Homework 6

Charles Hewitt Dix

ABSTRACT

This homework has only a computational part, with three different tasks. You will experiment with exploding-reflector modeling and reverse-time migration and will process a field dataset from the US East Coast.

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`

You are welcome to do the assignment on your personal computer by installing the required environments. In this case, you can obtain all homework assignments from the `Madagascar` repository by running

```
svn co http://svn.code.sf.net/p/rsf/code/trunk/book/geo384w/hw6
```

COMPUTATIONAL PART

1. We start by returning to the model from Homeworks 1 and 5 with a hyperbolic reflector under a constant velocity layer. The model is shown in Figure 1. Now we will approach the imaging task using reverse-time migration with a two-way wave extrapolation. Figure 2a shows synthetic zero-offset data generated by Kirchhoff modeling. Figure 8 shows an image generated by zero-offset reverse-time migration using an explicit finite-difference wave extrapolation in time.

   Your task: Change the program for reverse-time migration to implement forward-time modeling using the "exploding reflector" approach.

   (a) Change directory

   ```
   cd hw6/hyper
   ```

   (b) Run

   ```
   scons view
   ```

to generate figures and display them on your screen.

(c) Run

```
scons wave.vpl
```

to observe a movie of reverse-time wave extrapolation.

(d) Edit the program in the `rtm.c` file to implement a process opposite to migration: starting from the reflectivity image like the one in Figure 8 and generating zero-offset data like the one in Figure 2a.

(e) Run

```
scons view
```

again to observe the differences.

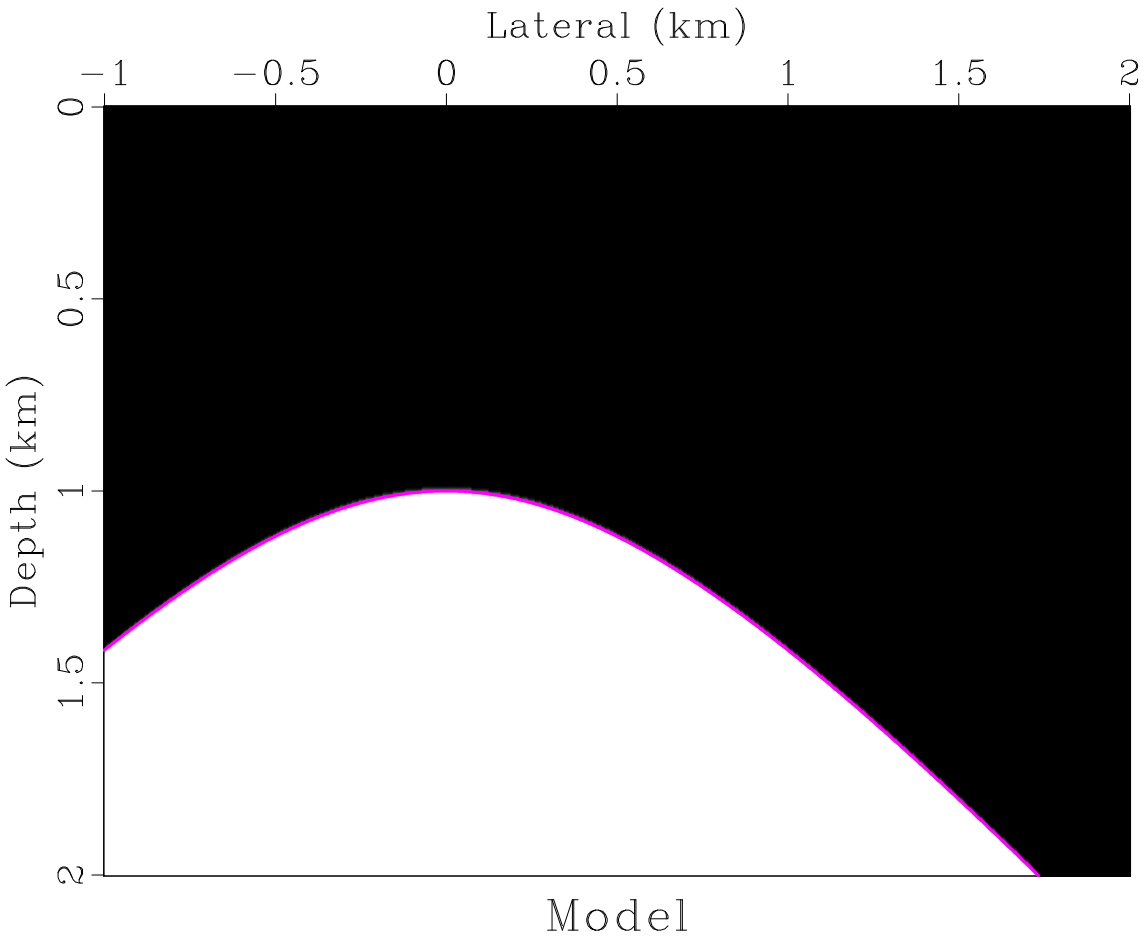
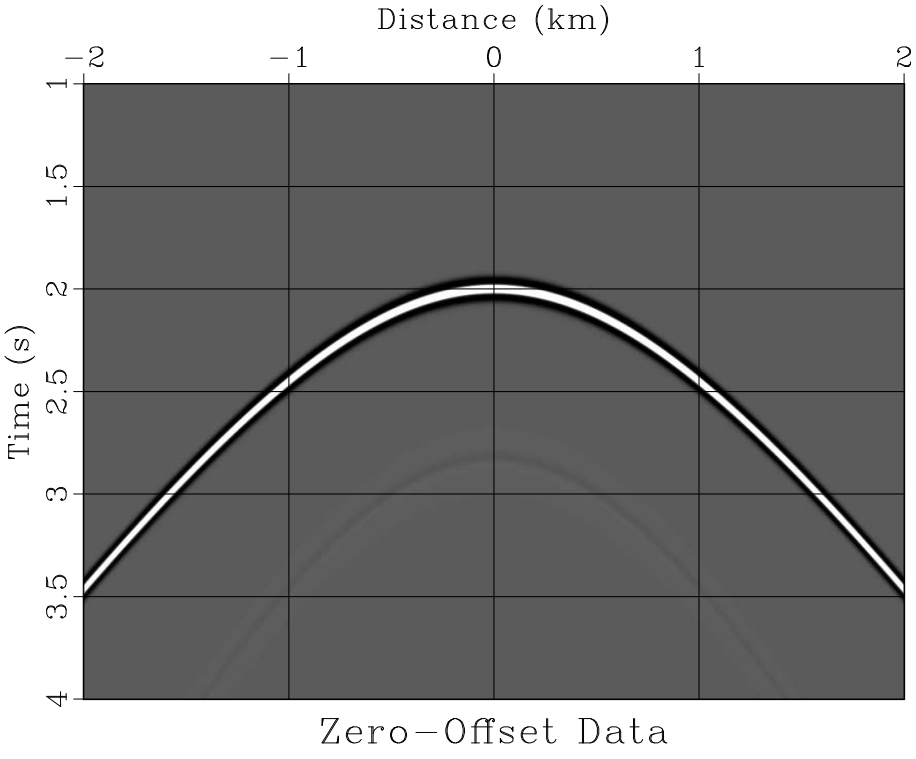
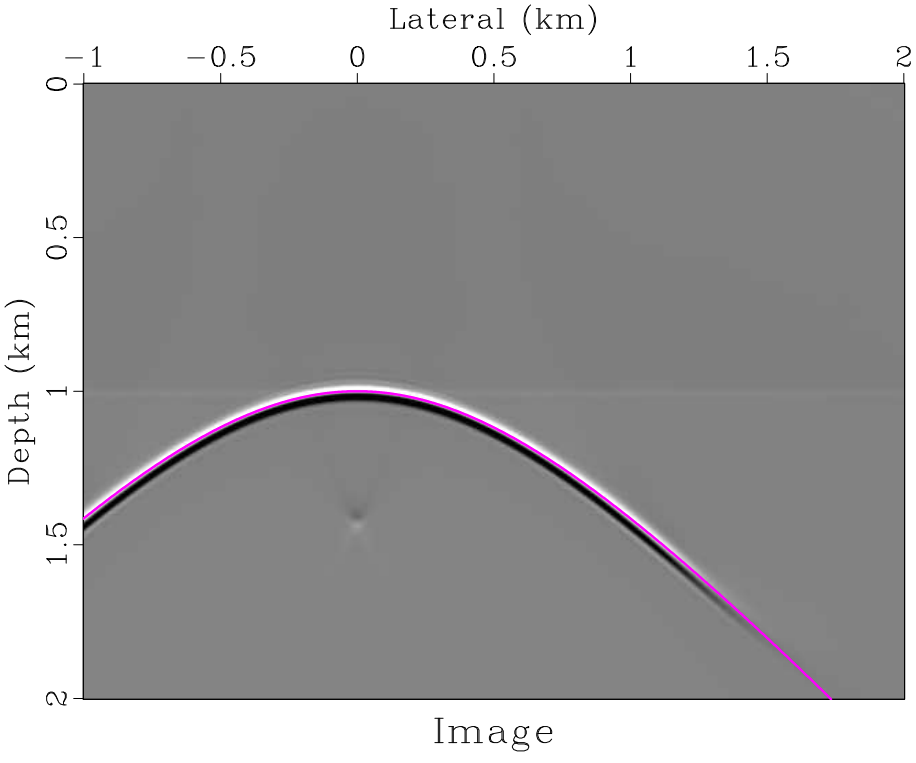


