

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).

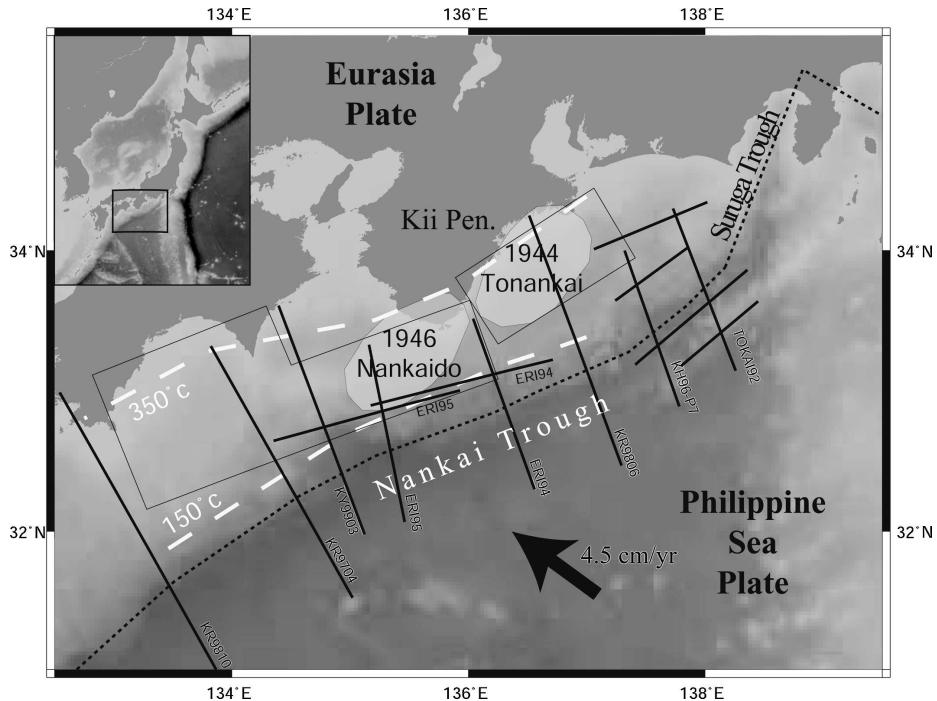


Figure 1: Location of the Nankai Trough (Image from Mochiduki and Obana (2003)).
project/ geo

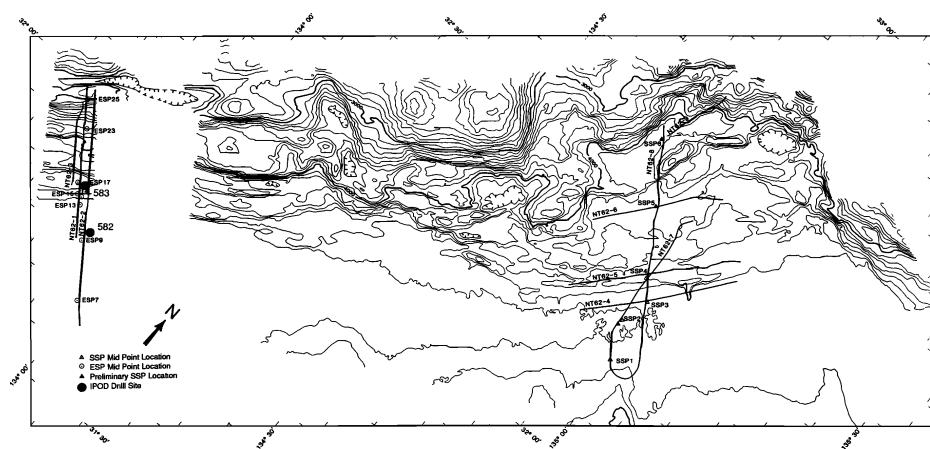


Figure 2: Seismic acquisition (Image from Mochiduki and Obama (2003)).

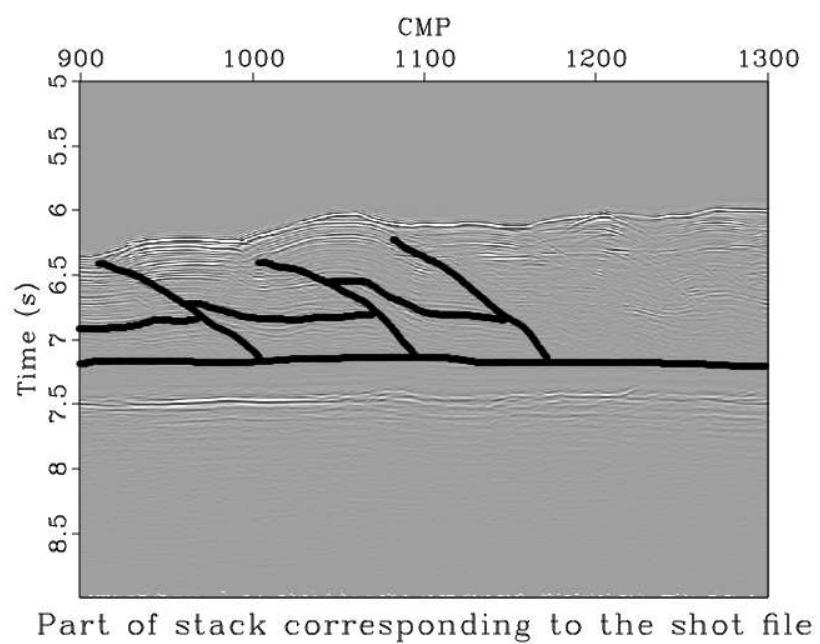


Figure 3: Geologic structure. [project/ stack-shot-interpret](#)

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 in eastern area and 8.33-m bins in western areas.
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.

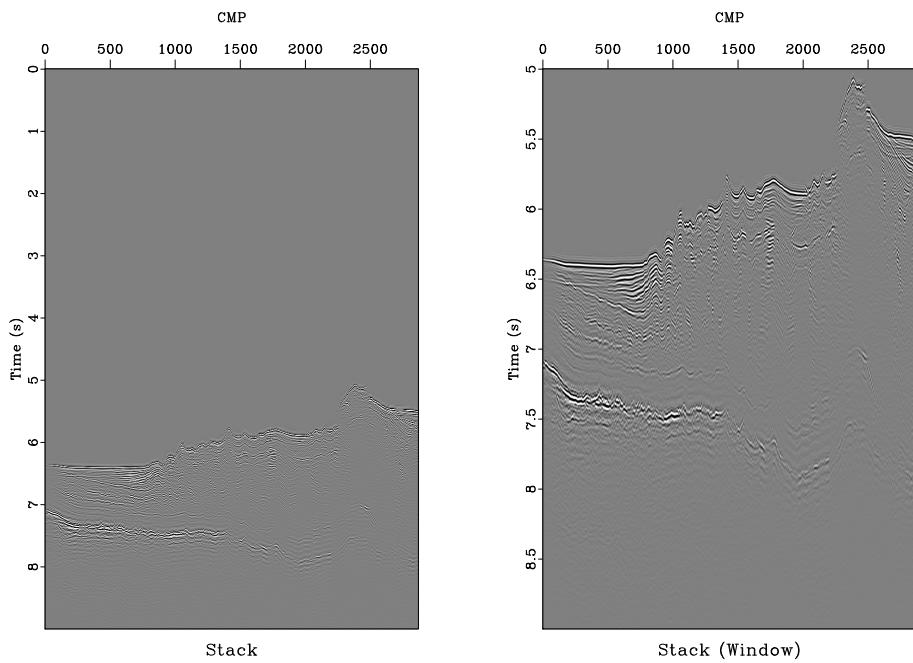


Figure 4: Published stacked data. [project/ stackd](#)

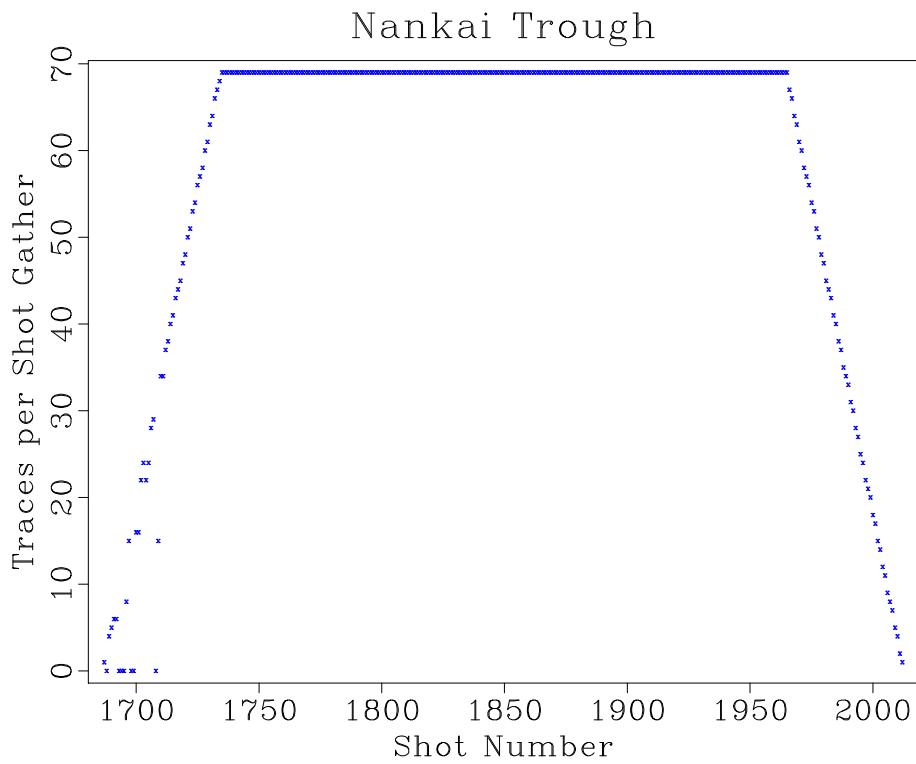


Figure 5: Number of traces per shot-gather. [project/ smask](#)

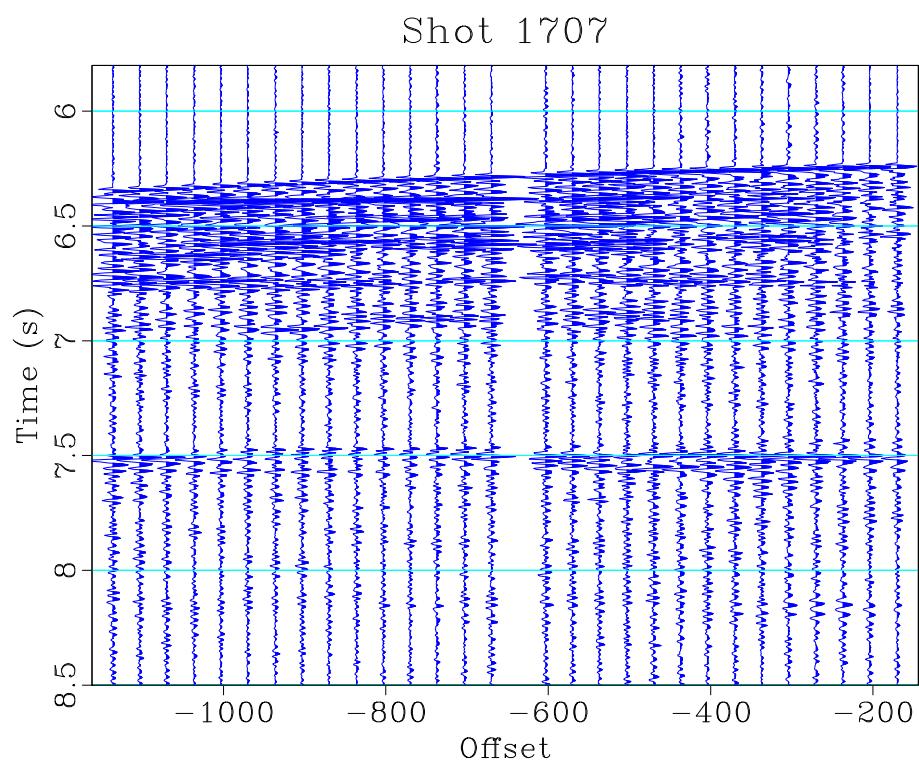


Figure 6: Shot-gather 1707. project/ shot

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.

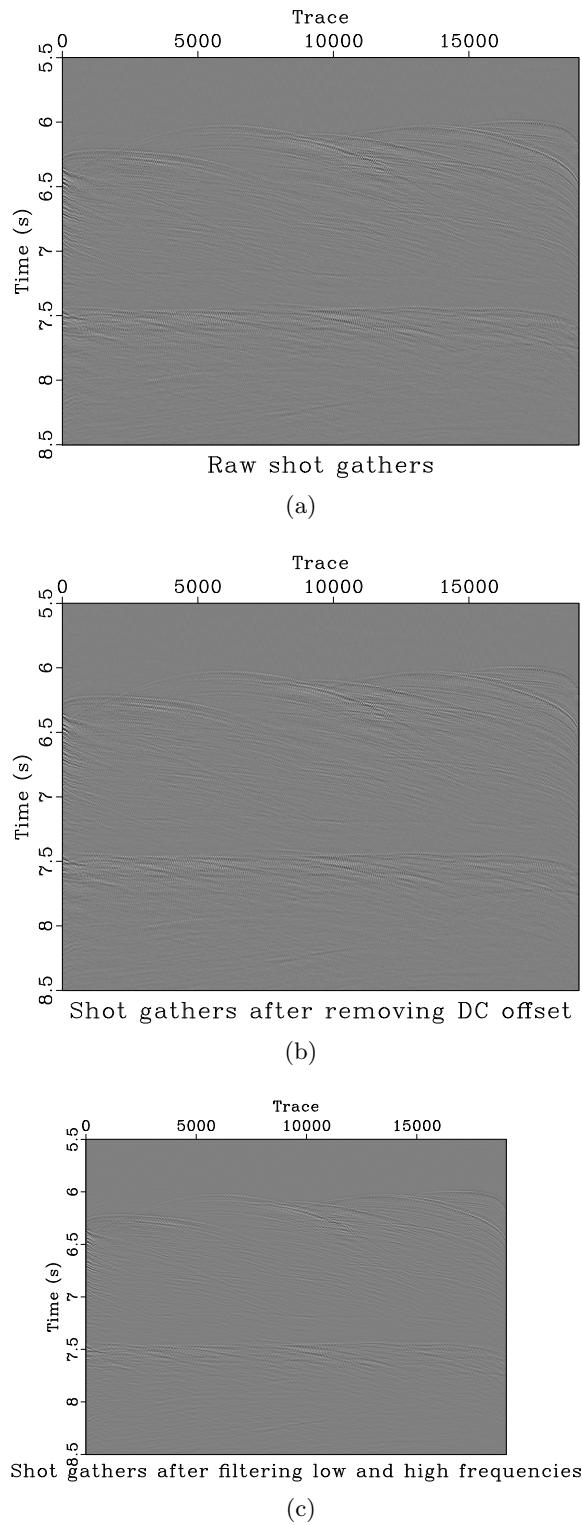


Figure 7: (a) Raw shot gathers (b) Shot gathers after DC removal. (c) Shot gathers after filtering low and high frequencies. [project/ shots,shotsdc,shotsf](#)

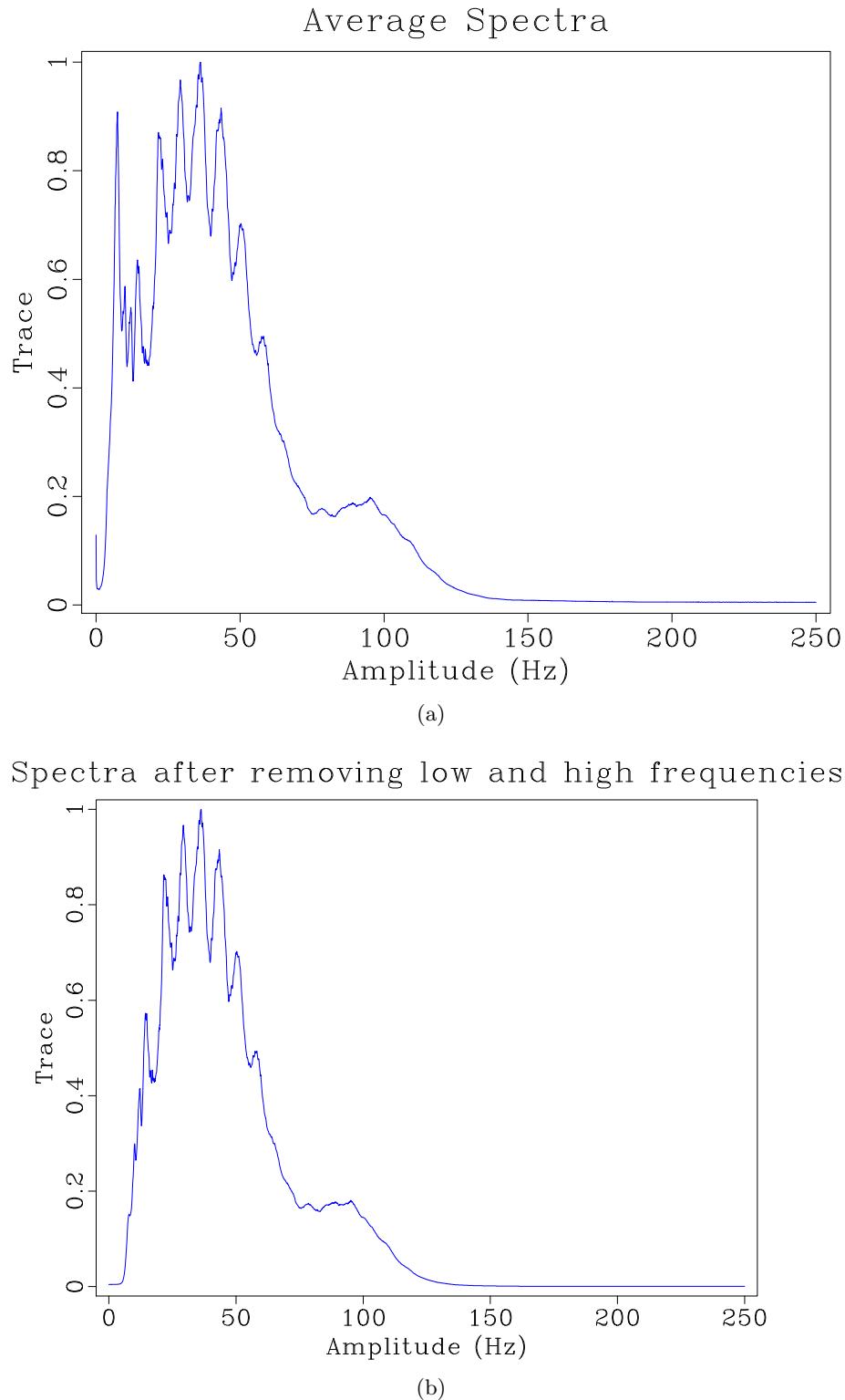


Figure 8: (a) Raw data frequency spectra. (b) Frequency spectra after filtering low and high frequencies. [project/ spectra,spectraf](#)

