

# Homework 4

*Alexey Malovichko*

## ABSTRACT

This homework has two parts. In the theoretical part, you will perform analytical derivations related to the shifted hyperbola approximation. In the computational part, you will experiment with imaging two synthetic datasets and a field dataset from the Gulf of Mexico using velocity continuation.

Completing the computational part of this homework assignment requires

- Madagascar software environment available from <http://www.ahay.org>
- L<sup>A</sup>T<sub>E</sub>X 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 https://rsf.svn.sourceforge.net/svnroot/rsf/trunk/book/geo384w/hw4
```

## THEORETICAL PART

Using the hyperbolic travelttime approximation

$$T = \sqrt{\left(T_0 - \frac{z}{V(x_0, T_0)}\right)^2 + \frac{(x_0 - x)^2}{V^2(x_0, T_0)}} \quad (1)$$

makes the geometrical imaging analysis equivalent to analyzing wave propagation in a constant-velocity medium. In particular, we can easily verify that the travelttime satisfies the isotropic eikonal equation

$$\left(\frac{\partial T}{\partial x}\right)^2 + \left(\frac{\partial T}{\partial z}\right)^2 = \frac{1}{V^2}. \quad (2)$$

Suppose that you switch to the more accurate shifted hyperbola approximation

$$T = \left(T_0 - \frac{z}{V}\right) \left(1 - \frac{1}{S}\right) + \frac{1}{S} \sqrt{\left(T_0 - \frac{z}{V}\right)^2 + S \frac{(x_0 - x)^2}{V^2}} \quad (3)$$

1. How would you need to modify the eikonal equation?
