# Wavefield modeling and migration in Madagascar

Junzhe Sun

Jan 8, 2015

# Outline

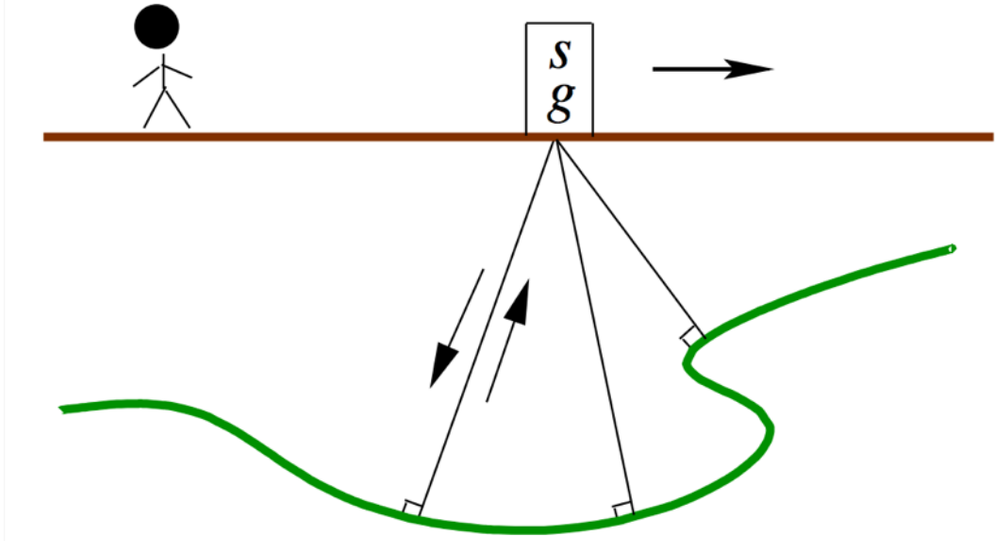
1. A brief review of exploding reflector modeling/ migration

2. Implementation and application of the technique in Madagascar

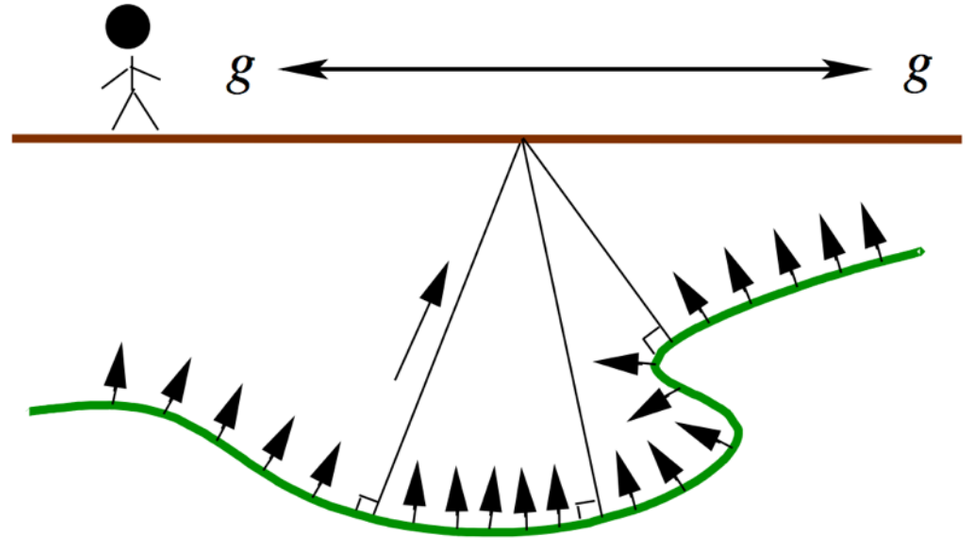
3. Interactive exercises

# exploding reflector concept

sources are deployed simultaneously along the reflectors in a medium of half the original velocity




*Zero−offset Section*




*Exploding Reflectors*

Claerbout, 1985

(figures from Paul Sava)

# wavefield extrapolation in time

$$\frac{1}{v^2}\frac{\partial^2 p}{\partial t^2} = \rho\nabla\cdot(\frac{1}{\rho}\nabla p) - f$$

- Modeling — forward propagation
- Migration — backward propagation

# numerical example

bash $ svn checkout http://svn.code.sf.net/p/rsf/code/trunk/book/rsf/school2015

bash$ cd $DIRECTORY/school2015/modmig/

bash$ vi SConstruct

replace $DIRECTORY with your download directory

# import RSF packages & ending rule

```
from rsf.proj import *
...
...
...
End()
```

Two basic components of a Madagascar SCons script

# set up parameters

```
nt = 900; dt = 0.002      # time sampling
nz = 201; dz = 10; oz = 0 # spatial sampling in z
nx = 401; dx = 10; ox = 0 # spatial sampling in x
nb = 30; decay = 0.01     # ABC parameters
snap = 5                  # snapshot interval
```

# create a model in ASCII

```
Flow('bvel2.asc',None,
     '''
     echo 0 1500 1000 1500 2000 1500 3000 1500
     4000 1500 n1=2 n2=5 in=$TARGET
     data_format=ascii_float ''')
```

**echo** is a linux command

# format conversion

```
Flow('lay2','bvel2.asc',
     '''
     dd form=native |
     spline n1=401 d1=10 o1=0
     ''')
```

**sfdd** converts the ASCII format into RSF format

# concatenate datasets

```
Flow('lays','lay1 lay2','cat axis=2 ${SOURCES[1:2]}')
```

try comparing the dimensions of lays.rsf and lay1.rsf using **sfin**

# generate layered velocity model from specified interfaces

```
Flow('vel1','lays',
     '''
     unif2 n1=%d d1=%g o1=%g v00=2000,2500,3000
     | put label1=Depth unit1=m label2=Distance
     unit2=m label=Velocity unit=m/s ''' % (nz,dz,oz) )
```

**sfunitf2** converts specified interfaces into layered model

# generate layered velocity model from specified interfaces

```
Flow('vel1','lays',
     '''
     unif2 n1=%d d1=%g o1=%g v00=2000,2500,3000
     | put label1=Depth unit1=m label2=Distance
     unit2=m label=Velocity unit=m/s ''' % (nz,dz,oz) )
```

