GEO 365N/384S Seismic Data Processing
Final Project

Team: Circle

ABSTRACT

In the final project, we processed a deep water 2D seismic line collected near the coast of Japan, over the Nankai Trough subduction zone. We went through the processing steps from raw data to surface-consistent correction, CMP sorting, gain, spherical divergence correction, velocity analysis, NMO, DMO, stack, and migration. The final processing result can be used for geologic interpretations.

DATA OVERVIEW

Geology and data acquisition

1. The data were collected near the coast of Japan, over the Nankai trough, where the Philippines plate is subducting beneath Eurasia. The Shikoku Basin section of the northern Philippine Sea plate has been subducting northwestward under southern Japan along the Nankai Trough (Figure 1).

2. 8 two-ship expanding spread profiles (ESPs), 6 split spread profile (SSPs), and 250 km of 96-channel, high-resolution multichannel seismic reflection (MCS) profiles were acquired in the Nankai Trough. Our 2D seismic line is NT62-8 in Figure 2.

3. The deformed sediments in the Nankai Trough consist of a terrigenous trench wedge overlying a Shikoku Basin sequence. Along line NT62-8 the trench sediments are less than 350m thick at the deformation front, and the trench wedge is about 12 km wide. The subduction-related deformation begins seaward of the base of the inner trench slope. The protothrust zone developed seaward of the first thrust is 2.5 km wide and is characterized by thickening and seaward tilting of the trench wedge (Figure 4). The decollement is localized near the top of the Shikoku Basin lower pelagic unit. There are thrusts after the protothrust zone (Figure 3).

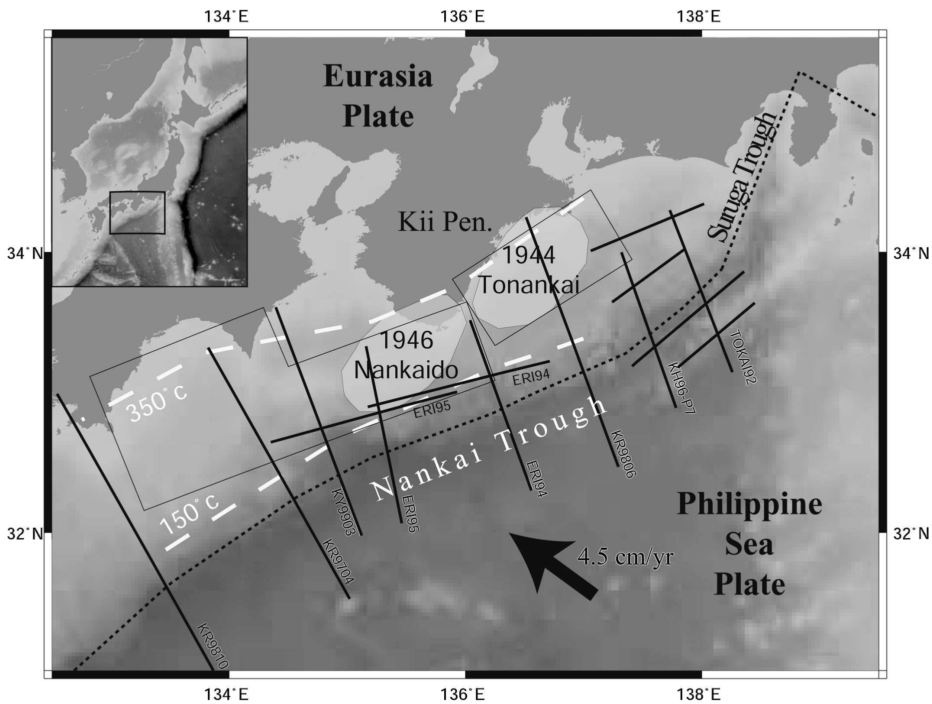


Figure 1: Location of the Nankai Trough (Image from Mochiduki and Obana (2003)). project/ geo

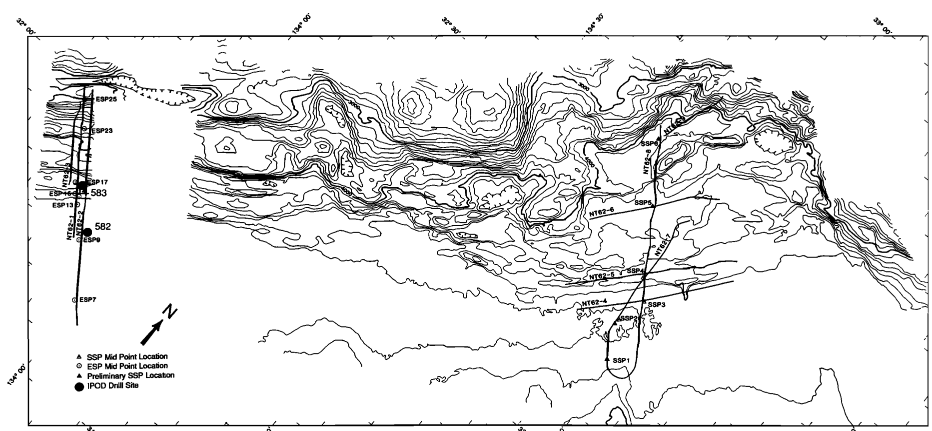


Figure 2: Seismic acquisition (Image from Mochiduki and Obana (2003)). project/ acquisition

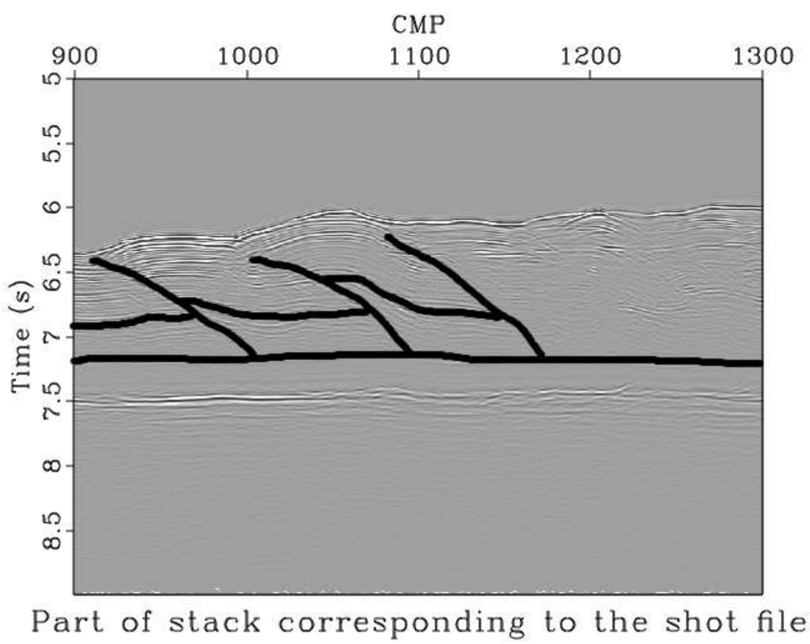


Figure 3: Geologic structure. project/ stack-shot-interpret

First look at data

1. The data were collected by the University of Texas, the University of Tulsa, and the University of Tokyo. The original data is in SU, so we converted it to RSF files. The processed dataset was published in Moore et al. (1990). Field data were sorted into 16.66-m bins.

2. There are two data files: shot-ordered gathers (Figure 7a) and published stacked data (Figure 4). The data were collected in very deep water, approximately 6 seconds two-way travel time, which is 4.5 km if the water velocity is 1500 m/s. The shot-gather data has 326 shot gathers with 19057 traces. The total time length of a trace is 11 seconds with a sample interval of 0.002 seconds. The published stacked data has 2250 traces with total time length of 11 seconds and 0.004 seconds sample interval. The shot-gathers data has 401 CMPs while the stacked data is the output of 2869 CMPs. Therefore, the shot-gather file is a subset of stacked data.

3. The number of traces per shot gather is Figure 5. The first shot gather, 1687, has one trace, then the number of traces per gather increases to a maximum of 69 at gather 1735 and keeps constant through gather 1965. After gather 1965, the number of traces per gather decreases to 1 at the last gather, 2012. There are missing traces at some shot gathers, for example at gather 1707 (Figure 6). The average frequency spectra of shot gathers is Figure 8a. Our data has low frequencies smaller than 10 Hz that need to be filtered. We later followed the example of the published dataset (Moore et al., 1990) to resample the file from 2 ms to 4 ms sample interval to reduce our prestack processing load by half. The Nyquist frequency of 4 ms sample interval is 125 Hz. Typical seismic data has good high frequency content up to around 50-70 Hz, above this are usually noise. Therefore, we filtered high frequencies above 125 Hz to prevent aliasing and remove noise.

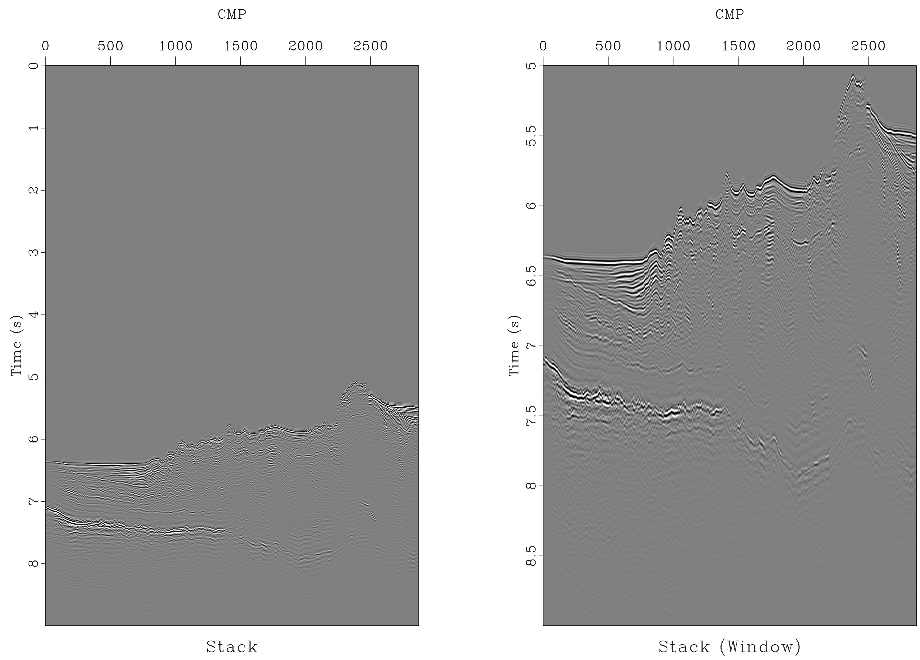


Figure 4: Published stacked data. project/ stackd

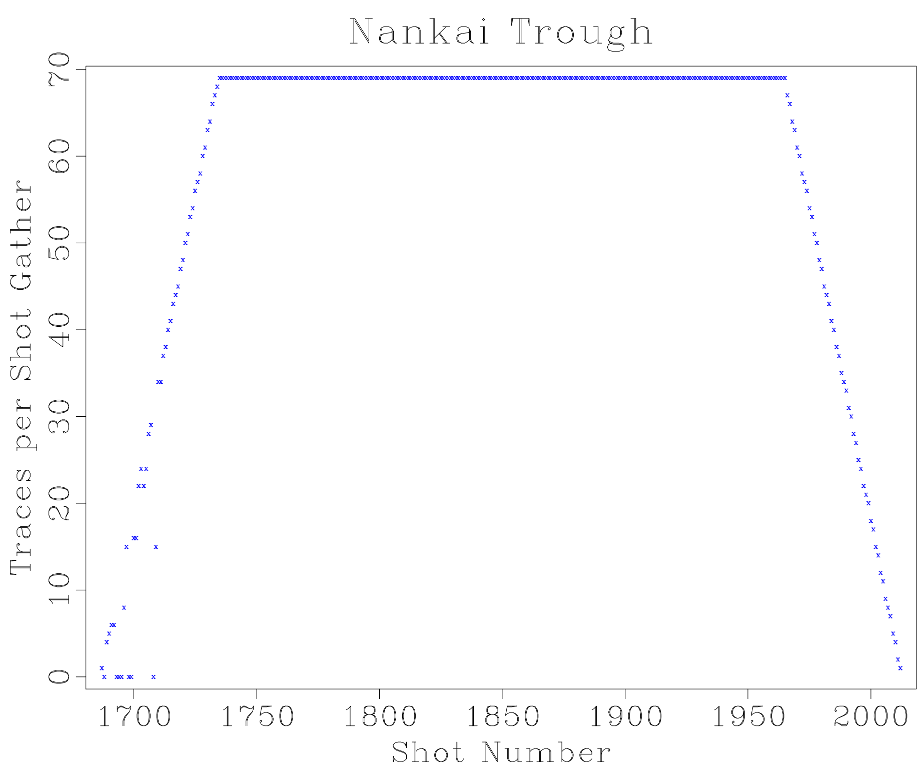


Figure 5: Number of traces per shot-gather. project/ smask

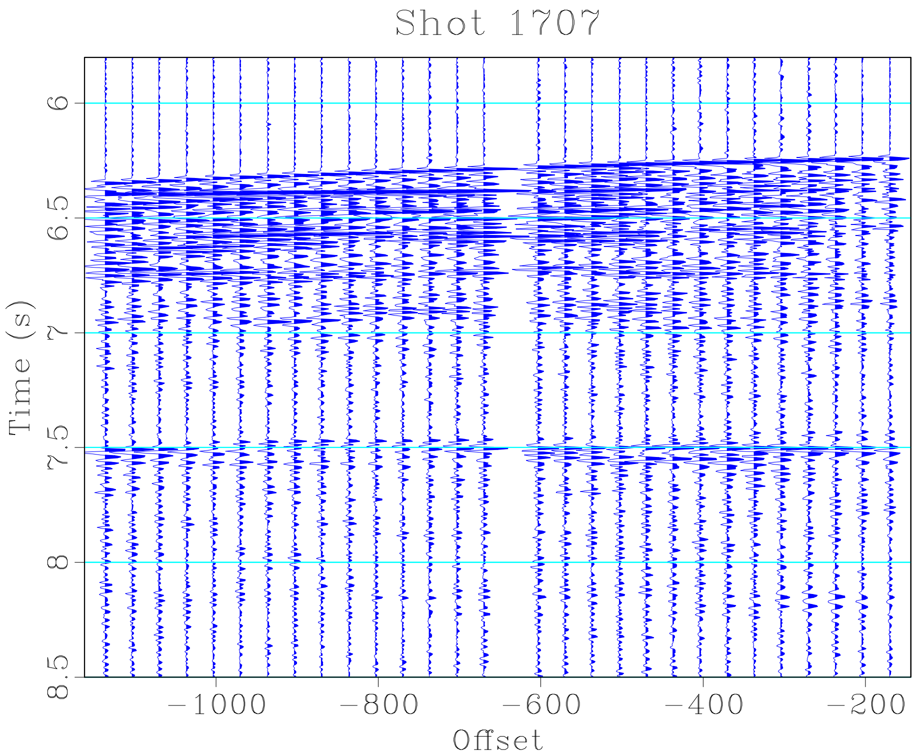


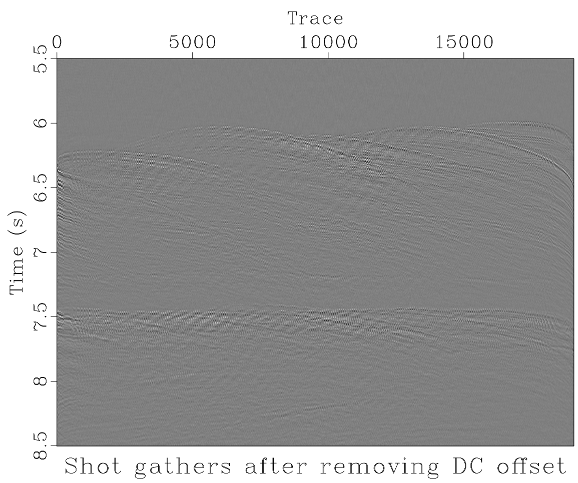
Figure 6: Shot-gather 1707. project/ shot

SURFACE-CONSISTENT AMPLITUDE BALANCING

1. The first step is to do surface-consistent amplitude balancing. Typical seismic instruments introduce significant direct current (DC) offset that is effectively added to the desired signal from the sensor. We removed DC offset from the shot gathers data to facilitate processing and analysis of data (Figure 7b). We then filtered the low frequencies and high frequencies in the data (Figure 7c).

2. We created a mask to remove zero traces and calculated trace amplitudes of the shot gathers displayed in shot-offset coordinates (Figure 10a). The stripes in the amplitude might be caused by near-surface conditions in deploying sources and receivers. The horizontal stripes come from the offset term (Figure 9a). The vertical stripes come from the shot term (Figure 9b). The diagonal stripes come from the CMP and receiver terms (Figure 9c and Figure 9d). The surface-consistent model (Taner and Koehler, 1981) tries to explain the trace amplitude using a product of source, receiver, offset, and midpoint factors. After running iterative inversion using the least-squares method and the conjugate-gradient algorithm (Hestenes and Stiefel, 1952; Fletcher and Reeves, 1964), the stripes are predicted (Figure 10b). The shot gathers after the surface-consistent amplitude correction are the right figure in Figure 11.

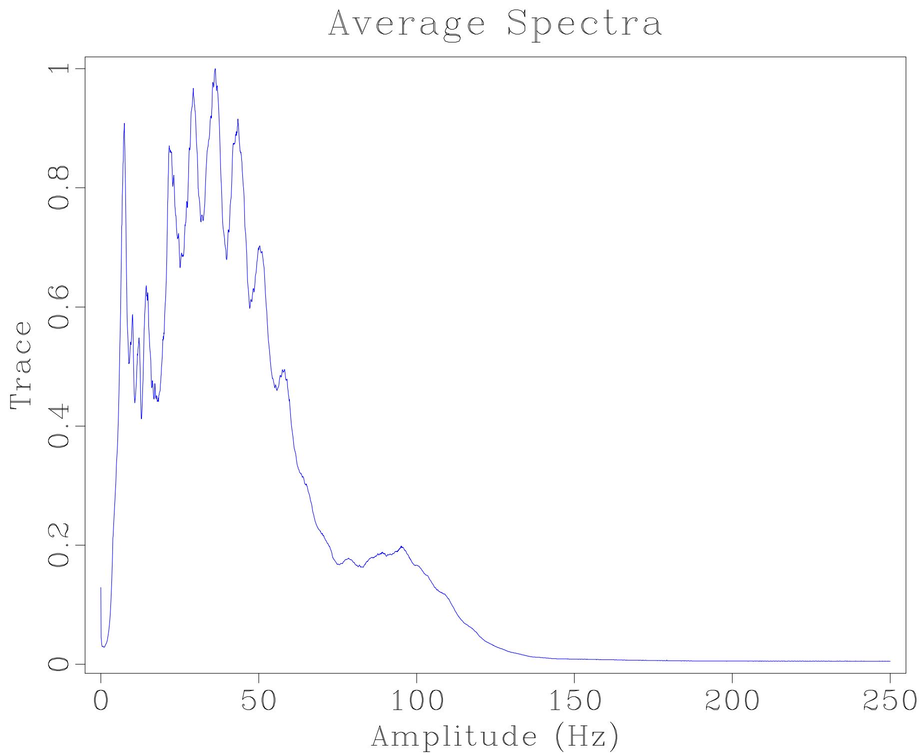
(a)



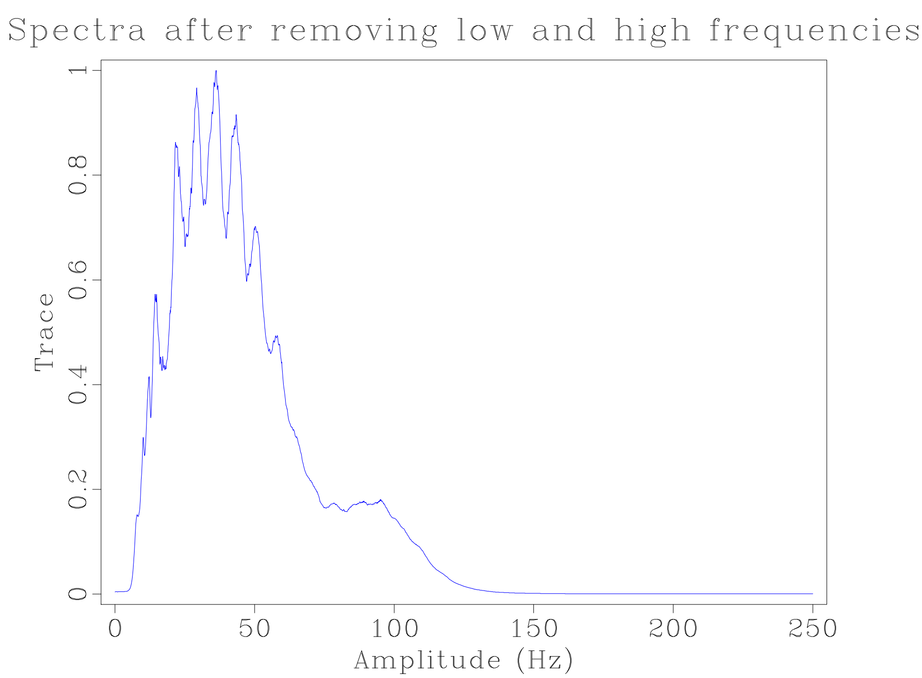
(b)

