seismic imaging tutorial

“exploding reflector” modeling/migration

Paul Sava

Center for Wave Phenomena
Colorado School of Mines
psava@mines.edu

assignment

modify the acquisition parameters to explore the illumination at different locations in the subsurface

import packages

```
from rsf.proj import *
import sigsbee
import rsf.recipes.fdmod as fdmod
```

import packages

- `sigsbee`: model-specific python module
- `fdmod`: generic modeling and plotting module

setup main parameters

```
par=sigsbee.paramwin() # Sigsbee2A parameters
par['nt']=2001         # time steps (samples)
par['kt']=100          # wavelet delay (samples)
par['dt']=0.001        # time sampling (ms)
par['nb']=100          # boundary (grid points)
fdmod.param(par)       # plotting parameters
```

setup main parameters

- `par`: dictionary containing all parameters
- override parameters in the `SConstruct`

source coordinates

```
# source coordinates (exploding reflectors)
fdmod.boxarray('ss',5,2,1,12,8,1,par)

# plot sources
Plot('ss',fdmod.ssplot('plotfat=10 symbol=.',par))
```

source coordinates

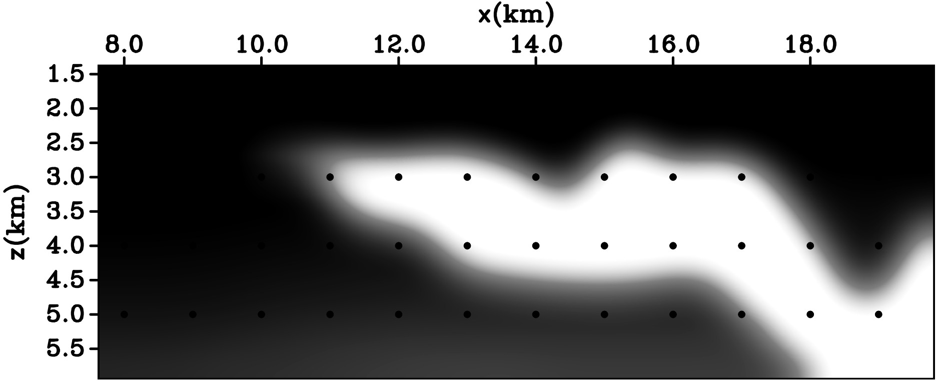
- `5,2,1`: *n*, *o*, *d* in the z direction
- `12,8,1`: *n*, *o*, *d* in the x direction

receiver coordinates

```
par['jr']=4    # receiver jump (grid points)
par['nr']=100  # number of receivers
par['fr']=500  # receivers origin (grid points)

# receiver coordinates
fdmod.horizontal('tt',par['oz']+par['dz'],par)
Flow('rr',
     'tt',
     'window n2=%(nr)d j2=%(jr)d f2=%(fr)d'%par)

# plot receivers
Plot('rr',fdmod.rrplot('plotfat=10',par))
```

