

Homework 3

Gustav Kirchhoff

ABSTRACT

This homework has two parts. In the theoretical part, you will perform several analytical derivations related to geometrical integration and reflections from elliptical reflections. In the computational part, you will experiment with imaging a synthetic dataset and a field dataset from the Gulf of Mexico.

Completing the computational part of this homework assignment requires

- Madagascar software environment available from <http://www.ahay.org>
- L^AT_EX 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/hw3
```

THEORETICAL PART

1. Consider a 2-D common-midpoint gather $G(t, x)$, which contains a geometrical event $A_0 f(t - T(x))$ with a constant amplitude A_0 along a parabolic shape

$$T(x) = t_0 + a x^2 . \quad (1)$$

The gather gets transformed by the slant-stack (Radon transform) operator

$$R(\tau, p) = \mathbf{D}_t^{1/2} \int G(\tau + px, x) dx . \quad (2)$$

where $\mathbf{D}_t^{1/2}$ is a waveform-correcting half-order derivative operator.

Using the theory of geometrical integration, show that $R(\tau, p)$ will contain a geometrical event $A_1(p) f(\tau - T_1(p))$. Determine $T_1(p)$ and $A_1(p)$.

2. Consider source and receiver coordinates s and r on the surface of a 2-D constant-velocity medium with velocity V .

- (a) Assuming that the reflection traveltime is T , show that the reflection point $\{x, z\}$ must belong to an ellipse (*migration impulse response*)

$$z(x) = \sqrt{R^2 - \alpha(x - m)^2}, \quad (3)$$

where $R = VT_n/2$, $T_n = \sqrt{T^2 - \frac{4h^2}{V^2}}$, $h = (r - s)/2$, and $m = (r + s)/2$.