(c)

Figure 7: (a) Raw shot gathers (b) Shot gathers after DC removal. (c) Shot gathers after filtering low and high frequencies. project/ shots,shotsdc,shotsf



(a)



(b)

Figure 8: (a) Raw data frequency spectra. (b) Frequency spectra after filtering low and high frequencies. project/ spectra,spectraf

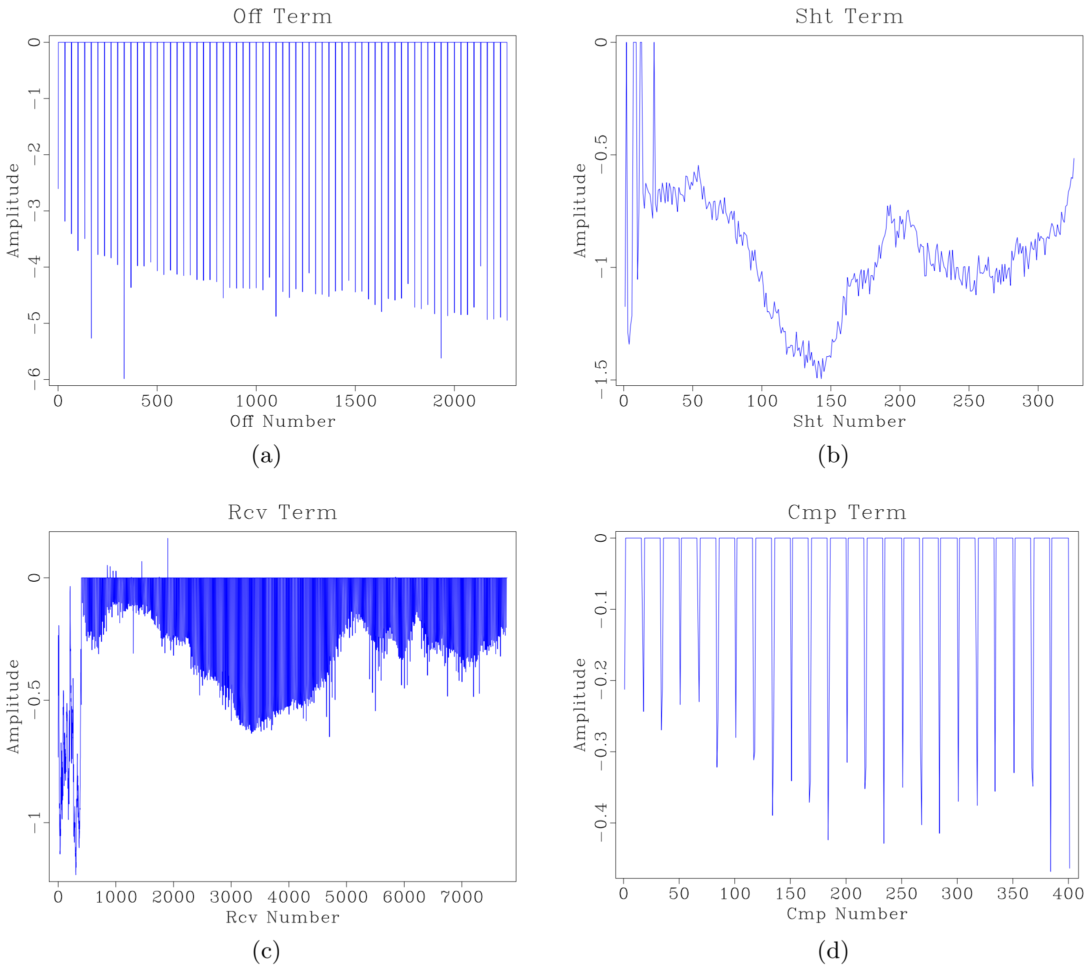


Figure 9: Estimated surface-consistent factors. project/ off,sht,rcv,cmp

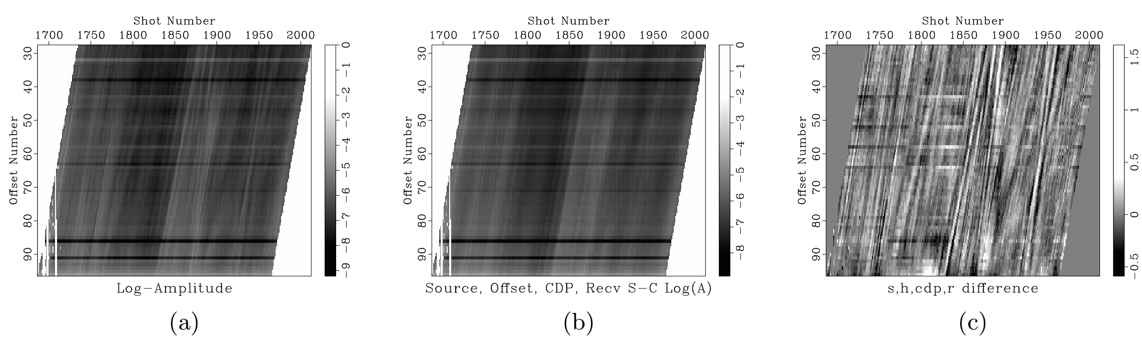


Figure 10: Trace amplitudes from the data. (a) Initial. (b) Estimated surface-consistent. (c) Difference. project/ varms,recvvscarms,recvadiff

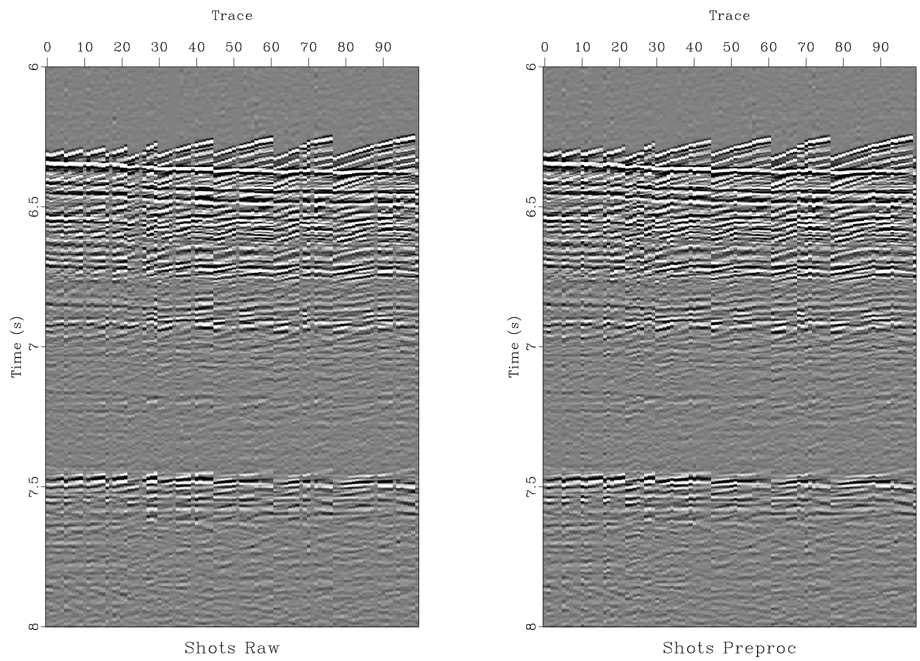


Figure 11: (a) Raw shot gathers (b) Shot gathers after surface-consistent amplitude correction. project/ shotsfc

CMP SORTING

1. The shot gathers are resampled to 4 ms sample interval and applied spherical divergence correction (Figure 12). We then sorted the data to CMP (Figure 13). The fold values of data is Figure 14. We have 401 CMPs but the first CMPs are not full fold. The first full-fold CMP gather is CMP 933 with 48 traces.

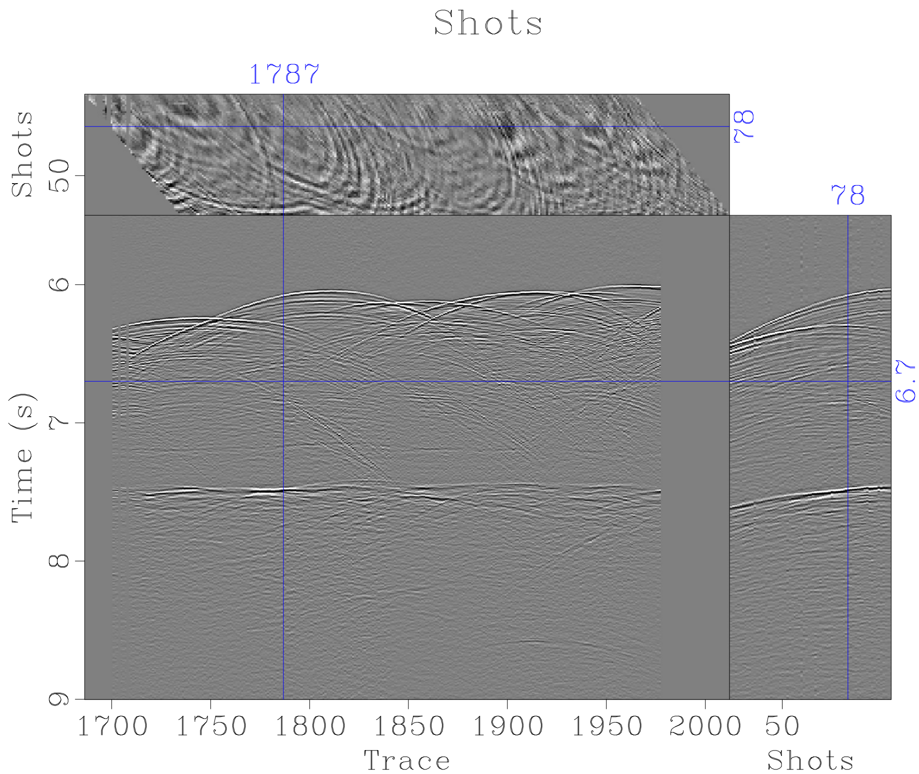


Figure 12: Shot gathers after resampling. project/ shots2

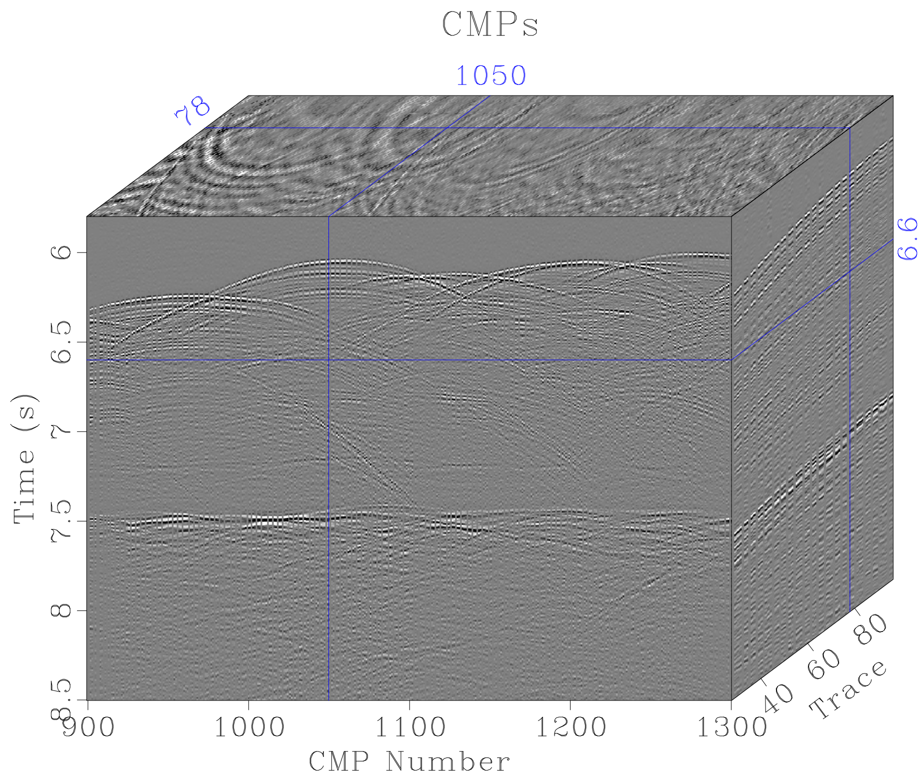


Figure 13: CMP gathers. project/ cmps

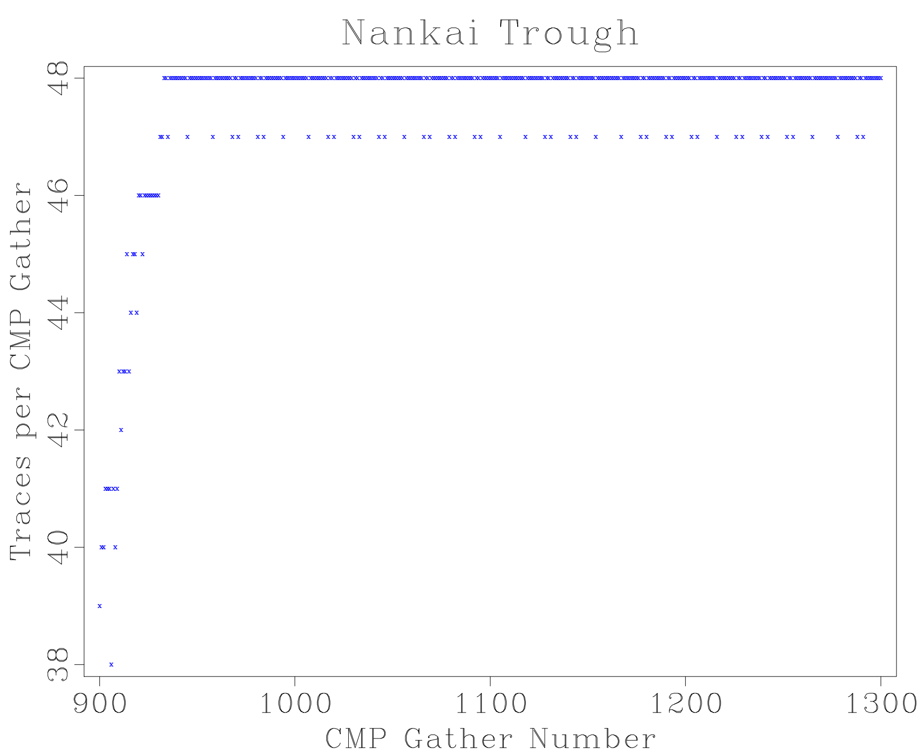


Figure 14: Fold values of data. project/ cmask

VELOCITY ANALYSIS, NMO, DMO, AND STACK

1. We examined CMP gather 1280 (Figure 15). We did velocity analysis using semblance scan and automatically picked the velocity (Figure 16). We then applied NMO using picked velocity to flatten the events (Figure 17).

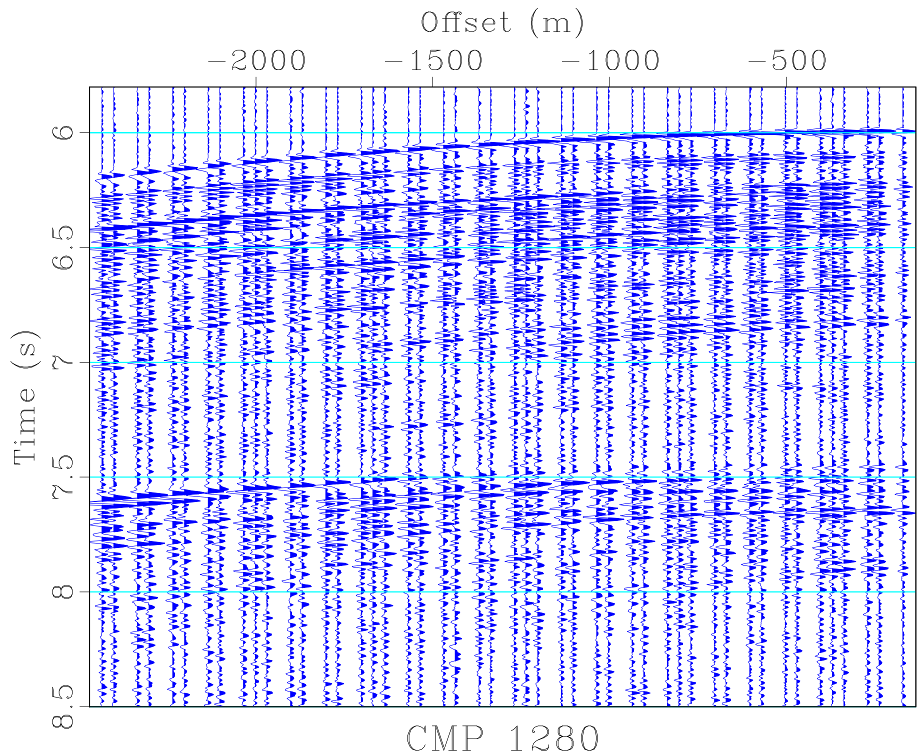


Figure 15: CMP 1280. project/ cmp1

2. We applied the same procedure with all CMP gathers. NMO velocity (Figure 18) was picked using semblance scan. The events were flatten (Figure 19). We then stacked all the CMP gathers (Figure 20a). Our stack result does not resemble the published stacked data, which is actually a stacked and migrated file.

3. We also tried DMO to correct for dips. We created a constant-velocity stack with 60 velocities starting from 1400 m/s and spacing interval of 20 m/s (Figure 21). We applied Fowler's method (Fowler, 1988) to transform the stack volume into the frequency-wavenumber domain and applied the velocity mapping (Figure 22). The result after applying Fowler's method is Figure 23. We then picked the DMO-corrected velocity automatically from the envelope (Figure 24). After the velocity was picked, we generated a DMO stack by slicing through the velocity cube (Figure 25). We examined CMP 1280 where there is a dipping event to evaluate the velocity picking and to observe the change brought by DMO (Figure 26). The velocity is corrected in the shallow layers to account for the dips. The picked velocity is smoother and increases with depth. We extracted a small window of DMO stack and normal stack for comparison (Figure 27). The events in DMO stack are clearer than in normal stack.

