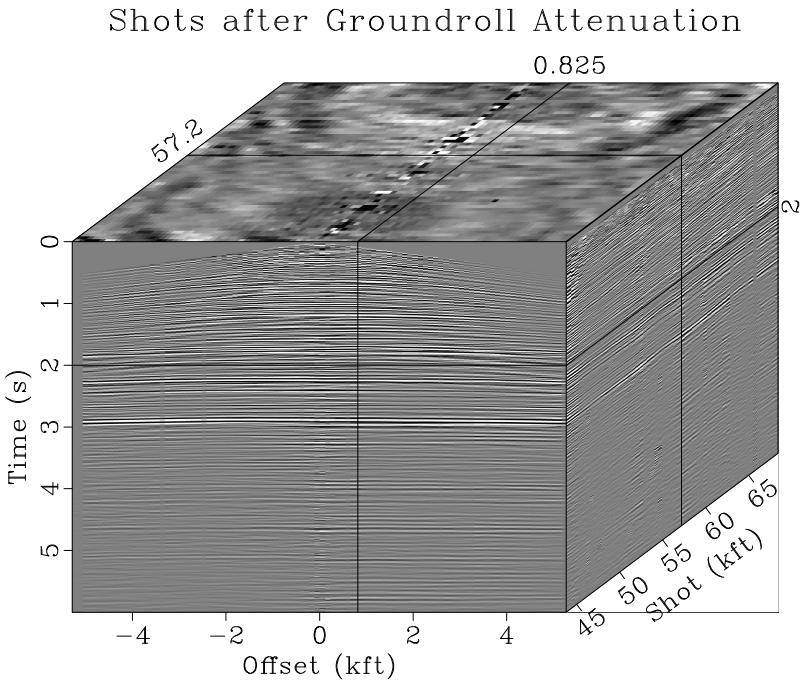


Figure 1: Alaska shot gathers after groundroll removal by *Emc-Hammer*. alaska/ rshots

```
scons arms.view
```

to display trace amplitudes for Alaska shot gathers (Figure 2a). The amplitudes show a strong offset trend. To display them after removing the trend (Figure 2b), run

```
scons arms2.view
```

Do you notice stripes in the amplitude going in different directions? These stripes might be caused by near-surface conditions in deploying sources and receivers. The surface-consistent model (Taner and Koehler, 1981) tries to explain the trace amplitude as a function of source and receiver locations $A(s,r)$ using a product of several factors (source, receiver, offset, and midpoint):

$$A(s,r) \approx A_s(s)\, A_r(r)\, A_x(r-s)\, A_m\left(\frac{r+s}{2}\right) \ . \tag{1}$$

By applying logarithm to the amplitude, we can turn the product in equation (1) into a sum

$$\log\left[A(s,r)\right] \approx L_s(s) + L_r(r) + L_x(r-s) + L_m\left(\frac{r+s}{2}\right) \ . \tag{2}$$

4. In its discrete version, equation (2) represents a system of linear equations with given $\log\left[A(s,r)\right]$ and unknown factors L_s, L_r, L_x, and L_m.

   Does this system have more equations than unknowns or more unknowns than equations?

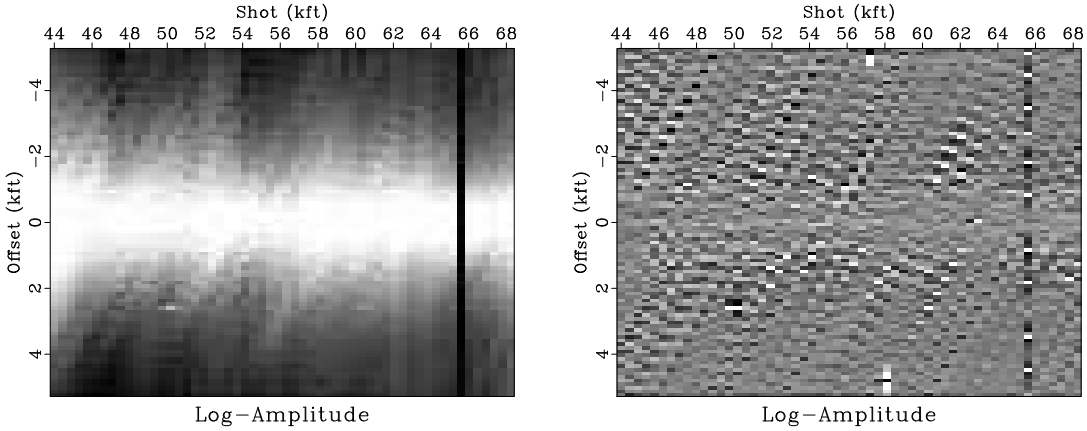


Figure 2: Trace amplitudes before (a) and after (b) removing the long-period offset trend. alaska/ arms,arms2

Answer:

Does this system have a unique solution?

Answer:

We will attempt to solve the system represented by equation (2) using the the least-squares method and the iterative conjugate-gradient algorithm (Hestenes and Stiefel, 1952; Fletcher and Reeves, 1964). The algorithm requires implementing the linear operator and its adjoint. The implementation of the linear operator from equation (2) is provided in the `surface-consistent.c` program.

alaska/surface-consistent.c

```
/* Surface-consistent decomposition */
#include <rsf.h>

int main(int argc, char* argv[])
{
    bool adj, verb;
    int nd, nm, nx, im, min, max, id, i, ix, sx;
    int **indx, *size;
    float *model, *data;
    sf_file inp, index, out;

    sf_init(argc,argv);

    if (!sf_getbool("adj",&adj)) adj=true;
    /* adjoint flag */
    if (!sf_getbool("verb",&verb)) verb=false;
```

```
17      /* verbosity flag */
18
19      inp = sf_input("in");
20      if (SF_FLOAT != sf_gettype(inp))
21          sf_error("Need float input");
22
23      out = sf_output("out");
24
25      index = sf_input("index");
26      if (SF_INT != sf_gettype(index))
27          sf_error("Need int index");
28
29      if (!sf_histint(index,"n1",&nd))
30          sf_error("No n1= in index");
31      nm = sf_leftsize(index,1);
32
33      if (adj) {
34          if (nd != sf_filesize(inp))
35              sf_error("Wrong data size");
36      } else {
37          sf_putint(out,"n1",nd);
38      }
39
40      data = sf_floatalloc(nd);
41      indx = sf_intalloc2(nd,nm);
42      size = sf_intalloc(nm);
43
44      sf_intread(indx[0],nd*nm,index);
45
46      nx = 0;
47      for (im=0; im < nm; im++) {
48          min = max = indx[im][0];
49          for (id=1; id < nd; id++) {
50              i = indx[im][id];
51              if (i < min) min=i;
52              if (i > max) max=i;
53          }
54          if (min) {
55              for (id=0; id < nd; id++) {
56                  indx[im][id] -= min;
57              }
58          }
59          size[im]=max-min+1;
60          nx += size[im];
61          if (verb) sf_warning("size%d=%d",im+1,size[im]);
62      }
63
64      if (adj) {
```

```
65          sf_putint(out,"n1",nx);
66      } else {
67          if (nx != sf_filesize(inp))
68              sf_error("Wrong model size");
69      }
70
71      model = sf_floatalloc(nx);
72
73      if (adj) {
74          sf_floatread(data,nd,inp);
75          for (ix=0; ix < nx; ix++) {
76              model[ix] = 0.0f;
77          }
78      } else {
79          sf_floatread(model,nx,inp);
80          for (id=0; id < nd; id++) {
81              data[id] = 0.0f;
82          }
83      }
84
85      sx=0;
86      for (im=0; im < nm; im++) {
87          for (id=0; id < nd; id++) {
88              ix = indx[im][id]+sx;
89
90              if (adj) {
91                  model[ix] += data[id];
92              } else {
93                  data[id] += model[ix];
94              }
95          }
96          sx += size[im];
97      }
98
99      if (adj) {
100         sf_floatwrite(model,nx,out);
101     } else {
102         sf_floatwrite(data,nd,out);
103     }
104
105     exit(0);
106 }
```

To run the iterative inversion and to see the result of matching the data (logarithm of the amplitude), run

```
scons scarms.view
```

Does it manage to predict the stripes seen in Figure 2b?

