2D modeling and basic processing with Madagascar

Pongthep Thongsang, Lian Jiang, Hao Li, and Karl Schleicher[1]

ABSTRACT

This document demonstrates finite difference modeling and simple processing using the Madagascar software package. The paper uses this simple processing sequence to teach the basics of Madagascar.

The SEGY file of the Marmousi2 model is downloaded from the Internet and used to create a few synthetic shots. Geometry is computed and loaded in the trace headers. Simple processing including amplitude correction, CMP sorting, velocity conversions, NMO correction, stack, Kirchhoff migration, and least squares Kirchhoff migration are applied. Displays of each processing stage are created. The full processing sequence takes less that 6 minutes on a 2013 MacBook Pro computer.

This processing sequence was developed at the 2017 Seismic Working Workshop - Reproducible Tutorials in Houston. All processing can be reproduced using an SConstruct file.

INTRODUCTION

There are several Madagascar tutorials. This tutorial differs from the other tutorials by focusing on the using existing Madagascar programs to create a reflection seismic synthetic and processing it using basic, standard techniques. The computation runs quickly which makes it easy to modify the SConstruct files and run new experiments. Processing can be reproduced using an SConstruct file listed in the appendix or by using the file:

$RSFSRC/book/data/mourmousi2mp/modelproc/SConstruct

in the Madagascar distribution.

DOWNLOAD, REFORMAT, SMOOTH, AND DISPLAY THE MARMOUSI2 VELOCITY MODEL

We use the p-wave velocity field from the Marmousi2 model described by Irons (2005). Figure 1 shows the part of the model used in this paper. Figures 2 and 3 show the model smoothed for modeling and processing (NMO and migration).

These figures can be created with the command:

```
scons vp.view vpforward.view vpmigration.view
```