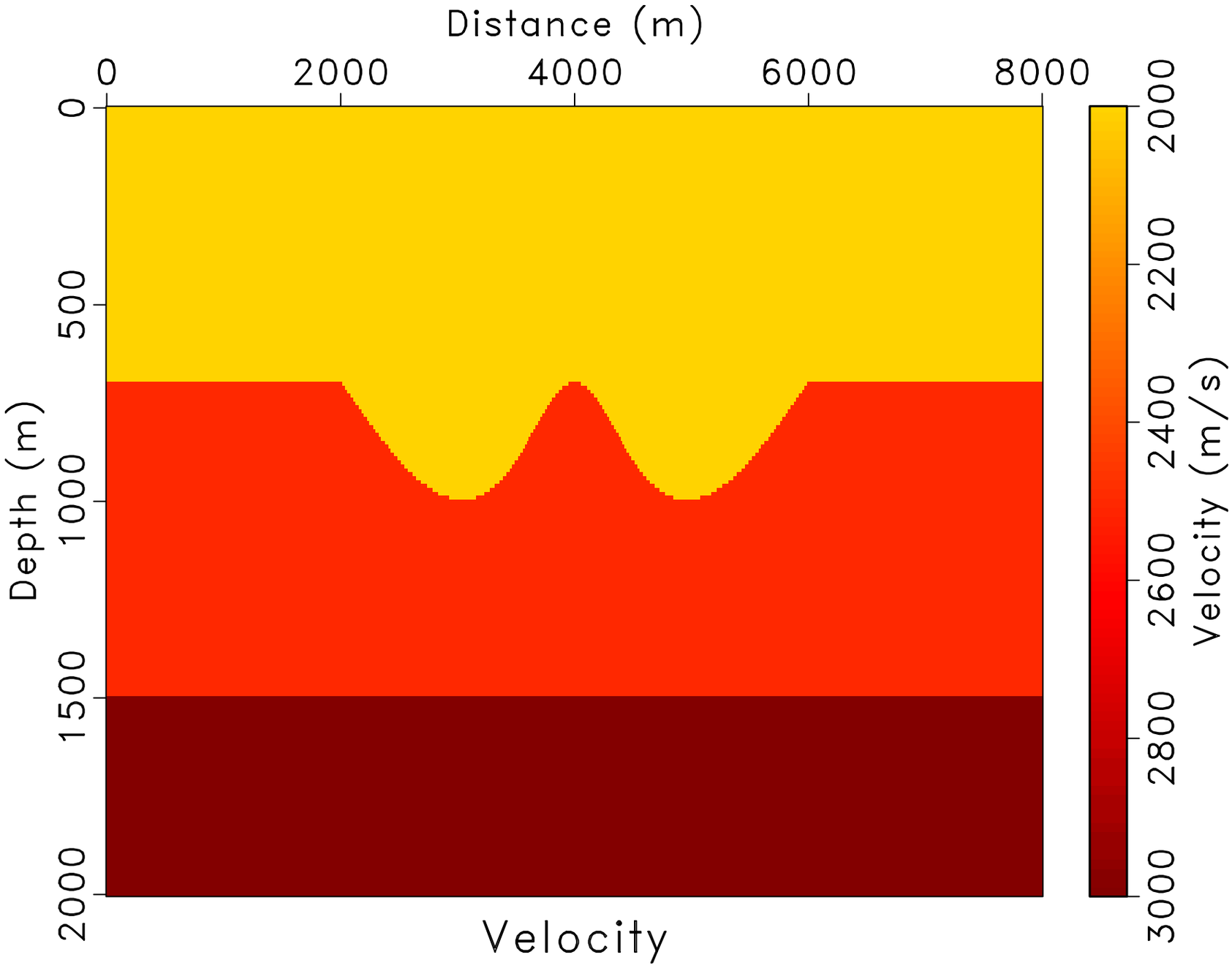
the **%** operator substitutes the values in parenthesis according to their sequence

# extend the velocity model

```
Flow('leftv','vel1',
    '''
    window n2=1 |spray axis=2 n=200 d=10 o=-2000''')
Flow('rightv','vel1',
    '''
    window n2=1 f2=400 |spray axis=2 n=200 d=10 o=4010''')
Flow('vel','leftv vel1 rightv','cat axis=2 ${SOURCES[1:3]} | put o2=0')
```

pad the velocity model on both sides

# layered velocity



# calculated reflectivity from velocity model

```
Flow('ref','vel',
     '''
     depth2time velocity=$SOURCE nt=1000 dt=0.002 |
     ai2refl |ricker1 frequency=10 |
     time2depth velocity=$SOURCE ''')
```

**sfdepth2time** converts depth to time along the first axis
**sftime2depth** converts time to depth along the first axis

# calculated reflectivity from velocity model

```
Flow('ref','vel',
     '''
     depth2time velocity=$SOURCE nt=1000 dt=0.002 |
     ai2refl |ricker1 frequency=10 |
     time2depth velocity=$SOURCE ''')
```