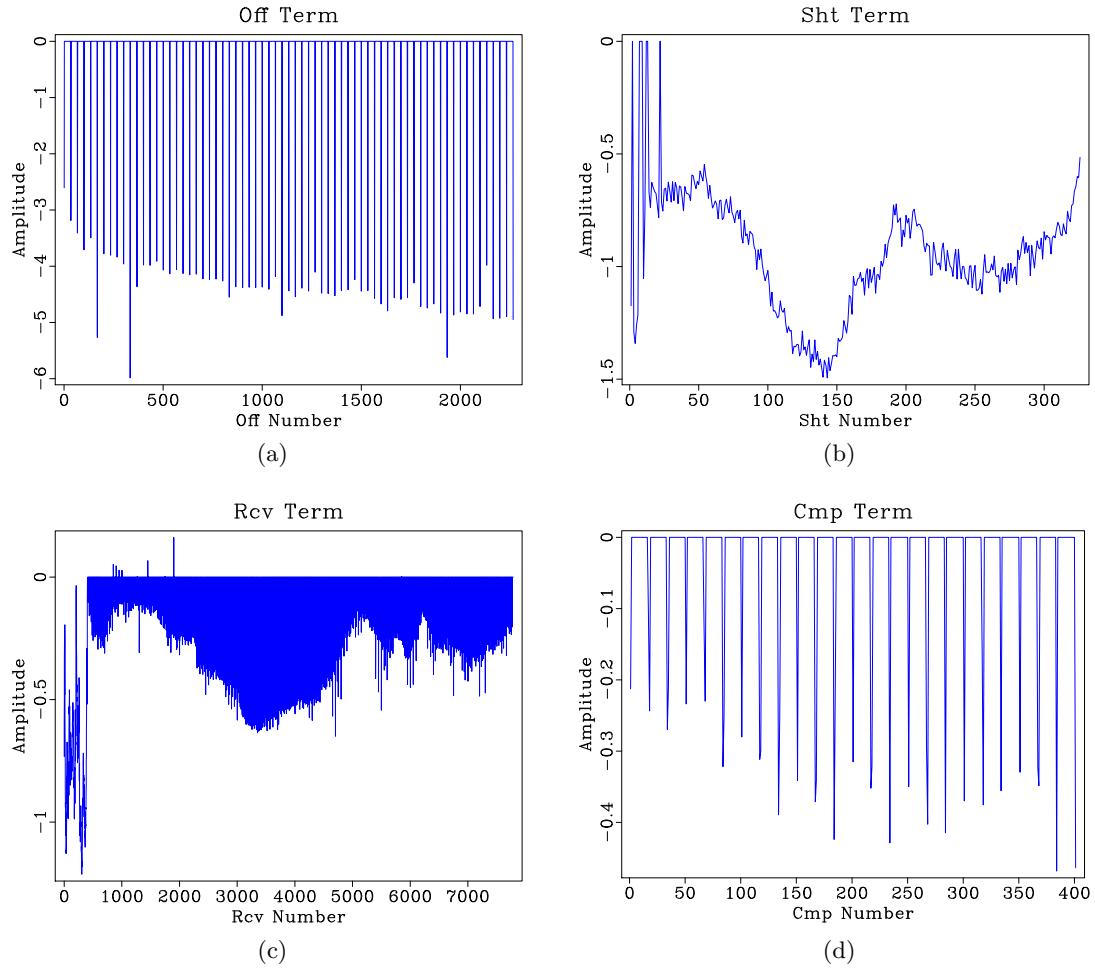


Figure 9: Estimated surface-consistent factors. [project](#) / off,sht,recv,cmp

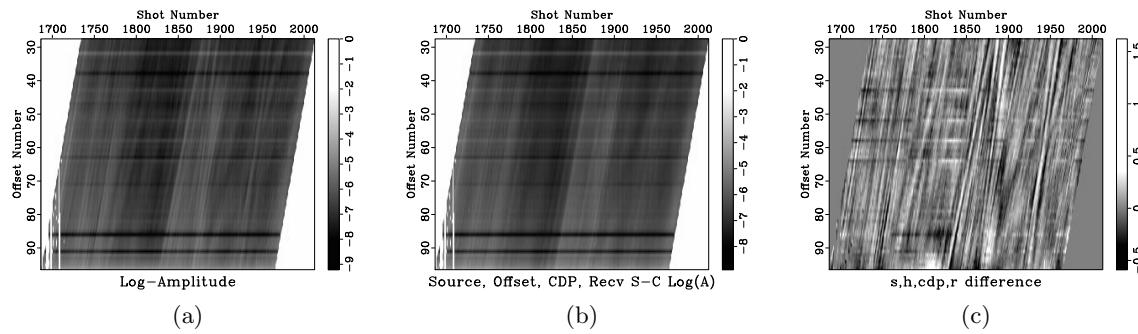


Figure 10: Trace amplitudes from the data. (a) Initial. (b) Estimated surface-consistent. (c) Difference. [project](#) / [varms](#),[recvvscarms](#),[recvadiff](#)

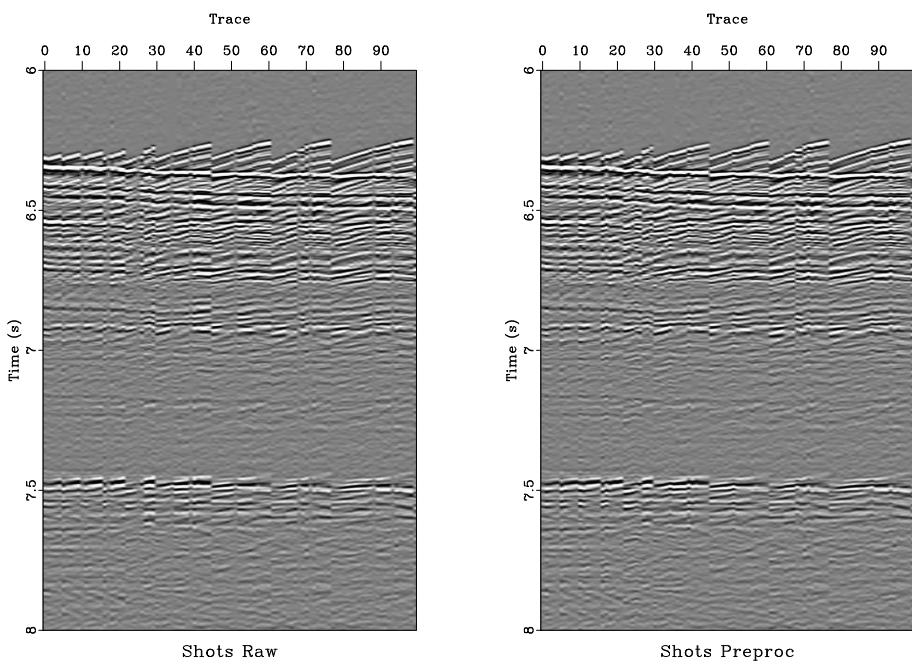


Figure 11: (a) Raw shot gathers (b) Shot gathers after surface-consistent amplitude correction. [\[project/ shotsfc\]](#)

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.

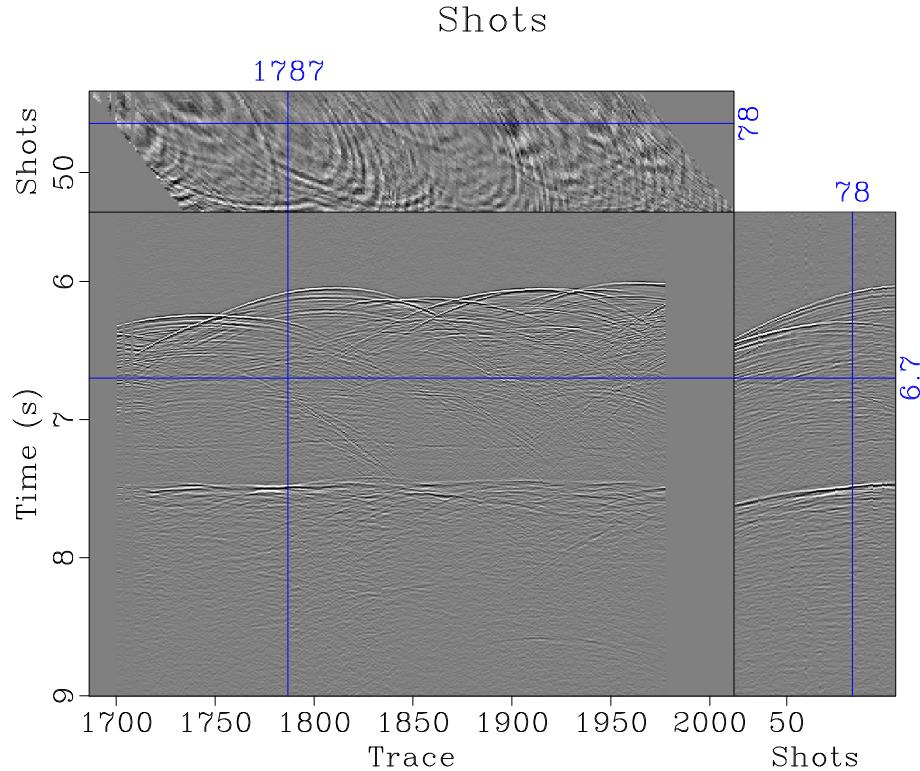


Figure 12: Shot gathers after resampling. [project/ shots2](#)

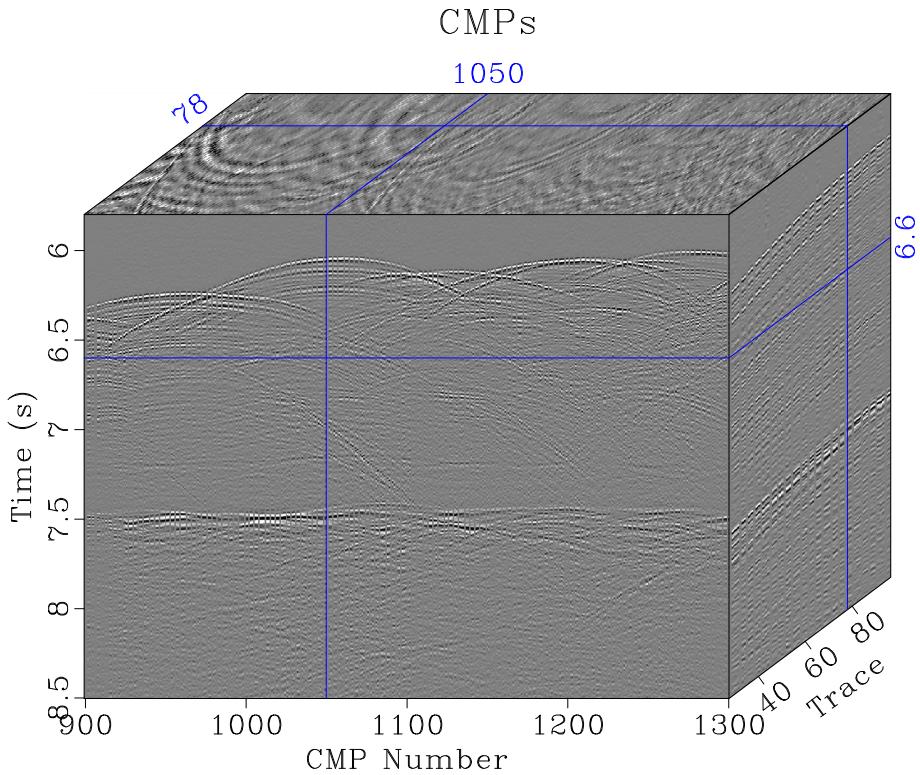


Figure 13: CMP gathers. [\[project/ cmgs\]](#)

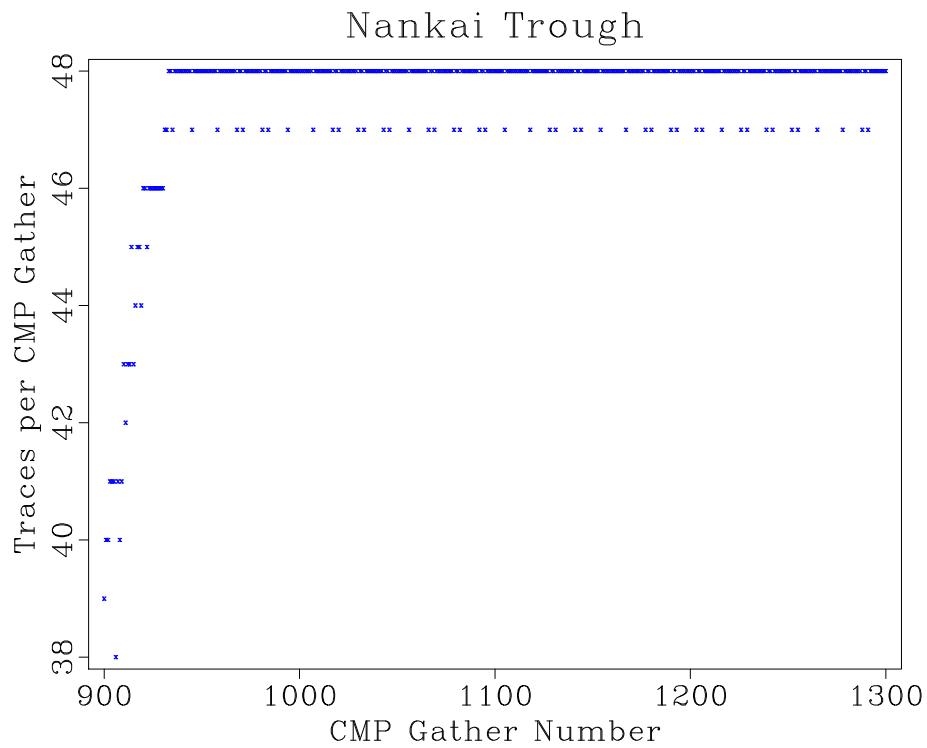


Figure 14: Fold values of data. [\[project/ cmask\]](#)

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).

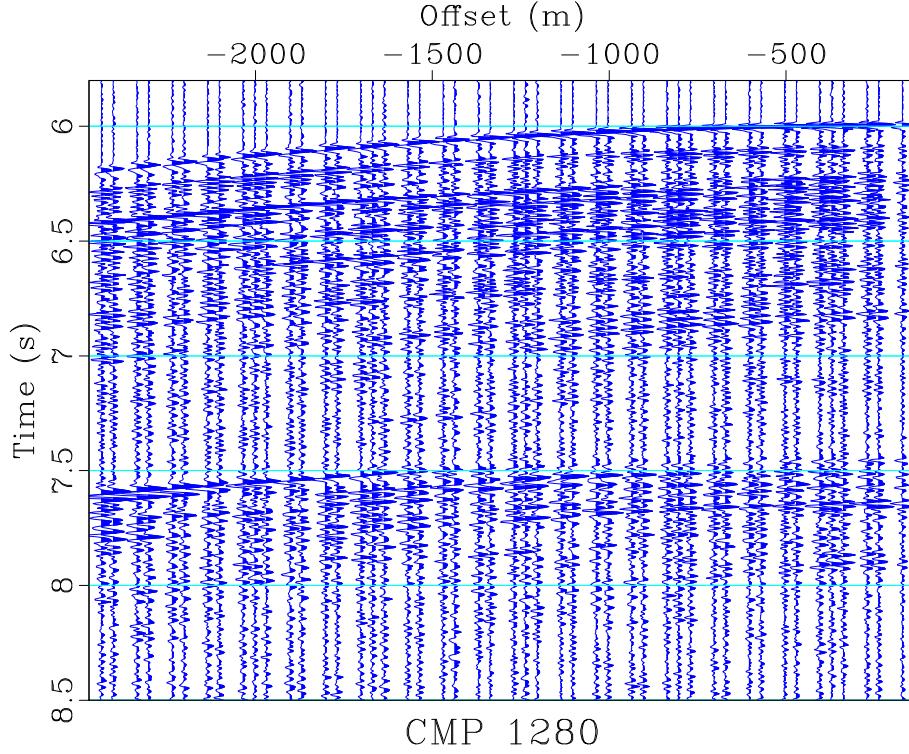


Figure 15: CMP 1280. [project / cmp1](#)

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.

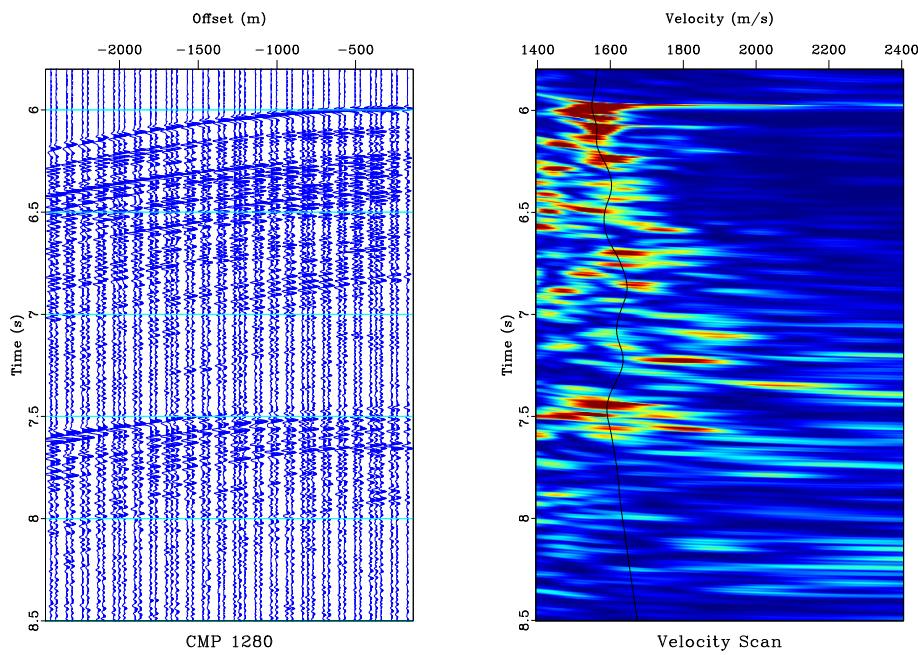


Figure 16: Velocity analysis for CMP 1280. [project/ vscan](#)

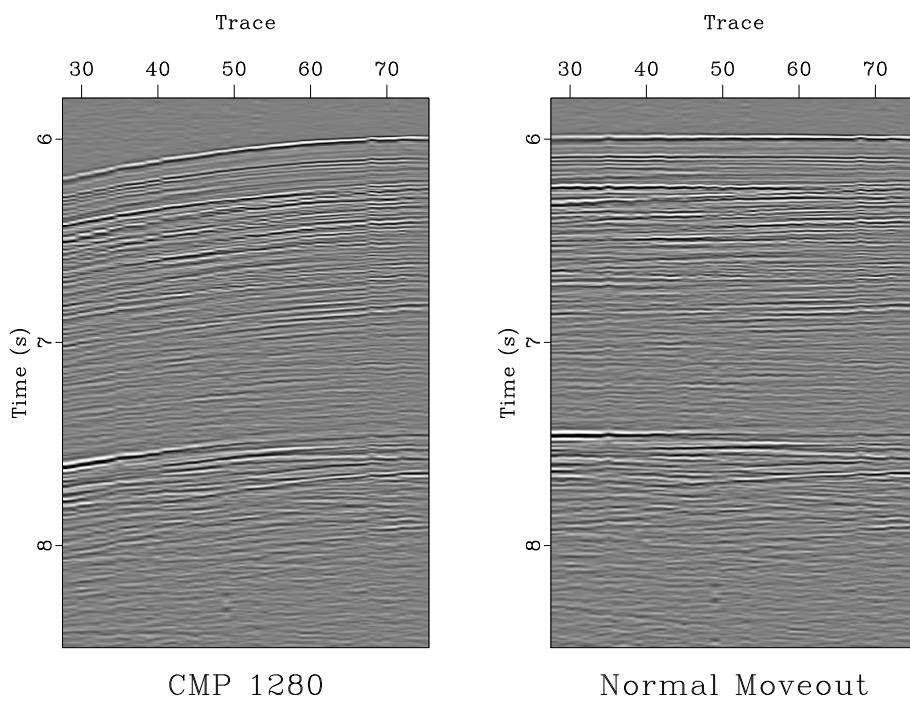


Figure 17: CMP 1280 after NMO. [project/ nmo1](#)