[1]**e-mail:** k_schleicher@hotmail.com

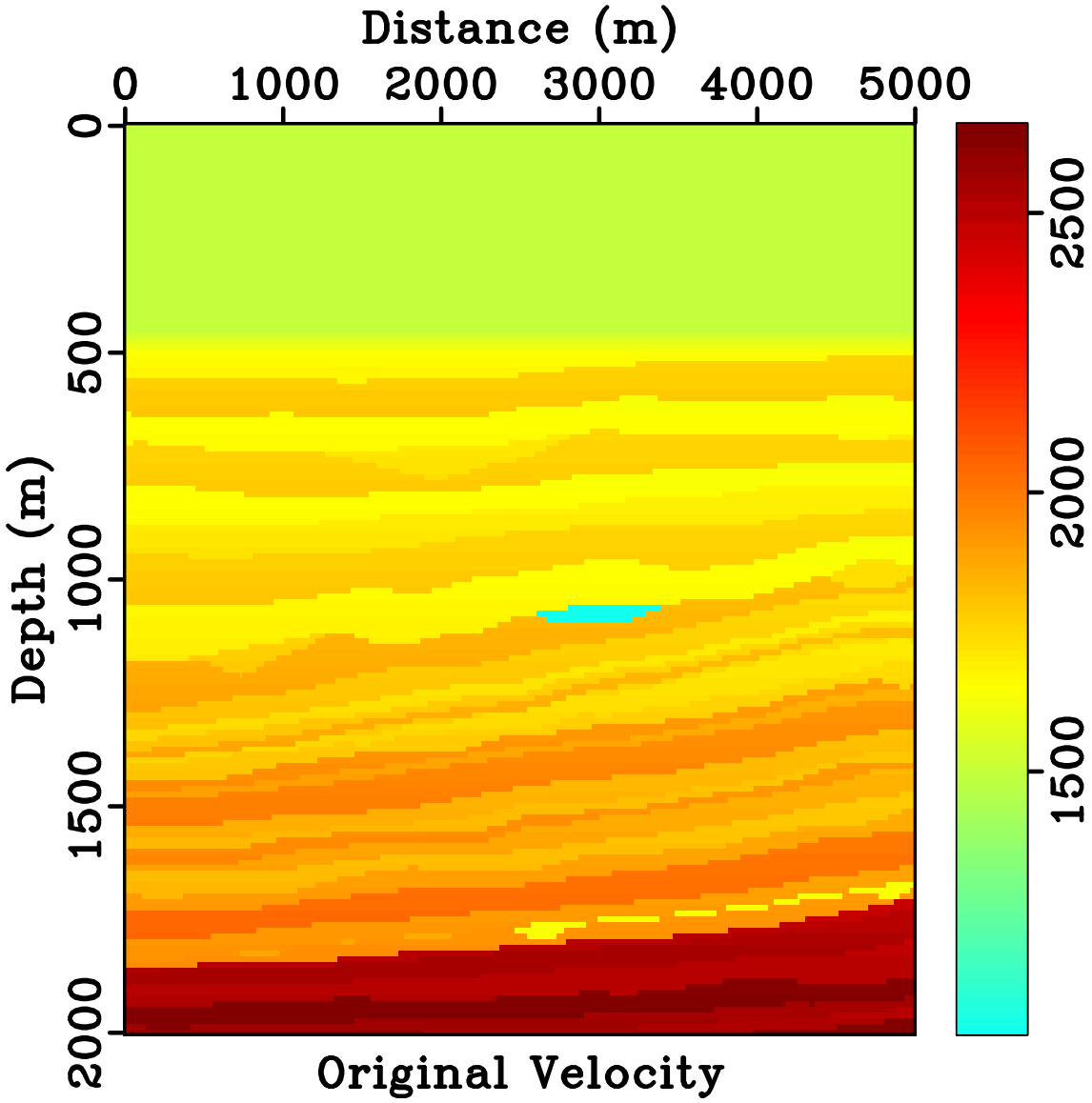

Figure 1: A portion of Marmousi2 velocity used in this paper. modelproc/ vp

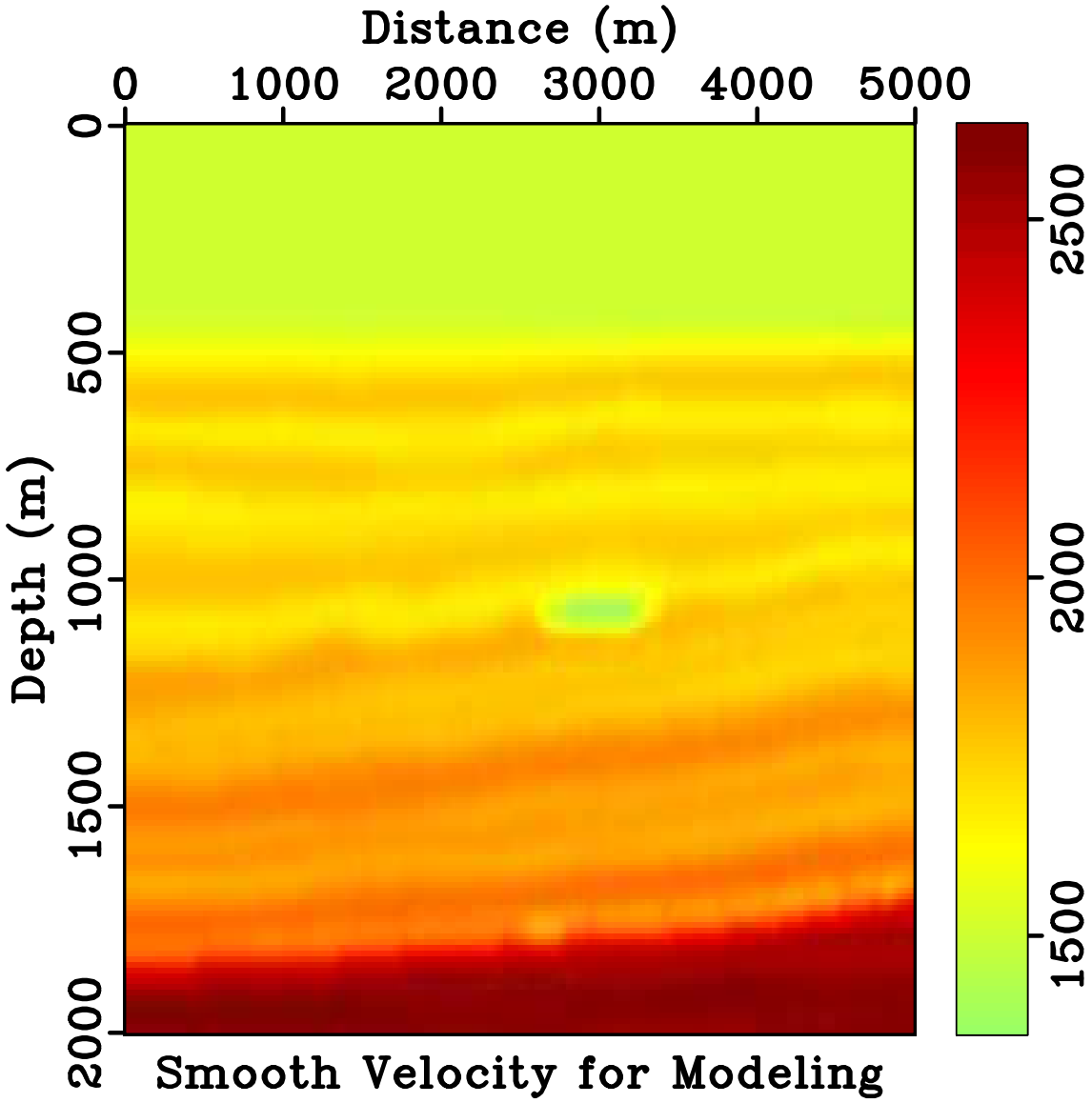

Figure 2: The velocity smoothed for modeling. modelproc/ vpforward

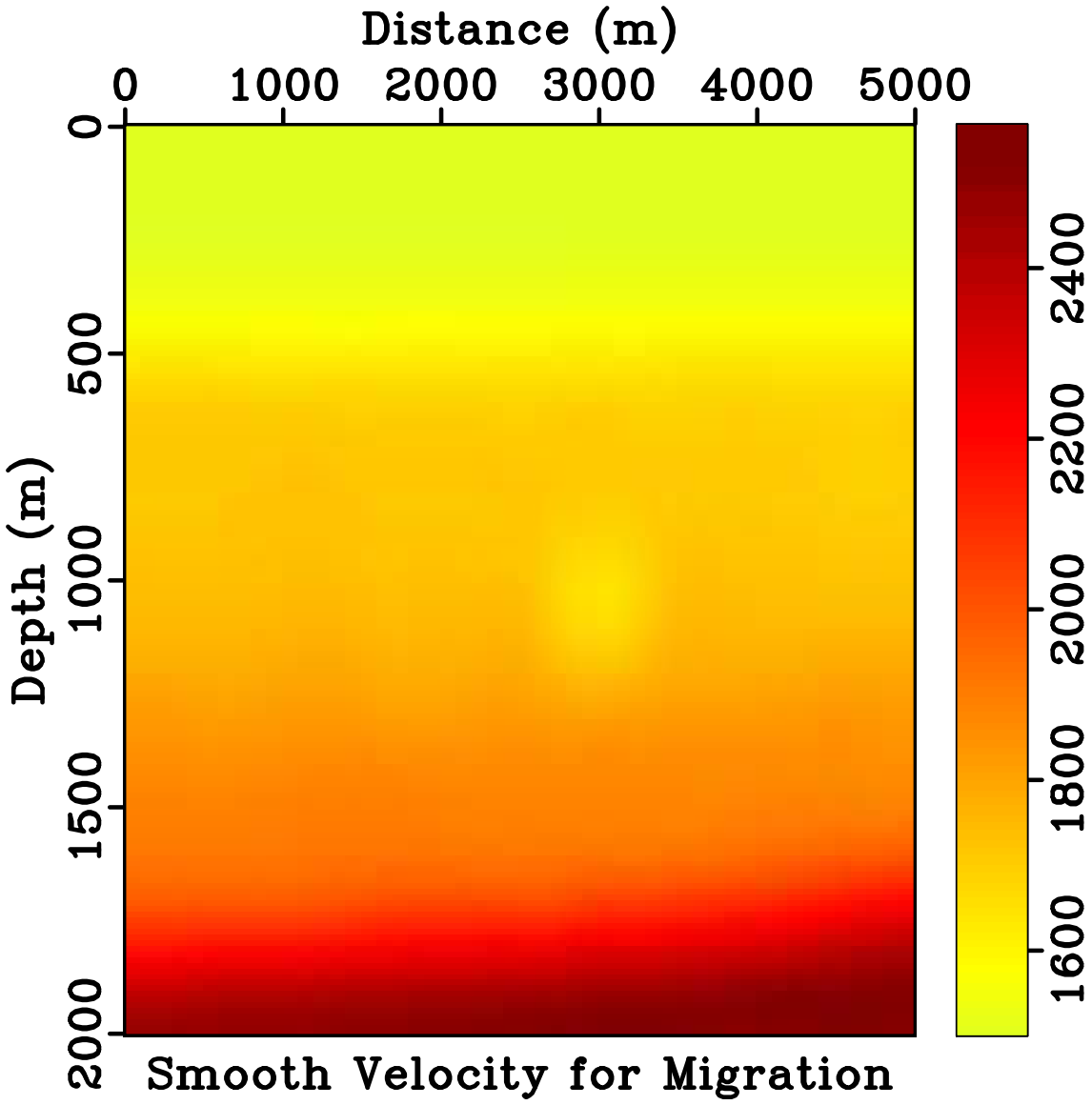

Figure 3: The interval velocity smoothed for NMO and migration. modelproc/ vpmigration

FINITE DIFFERENCE MODEL SHOTPOINTS, MODEL AND SUBTRACT THE DIRECT ARRIVALS

The wave equation program `sfmodeling2d` (Yang, 2014) is used to create 10 synthetic shot records using the slightly smoothed Marmousi model. One synthetic shotpoint is shown in Figure 4. We also ran `sfmodeling2d` with a constant velocity to model the direct arrivals. The direct arrivals were subtracted from the synthetic and shown in Figure 5

These figures and some others can be created with the command:

```
scons shots.view shot2500.view mutearr.view mutearr2500.view
```

COMPUTE RMS VELOCITIES, NMO, T-SHIFT, AMPLITUDE CORRECTION, STACK

The basic trace processing stages of sorting to CMP gathers, time shifting, amplitude correction, normal move out, and stack are applied using the trace and header (tah) programs. These programs all start with sftah. The sftahheadermath program was used to load the geometry into the trace headers. The sftahnmo program requires an RMS velocity in m/s at each input CMP and each time sample. This is computed from the original interval velocity model by stretching to time, squaring the velocity, integrating, dividing by time, and taking the square root using the programs sfdepth2time, sfmul, sfcausint, and sfmath. This creates an RMS velocity every 12.49 meters. Intermediate traces are created using sfinterleave. The stack is computed by sorting to CMP order, removing the 250 ms delay in the modeling Ricker wavelet, NMO and stack, (Figure 6).

This processing can be run with the command:

```
scons stack.view
```

POST STACK KIRCHHOFF MIGRATION AND LEAST SQUARES KIRCHHOFF MIGRATION

The stack and RMS velocity were input to sfkirchnew to apply post-stack Kirchhoff migration (Figure 7). Least squares Kirchhoff migration was computed using the `sfkirchinvs` program (Figure 8).

These processing can be reproduced with the command:

```
scons ktmig.view invsparse.view
```

ACKNOWLEDGEMENTS

This processing sequence was developed at the 2017 Seismic Working Workshop - Reproducible Tutorials in Houston (Schleicher, 2017). This working workshop was supported