2. How would you need to modify the following expressions for the escape time and location for use in the angle-domain time migration?

$$\hat{T} = \frac{T_0 - z/V}{\cos \theta} \quad (4)$$

$$\hat{x} = x_0 + V \left( T_0 - \frac{z}{V} \right) \tan \theta \quad (5)$$

## COMPUTATIONAL PART

1. In the first part of the computational assignment, you will experiment with imaging a synthetic seismic reflection dataset from Homework 3 using prestack velocity continuation.

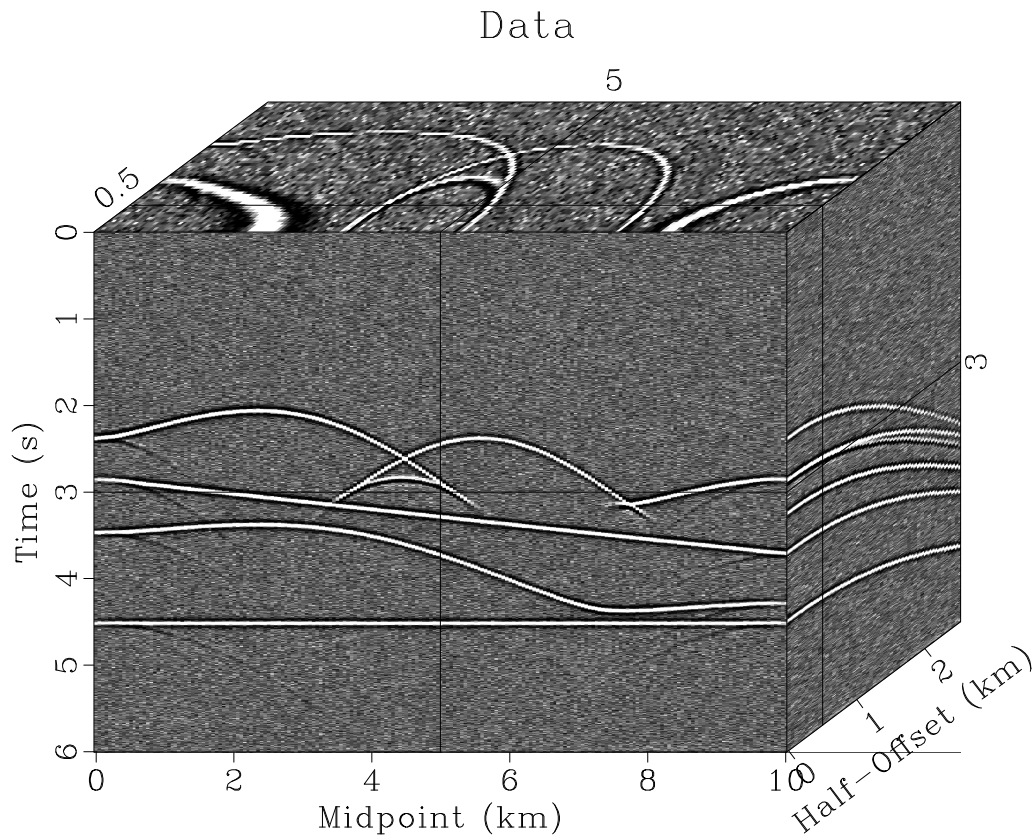
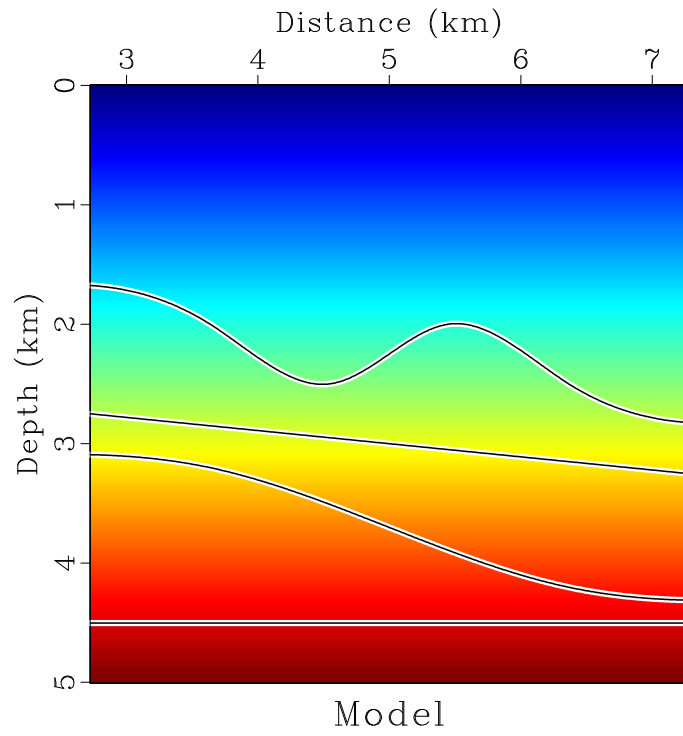


Figure 1: 2-D synthetic data.

Figure 6 shows a synthetic reflection dataset computed from a reflector model shown in Figure ?? and assuming a velocity model with a constant vertical gradient  $V(z) = 1.5 + 0.25z$ . A small amount of random noise is added to the data.

Figure 2: (a) Synthetic model: curved reflectors in a  $V(z)$  velocity.



### Initial Migration

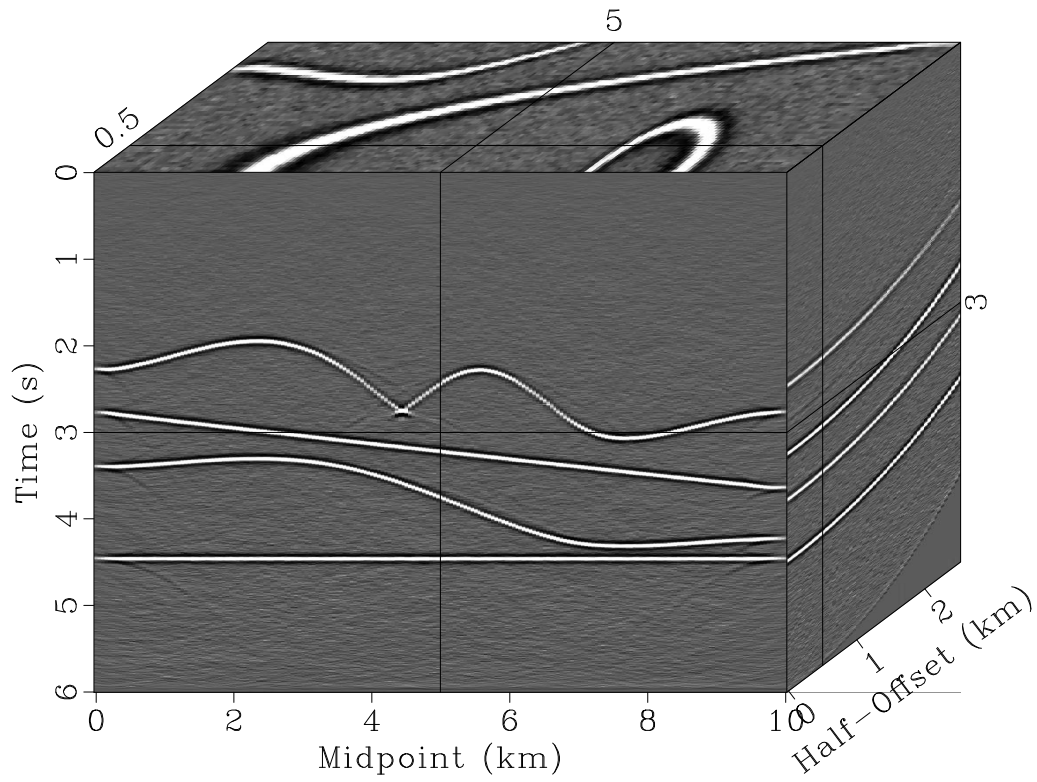


Figure 3: Initial constant-velocity migration.

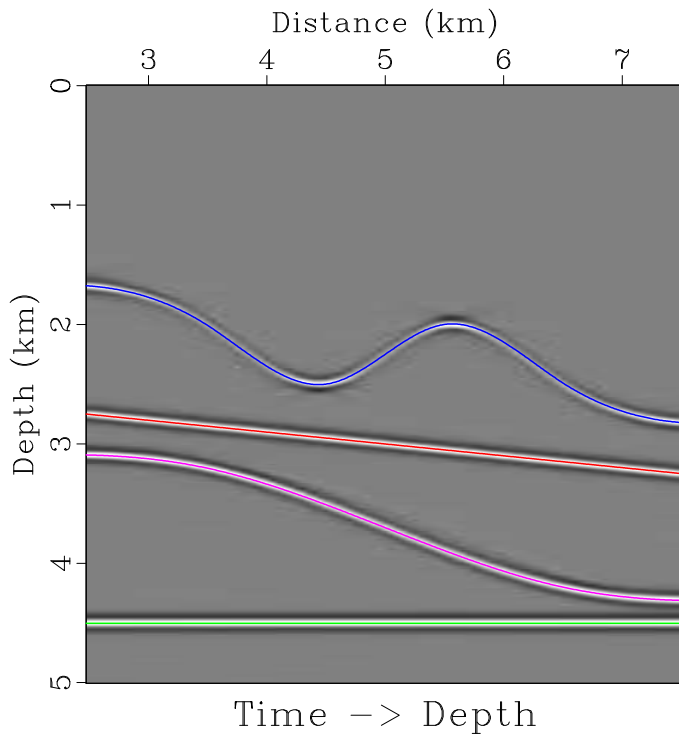


Figure 4: Time migration converted to depth, with reflectors overlaid.

Figure 3 shows an initial prestack common-offset time migration using a constant velocity of 1.5 km/s. Figure 4 shows the result of prestack time migration after velocity continuation, extraction of a velocity slice, and conversion from time to depth.

(a) Change directory