Figure 1: Synthetic velocity model with a hyperbolic reflector.

```
1 from rsf.proj import *
2
3 # Make a reflector model
```



a



b

Figure 2: (a) Synthetic zero-offset data corresponding to the model in Figure 1. (b) Image generated by reverse-time exploding-reflector migration. The location of the exact reflector is indicated by a curve.

```
Flow('refl',None,
     '''
     math n1=10001 o1=-4 d1=0.001 output="sqrt(1+x1*x1)"
     ''')
Flow('model','refl',
     '''
     window min1=-3 max1=3 j1=5 |
     unif2 n1=401 d1=0.005 v00=1,2 |
     put label1=Depth unit1=km label2=Lateral unit2=km |
     smooth rect1=3
     ''')

Plot('model',
     '''
     window min2=-1 max2=2 |
     grey allpos=y bias=1 title=Model
     ''')
Plot('refl',
     '''
     graph wanttitle=n wantaxis=n yreverse=y
     min1=-1 max1=2 min2=0 max2=2
     plotfat=3 plotcol=4
     ''')
Result('model','model refl','Overlay')

# Kirchoff zero-offset modeling
Flow('dip','refl','math output="x1/input" ')

Flow('data','refl dip',
     '''
     kirmod nt=1501 dt=0.004 freq=10
     ns=1201 s0=-3 ds=0.005 nh=1 dh=0.01 h0=0
     twod=y vel=1 dip=${SOURCES[1]} verb=y |
     window | costaper nw2=200
     ''')
Result('data',
     '''
     window min2=-2 max2=2 min1=1 max1=4 |
     grey title="Zero-Offset Data" grid=y
     label1=Time unit1=s label2=Distance unit2=km
     ''')

# Reverse-time migration
proj = Project()
prog = proj.Program('rtm.c')
```

```
49
50 Flow('image wave','model %s data' % prog[0],
51       '''
52       pad beg1=200 end1=200 | math output=0.5 |
53       ./${SOURCES[1]} data=${SOURCES[2]}
54       wave=${TARGETS[1]} jt=20 n0=200
55       ''')
56
57 # Display wave propagation movie
58 Plot('wave',
59       '''
60       window min2=-1 max2=2 min1=0 max1=2 f3=33 |
61       grey wanttitle=n gainpanel=all
62       ''',view=1)
63
64 # Display image
65 Plot('image',
66       '''
67       window min2=-1 max2=2 min1=0 max1=2 |
68       grey title=Image
69       ''')
70 Result('image','image refl','Overlay')
71
72 End()
```