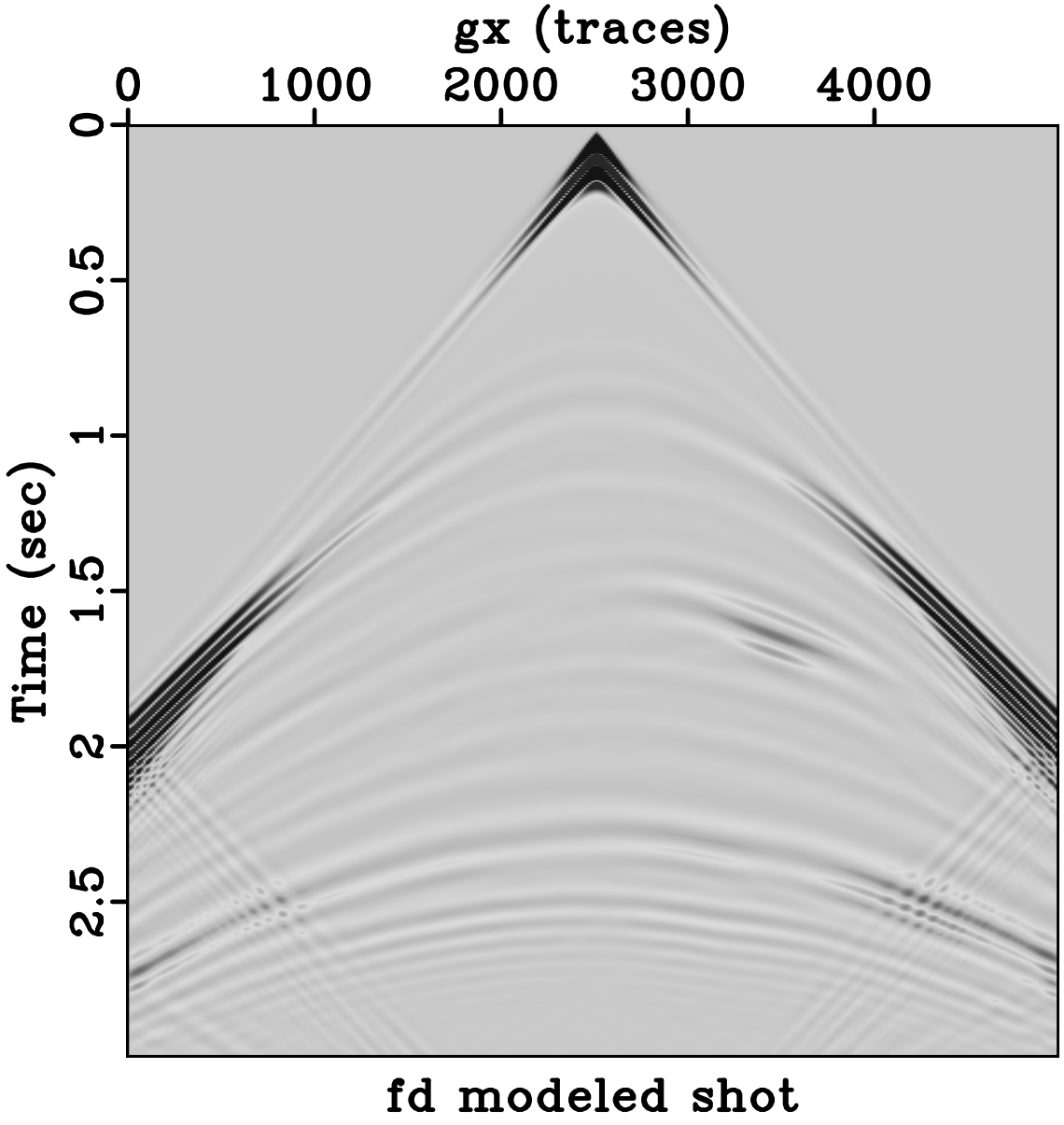

Figure 4: Shotpoint at x=2500 computed with acoustic finite difference. modelproc/ shot2500

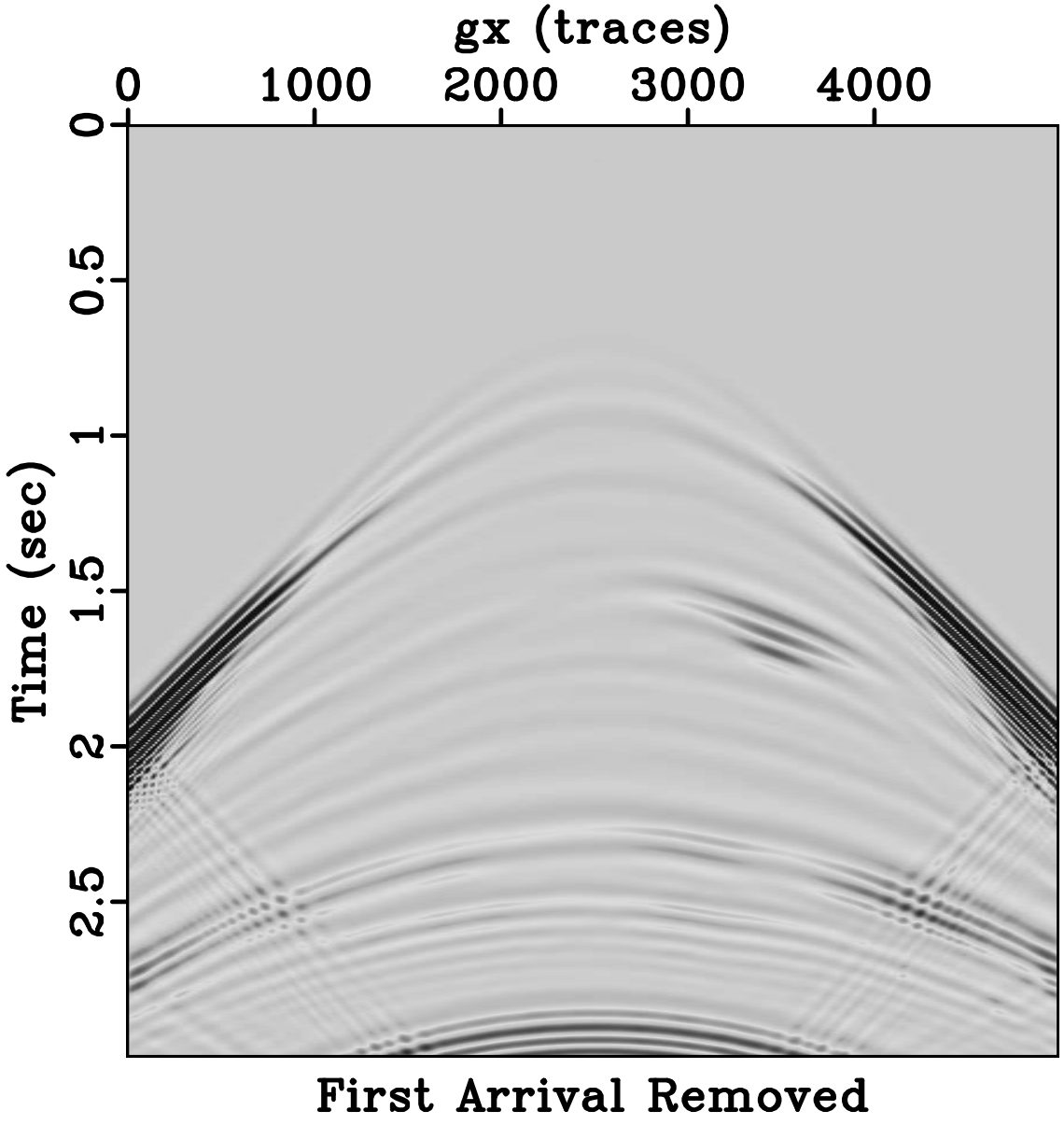

Figure 5: Shotpoint at x=2500 after subtracting the modelled direct arrivals. modelproc/ mutearr2500