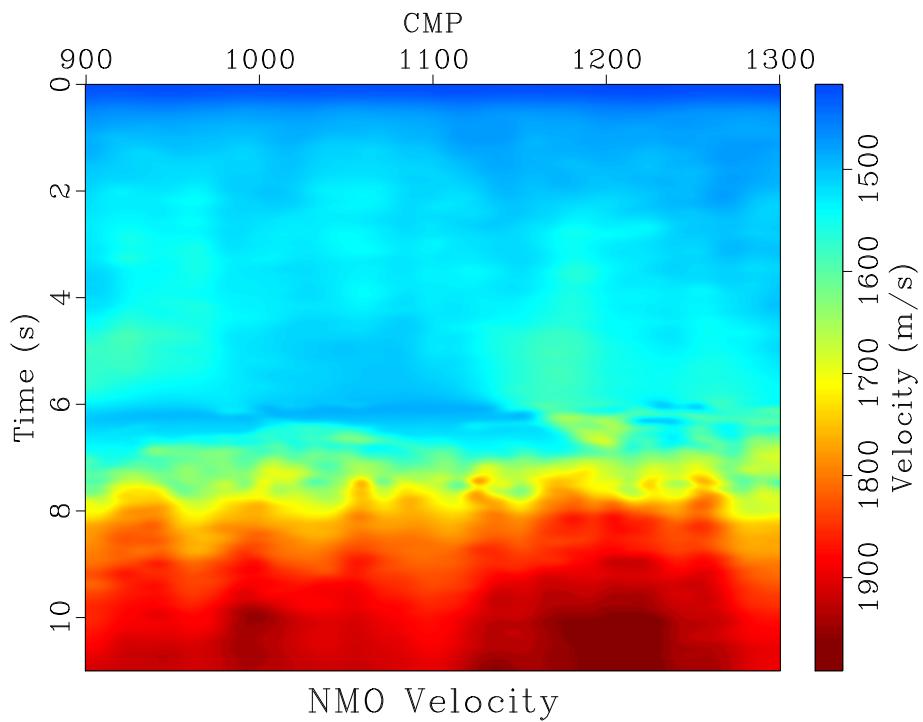


Figure 18: NMO velocity. [project/ picks](#)

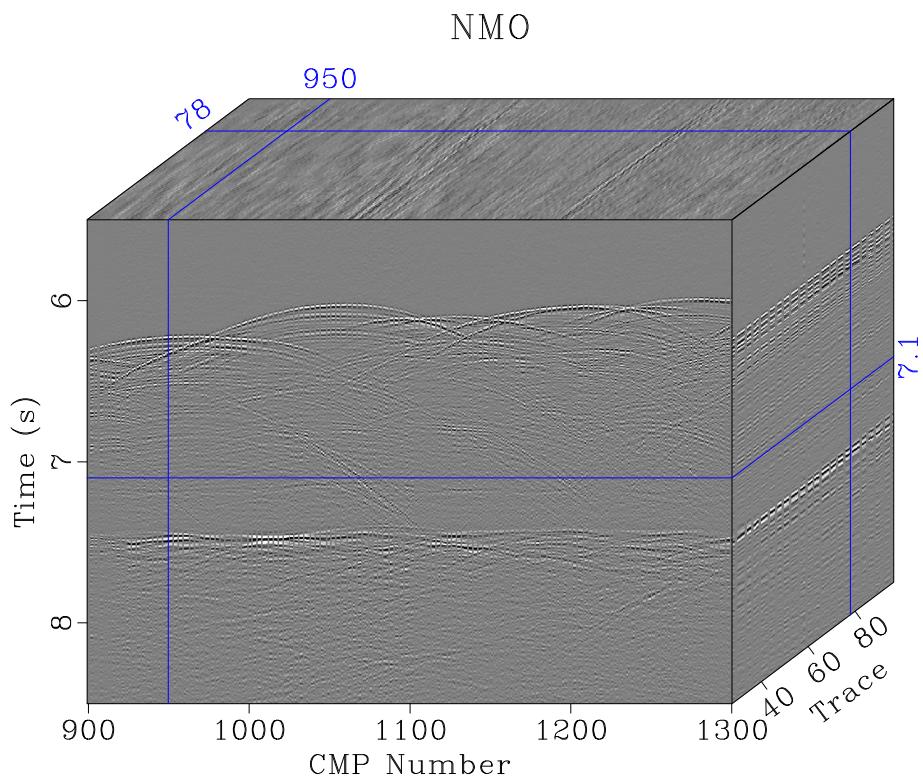


Figure 19: CMP gathers after NMO. [project/ nmos](#)

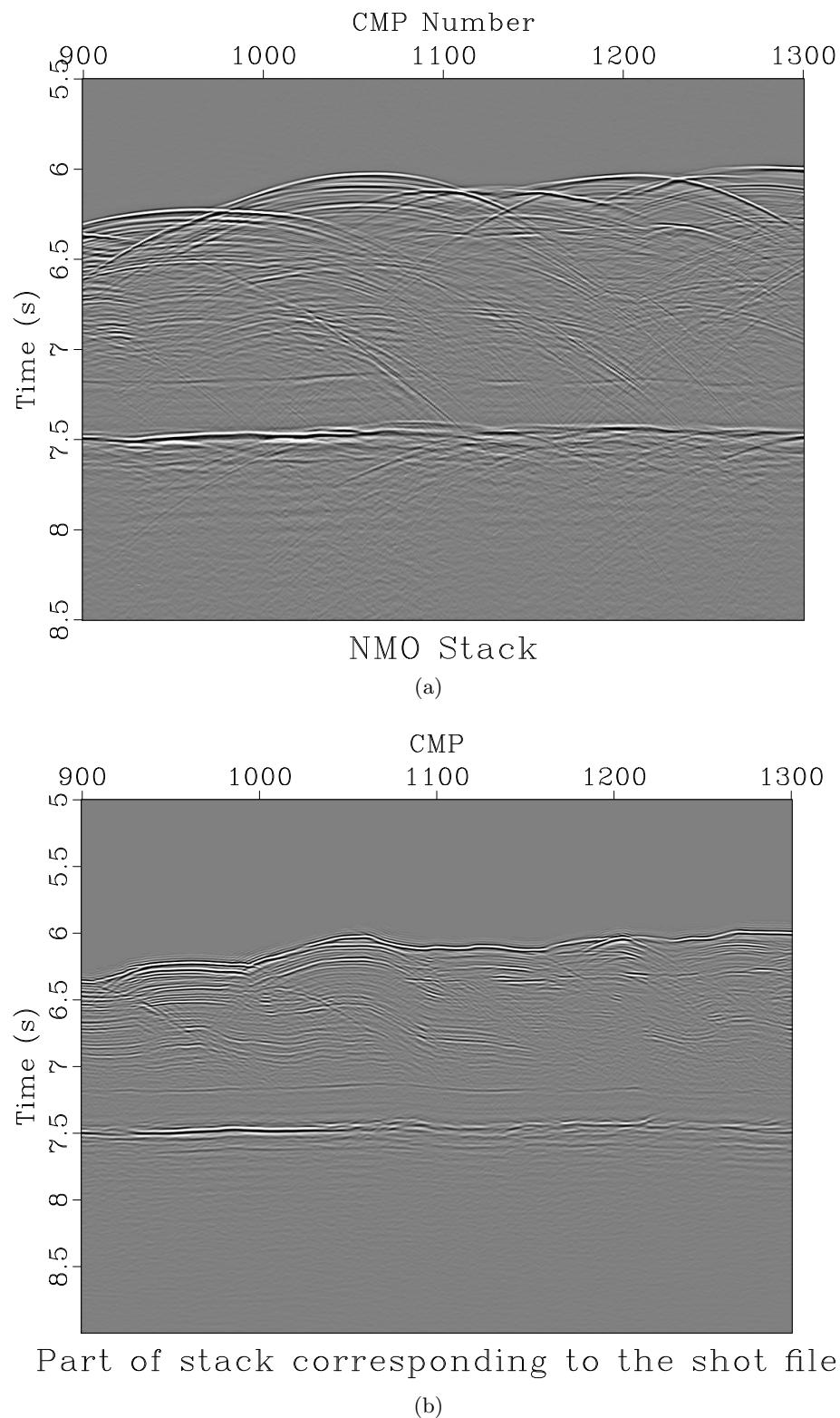


Figure 20: (a) Stacked data. (b) Published stacked data. [project/ stack,stack-shot](#)

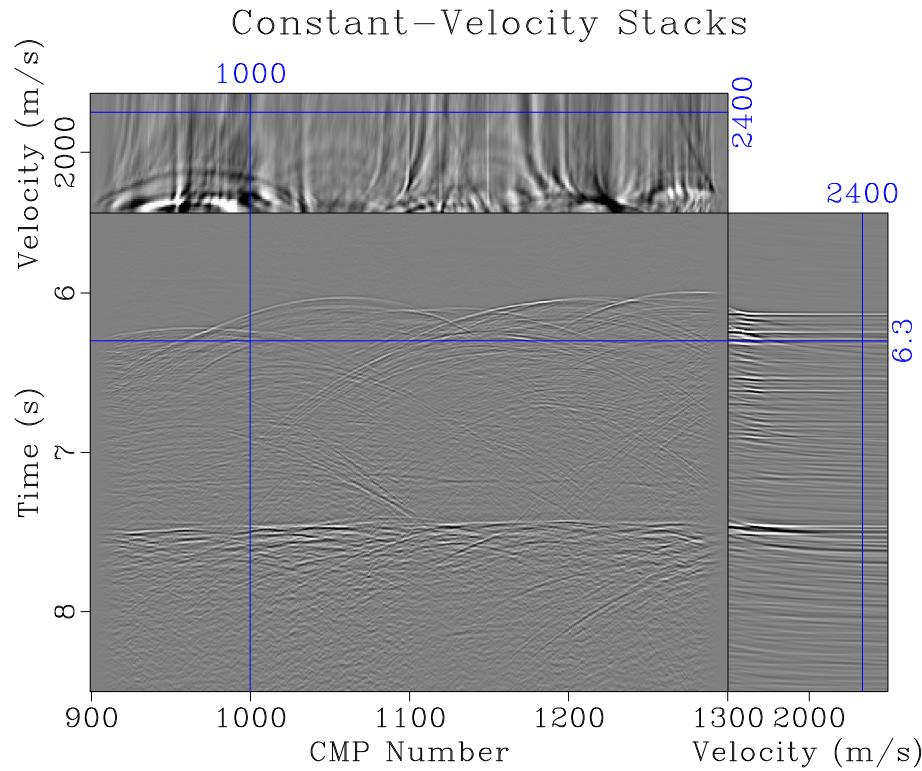


Figure 21: NMO stack with an ensemble of constant velocities. [\[project/ stacks\]](#)

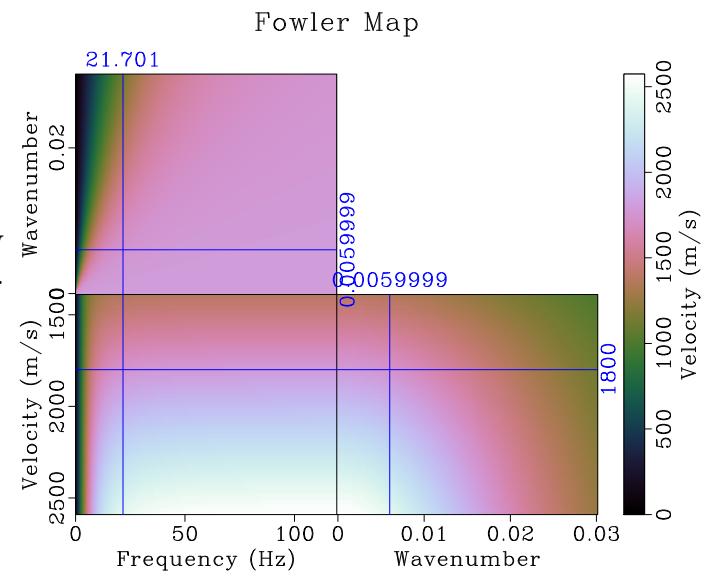


Figure 22: Fourier-domain velocity map used in Fowler's DMO method.

[\[project/ map\]](#)

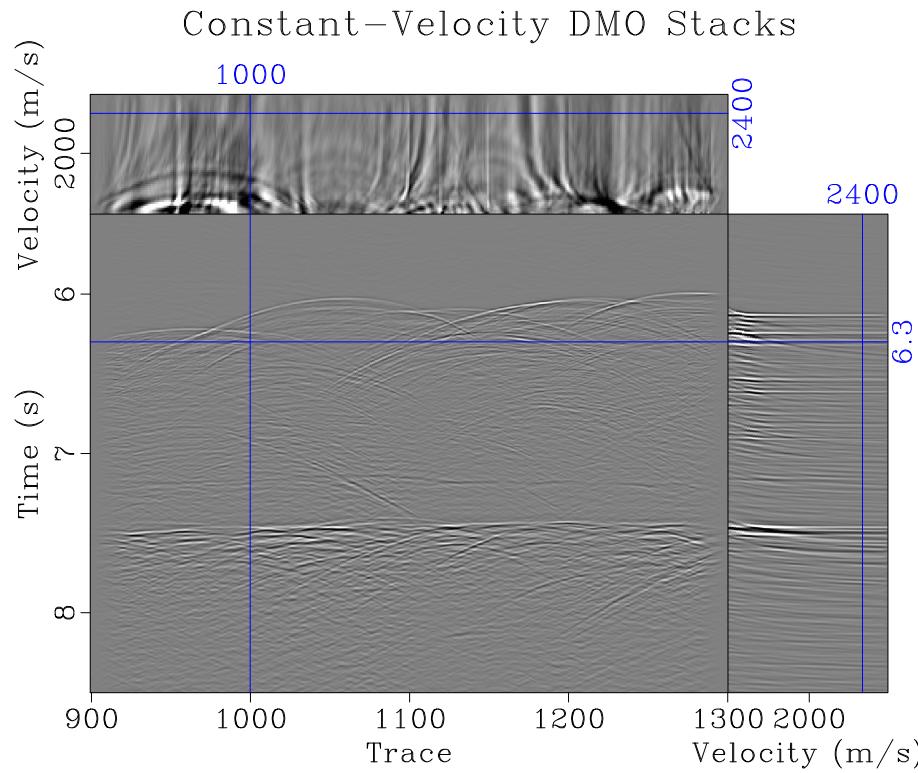


Figure 23: Nankai dataset after DMO stacking with an ensemble of constant velocities.
[project/ dmo](#)

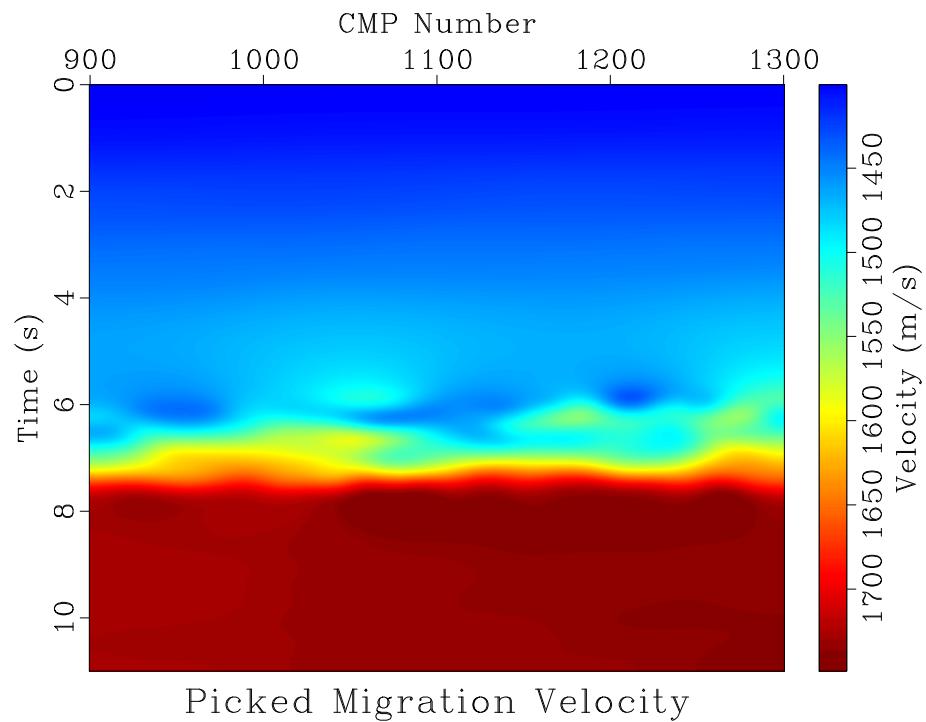


Figure 24: Migration velocity picked automatically from DMO stacks. [project/ vpick](#)

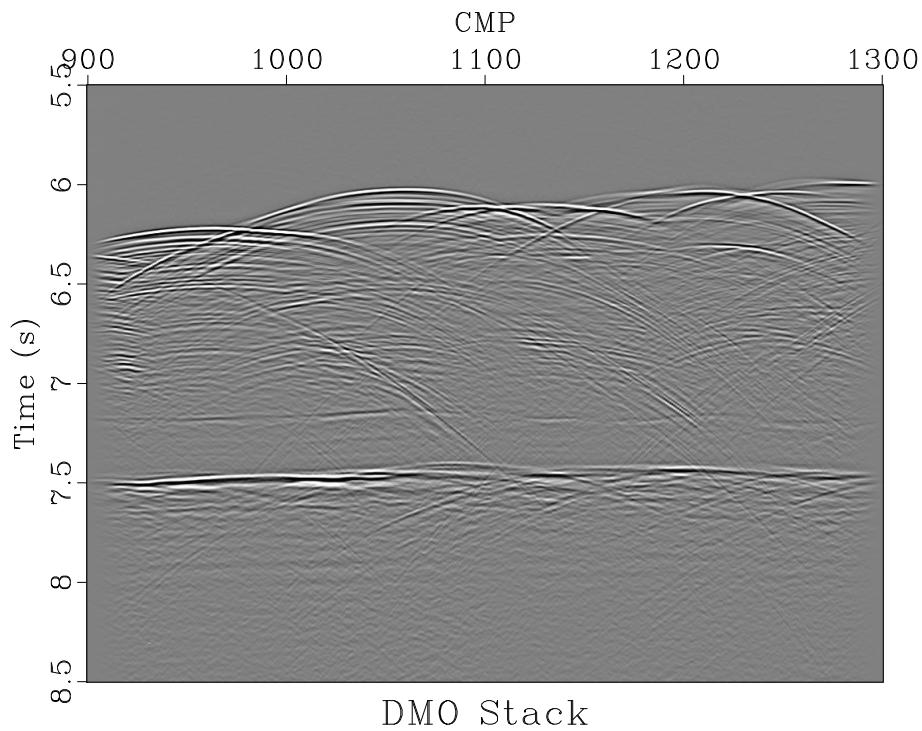


Figure 25: DMO stack generated by slicing the constant-velocity stacks.

[project/ slice](#)

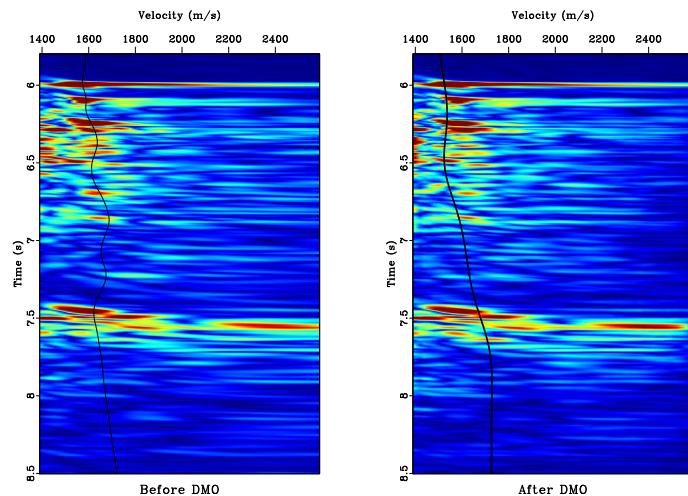


Figure 26: Comparison of velocity analysis (a) before and (b) after DMO at a selected CMP location.