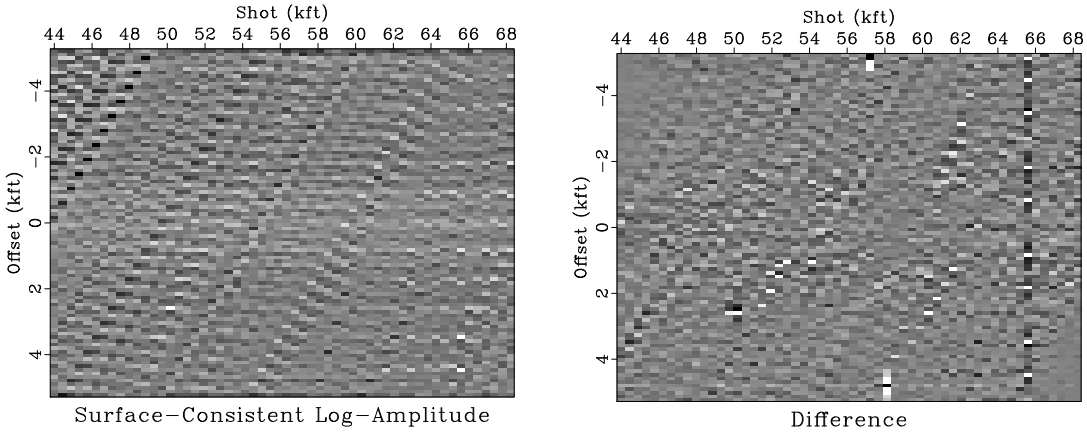


Figure 3: (a) Estimated surface-consistent trace amplitudes. (b) Difference with Figure 2b. alaska/ scarms,adiff

The individual factors (composing the estimated model vector) are shown in Figure 4.

To display the shot gathers after the surface-consistent amplitude correction (Figure 5), run

```
scons ashots.view
```

To compare the output with the input, run

```
sfpen Fig/rshots.vpl Fig/ashots.vpl
```

5. After the correction, we can proceed with the data processing sequence by doing the familiar steps of sorting into CMP gathers, velocity analysis, NMO, and stack. Run

   ```
   scons stack0.view
   ```

   to go through these steps. The resultant stack (Figure 6a) appears to contain spiky noise. We can try to remove it by median filtering (Figure 6b)

   ```
   scons stack1.view
   ```

   The NMO velocity extracted by the automatic picker is shown in Figure 7, and the velocity analysis and NMO applied to an individual CMP gather are shown in Figure 8, displayed with

   ```
   scons nmo2.view
   ```

6. Your task:

   (a) By modifying the `SConstruct` file, extract velocity analysis and NMO from several other CMP gathers and examine the results.