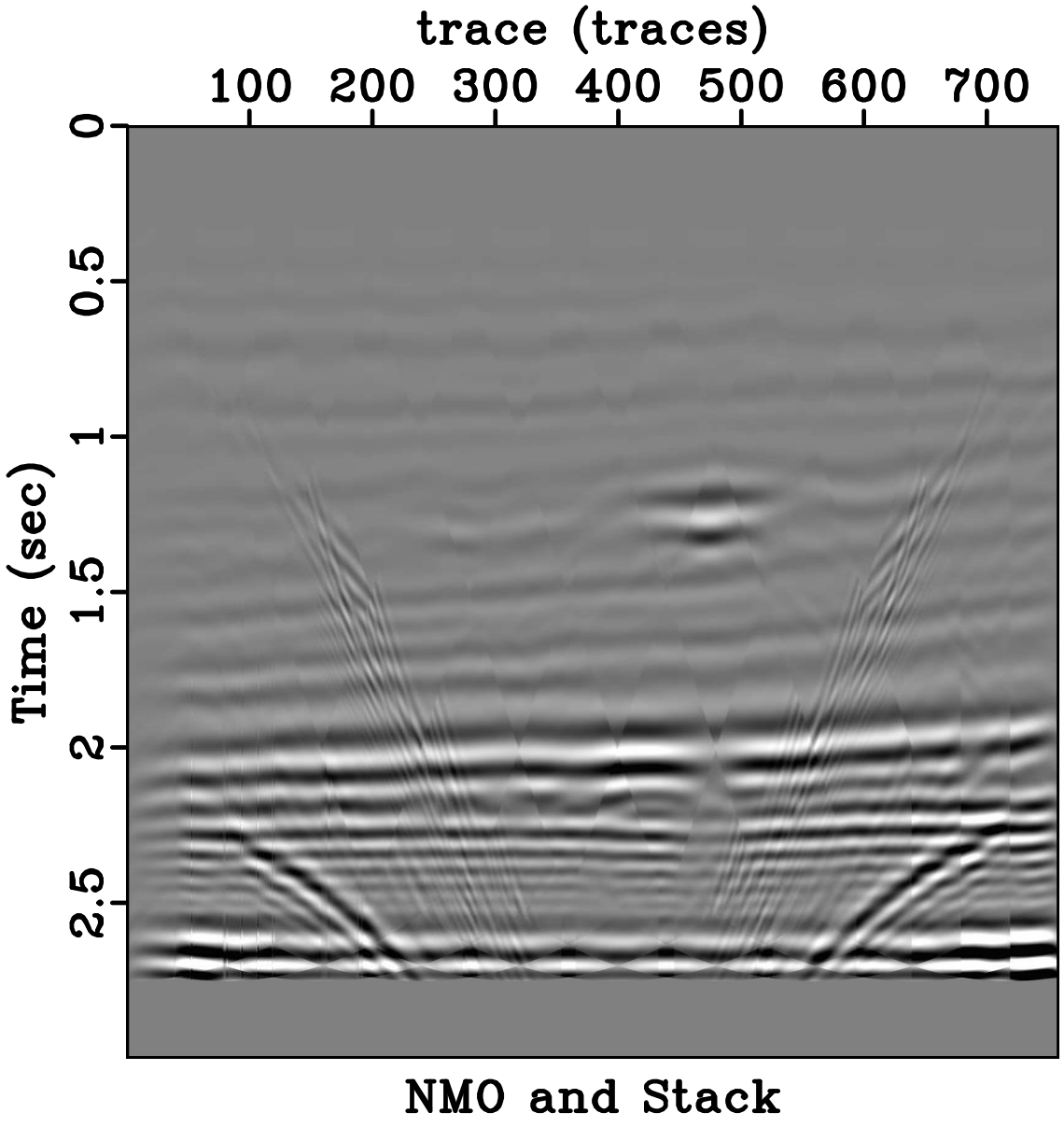

Figure 6: CDP stack of 10 shotpoints. modelproc/ stack

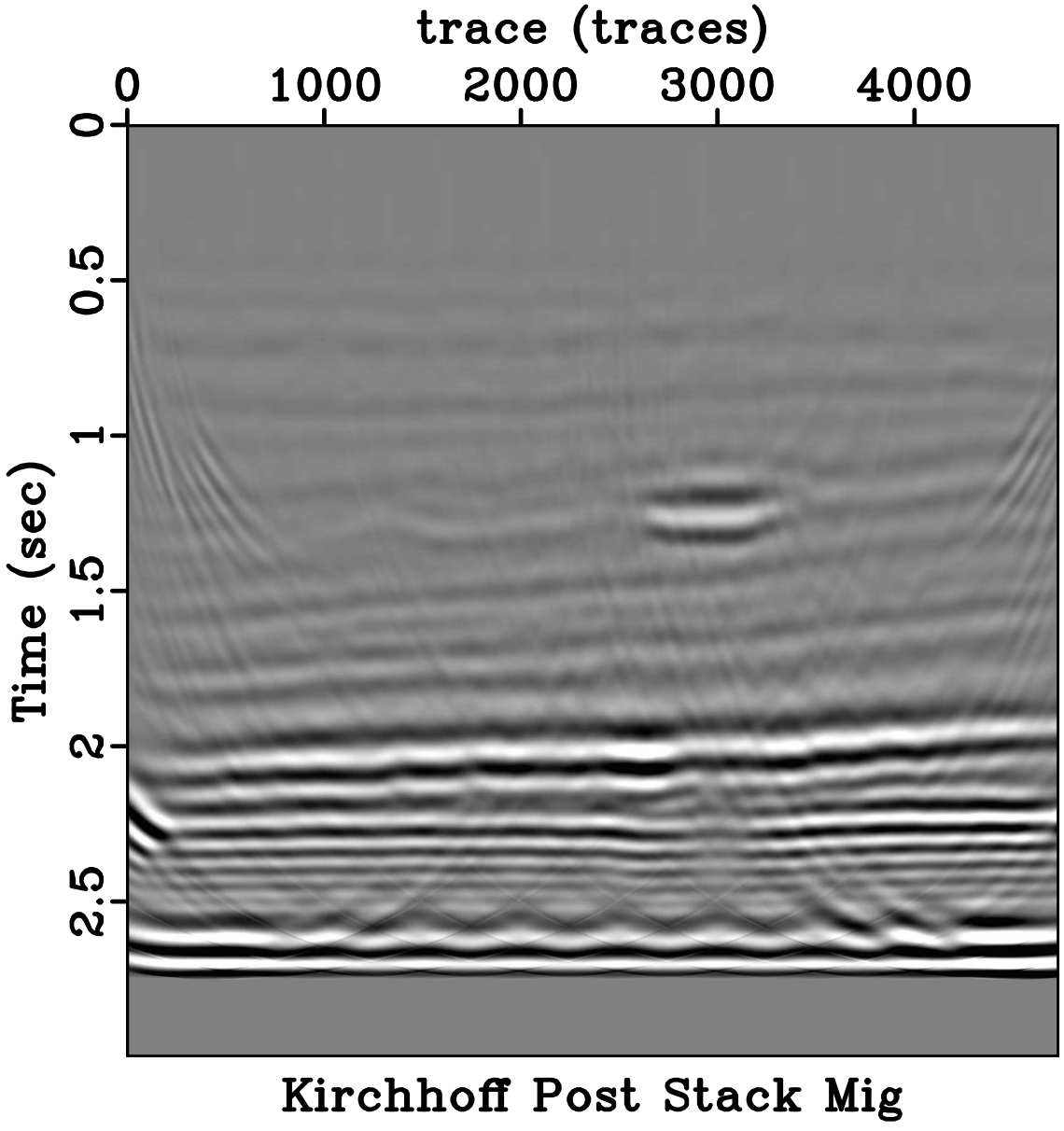

Figure 7: Post stack Kirchhoff migration. modelproc/ ktmig

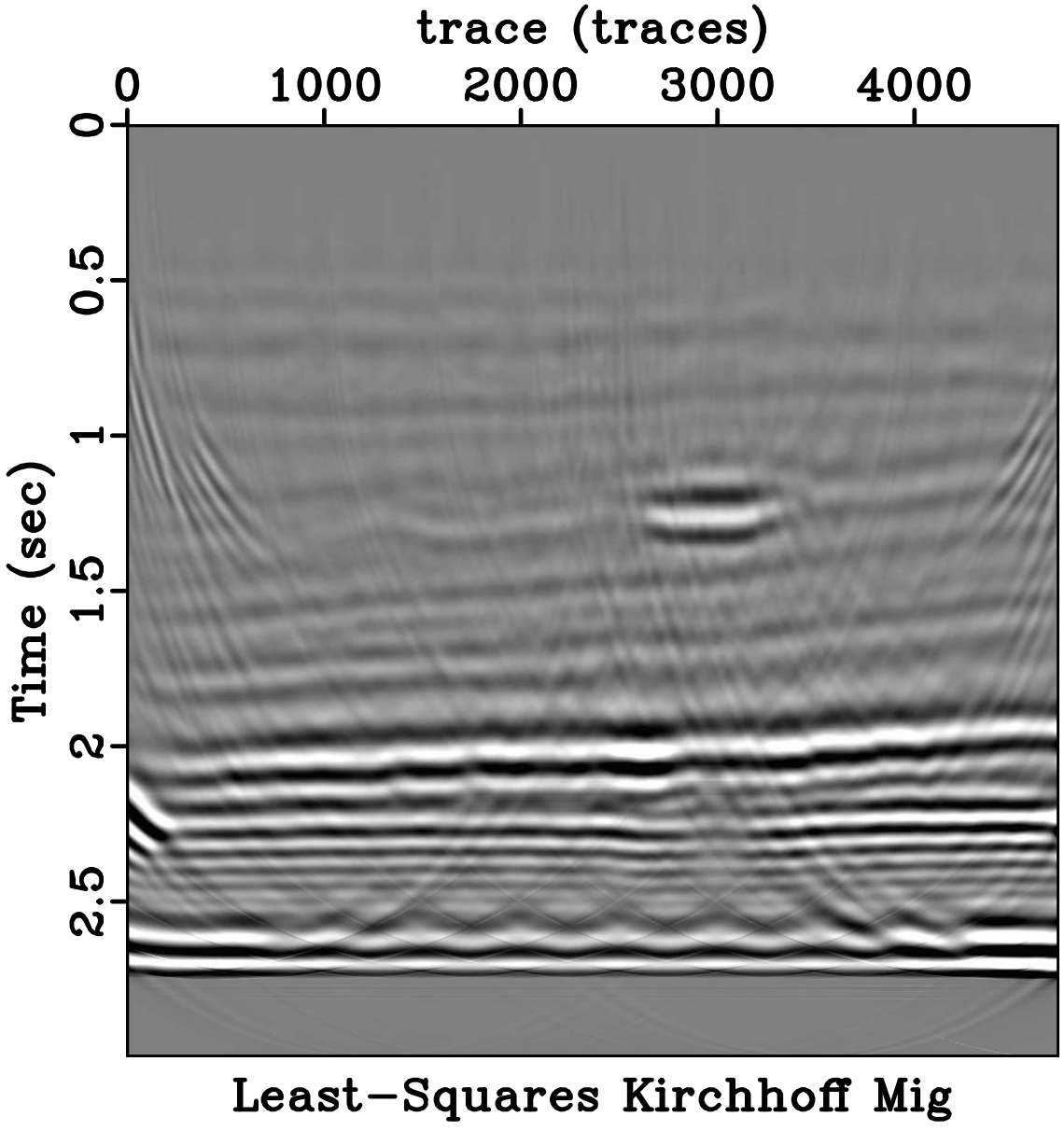

Figure 8: Least squares Kirchhoff migration. modelproc/ invsparse

by Wavelets (The University of Houston student SEG chapter), The Bureau of Economic Geology at the University of Texas, and SEG.

We thank Allied Geophysical Laboratories (AGL)[2] for creating and sharing the Marmousi2 model and synthetic data (Martin et al., 2002).

APPENDIX

The file `$RSFSRC/book/data/marmousi2mp/modelproc/SConstruct` is a script that applies the processing described in this paper. The contents of the file are listed below.

```
from rsf.proj import *

# Retrieve data
Fetch('vp_marmousi-ii.segy',"marm2")

# Processing units will be m and s.
# Input vel file is in km/s. mult by 1000 to m/s
# Use km in space domain and km/s for velocity

Flow('vp vp_hdr vp_hfile.txt vp_hfile.bin','vp_marmousi-ii.segy',
     '''