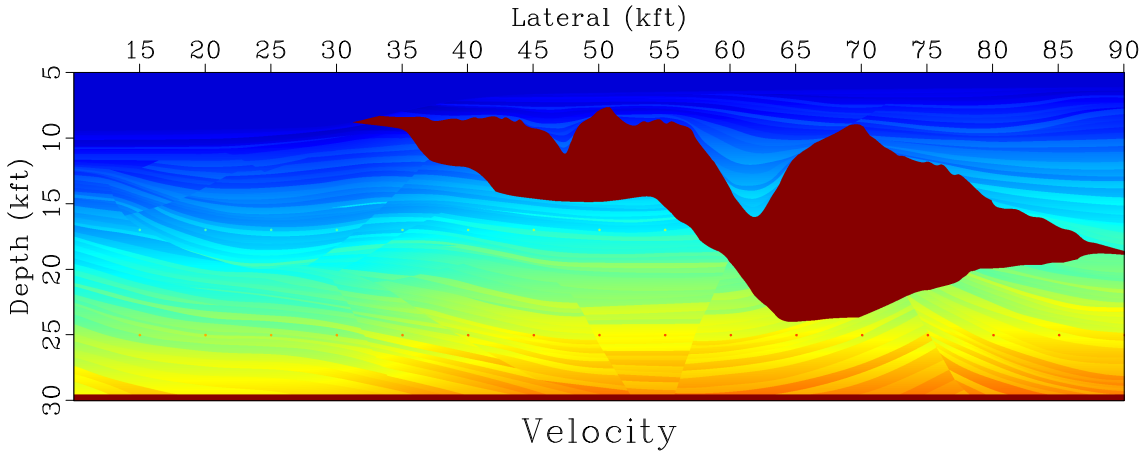
```
1  /* 2-D zero-offset reverse-time migration */
2  #include <rsf.h>
3
4  #ifdef _OPENMP
5  #include <omp.h>
6  #endif
7
8  static int n1, n2;
9  static float c0, c11, c21, c12, c22;
10
11 static void laplacian(float **uin  /* [n2][n1] */,
12                        float **uout /* [n2][n1] */)
13 /* Laplacian operator, 4th-order finite-difference */
14 {
15     int i1, i2;
16
17 #ifdef _OPENMP
18 #pragma omp parallel for                              \
19     private(i2,i1)                                    \
20     shared(n2,n1,uout,uin,c11,c12,c21,c22,c0)
```

```
21 #endif
22     for (i2=2; i2 < n2-2; i2++) {
23         for (i1=2; i1 < n1-2; i1++) {
24             uout[i2][i1] =
25                 c11*(uin[i2][i1-1]+uin[i2][i1+1]) +
26                 c12*(uin[i2][i1-2]+uin[i2][i1+2]) +
27                 c21*(uin[i2-1][i1]+uin[i2+1][i1]) +
28                 c22*(uin[i2-2][i1]+uin[i2+2][i1]) +
29                 c0*uin[i2][i1];
30         }
31     }
32 }
33
34
35 int main(int argc, char* argv[])
36 {
37     int it,i1,i2;          /* index variables */
38     int nt,n12,n0,nx, jt;
39     float dt,dx,dz, dt2,d1,d2;
40
41     float  **vv, **dd;
42     float  **u0,**u1,u2,**ud; /* tmp arrays */
43
44     sf_file data, imag, modl, wave; /* I/O files */
45
46     /* initialize Madagascar */
47     sf_init(argc,argv);
48
49     /* initialize OpenMP support */
50     omp_init();
51
52     /* setup I/O files */
53     modl = sf_input ("in");   /* velocity model */
54     imag = sf_output("out");  /* output image */
55
56     data = sf_input ("data"); /* seismic data */
57     wave = sf_output("wave"); /* wavefield */
58
59     /* Dimensions */
60     if (!sf_histint(modl,"n1",&n1)) sf_error("n1");
61     if (!sf_histint(modl,"n2",&n2)) sf_error("n2");
62
63     if (!sf_histfloat(modl,"d1",&dz)) sf_error("d1");
64     if (!sf_histfloat(modl,"d2",&dx)) sf_error("d2");
65
```

