

Homework 1

Isaac Newton

ABSTRACT

This homework has three parts. In the theoretical part, you will derive some new forms of ray tracing equations and their solutions. In the computational part, you will experiment with wave propagation in a simple synthetic model. In the programming part, you can modify a wave modeling program to implement anisotropic wave propagation.

PREREQUISITES

Completing the computational part of this homework assignment requires

- Madagascar software environment available from <http://www.ahay.org>
- L^AT_EX environment with SEGT_EX available from <http://www.ahay.org/wiki/SEGT_EX>

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
```

The necessary environment is also installed in a computer lab at the Department of Geological Sciences.

THEORETICAL PART

You can either write your answers on paper or edit them in the file `hw1/paper.tex`. Please show all the mathematical derivations that you perform.

1. In class, we used a mysterious parameter σ to represent a variable continuously increasing along a ray. There are other variables that can play a similar role.

- (a) Transform the isotropic ray tracing system

$$\frac{d\mathbf{x}}{d\sigma} = \mathbf{p} \quad (1)$$

$$\frac{d\mathbf{p}}{d\sigma} = S(\mathbf{x}) \nabla S \quad (2)$$

$$\frac{dT}{d\sigma} = S^2(\mathbf{x}) \quad (3)$$

into an equivalent system that uses λ instead of σ , where λ represents the length of the ray trajectory:

$$\frac{d\mathbf{x}}{d\lambda} = \quad (4)$$

$$\frac{d\mathbf{p}}{d\lambda} = \quad (5)$$

$$\frac{dT}{d\lambda} = S(\mathbf{x}) . \quad (6)$$

Remember to check physical dimensions.

- (b) Suppose you are given $T(\mathbf{x})$ – the traveltime from the source to all points \mathbf{x} in the domain of interest. Your task is to find $\lambda(\mathbf{x})$ - the length of the ray trajectory at all \mathbf{x} . Derive a first-order partial differential equation that connects $\nabla\lambda$ and ∇T .
2. The elliptically anisotropic 2-D eikonal equation has the form

$$V_1^2(\mathbf{x}) \left(\frac{\partial T}{\partial x_1} \right)^2 + V_2^2(\mathbf{x}) \left(\frac{\partial T}{\partial x_2} \right)^2 = 1 , \quad (7)$$

where $\mathbf{x} = \{x_1, x_2\}$ is a point in space, $T(\mathbf{x})$ is the traveltime, $V_1(\mathbf{x})$ is the horizontal velocity, and $V_2(\mathbf{x})$ is the vertical velocity.

- (a) Derive the ray tracing system

$$\frac{dx_1}{dT} = \quad (8)$$

$$\frac{dx_2}{dT} = \quad (9)$$

$$\frac{dp_1}{dT} = \quad (10)$$

$$\frac{dp_2}{dT} = \quad (11)$$

where p_1 represents $\partial T/\partial x_1$ and p_2 represents $\partial T/\partial x_2$.

- (b) Assuming constant (but unequal) velocities V_1 and V_2 , solve the ray tracing system for a point source at the origin $x_1(0) = 0$, $x_2(0) = 0$ to show that the wavefronts in this case have an elliptical shape.

(c) The isotropic eikonal equation

$$\left(\frac{\partial T}{\partial x_1}\right)^2 + \left(\frac{\partial T}{\partial x_2}\right)^2 = S^2(\mathbf{x}) \quad (12)$$

describes wavefronts of the wave equation

$$S^2(\mathbf{x}) \frac{\partial^2 P}{\partial t^2} = \nabla^2 P + \dots \quad (13)$$

with omitted possible first- and zero-order terms. What wave equation corresponds to equation (7)?

$$\frac{\partial^2 P}{\partial t^2} = \quad (14)$$

COMPUTATIONAL PART

In this part, we will simulate and observe acoustic wave propagation in a simple velocity model shown in Figure 1(a). A wave snapshot is shown in Figure 1(b).

1. Change directory to `geo384w/hw1/wave`

2. Run

```
scons model.view
```

to generate and view the velocity model from Figure 1(a).

3. Run