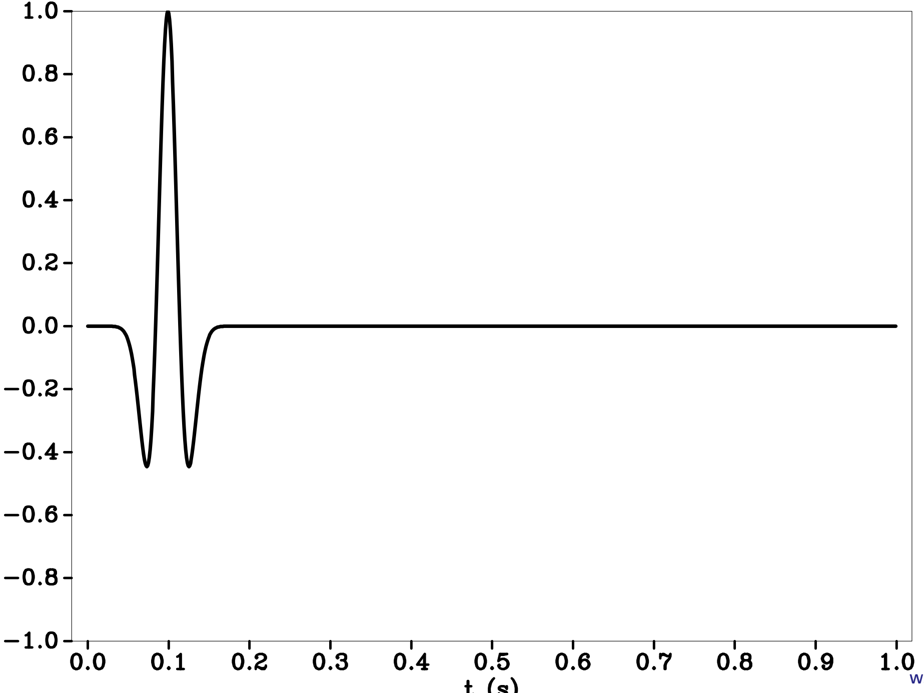
velocity/density models

```
# get velocity
sigsbee.getstrvelwin('vstr',par)
Flow(   'velo',
        'vstr',
        'smooth rect1=100 rect2=100 repeat=1')

# plot velocity
Plot(   'velo',fdmod.cgrey('allpos=y bias=1.43',par)
Result('velo',['velo','ss','rr'],'Overlay')

# density
Flow('dens','velo','math output=1')
```


x(km)
8.0
10.0
12.0
14.0
16.0
18.0
z(km)
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

source wavelet

```
# construct wavelet
fdmod.wavelet('wav_',15,par)

# transpose wavelet
Flow(   'wav','wav_','transp')

# plot wavelet
Result('wav','window n2=1000 |'
        + fdmod.waveplot('',par))
```



WSI

FD modeling

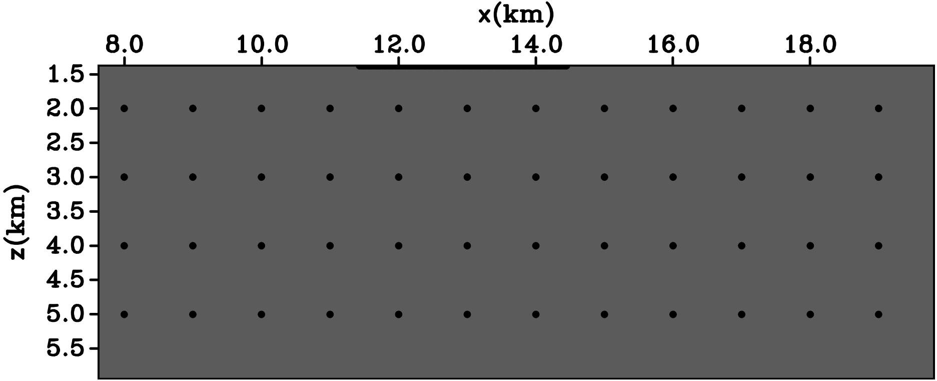
```
# run FD modeling
fdmod.awefd1('temp','wfld',
             'wav','velo','dens',
             'ss','rr',
             'free=n',par)
```