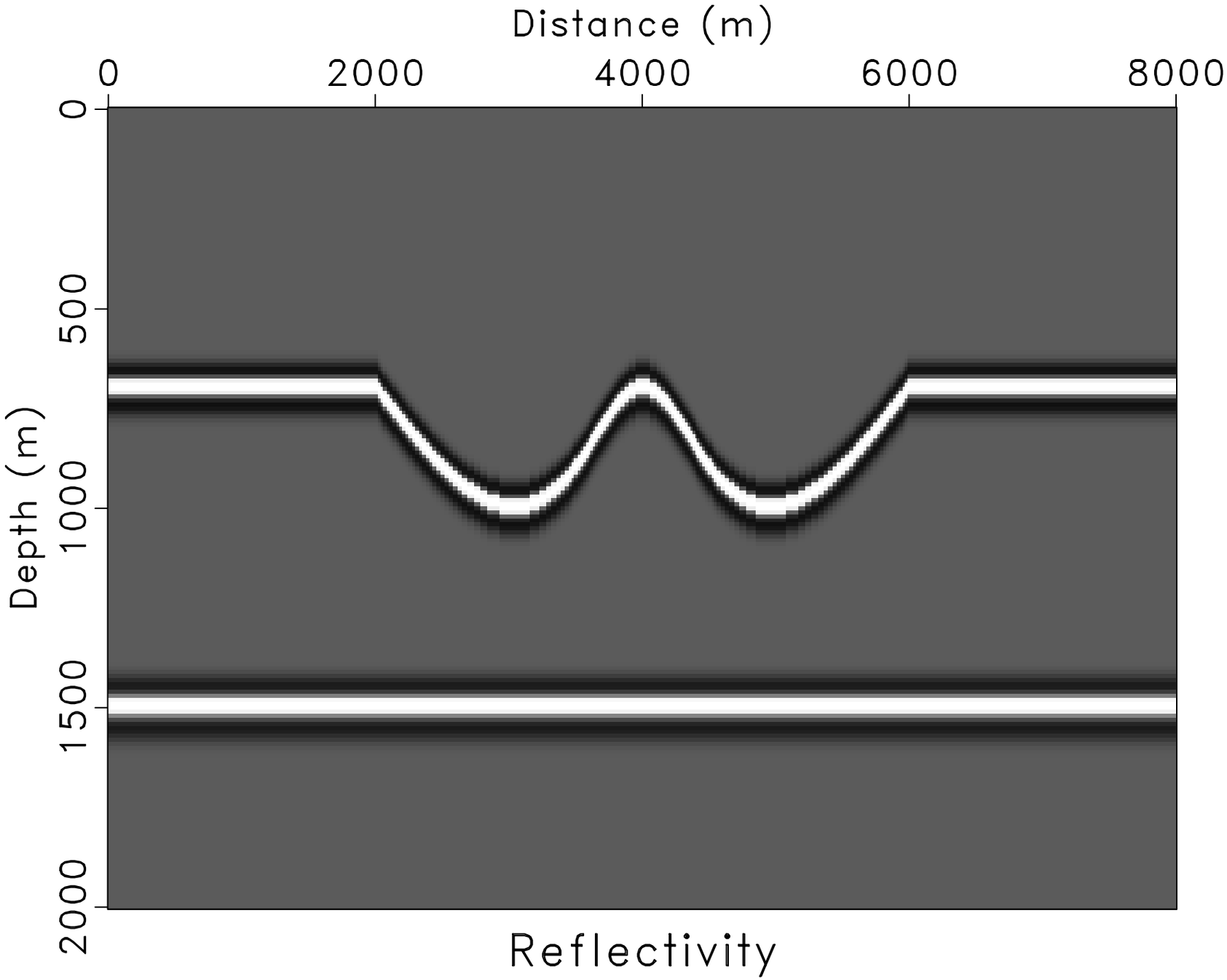
**sfai2refl** converts acoustic impedance to reflectivity

# calculated reflectivity from velocity model

```
Flow('ref','vel',
     '''
     depth2time velocity=$SOURCE nt=1000 dt=0.002 |
     ai2refl |ricker1 frequency=10 |
     time2depth velocity=$SOURCE ''')
```

**sfricker1** convolves a Ricker wavelet with a peak frequency of 10 Hz

# reflectivity

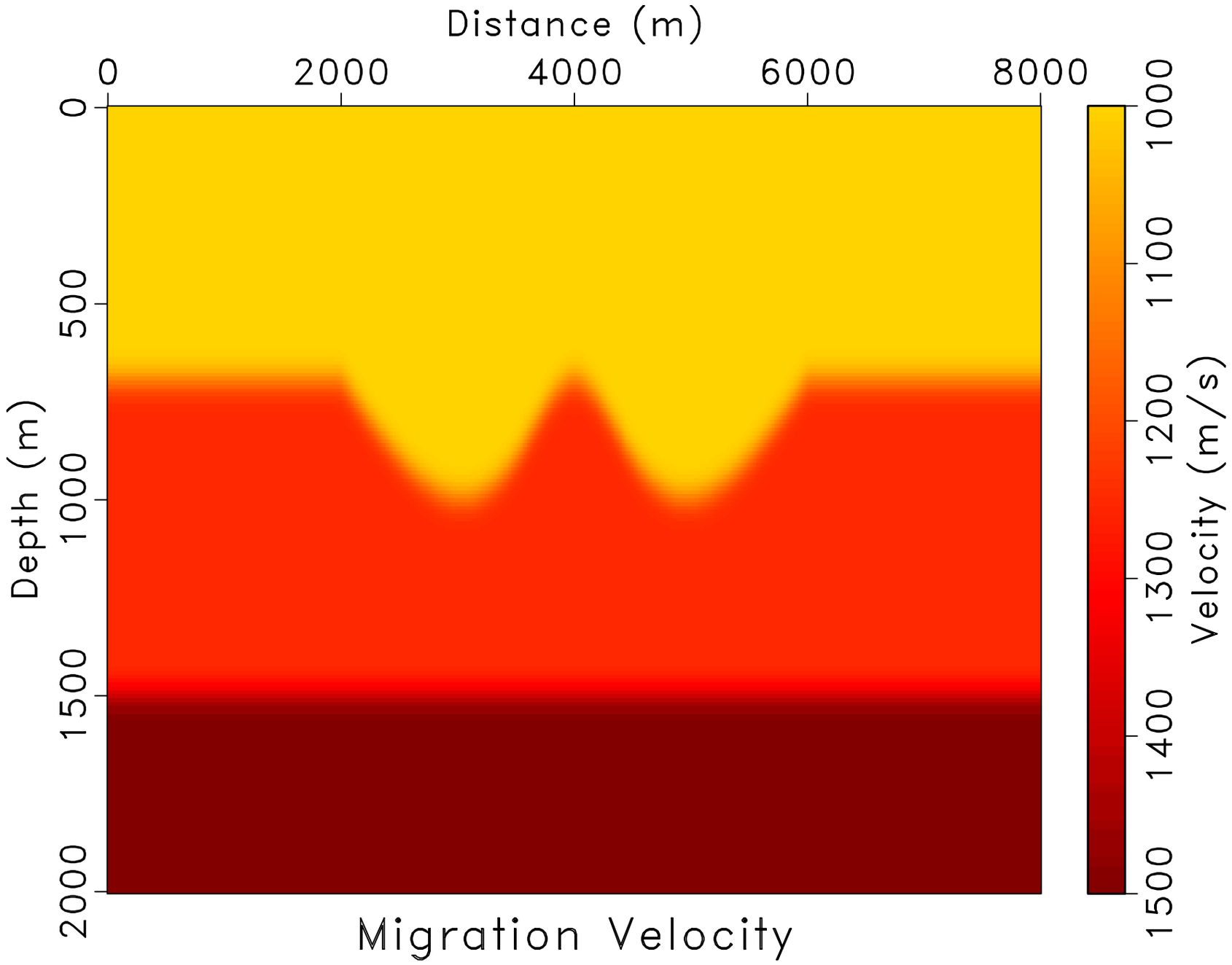


# create migration velocity using stratigraphic velocity

```
Flow('migvel','vel','smooth rect1=5 rect2=5 repeat=3 |
math output="input/2" ')
```

zero-offset migration uses half the (smoothed) velocity

# smoothed velocity (halved)



# generate executable program using c file

```
proj = Project()
prog = proj.Program('zofdrtm2.c')
```

program compilation on the fly

# 4th order in space, 2nd order in time FD exploding reflector modeling/migration

```
/* 2-D finite-difference zero-offset exploding reflector modeling/migration */

#include <rsf.h>

#ifdef _OPENMP
#include <omp.h>
#endif

/*****************************************************/
/* wave propagation struct */
typedef struct Par {
…

int fdexp(float **img, float **dat, bool adj, par pars, float **vv, float ***wvfld)
/*< zero-offset exploding reflector modeling/migration >*/
…
int main(int argc, char* argv[])
…
```

