

In this homework, you will use a finite-differences modeling code, similar to the one you wrote in the preceding homework, to implement basic reverse time migration. I do not expect you to be concerned with the efficiency of your implementation at this time. This implementation of reverse-time migration does not require that you write any new C code. You will use pre-existing Madagascar programs, but you will modify the `SConstruct` file to combine those programs.

**This is an individual assignment and absolutely no collaboration on code is allowed.**

Figure 1: Velocity.

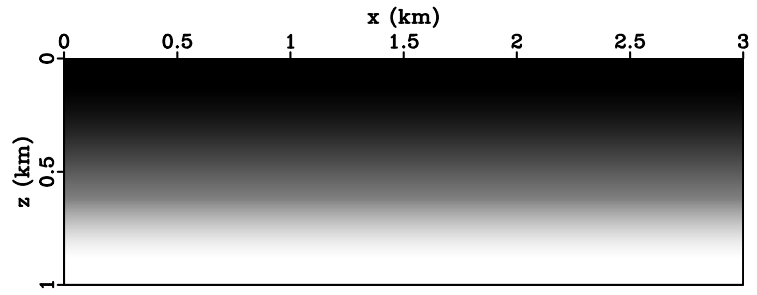


Figure 2: Density.

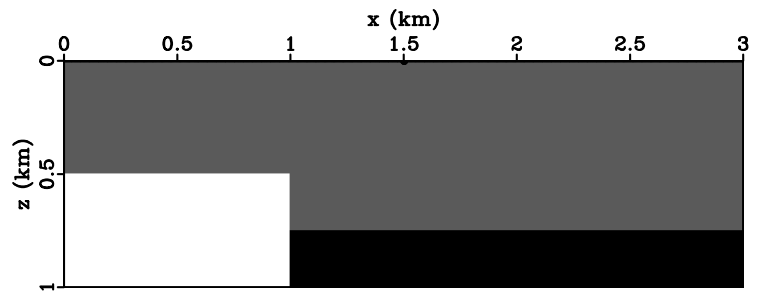
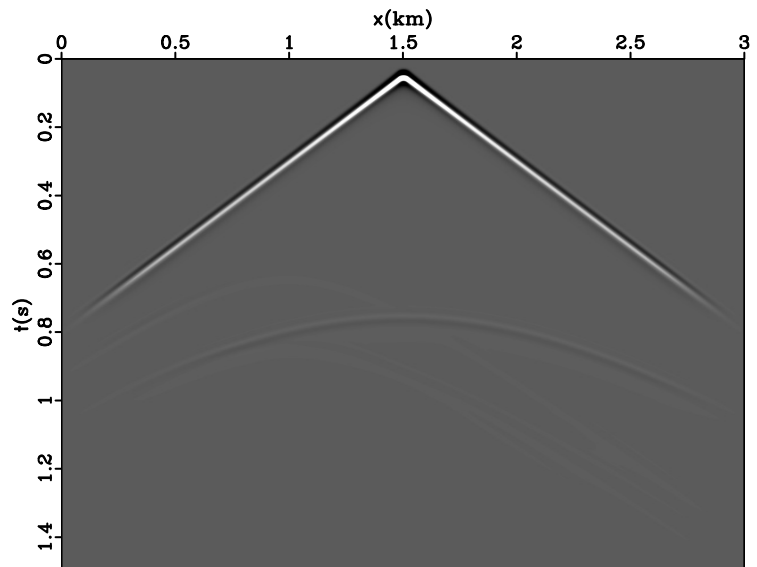


Figure 3: Data w/ direct arrival.



## EXERCISE

Using the finite-differences modeling function `awefd`, construct an image of the subsurface. This function takes the following parameters:

Figure 4: Image.

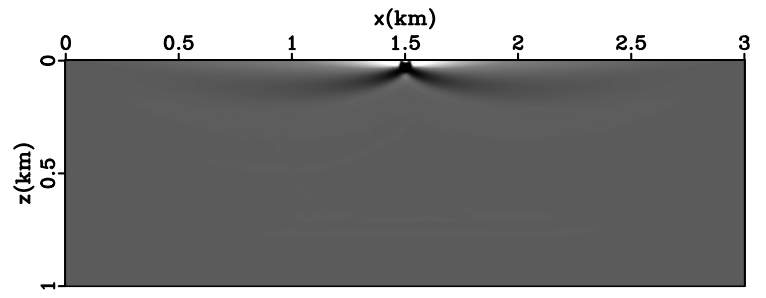


Figure 5: Data w/o direct arrival.