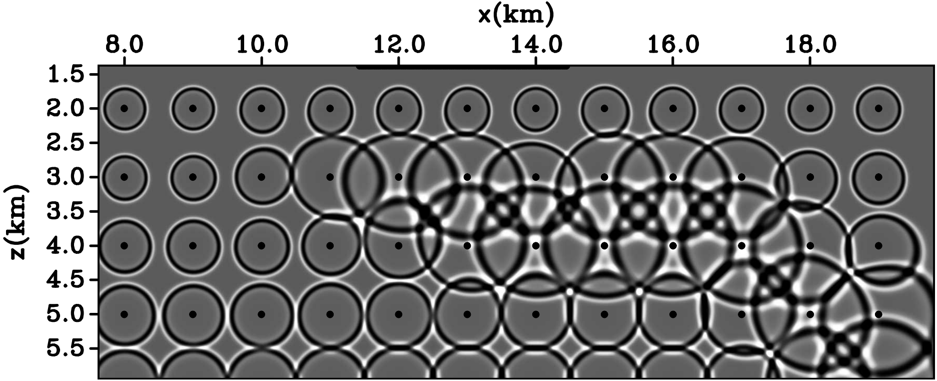
FD modeling parameters

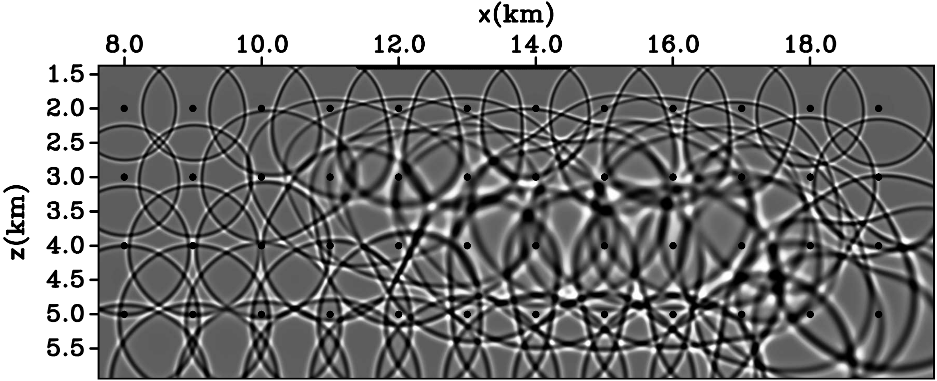
data,
wavefield,
wavelet,
velocity,
density,
source coordinates,
receiver coordinates,
optional parameters,
parameter dictionary

plot wavefield

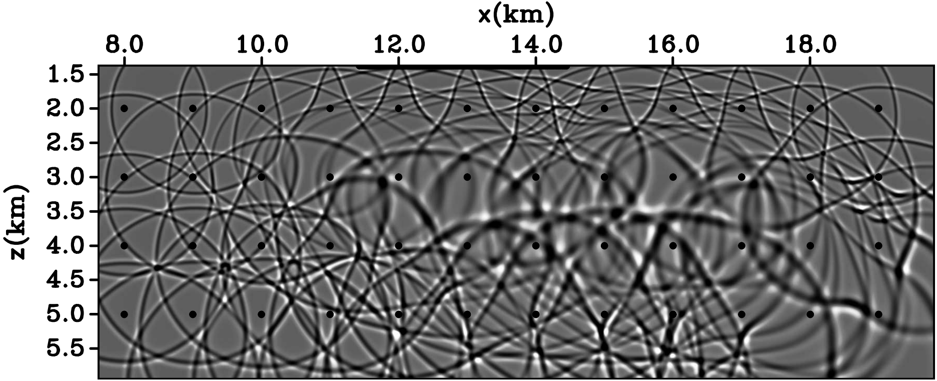
```
# generate wavefield movie
Plot('wfld',fdmod.wgrey('pclip=99',par),view=1)

# plot wavefield frames
for i in range(7):
    tag = '-%02d' %(i)
    fdmod.wframe('wfld'+tag,
                 'wfld',i,'pclip=99',par)
    Result( 'wfld'+tag,
           ['wfld'+tag,'ss','rr'],'Overlay')
```