- (b) Find α .
- (c) Consider that the ellipse in equation 3 as a reflection surface and apply Fermat's principle to find the reflection traveltime $T_0(x_0)$ for observations with sources and receivers coincident at x_0 .

COMPUTATIONAL PART

1. In the first part, you will experiment with creating and imaging a synthetic seismic reflection dataset.

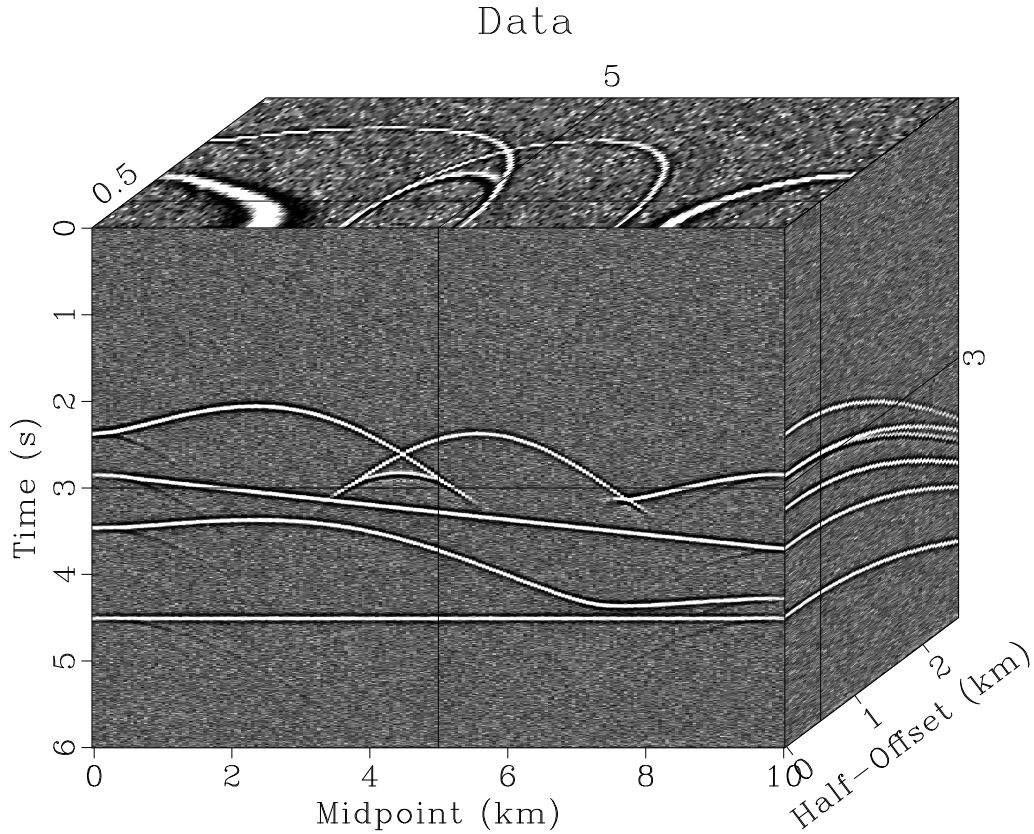


Figure 1: 2-D synthetic data.

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

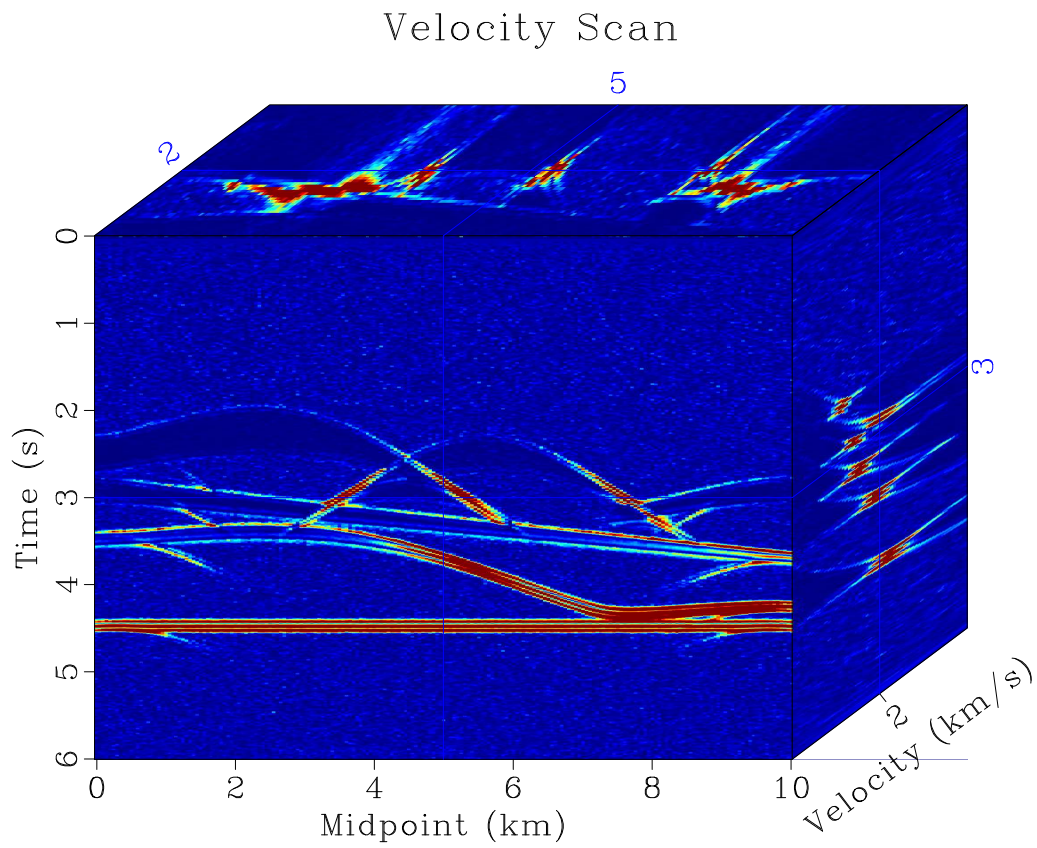
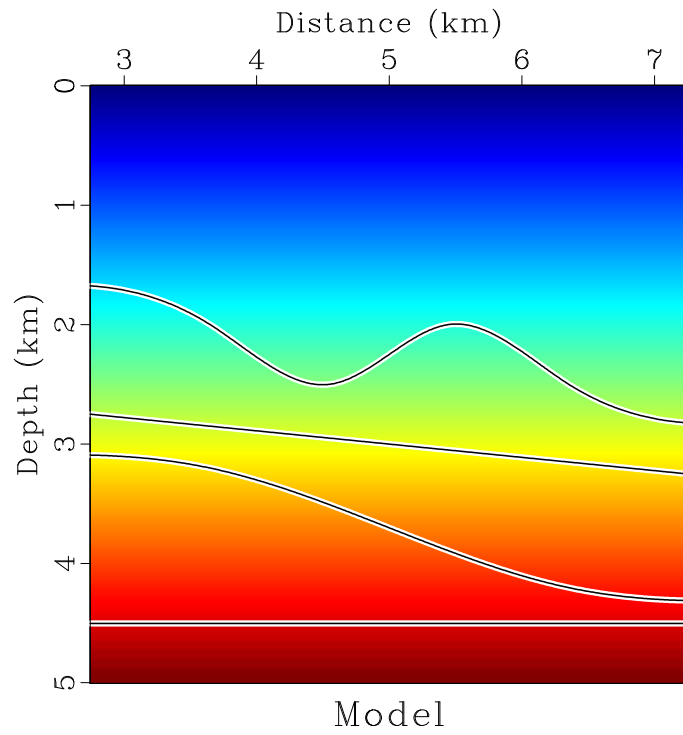


Figure 3: Velocity semblance scan.

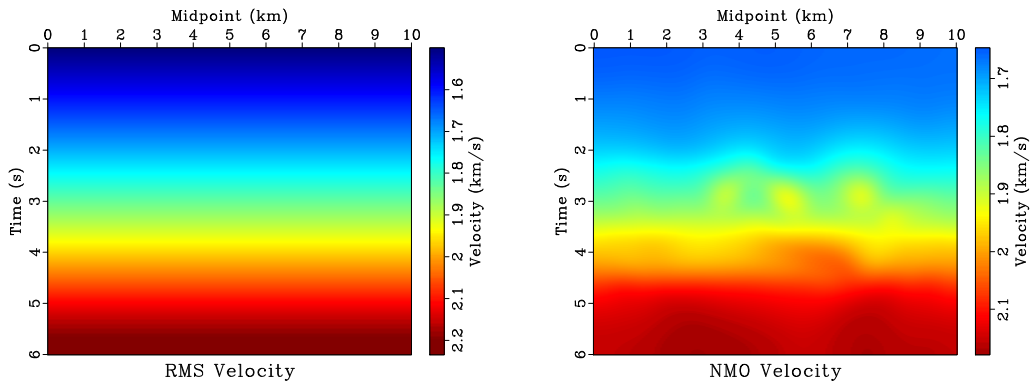


Figure 4: RMS velocity (a) and picked NMO velocity (b).

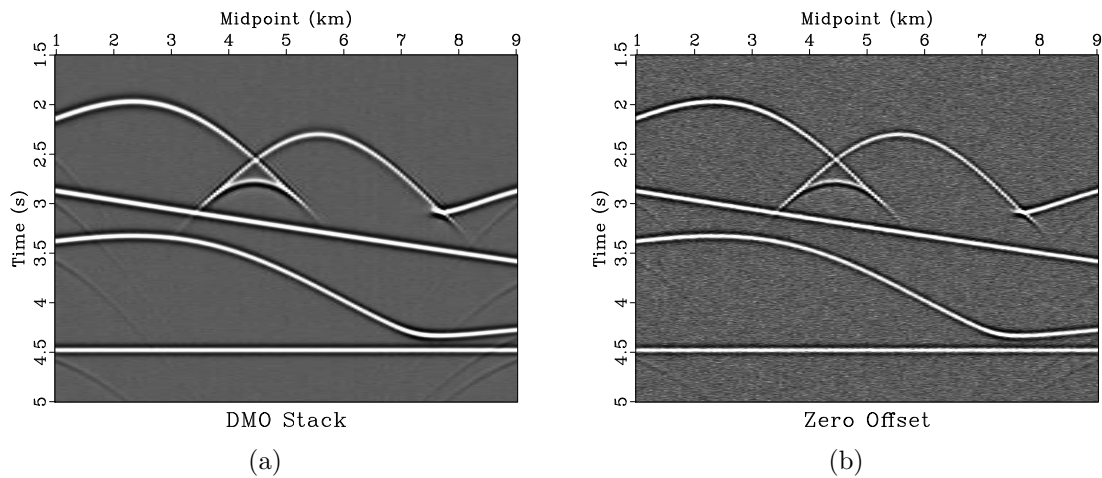


Figure 5: (a) DMO stack. (b) Zero-offset section.