```
scons wave.vpl
```

to generate and observe a propagating wave on your screen.

4. Run

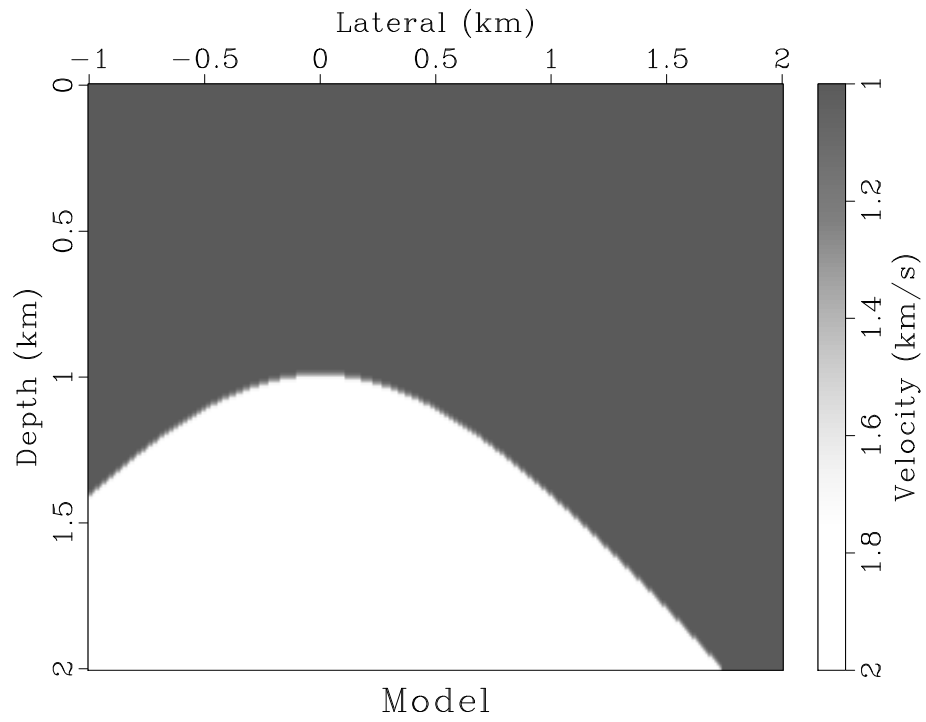
```
scons fronts.vpl
```

to generate and observe a propagating first-arrival wavefront on your screen.

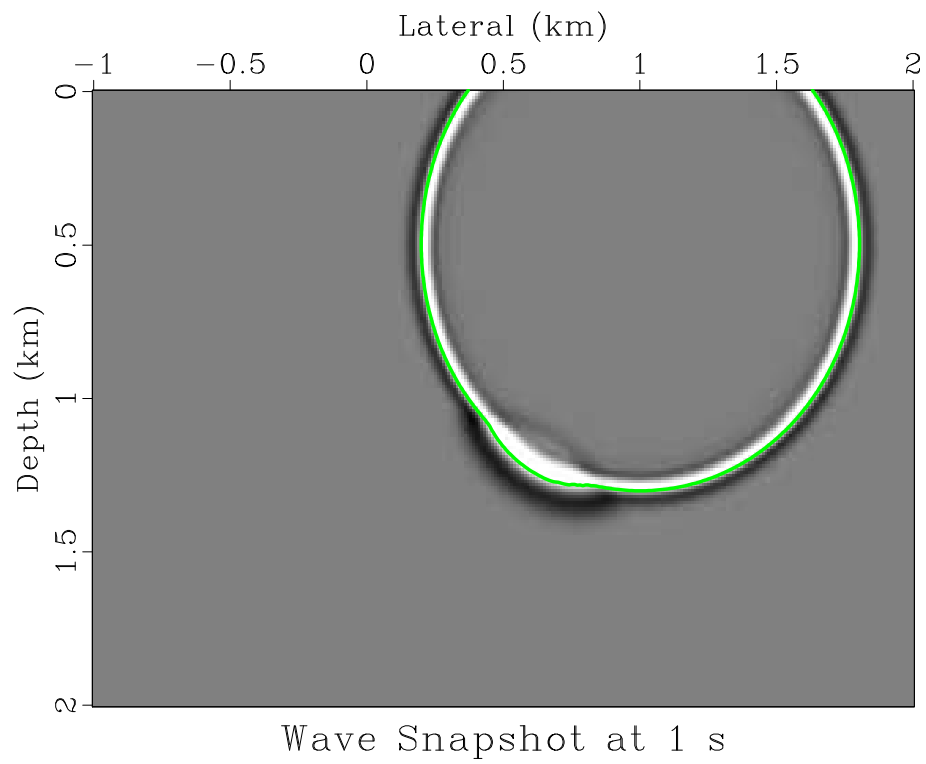
5. Run