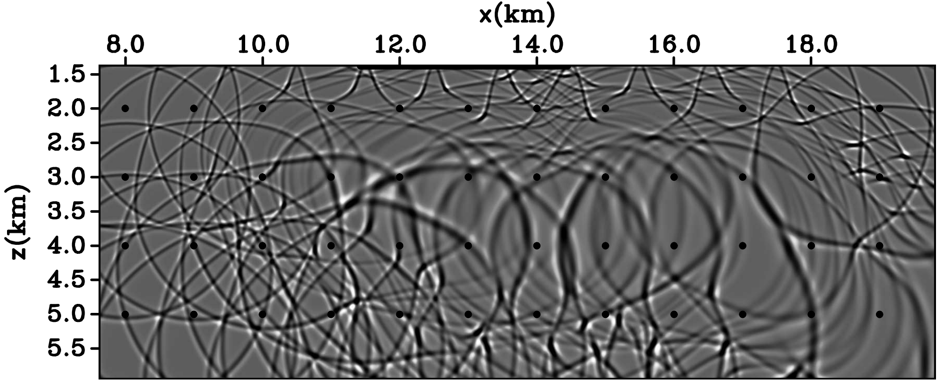

x(km)
8.0
10.0
12.0
14.0
16.0
18.0
z(km)
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5


x(km)
8.0
10.0
12.0
14.0
16.0
18.0
z(km)
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

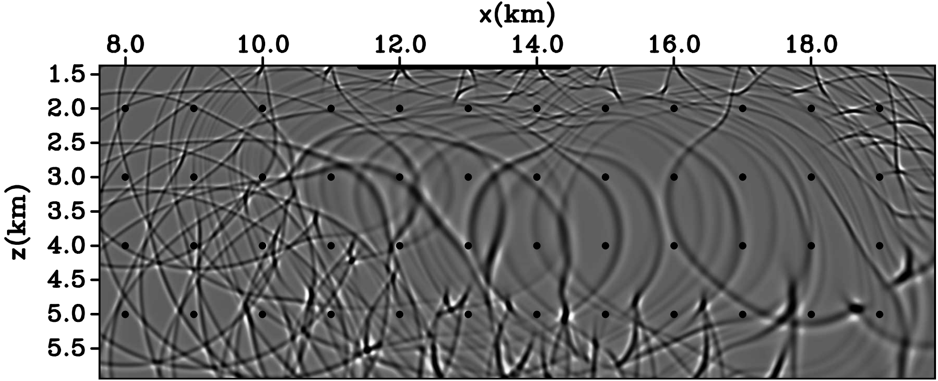


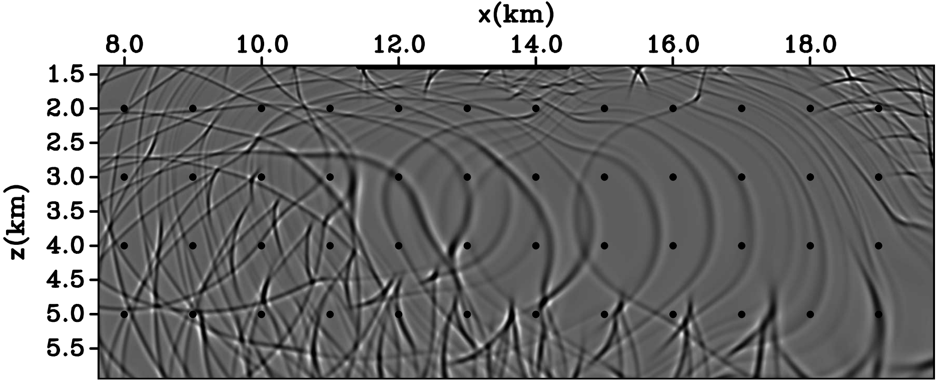
WSI


x(km)
8.0
10.0
12.0
14.0
16.0
18.0
z(km)
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5