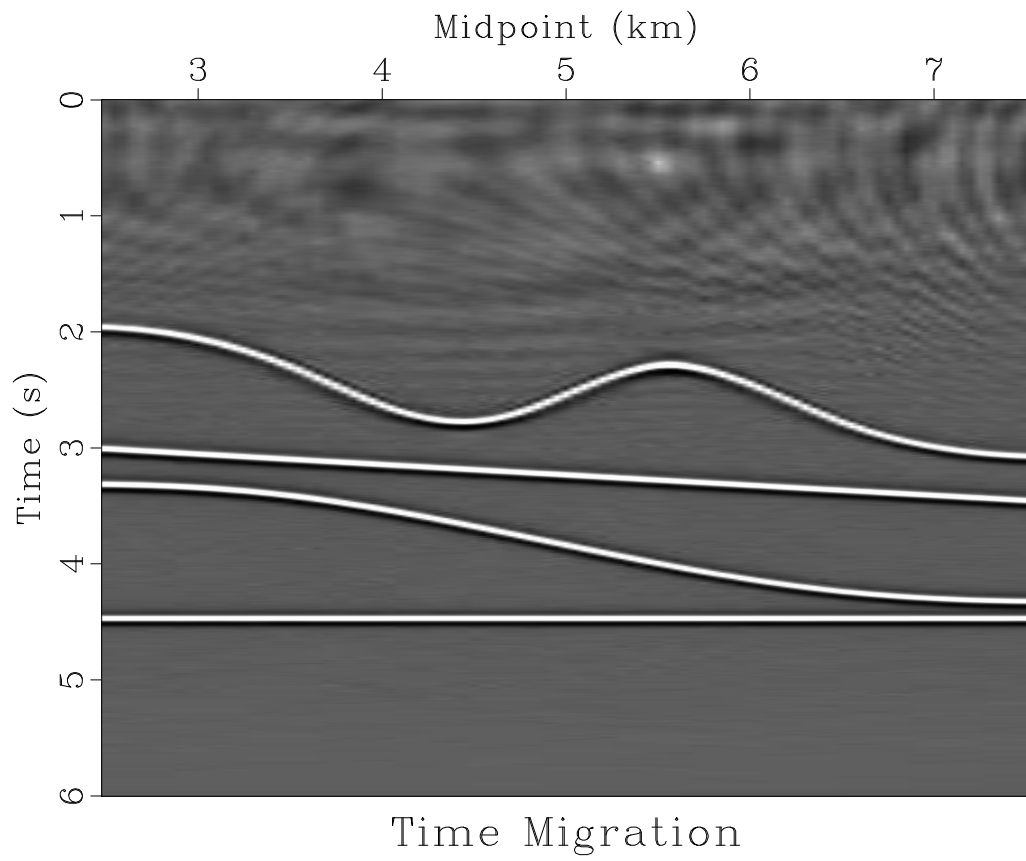


Figure 6: Kirchhoff poststack time migration.

Figure 7: Time migration converted to depth, with reflectors overlaid.

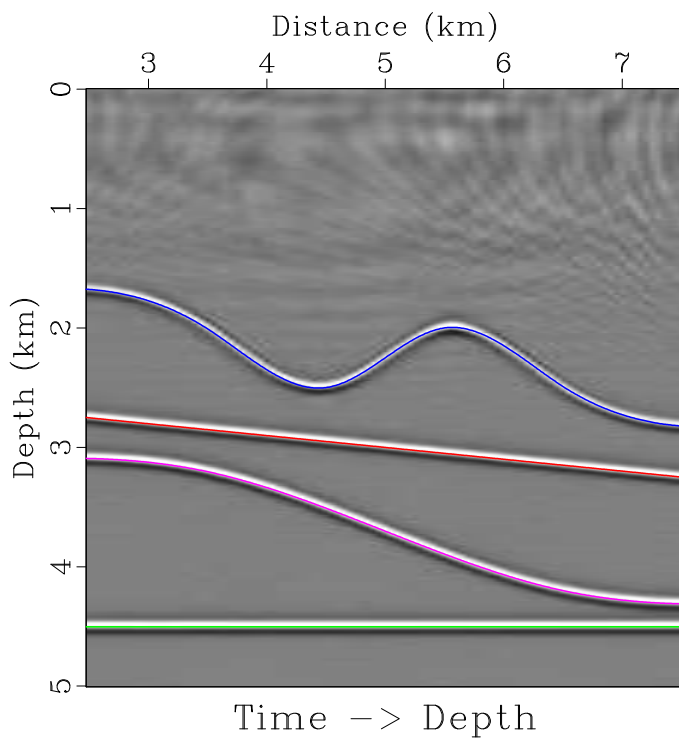


Figure 1 shows a synthetic reflection dataset computed from a reflector model shown in Figure 2 and assuming a velocity model with a constant vertical gradient $V(z) = 1.5 + 0.25z$. A small amount of random noise is added to the synthetic data.

Figure 3 shows a conventional velocity semblance scan. Figure 4 compares the RMS velocity with the NMO velocity picked from the scan. Figure 5 compares a DMO stack and the zero-offset section from the data. Finally, Figure ?? shows the result of Kirchhoff time migration before and after conversion from time to depth.

- (a) Change directory