**zofdrtm2.c**

# exploding reflector modeling

```
Flow('dat wave','ref %s migvel' % prog[0],
     '''
     ./${SOURCES[1]} vel=${SOURCES[2]} verb=y nt=
     %d dt=%g mig=n nb=%d c=%g snap=%d
     snaps=${TARGETS[1]} ''' %(nt,dt,nb,decay,snap) )
```

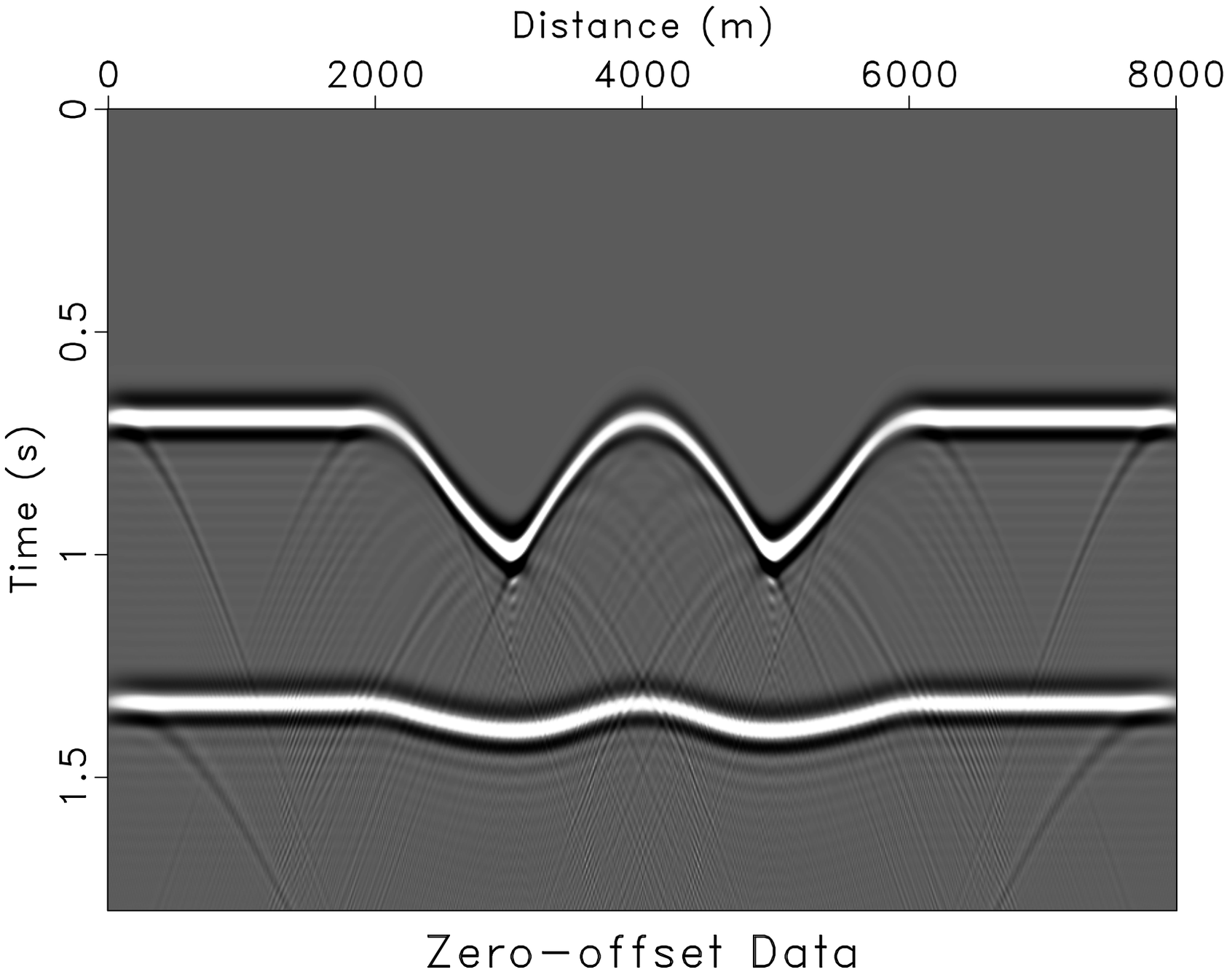
vel -> velocity file, verb -> verbosity flag, nt -> number of time samples, dt -> time sampling interval, mig -> migration flag, nb -> width of ABC, c -> decay parameter, snap -> snapshot interval, snaps -> snapshot file

# plot the zero-offset data and wave propagation movie

```
Result('dat','grey title="Zero-offset Data" ')
Plot('wavem','wave','window j3=2 | grey gainpanel=all
title=Forward')
```