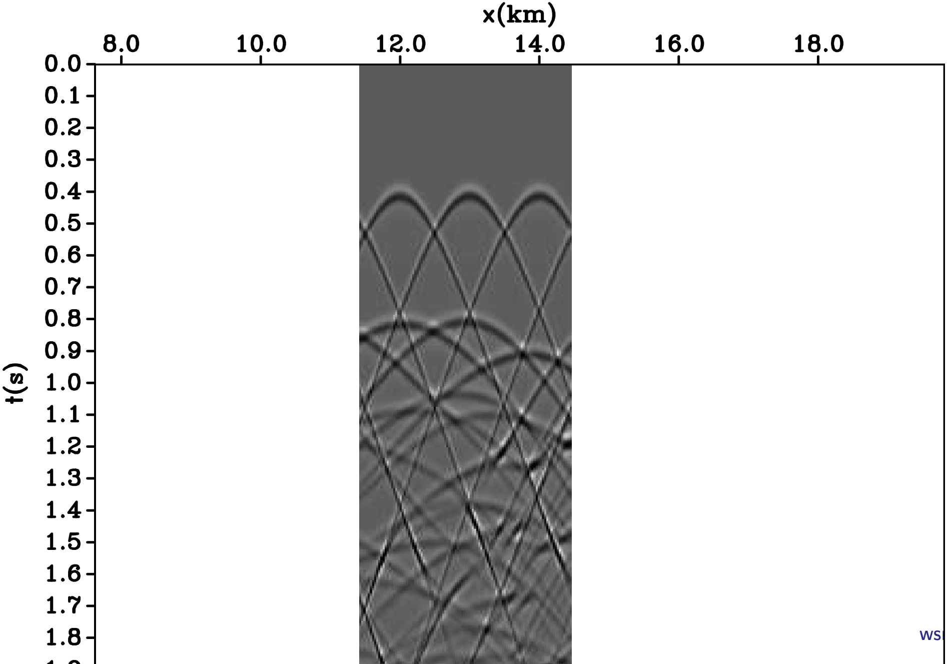
WSI


x(km)
8.0
10.0
12.0
14.0
16.0
18.0
z(km)
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5


x(km)
8.0
10.0
12.0
14.0
16.0
18.0
z(km)
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

plot data

```
# undo wavelet delay
Flow('data','temp',
     '''
     window squeeze=n f2=%(kt)d |
     pad end2=%(kt)d |
     put o2=%(ot)g
     ''' %par)

# plot data
Result('data','window j2=4 | transp |'
       + fdmod.dgrey('',par))
```


x(km)
8.0
10.0
12.0
14.0
16.0
18.0
t(s)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8

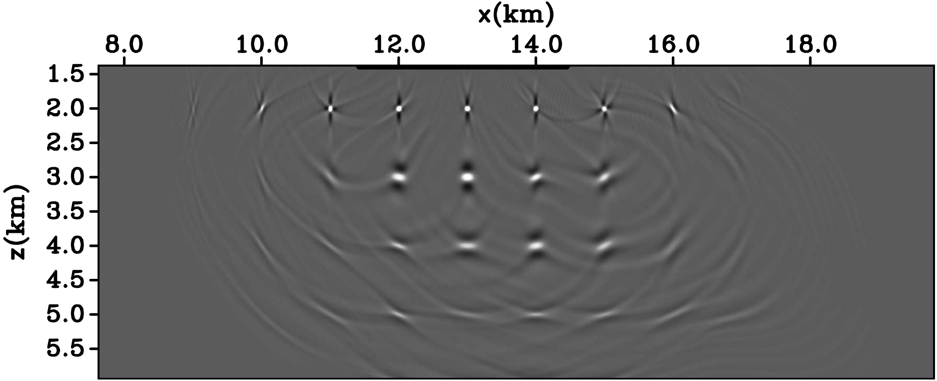
ws

FD migration

```
# run FD migration
fdmod.zom('imag','jdat',
          'data','velo','dens',
          'rr','rr',
          'free=n',par)
```

plot image

```
# plot image
Plot(    'imag','bandpass flo=2 |'
        + fdmod.cgrey('pclip=99.99',par))
Result('imag',['imag','rr'],'Overlay')
```


x(km)
8.0
10.0
12.0
14.0
16.0
18.0
z(km)
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

closing rules

End ( )

the contest

Assuming that each receiver costs a fixed amount of \$, construct the receiver array that gives you the best image for the lowest cost.

- you can consider multiple arrays
- you can place the receivers anywhere on the surface

http://reproducibility.org

MADAGASCAR