```
cd hw4/synth
```

(b) Run

```
scons view
```

to generate the figures and display them on your screen. If you are on a computer with multiple CPUs, you can also try

```
pscons view
```

to run certain computations in parallel.

(c) Run

```
pscons velcon.vpl
```

to display a movie of the velocity continuation process.

(d) Run

```
pscons semb.vpl
```

to display a movie slicing through a semblance cube computed from velocity continuation.

- (e) The processing flow in the `SConstruct` file involves some cheating: the exact RMS velocity is used to extract the final image. Modify the processing flow so that only properties estimated from the data get used.

```

1 from rsf.proj import *
2
3 # Generate a reflector model
4
5 layers = (
6     ((0,2),(3.5,2),(4.5,2.5),(5,2.25),
7      (5.5,2),(6.5,2.5),(10,2.5)),
8     ((0,2.5),(10,3.5)),
9     ((0,3.2),(3.5,3.2),(5,3.7),
10     (6.5,4.2),(10,4.2)),
11     ((0,4.5),(10,4.5))
12 )
13
14 nlays = len(layers)
15 for i in range(nlays):
16     inp = 'inp%d' % i
17     Flow(inp+'.asc',None,
18         '''
19         echo %s in=$TARGET
20         data_format=ascii_float n1=2 n2=%d
21         ''' % \
22         (' '.join(map(lambda x: ' '.join(map(str,x)),
23                     layers[i])),len(layers[i])))
24
25 dim1 = 'o1=0 d1=0.01 n1=1001'
26 Flow('lay1','inp0.asc',
27     'dd form=native | spline %s fp=0,0' % dim1)
28 Flow('lay2',None,
29     'math %s output="2.5+x1*0.1" ' % dim1)
30 Flow('lay3','inp2.asc',
31     'dd form=native | spline %s fp=0,0' % dim1)
32 Flow('lay4',None,'math %s output=4.5' % dim1)
33
34 Flow('lays','lay1 lay2 lay3 lay4',
35     'cat axis=2 ${SOURCES[1:4]}')
36
37 graph = '''
38 graph min1=2.5 max1=7.5 min2=0 max2=5
39 yreverse=y wantaxis=n wanttitle=n screenratio=1

```