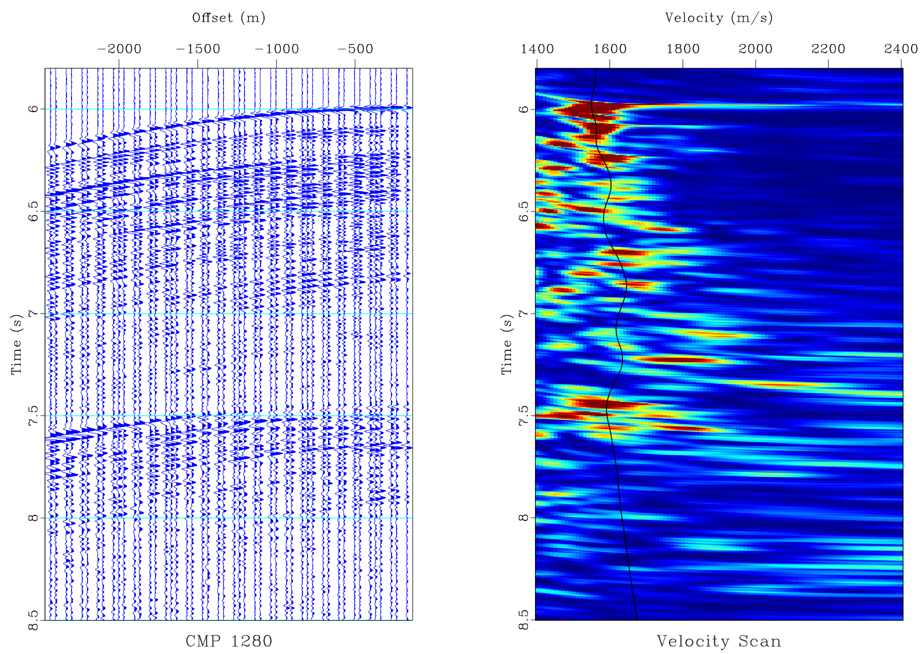


Figure 16: Velocity analysis for CMP 1280. project/ vscan

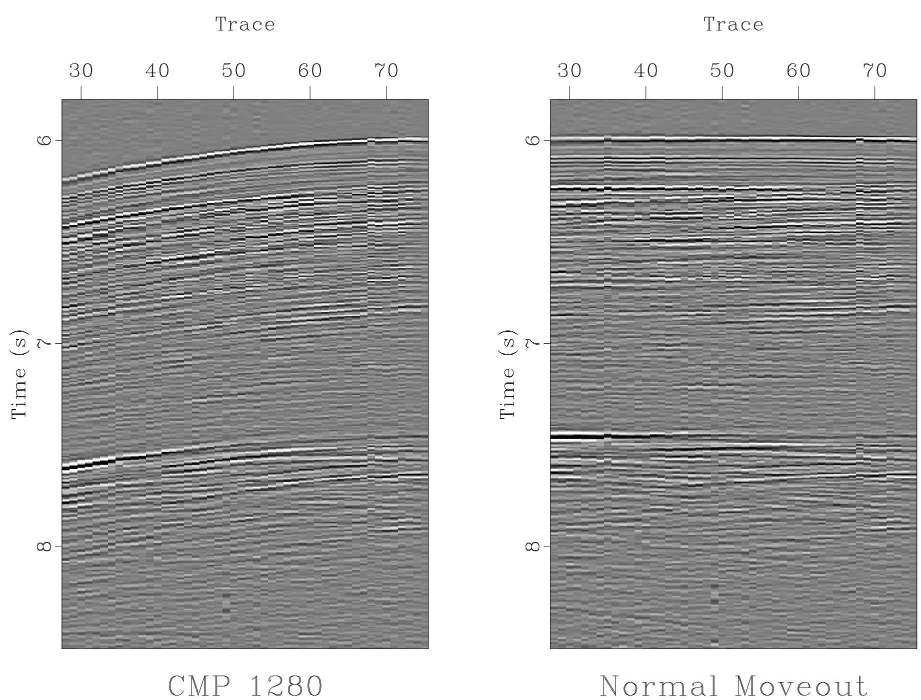


Figure 17: CMP 1280 after NMO. project/ nmo1

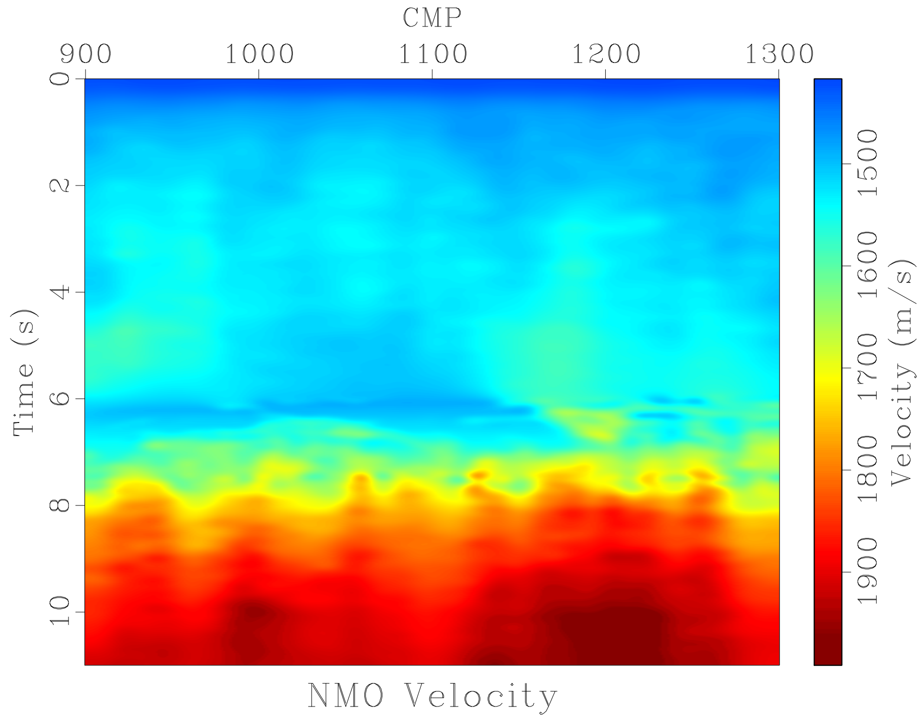


Figure 18: NMO velocity. project/ picks

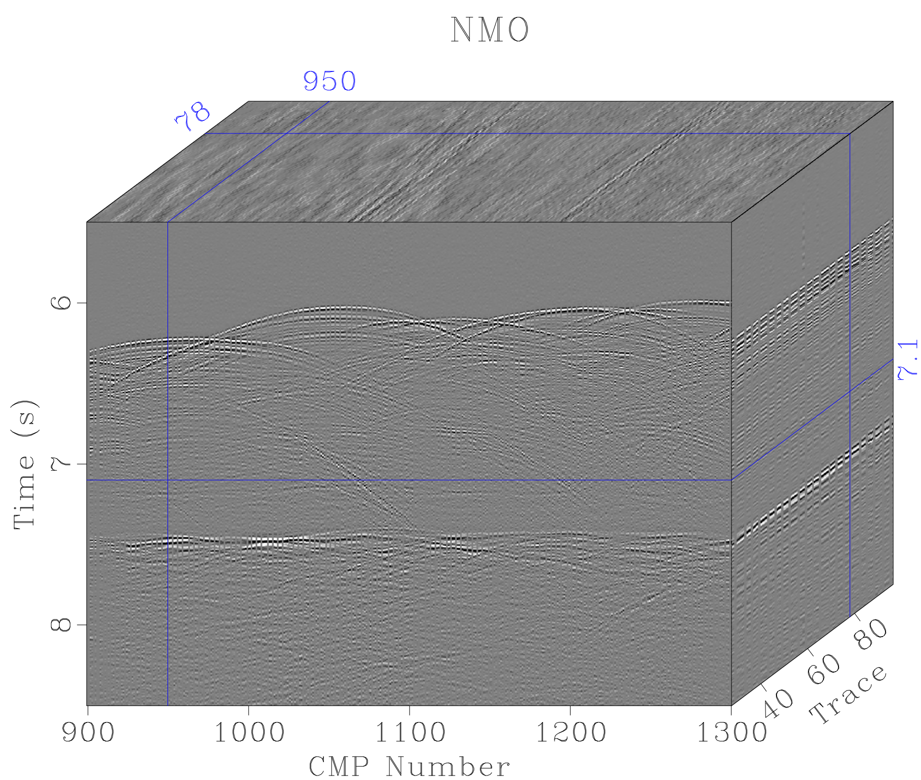


Figure 19: CMP gathers after NMO. project/ nmos

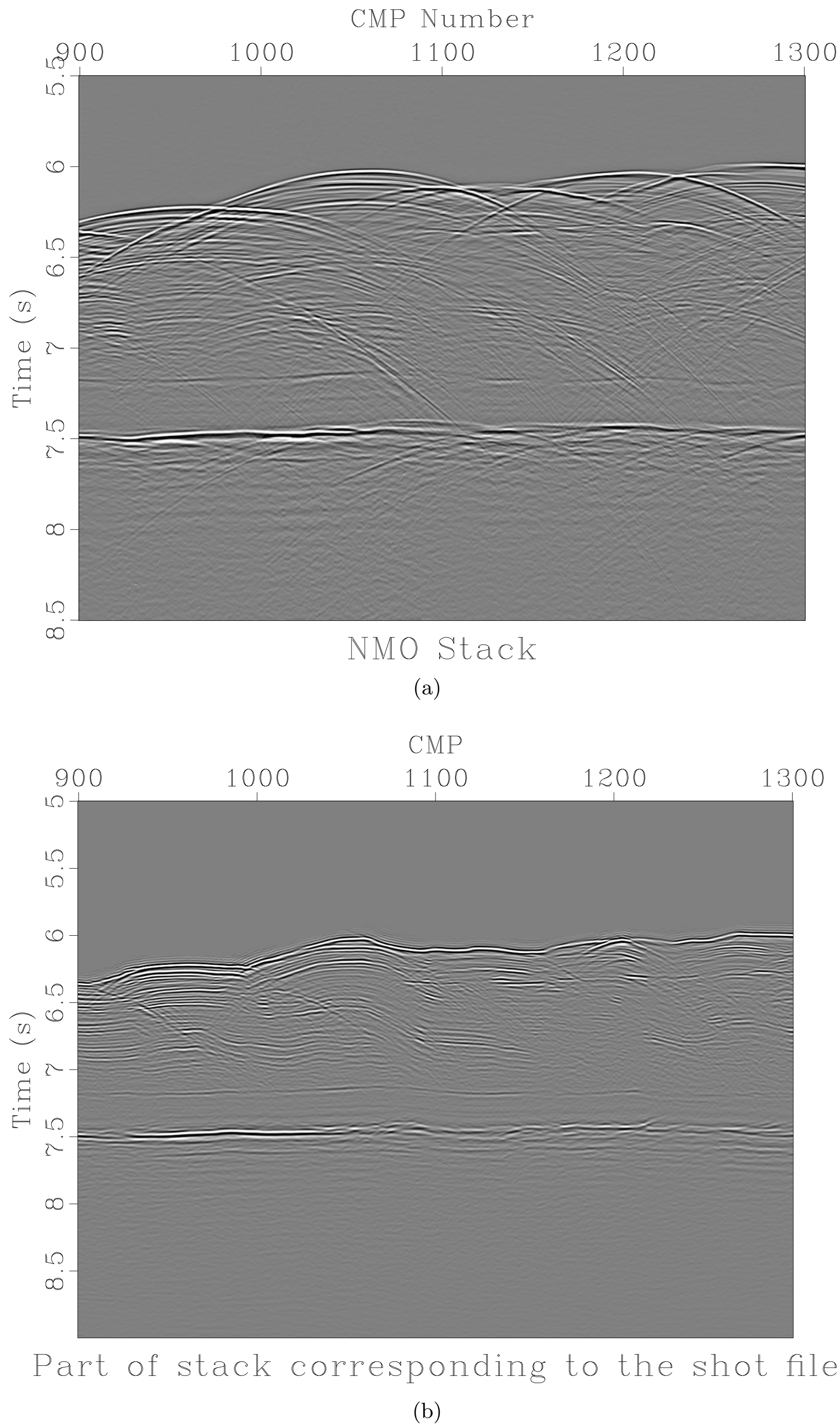


Figure 20: (a) Stacked data. (b) Published stacked data. project/ stack,stack-shot

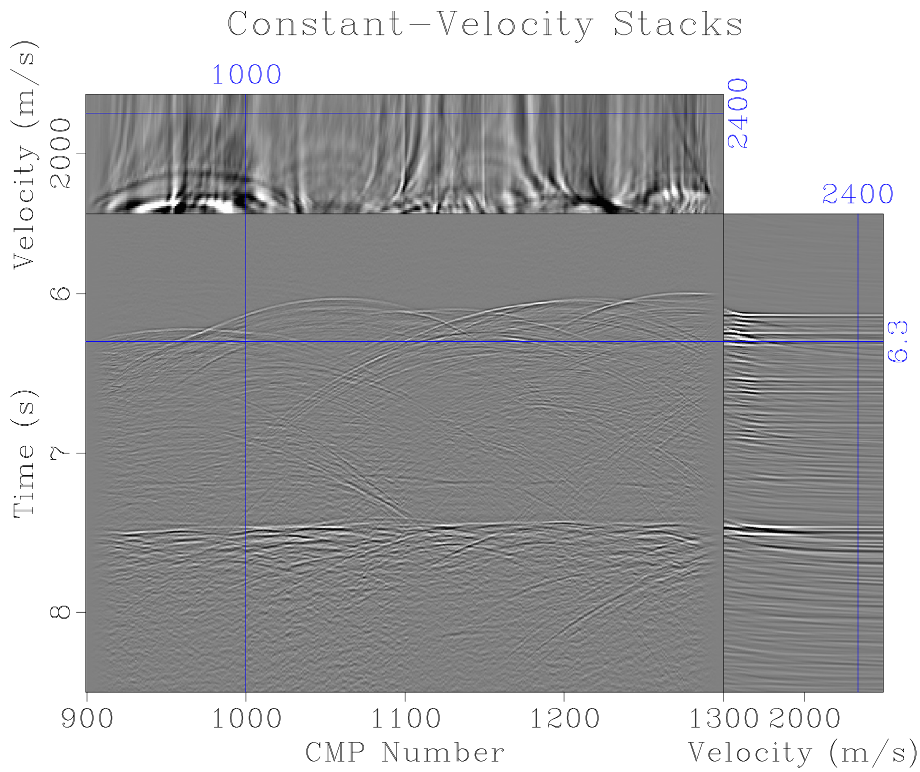


Figure 21: NMO stack with an ensemble of constant velocities. project/ stacks

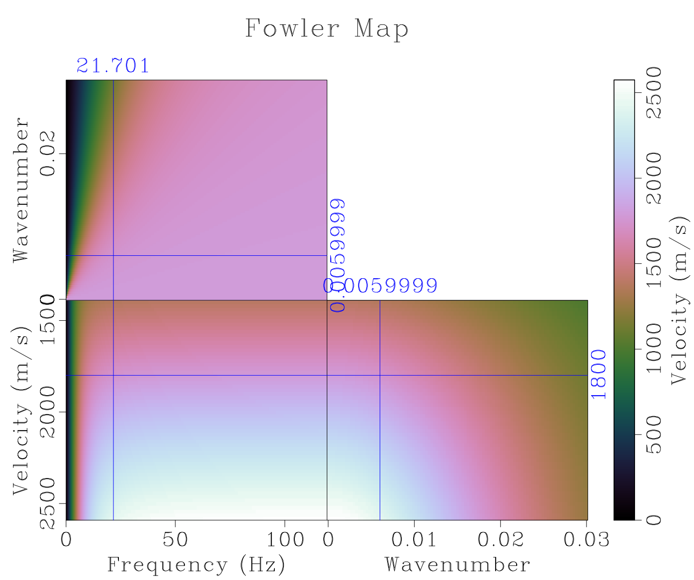


Figure 22: Fourier-domain velocity map used in Fowler's DMO method. project/ map

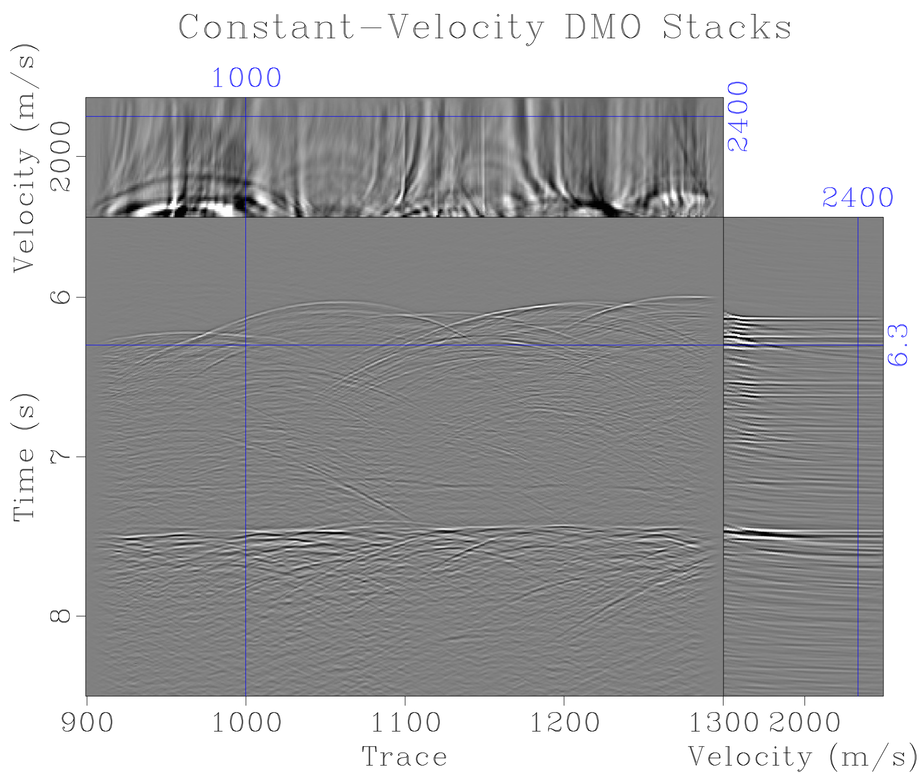


Figure 23: Nankai dataset after DMO stacking with an ensemble of constant velocities. project/ dmo

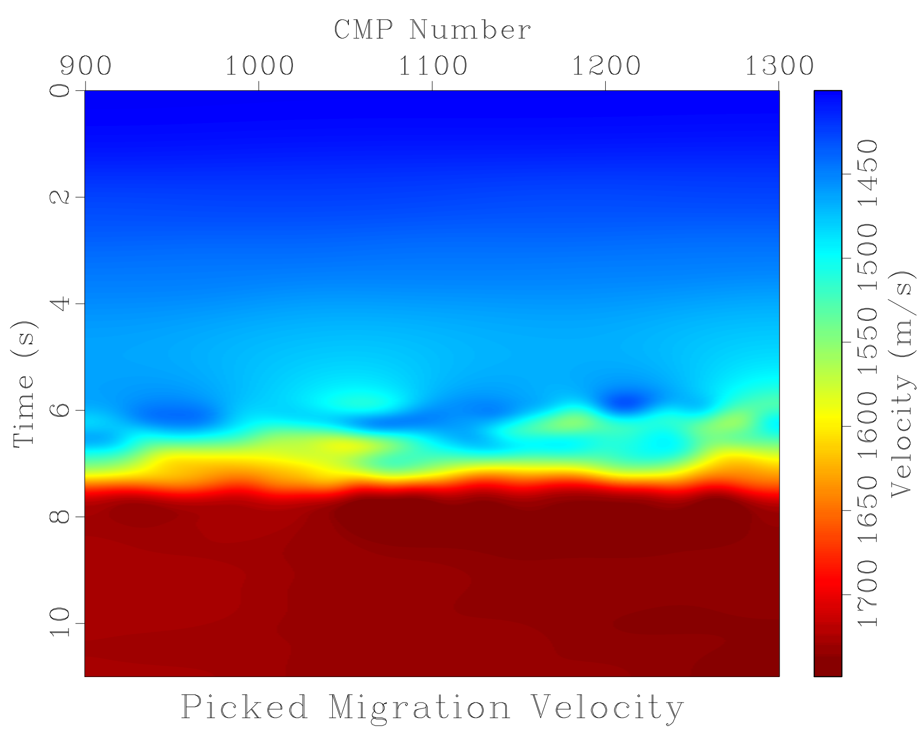


Figure 24: Migration velocity picked automatically from DMO stacks. project/ vpick

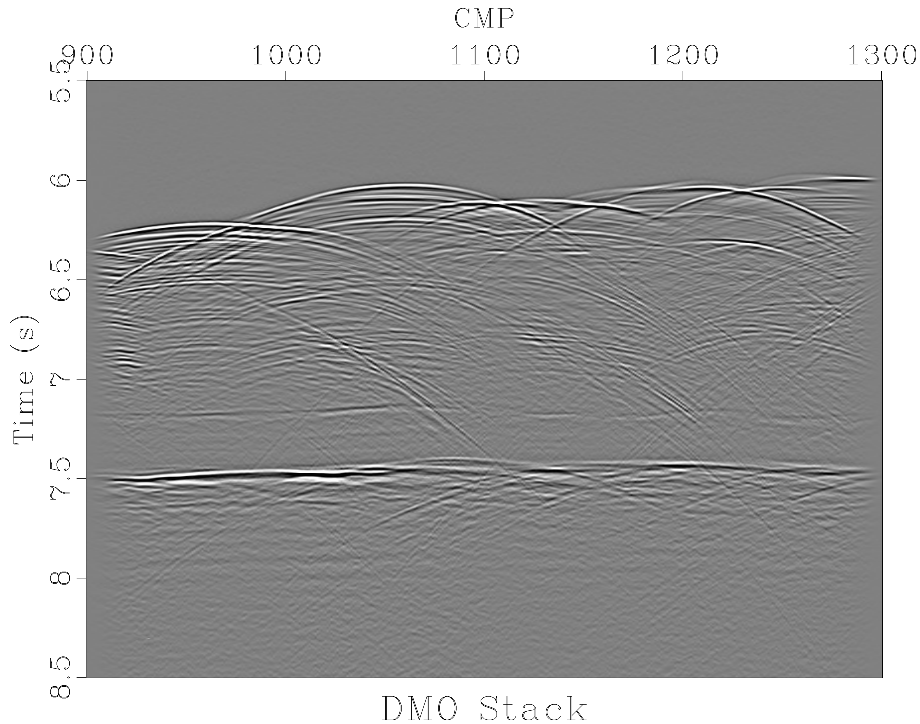


Figure 25: DMO stack generated by slicing the constant-velocity stacks. project/ slice

Figure 26: Comparison of velocity analysis (a) before and (b) after DMO at a selected CMP location. project/ envelope

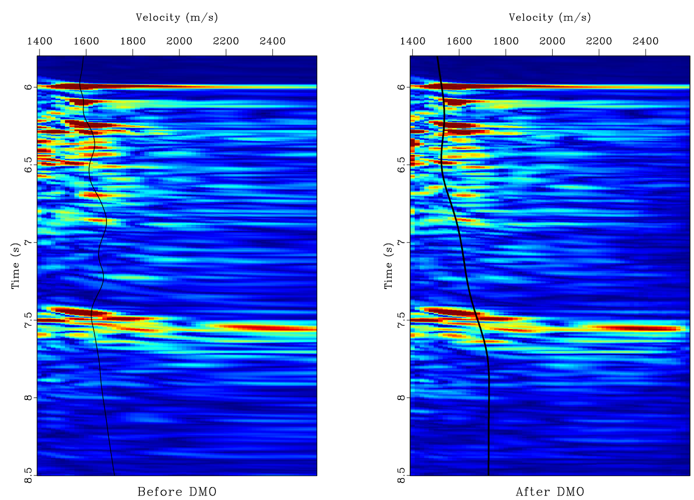
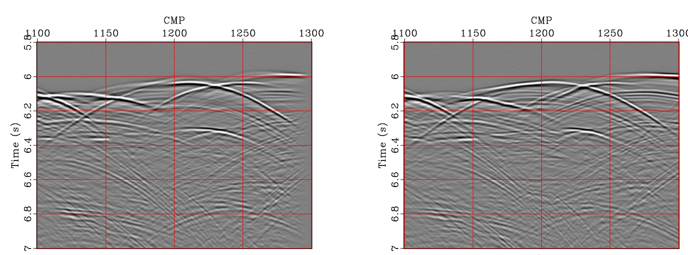


Figure 27: (a) Windowed DMO stack (b) Windowed normal stack. project/ zoomd



MIGRATION

Stolt migration

1. We used Stolt migration based on the Fourier transform (Stolt, 1985). We first mapped from ω to ω_0 in frequency-wavenumber domain (Figure 28). The 2-D cosine transform of the data before and after Stolt mapping is Figure 29. The data after Stolt migration with constant velocity 1500 m/s is Figure 30. The diffractions are not imaged properly because of wrong velocity (Middle figure in Figure 51a, Figure 52a, Figure 53a).