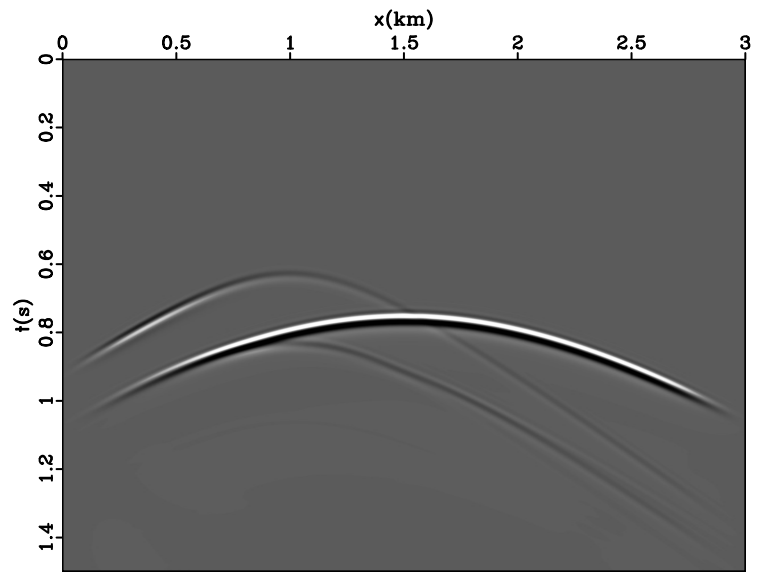
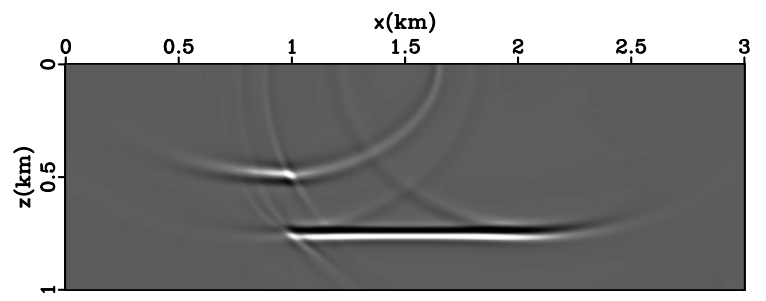


Figure 6: Image.



`awefd(odat, owfl, idat, velo, dens, sou, rec, custom, par)`

- `odat`: output data  $d(x, t)$
- `owfl`: output wavefield  $u(z, x, t)$
- `idat`: input data (wavelet)
- `velo`: velocity model  $v(z, x)$
- `dens`: density model  $\rho(z, x)$
- `sou`: source coordinates
- `rec`: receiver coordinates
- `custom`: custom parameters
- `par`: parameter dictionary

Design an imaging procedure following the generic scheme developed in class. Your task is to identify Madagascar programs necessary to implement reverse-time migration in two different ways and generate the appropriate `Flows` in the `SConstruct`. Explain in detail how your imaging procedures work.

1. Use your imaging procedure to generate images based on recorded data in Figures 3 and 5. For this exercise, use the constant density `rb.rsx` for imaging. Include those two images in this document. Are the images different from each-other? How? Why?
2. Use your imaging procedure to generate images based on recorded data in Figures 3 and 5. For this exercise, use the variable density `ra.rsx` for imaging. Include those two images in this document. Are the images different from each-other? How? Why? How do your images compare with the ones from the preceding exercise?

### WRAP-UP

After you are satisfied that your document looks ok, print it from the PDF viewer and bring it to class.