```

40  ' ' '
41  Plot( 'lays0 ', 'lays ', graph + ' plotfat=10 plotcol=0')
42  Plot( 'lays1 ', 'lays ', graph + ' plotfat=2 plotcol=7')
43  Plot( 'lays2 ', 'lays ', graph + ' plotfat=2')
44
45  # Velocity
46
47  Flow( 'vofz ', None,
48      ' ' '
49      math output="1.5+0.25*x1"
50      d2=0.05 n2=201 d1=0.01 n1=501
51      label1=Depth unit1=km
52      label2=Distance unit2=km
53      label=Velocity unit=km/s
54      ' ' ' )
55  Plot( 'vofz ',
56      ' ' '
57      window min2=2.75 max2=7.25 |
58      grey color=j allpos=y bias=1.5
59      title=Model screenratio=1
60      ' ' ' )
61
62  Result( 'model ', 'vofz lays0 lays1 ', 'Overlay ')
63
64  # Model data
65
66  Flow( 'dips ', 'lays ', 'deriv | scale dscale=100')
67  Flow( 'modl ', 'lays dips ',
68      ' ' '
69      kirmod cmp=y dip=${SOURCES[1]}
70      nh=51 dh=0.1 h0=0
71      ns=201 ds=0.05 s0=0
72      freq=10 dt=0.004 nt=1501
73      vel=1.5 gradz=0.25 verb=y |
74      tpow tpow=1 |
75      put d2=0.05 label3=Midpoint unit3=km
76      ' ' ', split=[1,1001], reduce='add')
77
78  # Add random noise
79  Flow( 'data ', 'modl ', 'noise var=1e-6 seed=101811')
80
81  Result( 'data ',
82      ' ' '
83      byte |
84      transp plane=23 |

```

```

85     grey3 flat=n frame1=750 frame2=100 frame3=10
86     label1=Time unit1=s
87     label3=Half-Offset unit3=km
88     title=Data point1=0.8 point2=0.8
89     ' ' ')
90
91 # Initial constant-velocity migration
92 #####
93 Flow('mig', 'data',
94     ' ' ,
95     transp plane=23 |
96     spray axis=3 n=1 d=0.1 o=0 |
97     preconstkirch vel=1.5 |
98     halfint inv=1 adj=1
99     ' ' , split=[2,51], reduce='cat axis=4')
100
101 Result('mig',
102     ' ' ,
103     byte | window |
104     grey3 flat=n frame1=750 frame2=100 frame3=10
105     label1=Time unit1=s
106     label3=Half-Offset unit3=km
107     title="Initial Migration" point1=0.8 point2=0.8
108     ' ' ')
109
110 # Velocity continuation
111 #####
112
113 Flow('thk', 'mig',
114     'window | transp plane=23 | cosft sign3=1')
115 Flow('velconk', 'thk',
116     'fourvc nv=81 dv=0.01 v0=1.5 verb=y',
117     split=[3,201])
118 Flow('velcon', 'velconk', 'cosft sign3=-1')
119
120 Plot('velcon',
121     ' ' ,
122     transp plane=23 memsize=1000 |
123     window min2=2.5 max2=7.5 |
124     grey title="Velocity Continuation"
125     ' ' , view=1)
126
127 # Continue data squared
128 Flow('thk2', 'mig',
129     ' ' ,

```

```

130     mul $SOURCE |
131     window | transp plane=23 | cosft sign3=1
132     ' ' ')
133 Flow( 'velconk2 ', 'thk2 ',
134       'fourvc nv=81 dv=0.01 v0=1.5 verb=y',
135       split=[3,201])
136 Flow( 'velcon2 ', 'velconk2 ', 'cosft sign3=-1')
137
138 # Compute semblance
139 Flow( 'semb ', 'velcon velcon2 ',
140       ' ' ')
141     mul $SOURCE | divn den=${SOURCES[1]} rect1=25
142     ' ' ', split=[3,201])
143
144 Plot( 'semb ',
145       ' ' ')
146     byte gainpanel=all allpos=y |
147     transp plane=23 |
148     grey3 flat=n frame1=750 frame2=0 frame3=48
149     label1=Time unit1=s color=j
150     label3=Velocity unit3=km/s movie=2 dframe=5
151     title=Semblance point1=0.8 point2=0.8
152     ' ' ', view=1)
153
154 # Extracting images
155 #####
156 Flow( 'voft ', 'vofz ',
157       'depth2time velocity=$SOURCE dt=0.004 nt=1501')
158 Flow( 'vrms ', 'voft ',
159       ' ' ')
160     add mode=p $SOURCE | causint |
161     math output="sqrt(input*0.004/(x1+0.004))"
162     ' ' ')
163
164 # Using vrms is CHEATING
165 #####
166 Flow( 'slice ', 'velcon vrms ', 'slice pick=${SOURCES[1]} ' )
167
168 # Using vofz is CHEATING
169 #####
170 Flow( 'dmig ', 'slice vofz ',
171       'time2depth velocity=${SOURCES[1]} ' )
172
173 Plot( 'dmig ',
174       ' ' ')

```