```
scons snap.view
```

to generate and view a wave snapshot selected at 1 s as shown in Figure 1(b).



(a)



(b)

Figure 1: (a) Velocity model for simple wave propagation experiments. (b) Wave snapshot with overlaid first-arrival wavefront.

6. Open the file `SConstruct` in your favorite editor. Your task is to make the following modifications in it:

- Find a parameter responsible for selecting the time frame for the snapshot in Figure 1(b). Modify it to increase the time from 1 s to your favorite point in the movie. Run `scons snap.view` again to verify your change.
- Can you observe a geometrical part of the wave that is not captured by the first-arrival wavefront? What is its physical meaning?
- Find a parameter in the `SConstruct` file responsible for the vertical smoothness of the model in Figure 1(a). Modify it to increase the smoothness of the model in such a way that the first-arrival wavefront follows the wave geometry exactly. Run `scons snap.view` again to verify your change.
- (EXTRA CREDIT) For extra credit, modify `SConstruct` to generate a movie of pictures like Figure 1(b) for a gradually increasing smoothness.

```

1 from rsf.proj import *
2
3 # Make a velocity model with a hyperbolic reflector
4 Flow('model',None,
5     ' ',
6     math n1=301 o1=-1 d1=0.01 output="sqrt(1+x1*x1)" |
7     unif2 n1=201 d1=0.01 v00=1,2 |
8     put label1=Depth unit1=km label2=Lateral unit2=km
9     label=Velocity unit=km/s |
10    smooth rect1=3
11    ' ')
12
13 # Plot model
14 Result('model',
15     ' ',
16     grey allpos=y title=Model
17     scalebar=y barreverse=y
18     ' ')
19
20 # Source wavelet
21 Flow('wavelet',None,
22     ' ',
23     spike nsp=1 n1=2000 d1=0.001 k1=201 |
24     ricker1 frequency=10
25     ' ')
26
27 # Extended model (for absorbing boundaries)
28 Flow('left', 'model',
29     ' ',

```

```

30     window n2=1 | spray axis=2 n=50 o=-1.5 d=0.01 |
31     math output="input*exp(-4*(-1-x2)^2)"
32     ' ' ')
33 Flow( 'right' , 'model' ,
34     ' ' ' )
35     window n2=1 f2=300 | spray axis=2 n=50 o=3.01 d=0.01 |
36     math output="input*exp(-4*(3-x2)^2)"
37     ' ' ' )
38 Flow( 'emodel2' , 'left model right' ,
39     'cat axis=2 ${SOURCES[1:3]} ' )
40
41 Flow( 'top' , 'emodel2' ,
42     ' ' ' )
43     window n1=1 | spray axis=1 n=50 o=-0.5 d=0.01 |
44     math output="input*exp(-4*x1^2)"
45     ' ' ' )
46 Flow( 'bottom' , 'emodel2' ,
47     ' ' ' )
48     window n1=1 f1=200 | spray axis=1 n=50 o=2.01 d=0.01 |
49     math output="input*exp(-4*(2-x1)^2)"
50     ' ' ' )
51 Flow( 'emodel' , 'top emodel2 bottom' ,
52     'cat axis=1 ${SOURCES[1:3]} ' )
53
54 # Source location
55 Flow( 'source' , None ,
56     ' ' ' )
57     spike k1=101 k2=251
58     n2=401 o2=-1.5 d2=0.01 label1=Depth unit1=km
59     n1=301 o1=-0.5 d1=0.01 label2=Lateral unit2=km
60     ' ' ' )
61
62 # Modeling
63 exe = Program( 'wave.c' )
64 Flow( 'wave' , 'source %s wavelet emodel' % exe[0] ,
65     ' ' ' )
66     ./${SOURCES[1]} wav=${SOURCES[2]} vel1=${SOURCES[3]} vel2=${SOURCES[3]}
67     jt=5 ft=200
68     ' ' ' )
69
70 # Movie of wave snapshots
71 Plot( 'wave' ,
72     ' ' ' )
73     window f1=50 f2=50 n1=201 n2=301 |
74     grey gainpanel=all title=Wave

```