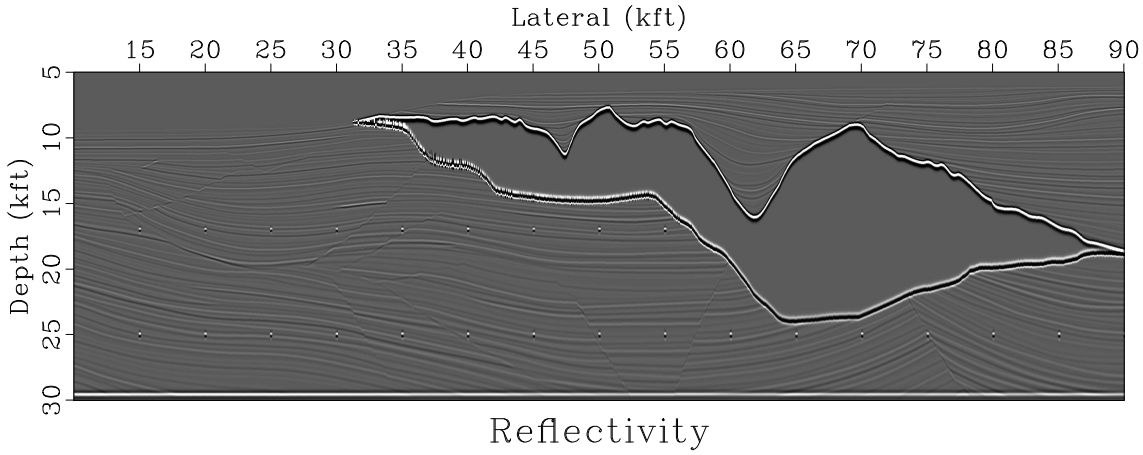
```
66      if (!sf_histint   (data,"n1",&nt)) sf_error("n1");
67      if (!sf_histfloat(data,"d1",&dt)) sf_error("d1");
68
69      if (!sf_histint(data,"n2",&nx) || nx != n2)
70          sf_error("Need n2=%d in data",n2);
71
72      n12 = n1*n2;
73
74      if (!sf_getint("n0",&n0)) n0=0;
75      /* surface */
76      if (!sf_getint("jt",&jt)) jt=1;
77      /* time interval */
78
79      sf_putint(wave,"n3",1+(nt-1)/jt);
80      sf_putfloat(wave,"d3",-jt*dt);
81      sf_putfloat(wave,"o3",(nt-1)*dt);
82
83      dt2 = dt*dt;
84
85      /* set Laplacian coefficients */
86      d1 = 1.0/(dz*dz);
87      d2 = 1.0/(dx*dx);
88
89      c11 = 4.0*d1/3.0;
90      c12=  -d1/12.0;
91      c21 = 4.0*d2/3.0;
92      c22=  -d2/12.0;
93      c0  = -2.0 * (c11+c12+c21+c22);
94
95      /* read data and velocity */
96      dd = sf_floatalloc2(nt,n2);
97      sf_floatread(dd[0],nt*n2,data);
98
99      vv = sf_floatalloc2(n1,n2);
100     sf_floatread(vv[0],n12,modl);
101
102     /* allocate temporary arrays */
103     u0=sf_floatalloc2(n1,n2);
104     u1=sf_floatalloc2(n1,n2);
105     ud=sf_floatalloc2(n1,n2);
106
107     for (i2=0; i2<n2; i2++) {
108         for (i1=0; i1<n1; i1++) {
109             u0[i2][i1]=0.0;
110             u1[i2][i1]=0.0;
```

```
111             ud[i2][i1]=0.0;
112             vv[i2][i1] *= vv[i2][i1]*dt2;
113         }
114     }
115
116     /* Time loop */
117     for (it=nt-1; it >= 0; it--) {
118         sf_warning("%d;",it);
119
120         laplacian(u1,ud);
121
122 #ifdef _OPENMP
123 #pragma omp parallel for                \
124     private(i2,i1,u2)                   \
125     shared(ud,vv,it,u1,u0,dd,n0)
126 #endif
127         for (i2=0; i2<n2; i2++) {
128             for (i1=0; i1<n1; i1++) {
129                 /* scale by velocity */
130                 ud[i2][i1] *= vv[i2][i1];
131
132                 /* time step */
133                 u2 =
134                     2*u1[i2][i1]
135                     - u0[i2][i1]
136                     + ud[i2][i1];
137
138                 u0[i2][i1] = u1[i2][i1];
139                 u1[i2][i1] = u2;
140             }
141
142             /* inject data */
143             u1[i2][n0] += dd[i2][it];
144         }
145
146         if (0 == it%jt)
147             sf_floatwrite(u1[0],n12,wave);
148     }
149     sf_warning(".");
150
151     /* output image */
152     sf_floatwrite(u1[0],n12,imag);
153
154     exit (0);
155 }
```

2. Figure 3 shows the Sigsbee velocity model (Paffenholz et al., 2002) and its an approximate filtered reflectivity.



a



b

Figure 3: (a) Sigsbee velocity model. (b) Approximate reflectivity of the Sigsbee model (an ideal image).

Your task: Apply your exploding-reflector modeling and migration program from the previous task to generate zero-offset data for Sigsbee and image it.

(a) Change directory

```
cd hw6/sigsbee
```

(b) Run

```
scons view
```

to generate figures and display them on your screen.

(c) Modify the `SConstruct` file to implement the modeling and migration experiment.

(d) Include your results in the paper by editing the `hw6/paper.tex` file.

(e) For EXTRA CREDIT, make sure that your modeling and migration are adjoint to each other and apply least-squares exploding-reflector reverse-time migration to the Sigsbee model.

```
from rsf.proj import *

# Download velocity model from the data server
##############################################
vstr = 'sigsbee2a_stratigraphy.sgy'
Fetch(vstr,'sigsbee')
Flow('zvstr',vstr,'segyread read=data')

Flow('zvel','zvstr',
     '''
     put d1=0.025 o2=10.025 d2=0.025
     label1=Depth unit1=kft label2=Lateral unit2=kft |
     scale dscale=0.001
     ''')

Result('zvel',
       '''
       window f1=200 |
       grey title=Velocity titlesz=7 color=j
       screenratio=0.3125 screenht=4 labelsz=5
       mean=y
       ''')


# Compute approximate reflectivity
##################################
Flow('zref','zvel',
     '''
     depth2time velocity=$SOURCE nt=2501 dt=0.004 |
     ai2refl | ricker1 frequency=10 |
     time2depth velocity=$SOURCE
     ''')

Result('zref',
       '''
       window f1=200 |
```

```
        grey title=Reflectivity titlesz=7
        screenratio=0.3125 screenht=4 labelsz=5
        ''')


End()
```

3. In the last part of the homework, you will work with a field dataset: a 2-D line from the Blake Outer Ridge area offshore Florida and Georgia (Figure 4). It was collected by USGS in order to study the occurrence of methane hydrates. The presence of gas hydrates is manifested by a so-called BSR (bottom-simulating reflector). The dataset and its analysis for gas hydrate detection are described by Ecker et al. (1998, 2000).