```
cd hw3/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) Explain the cause of the difference between the RMS velocity and the NMO velocity in Figure 4.
- (d) Edit the `SConstruct` file to switch on the antialiasing parameter in Kirchhoff migration (by changing it from 0 to 1). Generate the migration figures again. What differences do you observe?
- (e) Edit the `kirchhoff.c` file to improve the result of migration. This task is open-ended. The change is up to you, as long as you can achieve an improvement. Possible options:
- Use a more accurate traveltime computation.
 - Introduce an amplitude weight.
 - Change from time to depth migration (you can assume a locally $V(z)$ medium and use results from previous homeworks.)
 - Change from poststack to prestack migration.
- (f) The processing flow in the `SConstruct` file involves some cheating: the exact depth velocity and the exact RMS velocity are used without being estimated from the data. Modify the processing flow so that only properties estimated from the data get used. This task is open-ended as well, different data processing strategies are possible.

```
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=0.01 n2=1001 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 # Velocity estimation
92 #####

```

```

93 Flow( 'voft ', 'vofz ',
94       'depth2time velocity=$SOURCE dt=0.004 nt=1501' )
95 Flow( 'vrms ', 'voft ',
96       ' ',
97       add mode=p $SOURCE | causint |
98       math output="sqrt(input*0.004/(x1+0.004))"
99       ' ')
100
101 # Velocity scan
102 Flow( 'vscan ', 'data ',
103       'vscan v0=1.5 dv=0.02 nv=51 semblance=y ',
104       split=[3,201], reduce='cat' )
105
106 Result( 'vscan ',
107         ' ',
108         byte allpos=y gainpanel=all |
109         transp plane=23 |
110         grey3 flat=n frame1=750 frame2=100 frame3=25
111         label1=Time unit1=s color=j
112         label3=Velocity unit3=km/s
113         label2=Midpoint unit2=km
114         title="Velocity Scan" point1=0.8 point2=0.8
115         ' ')
116
117 # Velocity picking
118 Flow( 'vnmo ', 'vscan ', 'pick rect1=100 rect2=10 | window' )
119
120 for vel in ( 'vrms ', 'vnmo' ):
121     Plot( vel,
122          ' ',
123          grey color=j allpos=y bias=1.5 clip=0.7
124          scalebar=y barreverse=y barunit=km/s
125          label2=Midpoint unit2=km label1=Time unit1=s
126          title="%s Velocity"
127          ' ' % vel[1:].upper() )
128 Result( 'vnmo ', 'vrms vnmo ', 'SideBySideIso' )
129
130 # Stacking
131 #####
132
133 Flow( 'nmo ', 'data vnmo ', 'nmo velocity=${SOURCES[1]} ' )
134 Flow( 'stack ', 'nmo ', 'stack' )
135
136 # Using vrms is CHEATING
137 #####

```

```

138 Flow( 'nmo0', 'data vrms', 'nmo velocity=${SOURCES[1]} ' )
139 Flow( 'dstack', 'nmo0',
140     ' ',
141     window f1=250 |
142     logstretch | fft1 |
143     transp plane=13 memsize=1000 |
144     finstack |
145     transp memsize=1000 |
146     fft1 inv=y | logstretch inv=y |
147     pad beg1=250 | put unit1=s
148     ' ' )
149
150 Flow( 'zoff', 'data', 'window n2=1' )
151
152 stacks = {
153     'stack': 'Stack with NMO Velocity',
154     'dstack': 'DMO Stack',
155     'zoff': 'Zero Offset'
156 }
157
158 for stack in stacks.keys():
159     Result( stack,
160         ' ',
161         window min1=1.5 max1=5 min2=1 max2=9 |
162         grey title="%s"
163         ' ' % stacks[stack])
164
165 # Kirchhoff Migration
166 #####
167
168 prog = Program( 'kirchhoff.c' )
169 exe = str( prog[0] )
170
171 # Using vrms is CHEATING
172 #####
173 Flow( 'tmig', 'dstack %s vrms' % prog[0],
174     './${SOURCES[1]} vel=${SOURCES[2]} antialias=0' )
175
176 Result( 'tmig',
177     ' ',
178     window min2=2.5 max2=7.5 |
179     grey title="Time Migration"
180     ' ' )
181
182 # Using vofz is CHEATING