2. We tried a more realistic velocity distribution starting with 1500 m/s (water velocity) at the surface and increasing quadratically with vertical time (Figure 31). We first migrated the data with a number of constant velocities in the range from 1500 to 2452 m/s with the spacing interval of 8 m/s. We then sliced through this ensemble of migrations to create an image (Figure 32) (Mikulich and Hale, 1992). The diffractions are imaged better but still need to be improved by improving the migration velocity (Right figure in Figure 51a, Figure 52a, Figure 53a). Comparing with the published stacked and migrated data (Right figure in Figure 51c, Figure 52c, Figure 53c), the diffractions are more collapsed and the events are clearer.

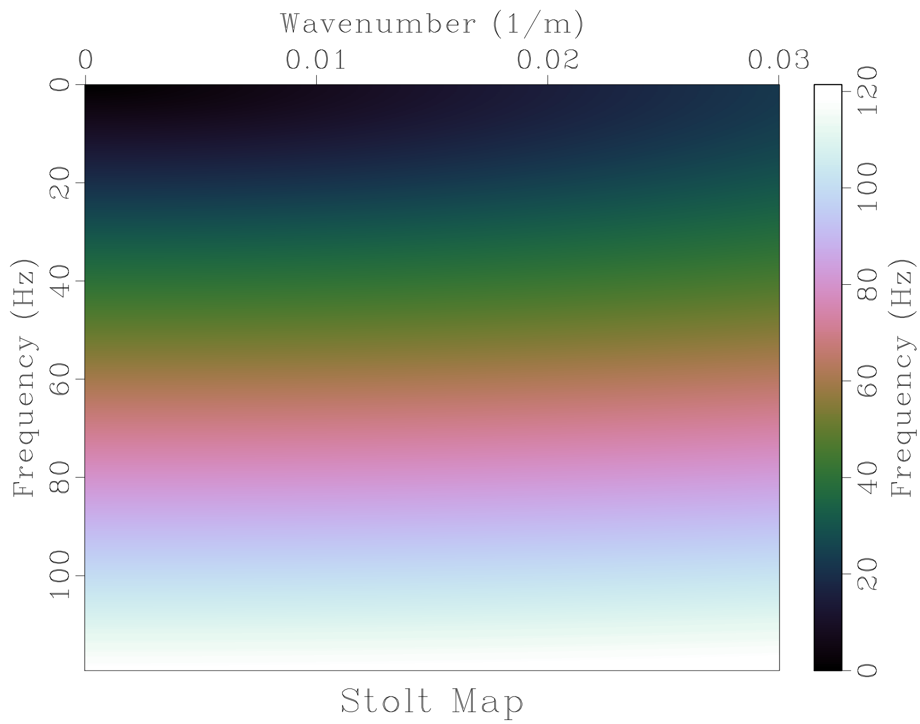


Figure 28: Stolt map. project/ map2

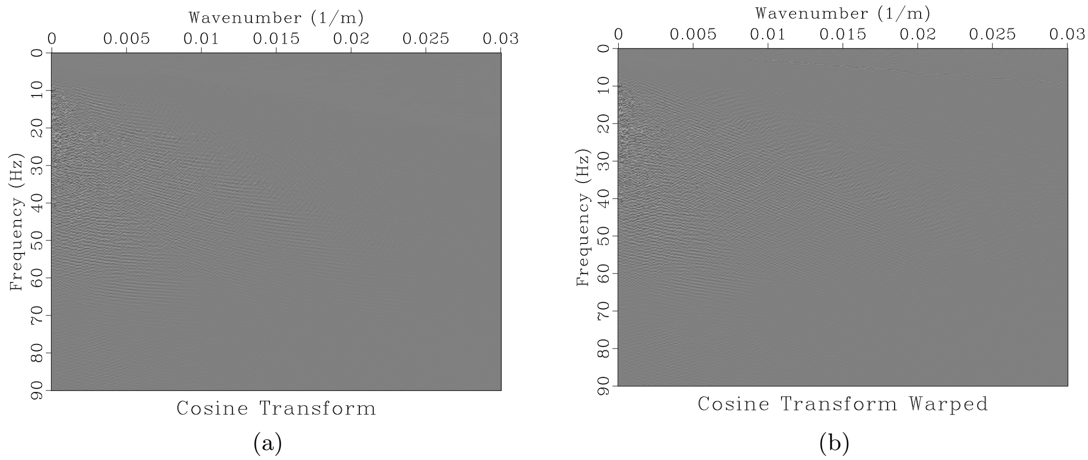


Figure 29: Nankai stack in the Cosine transform domain before (a) and after (b) Stolt migration with velocity 1500 m/s. project/ cosft,cosft2

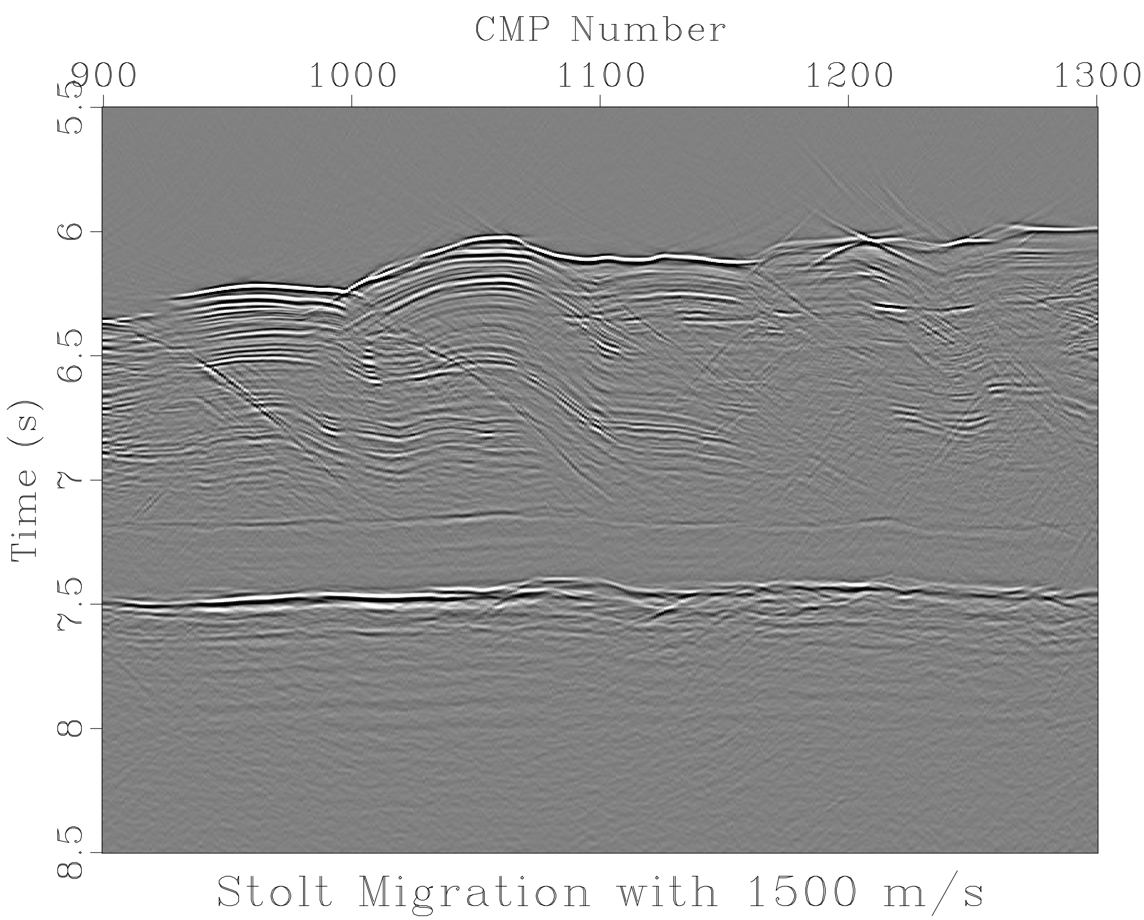


Figure 30: Nankai stack Stolt migrated with velocity of 1500 m/s. project/ mig2

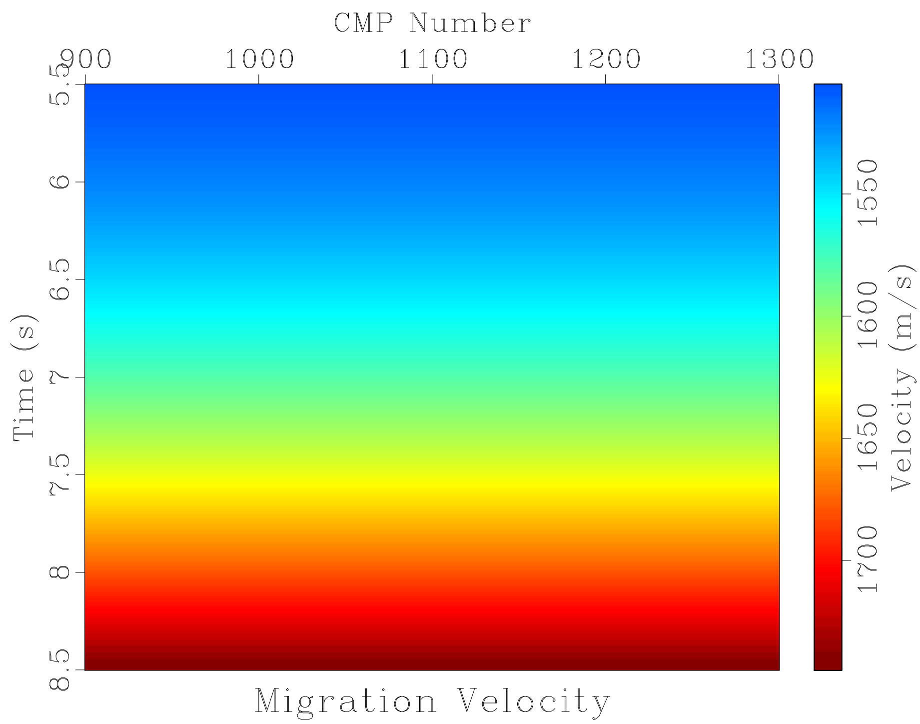


Figure 31: Velocity distribution. project/ vmig

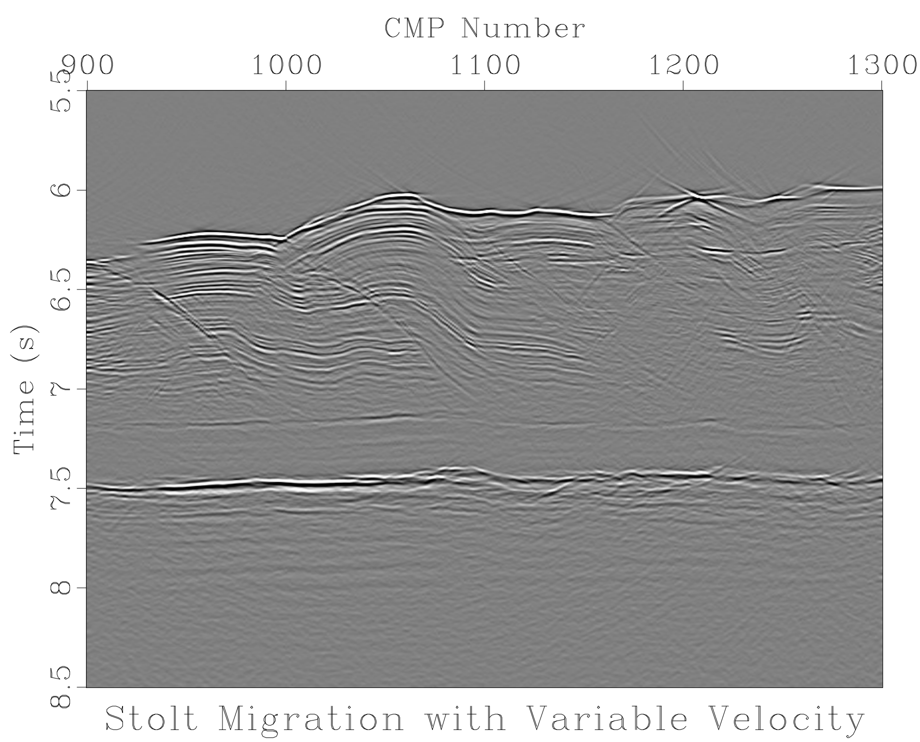


Figure 32: Nankai stack Stolt migrated with velocity increasing with time. project/ migd

Gazdag migration

1. To corretly image the diffractions, we used the picked velocities in DMO for phase-shift migration also known as Gazdag migration (Gazdag, 1978). We used Dix conversion to calculate the interval velocities in time (Figure 33). Stolt migration with constant velocity 1550 m/s images the water bottom clearly (Figure 34) so the water velocity is about 1550 m/s. We set the velocity above 5.5 seconds to 1550 m/s and kept the dix velocities below 5.5 seconds (Figure 35). We then applied phase-shift migration in time coordinate to get the image (Figure 36). The diffractions are clearly imaged (Right figure in Figure 51b, Figure 52b, Figure 53b). Comparing with the published stacked and migrated data (Right figure in Figure 51c, Figure 52c, Figure 53c), the diffractions at ocean bottom are collapsed and the events are clearer.

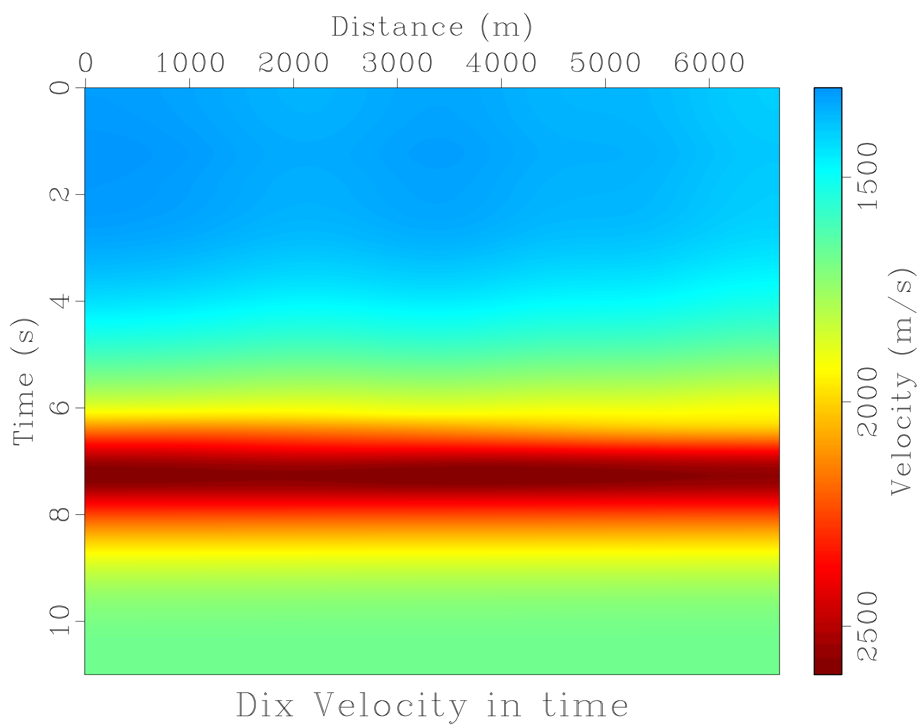


Figure 33: Dix velocity in time. project/ vdix2

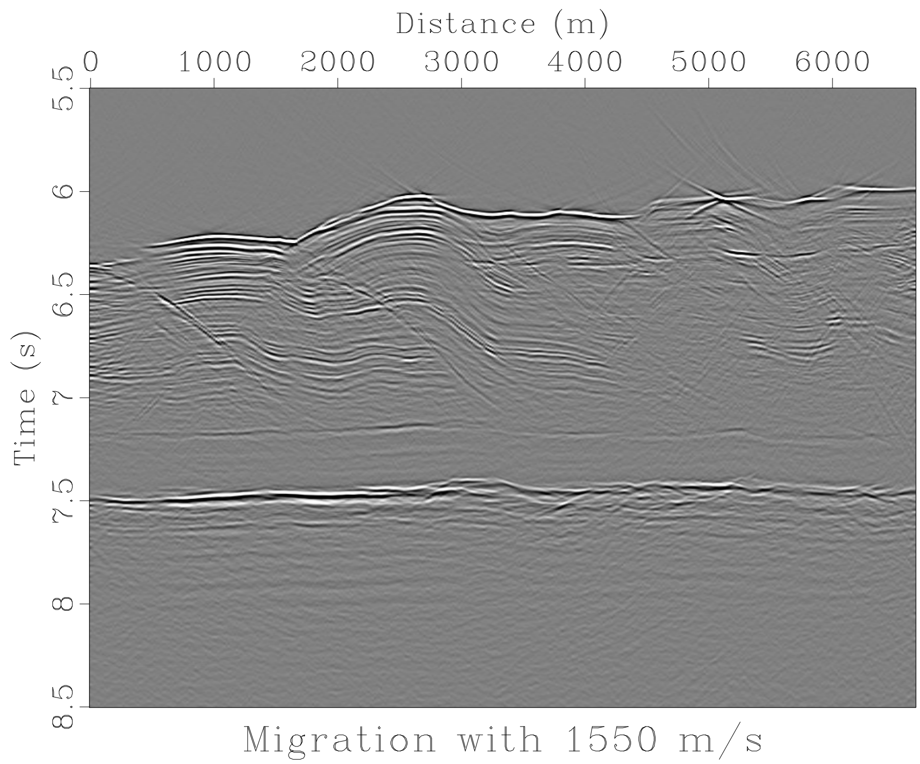


Figure 34: Stolt migration with constant velocity 1550 m/s. project/ mig1

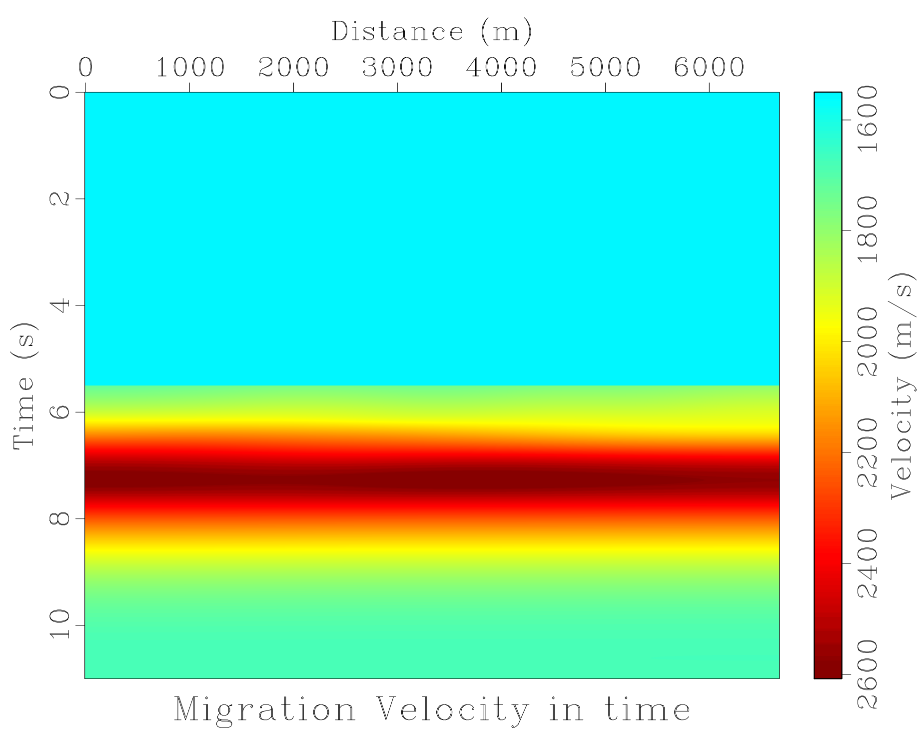


Figure 35: Migration velocity in time. project/ vmigr

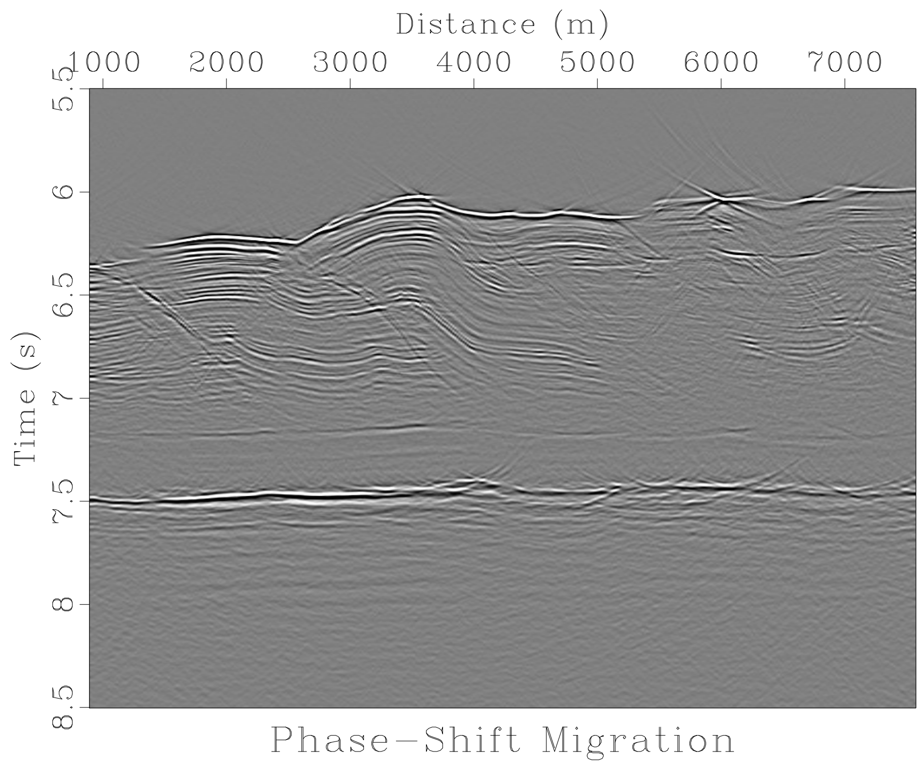


Figure 36: Gazdag migration. project/ image

Velocity continuation

1. In order to have correct migration velocity, we used the velocity continuation method proposed by (Fomel, 2003) to do the time migration velocity analysis. We first used Stolt migration with constant velocity 1400 m/s (Figure 37). We then used velocity continuation with 101 velocities starting with 1400 m and spacing interval of 10 m/s (Figure 38). We then sliced through this velocity continuation volume using stacking velocity to create a migration image (Figure 39). Migrating with stacking velocities is still not sufficient to correctly image the diffractions (Left figure in Figure 51b, Figure 52b, Figure 53b).