[project/ envelope](#)

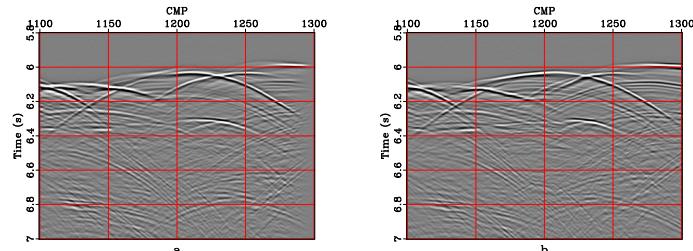


Figure 27: (a) Windowed DMO stack (b) Windowed normal stack.

[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.

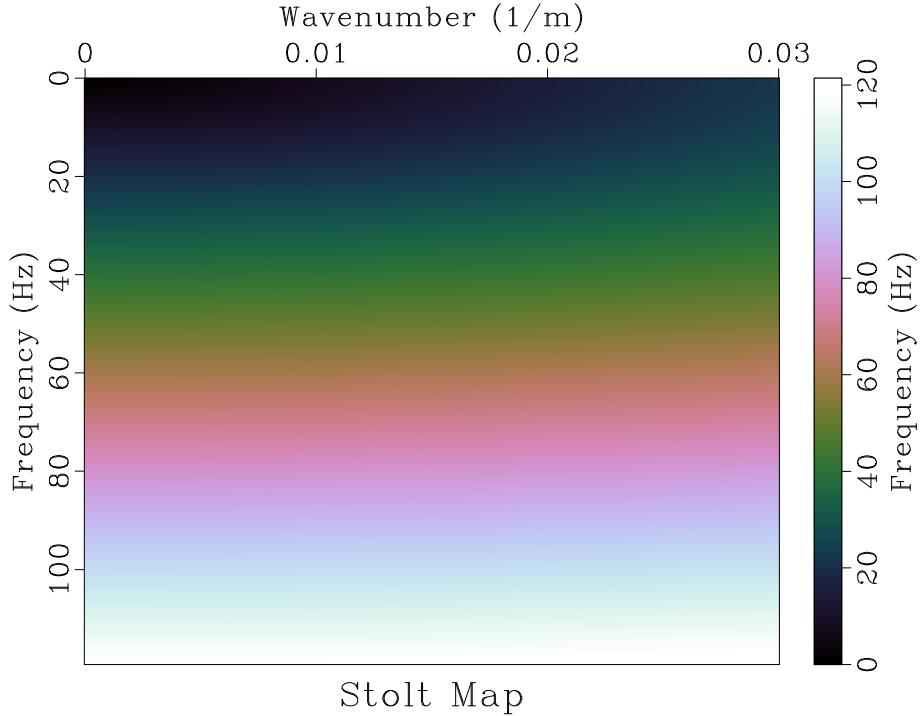


Figure 28: Stolt map. [project/ map2](#)

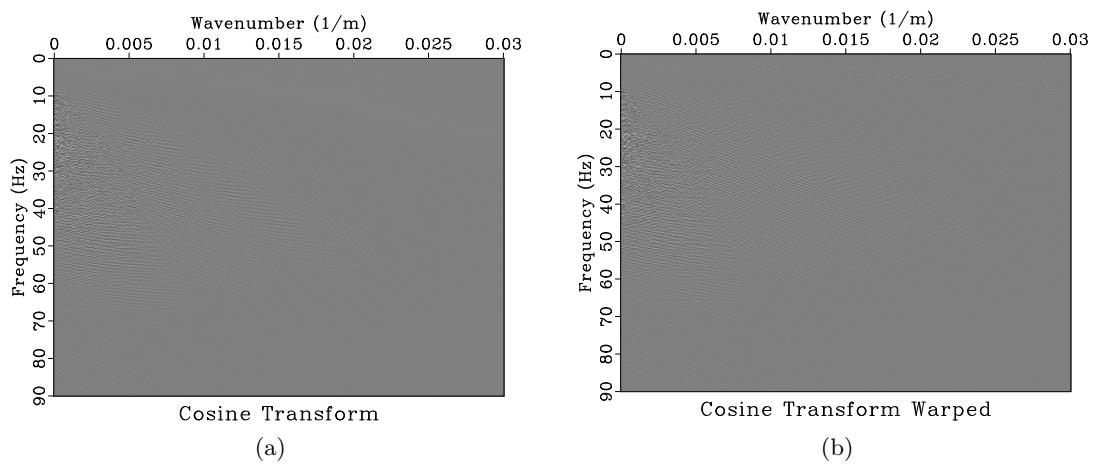


Figure 29: Nankai stack in the Cosine transform domain before (a) and after (b) Stolt migration with velocity 1500 m/s. [project/ cosft,cosft2](#)

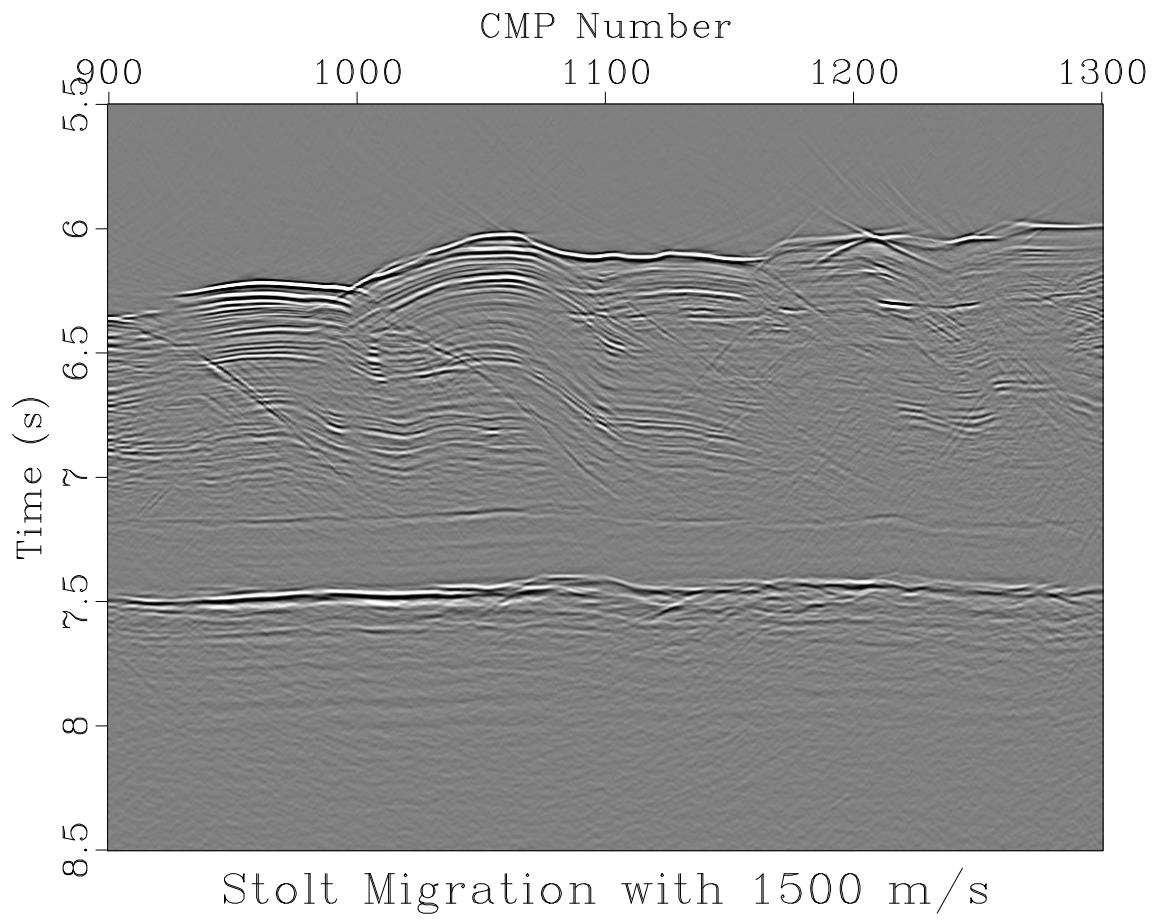


Figure 30: Nankai stack Stolt migrated with velocity of 1500 m/s. [project/ mig2](#)

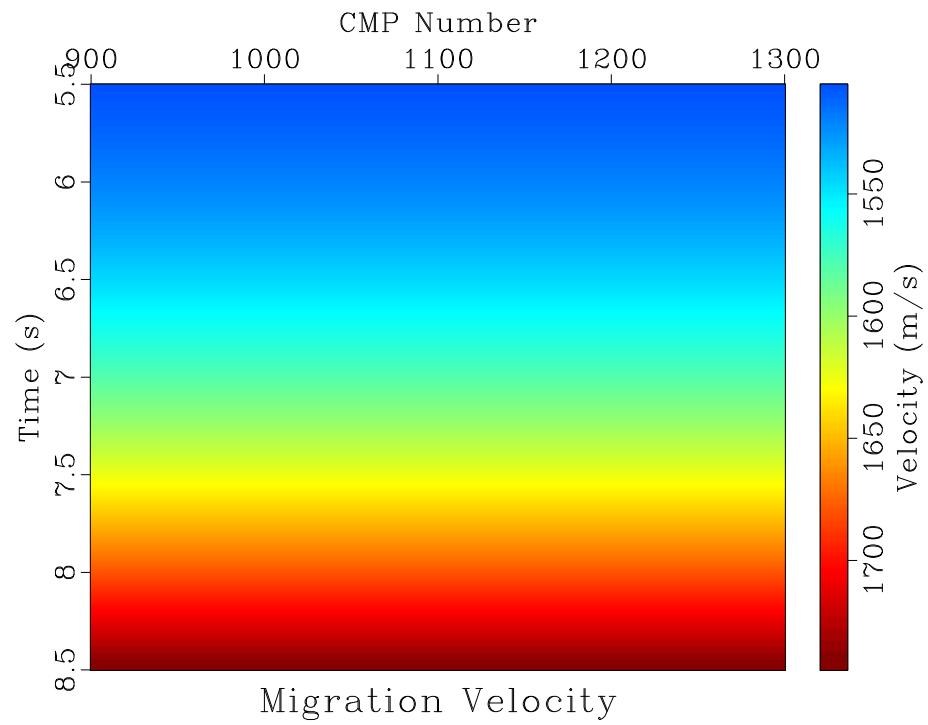


Figure 31: Velocity distribution. [\[project/ vmig\]](#)

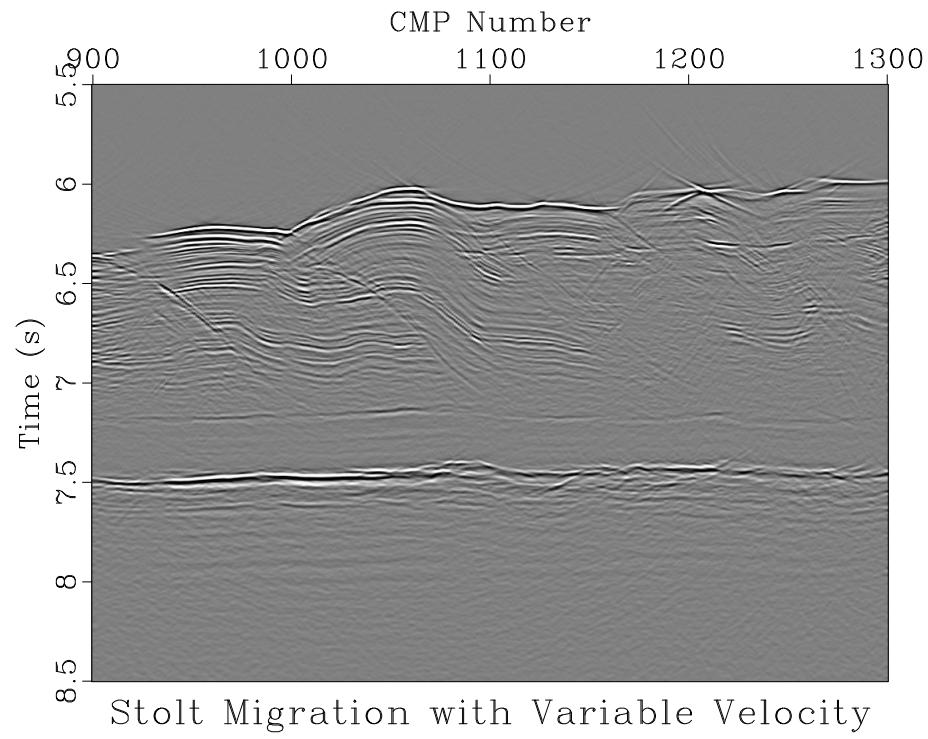


Figure 32: Nankai stack Stolt migrated with velocity increasing with time. [\[project/ migd\]](#)

Gazdag migration

1. To correctly 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.

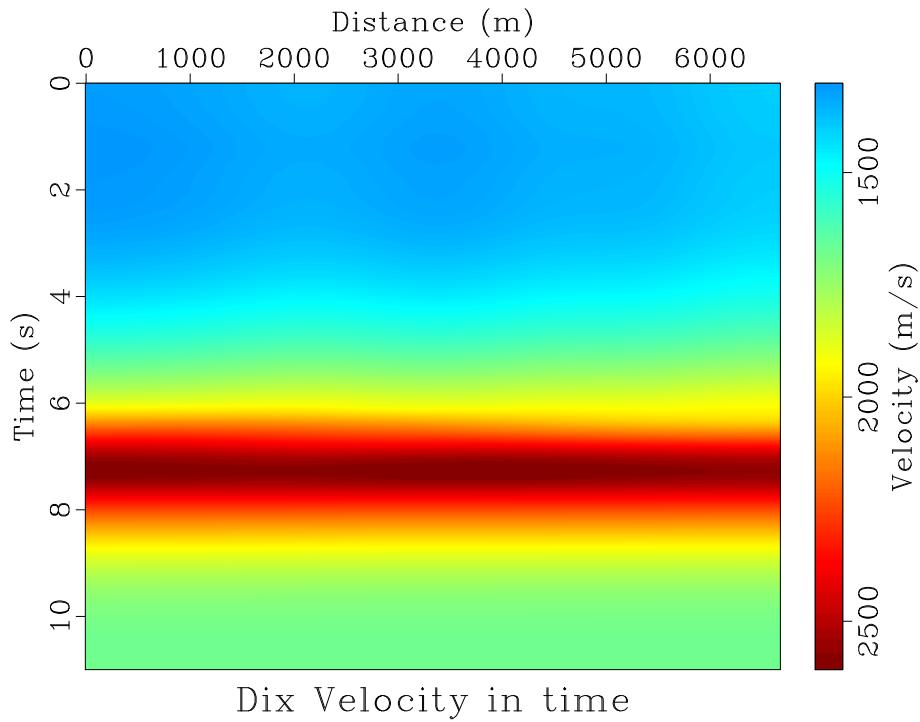


Figure 33: Dix velocity in time. [project/ vdix2](#)

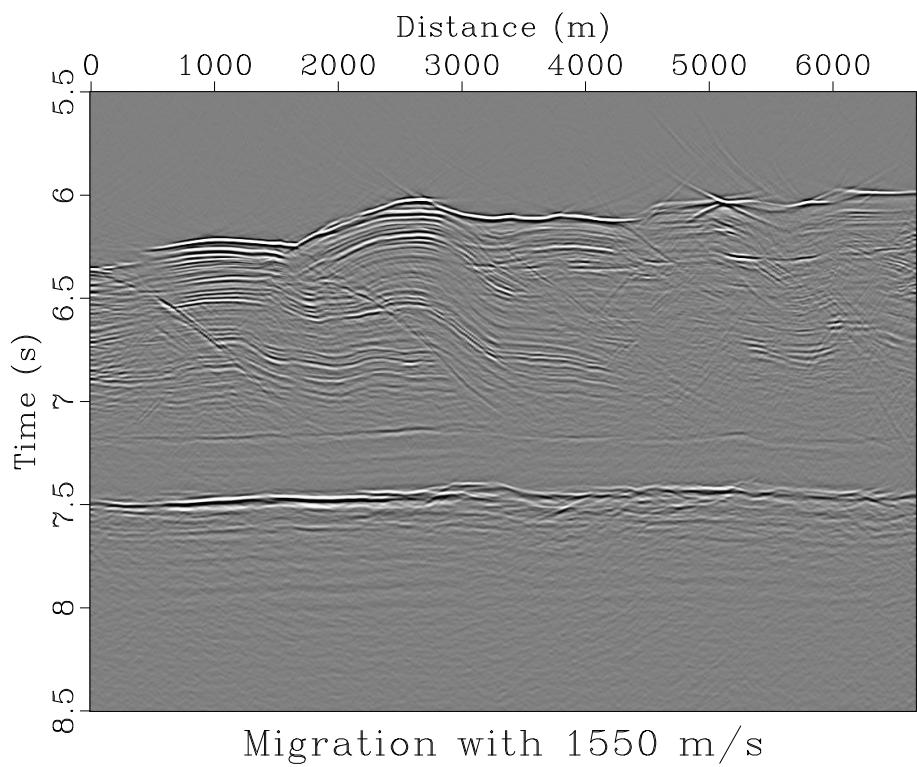


Figure 34: Stolt migration with constant velocity 1550 m/s. [\[project/ mig1\]](#)

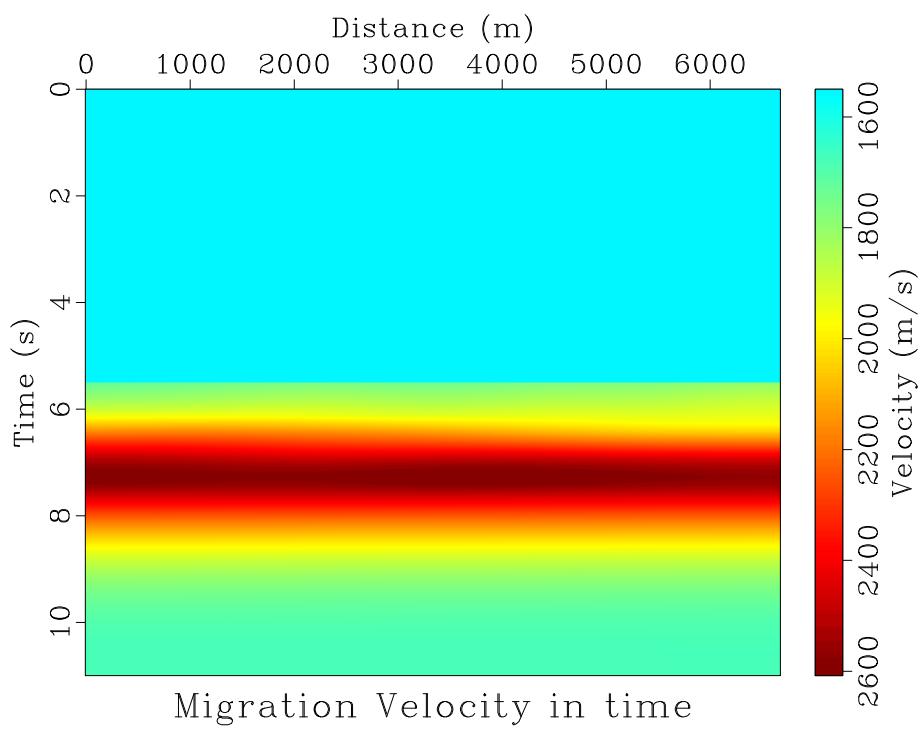


Figure 35: Migration velocity in time. [\[project/ vmigr\]](#)

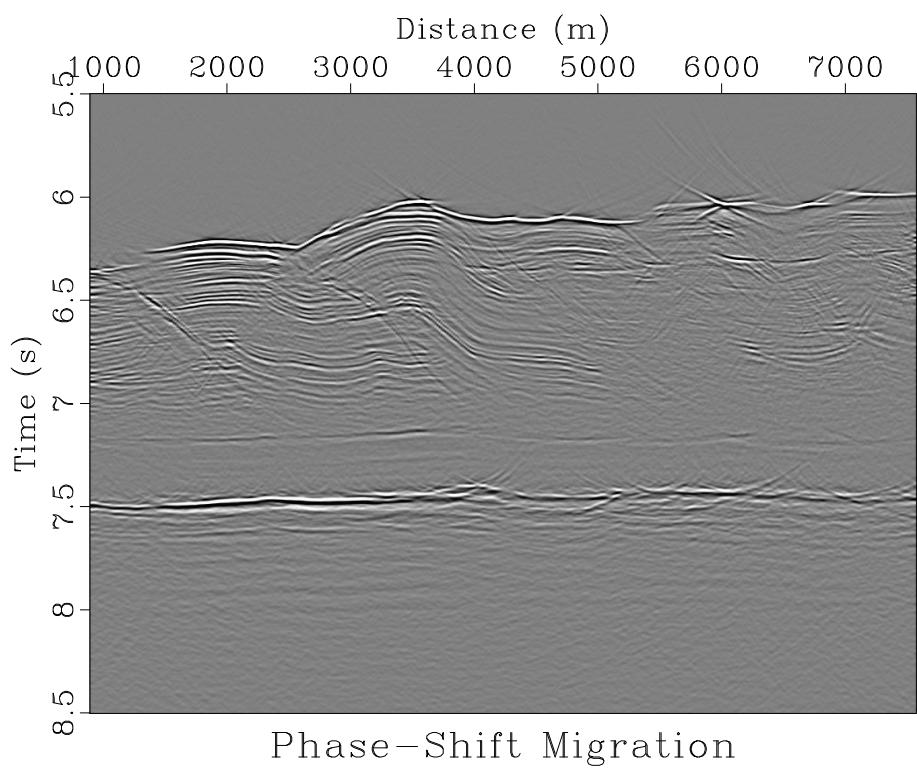


Figure 36: Gazdag migration. [project/ image](#)

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.