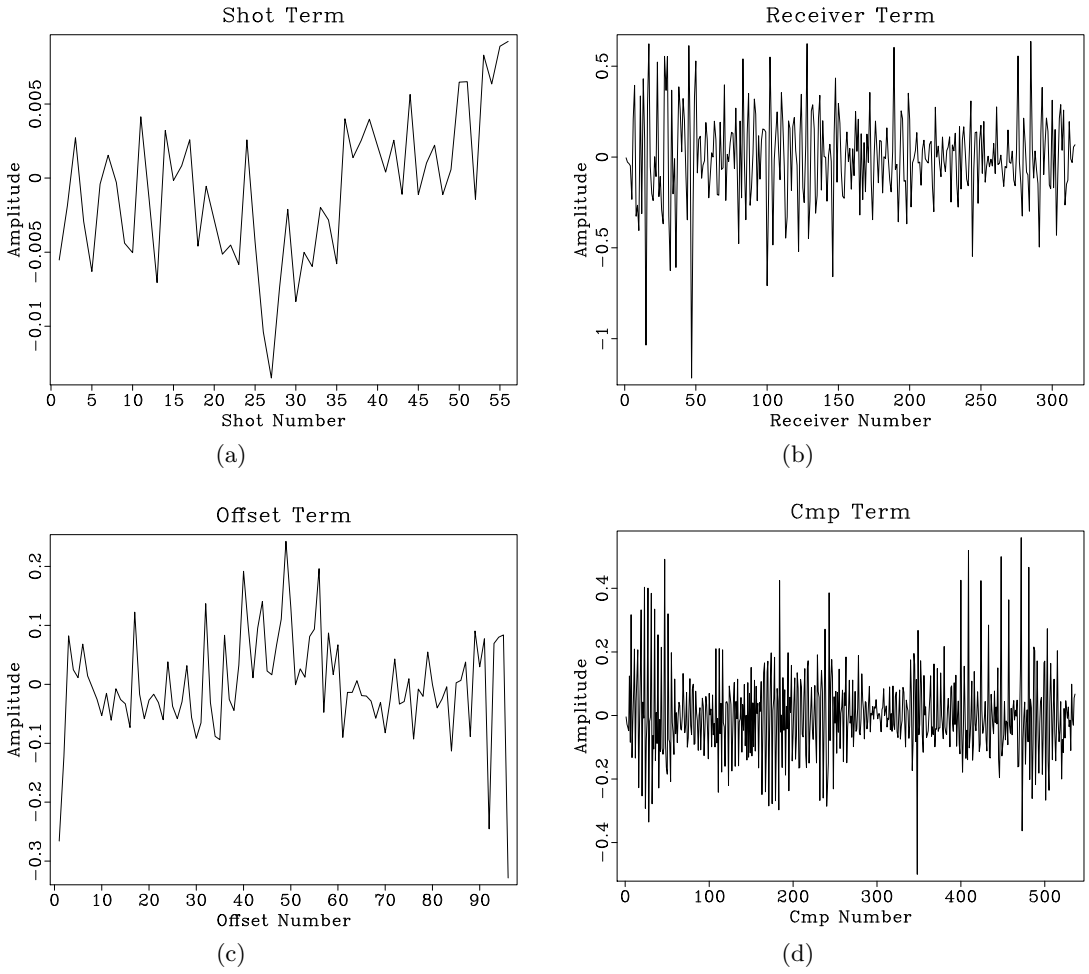


Figure 4: Estimated surface-consistent factors. alaska/ shot,receiver,offset,cmp

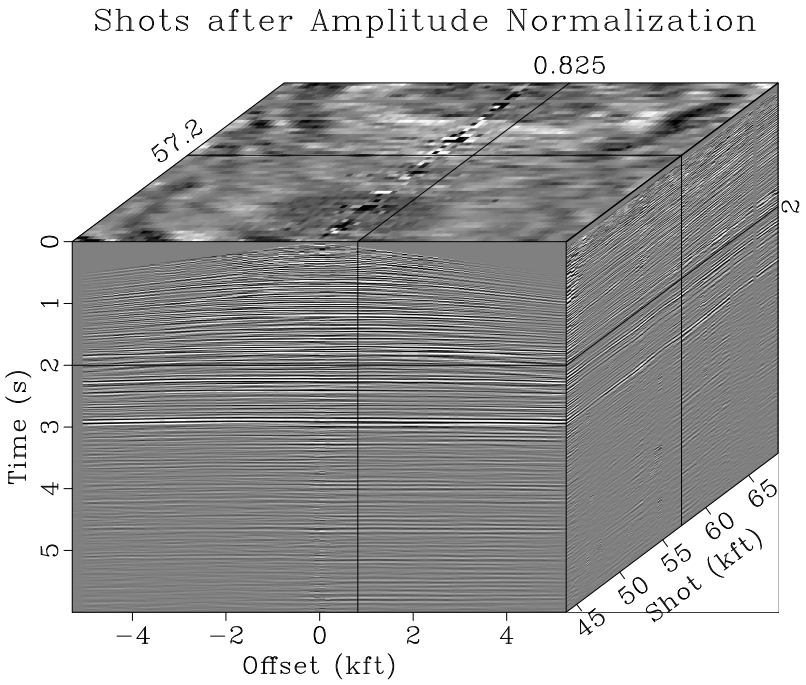
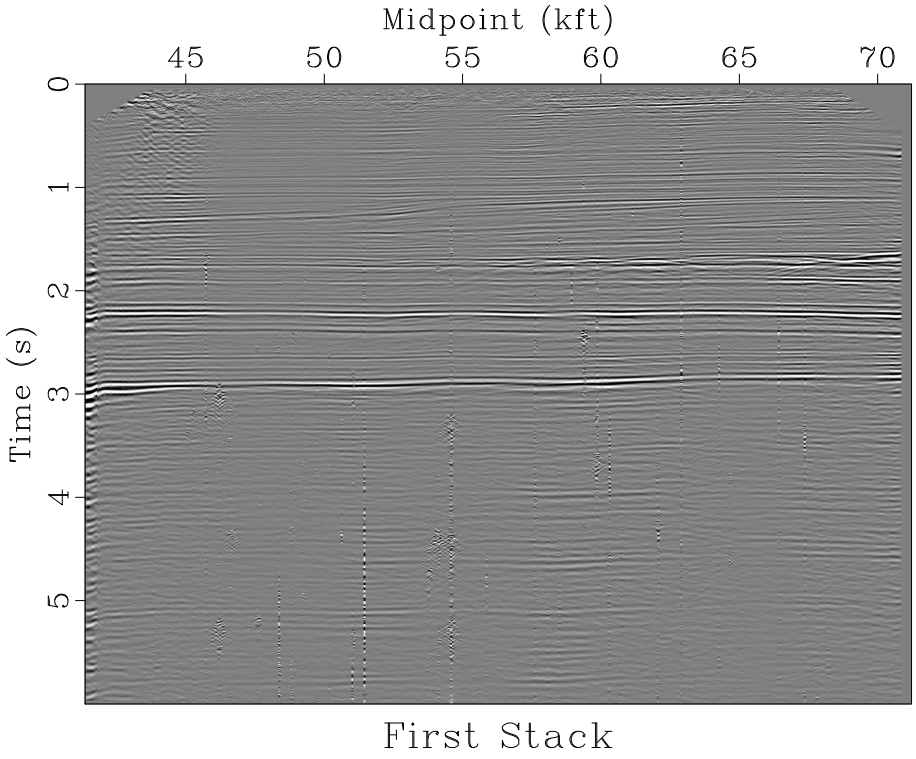


Figure 5: Alaska shot gathers after surface-consistent amplitude balancing. alaska/ ashots

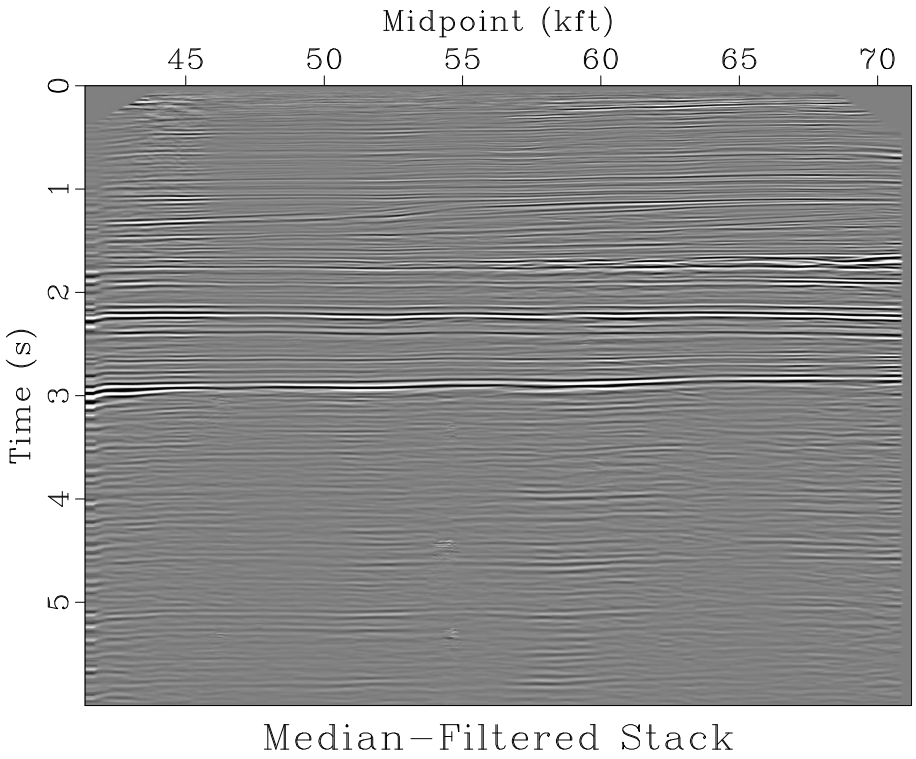
(b) Try to improve the quality of the stack. To achieve a better result, you are free to use any of the techniques that you learned previously.

alaska/SConstruct

```
from rsf.proj import *

# Shots after ground-roll removal by Emc-Hammer
Fetch('rshots.HH','alaska')
Flow('rshots','rshots.HH','dd form=native')

def plotshots(title):
    return '''
    pow pow1=1.5 | byte gainpanel=all pclip=90 |
    grey3 frame1=1000 frame2=55 frame3=30 title="%s"
    flat=n point1=0.7 point2=0.7
    ''' % title

Result('rshots',plotshots('Shots after Groundroll Attenuation'))

# Average trace amplitude
Flow('arms','rshots',
     'mul $SOURCE | stack axis=1 | math output="log(input)" ')
Result('arms','grey title=Log-Amplitude mean=y pclip=90')
```



(a)



(b)

Figure 6: (a) First attempt at NMO stack. (b) NMO stack after despiking by median filtering. alaska/ stack0,stack1