```

175 window max1=5 min2=2.5 max2=7.5 |
176 grey title="Time -> Depth" screenratio=1
177 label2=Distance label1=Depth unit1=km
178 ' ' ' )
179
180 Result ( 'dmig' , 'Overlay' )
181 Result ( 'dmig2' , 'dmig lays2' , 'Overlay' )
182
183 End ( )

```

2. In the second part of the computational assignment, we will use velocity continuation again but this time on a synthetic zero-offset section containing diffraction events.

Figure 5 shows a famous Sigsbee synthetic velocity model. We will focus on the left part of the model, which is appropriate for time-domain imaging. A synthetically generated zero-offset section is shown in Figure 6.

Our processing strategy is to extract diffractions from the data (Figure 7) and to image them using zero-offset velocity continuation (Figure 8). In addition, we are going to analyze the image by expanding it in dip angles by using dip-angle migration (Figure 9).

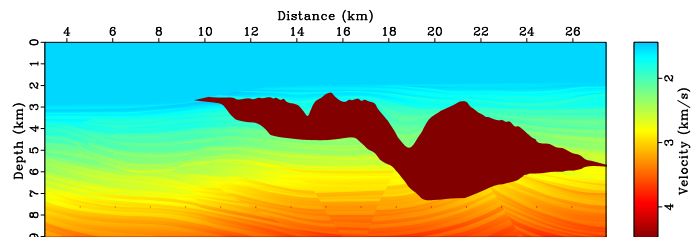


Figure 5: Sigsbee velocity model.

- (a) Change directory

```
cd hw4/sigsbee
```

- (b) Run

```
scons view
```

to generate the figures and display them on your screen. If you are on a computer with multiple CPUs, you can also try