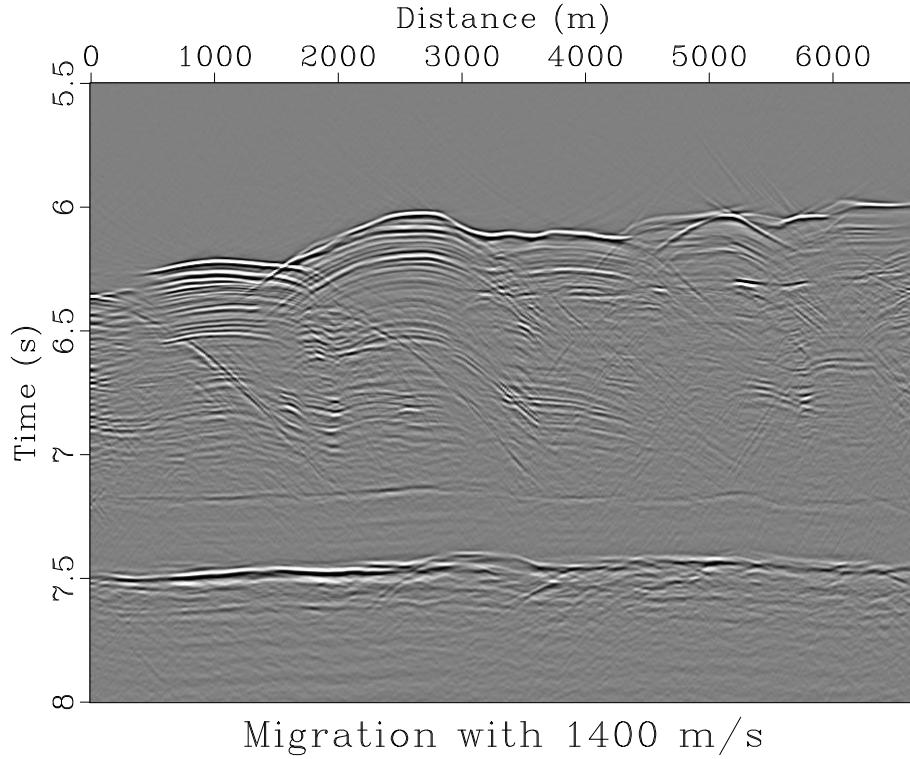


Figure 37: Stolt migration with velocity 1400 m/s. [project/ first](#)

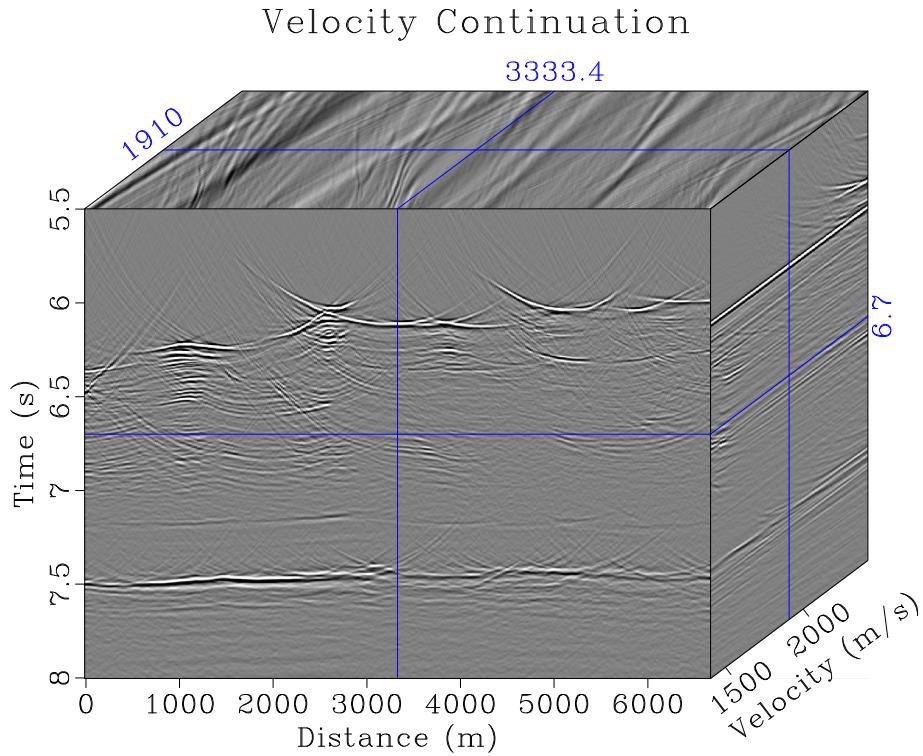


Figure 38: Velocity continuation. [project/ velcon](#)

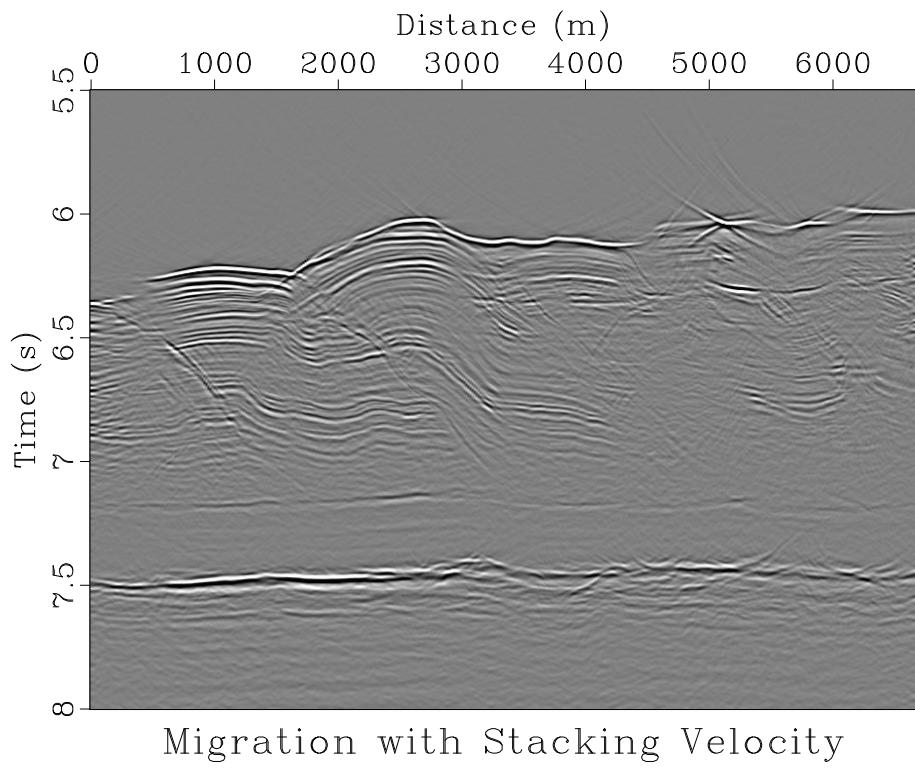


Figure 39: Migration with the stacking velocity. [project/ mstack](#)

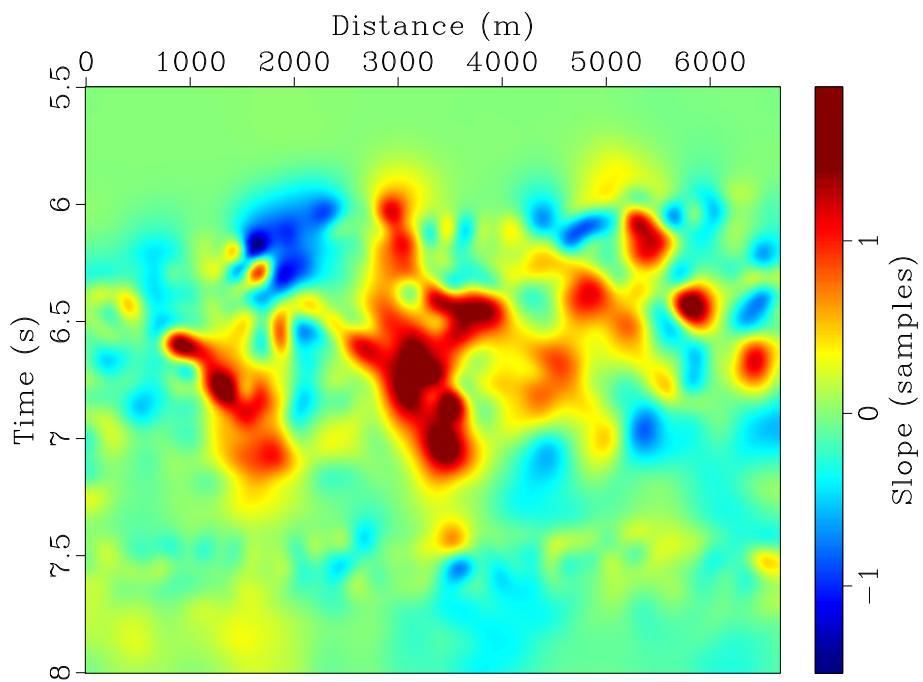


Figure 40: Estimated dips. [project/ dip](#)

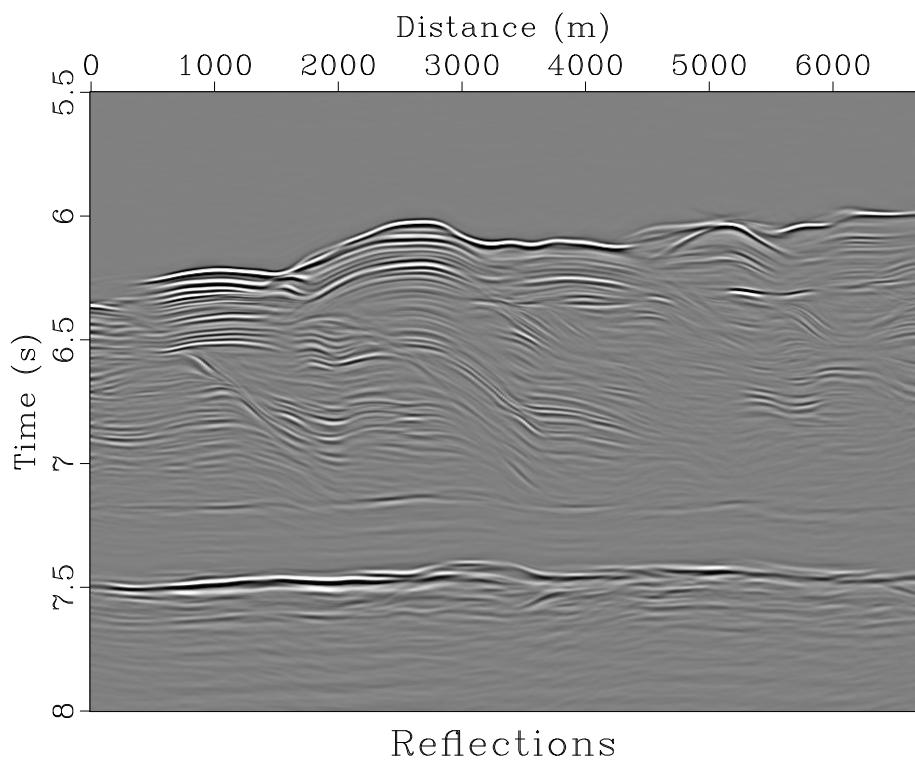


Figure 41: Reflections. [project/ refl](#)

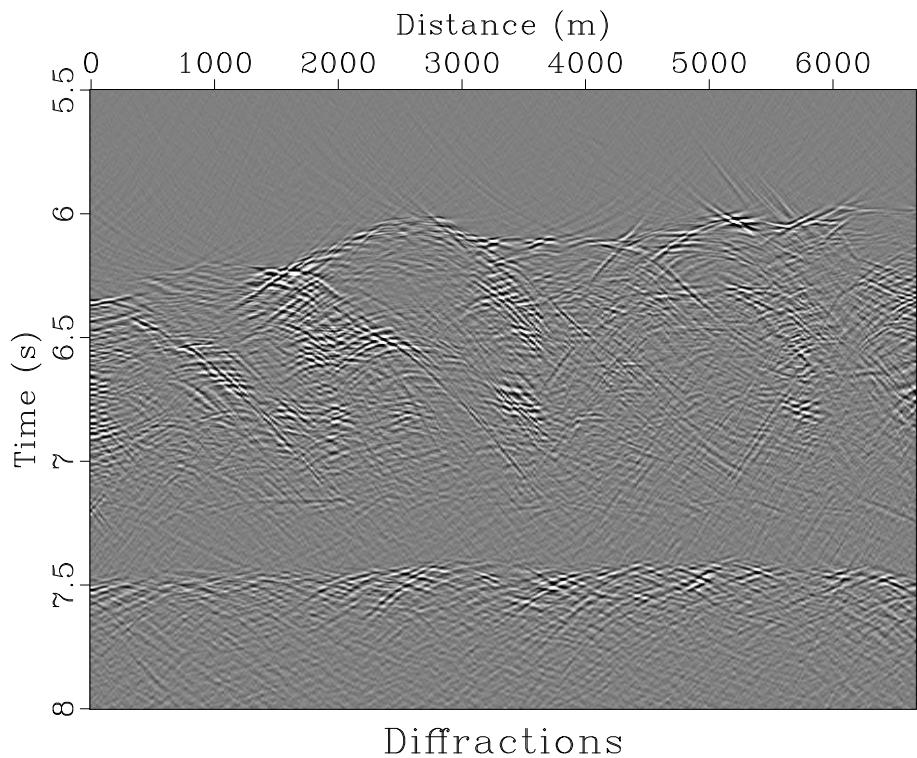


Figure 42: Diffractions. [project/ diff](#)

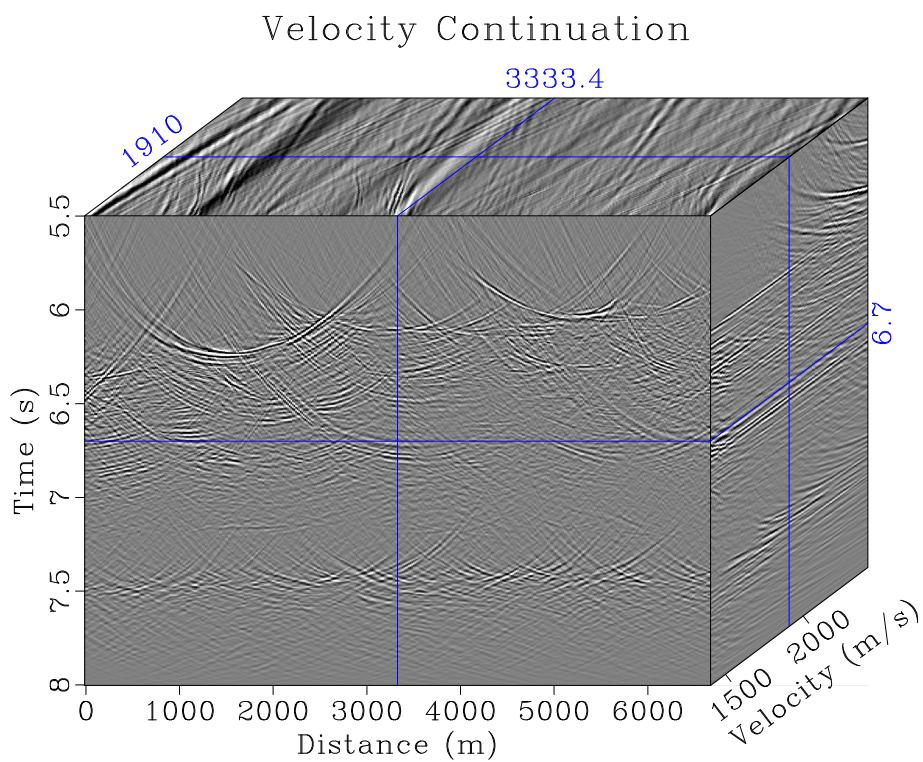


Figure 43: Velocity continuation with diffractions. [project/ velcond](#)

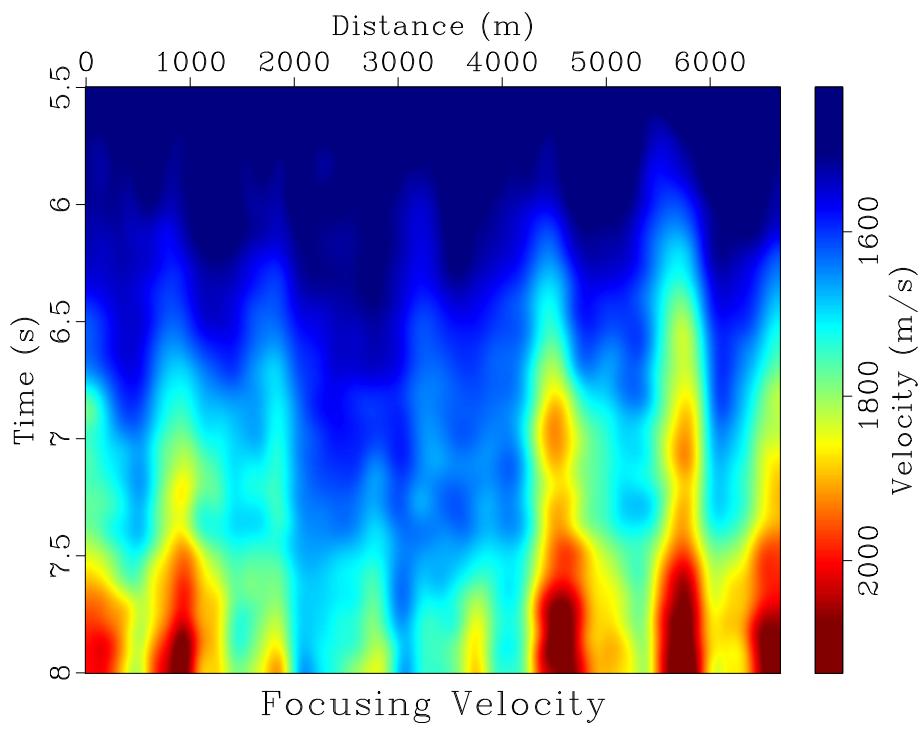


Figure 44: Focusing velocity. [project/ fpik](#)