```

```

183 #####
184 Flow( 'dmig', 'tmig vofz',
185       'time2depth velocity=${SOURCES[1]} ')
186
187 Plot( 'dmig',
188       ', , ,
189       window max1=5 min2=2.5 max2=7.5 |
190       grey title="Time -> Depth" screenratio=1
191       label2=Distance label1=Depth unit1=km
192       ' ' ' )
193
194 Result( 'dmig', 'Overlay' )
195 Result( 'dmig2', 'dmig lays2', 'Overlay' )
196
197 End()

```

```

1  /* 2-D Poststack Kirchhoff time migration. */
2  #include <rsf.h>
3
4  static void doubint(int nt, float *trace)
5  /* causal and anticausal integratio */
6  {
7      int it;
8      float tt;
9
10     tt = trace[0];
11     for (it=1; it < nt; it++) {
12         tt += trace[it];
13         trace[it] = tt;
14     }
15     tt = trace[nt-1];
16     for (it=nt-2; it >=0; it--) {
17         tt += trace[it];
18         trace[it] = tt;
19     }
20 }
21
22 static float pick(float ti, float deltat,
23                  const float *trace,
24                  int nt, float dt, float t0)
25 /* pick a travelttime sample from a trace */
26 {
27     int it, itm, itp;
28     float ft, tm, tp, ftm, ftp, imp;
29

```

```

30     ft = (ti-t0)/dt; it = floorf(ft); ft -= it;
31     if ( it < 0 || it >= nt-1) return 0.0;
32
33     tm = ti-deltat-dt;
34     ftm = (tm-t0)/dt; itm = floorf(ftm); ftm -= itm;
35     if (itm < 0) return 0.0;
36
37     tp = ti+deltat+dt;
38     ftp = (tp-t0)/dt; itp = floorf(ftp); ftp -= itp;
39     if (itp >= nt-1) return 0.0;
40
41     imp = dt/(dt+tp-tm);
42     imp *= imp;
43
44     return imp*(
45         2.*(1-ft)*trace[it] + 2.*ft*trace[it+1] -
46         (1-ftm)*trace[itm] - ftm*trace[itm+1] -
47         (1-ftp)*trace[itp] - ftp*trace[itp+1]);
48 }
49
50 int main(int argc, char* argv[])
51 {
52     int nt, nx, nz, ix, iz, iy, i;
53     float *trace, **out, **v;
54     float x,z, dx, ti, tx, t0,dt, z0,dz, vi,aal;
55     sf_file inp, mig, vel;
56
57     sf_init (argc,argv);
58     inp = sf_input("in");
59     vel = sf_input("vel");
60     mig = sf_output("out");
61
62     if (!sf_histint(inp,"n1",&nt)) sf_error("No n1=");
63     if (!sf_histint(inp,"n2",&nx)) sf_error("No n2=");
64
65     if (!sf_histfloat(inp,"o1",&t0)) sf_error("No o1=");
66     if (!sf_histfloat(inp,"d1",&dt)) sf_error("No d1=");
67     if (!sf_histfloat(inp,"d2",&dx)) sf_error("No d2=");
68
69     if (!sf_getint("nz",&nz)) nz=nt;
70     if (!sf_getfloat("dz",&dz)) dz=dt;
71     if (!sf_getfloat("z0",&z0)) z0=t0;
72
73     if (!sf_getfloat("antialias",&aal)) aal=1.0;
74     /* antialiasing */