```

75     ' ', view=1)
76
77 # Favorite time moment
78 #####
79 time = 1.0 # !!!!!!! MODIFY ME
80 #####
81
82 # Wavefield snapshot
83 Plot('snap', 'wave',
84     ' ',
85     window f1=50 f2=50 n1=201 n2=301 n3=1 min3=%g |
86     grey title="Wave Snapshot at %g s"
87     label1=Depth unit1=km label2=Lateral unit2=km
88     ' ' % (time, time))
89
90 # First-arrival traveltimes
91 Flow('first', 'model',
92     'eikonal yshot=1 zshot=0.5 | add add=0.2')
93
94 # Movie of first-arrival wavefronts
95 fronts = []
96 for snap in range(180):
97     front = 'front%d' % snap
98     fronts.append(front)
99     tsnap = 0.2+snap*0.01
100    Plot(front, 'first',
101        'contour nc=1 c0=%g title="%g s" ' % (tsnap, tsnap))
102    Plot('fronts', fronts, 'Movie', view=1)
103
104 # First-arrival wavefront
105 Plot('front', 'first',
106     ' ',
107     contour nc=1 c0=%g wanttitle=n wantaxis=n
108     plotcol=3 plotfat=5
109     ' ' % time)
110
111 # Overlay wavefront and traveltimes
112 Result('snap', 'snap front', 'Overlay')
113
114 End()

```

PROGRAMMING PART (EXTRA CREDIT)

For extra credit, you can modify the wave modeling program to include anisotropic wave propagation effects. The program below (slightly modified from the original version by Paul Sava) implements wave modeling with equation

$$\frac{1}{V^2(\mathbf{x})} \frac{\partial^2 P}{\partial t^2} = \nabla^2 P + F(\mathbf{x}, t), \quad (15)$$

where $F(\mathbf{x}, t)$ is the source term. The implementation uses finite-difference discretization (second-order in time and fourth-order in space). Stepping in time involves the following computations:

$$\mathbf{P}_{t+\Delta t} = [\nabla^2 \mathbf{P}_t + F(\mathbf{x}, t)] V^2(\mathbf{x}) \Delta t^2 + 2\mathbf{P}_t - \mathbf{P}_{t-\Delta t}, \quad (16)$$

where \mathbf{P} represents the propagating wavefield discretized at different time steps.

```

1  /* 2-D finite-difference acoustic wave propagation */
2  #include <stdio.h>
3
4  #include <rsf.h>
5
6  static int n1, n2;
7  static float c0, c11, c21, c12, c22;
8
9  static void laplacian(float **uin /* [n2][n1] */,
10                      float **uout /* [n2][n1] */)
11  /* Laplacian operator, 4th-order finite-difference */
12  {
13      int i1, i2;
14
15      for (i2=2; i2 < n2-2; i2++) {
16          for (i1=2; i1 < n1-2; i1++) {
17              uout[i2][i1] =
18                  c11*(uin[i2][i1-1]+uin[i2][i1+1]) +
19                  c12*(uin[i2][i1-2]+uin[i2][i1+2]) +
20                  c21*(uin[i2-1][i1]+uin[i2+1][i1]) +
21                  c22*(uin[i2-2][i1]+uin[i2+2][i1]) +
22                  c0*uin[i2][i1];
23          }
24      }
25  }
26
27  int main(int argc, char* argv[])
28  {
29      int it, i1, i2;          /* index variables */
30      int nt, n12, ft, jt;

```