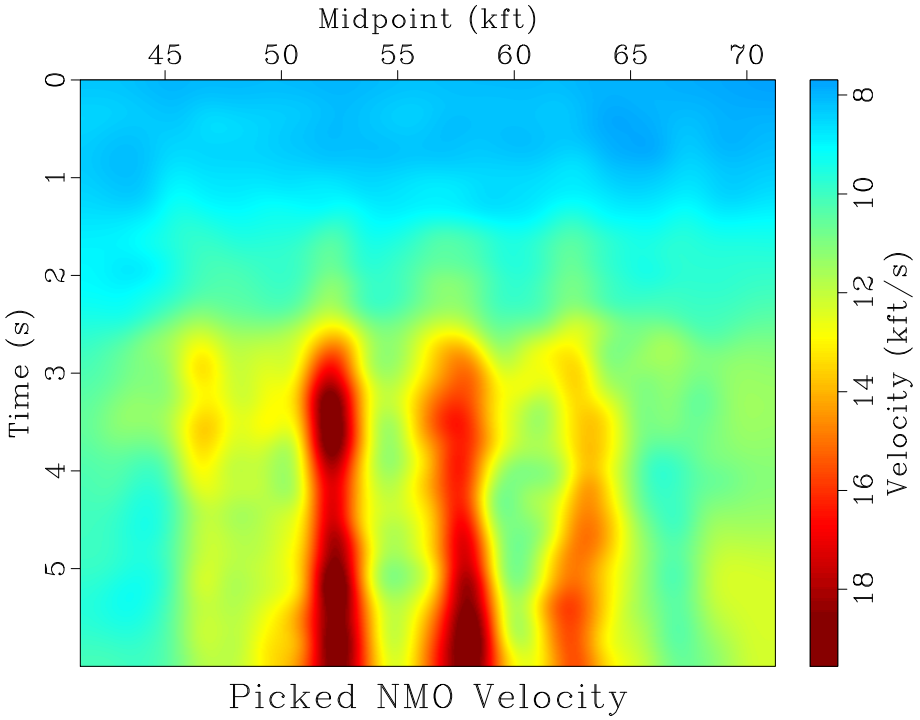


Figure 7: NMO velocity estimated by the automatic picker. alaska/ vpicks

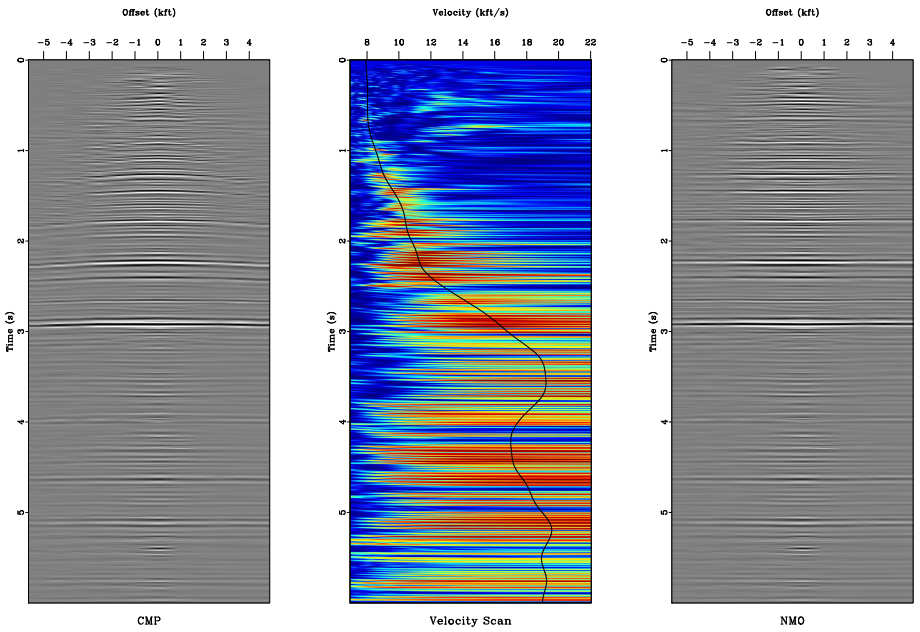


Figure 8: Velocity analysis and NMO applied to a selected CMP gather. alaska/ nmo2

```
20
21 # Remove long-period offset term
22 Flow('arms2','arms','smooth rect1=5 | add scale=-1,1 $SOURCE')
23 Result('arms2','grey title=Log-Amplitude clip=1.13')
24
25 # Integer indeces for different terms
26 Flow('shot','arms2','math output="(x2-44)/0.44" ')
27 Flow('offset','arms2','math output="(x1+5.225)/0.11" ')
28 Flow('receiver','arms2','math output="(x1+x2-44+5.225)/0.11" ')
29 Flow('cmp','arms2','math output="(x1/2+x2-44+5.225/2)*2/0.11" ')
30
31 nx = 96 # number of offsets
32 ns = 56 # number of shots
33 nt = nx*ns # number of traces
34
35 Flow('index','shot offset receiver cmp',
36      '''
37      cat axis=3 ${SOURCES[1:4]} | dd type=int |
38      put n1=%d n2=4 n3=1
39      ''' % nt)
40
41 # Transform from 2-D to 1-D
42 Flow('arms1','arms2','put n2=1 n1=%d' % nt)
43
44 prog = Program('surface-consistent.c')
45 sc = str(prog[0])
46
47 Flow('model',['arms1','index',sc],
48      './${SOURCES[2]} index=${SOURCES[1]} verb=y')
49
50 # Least-squares inversion by conjugate gradients
51 Flow('sc',['arms1','index',sc,'model'],
52      '''
53      conjgrad ./${SOURCES[2]} index=${SOURCES[1]}
54      mod=${SOURCES[3]} niter=30
55      ''')
56
57 Flow('scarms',['sc','index',sc],
58      '''
59      ./${SOURCES[2]} index=${SOURCES[1]} adj=n |
60      put n1=%d n2=%d
61      ''' % (nx,ns))
62 Result('scarms',
63        '''
64        grey mean=y title="Surface-Consistent Log-Amplitude"
65        clip=1.13
66        ''')
67
```

```
Flow('adiff','arms2 scarms','add scale=1,-1 ${SOURCES[1]}')
Result('adiff','grey title=Difference clip=1.13')

size=dict(shot=ns,offset=nx,receiver=316,cmp=536)

f1 = 0
for case in ('shot','offset','receiver','cmp'):
    n1=size[case]

    Result(case,'sc',
           '''
           window n1=%d f1=%d | put o1=1 d1=1 |
           graph title="%s Term"
           label1="%s Number" unit1= label2=Amplitude unit2=
           ''' % (n1,f1,case.capitalize(),case.capitalize()))
    f1 += n1

Flow('ampl','scarms',
     'math output="exp(-input/2)" | spray axis=1 n=3000 d=0.002 o=0')
Flow('ashots','rshots ampl','mul ${SOURCES[1]} | cut n3=1 f3=49')

Result('ashots',plotshots('Shots after Amplitude Normalization'))

# Sort to CMPs

Flow('cmps mask','ashots',
     'shot2cmp half=n mask=${TARGETS[1]} | pow pow1=2')
Flow('mask1','mask','spray axis=1 n=1')

# Select one CMP

Flow('cmp1','cmps','window n3=1 f3=200')
Plot('cmp1','grey title=CMP')

Flow('vscan1','cmp1','vscan semblance=y half=n nv=151 v0=7 dv=0.1')
Plot('vscan1','grey color=j allpos=y title="Velocity Scan" ')

Flow('vpick1','vscan1','pick rect1=100')
Plot('vpick1',
     '''
     graph plotcol=7 plotfat=5 wantaxis=n wanttitle=n
     yreverse=y transp=y min2=7 max2=22 pad=n
     ''')
Plot('vscanpick1','vscan1 vpick1','Overlay')

Flow('nmo1','cmp1 vpick1','nmo velocity=${SOURCES[1]} half=n')
Plot('nmo1','grey title=NMO')