```

```

75
76 v = sf_floatalloc2 (nz, nx);
77 sf_floatread (v[0], nz*nx, vel);
78
79 trace = sf_floatalloc (nt);
80 out = sf_floatalloc2 (nz, nx);
81
82 for (i=0; i < nz*nx; i++) {
83     out[0][i] = 0.;
84 }
85
86 for (iy=0; iy < nx; iy++) {
87     sf_floatread (trace, nt, inp);
88     doubint (nt, trace);
89
90     for (ix=0; ix < nx; ix++) {
91         x = (ix-iy)*dx;
92
93         for (iz=0; iz < nz; iz++) {
94             z = z0 + iz*dz;
95             vi = v[ix][iz];
96
97             /* hypot(a, b) = sqrt(a*a+b*b) */
98             ti = hypotf(z, 2.0*x/vi);
99
100            /* tx = |dt/dx| */
101            tx = 4.0*fabsf(x)/(vi*vi*(ti+dt));
102
103            out[ix][iz] +=
104                pick(ti, tx*dx*aal, trace, nt, dt, t0);
105        }
106    }
107 }
108
109 sf_floatwrite (out[0], nz*nx, mig);
110
111 exit (0);
112 }

```

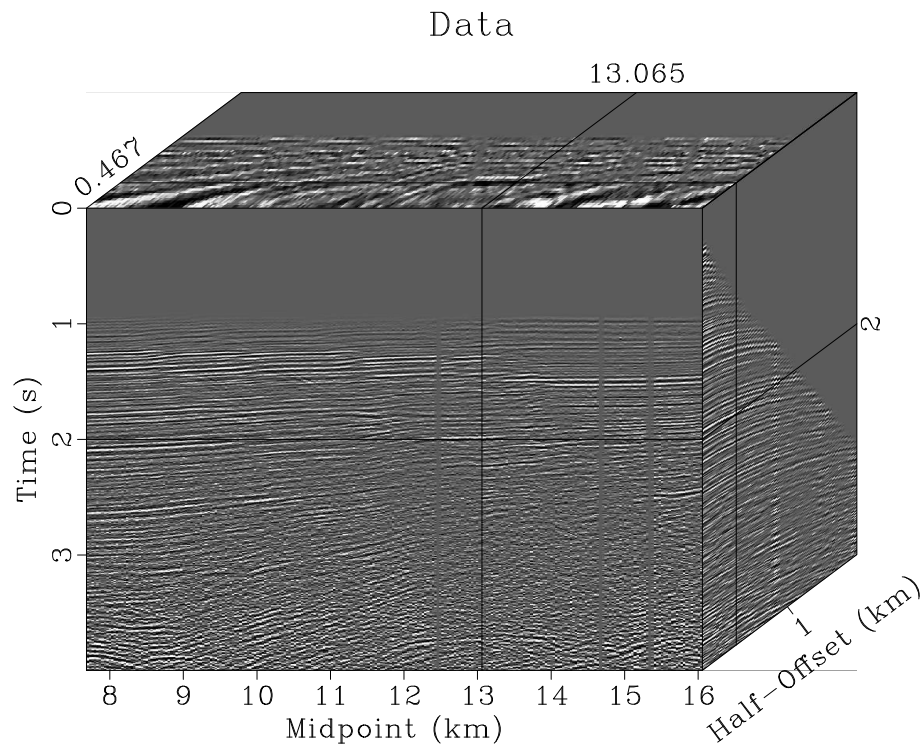


Figure 8: 2-D field dataset from the Gulf of Mexico.

2. In the second part of the computational assignment, you will use the processing strategy developed for synthetic data, to process a 2-D field dataset from the Gulf of Mexico, shown in Figure 8.

- (a) Change directory

```
cd hw3/gulf
```

- (b) Run

```
scons view
```

to generate the figures and display them on your screen.

- (c) Edit the `SConstruct` file to implement your processing strategy. Make sure to select appropriate processing parameters.

```

1 from rsf.proj import *
2
3 Fetch('beinew.HH', 'midpts')
4 Flow('gulf', 'beinew.HH',
5     ', , ,')
6     dd form=native |
7     put
8     label1=Time unit1=s

```

```

9      label2=Half-Offset unit2=km
10     label3=Midpoint unit3=km
11     ''')
12
13 Result('gulf',
14        ', ',
15        byte |
16        transp plane=23 |
17        grey3 flat=n frame1=500 frame2=160 frame3=10
18        title=Data point1=0.8 point2=0.8
19        ''')
20
21 End()

```

COMPLETING THE ASSIGNMENT

1. Change directory to `hw3`.
2. Edit the file `paper.tex` in your favorite editor and change the first line to have your name instead of Kirchhoff's.
3. Run


```
sftour scon lock
```

 to update all figures.
4. Run


```
sftour scon -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.