# SCONSTRUCT

```
##
# GPCN 658 — reverse—time migration
##
from rsf.proj import *
import fdm

# -----
par = dict(
    nt=1500, ot=0, dt=0.001, lt='t', ut='s',
    nx=601, ox=0, dx=0.005, lx='x', ux='km',
    nz=201, oz=0, dz=0.005, lz='z', uz='km',
    kt=50, nb=100, jsnap=50, jdata=1, frq=35
)
fdm.param(par)

par['xk']=50
par['xl']=par['nx']-50

par['xsou']=par['ox']+par['nx']/2*par['dx']
par['zsou']=par['oz']

# -----
# wavelet
fdm.wavelet('wav',par['frq'],par)
Flow('wav','wav','transp')
Result('wav','window n2=500 |' + fdm.waveplot('',par))

# -----
# sources coordinates
fdm.point('ss',par['xsou'],par['zsou'],par)
Plot('ss',fdm.ssplot('',par))

# receivers coordinates
fdm.horizontal('rr',0,par)
Plot('rr',fdm.rrplot('',par))

# -----
# velocity
Flow('vo',None,
    '''
    math output="2.0+0.25*xl"
    n1=%%(nz)d o1=%%(oz)g d1=%%(dz)g
    n2=%%(nx)d o2=%%(ox)g d2=%%(dx)g
    ''' % par)

Plot('vo',fdm.cgrey('allpos=y bias=2.0 pclip=100',par))
Result('vo',['vo','ss','rr'],'Overlay')

# -----
# density
Flow('ra',None,
    '''
    spike nsp=2 mag=+0.5,-0.5
    n1=%%(nz)d o1=%%(oz)g d1=%%(dz)g k1=101,151 l1=%%(nz)d,%%(nz)d
    n2=%%(nx)d o2=%%(ox)g d2=%%(dx)g k2=1,201 l2=200,%%(nx)d |
    add add=2
    ''' % par)
Plot('ra',fdm.cgrey('allpos=y bias=1.5 pclip=100',par))
Result('ra',['ra','ss','rr'],'Overlay')

Flow('rb','ra','math output=1')

# -----
# edge taper
Flow('taper',None,
    '''
    spike nsp=1 mag=1
    n1=%%(nx)d d1=%%(dx)g o1=%%(ox)g k1=%%(xk)d l1=%%(xl)d
    n2=%%(nt)d d2=%%(dt)g o2=%%(ot)g |
    smooth rectl=50
    ''' % par)
Result('taper','transp |'+fdm.dgrey('pclip=99',par))

# -----
# finite—differences modeling
fdm.awefd('dd','ww','wav','vo','ra','ss','rr','jsnap=1 fsrf=n',par)
fdm.awefd('do','wo','wav','vo','rb','ss','rr','jsnap=1 fsrf=n',par)

Result('ww','window j3=%%(jsnap)d |'%par + fdm.wgrey('pclip=99.9',par))
Result('wo','window j3=%%(jsnap)d |'%par + fdm.wgrey('pclip=99.9',par))

# data w/ direct arrivals
Flow('dr0','dd taper',
    'add mode=p ${SOURCES[1]}')

# data w/o direct arrivals
Flow('dr1','dd do taper',
    'math r=${SOURCES[0]} d=${SOURCES[1]} t=${SOURCES[2]} output="(r-d)*t"')

for j in range(2):
    dtag="d%d"%j
    Result('dr'+dtag,'transp |' + fdm.dgrey('pclip=99.9',par))

denss = ['rb','rb']
swfls = ['ww','wo']
idats = ['dr0','dr1']
imags = ['imag0','imag1']
odats = ['odat0','odat1']
tdats = ['tdat0','tdat1']
twfls = ['twfl0','twfl1']
rwfls = ['rwfl0','rwfl1']
velo,sou,rec,custom='vo','rr','rr','jsnap=1 fsrf=n'
for i in range(2):
    Flow(odats[i],idats[i],'reverse which=2 opt=i verb=y')
    fdm.awefd(tdats[i],twfls[i],odats[i],velo,denss[i],sou,rec,custom,par)
    Flow(rwfls[i],twfls[i],'reverse which=4 opt=i verb=y')
    Flow(imags[i],rwfls[i]+' '+swfls[i],'xcor2d uu=${SOURCES[1]} axis=3 verb=y nbuf=100')
    Result(imags[i],'window j3=%%(jsnap)d |'%par + fdm.wgrey('pclip=99.9',par))
# Reverse—time migration
def rtm(sw,idat,velo,dens,sou,rec,custom,imag):
```

```

Flow('rdat', idat, 'reverse which=2 opt=i verb=y')
fdm.awefd('odat', 'tw', 'rdat', 'velo', 'dens', 'sou', 'rec', 'custom', par)
Flow('rw', 'tw', 'reverse which=4 opt=i verb=y')
Flow(imag, 'rw' + ' ' + sw, 'xcor2d uu=${SOURCES[1]} axis=3 verb=y nbuf=100')
Result(imag, 'window j3=%(jsnap)d |' % par + 'fdm.wgrey('pclip=99.9', par))
# Reverse-time migration
Flow('tdat', 'dr0', 'reverse which=2 opt=i verb=y')
fdm.awefd('dt', 'tw', 'tdat', 'vo', 'ra', 'rr', 'rr', 'jsnap=1 fsrf=n', par)
Flow('rw', 'tw', 'reverse which=4 opt=i verb=y')
Flow('imag0', 'rw ww', 'xcor2d uu=${SOURCES[1]} axis=3 verb=y nbuf=100')
Result('imag0', 'window j3=%(jsnap)d |' % par + 'fdm.wgrey('pclip=99.9', par))
# Reverse-time migration
def rtm(imag, sdat, rdat, velo, dens, sacq, racq, iacq, custom, par):

    swfl = imag + '.us' # source wavefield
    rwfl = imag + '.ur' # receiver wavefield
    twfl = imag + '.ut' # temp wavefield
    sout = imag + '.ds' # source data (not the input sdat!)
    rout = imag + '.dr' # receiver data (not the input rdat!)

    iwindow = ' ' + \
    ...

    nq1=%(nqz)d oq1=%(oqz)g
    nq2=%(nqx)d oq2=%(oqx)g
    jsnap=%(jdata)d jdata=%(jdata)d
    ... % par + ' '

    # source wavefield (z,x,t)
    fdmod.awefd(sout, swfl, sdat, velo, dens, sacq, iacq, custom+iwindow, par)

    # receiver wavefield (z,x,t)
    tdat = imag + '.tds'
    Flow(tdat, rdat, 'reverse which=2 opt=i verb=y')
    fdmod.awefd(rout, twfl, tdat, velo, dens, racq, iacq, custom+iwindow, par)
    Flow(rwfl, twfl, 'reverse which=4 opt=i verb=y')

    # conventional (cross-correlation zero-lag) imaging condition
    Flow(imag, [swfl, rwfl], 'xcor2d uu=${SOURCES[1]} axis=3 verb=y nbuf=100')

# -----
##
# Here add rules for your assignment.
#
# use Flow()
# use Result()
#
# find Madagascar programs using the command "sfdoc -k ."
# find the documentation of Madagascar programs by typing the program name
##

End()

```