```

31 float dt ,d1 ,d2 ,dt2;
32
33 float **ww,**v1,**v2,**rr; /* I/O arrays*/
34 float **u0,**u1,**u2,**ud; /* tmp arrays */
35
36 sf_file Fw,Fv1,Fv2,Fr,Fo,Fd; /* I/O files */
37 sf_axis at ,a1 ,a2; /* cube axes */
38
39 sf_init (argc ,argv);
40
41 /* setup I/O files */
42 Fr = sf_input ("in"); /* source position */
43 Fo = sf_output ("out"); /* output wavefield */
44
45 Fw = sf_input ("wav"); /* source wavelet */
46 Fv1 = sf_input ("vel1"); /* vertical velocity */
47 Fv2 = sf_input ("vel2"); /* horizontal velocity */
48
49 /* Read/Write axes */
50 at = sf_iaxa (Fw,1); nt = sf_n (at); dt = sf_d (at);
51 a1 = sf_iaxa (Fr,1); n1 = sf_n (a1); d1 = sf_d (a1);
52 a2 = sf_iaxa (Fr,2); n2 = sf_n (a2); d2 = sf_d (a2);
53 n12 = n1*n2;
54
55 if (!sf_getint ("ft",&ft)) ft=0;
56 /* first recorded time */
57 if (!sf_getint ("jt",&jt)) jt=1;
58 /* time interval */
59
60 sf_putint (Fo,"n3" ,(nt-ft)/jt);
61 sf_putfloat (Fo,"d3" ,jt*dt);
62 sf_putfloat (Fo,"o3" ,ft*dt);
63
64 dt2 = dt*dt;
65
66 /* set Laplacian coefficients */
67 d1 = 1.0/(d1*d1);
68 d2 = 1.0/(d2*d2);
69
70 c11 = 4.0*d1/3.0;
71 c12= -d1/12.0;
72 c21 = 4.0*d2/3.0;
73 c22= -d2/12.0;
74 c0 = -2.0 * (c11+c12+c21+c22);
75

```

```

76  /* read wavelet, velocity & source position */
77  rr=sf_floatalloc2(n1,n2); sf_floatread(rr[0],n12,Fr);
78  ww=sf_floatalloc(nt);    sf_floatread(ww ,nt ,Fw);
79  v1=sf_floatalloc2(n1,n2); sf_floatread(v1[0],n12,Fv1);
80  v2=sf_floatalloc2(n1,n2); sf_floatread(v2[0],n12,Fv2);
81
82  /* allocate temporary arrays */
83  u0=sf_floatalloc2(n1,n2);
84  u1=sf_floatalloc2(n1,n2);
85  u2=sf_floatalloc2(n1,n2);
86  ud=sf_floatalloc2(n1,n2);
87
88  for (i2=0; i2<n2; i2++) {
89      for (i1=0; i1<n1; i1++) {
90          u0[i2][i1]=0.0;
91          u1[i2][i1]=0.0;
92          u2[i2][i1]=0.0;
93          ud[i2][i1]=0.0;
94          v1[i2][i1] *= v1[i2][i1]*dt2;
95      }
96  }
97
98  /* Time loop */
99  for (it=0; it<nt; it++) {
100     laplacian(u1,ud);
101
102     for (i2=0; i2<n2; i2++) {
103         for (i1=0; i1<n1; i1++) {
104             /* inject wavelet */
105             ud[i2][i1] += ww[it] * rr[i2][i1];
106             /* scale by velocity */
107             ud[i2][i1] *= v1[i2][i1];
108             /* time step */
109             u2[i2][i1] =
110                 2*u1[i2][i1]
111                 - u0[i2][i1]
112                 + ud[i2][i1];
113
114             u0[i2][i1] = u1[i2][i1];
115             u1[i2][i1] = u2[i2][i1];
116         }
117     }
118
119     /* write wavefield to output */
120     if (it >= ft && 0 == (it-ft)%jt) {

```