2. In order to have more focused velocity to image the diffractions correctly, we used plane-wave destruction (Fomel, 2002) to estimate the dip of reflections and diffractions (Figure 40). We then separated reflections from diffractions (Figure 41). The diffractions are in Figure 42. We then used velocity continuation to analyze time migration diffraction velocity (Figure 43). The focusing velocity was picked (Figure 44). The diffractions are focused (Figure 45). Nankai stack migration with focusing velocity is Figure 46. Focusing velocities improve the quality of migrated image (Middle figure in Figure 51b, Figure 52b, Figure 53b). Comparing with the published stacked and migrated data (Right figure in Figure 51c, Figure 52c, Figure 53c), the diffractions at ocean bottom are collapsed and the events are clearer.

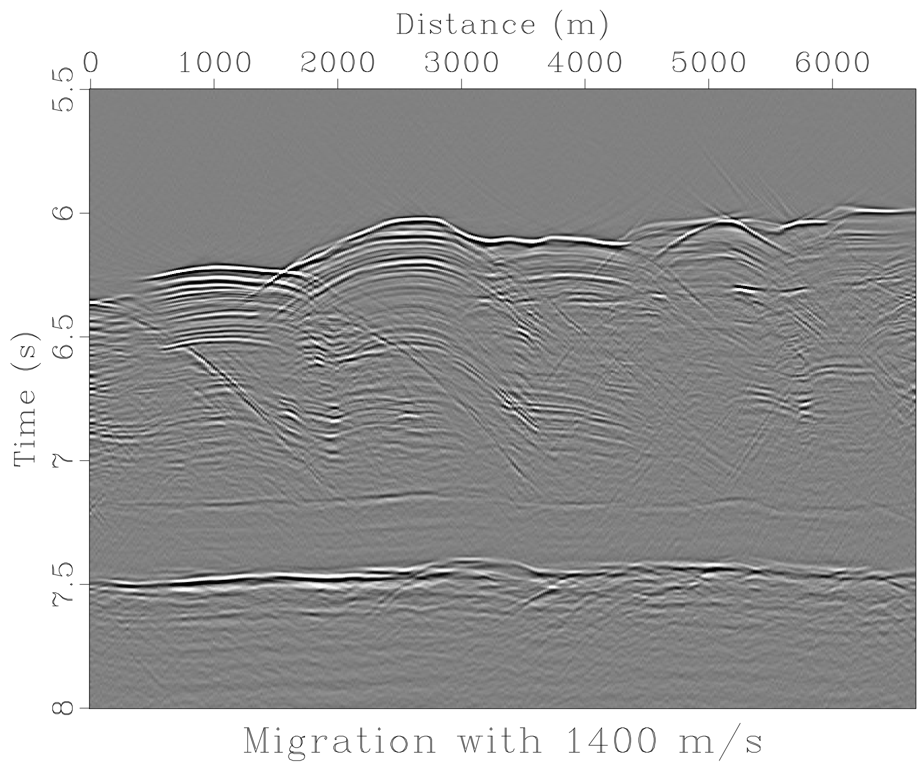


Figure 37: Stolt migration with velocity 1400 m/s. project/ first

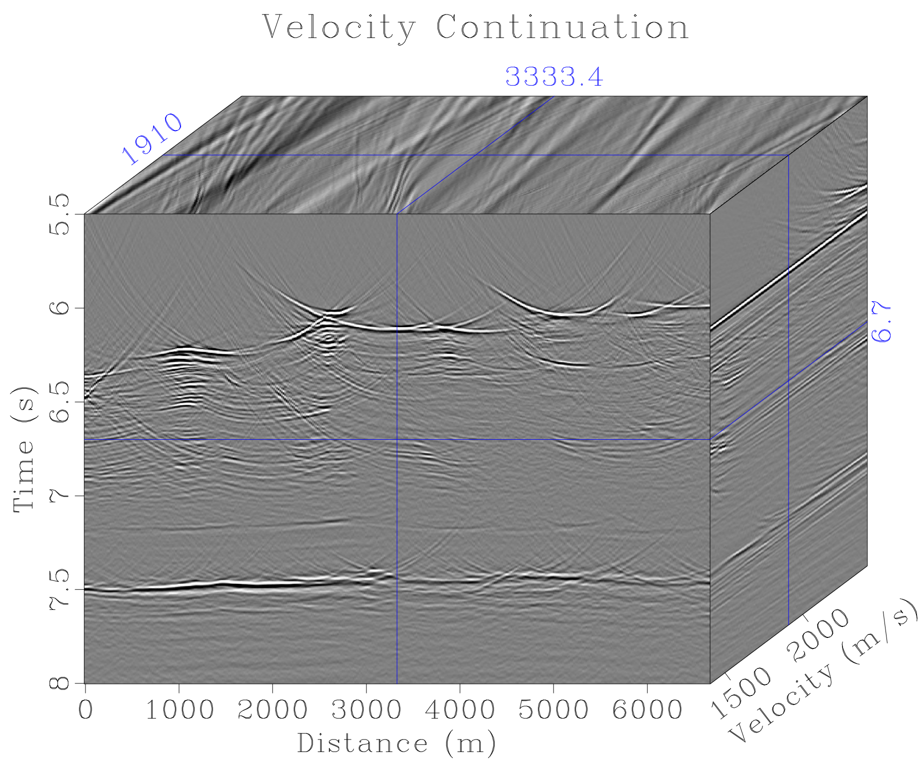


Figure 38: Velocity continuation. project/ velcon

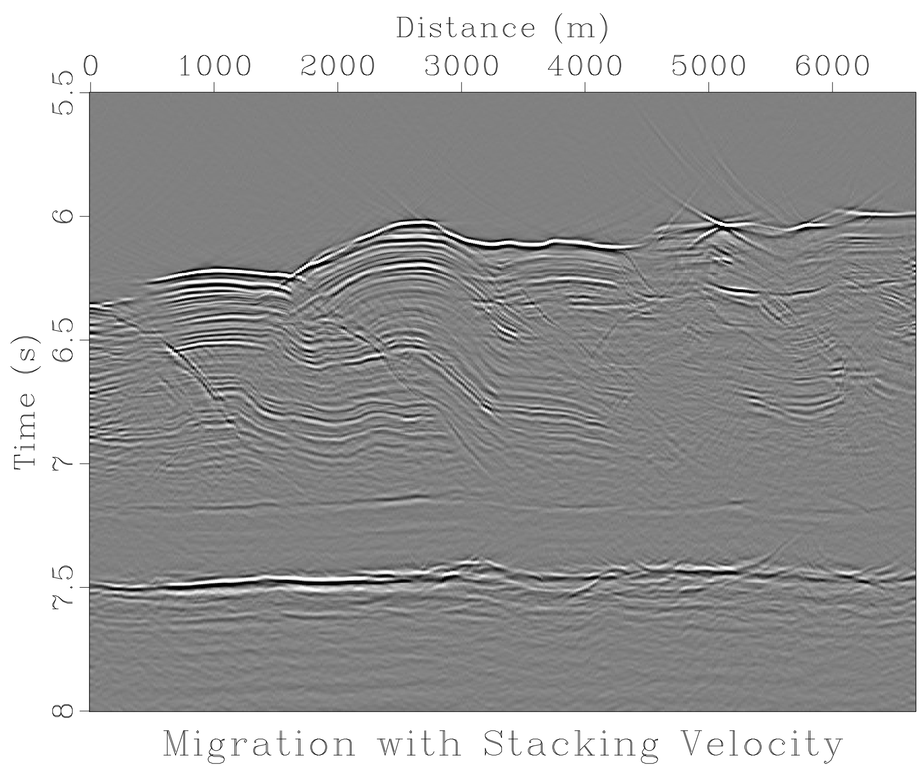


Figure 39: Migration with the stacking velocity. project/ mstack

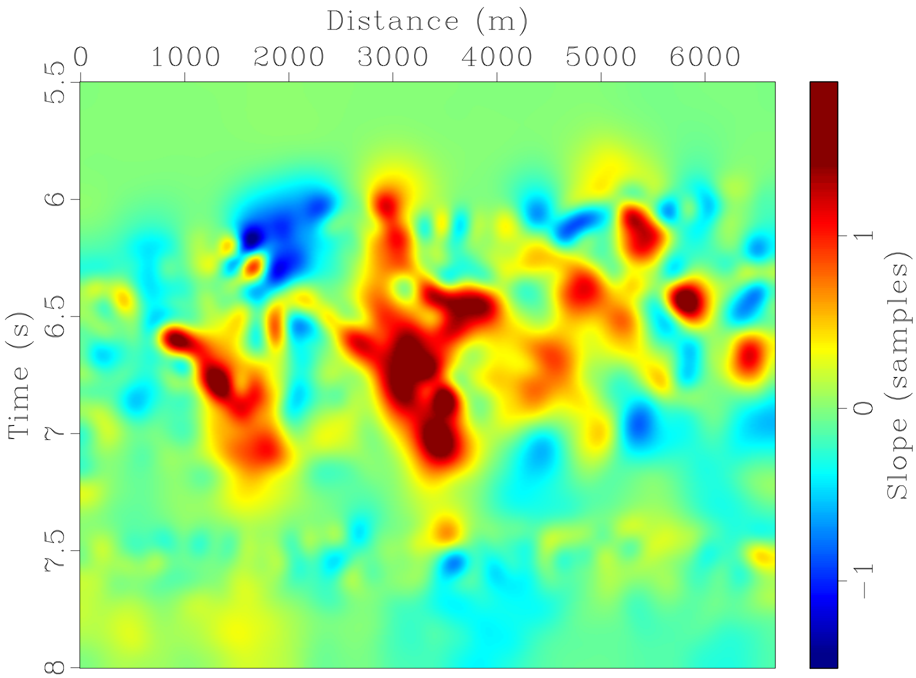


Figure 40: Estimated dips. project/ dip

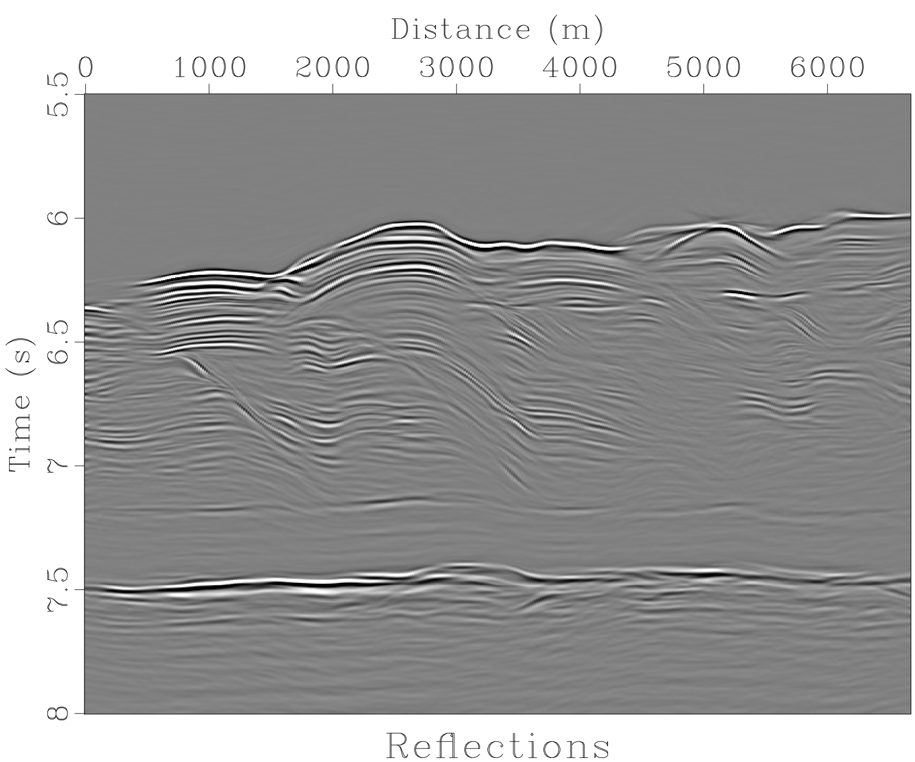


Figure 41: Reflections. project/ refl

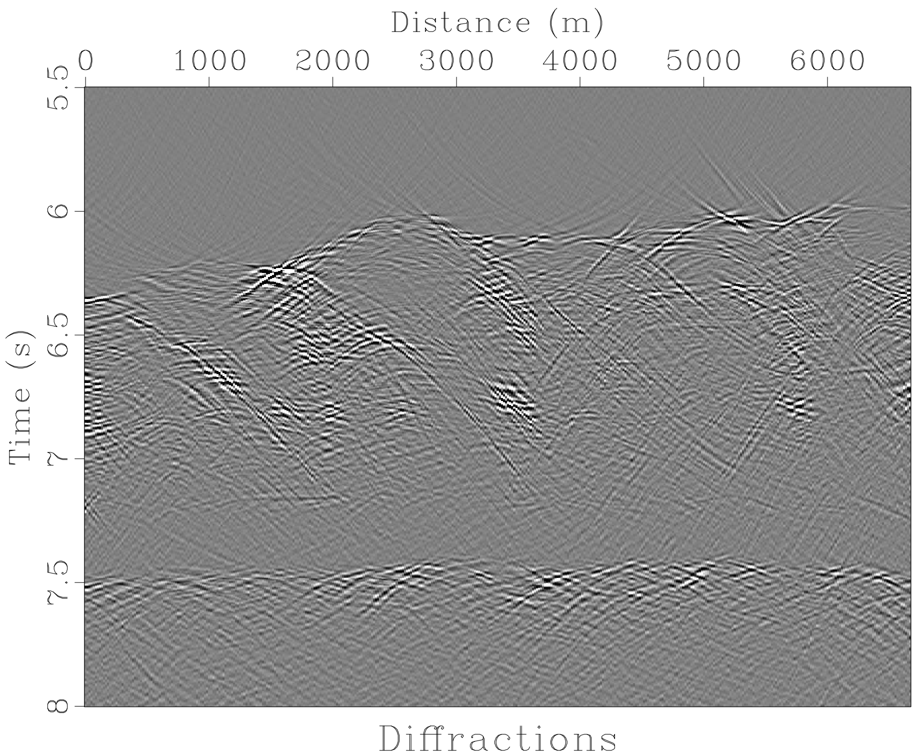


Figure 42: Diffractions. project/ diff

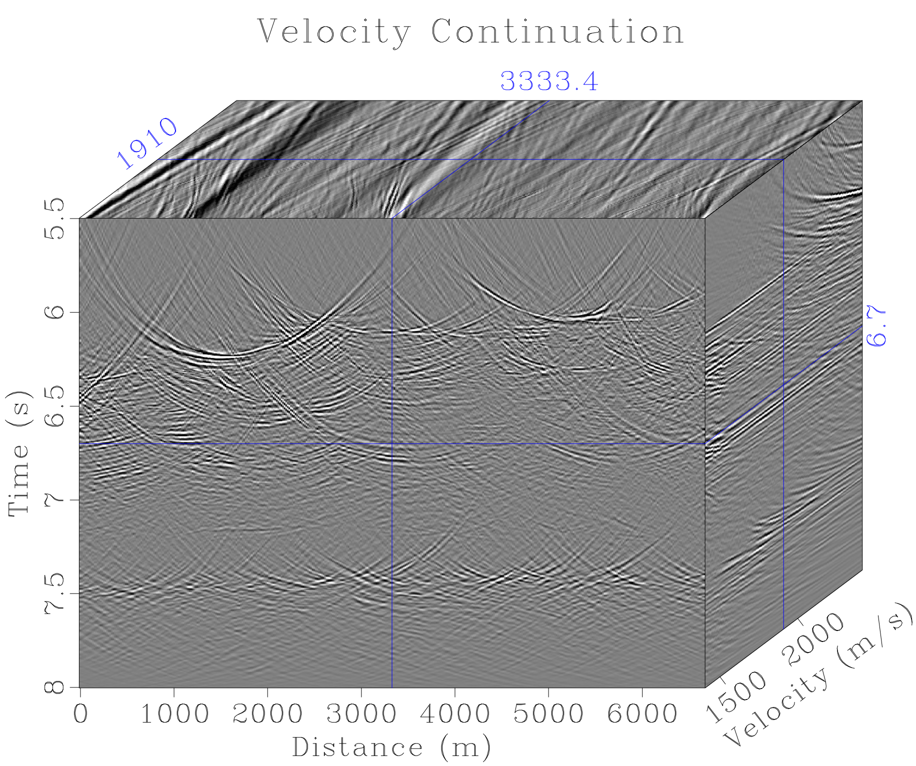


Figure 43: Velocity continuation with diffractions. project/ velcond

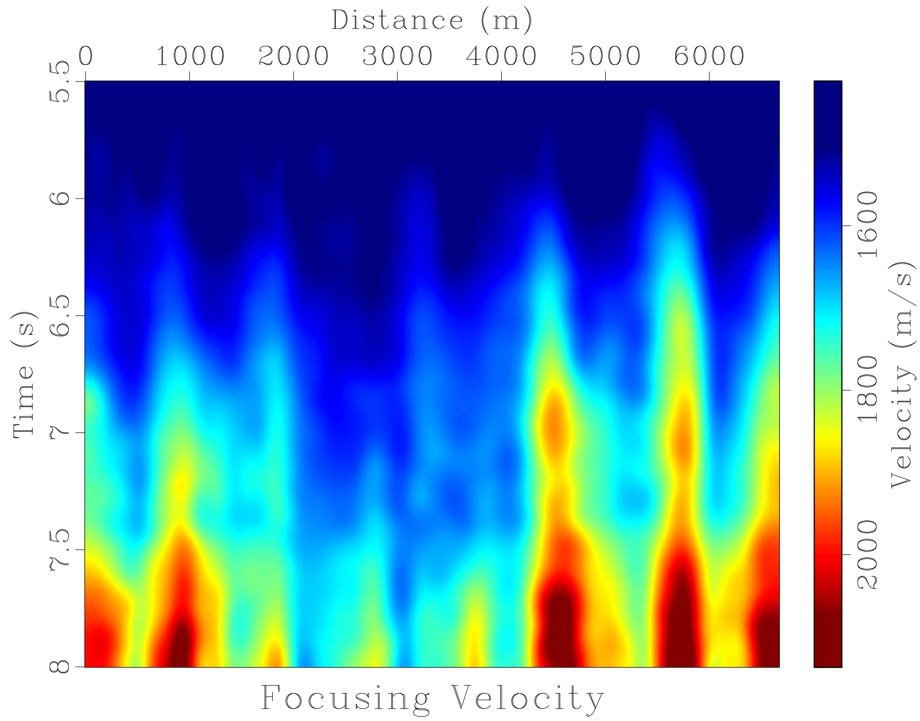


Figure 44: Focusing velocity. project/ fpik

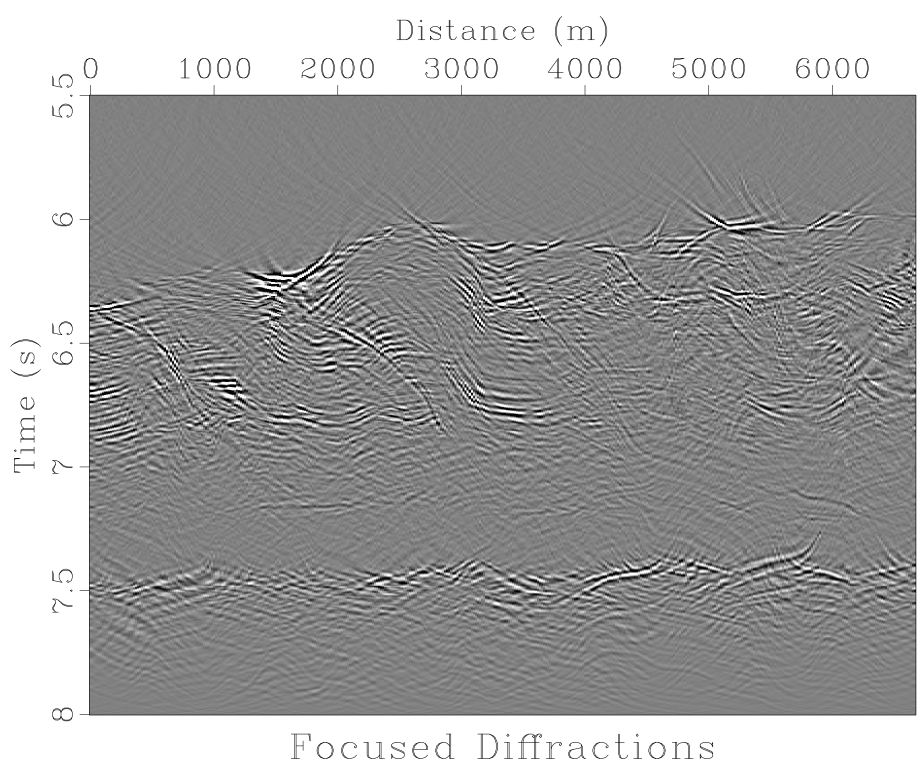


Figure 45: Focused diffractions. project/ mdif

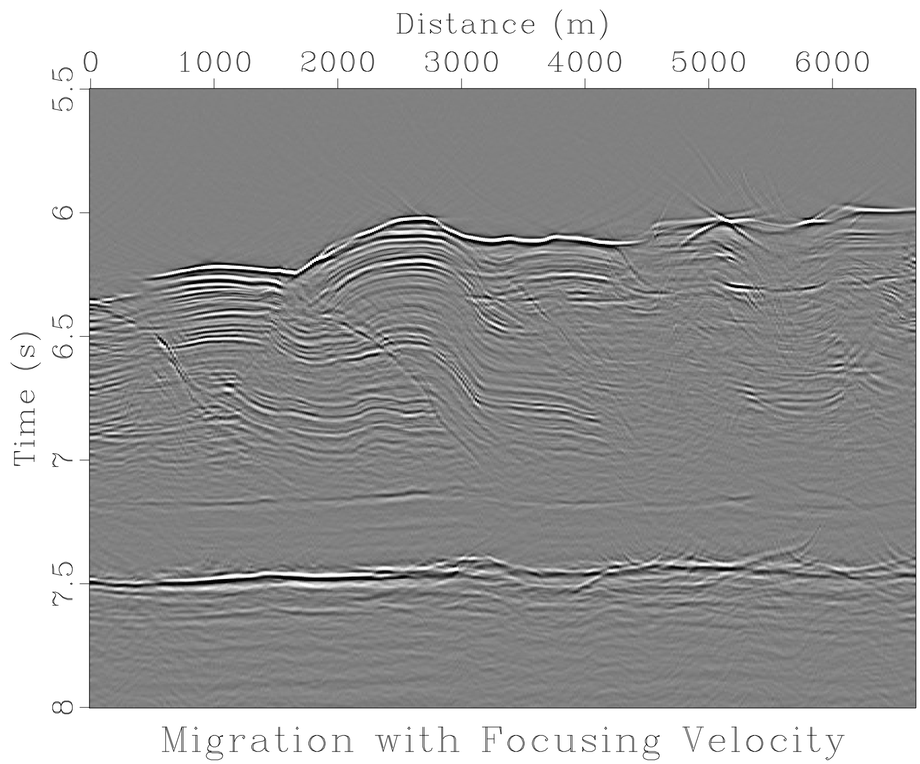


Figure 46: Migration with focusing velocity. project/ mfpik

Reverse-time migration

1. In order to account for laterally variations in velocity, we used reverse-time migration by the lowrank approximation (Fomel et al., 2013). We converted the time-migration velocity to depth for creating an intitial velocity model for depth migration (Figure 47). We used reverse-time migration to migrate the data (Figure 48). Because there are not much laterally variations in velocity so there are few improvements in RTM image (Left figure in Figure 51c, Figure 52c, Figure 53c). Comparing with the published stacked and migrated data (Right figure in Figure 51c, Figure 52c, Figure 53c), the diffractions at ocean bottom are collapsed and the events are clearer and more continuous.

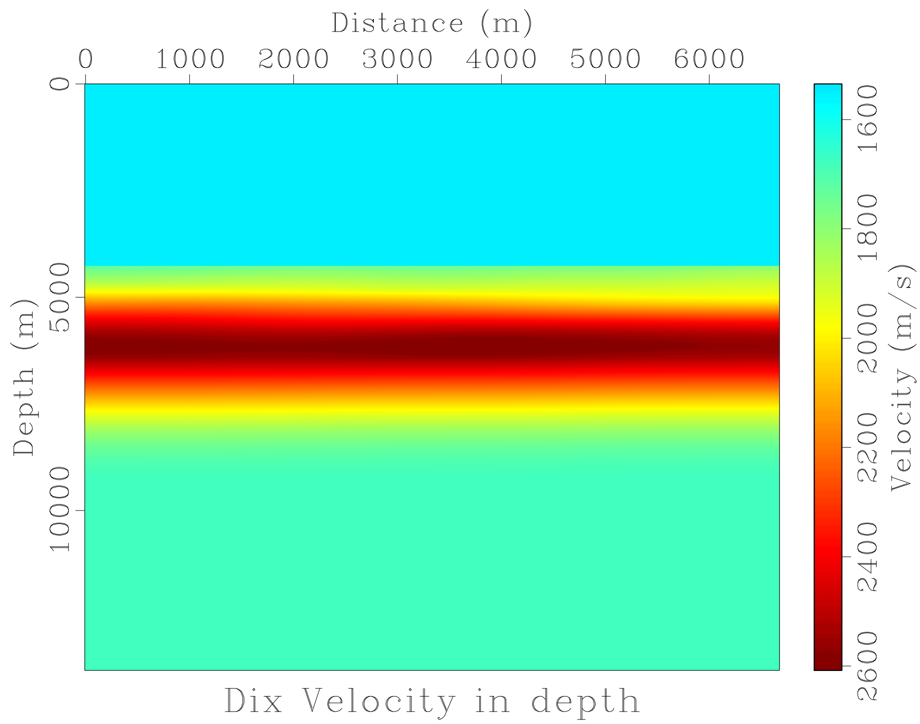


Figure 47: Dix velocity in depth. project/ vdixz