wavem.vpl stores forward wave propagation a movie

# zero-offset data



# exploding reflector migration

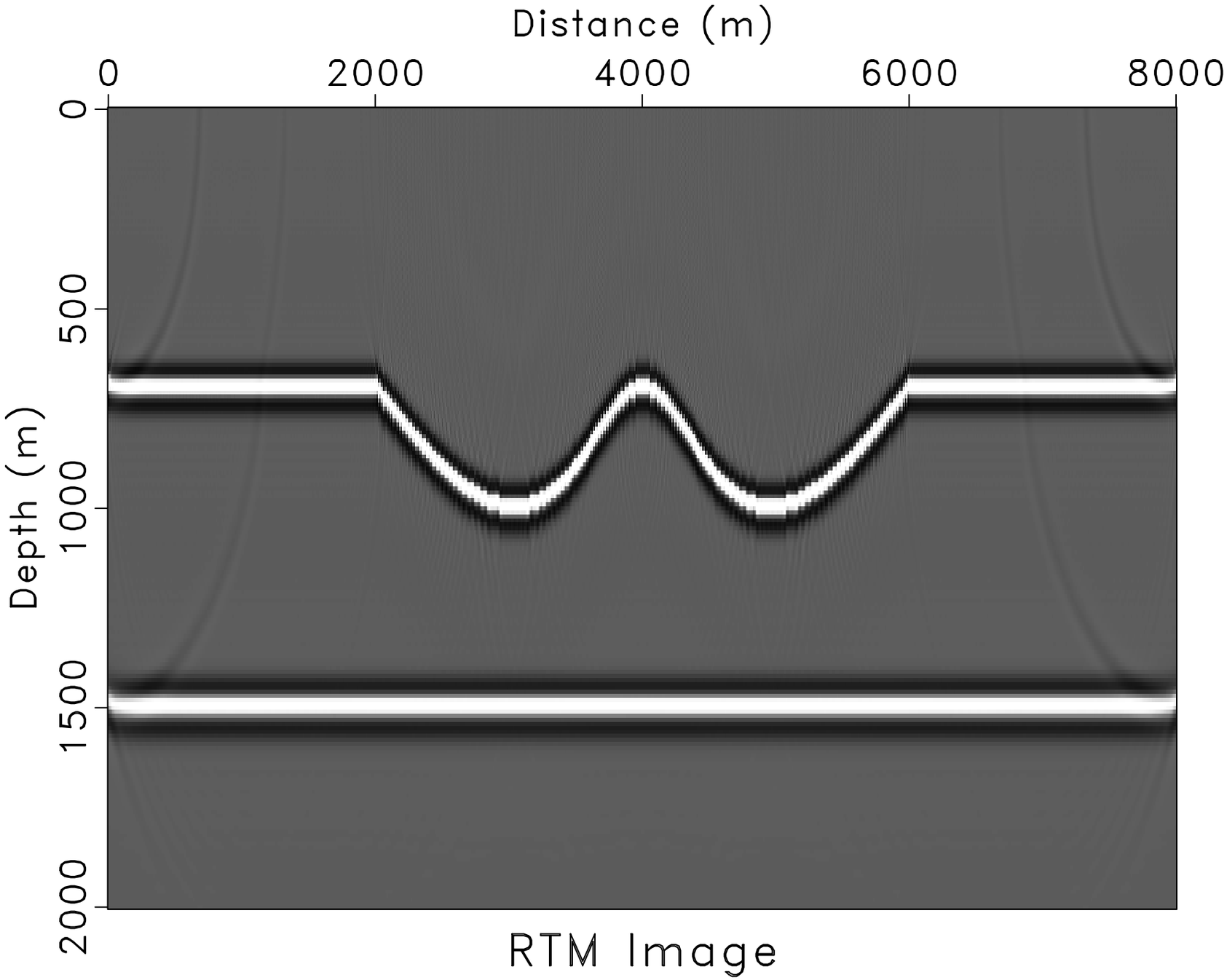
```
Flow('img wave2','dat %s migvel' % prog[0],
     '''
./${SOURCES[1]} vel=${SOURCES[2]} verb=y
mig=y nb=%d c=%g snap=%d snaps=
${TARGETS[1]} ''' %(nb,decay,snap) )
```

migration flag is turned on

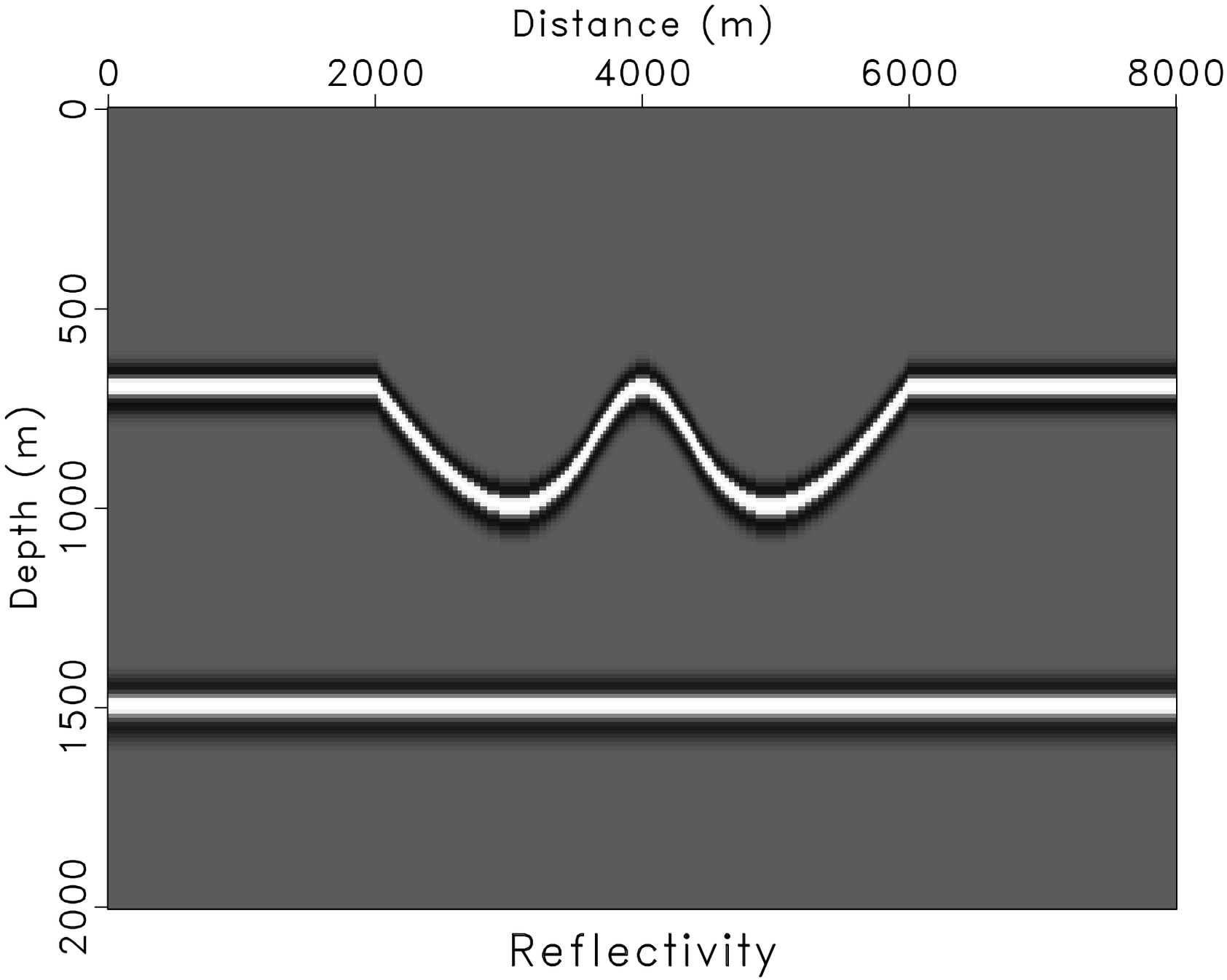
# plot the migrated image and wave propagation movie

```
Result('img','grey title="RTM Image" ')
Plot('wave2m','wave2','window j3=2 | grey
gainpanel=all title=Backward')
```