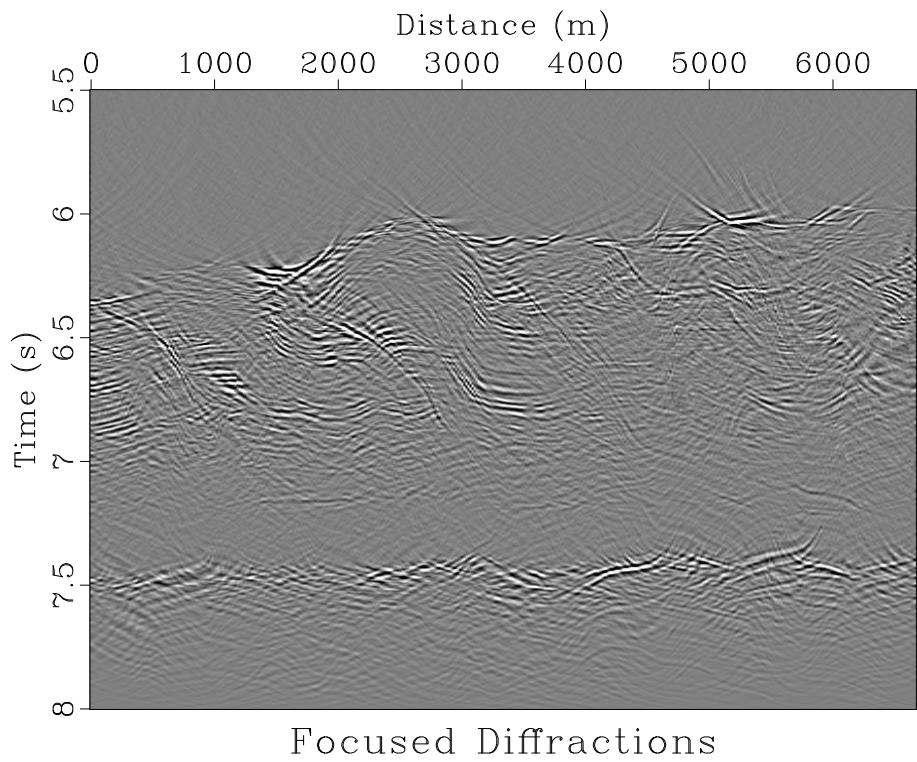


Figure 45: Focused diffractions. [project/ mdif](#)

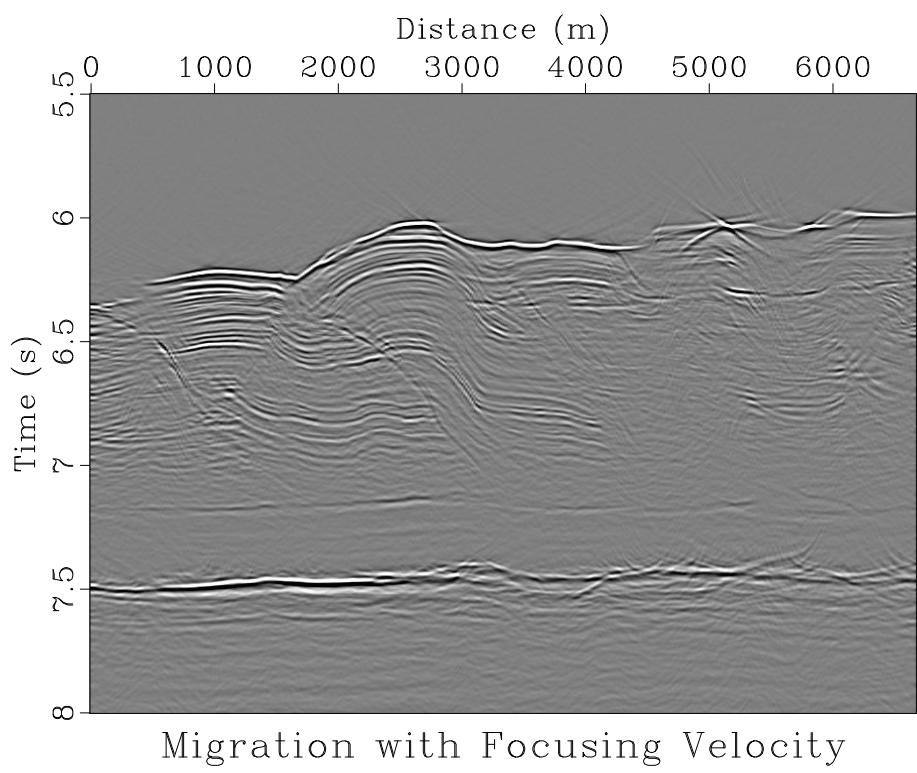


Figure 46: Migration with focusing velocity. [\[project/ mfpik\]](#)

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.

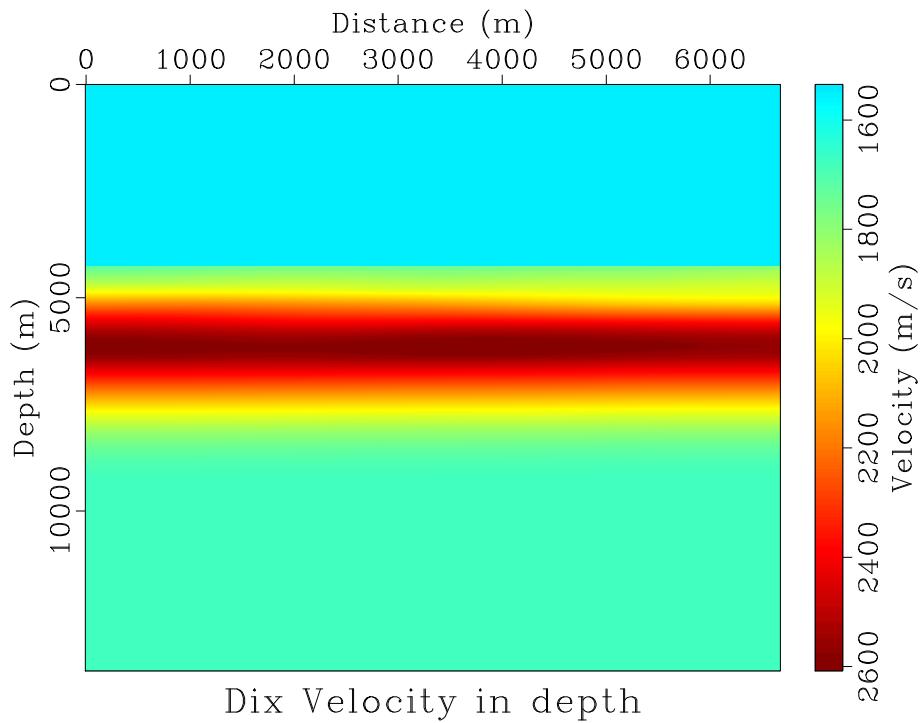


Figure 47: Dix velocity in depth. [project / vdixz](#)

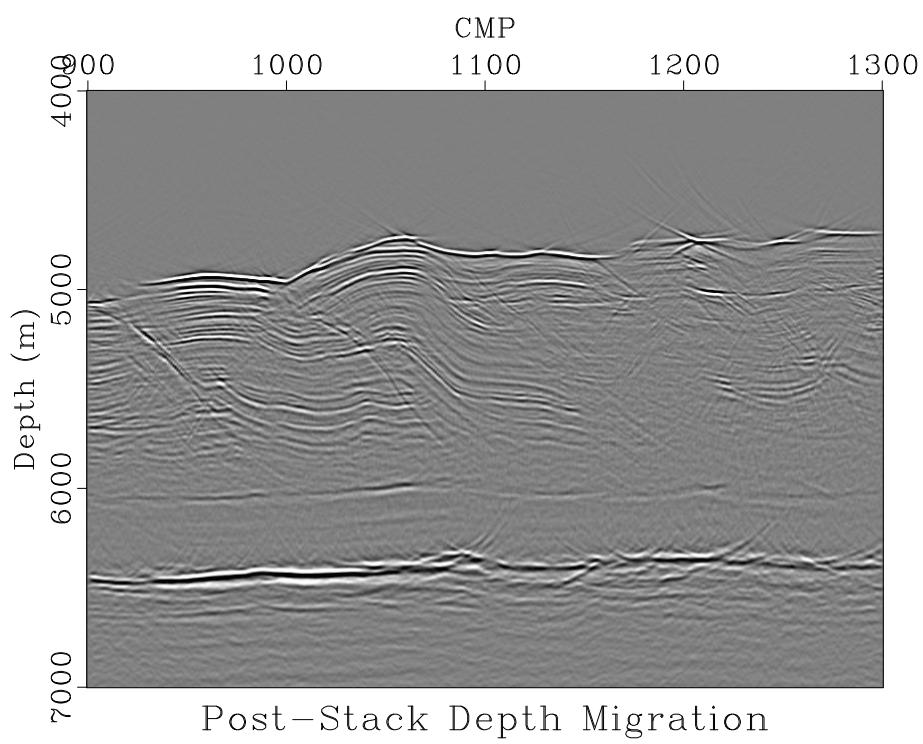


Figure 48: Reverse-time migration. [project/ rtm](#)

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.

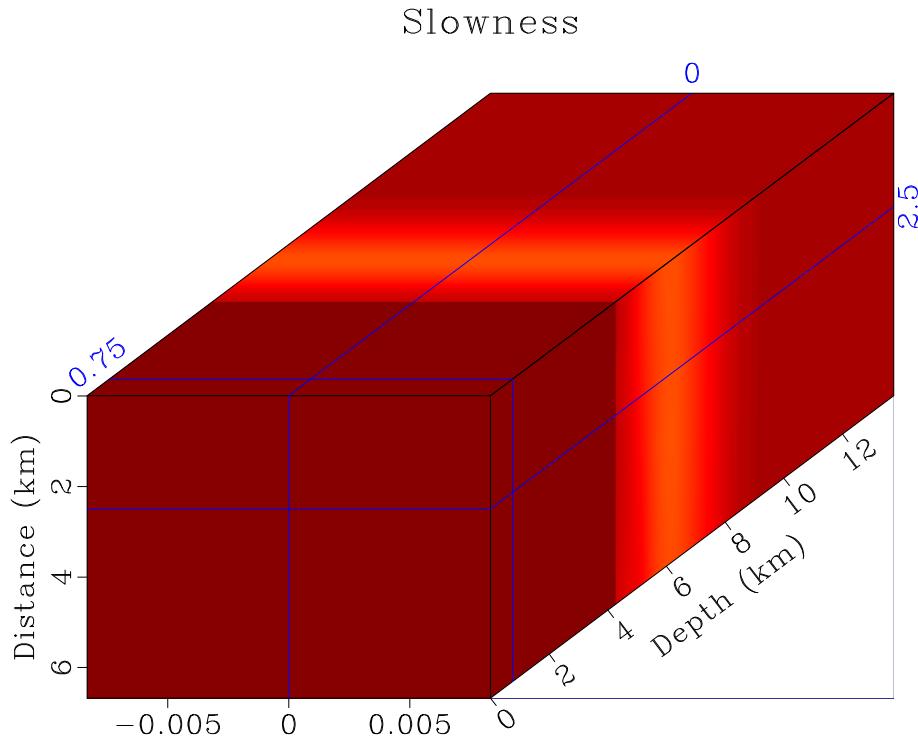


Figure 49: Slowness. [project/ slo](#)

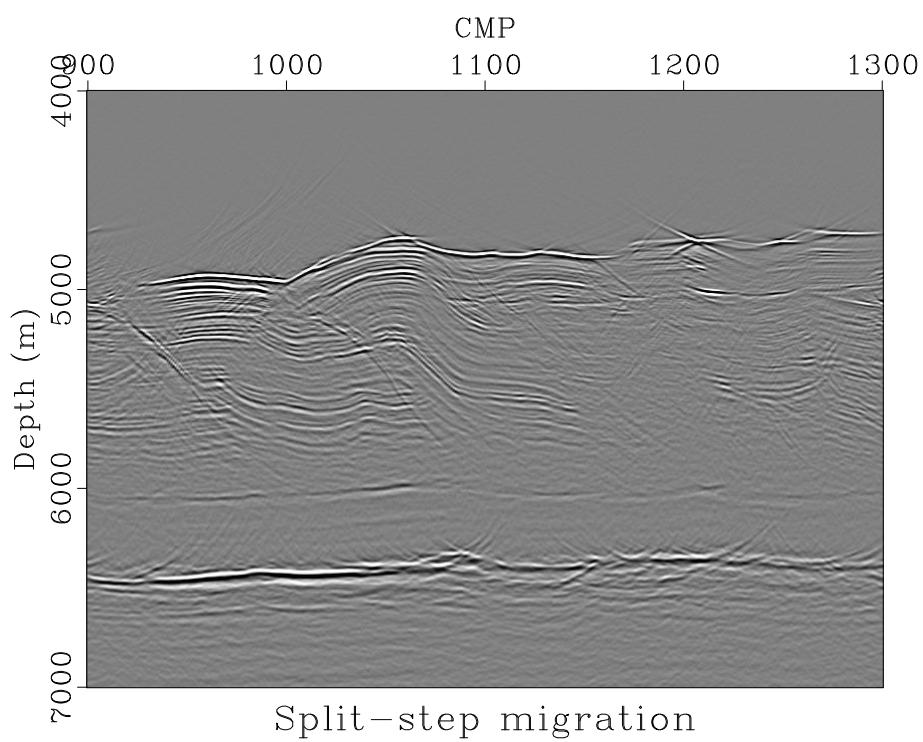


Figure 50: Split-step migration. [project/ mig](#)

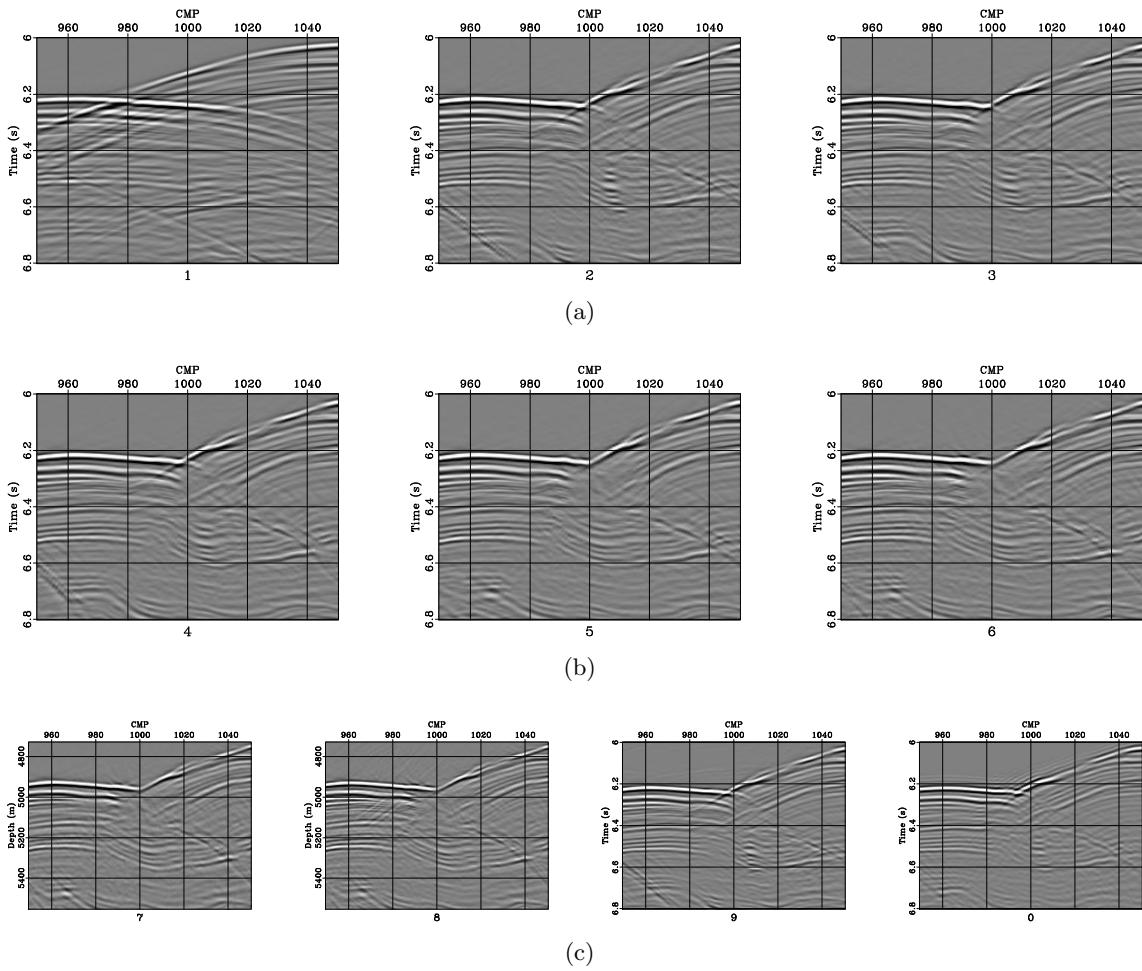


Figure 51: Migration comparison (a1) DMO stack (a2) Stolt migration with constant velocity 1500 m/s (a3) Stolt migration with variable velocity (b4) Migration with stacking velocity (b5) Migration with focusing velocity (b6) Gazdag migration (c7) Reverse-time migration (c8) Split-step migration (c9)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

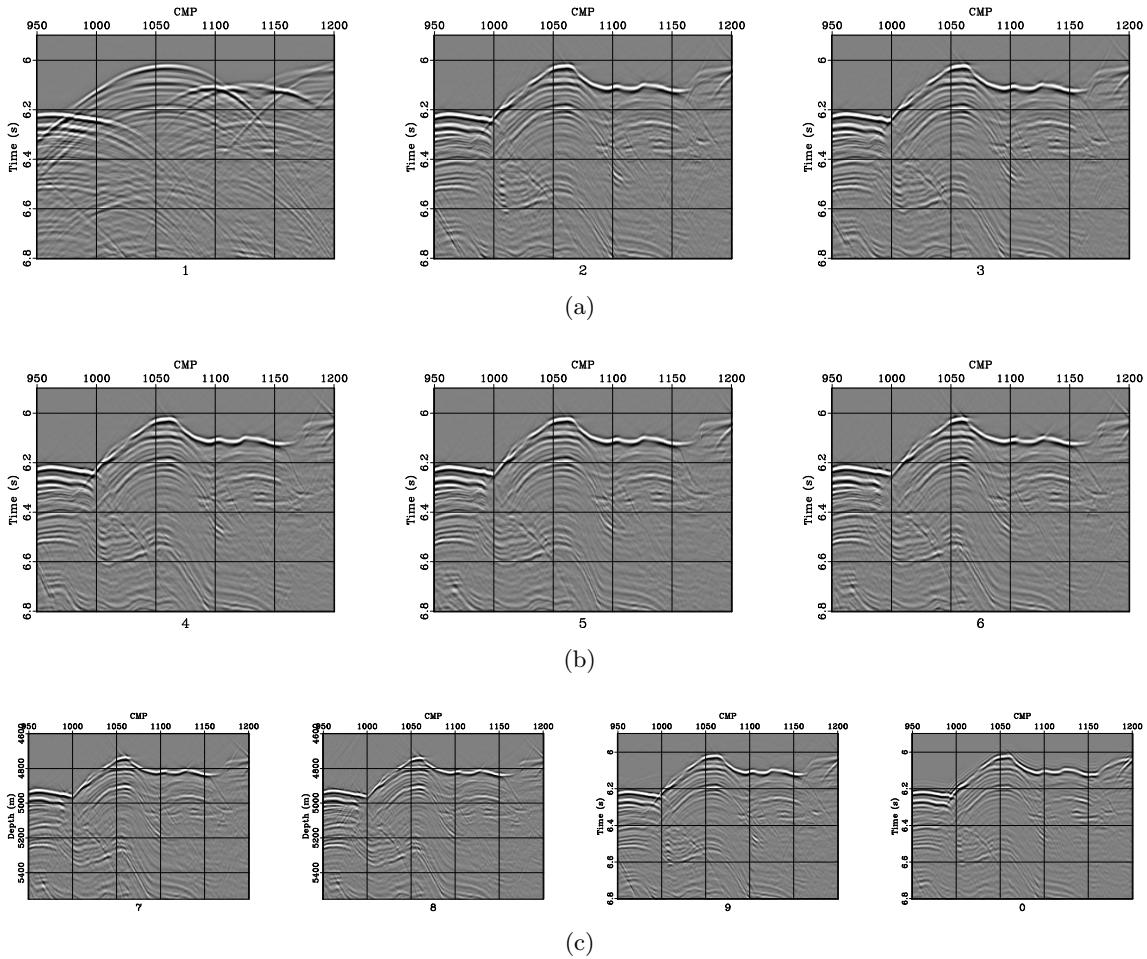


Figure 52: Migration comparison (a1) DMO stack (a2) Stolt migration with constant velocity 1500 m/s (a3) Stolt migration with variable velocity (b4) Migration with stacking velocity (b5) Migration with focusing velocity (b6) Gazdag migration (c7) Reverse-time migration (c8) Split-step migration (c9)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