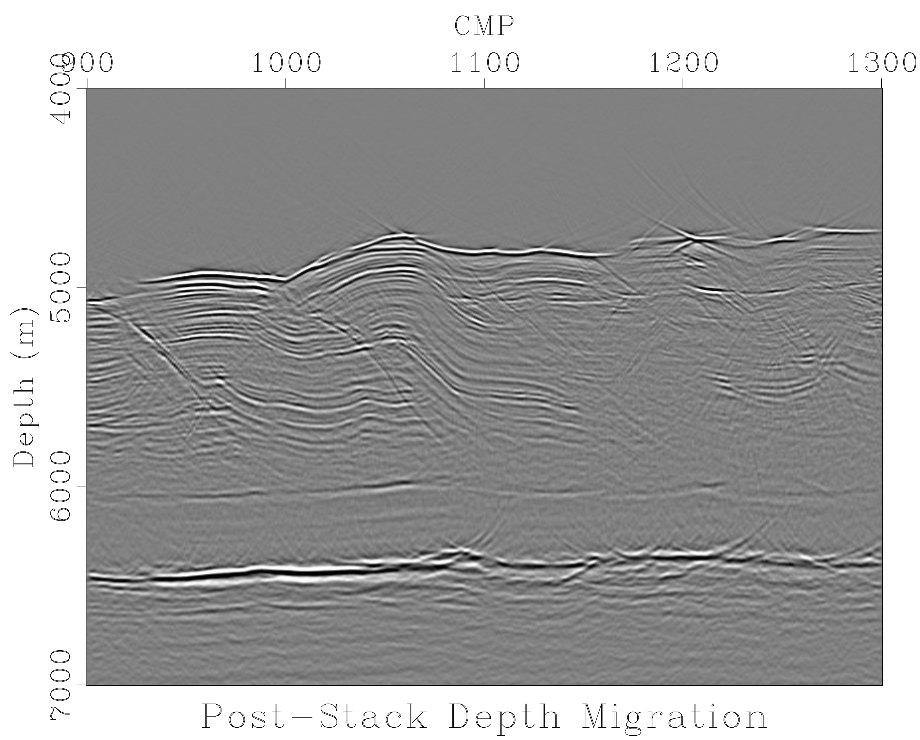


Figure 48: Reverse-time migration. project/ rtm

Split-step migration

1. In order to account for laterally variations in velocity, we used split-step migration (Stoffa et al., 1990). We calculated slowness from the depth-migration velocity (Figure 49). We then used the split-step migration to migrate the data (Figure 50). Because there are not much laterally variations in velocity so there are few improvements in Split-step image (Second left figure in Figure 51c, Figure 52c, Figure 53c). Comparing with the published stacked and migrated data (Right figure in Figure 51c, Figure 52c, Figure 53c), the diffractions at ocean bottom are collapsed and the events are clearer and more continuous.

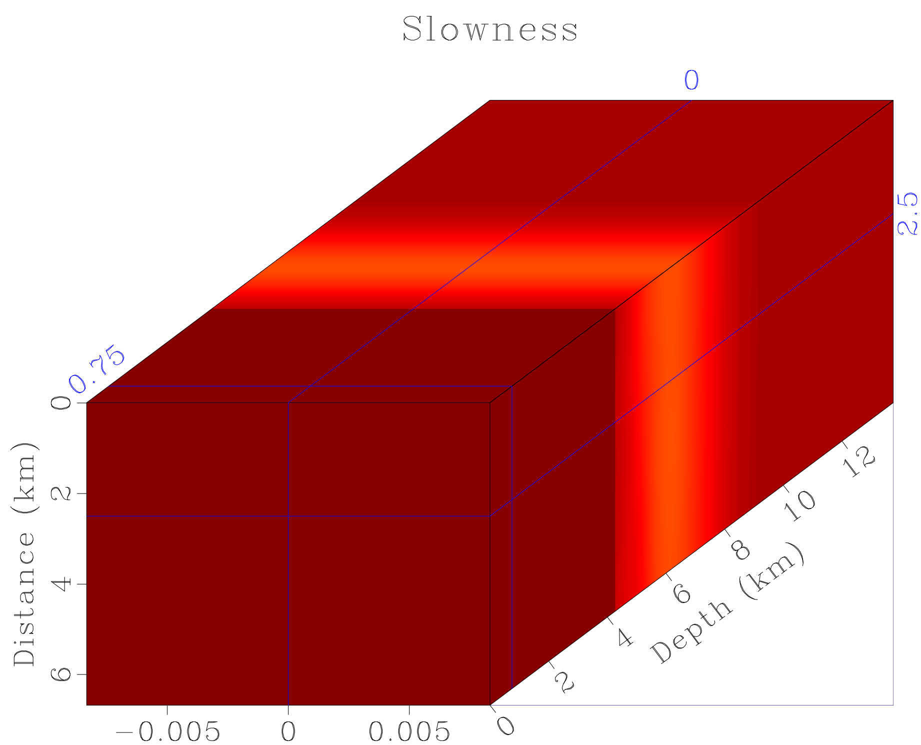


Figure 49: Slowness. project/ slo

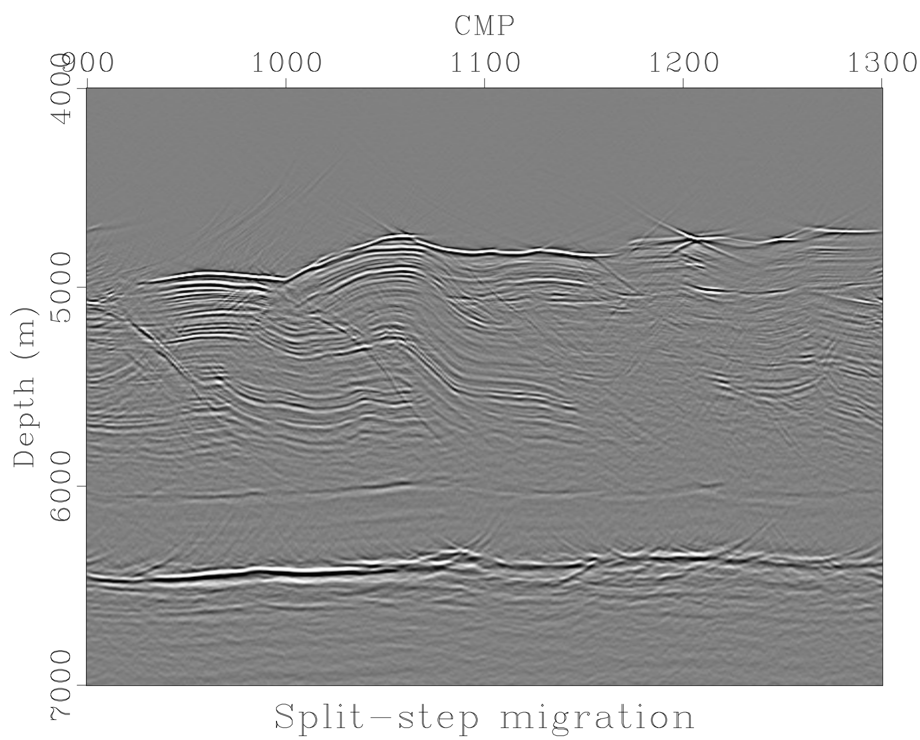


Figure 50: Split-step migration. project/ mig

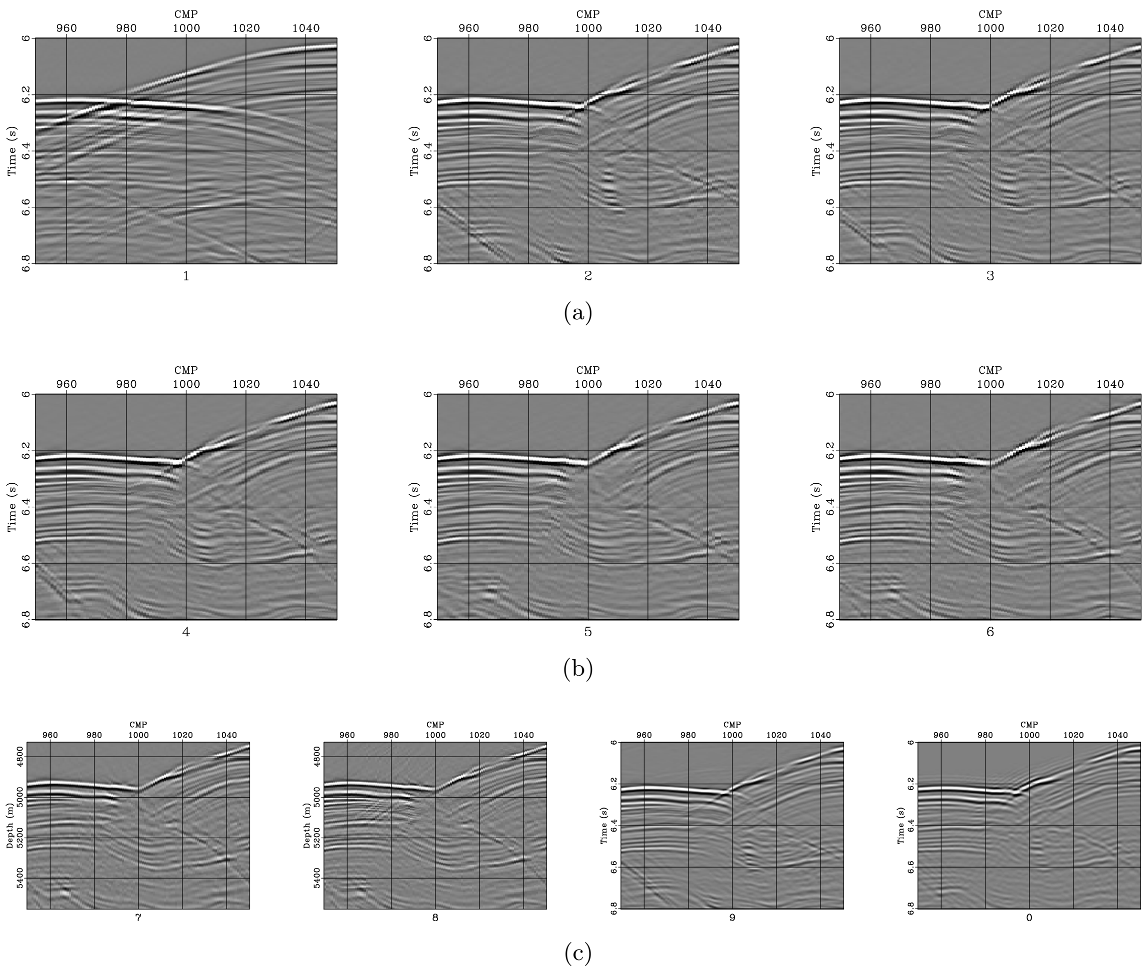
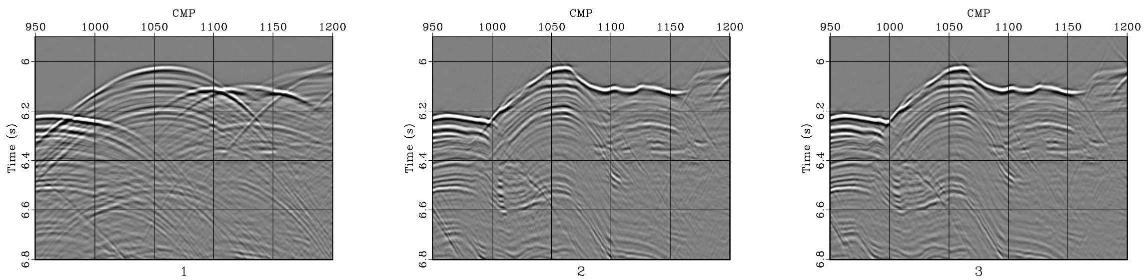
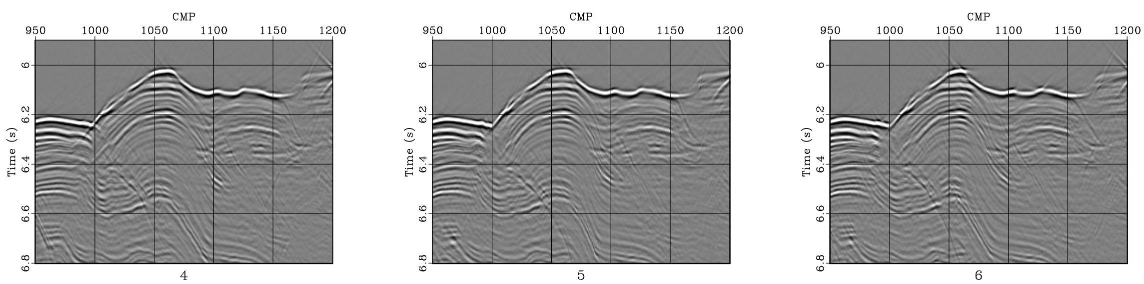


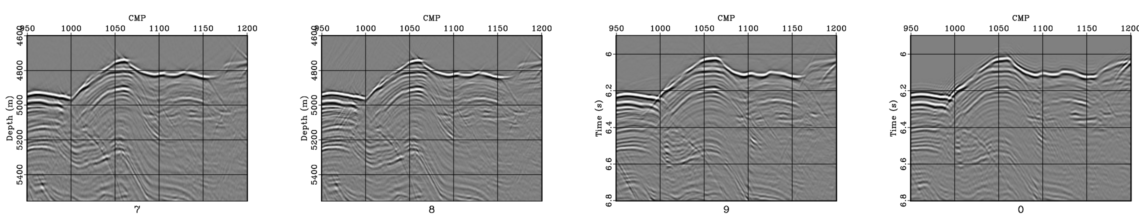
Figure 51: Migration comparison (a1) DMO stack (a2) Stolt migration with constant velocity 1500 m/s (a3) Stolt migration with variable velocity (b4) Migration with stacking velocity (b5) Migration with focusing velocity (b6) Gazdag migration (c7) Reverse-time migration (c8) Split-step migration (c9) Post-stack Kirchhoff time migration (c0) Published stacked and migrated data. project/ zoom1,zoom2,zoom3



(a)

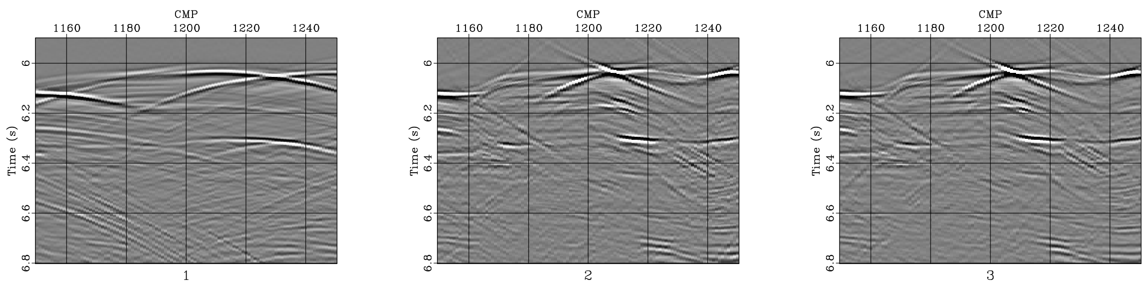


(b)

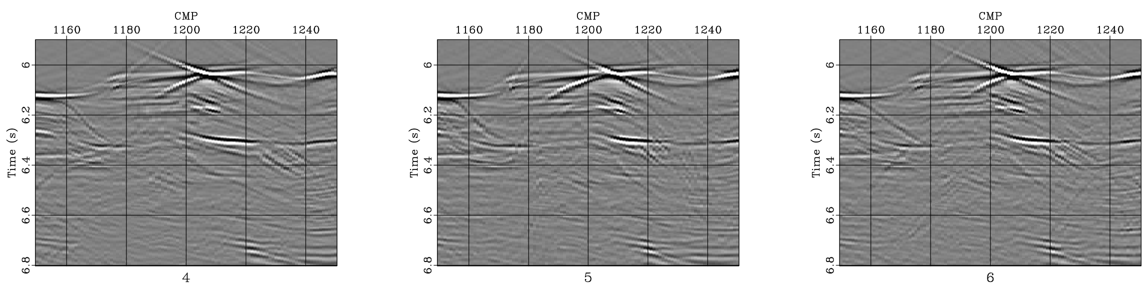


(c)

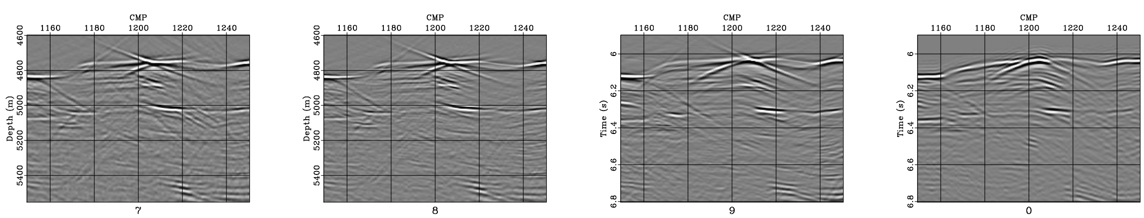
Figure 52: Migration comparison (a1) DMO stack (a2) Stolt migration with constant velocity 1500 m/s (a3) Stolt migration with variable velocity (b4) Migration with stacking velocity (b5) Migration with focusing velocity (b6) Gazdag migration (c7) Reverse-time migration (c8) Split-step migration (c9) Post-stack Kirchhoff time migration (c0) Published stacked and migrated data. project/ zoom4,zoom5,zoom6



(a)



(b)



(c)

Figure 53: Migration comparison (a1) DMO stack (a2) Stolt migration with constant velocity 1500 m/s (a3) Stolt migration with variable velocity (b4) Migration with stacking velocity (b5) Migration with focusing velocity (b6) Gazdag migration (c7) Reverse-time migration (c8) Split-step migration (c9) Post-stack Kirchhoff time migration (c0) Published stacked and migrated data. project/ zoom7,zoom8,zoom9

INTERPRETATIONS

1. Our processing result has improvements on imaging the diffractions. Based on the final processing output, we can have some geologic interpretations.