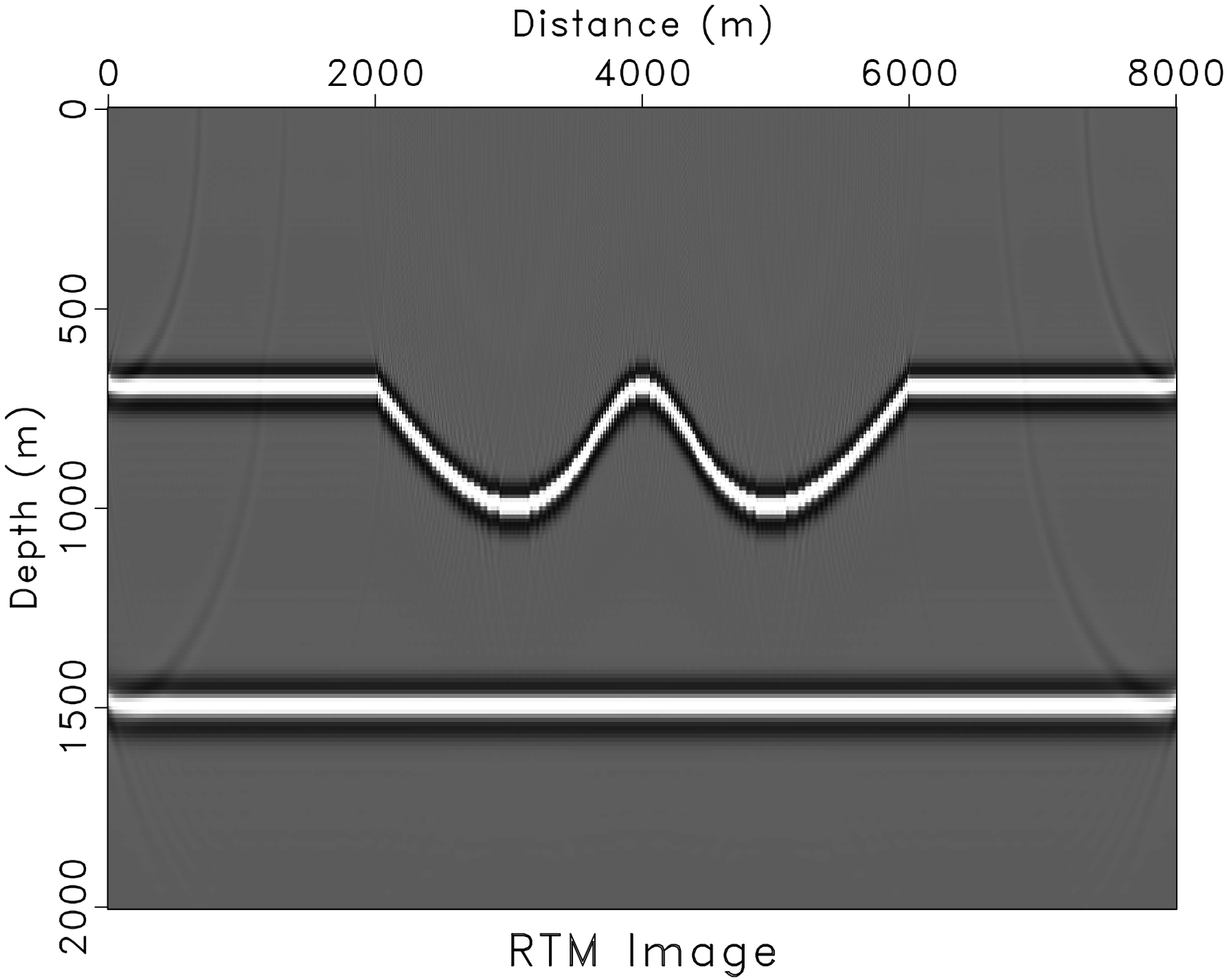
# zero-offset migration



# reflectivity



# zero-offset migration



# assignment

- create your own model;
- model the zero-offset data & perform exploding reflector migration.

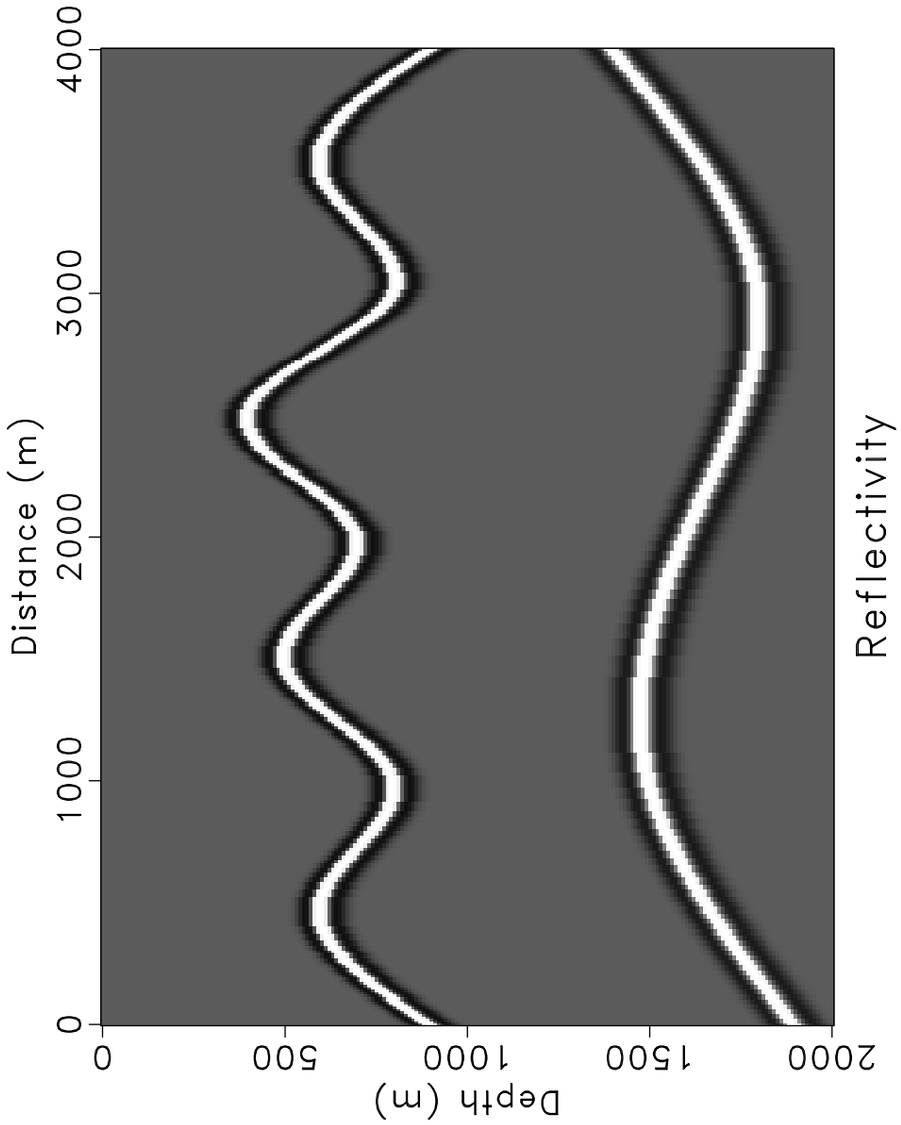
- Do you see more/less dispersion and imaging artifacts after changing the velocity?