```
pscons view
```

to run certain computations in parallel.

- (c) Generate a movie displaying the velocity continuation process. Is it possible to detect velocities from focusing zero-offset diffractions?

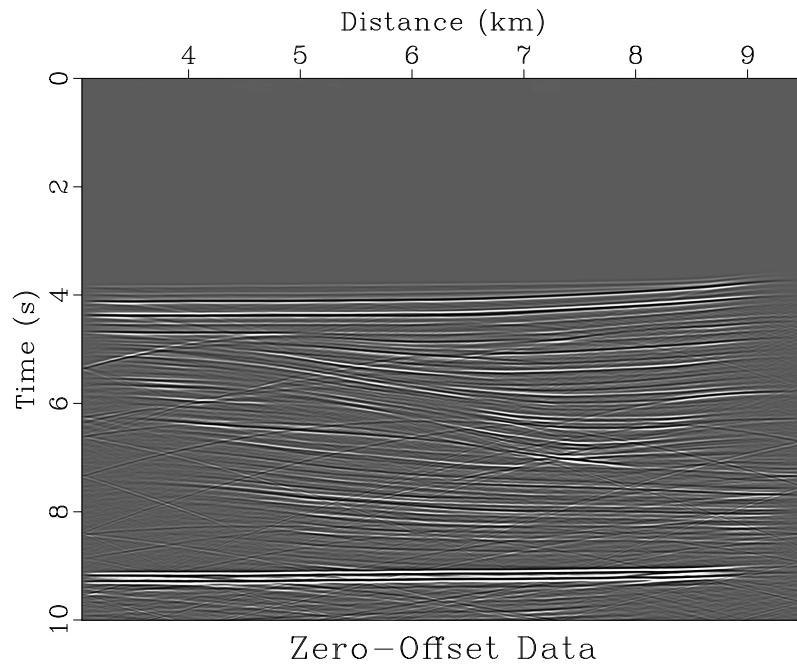


Figure 6: Zero-offset synthetic data.

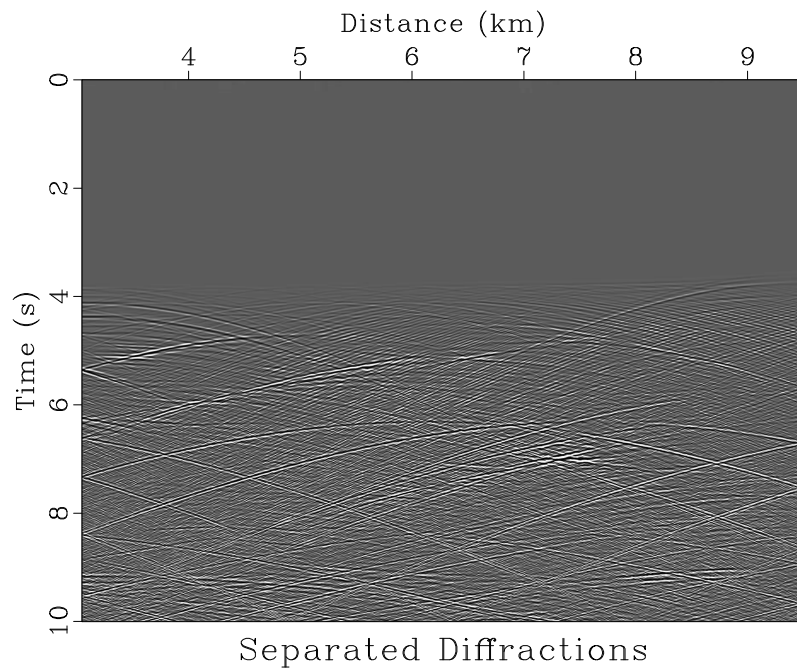


Figure 7: Diffractions extracted from the data by plane-wave destruction.

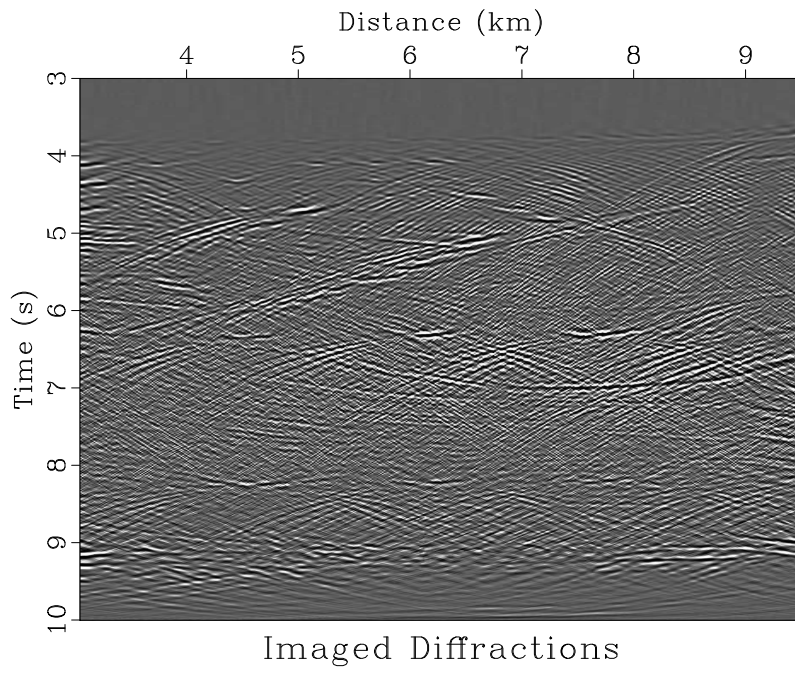


Figure 8: Time-migrated image of diffractions.

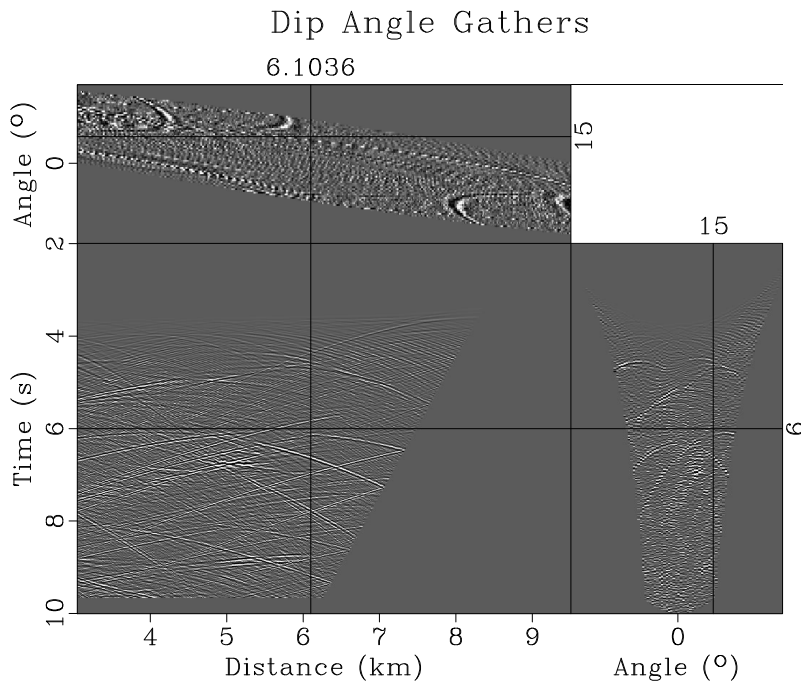


Figure 9: Dip angle gathers from constant-velocity angle-domain migration.

- (d) Modify the program in the `anglemig.c` file to input a variable migration velocity instead of using a constant velocity. Regenerate Figure 9 using a variable velocity

```
pscons anglemig.view
```

Do you notice a difference?

- (e) For EXTRA CREDIT, find a way to estimate migration velocity from the data.

```

1 from rsf.proj import *
2
3 # Download velocity model from the data server
4 #####
5 vstr = 'sigsbee2a_stratigraphy.sgy'
6 Fetch(vstr, 'sigsbee')
7 Flow('zvstr', vstr, 'segyread read=data')
8
9 Flow('vel', 'zvstr',
10      '',
11      put d1=0.00762 o2=3.05562 d2=0.00762
12      label1=Depth unit1=km label2=Distance unit2=km
13      label=Velocity unit=km/s |
14      scale dscale=0.0003048
15      '')
16 Result('vel',
17        '',
18        grey wanttitle=n color=j allpos=y
19        screenratio=0.3125 screenht=4 labelsz=4
20        scalebar=y barreverse=y
21        '')
22
23 # Window a portion
24 Flow('vel2', 'vel', 'window max2=9.5')
25
26 dt = 0.002
27 nt = 5001
28
29 # Convert to RMS
30 Flow('voft', 'vel2',
31      'depth2time velocity=$SOURCE dt=%g nt=%d' % (dt, nt))
32 Flow('vrms', 'voft',
33      '',
34      add mode=p $SOURCE | causint |
35      math output="sqrt(input*%g/(x1+%g))" |
36      window j1=2 j2=2

```