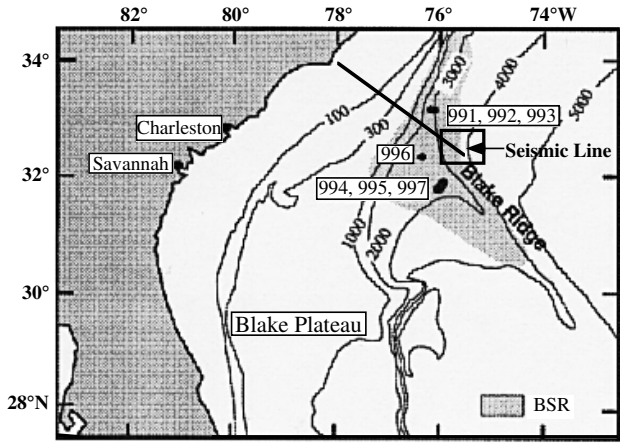


Figure 4: Map of the Blake Outer Ridge area. An region of known gas hydrate distribution as mapped from seismic bottom simulating reflectors (BSR) is highlighted. The seismic survey area is marked by a rectangle.

The following figures show the dataset at different stages of seismic data processing: from initial data to an image in depth.

Your task: Modify the data processing sequence to create a justifiably better image.

```
from rsf.proj import *

# get data
Fetch('cmps-tp.HH','blake')

# CMP (common midpoint) gathers
Flow('cmps','cmps-tp.HH',
     'dd form=native | reverse which=2')
# one CMP
Flow('cmp','cmps',
     '''
     window f3=950 n3=1 max1=6 |
```

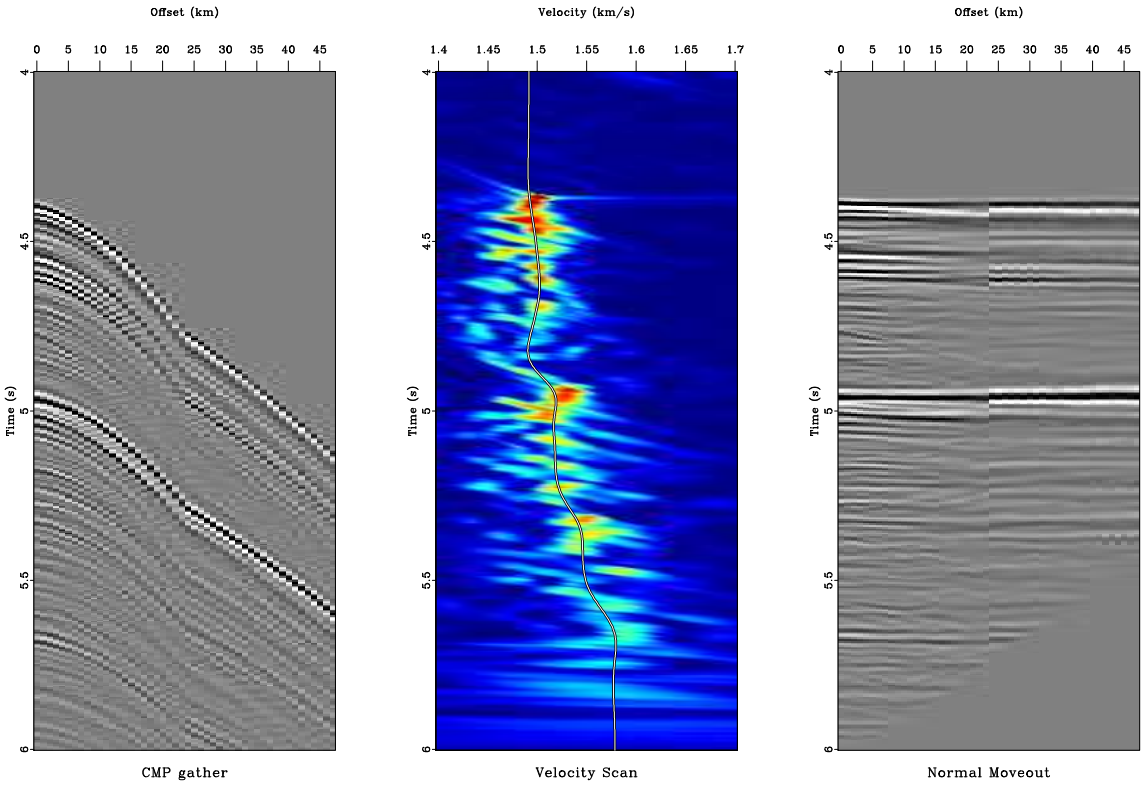
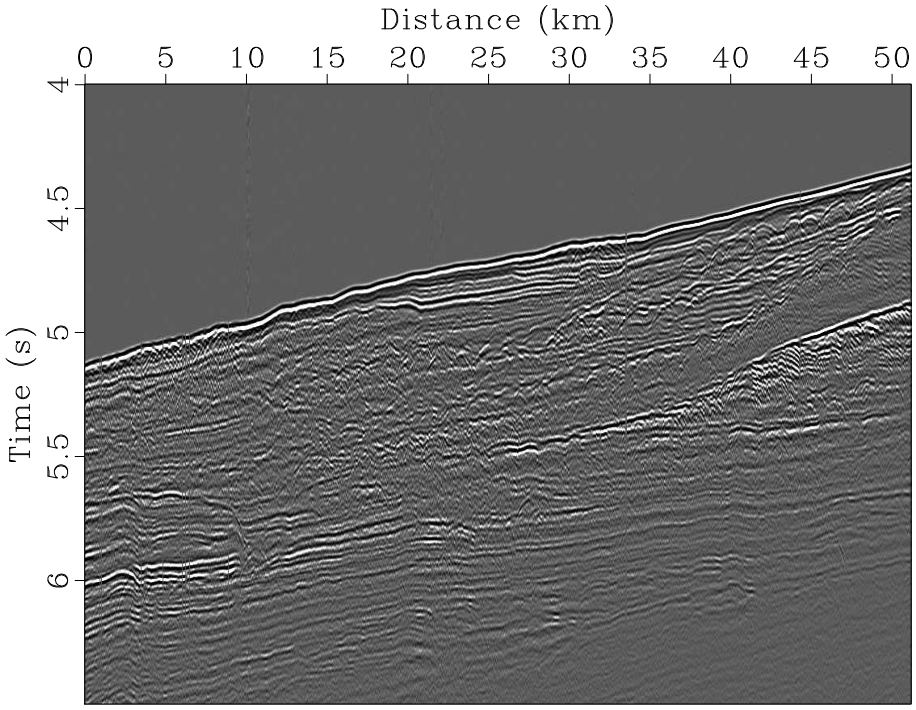
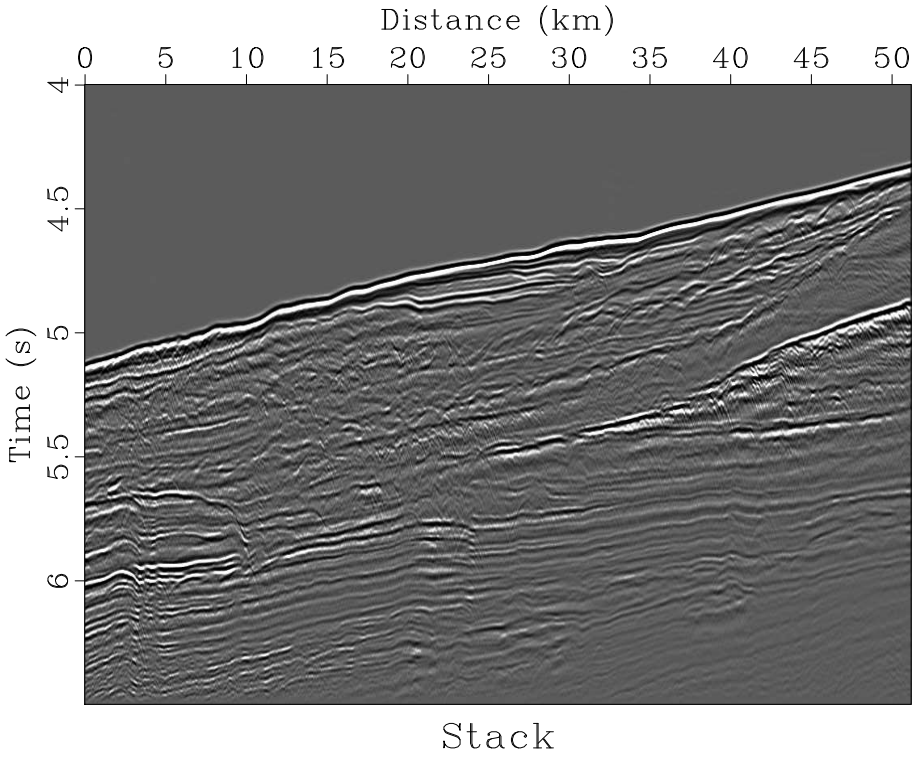