     sfsegyread tape=$SOURCE tfile=${TARGETS[1]} \
                hfile=${TARGETS[2]} bfile=${TARGETS[3]} \
     | sfput d1=1.249 o1=0 label1=Depth     unit1=m  \
          d2=1.249 o2=0 label2=Distance unit2=m  \
     | sfwindow min1=0 max1=2000 min2=0 max2=5000 j1=10 j2=10 \
     | sfmath output='input*1000'
     ''')

# plot the input velocity.  Use allpos=y to plot velocity.
Result('vp',
       '''
       grey color=j titlefat=4 titlesz=8 labelfat=4 labelsz=8
            screenratio=1.0 barwidth=0.6
            allpos=y scalebar=y n1tic=50 n2tic=40 verb=y
            title="Original Velocity"
       ''')

# smooth for vp finite difference modeling and plot. (allpos=y for velocity)
Flow('vpforward','vp','smooth rect1=3 rect2=3 repeat=3')
Result('vpforward',
       '''
       grey color=j titlefat=4 titlesz=8 labelfat=4 labelsz=8
            screenratio=1.0 barwidth=0.6
            allpos=y scalebar=y n1tic=50 n2tic=40 verb=y
            title="Smooth Velocity for Modeling"
       ''')


# smooth vp for nmo/migration and plot. (allpos=y for velocity)
Flow('vpmigration','vp','smooth rect1=10 rect2=10 repeat=5')
Result('vpmigration',
       '''
       grey color=j titlefat=4 titlesz=8 labelfat=4 labelsz=8
            screenratio=1.0 barwidth=0.6
            allpos=y scalebar=y n1tic=50 n2tic=40 verb=y
            title="Smooth Velocity for Migration"
       ''')

# model 10 shotpoints
# put for sftah... need d3, distance between shotpoints.
#kls need to add parameter time=time.rsf (default is time without .rsf)
Flow('shots sfmodelingtime','vpforward',
        '''
         sfmodeling2d csdgather=n fm=10 amp=1 dt=0.0015 ns=10 ng=400 nt=2000
            sxbeg=1 szbeg=1 jsx=40 jsz=0 gxbeg=0 gzbeg=0 jgx=1 jgz=0
            time=${TARGETS[1]}
         | sfput d3=500 label3=sx label2=gx unit2=traces o3=0
         ''')

# plot all 10 shots as movie
Result('shots',
       '''
       sfgrey color=j titlefat=4 titlesz=8 labelfat=4 labelsz=8
            screenratio=1.0
            allpos=n scalebar=n n1tic=50 n2tic=40
            title="fd modeled shots"
       ''')