```

37     ' ' % (dt, dt))
38
39 # Download zero-offset from the data server
40 #####
41 Fetch('data0.rsf', 'sigsbee')
42 Flow('data', 'data0', 'dd form=native')
43
44 Result('data',
45       'grey title="Zero-Offset Data" ')
46
47 # Slope estimation
48 Flow('dip', 'data', 'dip rect1=100 rect2=10')
49 Result('dip',
50       ' '
51       'grey color=j scalebar=y
52       title="Dominant Slope"
53       barlabel=Slope barunit=samples
54       ' ')
55
56 # Plane-wave destruction
57 Flow('dif', 'data dip', 'pwd dip=${SOURCES[1]}')
58 Result('dif', 'grey title="Separated Diffractions" ')
59
60 # Velocity continuation
61 Flow('fourier', 'dif', 'cosft sign2=1')
62 Flow('velconf', 'fourier',
63     ' '
64     'stolt vel=1.5 | spray axis=2 n=1 o=0 d=1 |
65     fourvc pad2=4096 nv=61 dv=0.02 v0=1.4 verb=y
66     ' ', split=[2,424], reduce='cat axis=3')
67 Flow('velcon', 'velconf',
68     ' '
69     'transp plane=23 memsize=1000 |
70     cosft sign2=-1 |
71     transp plane=23 memsize=1000
72     ' ')
73
74 # Picking a slice
75 #####
76 Flow('dimage', 'velcon vrms',
77     'slice pick=${SOURCES[1]}')
78 Result('dimage',
79     ' '
80     'window min1=3 |
81     grey title="Imaged Diffractions"

```

```

82     ' ' ' )
83
84 # Angle-gather migration
85 #####
86 prog = Program( 'anglemig.c',
87                 CPPDEFINES='NO_BLAS', LIBS=['rsf', 'm'] )
88
89 Flow( 'anglemig', 'dif %s' % prog[0],
90       './${SOURCES[1]} vel=2.5 na=90 a0=-45 da=1' )
91
92 Result( 'anglemig',
93         ' ' '
94         window min2=2 |
95         transp | transp plane=23 memsize=1000 |
96         byte gainpanel=all | grey3
97         frame1=1000 frame2=200 frame3=60 unit3="\^o\_-"
98         title="Dip Angle Gathers" point1=0.7 point2=0.7
99         ' ' ' )
100
101 End()

```

```

1  /* 2-D angle-domain zero-offset migration. */
2  #include <rsf.h>
3
4  static float get_sample (float **dat,
5                          float t, float y,
6                          float t0, float y0,
7                          float dt, float dy,
8                          int nt, int ny)
9  /* extract data sample by linear interpolation */
10 {
11     int it, iy;
12
13     y = (y - y0)/dy; iy = floorf (y);
14     y -= (float)iy;
15     if (iy < 0 || iy >= (ny - 1)) return 0.0;
16     t = (t - t0)/dt; it = floorf (t);
17     t -= (float)it;
18     if (it < 0 || it >= (nt - 1)) return 0.0;
19
20     return (dat[iy][it]*(1.0 - y)*(1.0 - t) +
21            dat[iy][it + 1]*(1.0 - y)*t +
22            dat[iy + 1][it]*y*(1.0 - t) +
23            dat[iy + 1][it + 1]*y*t);
24 }

```

```

25
26 int main (int argc, char* argv[])
27 {
28     int iz, ix, nx, iy, ia, na, nt;
29     float dt, dy, vel, da, a0, dx, z, t, y, x, a;
30     float **dat, *imag;
31     sf_file data, imag;
32
33     sf_init (argc, argv);
34
35     data = sf_input ("in");
36     imag = sf_output ("out");
37
38     /* get dimensions */
39     if (!sf_histint (data, "n1", &nt)) sf_error ("n1");
40     if (!sf_histint (data, "n2", &nx)) sf_error ("n2");
41     if (!sf_histfloat (data, "d1", &dt)) sf_error ("d1");
42     if (!sf_histfloat (data, "d2", &dx)) sf_error ("d2");
43
44     if (!sf_getint ("na", &na)) sf_error ("Need na=");
45     /* number of angles */
46     if (!sf_getfloat ("da", &da)) sf_error ("Need da=");
47     /* angle increment */
48     if (!sf_getfloat ("a0", &a0)) sf_error ("Need a0=");
49     /* initial angle */
50
51     sf_shiftdim (data, imag, 1);
52
53     sf_putint (imag, "n1", na);
54     sf_putfloat (imag, "d1", da);
55     sf_putfloat (imag, "o1", a0);
56     sf_putstr (imag, "label1", "Angle");
57
58     /* degrees to radians */
59     a0 *= SF_PI/180.;
60     da *= SF_PI/180.;
61
62     if (!sf_getfloat ("vel", &vel)) vel=1.5;
63     /* constant velocity */
64
65     dat = sf_floatalloc2 (nt, nx);
66     sf_floatread (dat[0], nt*nx, data);
67
68     imag = sf_floatalloc (na);
69

```