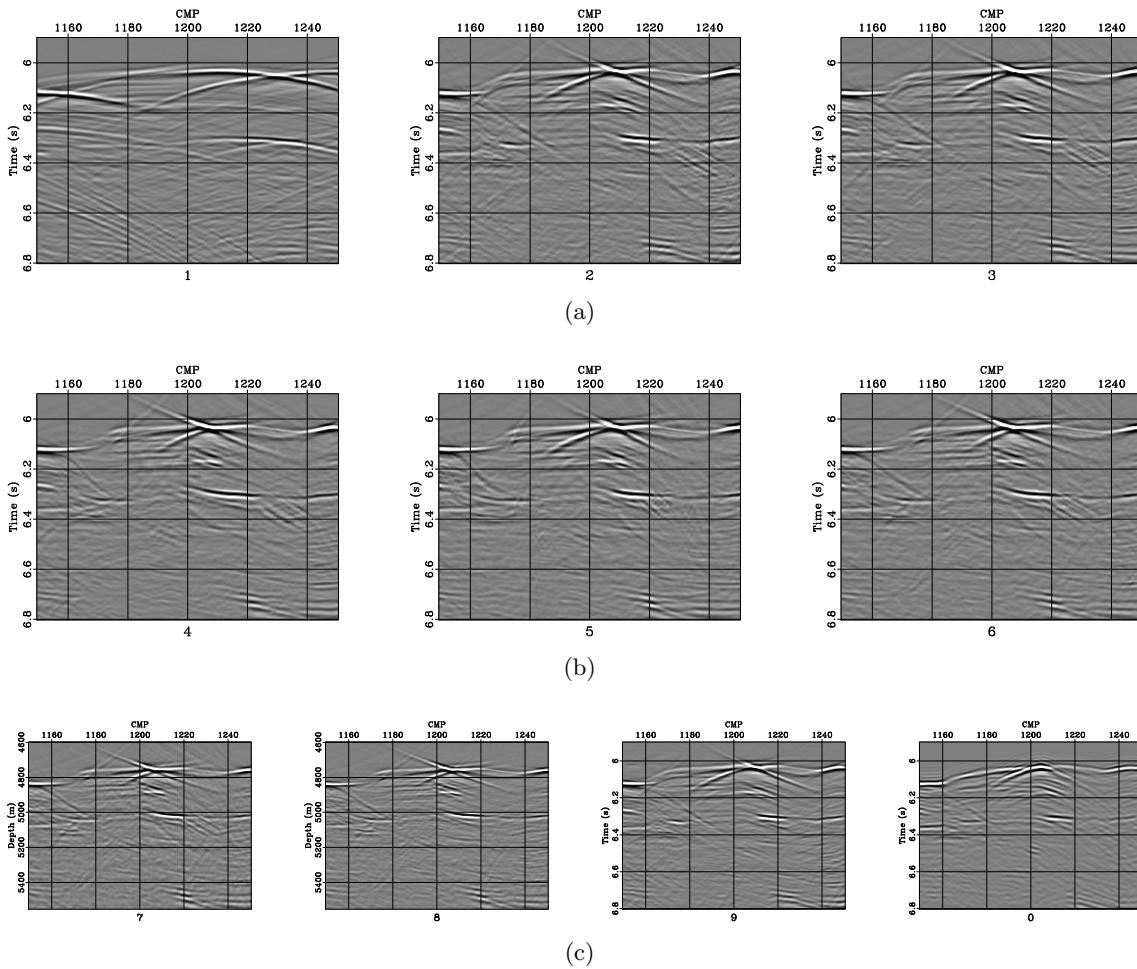


Figure 53: Migration comparison (a1) DMO stack (a2) Stolt migration with constant velocity 1500 m/s (a3) Stolt migration with variable velocity (b4) Migration with stacking velocity (b5) Migration with focusing velocity (b6) Gazdag migration (c7) Reverse-time migration (c8) Split-step migration (c9)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

INTERPRETATIONS

1. 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.
2. 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.
3. 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.
4. 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

```

1 from rsf.proj import *
2
3
4 def section(title ,label1='Time' ,unit1='s' ,min1=5.8,max1=8.0,extra=" "):
5     return ''
6     window min1=%g max1=%g |
7         grey title=%s"
8     label1=%s" unit1=%s" label2=Distance unit2=m %s
9     ''% (min1,max1,title ,label1 ,unit1 ,extra)
10
11 # Download data
12
13 Fetch( 'Nshots.su' , 'nankai' )
14
15 # Convert from SU to RSF
16
17 Flow( 'shots tshots' , 'Nshots.su' ,
18       '',
19       suread suxdr=y tfile=${TARGETS[1]}
20       '')
21
22 Result( 'spectra' , 'shots' ,
23       '',
24       spectra all=y | scale axis=1 |
25       graph title="Average Spectra" label1=Amplitude

```

```

26      ''')
27 Result( 'shots' ,
28         '',
29         window min1=5.5 max1=8.5 |
30         grey title="Raw shot gathers"
31         ''')
32
33 Fetch( 'Nstack.su' , 'nankai' )
34
35 Flow( 'stackd tstackd' , 'Nstack.su' ,
36         '',
37         suread suxdr=y tfile=${TARGETS[1]}
38         ''')
39 Flow( 'stack-win' , 'stackd' , 'window min1=5' )
40 Flow( 'stack-shot' , 'stack-win' ,
41         '',
42         window min2=900 max2=1300 | put label2=CMP
43         ''')
44 Plot( 'stackd' , 'window j1=2 j2=2 | grey title=Stack label2=CMP' )
45 Plot( 'stack-win' ,
46         '',
47         window j2=2 | grey title="Stack (Window)" label2=CMP
48         ''')
49 Plot( 'stack-shot' , 'stack-win' ,
50         '',
51         window min2=900 max2=1300 |
52         grey title="Part of stack corresponding to the shot file"
53         label2=CMP
54         ''')
55 Result( 'stack-shot' , 'stack-win' ,
56         '',
57         window min2=900 max2=1300 |
58         grey title="Part of stack corresponding to the shot file"
59         label2=CMP
60         ''')
61 Result( 'stackd' , 'stackd stack-win' , 'SideBySideAniso' )
62
63 ##### DC Removal
64
65 Flow( 'mean' , 'shots' ,
66         '',
67         stack axis=1 | spray axis=1 n=5500 o=0.0 d=0.002
68         ''')
69
70 Flow( 'shotsdc' , 'shots mean' , 'add scale=1,-1 ${SOURCES[1]} ')
71 Result( 'shotsdc' ,
72         '',
73         window min1=5.5 max1=8.5 |

```

```

74     grey title="Shot gathers after removing DC offset"
75     ,')
76
77 ##### Bandpass Filtering
78
79 Flow('shotsf','shotsdc','bandpass flo=10 fhi=125')
80 Result('shotsf',
81     ,,
82     window min1=5.5 max1=8.5 |
83     grey title="Shot gathers after filtering low and high frequencies"
84     ,')
85 Result('spectraf','shotsf',
86     ,,
87     spectra all=y | scale axis=1 |
88     graph title="Spectra after removing low and high frequencies"
89     label1=Amplitude
90     ,')
91
92 ##### Mask zero traces
93
94 Flow('mask0','shotsf','mul $SOURCE | stack axis=1 | mask min=1e-20')
95 Flow('shots0','shotsf mask0','headerwindow mask=${SOURCES[1]}')
96
97 # update a database
98 Flow('tshots0','tshots mask0','headerwindow mask=${SOURCES[1]}')
99
100 ##### Surface consistent
101
102 # Average trace amplitude
103 Flow('arms','shots0',
104     'mul $SOURCE | stack axis=1 | math output="log(input)"')
105 Result('arms','grey title=Log-Amplitude mean=y pclip=90')
106
107 # shot/offset indeces: fldr and tracf
108
109 Flow('indexshot','tshots0','window n1=1 f1=2')
110
111 Flow('offsets4index','tshots0',
112     ,,
113     headermath output=offset | dd type=float | window
114     ,')
115
116 Flow('offsetindex','offsets4index',
117     ,,
118     math output="abs(input) - 170" | dd type=int
119     ,')
120
121 # receiver/midpoint

```

```

122
123 Flow( 'midpoint' , 'tshots0' , 'window n1=1 f1=5' )
124
125 Flow( 'cmps4index' , 'tshots0' ,
126   '',
127     headermath output=cdp | dd type=float |
128     math output="input*16.667" | window
129   '')
130
131 Flow( 'recv' , 'cmps4index offsets4index' ,
132   '',
133     add scale=1,0.5 ${SOURCES[1]} |
134     math output="input - 13799" | dd type=int
135   '')
136
137 Flow( 'index' , 'indexshot offsetindex' ,
138   '',
139     cat axis=2 ${SOURCES[1]}
140   '')
141
142 Flow( 'extindex' , 'index midpoint' ,
143   '',
144     cat axis=2 ${SOURCES[1]}
145   '')
146
147 Flow( 'extindrecv' , 'extindex recv' ,
148   '',
149     cat axis=2 ${SOURCES[1]}
150   '')
151
152 def plot( title ):
153   return '',
154   spray axis=1 n=1 |
155   intbin head=${SOURCES[1]} yk=fldr rk=tracf | window |
156   grey title="%s" label2="Shot Number" unit2=
157   label1="Offset Number" unit1= scalebar=y
158   '' % ( title )
159
160 def plotb( title , bias=-5):
161   return '',
162   spray axis=1 n=1 |
163   intbin head=${SOURCES[1]} yk=fldr rk=tracf | window |
164   grey title="%s" label2="Shot Number" unit2=
165   label1="Offset Number" unit1= scalebar=y clip=3 bias=%g
166   '' % ( title , bias )
167
168 # Display in shot/offset coordinates
169 Flow( 'varms' , 'arms tshots0' ,

```

```

170      '',
171      spray axis=1 n=1 |
172      intbin head=${SOURCES[1]} yk=fldr xk=tracf | window
173      ''')
174 Result('varms', 'arms tshots0', plotb('Log-Amplitude'))
175
176 prog = Program('surface-consistent.c')
177 sc = str(prog[0])
178
179 # recv index
180
181 # get model dimensions
182 Flow('recvmodel',[ 'arms', 'extindrecv',sc ],
183       './${SOURCES[2]} index=${SOURCES[1]} verb=y')
184
185 # find a term
186 Flow('recvsc',[ 'arms', 'extindrecv',sc , 'recvmodel' ],
187       '',
188       conjgrad ./${SOURCES[2]} index=${SOURCES[1]}
189       mod=${SOURCES[3]} niter=150
190       ''')
191
192 # project to a data space
193 Flow('recvscarms',[ 'recvsc', 'extindrecv',sc ],
194       './${SOURCES[2]} index=${SOURCES[1]} adj=n')
195
196 Result('recvscarms', 'recvscarms tshots0',
197         plotb('Source, Offset, CDP, Recv S-C Log(A)'))
198
199 # compute difference
200 Flow('recvadiff', 'arms recvscarms', 'add scale=1,-1 ${SOURCES[1]}')
201
202 Result('recvadiff', 'recvadiff tshots0', plot('s,h,cdp,r difference'))
203
204 size=dict(sht=326, off=2266, rcv=7784, cmp=401)
205 f1 = 0
206 for case in ('sht', 'off', 'rcv', 'cmp'):
207     n1=size[case]
208
209     Result(case, 'recvsc',
210           '',
211           window n1=%d f1=%d | put o1=1 d1=1 |
212           graph title="%s Term"
213           label1="%s Number" unit1= label2=Amplitude unit2=
214           '' % (n1,f1,case.capitalize(),case.capitalize()))
215     f1 += n1
216
217 #### apply to traces to all times

```

```

218 Flow( 'ampl' , 'recvscarms' ,
219   'math output="exp(-input/2)" | spray axis=1 n=5500 d=0.002 o=0' )
220
221 Flow( 'shots-preproc' , 'shots0 ampl' , 'mul ${SOURCES[1]}')
222
223 Plot( 'shots-preproc' , 'shots-preproc' ,
224   ' ' ,
225   'window n2=100 |
226   grey min1=6.0 max1=8.0 title="Shots Preproc"
227   ' ')
228
229 Plot( 'shots-raw' , 'shots0' ,
230   ' ' ,
231   'window n2=100 |
232   grey min1=6.0 max1=8.0 title="Shots Raw"
233   ' ')
234
235 Result( 'shotsfc' , 'shots-raw shots-preproc' , 'SideBySideAniso' )
236
237 # Resample to 4 ms
238
239 Flow( 'subsamples' , 'shots-preproc' ,
240   ' ' ,
241   'bandpass fhi=125 | window j1=2
242   ' ')
243
244 Result( 'spectrasub' , 'subsamples' ,
245   'spectra all=y | graph title="Subsampled Spectra" ')
246
247 Result( 'spectra-check' , 'shots-preproc' ,
248   'spectra all=y | graph max1=160 title="Spectra Check" ')
249
250
251 # Extract shots
252 Flow( 'shots2' , 'subsamples tshots0' ,
253   ' ' ,
254   'intbin xk=tracf yk=fldr head=${SOURCES[1]}
255   ' ')
256 Result( 'shots2' ,
257   ' ' ,
258   'window min1=5.5 max1=9 | byte gainpanel=all |
259   transp plane=23 memsize=5000 |
260   grey3 frame1=300 frame2=100 frame3=50
261   point1=0.8 point2=0.8 clip=8.66
262   title="Shots" label3="Shots"
263   ' ')
264
265

```

```

266 # Create a mask to remove misfired shots
267
268 Flow( 'smask' , 'shots2' , 'mul $SOURCE | stack axis=1 | mask min=1e-20' )
269
270 Result( 'smask' ,
271   ' ' ,
272   dd type=float | stack axis=1 norm=n |
273   graph symbol=x title="Nankai Trough"
274   label2="Traces per Shot Gather"
275   label1="Shot Number" unit1=
276   ' ' )
277
278 Flow( 'offsets' , 'tshots0' ,
279   ' ' ,
280   window n1=1 f1=11 squeeze=n | dd type=float |
281   intbin head=$SOURCE xk=tracf yk=fldr
282   ' ' )
283
284
285
286 # Select one shot (fldr)
287
288 shot=1707
289
290 Flow( 'mask' , 'smask' , 'window n2=1 min2=%g' % shot )
291
292 Flow( 'shot' , 'shots2 mask' ,
293   ' ' ,
294   window n3=1 min3=%g squeeze=n |
295   headerwindow mask=${SOURCES[1]}
296   ' ' % shot )
297
298 Flow( 'offset' , 'offsets mask' ,
299   ' ' ,
300   window n3=1 min3=%g squeeze=n |
301   headerwindow mask=${SOURCES[1]}
302   ' ' % shot )
303
304 Result( 'shot' , 'shot offset' ,
305   ' ' ,
306   window min1=5.8 max1=8.5 |
307   wiggle xpos=${SOURCES[1]} yreverse=y transp=y poly=y
308   title="Shot 1707" label2=Offset
309   ' ' )
310
311 Plot( 'shot' ,
312   ' ' ,
313   window min1=5.8 max1=8.5 |

```

```

314     grey title="Selected Shot" clip=2
315     ''')
316
317
318 # Extract CMPs and apply t^2 gain
319
320 Flow( 'cmps maskcmp' , 'subsamples tshots0' ,
321     '',
322     intbin head=$SOURCES[1] mask=$TARGETS[1]
323     xk=tracf yk=cdp           |
324     pow pow1=2
325     ''')
326
327 Result( 'cmps' ,
328     '',
329     window min1=5.8 max1=8.5 |
330     byte gainpanel=all | transp plane=23 memsize=5000 |
331     grey3 frame1=200 frame2=150 frame3=50
332     title="CMPs" label2="CMP Number"
333     flat=n point1=0.8 point2=0.8
334     ''')
335
336 Flow( 'cmask' , 'cmps' , 'mul $SOURCE | stack axis=1 | mask min=1e-20' )
337
338 Result( 'cmask' ,
339     '',
340     dd type=float |
341     stack axis=1 norm=n |
342     graph symbol=x title="Nankai Trough"
343     label2="Traces per CMP Gather"
344     label1="CMP Gather Number" unit1=
345     ''')
346
347 Flow( 'offs' , 'tshots0' ,
348     '',
349     window n1=1 f1=11 squeeze=n | dd type=float |
350     intbin xk=tracf yk=cdp head=$SOURCE
351     ''')
352
353 # Examine one CMP gather
354
355 Flow( 'mask1' , 'cmask' , 'window n2=1 min2=1280' )
356
357 Flow( 'cmp1' , 'cmps mask1' ,
358     '',
359     window n3=1 min3=1280 | headerwindow mask=$SOURCES[1]
360     ''')
361

```

```

362 Flow( 'off' , 'offs mask1' ,
363   '',
364   'window n3=1 min3=1280 squeeze=n | headerwindow mask=${SOURCES[1]}'
365   '')
366
367 Result( 'cmp1' , 'cmp1 off' ,
368   '',
369   'window min1=5.8 max1=8.5 |
370   wiggle xpos=${SOURCES[1]} title="CMP 1280"
371   yreverse=y transp=y poly=y label2=Offset unit2=m
372   wherexlabel=t wheretitle=b
373   ')
374
375 Plot( 'cmp1' , 'cmp1 off' ,
376   '',
377   'window min1=5.8 max1=8.5 |
378   wiggle xpos=${SOURCES[1]} title="CMP 1280"
379   yreverse=y transp=y poly=y label2=Offset unit2=m
380   wherexlabel=t wheretitle=b
381   ')
382
383 # Velocity analysis and NMO
384
385 Flow( 'vscan' , 'cmp1 off mask1' ,
386   '',
387   'vscan half=n offset=${SOURCES[1]}
388   v0=1400 nv=101 dv=10 semblance=y
389   ')
390
391 Plot( 'vscan' ,
392   '',
393   'window min1=5.8 max1=8.5 |
394   grey color=j allpos=y title="Velocity Scan" unit2=m/s
395   ')
396
397 Flow( 'pick' , 'vscan' ,
398   '',
399   'mutter inner=y half=n t0=5 x0=1400 v0=75 |
400   pick v0=1500 rect1=25
401   ')
402
403 Plot( 'pick' ,
404   '',
405   'window min1=5.8 max1=8.5 |
406   graph transp=y yreverse=y plotcol=7 plotfat=3
407   pad=n min2=1400 max2=2400 wanttitle=n wantaxis=n
408   ')
409