2. Oceanic crust of the Pacific Plate lies under sediments at the left side of Figure 4, and is subducted at the trench, which is filled with recent sediments (CDPs 200-600, approximately). On the right of these flat-lying sediments, there is an accretionary prism with distorted sediment layer thickens to the right, which results in interesting topography.

3. The reflector, which is about 300 ms later, mimic the ocean-bottom topography and cuts across the folded sediments, is called the bottom-simulating reflector (Forel et al., 2005). It reflects a change in fluid phases.

4. The Shikoku Basin sequence in our 2D seismic line has two units. A lower 400-m-thick transparent unit lies directly on oceanic basement and is overlain by a 500-m-thick layered unit. The boundary of the protothrust zone is marked by the first major thrust fault which uplifts trench sediments in a ramp anticline. Thrust ramps are developed farther landward, forming a series of parallel linear structural terraces.

5. The decollement is within the Shikoku Basin sedimentray section. The decollement is the Pacific Plate oceanic crust at about 7.2 seconds. It appears to be flatten. The thrusts are imaged as a fault plane reflection and has an overlying fault-bend fold.

project/SConstruct

```
from rsf.proj import *


def section(title,label1='Time',unit1='s',min1=5.8,max1=8.0,extra=''):
    return '''
    window min1=%g max1=%g |
    grey title="%s"
    label1="%s" unit1="%s" label2=Distance unit2=m %s
    ''' % (min1,max1,title,label1,unit1,extra)

# Download data

Fetch('Nshots.su','nankai')

# Convert from SU to RSF

Flow('shots tshots','Nshots.su',
         '''
         suread suxdr=y tfile=${TARGETS[1]}
         ''')

Result('spectra','shots',
        '''
```

```
         spectra all=y | scale axis=1 |
         graph title="Average Spectra" label1=Amplitude
         ''')
Result('shots',
         '''
         window min1=5.5 max1=8.5 |
         grey title="Raw shot gathers"
         ''')

Fetch('Nstack.su','nankai')

Flow('stackd tstackd','Nstack.su',
      '''
      suread suxdr=y tfile=${TARGETS[1]}
      ''')
Flow('stack-win','stackd','window min1=5')
Flow('stack-shot','stack-win',
      '''
      window min2=900 max2=1300 | put label2=CMP
      ''')
Plot('stackd','window j1=2 j2=2 | grey title=Stack label2=CMP')
Plot('stack-win',
      '''
      window j2=2 | grey title="Stack (Window)" label2=CMP
      ''')
Plot('stack-shot','stack-win',
      '''
      window min2=900 max2=1300 |
      grey title="Part of stack corresponding to the shot file"
      label2=CMP
      ''')
Result('stack-shot','stack-win',
         '''
         window min2=900 max2=1300 |
         grey title="Part of stack corresponding to the shot file"
         label2=CMP
         ''')
Result('stackd','stackd stack-win','SideBySideAniso')

####### DC Removal

Flow('mean','shots',
      '''
      stack axis=1 | spray axis=1 n=5500 o=0.0 d=0.002
      ''')

Flow('shotsdc','shots mean','add scale=1,-1 ${SOURCES[1]}')
Result('shotsdc',
```

```
        '''
        window min1=5.5 max1=8.5 |
        grey title="Shot gathers after removing DC offset"
        ''')

####### Bandpass Filtering

Flow('shotsf','shotsdc','bandpass flo=10 fhi=125')
Result('shotsf',
        '''
        window min1=5.5 max1=8.5 |
        grey title="Shot gathers after filtering frequencies"
        ''')
Result('spectraf','shotsf',
        '''
        spectra all=y |  scale axis=1 |
        graph title="Spectra after removing low and high frequencies"
        label1=Amplitude
        ''')

####### Mask zero traces

Flow('mask0','shotsf','mul $SOURCE | stack axis=1 | mask min=1e-20')
Flow('shots0','shotsf mask0','headerwindow mask=${SOURCES[1]}')

# update a database
Flow('tshots0','tshots mask0','headerwindow mask=${SOURCES[1]}')

####### Surface consistent

# Average trace amplitude
Flow('arms','shots0',
     'mul $SOURCE | stack axis=1 | math output="log(input)"')
Result('arms','grey title=Log-Amplitude mean=y pclip=90')

# shot/offset indeces: fldr and tracf

Flow('indexshot','tshots0','window n1=1 f1=2')

Flow('offsets4index','tshots0',
     '''
     headermath output=offset | dd type=float | window
     ''')

Flow('offsetindex','offsets4index',
     '''
     math output="abs(input) - 170" | dd type=int
     ''')
```

```

# receiver/midpoint

Flow('midpoint','tshots0','window n1=1 f1=5')

Flow('cmps4index','tshots0',
     '''
     headermath output=cdp | dd type=float |
     math output="input*16.667" | window
     ''')

Flow('recv','cmps4index offsets4index',
     '''
     add scale=1,0.5 ${SOURCES[1]} |
     math output="input - 13799" | dd type=int
     ''')

Flow('index','indexshot offsetindex',
     '''
     cat axis=2 ${SOURCES[1]}
     ''')

Flow('extindex','index midpoint',
     '''
     cat axis=2 ${SOURCES[1]}
     ''')

Flow('extindrecv','extindex recv',
     '''
     cat axis=2 ${SOURCES[1]}
     ''')

def plot(title):
    return '''
    spray axis=1 n=1 |
    intbin head=${SOURCES[1]} yk=fldr xk=tracf | window |
    grey title="%s" label2="Shot Number" unit2=
    label1="Offset Number" unit1= scalebar=y
    ''' % (title)

def plotb(title,bias=-5):
    return '''
    spray axis=1 n=1 |
    intbin head=${SOURCES[1]} yk=fldr xk=tracf | window |
    grey title="%s" label2="Shot Number" unit2=
    label1="Offset Number" unit1= scalebar=y clip=3 bias=%g
    ''' % (title,bias)