```

70
71     for (ix = 0; ix < nx; ix++) {
72         x = ix*dx;
73         sf_warning ("CMP %d of %d;", ix, nx);
74
75         for (iz = 0; iz < nt; iz++) {
76             z = iz*dt;
77
78             for (ia = 0; ia < na; ia++) {
79                 a = a0+ia*da;
80
81                 t = z/cosf(a);
82                 /* escape time */
83                 y = x+0.5*vel*t*sinf(a);
84                 /* escape location */
85
86                 img[ia] = get_sample (dat,t,y,0.,0.,
87                                     dt,dx,nt,nx);
88             }
89
90             sf_floatwrite (img, na, imag);
91         } /* iz */
92     } /* ix */
93
94     exit(0);
95 }

```

3. In the final part of the computational assignment, we return to the 2-D field dataset from the Gulf of Mexico. The zero-offset data after a DMO stack are shown in Figure 10.

- (a) Change directory

```
cd hw4/gulf
```

- (b) Run

```
scons view
```

to generate Figure 10 and display it on your screen.

- (c) Edit the `SConstruct` file to implement a processing flow involving velocity continuation and angle-gather migration. Make sure to select appropriate processing parameters.

```

1 from rsf.proj import *
2

```

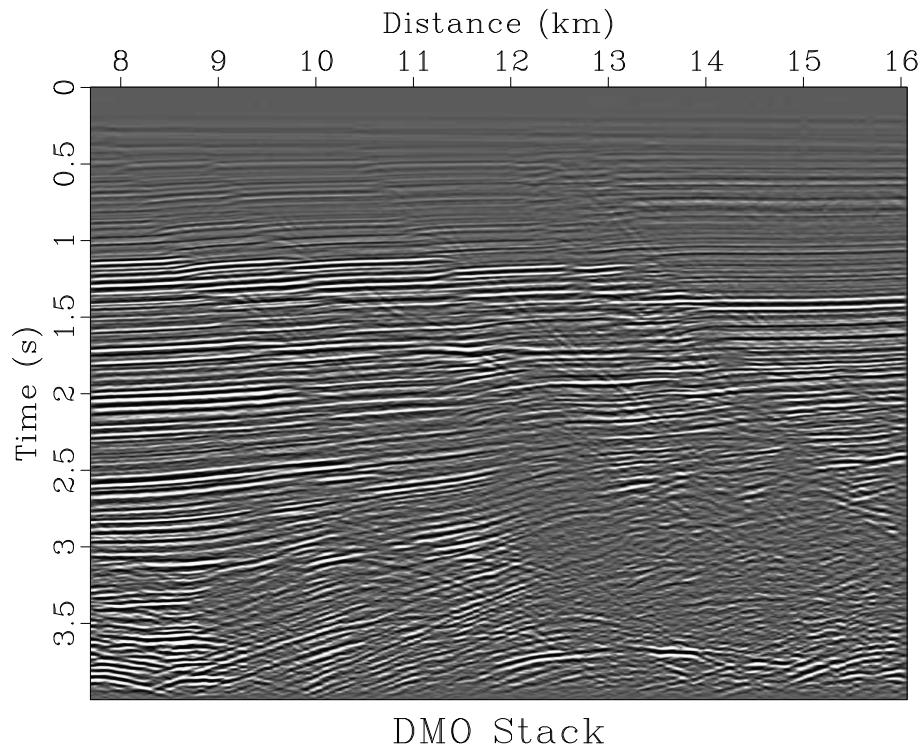


Figure 10: 2-D field dataset from the Gulf of Mexico after DMO stack.

```

3 Fetch('bei-stack.rsf','midpts')
4 Flow('stack','bei-stack',
5     '','')
6     dd form=native |
7     put label2=Distance unit2=km label1=Time unit1=s
8     '')
9
10 Result('stack','grey title="DMO Stack" ')
11
12 End()

```

## COMPLETING THE ASSIGNMENT

1. Change directory to `hw4`.
2. Edit the file `paper.tex` in your favorite editor and change the first line to have your name instead of Malovichko's.

3. Run

```
sftour sconslack
```

to update all figures.

4. Run

```
sftour sconslack -c
```

to remove intermediate files.

5. Run

```
sconslack pdf
```

to create the final document.

6. Submit your result (file `paper.pdf`) on paper or by e-mail.