```

```
Result('nmo','cmp1 vscanpick1 nmo1','SideBySideAniso')

# Apply to all CMPs

Flow('vscans','cmps mask1',
     '''
     vscan semblance=y half=n nv=151 v0=7 dv=0.1
     mask=${SOURCES[1]}
     ''',split=[3,'omp'])
Flow('vpicks','vscans','pick rect1=100 rect2=20')

Result('vpicks',
       '''
       grey color=j scalebar=y barreverse=y mean=y
       title="Picked NMO Velocity"
       ''')

Flow('nmos','cmps vpicks','nmo velocity=${SOURCES[1]} half=n')

# Examine one CMP

cmp = 200 # !!! MODIFY ME !!!

Flow('vscan2','vscans','window n3=1 f3=%d' % cmp)
Plot('vscan2','grey color=j allpos=y title="Velocity Scan" ')

Flow('vpick2','vpicks','window n2=1 f2=%d' % cmp)
Plot('vpick2',
     '''
     graph plotcol=7 plotfat=5 wantaxis=n wanttitle=n
     yreverse=y transp=y min2=7 max2=22 pad=n
     ''')
Plot('vscanpick2','vscan2 vpick2','Overlay')

Flow('nmo2','nmos','window n3=1 f3=%d' % cmp)
Plot('nmo2','grey title=NMO')

Result('nmo2','cmp1 vscanpick2 nmo2','SideBySideAniso')

# Stack

Flow('stack0','nmos','stack')
Result('stack0','grey title="First Stack" ')

Flow('stack1','stack0','despike2 wide2=10')
Result('stack1','grey title="Median-Filtered Stack" ')

End()
```

VIKING GRABEN DATA

In the next part of the assignment, we will take a step back in processing the Viking Graben dataset and use the original unprocessed data.

1. Change directory to `hw4/viking`.
2. Run

   ```
   scons -c
   ```

   to remove (clean) previously generated files.
3. In addition to seismic data, Mobil Oil provided a far-field wavelet recorded in the water. To display this wavelet, run

   ```
   scons wavelet.view
   ```

   As typical for marine data, the wavelet, instead of being a perfect impulse, contains additional components (bubble pulse, ghost reflection, etc.) The wavelet spectrum, displayed with

   ```
   scons spectrum.view
   ```

   shows frequencies missing from the ideal flat response.

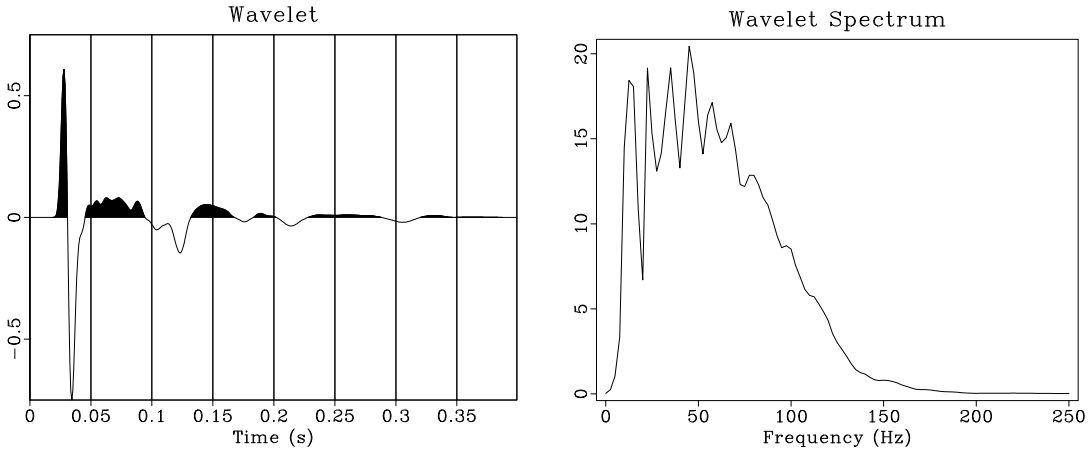


Figure 9: Far-field wavelet (a) and its Fourier spectrum (b).

4. The task of *deconvolution* is to collapse the wavelet to something closer to a pulse by convolving it with a filter (Webster, 1978; Robinson and Osman, 1996). The filter estimation is another classic application of the least-squares inversion approach. The *prediction-error filter* (PEF) is defined by auto-regression: the filter coefficients represent weights, which combine with shifted version of the input signal to predict the signal itself:

$$-d_t \approx \sum_{i=1}^{N} f_i \, d_{t-i} \; . \tag{3}$$

In equation (3), d_t is the given data, and f_i for $i = 1, 2, \ldots, N$ are free coefficients of the prediction-error filter.

As in the surface-consistent problem, the least-squares solution can be found using a generic conjugate-gradient program, which simply requires an implementation of the convolution operator in the right-hand side of equation (3) and its adjoint. An implementation is provided in the `convolve.c` program.

viking/convolve.c

```
/* Match filtering */
#include <rsf.h>

int main(int argc, char* argv[])
{
    bool adj;
    int n1, n2, i1, i2, i, j, nf;
    float *data, *target, *filter;
    sf_file inp, out, other;

    sf_init(argc,argv);
    inp = sf_input("in");
    out = sf_output("out");
    other = sf_input("data");

    if (!sf_getbool("adj",&adj)) adj=false;
    /* adjoint flag */

    if (adj) {
        /* input data, output filter */
        if (!sf_histint(inp,"n1",&n1)) sf_error("No n1=");
        if (!sf_histint(inp,"n2",&n2)) sf_error("No n2=");
        if (!sf_getint("nf",&nf)) sf_error("Need nf=");
        /* filter size */

        sf_putint(out,"n1",nf);
        sf_putint(out,"n2",1);
    } else {
        /* input filter, output data */
        if (!sf_histint(inp,"n1",&nf)) sf_error("No n1=");
        if (!sf_histint(other,"n1",&n1)) sf_error("No n1=");
```

```
32          if (!sf_histint(other,"n2",&n2)) sf_error("No n2=");
33
34          sf_fileflush(out,other);  /* copy data dimensions */
35      }
36
37      filter = sf_floatalloc(nf);
38      target = sf_floatalloc(n1);
39      data = sf_floatalloc(n1);
40
41      if (adj) {
42          for (i=0; i < nf; i++) filter[i]=0.0f;
43      } else {
44          sf_floatread(filter,nf,inp);
45      }
46
47      for (i2=0; i2 < n2; i2++) {
48          sf_floatread(data,n1,other);
49
50          if (adj) {
51              sf_floatread(target,n1,inp);
52          } else {
53              for (i1=0; i1 < n1; i1++) target[i1] = 0.0f;
54          }
55
56          for (i1=0; i1 < n1; i1++) {
57              for (i=0; i < nf; i++) {
58
59                  j=i1-i-1;
60
61                  /* zero value boundary conditions */
62                  if (j < 0 || j >= n1) continue;
63
64                  if (adj) {
65                      filter[i] += data[j]*target[i1];
66                  } else {
67                      target[i1] += data[j]*filter[i];
68                  }
69              }
70          }
71
72          if (!adj) sf_floatwrite(target,n1,out);
73      }
74
75      if (adj) sf_floatwrite(filter,nf,out);
76
77      exit(0);
78  }
```

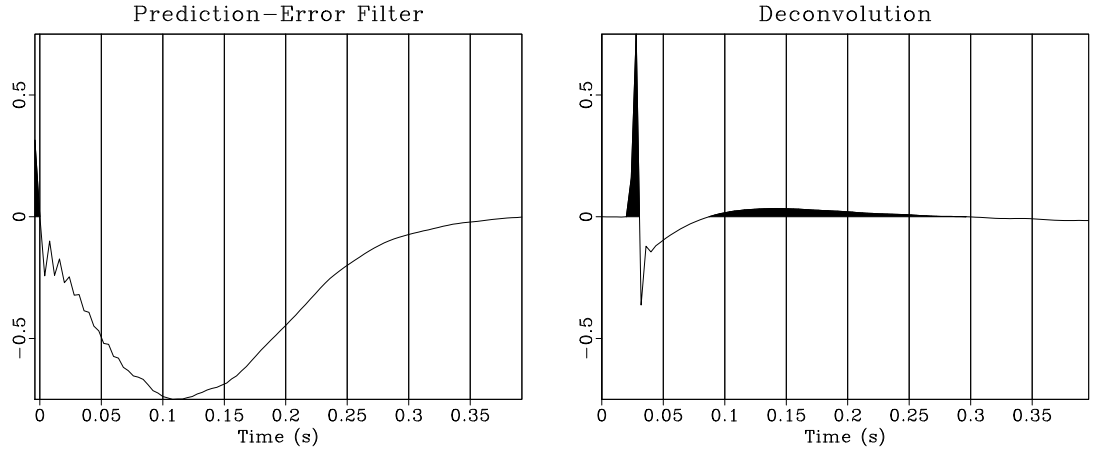
To test the program using the generic dot-product test, you can run

```
scons filter0.rsf wavelet4.rsf
scons convolve.exe
sfdottest ./convolve.exe nf=100 mod=filter0.rsf dat=wavelet4.rsf
```

The test should show pairs of numbers matching to several significant digits.