```

```
# Display in shot/offset coordinates
Flow('varms','arms tshots0',
      '''
      spray axis=1 n=1 |
      intbin head=${SOURCES[1]} yk=fldr xk=tracf | window
      ''')
Result('varms','arms tshots0',plotb('Log-Amplitude'))

prog = Program('surface-consistent.c')
sc = str(prog[0])

# recv index

# get model dimensions
Flow('recvmodel',['arms','extindrecv',sc],
      './${SOURCES[2]} index=${SOURCES[1]} verb=y')

# find a term
Flow('recvsc',['arms','extindrecv',sc,'recvmodel'],
      '''
      conjgrad ./${SOURCES[2]} index=${SOURCES[1]}
      mod=${SOURCES[3]} niter=150
      ''')

# project to a data space
Flow('recvscarms',['recvsc','extindrecv',sc],
      './${SOURCES[2]} index=${SOURCES[1]} adj=n')

Result('recvvscarms','recvscarms tshots0',
        plotb('Source, Offset, CDP, Recv S-C Log(A)'))

# compute difference
Flow('recvadiff','arms recvscarms','add scale=1,-1 ${SOURCES[1]}')

Result('recvadiff','recvadiff tshots0',plot('s,h,cdp,r difference'))

size=dict(sht=326,off=2266,rcv=7784,cmp=401)
f1 = 0
for case in ('sht','off','rcv','cmp'):
    n1=size[case]

    Result(case,'recvsc',
           '''
           window n1=%d f1=%d | put o1=1 d1=1 |
           graph title="%s Term"
           label1="%s Number" unit1= label2=Amplitude unit2=
           ''' % (n1,f1,case.capitalize(),case.capitalize()))
    f1 += n1
```

```

### apply to traces to all times

Flow('ampl','recvscarms',
     'math output="exp(-input/2)" | spray axis=1 n=5500 d=0.002 o=0')

Flow('shots-preproc','shots0 ampl','mul ${SOURCES[1]}')

Plot('shots-preproc','shots-preproc',
     '''
     window n2=100 |
     grey min1=6.0 max1=8.0 title="Shots Preproc"
     ''')

Plot('shots-raw','shots0',
     '''
     window n2=100 |
     grey min1=6.0 max1=8.0 title="Shots Raw"
     ''')

Result('shotsfc','shots-raw shots-preproc','SideBySideAniso')

# Resample to 4 ms

Flow('subsamples','shots-preproc',
     '''
     bandpass fhi=125 | window j1=2
     ''')

Result('spectrasub','subsamples',
       'spectra all=y | graph title="Subsampled Spectra"')

Result('spectra-check','shots-preproc',
       'spectra all=y | graph max1=160 title="Spectra Check"')

# Extract shots
Flow('shots2','subsamples tshots0',
        '''
        intbin xk=tracf yk=fldr head=${SOURCES[1]}
        ''')
Result('shots2',
       '''
       window min1=5.5 max1=9 | byte gainpanel=all |
       transp plane=23 memsize=5000 |
       grey3 frame1=300 frame2=100 frame3=50
       point1=0.8 point2=0.8 clip=8.66
       title="Shots" label3="Shots"
       ''')
```

```


# Create a mask to remove misfired shots

Flow('smask','shots2','mul $SOURCE | stack axis=1 | mask min=1e-20')

Result('smask',
        '''
        dd type=float |
        stack axis=1 norm=n |
        graph symbol=x title="Nankai Trough"
        label2="Traces per Shot Gather"
        label1="Shot Number" unit1=
        ''')

Flow('offsets','tshots0',
      '''
      window n1=1 f1=11 squeeze=n | dd type=float |
      intbin head=$SOURCE xk=tracf yk=fldr
      ''')


# Select one shot (fldr)

shot=1707

Flow('mask','smask','window n2=1 min2=%g' % shot)

Flow('shot','shots2 mask',
      '''
      window n3=1 min3=%g squeeze=n |
      headerwindow mask=${SOURCES[1]}
      ''' % shot)

Flow('offset','offsets mask',
      '''
      window n3=1 min3=%g squeeze=n |
      headerwindow mask=${SOURCES[1]}
      ''' % shot)

Result('shot','shot offset',
        '''
        window min1=5.8 max1=8.5 |
        wiggle xpos=${SOURCES[1]} yreverse=y transp=y poly=y
        title="Shot 1707" label2=Offset
        ''')

Plot('shot',
```

```
    '''
    window min1=5.8 max1=8.5 |
    grey title="Selected Shot" clip=2
    ''')


# Extract CMPs and apply t^2 gain

Flow('cmps maskcmp','subsamples tshots0',
    '''
     intbin head=${SOURCES[1]} mask=${TARGETS[1]}
     xk=tracf yk=cdp           |
     pow pow1=2
     ''')

Result('cmps',
       '''
       window min1=5.8 max1=8.5 |
       byte gainpanel=all | transp plane=23 memsize=5000 |
       grey3 frame1=200 frame2=150 frame3=50
       title="CMPs" label2="CMP Number"
       flat=n point1=0.8 point2=0.8
       ''')

Flow('cmask','cmps','mul $SOURCE | stack axis=1 | mask min=1e-20')

Result('cmask',
      '''
      dd type=float |
      stack axis=1 norm=n |
      graph symbol=x title="Nankai Trough"
      label2="Traces per CMP Gather"
      label1="CMP Gather Number" unit1=
      ''')

Flow('offs','tshots0',
    '''
    window n1=1 f1=11 squeeze=n | dd type=float |
    intbin xk=tracf yk=cdp head=$SOURCE
    ''')

# Examine one CMP gather

Flow('mask1','cmask','window n2=1 min2=1280')

Flow('cmp1','cmps mask1',
    '''
    window n3=1 min3=1280 | headerwindow mask=${SOURCES[1]}
```

```
       ''')

Flow('off','offs mask1',
     '''
     window n3=1 min3=1280 squeeze=n | headerwindow mask=${SOURCES[1]}
     ''')

Result('cmp1','cmp1 off',
       '''
       window min1=5.8 max1=8.5 |
       wiggle xpos=${SOURCES[1]} title="CMP 1280"
       yreverse=y transp=y poly=y label2=Offset unit2=m
       wherexlabel=t wheretitle=b
       ''')

Plot('cmp1','cmp1 off',
     '''
     window min1=5.8 max1=8.5 |
     wiggle xpos=${SOURCES[1]} title="CMP 1280"
     yreverse=y transp=y poly=y label2=Offset unit2=m
     wherexlabel=t wheretitle=b
     ''')

# Velocity analysis and NMO

Flow('vscan','cmp1 off mask1',
     '''
     vscan half=n offset=${SOURCES[1]}
     v0=1400 nv=101 dv=10 semblance=y
     ''')

Plot('vscan',
     '''
     window min1=5.8 max1=8.5 |
     grey color=j allpos=y title="Velocity Scan" unit2=m/s
     ''')

Flow('pick','vscan',
     '''
     mutter inner=y half=n t0=5 x0=1400 v0=75 |
     pick v0=1500 rect1=25
     ''')

Plot('pick',
     '''
     window min1=5.8 max1=8.5 |
     graph transp=y yreverse=y plotcol=7 plotfat=3
     pad=n min2=1400 max2=2400 wanttitle=n wantaxis=n
```

```
        ''')

Plot('vscanp','vscan pick','Overlay')

Flow('nmo','cmp1 off mask1 pick',
        '''
        nmo half=n offset=${SOURCES[1]}
        velocity=${SOURCES[3]}
        ''')

Result('nmo','window min1=5.8 max1=8.5 | grey title="Normal Moveout"')

Plot('cmpg','cmp1',
        '''
        window min1=5.8 max1=8.5 |
        grey title="CMP 1280" labelsz=12 titlesz=18
        ''')

Plot('nmog','nmo',
        '''
        window min1=5.8 max1=8.5 |
        grey title="Normal Moveout" labelsz=12 titlesz=18
        ''')

Result('nmo1','cmpg nmog','SideBySideAniso')

Result('vscan','cmp1 vscanp','SideBySideAniso')

# Apply to all CMPs

Flow('cmask2','cmask','transp plane=23 | transp plane=21')

Flow('vscans','cmps offs cmask2',
        '''
        vscan half=n offset=${SOURCES[1]} mask=${SOURCES[2]}
        v0=1400 nv=101 dv=10 semblance=y nb=5
        ''',split=[3,'omp'])

Flow('picks','vscans',
        '''
        mutter inner=y half=n t0=5 x0=1400 v0=75 |
        pick v0=1500 rect1=25 rect2=10
        ''')

Result('picks',
          '''
          window | grey color=j mean=y scalebar=y title="NMO Velocity"
          label2=CMP barreverse=y barlabel=Velocity barunit=m/s
```

```
         ''')

Flow('nmos','cmps offs cmask picks',
      '''
      nmo half=n offset=${SOURCES[1]} mask=${SOURCES[2]}
      velocity=${SOURCES[3]}
      ''')

Result('nmos',
         '''
         window min1=5.5 max1=8.5 |
              byte gainpanel=all | transp plane=23 memsize=5000 |
         grey3 frame1=400 frame2=50 frame3=50
              title="NMO" label2="CMP Number"
         flat=n point1=0.8 point2=0.8
              ''')

Flow('stack','nmos','stack')

Result('stack',
        '''
        window min1=5.5 max1=8.5 |
        grey title="NMO Stack" label2="CMP Number"
        ''')

Plot('stack',
      '''
      window min1=5.5 max1=8.5 |
      grey title="NMO Stack" label2="CMP Number"
      ''')

# Try DMO

nv=60

Flow('stacks','cmps offs maskcmp',
      '''
      stacks half=n v0=1400 nv=%g dv=20
      offset=${SOURCES[1]} mask=${SOURCES[2]}
      '''%nv, split=[3,'omp'])

Flow('stackst','stacks','costaper nw3=20')

Result('stacks','stackst',
        '''
        window min1=5.5 max1=8.5 |byte gainpanel=all |
        transp plane=23 memsize=5000 |
        grey3 frame1=200 frame2=100 frame3=50 point1=0.8 point2=0.8
```

```
        title="Constant-Velocity Stacks" label3=Velocity unit3=m/s
        label2="CMP Number"
        ''')

# Apply double Fourier transform (cosine transform)

Flow('cosft3','stackst',
      '''
      put d3=16.667 o3=0 label2=Distance unit2=m |
      cosft sign1=1 sign3=1
      ''')

# Transpose f-v-k to v-f-k

Flow('transp','cosft3','transp')

# Fowler DMO: mapping velocities

Flow('map','transp',
      '''
      math output="x1/sqrt(1+0.25*x3*x3*x1*x1/(x2*x2))" |
      cut n2=1
      ''')
Result('map',
        '''
        window j3=4 |
        byte gainpanel=all allpos=y bar=bar.rsf |
        grey3 title="Fowler Map" label1=Velocity
        unit1=m/s label3=Wavenumber barlabel=Velocity barunit=m/s
        frame1=20 frame2=500 frame3=20 color=x scalebar=y
        ''')

Flow('fowler','transp map','iwarp warp=${SOURCES[1]} | transp')

# Inverse Fourier transform

Flow('dmo','fowler','cosft sign1=-1 sign3=-1')

Result('dmo',
        '''
        put d3=1 o3=900 unit2='' label2=Trace | window min1=5.5 max1=8.5 |
        byte gainpanel=all | transp plane=23 memsize=5000 |
        grey3 frame1=200 frame2=100 frame3=50 point1=0.8 point2=0.8
        title="Constant-Velocity DMO Stacks"
        label3=Velocity unit3=m/s
        ''')

# Compute envelope for picking
```

```

Flow('envelope','dmo','envelope | scale axis=2')

# Mute and Pick velocity

Flow('vpick','envelope',
     '''
        mutter v0=130 x0=1300 t0=4.0 half=n inner=n |
        mutter x0=1400 v0=20 t0=5.0 half=n inner=y |
        mutter v0=2500 x0=1400 t0=5.8 half=n inner=n |
        mutter v0=500 x0=1400 t0=7.0 half=n inner=y  |
        pick rect1=80 rect2=20 vel0=1400
        ''')

Result('vpick',
       '''
       put d2=1 o2=900 unit2= label2="CMP Number" |
       grey mean=y color=j scalebar=y
       barreverse=y barunit=m/s barlabel=Velocity
       title="Picked Migration Velocity" label2="CMP Number" unit2=
       ''')

# Take a slice

Flow('slice','dmo vpick',
     '''
     slice pick=${SOURCES[1]} |
     put d2=1 o2=900 unit2= label2="CMP Number"
     ''')

Result('slice',
       '''
       window min1=5.5 max1=8.5 |
       grey title="DMO Stack"
       ''')

Plot('slice',
     '''
     window min1=5.5 max1=8.5 |
     grey title="DMO Stack" label2="CMP Number"
     ''')

# Check one CMP location

p = 380

min1=5.8
max1=8.5
```

```

Flow('before','stackst',
     '''
     window n3=1 f3=%d min1=%g max1=%g | envelope
     ''' % (p,min1,max1))
Flow('after','envelope',
     '''window n3=1 f3=%d min1=%g max1=%g
     ''' % (p,min1,max1))

for case in ('before','after'):
    Plot(case,
         '''
         grey color=j allpos=y title="%s DMO"
         label2=Velocity unit2=m/s
         ''' % case.capitalize())

Flow('vpick1','vpick',
     '''
     window n2=1 f2=%d min1=%g max1=%g
     '''%(p,min1,max1))

Plot('vpick1',
     '''
      graph yreverse=y transp=y plotcol=7 plotfat=7
      pad=n min2=%g max2=%g wantaxis=n wanttitle=n
     ''' % (1400,2600))

Plot('before2','before pick','Overlay')
Plot('after2','after vpick1','Overlay')
Result('envelope','before2 after2','SideBySideAniso')

##################################################
# Zoom an interesting area
##################################################
min1,max1=5.8,7
min2,max2=1100,1300

Flow('box.asc',None,
     '''
     echo %s n1=2 n2=5 data_format=ascii_float in=$TARGET
     ''' % ' '.join(map(str,(min1,min2,max1,min2,
                              max1,max2,min1,max2,min1,min2))))
Plot('box','box.asc',
     '''
     dd form=native type=complex | window |
     graph transp=y yreverse=y min1=0 max1=4 min2=0 max2=26.7625
     wanttitle=n plotfat=5 plotcol=6 wantaxis=n
     ''')
```

```

for i in range (2):
    case=('slice','stack')[i]
    zoomd = case + '-zoomd'
    Flow(zoomd,case,
         '''
         window min1=%g max1=%g min2=%g max2=%g
         ''' % (min1,max1,min2,max2))
    Plot(zoomd,'grey title=%s grid=y gridcol=5 label2=CMP' % ('ab'[i]))
Result('zoomd','slice-zoomd stack-zoomd',
'SideBySideIso')

# Stolt Migration

# 2D cosine transform

Flow('cosft','slice',
     '''
     put d2=16.667 label2=Distance unit2=m |
     cosft sign1=1 sign2=1 |
     put label2=Wavenumber
     ''')
Result('cosft',
       '''
       window max1=90 |
       grey title="Cosine Transform" pclip=95 label1=Frequency
       ''')

# Stolt migration with constant velocity

Flow('map2','cosft',
     '''
     math output="sqrt(x1*x1+%g*x2*x2)"
     '''%(562500))

Result('map2',
       '''
       grey color=x allpos=y scalebar=y
       title="Stolt Map" barlabel=Frequency barunit=Hz label1=Frequency
       ''')

Flow('cosft2','cosft map2','iwarp warp=${SOURCES[1]} inv=n')

Result('cosft2',
       '''
       window max1=90 |
       grey title="Cosine Transform Warped" pclip=95 label1=Frequency
       ''')
```

```

Flow('mig2','cosft2',
     '''
     cosft sign1=-1 sign2=-1 |

     put d2=1 unit2=''
     ''')

Result('mig2',
       '''
       window min1=5.5 max1=8.5 |
       grey title="Stolt Migration with 1500 m/s"
       label2="CMP Number"
       ''')

# Ensemble of Stolt migrations with different velocities

Flow('spray','cosft',
     'spray axis=3 n=120 o=1500 d=8 label=Velocity unit=m/s')

Flow('map1','spray','math output="sqrt(x1*x1+0.25*x3*x3*x2*x2)" ')

Result('map1',
       '''
       byte gainpanel=all allpos=y |
       grey3 title="Stolt Ensemble Map"
       frame1=400 frame2=50 frame3=50 color=x
       ''')

Flow('migd','spray map1',
     '''
     iwarp warp=${SOURCES[1]} inv=n |
     cosft sign1=-1 sign2=-1
     ''')

Flow('migt','migd','transp plane=23 memsize=5000')

Plot('migd',
     '''
     window min1=5.5 max1=8.5 |
     grey title="Ensemble of Stolt Migrations"
     ''',view=1)

# Migration velocity increasing with time

Flow('vmig1','slice',
     '''
     pad beg1=1250 | math output="1500" |
```

```
        window n1=1250 | put o1=0
        ''')

Flow('vmig2','slice',
        '''
        window min1=5 | put o1=0 |
        math output="1500+20*x1*x1" | put o1=5
        ''')

Flow('vmig','vmig1 vmig2','cat ${SOURCES[1:2]} axis=1')

Result('vmig',
           '''
           window min1=5.5 max1=8.5 |
           grey color=j mean=y barreverse=y title="Migration Velocity"
           scalebar=y barlabel=Velocity barunit=m/s label2="CMP Number"
           unit2=
           ''')
# Slice through the ensemble of migrations

Flow('slice1','migt vmig',
        '''
        slice pick=${SOURCES[1]} | put d2=1 o2=900
        ''')

Result('migd','slice1',
           '''
           window min1=5.5 max1=8.5 |
           grey label2="CMP Number"
           unit2= title="Stolt Migration with Variable Velocity"
           ''')

# Dix conversion to interval velocity
Flow('semb','vscans picks','slice pick=${SOURCES[1]}')
Flow('vdix','picks semb',
        '''
        dix weight=${SOURCES[1]} rect1=50 rect2=10
        ''')

# Velocity continuation
Flow('first','slice',
        '''
        put o2=0 d2=16.667 label2=Distance unit2=m o3=0 |
        cosft sign2=1 |
        stolt vel=1400
        ''')

Flow('mig0','first',
```

```
        '''
        window min1=5.5 max1=8.0 |
        cosft sign2=-1
        ''')

Result('first','mig0',
          '''
          grey title="Migration with 1400 m/s"
          ''')

Flow('velcon','first',
        '''
        vczo nv=101 dv=10 v0=1400 verb=y |
        window min1=5.5 max1=8.0 |
        transp plane=23 memsize=1000 |
        cosft sign2=-1
        ''')
Result('velcon',
          '''
          byte gainpanel=a |
          grey3 frame1=300 frame2=200 frame3=50
          point1=0.8 point2=0.8 flat=n unit3=m/s
          title="Velocity Continuation"
          ''')
Plot('movie','velcon','grey title="velocity continuation"',view=1)

# Migration with the stacking velocity
Flow('mpick','picks','window min1=5.5 max1=8.0')
Flow('mstack','velcon mpick',
        '''
        transp plane=23 memsize=1000 |
        slice pick=${SOURCES[1]}
        ''')
Result('mstack',
          'grey title="Migration with Stacking Velocity"')

# Separating diffractions

Flow('dip','mig0','dip rect1=20 rect2=10 order=2')
Result('dip',
          '''
          grey color=j scalebar=y wanttitle=n
          barlabel=Slope barunit=samples
          ''')

Flow('refl','mig0 dip',
        '''
        pwspray dip=${SOURCES[1]} ns=5 order=2 reduce=stack
```

```
        ''')
Result('refl',
          '''
          window min1=5.5 max1=8 | grey title="Reflections"
          ''')

Flow('diff','mig0 refl','add scale=1,-1 ${SOURCES[1]}')
Result('diff','window min1=5.5 max1=8 | grey title="Diffractions"')

Flow('stoltd','diff',
        '''
        cosft sign2=1 |
        pad beg1=1250
        ''')

Flow('velcond','stoltd',
        '''
        vczo nv=101 dv=10 v0=1400 verb=y pad2=3000 |
        window min1=5.5 max1=8 |
        transp plane=23 memsize=1000 |
        cosft sign2=-1
        ''')
Plot('velcond','grey title="Velocity Continuation"',view=1)
Result('velcond',
          '''
          byte gainpanel=a |
          grey3 frame1=300 frame2=200 frame3=50
          point1=0.8 point2=0.8 flat=n unit3=m/s
          title="Velocity Continuation"
          ''')

Flow('focus','velcond',
        '''
        transp plane=23 |
        pad beg1=1250
        focus rect1=40 rect3=20 |
        window min1=5.5 |
        math output="1/abs(input)" |
        clip clip=8 |
        scale axis=2
        ''')

Flow('fpik','focus','pick v0=1400 rect1=25 rect2=10')

Result('fpik',
          '''
          window | grey color=j allpos=y bias=1500
          scalebar=y title="Focusing Velocity"
```

```
        barreverse=y barlabel=Velocity barunit=m/s
        ''')

Flow('mdif','velcond fpik',
       '''
       transp plane=23 memsize=1000 |
       slice pick=${SOURCES[1]}
       ''')
Result('mdif',
        'grey title="Focused Diffractions"')

Flow('mfpik','velcon fpik',
       '''
       transp plane=23 memsize=1000 |
       slice pick=${SOURCES[1]}
       ''')
Result('mfpik',
        'grey title="Migration with Focusing Velocity"')

# RTM
# Dix conversion to interval velocity
Flow('first1','slice',
       '''
       put o2=0 d2=16.667 label2=Distance unit2=m o3=0 |
       cosft sign2=1 |
       stolt vel=1550
       ''')

Flow('mig1','first1',
       '''

       cosft sign2=-1
       ''')

Result('mig1',
        '''
        window min1=5.5 max1=8.5 |
        grey title="Migration with 1550 m/s"
        ''')
Flow('weight','envelope vpick','slice pick=${SOURCES[1]}')
Flow('vdix2','vpick weight',
      'dix rect1=20 rect2=50 weight=${SOURCES[1]}')
Result('vdix2',
        '''

        grey color=j mean=y barreverse=y
        title="Dix Velocity in time"
        scalebar=y barlabel=Velocity
```

```
          barunit=m/s label2="Distance" unit2=m
          ''')
Flow('vmigr1','slice',
        '''
        pad beg1=1375 | math output="1550" |
        window n1=1375 | put o1=0 d2=16.667 o2=0
        ''')
Flow('vmigr2','vdix2','window min1=5.5')
Flow('vmigr','vmigr1 vmigr2','cat ${SOURCES[1:2]} axis=1')

Result('vmigr',
          '''

          grey color=j mean=y barreverse=y
          title="Migration Velocity in time"
          scalebar=y barlabel=Velocity
          barunit=m/s label2="Distance" unit2=m
          ''')

# Gazdag migration
Flow('gazdag','cosft vmigr',
      'cosft sign1=-1 | gazdag velocity=${SOURCES[1]} verb=y')

Flow('image','gazdag','cosft sign2=-1')

Result('image',
          '''
          window max1=8.5 min1=5.5 | grey
          title="Phase-Shift Migration"
          ''')

# Kirchhoff time migration
Flow('kmig','slice vpick',
        '''
        put d2=16.667 o2=0 | kirchnew velocity=${SOURCES[1]}
        ''')
Result('kmig',
          '''
          window max1=8.5 min1=5.5 | grey
          title="Kirchhoff Post-stack Migration"
          ''')

Flow('vdixz','vmigr',
        '''
        time2depth velocity=$SOURCE intime=y nz=2750 z0=0 dz=5 |
        put label1=Depth unit1=m
        ''')
Result('vdixz',
```

```
          '''

          grey color=j mean=y barreverse=y
          title="Dix Velocity in depth"
          scalebar=y barlabel=Velocity
          barunit=m/s label2="Distance" unit2=m
          ''')
Flow('fft','vdixz','transp | fft1 | fft3 axis=2 pad=1')
Flow('right left','vdixz fft',
        '''
        transp | scale dscale=0.5 |
        isolr2 seed=2016 dt=0.002 npk=50
        fft=${SOURCES[1]} left=${TARGETS[1]}
        ''')

Flow('rtm snaps','slice left right',
        '''
        put o2=0 d2=16.667 | spline n1=5500 o1=0 d1=0.002 |
        reverse which=1 |
        transp | fftexp0 mig=y snap=10 snaps=${TARGETS[1]}
        left=${SOURCES[1]} right=${SOURCES[2]}
        nz=2750 dz=5
        ''')

Result('rtm',
          '''
          put d2=1 o2=900 unit1=m label1=Depth
          unit2= label2=CMP | window min1=4000 max1=7000 |
          grey title="Post-Stack Depth Migration"
          ''')
Plot('snaps','grey title=Snapshots gainpanel=all unit2=m',view=1)

# Split-step
Flow('slo','vdixz',
        '''
        put d4=1 o4=1 d2=0.016667 d1=0.005 d3=0.016667 |
        transp | transp plane=23 |
        math output="1/(input/1000)"
        ''')
Result('slo',
          '''
          put label1=Distance unit1=km label2=
          unit2= unit3=km | byte |
          sfgrey3 color=j frame1=150 frame2=150 frame3=150
          flat=n title="Slowness"
          ''')
Flow('fft2','slice',
        '''
```

```
        put d2=0.016667 o2=0 d3=0.016667 | fft1 |
        transp plane=12 | transp plane=23
        ''')
Flow('mig','fft2 slo',
        '''
        zomig3 ompnth=1 mode=m --readwrite=y verb=y
        nrmax=10 slo=${SOURCES[1]}
        ''',split=[3,'omp',[0]],reduce='add')

Result('mig',
           '''
           put d1=1 o1=900 d3=5 d2=16.667 unit3=m
           label3=Depth unit1= label1=CMP |
           transp plane=23 | transp plane=12 |
           window min1=4000 max1=7000 | grey
           title="Split-step migration"
           ''')

##################################################
# Zoom an interesting area
##################################################
min1,max1=6.0,6.8
min2,max2=950,1050

for i in range (3):
    case=('slice','mig2','slice1')[i]
    zoom = case + '-zoom'
    Flow(zoom,case,
            '''
            window min1=%g max1=%g min2=%g max2=%g
            ''' % (min1,max1,min2,max2))
    Plot(zoom,'grey title=%s grid=y gridcol=9 label2=CMP unit2='%('123'[i]))
for i in range (5):
    case=('mstack','mfpik','image','stack-shot','kmig')[i]
    zoom2 = case + '-zoom'
    Flow(zoom2,case,
            '''
            put d2=1 o2=900 | window min1=%g max1=%g min2=%g max2=%g
            ''' % (min1,max1,min2,max2))
    Plot(zoom2,
           '''
           grey title=%s grid=y gridcol=9
           label2=CMP unit2=
           ''' % ('45609'[i]))
Plot('mig-zoom','mig',
        '''
        put d1=1 o1=900 d3=5 d2=16.667
        unit3=m label3=Depth unit1= label1=CMP |
```

```
       transp plane=23 | transp plane=12 |
       window min1=4730 max1=5550 min2=950 max2=1050 |
       grey title="%s" grid=y gridcol=9
       ''' % ('8'))
Plot('rtm-zoom','rtm',
       '''
       put d2=1 o2=900 unit1=m label1=Depth unit2= label2=CMP |
       window min1=4730 max1=5550 min2=950 max2=1050 |
       grey title="%s" grid=y gridcol=9
       ''' % ('7'))
Result('zoom1','slice-zoom mig2-zoom slice1-zoom','SideBySideIso')
Result('zoom2','mstack-zoom mfpik-zoom image-zoom','SideBySideIso')
Result('zoom3','rtm-zoom mig-zoom kmig-zoom stack-shot-zoom',
        'SideBySideIso')

################################################
# Zoom an interesting area
################################################
min1,max1=5.9,6.8
min2,max2=950,1200

for i in range (3):
     case=('slice','mig2','slice1')[i]
     zoom = case + '-zoom2'
     Flow(zoom,case,
            '''
            window min1=%g max1=%g min2=%g max2=%g
            ''' % (min1,max1,min2,max2))
     Plot(zoom,'grey title=%s grid=y gridcol=9 label2=CMP unit2='%('123'[i]))
for i in range (5):
     case=('mstack','mfpik','image','stack-shot','kmig')[i]
     zoom2 = case + '-zoom2'
     Flow(zoom2,case,
            '''
            put d2=1 o2=900 | window min1=%g max1=%g min2=%g max2=%g
            ''' % (min1,max1,min2,max2))
     Plot(zoom2,
            '''
            grey title=%s grid=y gridcol=9
            label2=CMP unit2=
            ''' % ('45609'[i]))
Plot('mig-zoom2','mig',
       '''
       put d1=1 o1=900 d3=5 d2=16.667 unit3=m
       label3=Depth unit1= label1=CMP |
       transp plane=23 | transp plane=12 |
       window min1=4600 max1=5550 min2=950 max2=1200 |
       grey title="%s" grid=y gridcol=9
```

```
        ''' % ('8'))
Plot('rtm-zoom2','rtm',
        '''
        put d2=1 o2=900 unit1=m label1=Depth unit2= label2=CMP |
        window min1=4600 max1=5550 min2=950 max2=1200 |
        grey title="%s" grid=y gridcol=9
        ''' % ('7'))
Result('zoom4','slice-zoom2 mig2-zoom2 slice1-zoom2','SideBySideIso')
Result('zoom5','mstack-zoom2 mfpik-zoom2 image-zoom2','SideBySideIso')
Result('zoom6','rtm-zoom2 mig-zoom2 kmig-zoom2 stack-shot-zoom2',
          'SideBySideIso')

###################################################
# Zoom an interesting area
###################################################
min1,max1=5.9,6.8
min2,max2=1150,1250

for i in range (3):
    case=('slice','mig2','slice1')[i]
    zoom = case + '-zoom3'
    Flow(zoom,case,
            '''
            window min1=%g max1=%g min2=%g max2=%g
            ''' % (min1,max1,min2,max2))
    Plot(zoom,
            '''
            grey title=%s grid=y gridcol=9 label2=CMP unit2=
            ''' % ('123'[i]))
for i in range (5):
    case=('mstack','mfpik','image','stack-shot','kmig')[i]
    zoom2 = case + '-zoom3'
    Flow(zoom2,case,
            '''
            put d2=1 o2=900 | window min1=%g max1=%g min2=%g max2=%g
            ''' % (min1,max1,min2,max2))
    Plot(zoom2,
            '''
            grey title=%s grid=y gridcol=9 label2=CMP unit2=
            ''' % ('45609'[i]))
Plot('mig-zoom3','mig',
        '''
        put d1=1 o1=900 d3=5 d2=16.667 unit3=m
        label3=Depth unit1= label1=CMP |
        transp plane=23 | transp plane=12 |
        window min1=4600 max1=5550 min2=1150 max2=1250 |
        grey title="%s" grid=y gridcol=9
        ''' % ('8'))
```

```
Plot('rtm-zoom3','rtm',
     '''
     put d2=1 o2=900 unit1=m label1=Depth unit2=
     label2=CMP | window min1=4600 max1=5550
     min2=1150 max2=1250 | grey title="%s" grid=y gridcol=9
     ''' % ('7'))
Result('zoom7','slice-zoom3 mig2-zoom3 slice1-zoom3','SideBySideIso')
Result('zoom8','mstack-zoom3 mfpik-zoom3 image-zoom3','SideBySideIso')
Result('zoom9','rtm-zoom3 mig-zoom3 kmig-zoom3 stack-shot-zoom3',
       'SideBySideIso')

End()
```

REFERENCES

Fletcher, R., and C. M. Reeves, 1964, Function minimization by conjugate gradients: Computer Journal, **7**, 149–154.

Fomel, S., 2002, Applications of planewave destruction filters: GEOPHYSICS, **67**, 1946–1960.

———, 2003, Timemigration velocity analysis by velocity continuation: GEOPHYSICS, **68**, 1662–1672.

Fomel, S., L. Ying, and X. Song, 2013, Seismic wave extrapolation using lowrank symbol approximation: Geophysical Prospecting, **61**, 526–536. (`http://ahay.org/RSF/book/tccs/lowrank/paper_html/`).

Forel, D., T. Benz, and W. Pennington, 2005, Seismic data processing with seismic un x: Society of Exploration Geophysicists.

Fowler, P., 1988, Seismic velocity estimation using prestack time migration: PhD thesis, Stanford University.

Gazdag, J., 1978, Wave equation migration with the phase-shift method: Geophysics, **43**, 1342–1351.

Hestenes, M. R., and E. Stiefel, 1952, Methods of conjugate gradients for solving linear systems: J. Res. NBS, **49**, 409–436.

Mikulich, W., and D. Hale, 1992, Steep-dip v(z) imaging from an ensemble of Stolt-like migrations: Geophysics, **57**, 51–59.

Mochiduki, K., and K. Obana, 2003, Seismic activities along the nankai trough.

Moore, G., T. H. Shipley, P. Stoffa, D. Karig, A. Taira, S. Kuramoto, H. Tokuyama, and K. Suyehiro, 1990, Structure of the nankai trough accretionary zone from multichannel seismic reflection data: **95**, 8753–8766.

Stoffa, P. L., J. T. Fokkema, R. M. de Luna Freire, and W. P. Kessinger, 1990, Splitstep fourier migration: GEOPHYSICS, **55**, 410–421.

Stolt, R. H., 1985, Migration by Fourier transform: Geophysics, **50**, 2219–2244. (Discussion and reply in GEO-60-5-1583).

Taner, M. T., and F. Koehler, 1981, Surface consistent corrections: Geophysics, **46**, 17–22.