Figure 5: Common midpoint gather (left), velocity analysis panel using normal moveout (middle), and common midpoint gather after normal moveout (right). A curve in the middle plot indicates an automatically picked velocity trend.

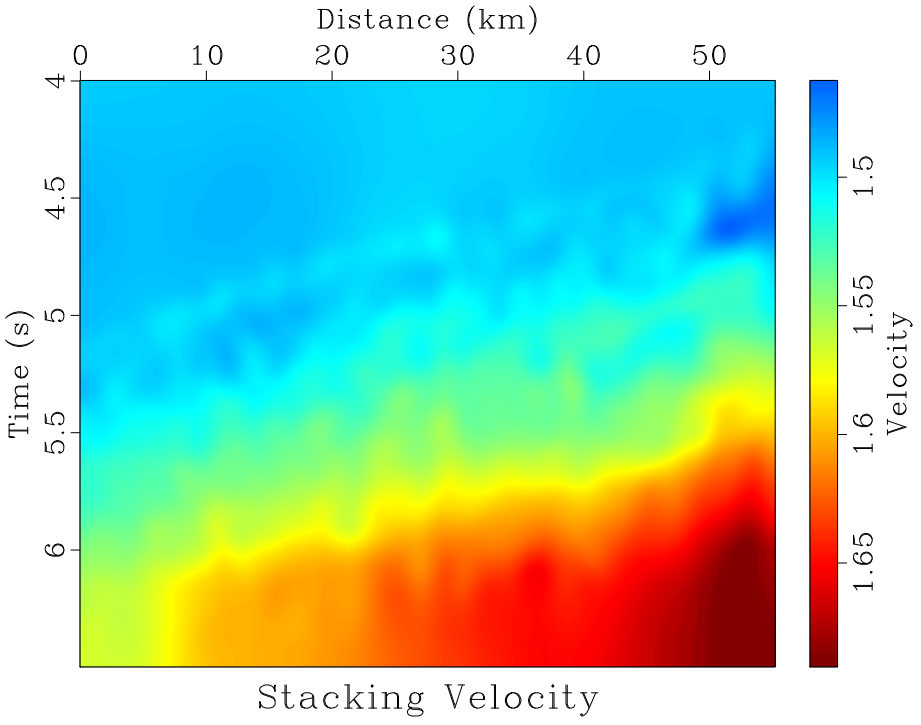


a

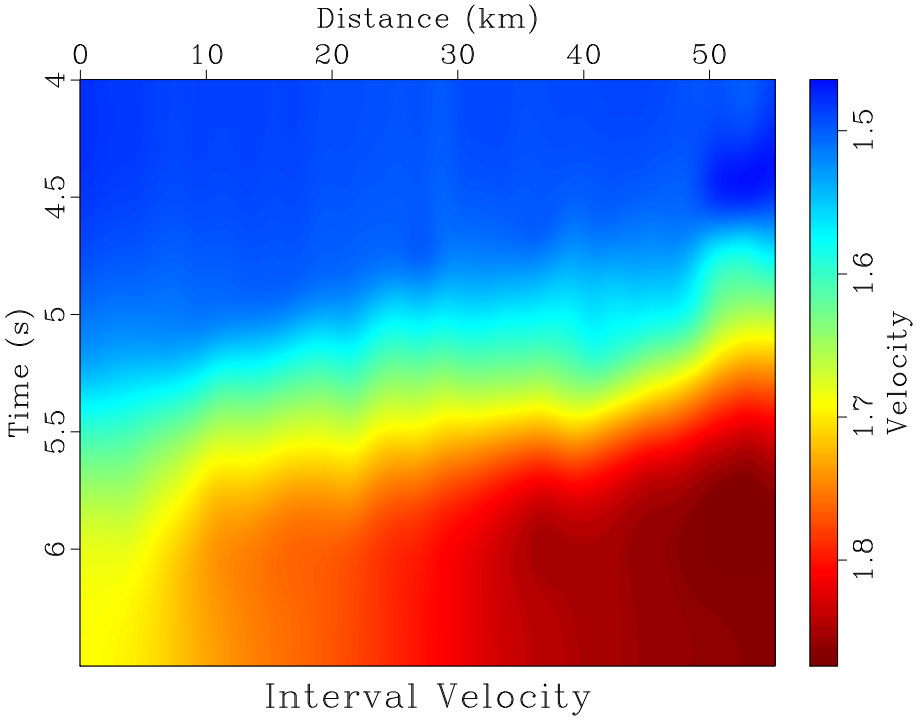


b

Figure 6: (a) Near-offset section. (b) Normal moveout stack.



a



b

Figure 7: (a) Picked stacking velocity. (b) Interval velocity estimated by Dix inversion.