To run the least-squares inversion and to estimate the filter, run

```
scons filter.view
```



(a) (b)

Figure 10: Estimated prediction-error filter (a) and the result of wavelet deconvolution (b).

Although deconvolution applied to the wavelet (Figure 10b) does not produce a perfect spike, it improves the spikiness of the wavelet and its spectral content.

5. Now we can apply the estimated filter to the data. Run

   ```
   scons viking1000.view
   ```

   to extract and display the first 1,000 traces from the Viking Graben dataset (Figure 11.) Run

   ```
   scons decon1000.view
   ```

   to display the result of deconvolution (Figure 12.)

   To apply deconvolution to the whole dataset, run

   ```
   scons decon.rsf
   ```

6. (**EXTRA CREDIT**) For an extra credit, you can write a program or a script, which takes the data as the main input and the filter as an auxiliary input and applies deconvolution. With your new program, you can replace the lines

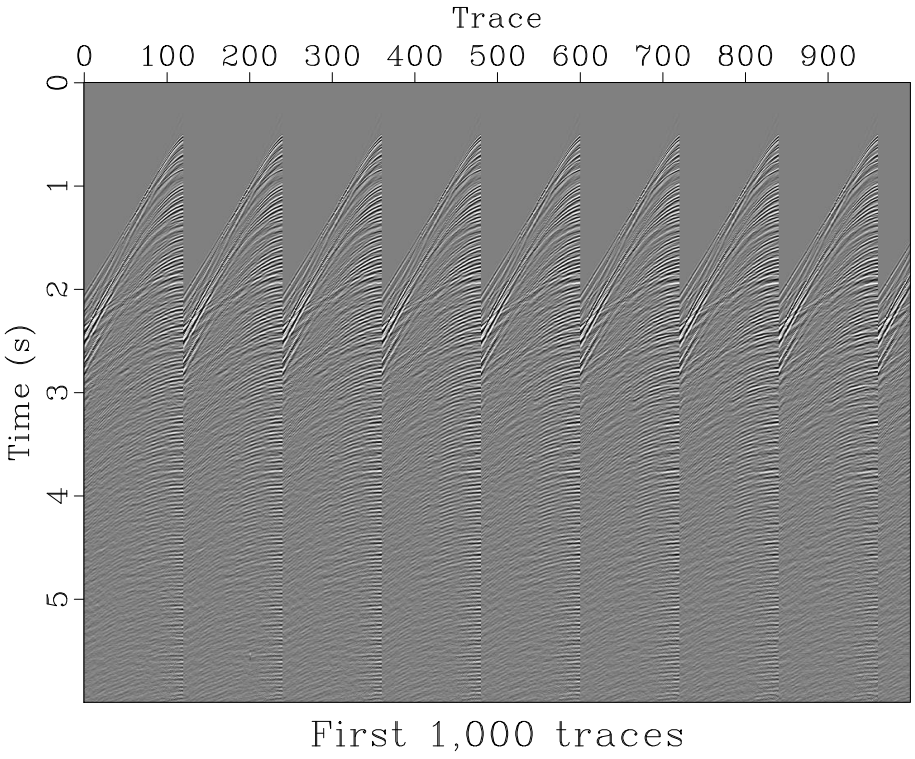


Figure 11: First 1,000 traces from the Viking Graben dataset.

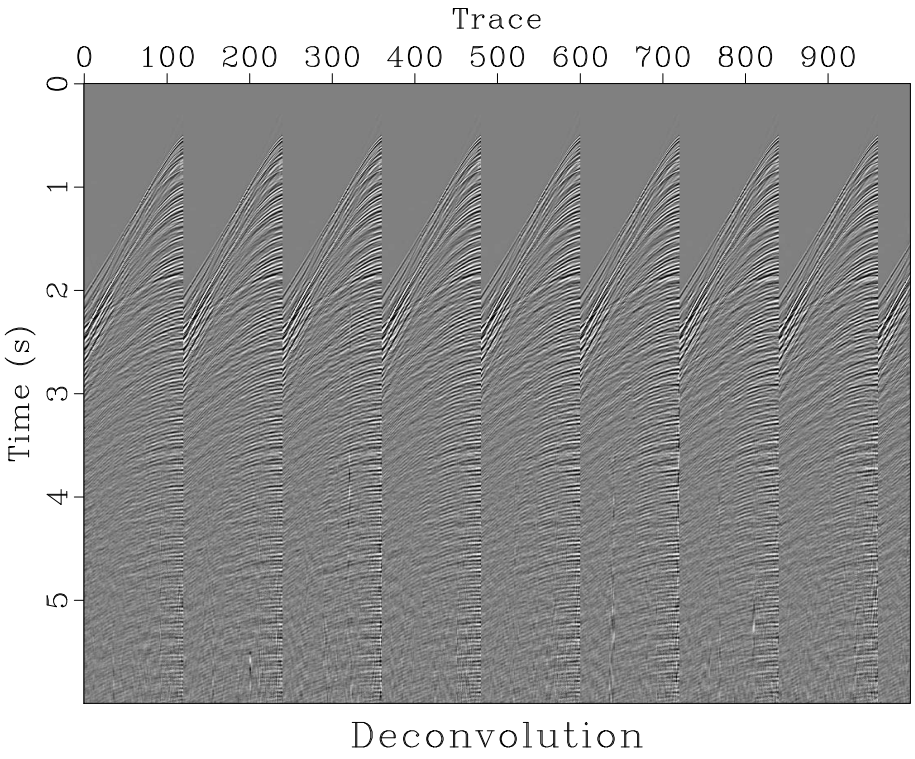


Figure 12: Viking Graben traces after deconvolution.

```
# Process all traces
Flow('decon',['filter','viking',convolve],
     '''
     ./\${SOURCES[2]} data=i\${SOURCES[1]} adj=n |
     add \${SOURCES[1]} scale=-1,1
     ''')
```

with something like

```
# Process all traces
Flow('decon',['viking','filter',decon],
     '''
     ./\${SOURCES[2]} filter=\${SOURCES[1]}
     ''',split=[2,'omp',[0]])
```

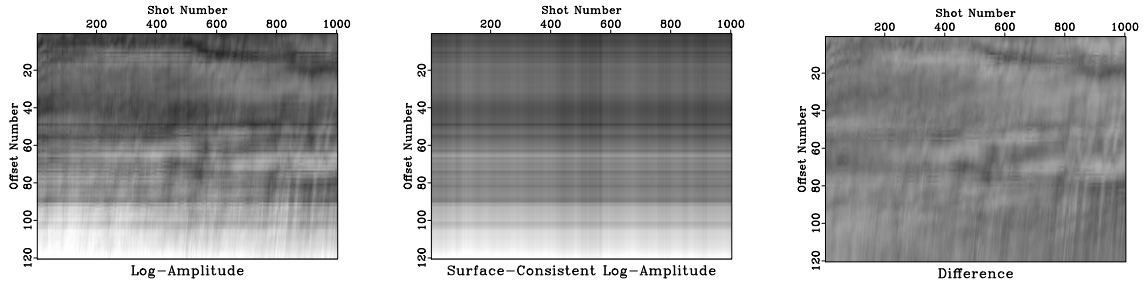
and process the data quickly in parallel.