```

121         sf_warning("%d;", it+1);
122         sf_floatwrite(u1[0], n12, Fo);
123     }
124 }
125 sf_warning(".");
126
127 exit (0);
128 }

```

Your task is to modify the code to implement your anisotropic equation (14). You will test your implementation using a constant velocity example shown in Figure 2.

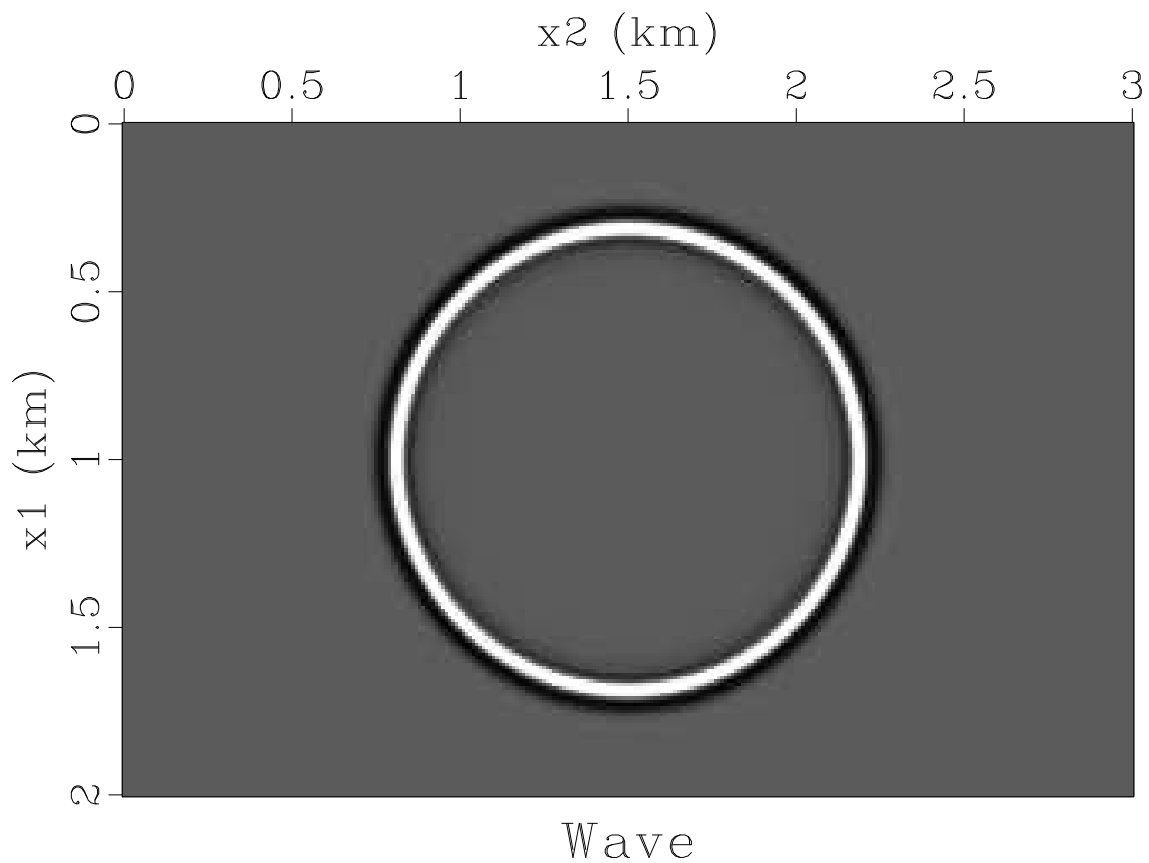


Figure 2: Wavefield snapshot for propagation from a point-source in a homogeneous medium. Modify the code to make wave propagation anisotropic.

1. Change directory to `geo384w/hw1/code`
2. Run

```
scons wave.vpl
```

to compile and run the program and to observe a propagating wave on your screen.

3. Open the file `wave.c` in your favorite editor and modify it to implement the wave operator from equation (14).

4. Run

```
scons wave.vpl
```

again to compile and test your program. If you want to add additional tests, modify the file `SConstruct`.

```

1 from rsf.proj import *
2
3 # Program compilation
4 #####
5
6 exe = Program( 'wave.c' )
7
8 # Constant velocity test
9 #####
10
11 # Source wavelet
12 Flow( 'wavelet', None,
13       ' ',
14       spike nsp=1 n1=1000 d1=0.001 k1=201 |
15       ricker1 frequency=10
16       ' ')
17
18 # Source location
19 Flow( 'source', None,
20       ' ',
21       spike n1=201 n2=301 d1=0.01 d2=0.01
22       label1=x1 unit1=km label2=x2 unit2=km
23       k1=101 k2=151
24       ' ')
25
26 # Velocity model
27 Flow( 'v1', 'source', 'math output=1' )
28 Flow( 'v2', 'source', 'math output=1.5' )
29
30 # Modeling
31 Flow( 'wave', 'source %s wavelet v1 v2' % exe[0],
32       ' ',

```

```

33     ./${SOURCES[1]} wav=${SOURCES[2]}
34     vel1=${SOURCES[3]} vel2=${SOURCES[4]}
35     ' ' ')
36
37 Plot('wave',
38     ' ' '
39     window j3=5 f3=200 |
40     grey gainpanel=all title=Wave
41     ' ' ',view=1)
42
43 Result('wave',
44     ' ' '
45     window n3=1 min3=0.9 |
46     grey title=Wave screenht=8 screenwd=12
47     ' ' ')
48
49 End()

```

COMPLETING THE ASSIGNMENT

1. Change directory to `geo384w/hw1`.
2. Edit the file `paper.tex` in your favorite editor and change the first line to have your name instead of Newton'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.