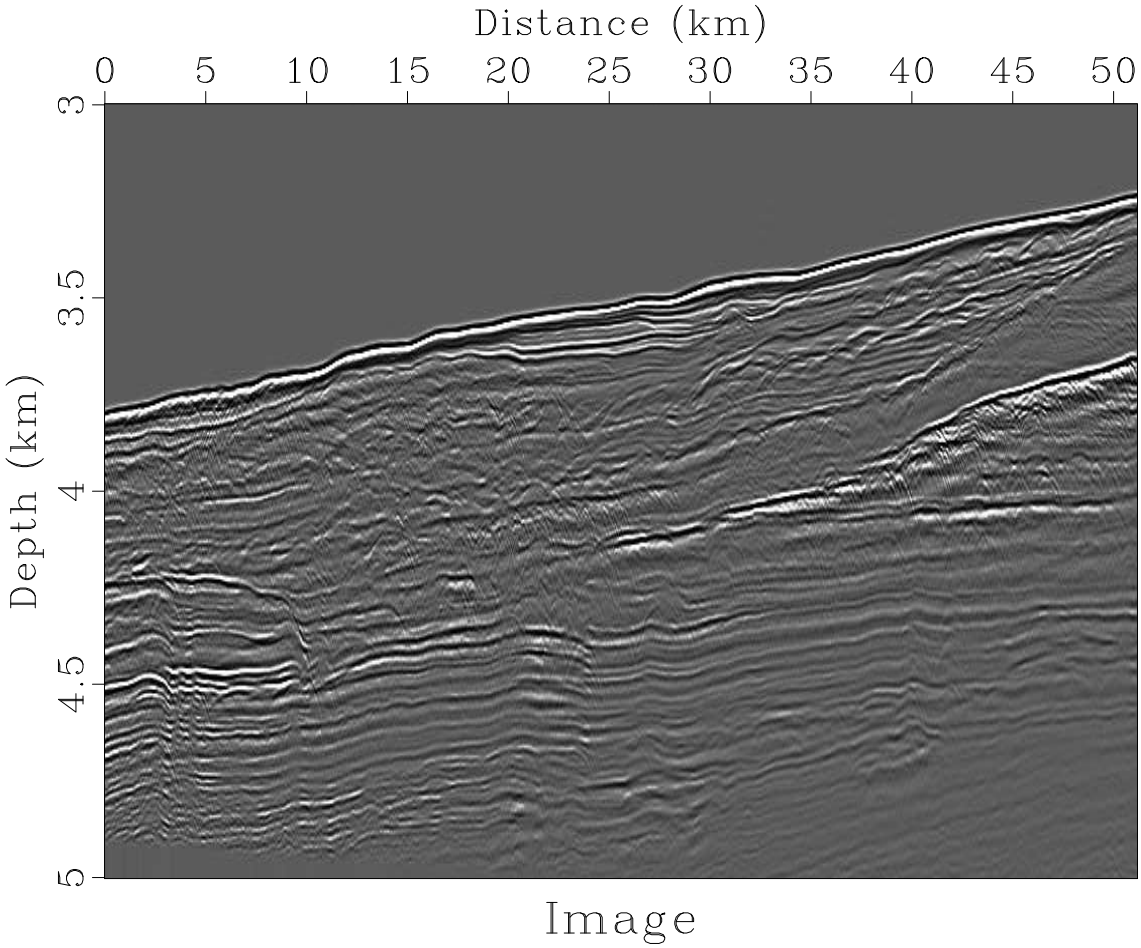


Figure 8: Seismic image created by converting stacked data from time to depth. Can you identify geological features that are not properly imaged?

```
13      put o2=0.0 d2=1
14      ''')
15  Plot('cmp',
16      '''
17      grey title="CMP gather"
18      unit1=s label2=Offset unit2=km
19      ''')
20
21  # Near-offset section
22  Result('noff','cmps',
23          '''
24          window n2=1 n3=1024 |
25          grey title="Near Offset Section"
26          unit1=s label2=Distance unit2=km
27          ''')
28
29  # offset maps
30  Flow('off1',None,'math n1=24 o1=0.4 d1=0.1  output=x1')
31  Flow('off2',None,'math n1=24 o1=2.8 d1=0.05 output=x1')
32  Flow('off','off1 off2','cat axis=1 ${SOURCES[1]}')
33  Flow('offs','off','spray n=111')
34
35  # Velocity analysis
36  ####################
37
38  vscan = '''
39  vscan offset=${SOURCES[1]}
40  v0=1.4 nv=61 dv=0.005 half=n semblance=y
41  '''
42  pick = 'pick rect1=20 rect2=3 vel0=1.5'
43
44  # compute semblance for one CMP
45  Flow('vscan','cmp off',vscan)
46  Plot('vscan',
47      '''
48      grey color=j allpos=y title="Velocity Scan"
49      unit1=s label2=Velocity unit2=km/s pclip=100
50      ''')
51
52  # pick maximum semblance for one CMP
53  Flow('pick','vscan',pick)
54  Plot('pick0','pick',
55      '''
56      graph transp=y yreverse=y min2=1.4 max2=1.7
57      plotcol=7 plotfat=10 pad=n wanttitle=n wantaxis=n
```

```
58        ''')
59  Plot('pick1','pick',
60        '''
61        graph transp=y yreverse=y min2=1.4 max2=1.7
62        plotcol=0 plotfat=1 pad=n wanttitle=n wantaxis=n
63        ''')
64  Plot('vscan2','vscan pick0 pick1','Overlay')
65
66  # compute semblance for every 10th CMP
67  Flow('vscans','cmps offs','window j3=10 | ' + vscan)
68
69  # pick max semblance for every 10th CMP
70  Flow('picks0','vscans',pick)
71
72  # interpolate picks on the original grid
73  Flow('picks','picks0',
74        'transp | remap1 n1=1105 d1=0.05 o1=0 | transp')
75  Result('picks',
76          '''
77          grey color=j scalebar=y barreverse=y
78          allpos=y bias=1.4
79          title="Stacking Velocity"
80          label1=Time unit1=s label2=Distance unit2=km
81          ''')
82
83  # Normal moveout and stack
84  ############################
85
86  nmo = '''
87  nmo offset=${SOURCES[1]}
88  velocity=${SOURCES[2]} half=n
89  '''
90
91  Flow('nmo','cmp off pick',nmo)
92  Plot('nmo',
93        '''
94        grey title="Normal Moveout"
95        unit1=s label2=Offset unit2=km
96        ''')
97  Result('nmo','cmp vscan2 nmo','SideBySideAniso')
98
99  Flow('nmos','cmps off picks',nmo)
100 Flow('stack','nmos','stack')
101 Result('stack',
102         '''
```

```
        window n2=1024 |
        grey title=Stack
        unit1=s label2=Distance unit2=km
        ''')

# Convert from time to depth
############################

# Dix inversion
Flow('semb','vscans picks0','slice pick=${SOURCES[1]}')
Flow('vel0','picks0 semb',
     'dix rect1=20 rect2=2 weight=${SOURCES[1]}')

# interpolate on the original grid
Flow('vel','vel0',
     'transp | remap1 n1=1105 d1=0.05 o1=0 | transp')
Result('vel',
       '''
        grey color=j scalebar=y barreverse=y
        allpos=y bias=1.4
        title="Interval Velocity"
        label1=Time unit1=s label2=Distance unit2=km
        ''')

Flow('image','stack vel',
     '''
     time2depth velocity=${SOURCES[1]}
     intime=y dz=0.005 nz=1001
     ''')
Result('image',
       '''
        window n2=1024 min1=3 | grey title=Image
        label1=Depth unit1=km label2=Distance unit2=km
        ''')

End()
```

(a) Change directory

```
cd hw6/blake
```

(b) Run

```
scons view
```

to generate figures and display them on your screen.

(c) Modify the `SConstruct` file. Check your results by running

```
scons view
```

again.

COMPLETING THE ASSIGNMENT

1. Change directory to `hw6`.
2. Edit the file `paper.tex` in your favorite editor and change the first line to have your name instead of Dix's.
3. Run

   ```
   sftour pscons lock
   ```

   to update all figures.
4. Run

   ```
   sftour pscons -c
   ```

   to remove intermediate files.
5. Run

   ```
   scons pdf
   ```

   to create the final document.
6. Submit your result (file `paper.pdf`) on paper or by e-mail.

REFERENCES

Ecker, C., J. Dvorkin, and A. Nur, 1998, Sediments with gas hydrates: Internal structure from seismic AvO: Geophysics, **63**, 1659–1669.

Ecker, C., J. Dvorkin, and A. M. Nur, 2000, Estimating the amount of gas hydrate and free gas from marine seismic data: Geophysics, **65**, 565–573.

Paffenholz, J., B. McLain, J. Zaske, and P. Keliher, 2002, Subsalt multiple attenuation and imaging: Observations from the Sigsbee2B synthetic dataset: 72nd Annual International Meeting, SEG, Soc. of Expl. Geophys., 2122–2125.