7. After deconvolution, we can proceed to surface-consistent amplitude correction, similar to the processing applied previously to the Alaska data. The trace amplitudes are shown in Figure 13a. Their surface-consistent estimate, using only the shot term L_s and the offset term L_x is shown in Figure 13b. The difference is shown in Figure 13c. To display the latter figure, run To apply deconvolution to the whole dataset, run

   ```
   scons vadiff.view
   ```

   Can you notice remaining stripes in the difference? Which of the four terms in equation (2) are the stripes coming from?

   Answer:



(a) (b) (c)

Figure 13: Trace amplitudes from the Viking Graben dataset. (a) Initial. (b) Estimated surface-consistent. (c) Difference.

8. Your task: modify the `SConstruct` file to improve the result by introducing additional terms to the fitting equation.

viking/SConstruct

```
from rsf.proj import *
```

```
2
3  # Download Viking Graben data
4  #Fetch('seismic.segy','viking')
5  Fetch('seismic.segy','VikingGrabbenLine12',
6        top='/home/p1/seismic_datasets/SeismicProcessingClass',
7        server='local')
8
9  # Convert from SEGY to RSF
10 Flow('viking tviking viking.asc viking.bin','seismic.segy',
11       '''
12       segyread tfile=${TARGETS[1]}
13       hfile=${TARGETS[2]} bfile=${TARGETS[3]}
14       ''')
15
16 # Far-field wavelet
17 Fetch('FarField.dat','Mobil_Avo_Viking_Graben_Line_12',
18       top='open.source.geoscience/open_data',
19       server='http://s3.amazonaws.com')
20
21 # Convert from ASCII to RSF
22 Flow('wavelet','FarField.dat',
23       '''
24       echo in=$SOURCE data_format=ascii_float n1=500 o1=0 d1=0.0008
25       label1=Time unit1=s n2=1 | dd form=native
26       ''')
27 Result('wavelet','wiggle poly=y title=Wavelet pclip=100')
28
29 Result('spectrum','wavelet',
30        '''
31        spectra | window max1=250 |
32        graph title="Wavelet Spectrum"
33        ''')
34
35 # Subsample to data sampling
36 Flow('wavelet4','wavelet','window j1=5 | pad n1=1500')
37
38 prog = Program('convolve.c')
39 convolve = str(prog[0])
40
41 # Estimate PEF by iterative least-squares inversion
42 Flow('filter0',None,'spike n1=100 k1=1')
43 Flow('filter',['wavelet4',convolve,'filter0'],
44       '''
45       conjgrad ./${SOURCES[1]} nf=100 data=${SOURCES[0]}
46       niter=100 mod=${SOURCES[2]}
47       ''')
48
49 Result('filter','filter filter0',
```

```
50          '''
51          pad beg1=1 | window n1=100 |
52          scale axis=-1 | add ${SOURCES[1]} |
53          wiggle poly=y title="Prediction-Error Filter" pclip=100
54          ''')
55
56  # Wavelet deconvolution
57  Flow('wdecon',['filter','wavelet4',convolve],
58       '''
59       ./${SOURCES[2]} data=${SOURCES[1]} adj=n |
60       add ${SOURCES[1]} scale=-1,1 | window n1=100
61       ''')
62  Result('wdecon','wiggle poly=y title=Deconvolution pclip=100')
63
64  # Apply to the first 1000 traces
65  Flow('viking1000','viking','window n2=1000')
66  Result('viking1000','pow pow1=2 | grey title="First 1,000 traces" ')
67
68  Flow('decon1000',['filter','viking1000',convolve],
69       '''
70       ./${SOURCES[2]} data=${SOURCES[1]} adj=n |
71       add ${SOURCES[1]} scale=-1,1
72       ''')
73  Result('decon1000',
74         'pow pow1=2 | grey title=Deconvolution ')
75
76  # Process all traces
77  Flow('decon',['filter','viking',convolve],
78       '''
79       ./${SOURCES[2]} data=${SOURCES[1]} adj=n |
80       add ${SOURCES[1]} scale=-1,1
81       ''')
82
83  # Average trace amplitude
84  Flow('arms','decon',
85       'mul $SOURCE | stack axis=1 | math output="log(input)" ')
86
87  # shot/offset indeces: fldr and tracf
88  Flow('index','tviking','window n1=2 f1=2 | transp')
89
90  def plot(title,bias=5):
91      return '''
92      spray axis=1 n=1 |
93      intbin head=${SOURCES[1]} yk=fldr xk=tracf | window |
94      grey title="%s" label2="Shot Number" unit2=
95      label1="Offset Number" unit1= bias=%d clip=3
96      ''' % (title,bias)
97
```

```
# Display in shot/offset coordinates
Result('varms','arms tviking',plot('Log-Amplitude'))

prog = Program('../alaska/surface-consistent.c')
sc = str(prog[0])

Flow('model',['arms','index',sc],
     '${SOURCES[2]} index=${SOURCES[1]} verb=y')

Flow('sc',['arms','index',sc,'model'],
     '''
     conjgrad ${SOURCES[2]} index=${SOURCES[1]}
     mod=${SOURCES[3]} niter=10
     ''')

Result('vshot','sc',
       '''
       window n1=1000 | put o1=1 d1=1 |
       graph title="Shot Term"
       label1="Shot Number" unit1= label2=Amplitude unit2=
       ''')
Result('voffset','sc',
       '''
       window f1=1000 n1=120 | put o1=1 d1=1 |
       graph title="Offset Term"
       label1="Offset Number" unit1= label2=Amplitude unit2=
       ''')

Flow('scarms',['sc','index',sc],
     '${SOURCES[2]} index=${SOURCES[1]} adj=n')
Result('vscarms','scarms tviking',
       plot('Surface-Consistent Log-Amplitude'))

Flow('adiff','arms scarms','add scale=1,-1 ${SOURCES[1]}')
Result('vadiff','adiff tviking',plot('Difference',0))

End()
```

TEAPOT DOME DATA

Next, we return to processing the Teapot Dome data. Our task is to estimate the surface-consistent amplitude normalization, similar to how it was done with the two previous datasets.