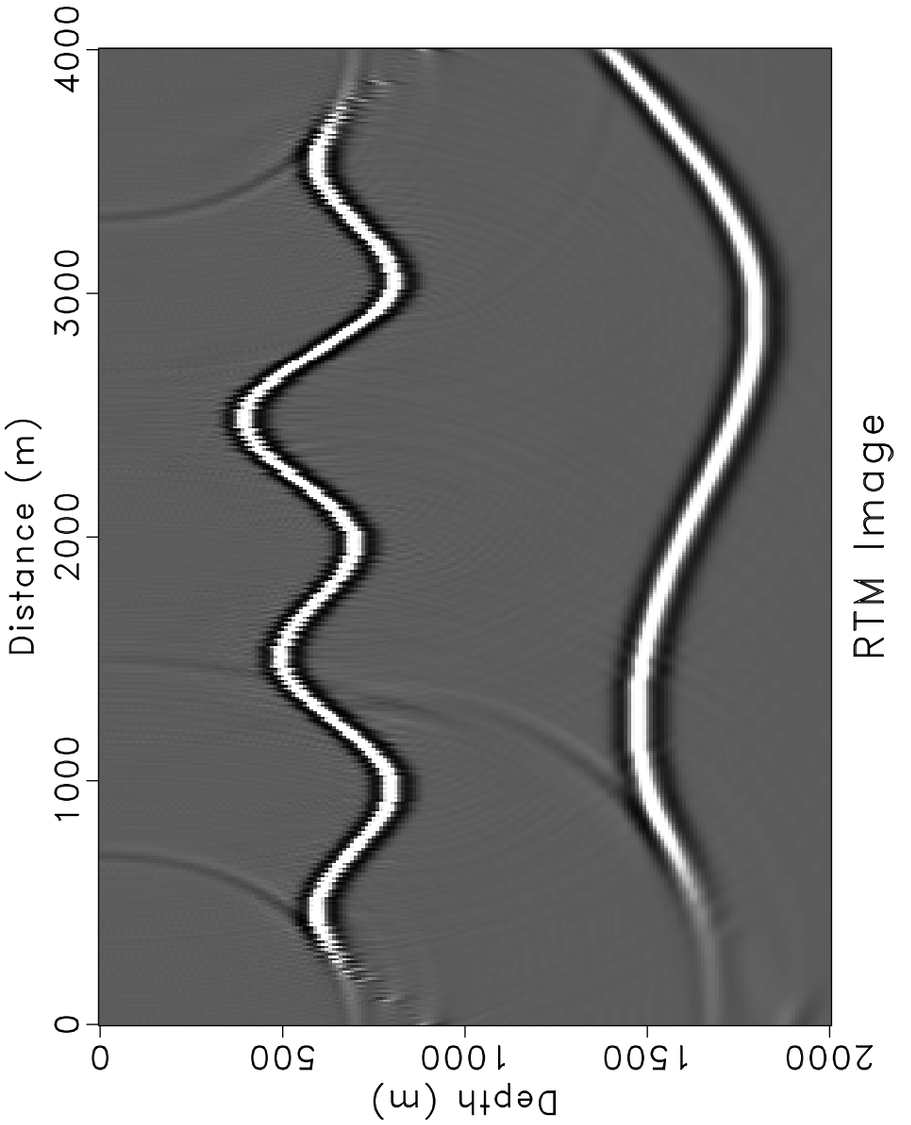
# where to modify

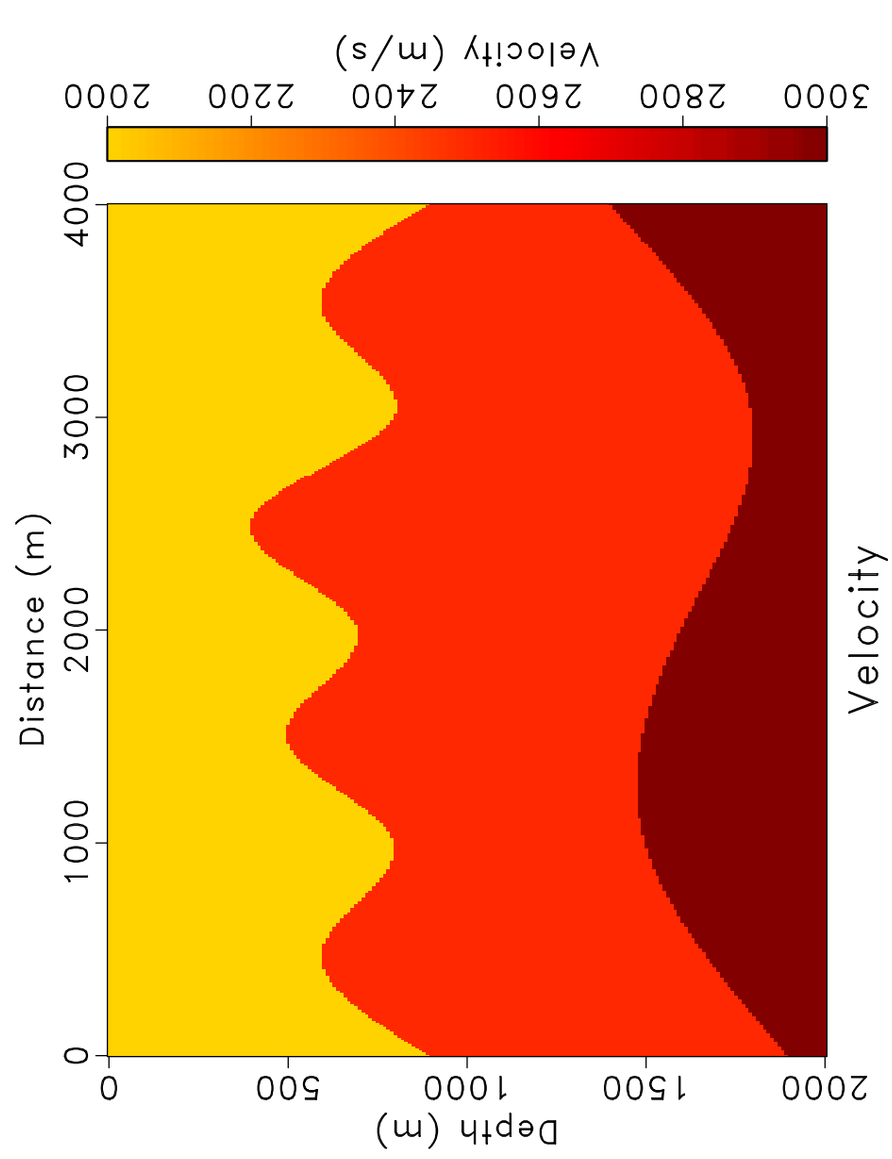
```
Flow('bvel2.asc',None,
     '''
     echo 0 1500 1000 1500 2000 1500 3000 1500
     4000 1500 n1=2 n2=5 in=$TARGET
     data_format=ascii_float ''')
```

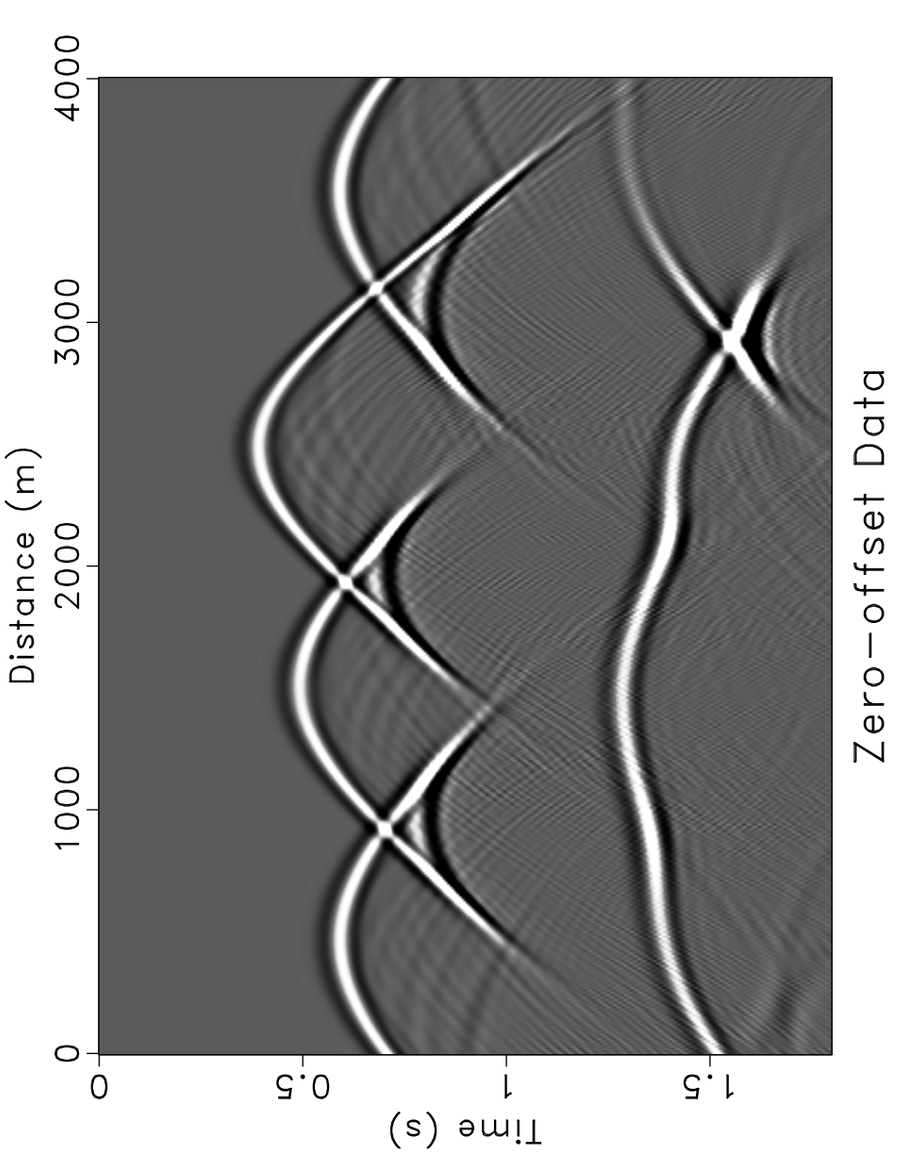
```
Flow('vel1','lays',
     '''
     unif2 n1=%d d1=%g o1=%g v00=2000,2500,3000
     | put label1=Depth unit1=m label2=Distance
     unit2=m label=Velocity unit=m/s ''' % (nz,dz,oz) )
```

for example…


Distance (m)
0
1000
2000
3000
4000
Depth (m)
0
500
1000
1500
2000
Reflectivity


Distance (m)
0
1000
2000
3000
4000
Depth (m)
0
500
1000
1500
2000
RTM Image


Velocity (m/s)
2000
2200
2400
2600
2800
3000
Distance (m)
0
1000
2000
3000
4000
Depth (m)
0
500
1000
1500
2000
Velocity


Distance (m)
0
1000
2000
3000
4000
Time (s)
0
0.5
1
1.5
Zero-offset Data

# “extra credits”

How to modify the zofdrtm2.c file to use higher-order FD schemes?

# THANKS!