```

[2] `http://www.agl.uh.edu/`

```
# plot shot at x=2500 for paper
Result('shot2500','shots',
       '''
       sfwindow n3=1 min3=2500 |
       sfgrey color=j titlefat=4 titlesz=8 labelfat=4 labelsz=8
              screenratio=1.0
              allpos=n scalebar=n n1tic=50 n2tic=40
              title="fd modeled shot"
       ''')

# make velocity field constant 1500 m/s and model direct arrivals
Flow('arr sfmodelingtime1','vpforward',
     '''
     sfmath output="1500" |
     sfmodeling2d csdgather=n fm=10 amp=1 dt=0.0015 ns=10 ng=400 nt=2000
         sxbeg=1 szbeg=1 jsx=40 jsz=0 gxbeg=0 gzbeg=0 jgx=1 jgz=0
            time=${TARGETS[1]}
     ''')


# subtract modeled direct arrivals (a type of noise) from shots
Flow('mutearr','shots arr',
     '''
     sfadd scale=-1,1 ${SOURCES[1]}
     ''')

# plot all 10 shots as movie
Result('mutearr',
       '''
       sfgrey color=j titlefat=4 titlesz=8 labelfat=4 labelsz=8 screenratio=1.0
              allpos=n scalebar=n n1tic=50 n2tic=40
              title="First Arrival Removed"
       ''')

# plot shot at x=2500 for paper
Result('mutearr2500','mutearr',
       '''
       sfwindow n3=1 min3=2500 |
       sfgrey color=j titlefat=4 titlesz=8 labelfat=4 labelsz=8 screenratio=1.0
              allpos=n scalebar=n n1tic=50 n2tic=40
              title="First Arrival Removed"
       ''')

dt = 0.0015
# stretch velocity from depth to time
Flow('vel_t','vpmigration',
     'sfdepth2time velocity=${SOURCES[0]} dt=%g nt=2000'%(dt))
# generate rms velocity :
# input is vel_t the vlocity stretched to time
# sfmul multiply input data by itself to make v**2
# sfcausint integrate v**2 (this does not include scaling by dt)
# sfmath scale by dt and divide by t
Flow('vrms','vel_t',
     '''
     sfmul $SOURCE |
     sfcausint |
     sfmath output="sqrt(input*%g/(x1+%g))" |
     sfput n3=1 d3=1 o3=0
     '''%(dt,dt))

# sftahnmo requires velocity in m/s for every cdp.
# d2=12.49, but cdp int is half this (ie 6.245 or 12.49/2)
# interleave the velocity to double the number of traces.  Should
# interpolationm but this will be pretty good.

Flow('vrms2_m','vrms vrms',
     '''
     interleave axis=2 ${SOURCES[1]} |
     put d2=6.245
     ''')

#Make an empty file for the segy headers.  SEGY standard contains 91 rows
Flow('mutearr_hdr.rsf',None,
     '''
     math n1=91 n2=4000 output="0"
     ''')

# build trace headers for the synthetic traces.  Will be used to sort traces.
Flow('Mmutearr.rsf Mmutearr_hdr.rsf','mutearr.rsf mutearr_hdr.rsf',
     '''
         sftahread input=${SOURCES[0]} makeheader=y
         | sftahheadermath outputkey=gx output="gx"
         | sftahheadermath outputkey=sx output="sx"
         | sftahheadermath outputkey=cdpx output="(gx+sx)/2"
         | sftahheadermath outputkey=cdp output="int(cdpx/6.245)"
         | sftahheadermath outputkey=offset output="abs(sx-gx)"
         | sftahgethw key=sx,gx,cdp,cdpx,offset
         | sftahwrite
        verbose=1
        mode=seq
        output=${TARGETS[0]}
        outheaders=${TARGETS[1]}
```

```
      ''',stdout=0,stdin=0)

# make cdp gathers for displays. Not used in this script
Flow('CDP CDP_hdr','Mmutearr.rsf Mmutearr_hdr.rsf',
       '''
       sftahsort input=$SOURCE sort="cdp offset"
       | sftahwrite output=${TARGETS[0]}
         mode=seq
       ''',stdout=0,stdin=0)

# Sort to cdp gathers, remove selay in ricker wavelet,
# gain for spreading correction, biuld headers used by sftahnmo to read
# space variant velocity, and stack the data
Flow('stack.rsf stack_hdr.rsf','Mmutearr.rsf Mmutearr_hdr.rsf vrms2_m.rsf',
      '''
       sftahsort input=$SOURCE sort="cdp offset"
       | sftahheadermath outputkey=sstat output=-250
       | sftahstatic sign=-1
       | sftahgain tpow=1.5
       | sftahheadermath outputkey=xline output=cdp*6.245
       | sftahheadermath outputkey=iline output=0
       | sftahnmo vfile=vrms2_m.rsf
       | sftahstack key=cdp xmute=0,5000 tmute=0,5
       | sftahwrite output=${TARGETS[0]} mode=seq
      ''',stdout=0,stdin=0)
# can use this line for v(t) velocity in sftahnmo
#       | sftahnmo tnmo=0,1,2,2.5,3,4 vnmo=1510,1600,1700,1800,1900,1980

# Display the stack
Result('stack',
        '''
        grey pclip=99 titlefat=4 titlesz=8 labelfat=4 labelsz=8 barwidth=0.6
        allpos=n scalebar=n n1tic=50 n2tic=40 verb=y screenratio=1.0
        title="NMO and Stack"
        ''')

# Kirchhoff migration and display
Flow('ktmig','stack.rsf vrms2_m.rsf',
      '''
      put d2=6.25 |
      kirchnew velocity=${SOURCES[1]}
      ''')
Result('ktmig',
        '''
        grey pclip=98 titlefat=4 titlesz=8 labelfat=4 labelsz=8 barwidth=0.6
        allpos=n scalebar=n n1tic=50 n2tic=40 verb=y screenratio=1.0
        title="Kirchhoff Post Stack Mig"
        ''')

# Kichhoff least squares migration and display.
Flow('invsparse err_sparse','stack vrms2_m.rsf',
      '''
      put d2=6.25 |
      kirchinvs velocity=${SOURCES[1]} niter=1 liter=3 ps=1
      err=${TARGETS[1]} verb=1
      ''')
Result('invsparse',
        '''
        grey pclip=98 titlefat=4 titlesz=8 labelfat=4 labelsz=8 barwidth=0.6
        allpos=n scalebar=n n1tic=50 n2tic=40 verb=y screenratio=1.0
        title="Least-Squares Kirchhoff Mig"
        ''')

End()
```

REFERENCES

Irons, T., 2005, Marmousi2 model: `http://www.ahay.org/RSF/book/data/marmousi2/paper_html`.

Martin, G., K. Marfurt, and S. Larsen, 2002, Marmousi-2: An updated model for the investigation of AVO in structurally complex areas: 72nd Ann. Internat. Mtg, Soc. of Expl. Geophys., 1979–1982.

Schleicher, K., 2017, Seg seismic working workshop - reproducible tutorials: `http://www.ahay.org/wiki/HoustonWW2017`.

Yang, P., 2014, A numerical tour of wave propagation: `http://www.ahay.org/RSF/book/xjtu/primer/paper_html`.