1. Change directory to `hw4/teapot`.
2. Run

```
scons -c
```

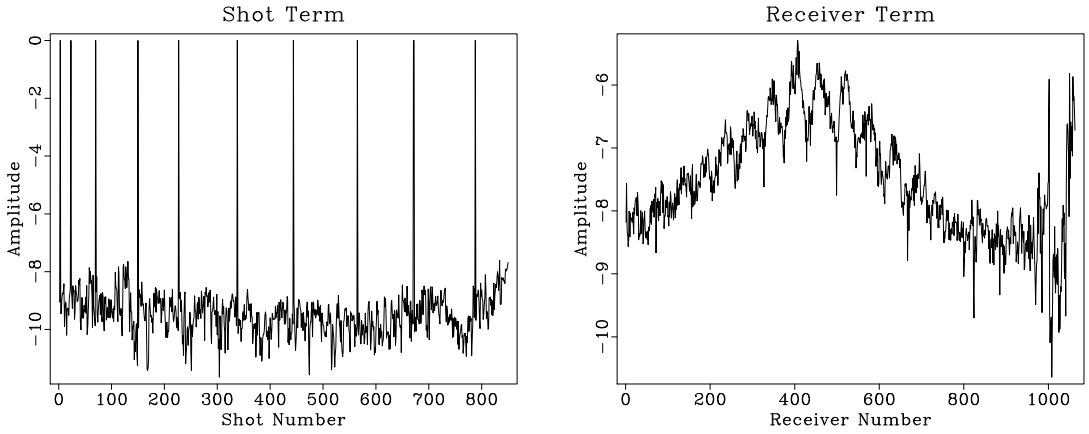
to remove (clean) previously generated files.

3. The process should look familiar by now. Figure 14 shows the two amplitude factors (L_s and L_r) estimated by least-squares inversion.

4. To visualize how the amplitude correction depends on the surface coordinates, we can display it by interpolating to a regular grid from the shot locations. To display the result (Figure 15), run

```
scons sint.view
```

5. Your task: modify the `SConstruct` file to interpolate in receiver coordinates (`gx` and `gy`) instead of shot coordinates (`sx` and `sy`). Do you notice any interesting differences?



(a) (b)

Figure 14: Estimated shot and receiver surface-consistent amplitude terms for the Teapot Dome dataset.

teapot/SConstruct

```
from rsf.proj import *

# Download Teapot Dome field data
#Fetch('npr3_field.sgy','teapot',
#      server='http://s3.amazonaws.com',top='')
Fetch('npr3_field.sgy','TeapotDome3D',
      top='/home/p1/seismic_datasets/SeismicProcessingClass',
      server='local')

# Convert to RSF
Flow('traces header header.asc','npr3_field.sgy',
     'segyread tfile=${TARGETS[1]} hfile=${TARGETS[2]}')

```

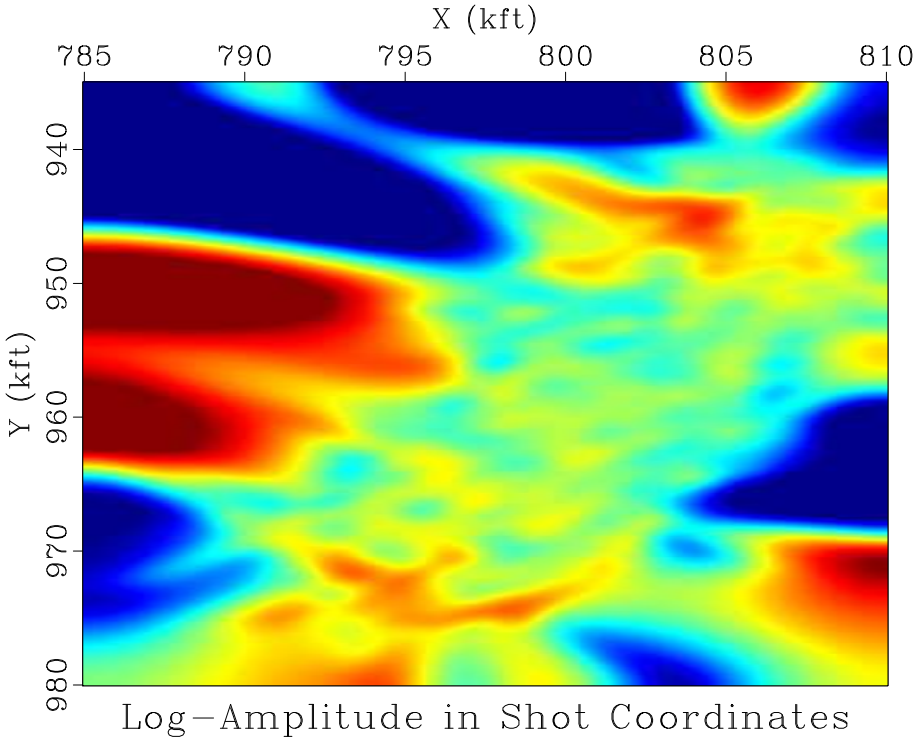


Figure 15: Surface-consistent amplitude interpolated using the shot coordinates.

```
# Seismic data corresponds to trid=1
Flow('trid','header','headermath output=trid | mask min=1 max=1')

# Average trace amplitude
Flow('arms','traces trid',
     '''
     mul $SOURCE | headerwindow mask=${SOURCES[1]} |
     stack axis=1 | math output="log(input)"
     ''')
Flow('theader','header trid','headerwindow mask=${SOURCES[1]}')

# shot/receiver indeces: fldr and tracf
Flow('index','theader','window n1=2 f1=2 | transp')

prog = Program('../alaska/surface-consistent.c')
sc = str(prog[0])

Flow('model',['arms','index',sc],
     '${SOURCES[2]} index=${SOURCES[1]} verb=y')

Flow('sc',['arms','index',sc,'model'],
     '''
     conjgrad ${SOURCES[2]} index=${SOURCES[1]}
```

```
37      mod=${SOURCES[3]} niter=50
38      ''')
39
40  Result('tshot','sc',
41         '''
42         window n1=850 | put o1=1 d1=1 |
43         graph title="Shot Term"
44         label1="Shot Number" unit1= label2=Amplitude unit2=
45         ''')
46  Result('treceiver','sc',
47         '''
48         window f1=850 n1=1063 | put o1=1 d1=1 |
49         graph title="Receiver Term"
50         label1="Receiver Number" unit1= label2=Amplitude unit2=
51         ''')
52
53  # Surface-consistent Log Amplitude for each trace
54  Flow('scarms',['sc','index',sc],
55       '${SOURCES[2]} index=${SOURCES[1]} adj=n')
56
57  # Interpolate to a regular grid
58
59  # !!!! MODIFY BELOW !!!
60
61  # Using sx and sy
62  Flow('scoord','theader',
63       'window n1=2 f1=21 | dd type=float | scale dscale=1e-6')
64  Flow('sint','scarms scoord',
65       '''
66       nnshape coord=${SOURCES[1]} rect1=10 rect2=10
67       o1=785 d1=0.1 n1=251 o2=935 d2=0.1 n2=451 niter=10
68       ''')
69  Result('sint',
70         '''
71         grey color=j title="Log-Amplitude in Shot Coordinates"
72         transp=n label1=X label2=Y unit1=kft unit2=kft
73         bias=-17 clip=3
74         ''')
75
76  End()
```

YOUR OWN DATA

Finally, add results from analyzing the dataset you selected for your course project. You can include the first steps of analyzing the data acquisition geometry and/or any other processing steps that you have taken.

SAVING YOUR WORK

It is important to save all files that you edit by hand (such as `paper.tex` and `SConstruct`) in a version-control system every time you make a revision. The completed assignment is due in two weeks and should be submitted through your team's private GitHub repository.

REFERENCES

Fletcher, R., and C. M. Reeves, 1964, Function minimization by conjugate gradients: Computer Journal, **7**, 149–154.

Hestenes, M. R., and E. Stiefel, 1952, Methods of conjugate gradients for solving linear systems: J. Res. NBS, **49**, 409–436.

Robinson, E. A., and O. M. Osman, eds., 1996, Deconvolution 2: Soc. of Expl. Geophys.

Taner, M. T., and F. Koehler, 1981, Surface consistent corrections: Geophysics, **46**, 17–22.

Webster, G. M., ed., 1978, Deconvolution: Soc. of Expl. Geophys.