```

```

410 Plot( 'vscamp' , 'vscan pick' , 'Overlay' )
411
412 Flow( 'nmo' , 'cmp1 off mask1 pick' ,
413      ' ' ,
414      nmo half=n offset=${SOURCES[1]}
415      velocity=${SOURCES[3]}
416      ' ' )
417
418 Result( 'nmo' , 'window min1=5.8 max1=8.5 | grey title="Normal Moveout" ' )
419
420 Plot( 'cmpg' , 'cmp1' ,
421      ' ' ,
422      window min1=5.8 max1=8.5 |
423      grey title="CMP 1280" labelsz=12 titlesz=18
424      ' ' )
425
426 Plot( 'nmog' , 'nmo' ,
427      ' ' ,
428      window min1=5.8 max1=8.5 |
429      grey title="Normal Moveout" labelsz=12 titlesz=18
430      ' ' )
431
432 Result( 'nmo1' , 'cmpg nmog' , 'SideBySideAniso' )
433
434 Result( 'vscan' , 'cmp1 vscamp' , 'SideBySideAniso' )
435
436 # Apply to all CMPs
437
438 Flow( 'cmask2' , 'cmask' , 'transp plane=23 | transp plane=21' )
439
440 Flow( 'vscans' , 'cmps offs cmask2' ,
441      ' ' ,
442      vscan half=n offset=${SOURCES[1]} mask=${SOURCES[2]}
443      v0=1400 nv=101 dv=10 semblance=y nb=5
444      ' ' , split=[3 , 'omp' ] )
445
446 Flow( 'picks' , 'vscans' ,
447      ' ' ,
448      mutter inner=y half=n t0=5 x0=1400 v0=75 |
449      pick v0=1500 rect1=25 rect2=10
450      ' ' )
451
452 Result( 'picks' ,
453      ' ' ,
454      window | grey color=j mean=y scalebar=y title="NMO Velocity"
455      label2=CMP barreverse=y barlabel=Velocity barunit=m/s
456      ' ' )
457

```

```

458 Flow( 'nmos' , 'cmps  offs  cmask  picks' ,
459   '   ,
460   nmo  half=n  offset=${SOURCES[1]}  mask=${SOURCES[2]}
461   velocity=${SOURCES[3]}
462   '   )
463
464 Result( 'nmos' ,
465   '   ,
466   window  min1=5.5  max1=8.5  |
467   byte  gainpanel=all  |  transp  plane=23  memsize=5000  |
468   grey3  frame1=400  frame2=50  frame3=50
469   title="NMO"  label2="CMP Number"
470   flat=n  point1=0.8  point2=0.8
471   '   )
472
473 Flow( 'stack' , 'nmos' , 'stack' )
474
475 Result( 'stack' ,
476   '   ,
477   window  min1=5.5  max1=8.5  |
478   grey  title="NMO Stack"  label2="CMP Number"
479   '   )
480
481 Plot( 'stack' ,
482   '   ,
483   window  min1=5.5  max1=8.5  |
484   grey  title="NMO Stack"  label2="CMP Number"
485   '   )
486
487 # Try DMO
488
489 nv=60
490
491 Flow( 'stacks' , 'cmps  offs  maskcmp' ,
492   '   ,
493   stacks  half=n  v0=1400  nv=%g  dv=20
494   offset=${SOURCES[1]}  mask=${SOURCES[2]}
495   '  %nv ,  split=[3 , 'omp' ] )
496
497 Flow( 'stackst' , 'stacks' , 'costaper  nw3=20' )
498
499 Result( 'stacks' , 'stackst' ,
500   '   ,
501   window  min1=5.5  max1=8.5  |  byte  gainpanel=all  |
502   transp  plane=23  memsize=5000  |
503   grey3  frame1=200  frame2=100  frame3=50  point1=0.8  point2=0.8
504   title="Constant-Velocity Stacks"  label3=Velocity  unit3=m/s
505   label2="CMP Number"
```

```

506      ''')
507
508 # Apply double Fourier transform (cosine transform)
509 Flow( 'cosft3' , 'stackst' ,
510       '',
511       put d3=16.667 o3=0 label2=Distance unit2=m |
512       cosft sign1=1 sign3=1
513       ''')
514
515 # Transpose f-v-k to v-f-k
516 Flow( 'transp' , 'cosft3' , 'transp' )
517
518 # Fowler DMO: mapping velocities
519
520 Flow( 'map' , 'transp' ,
521       '',
522       math output="x1/sqrt(1+0.25*x3*x3*x1*x1/(x2*x2))" |
523       cut n2=1
524       ''')
525 Result( 'map' ,
526         '',
527         window j3=4 |
528         byte gainpanel=all allpos=y bar=bar.rsf |
529         grey3 title="Fowler Map" label1=Velocity
530         unit1=m/s label3=Wavenumber barlabel=Velocity barunit=m/s
531         frame1=20 frame2=500 frame3=20 color=x scalebar=y
532         ''')
533
534 Flow( 'fowler' , 'transp map' , 'iwarp warp=${SOURCES[1]} | transp' )
535
536 # Inverse Fourier transform
537
538 Flow( 'dmo' , 'fowler' , 'cosft sign1=-1 sign3=-1' )
539
540 Result( 'dmo' ,
541         '',
542         put d3=1 o3=900 unit2=' label2=Trace | window min1=5.5 max1=8.5 |
543         byte gainpanel=all | transp plane=23 memsize=5000 |
544         grey3 frame1=200 frame2=100 frame3=50 point1=0.8 point2=0.8
545         title="Constant-Velocity DMO Stacks"
546         label3=Velocity unit3=m/s
547         ''')
548
549 # Compute envelope for picking
550
551 Flow( 'envelope' , 'dmo' , 'envelope | scale axis=2' )
552
553

```

```

554
555 # Mute and Pick velocity
556
557 Flow( 'vpick' , 'envelope' ,
558   '',
559     mutter v0=130 x0=1300 t0=4.0 half=n inner=n |
560     mutter x0=1400 v0=20 t0=5.0 half=n inner=y |
561     mutter v0=2500 x0=1400 t0=5.8 half=n inner=n |
562     mutter v0=500 x0=1400 t0=7.0 half=n inner=y |
563     pick rect1=80 rect2=20 vel0=1400
564   '')
565
566 Result( 'vpick' ,
567   '',
568     put d2=1 o2=900 unit2= label2="CMP Number" |
569     grey mean=y color=j scalebar=y
570     barreverse=y barunit=m/s barlabel=Velocity
571     title="Picked Migration Velocity" label2="CMP Number" unit2=
572   '')
573
574 # Take a slice
575
576 Flow( 'slice' , 'dmo vpick' ,
577   '',
578     slice pick=${SOURCES[1]} |
579     put d2=1 o2=900 unit2= label2="CMP Number"
580   '')
581
582 Result( 'slice' ,
583   '',
584     window min1=5.5 max1=8.5 |
585     grey title="DMO Stack"
586   '')
587
588 Plot( 'slice' ,
589   '',
590     window min1=5.5 max1=8.5 |
591     grey title="DMO Stack" label2="CMP Number"
592   '')
593
594 # Check one CMP location
595
596 p = 380
597
598 min1=5.8
599 max1=8.5
600
601 Flow( 'before' , 'stackst' ,

```

```

602      '',
603      window n3=1 f3=%d min1=%g max1=%g | envelope
604      ''' % (p, min1, max1))
605 Flow( 'after' , 'envelope' ,
606      '''window n3=1 f3=%d min1=%g max1=%g
607      ''' % (p, min1, max1))

608
609 for case in ('before' , 'after'):
610     Plot(case ,
611           '',
612           grey color=j allpos=y title="%s DMO"
613           label2=Velocity unit2=m/s
614           ''' % case.capitalize())

615
616 Flow( 'vpick1' , 'vpick' ,
617       '',
618       window n2=1 f2=%d min1=%g max1=%g
619       '''%(p, min1, max1))

620
621 Plot('vpick1' ,
622       '',
623       graph yreverse=y transp=y plotcol=7 plotfat=7
624       pad=n min2=%g max2=%g wantaxis=n wanttitle=n
625       ''' % (1400,2600))

626
627 Plot('before2' , 'before pick' , 'Overlay')
628 Plot('after2' , 'after vpick1' , 'Overlay')
629 Result('envelope' , 'before2 after2' , 'SideBySideAniso')

630 #####
631 # Zoom an interesting area
632 #####
633 min1,max1=5.8,7
634 min2,max2=1100,1300

635
636
637 Flow('box.asc' ,None ,
638       '',
639       echo "%s n1=2 n2=5 data_format=ascii_float in=$TARGET
640       ''' % ' '.join(map(str ,(min1,min2,max1,min2,
641                               max1,max2,min1,max2,min1,min2))))
642 Plot('box' , 'box.asc' ,
643       '',
644       dd form=native type=complex | window |
645       graph transp=y yreverse=y min1=0 max1=4 min2=0 max2=26.7625
646       wanttitle=n plotfat=5 plotcol=6 wantaxis=n
647       ''')
648
649 for i in range (2):

```

```

650 case=( 'slice' , 'stack')[ i ]
651 zoomd = case + '-zoomd'
652 Flow(zoomd, case,
653      '',
654      window min1=%g max1=%g min2=%g max2=%g
655      ' ' ' % (min1,max1,min2,max2))
656 Plot(zoomd, 'grey title=%s grid=y gridcol=5 label2=CMP' % ('ab'[i]))
657 Result('zoomd', 'slice-zoomd stack-zoomd',
658        'SideBySideIso')
659
660 # Stolt Migration
661
662 # 2D cosine transform
663
664 Flow('cosft', 'slice',
665       '',
666       put d2=16.667 label2=Distance unit2=m |
667       cosft sign1=1 sign2=1 |
668       put label2=Wavenumber
669       ' ' ')
670 Result('cosft',
671       '',
672       window max1=90 |
673       grey title="Cosine Transform" pclip=95 label1=Frequency
674       ' ' ')
675
676 # Stolt migration with constant velocity
677
678 Flow('map2', 'cosft',
679       '',
680       math output="sqrt(x1*x1+%g*x2*x2)"
681       ' ' %(562500))
682
683 Result('map2',
684       '',
685       grey color=x allpos=y scalebar=y
686       title="Stolt Map" barlabel=Frequency barunit=Hz label1=Frequency
687       ' ' ')
688
689 Flow('cosft2', 'cosft map2', 'iwarped warp=${SOURCES[1]} inv=n')
690
691 Result('cosft2',
692       '',
693       window max1=90 |
694       grey title="Cosine Transform Warped" pclip=95 label1=Frequency
695       ' ' ')
696
697 Flow('mig2', 'cosft2',

```

```

698      '',
699      cosft sign1=-1 sign2=-1 |
700
701      put d2=1 unit2=''
702      ''')
703
704 Result('mig2',
705      '',
706      window min1=5.5 max1=8.5 |
707      grey title="Stolt Migration with 1500 m/s"
708      label2="CMP Number"
709      ''')
710
711 # Ensemble of Stolt migrations with different velocities
712
713 Flow('spray','cosft',
714      'spray axis=3 n=120 o=1500 d=8 label=Velocity unit=m/s')
715
716 Flow('map1','spray','math output="sqrt(x1*x1+0.25*x3*x3*x2*x2)" ')
717
718 Result('map1',
719      '',
720      byte gainpanel=all allpos=y |
721      grey3 title="Stolt Ensemble Map"
722      frame1=400 frame2=50 frame3=50 color=x
723      ''')
724
725 Flow('migd','spray map1',
726      '',
727      iwarp warp=${SOURCES[1]} inv=n |
728      cosft sign1=-1 sign2=-1
729      ''')
730
731 Flow('migt','migd','transp plane=23 memsize=5000')
732
733 Plot('migd',
734      '',
735      window min1=5.5 max1=8.5 |
736      grey title="Ensemble of Stolt Migrations"
737      ''',view=1)
738
739 # Migration velocity increasing with time
740
741 Flow('vmig1','slice',
742      '',
743      pad beg1=1250 | math output="1500" |
744      window n1=1250 | put o1=0
745      ''')

```

```

746 Flow( 'vmig2' , 'slice' ,
747   ' ,
748   'window min1=5 | put o1=0 |
749   math output="1500+20*x1*x1" | put o1=5
750   ' ')
751
752 Flow( 'vmig' , 'vmig1 vmig2' , 'cat ${SOURCES[1:2]} axis=1' )
753
754 Result( 'vmig' ,
755   ' ,
756   'window min1=5.5 max1=8.5 |
757   grey color=j mean=y barreverse=y title="Migration Velocity"
758   scalebar=y barlabel=Velocity barunit=m/s label2="CMP Number"
759   unit2=
760   ' ')
761
762 # Slice through the ensemble of migrations
763
764 Flow( 'slice1' , 'migt vmig' ,
765   ' ,
766   'slice pick=${SOURCES[1]} | put d2=1 o2=900
767   ' ')
768
769 Result( 'migd' , 'slice1' ,
770   ' ,
771   'window min1=5.5 max1=8.5 |
772   grey label2="CMP Number"
773   unit2= title="Stolt Migration with Variable Velocity"
774   ' ')
775
776 # Dix conversion to interval velocity
777 Flow( 'semb' , 'vscans picks' , 'slice pick=${SOURCES[1]}' )
778 Flow( 'vdix' , 'picks semb' ,
779   ' ,
780   'dix weight=${SOURCES[1]} rect1=50 rect2=10
781   ' ')
782
783 # Velocity continuation
784 Flow( 'first' , 'slice' ,
785   ' ,
786   'put o2=0 d2=16.667 label2=Distance unit2=m o3=0 |
787   cosft sign2=1 |
788   stolt vel=1400
789   ' ')
790
791 Flow( 'mig0' , 'first' ,
792   ' ,
793   'window min1=5.5 max1=8.0 |

```

```

794     cosft sign2=-1
795     ''')
796
797 Result( 'first' , 'mig0' ,
798     '',
799     grey title="Migration with 1400 m/s"
800     ''')
801
802 Flow( 'velcon' , 'first' ,
803     '',
804     vczo nv=101 dv=10 v0=1400 verb=y |
805     window min1=5.5 max1=8.0 |
806     transp plane=23 memsize=1000 |
807     cosft sign2=-1
808     ''')
809 Result( 'velcon' ,
810     '',
811     byte gainpanel=a |
812     grey3 frame1=300 frame2=200 frame3=50
813     point1=0.8 point2=0.8 flat=n unit3=m/s
814     title="Velocity Continuation"
815     ''')
816 Plot( 'movie' , 'velcon' , 'grey title="velocity continuation"' , view=1)
817
818 # Migration with the stacking velocity
819 Flow( 'mpick' , 'picks' , 'window min1=5.5 max1=8.0' )
820 Flow( 'mstack' , 'velcon mpick' ,
821     '',
822     transp plane=23 memsize=1000 |
823     slice pick=${SOURCES[1]}
824     ''')
825 Result( 'mstack' ,
826     'grey title="Migration with Stacking Velocity"' )
827
828 # Separating diffractions
829
830 Flow( 'dip' , 'mig0' , 'dip rect1=20 rect2=10 order=2' )
831 Result( 'dip' ,
832     '',
833     grey color=j scalebar=y wanttitle=n
834     barlabel=Slope barunit=samples
835     ''')
836
837 Flow( 'refl' , 'mig0 dip' ,
838     '',
839     pwspray dip=${SOURCES[1]} ns=5 order=2 reduce=stack
840     ''')
841 Result( 'refl' ,

```

```

842      '''
843      window min1=5.5 max1=8 | grey title="Reflections"
844      ''')
845
846 Flow('diff','mig0 refl','add scale=1,-1 ${SOURCES[1]}')
847 Result('diff','window min1=5.5 max1=8 | grey title="Diffractions"')
848
849 Flow('stolt','diff',
850      ''',
851      cosft sign2=1 |
852      pad beg1=1250
853      ''')
854
855 Flow('velcond','stolt',
856      ''',
857      vczo nv=101 dv=10 v0=1400 verb=y pad2=3000 |
858      window min1=5.5 max1=8 |
859      transp plane=23 memsize=1000 |
860      cosft sign2=-1
861      ''')
862 Plot('velcond','grey title="Velocity Continuation",view=1)
863 Result('velcond',
864      ''',
865      byte gainpanel=a |
866      grey3 frame1=300 frame2=200 frame3=50
867      point1=0.8 point2=0.8 flat=n unit3=m/s
868      title="Velocity Continuation"
869      ''')
870
871 Flow('focus','velcond',
872      ''',
873      transp plane=23 |
874      pad beg1=1250
875      focus rect1=40 rect3=20 |
876      window min1=5.5 |
877      math output="1/abs(input)" |
878      clip clip=8 |
879      scale axis=2
880      ''')
881
882 Flow('fpik','focus','pick v0=1400 rect1=25 rect2=10')
883
884 Result('fpik',
885      ''',
886      window | grey color=j allpos=y bias=1500
887      scalebar=y title="Focusing Velocity"
888      barreverse=y barlabel=Velocity barunit=m/s
889      ''')

```

```

890
891 Flow( 'mdif' , 'velcond fpik' ,
892      '',''
893      transp plane=23 memsize=1000 |
894      slice pick=${SOURCES[1]}
895      '','')
896 Result( 'mdif' ,
897          'grey title="Focused Diffractions" ')
898
899 Flow( 'mfpik' , 'velcon fpik' ,
900      '',''
901      transp plane=23 memsize=1000 |
902      slice pick=${SOURCES[1]}
903      '','')
904 Result( 'mfpik' ,
905          'grey title="Migration with Focusing Velocity" ')
906
907 # RTM
908 # Dix conversion to interval velocity
909 Flow( 'first1' , 'slice' ,
910      '',''
911      put o2=0 d2=16.667 label2=Distance unit2=m o3=0 |
912      cosft sign2=1 |
913      stolt vel=1550
914      '','')
915
916 Flow( 'mig1' , 'first1' ,
917      '',''
918      cosft sign2=-1
919      '','')
920
921 Result( 'mig1' ,
922          '',''
923          window min1=5.5 max1=8.5 |
924          grey title="Migration with 1550 m/s"
925          '','')
926 Flow( 'weight' , 'envelope vpick' , 'slice pick=${SOURCES[1]} ')
927 Flow( 'vdix2' , 'vpick weight' ,
928          'dix rect1=20 rect2=50 weight=${SOURCES[1]} ')
929 Result( 'vdix2' ,
930          '',''
931
932          grey color=j mean=y barreverse=y
933          title="Dix Velocity in time"
934          scalebar=y barlabel=Velocity
935          barunit=m/s label2="Distance" unit2=m
936          '','')
937

```

```

938 Flow( 'vmigr1' , 'slice' ,
939   ' ,
940     pad beg1=1375 | math output="1550" |
941     window n1=1375 | put o1=0 d2=16.667 o2=0
942   ' ')
943 Flow( 'vmigr2' , 'vdix2' , 'window min1=5.5' )
944 Flow( 'vmigr' , 'vmigr1 vmigr2' , 'cat ${SOURCES[1:2]} axis=1' )
945
946 Result( 'vmigr' ,
947   ' ,
948
949   grey color=j mean=y barreverse=y
950   title="Migration Velocity in time"
951   scalebar=y barlabel=Velocity
952   barunit=m/s label2="Distance" unit2=m
953   ' ')
954
955 # Gazdag migration
956 Flow( 'gazdag' , 'cosft vmigr' ,
957   'cosft sign1=-1 | gazdag velocity=${SOURCES[1]} verb=y' )
958
959 Flow( 'image' , 'gazdag' , 'cosft sign2=-1' )
960
961 Result( 'image' ,
962   ' ,
963   window max1=8.5 min1=5.5 | grey
964   title="Phase-Shift Migration"
965   ' ')
966
967 # Kirchhoff time migration
968 Flow( 'kmig' , 'slice vpick' ,
969   ' ,
970     put d2=16.667 o2=0 | kirchnew velocity=${SOURCES[1]}
971   ' ')
972 Result( 'kmig' ,
973   ' ,
974   window max1=8.5 min1=5.5 | grey
975   title="Kirchhoff Post-stack Migration"
976   ' ')
977
978 Flow( 'vdixz' , 'vmigr' ,
979   ' ,
980     time2depth velocity=$SOURCE intime=y nz=2750 z0=0 dz=5 |
981     put label1=Depth unit1=m
982   ' ')
983 Result( 'vdixz' ,
984   ' ,
985

```

```

986     grey color=j mean=y barreverse=y
987     title="Dix Velocity in depth"
988     scalebar=y barlabel=Velocity
989     barunit=m/s label2="Distance" unit2=m
990     ''')
991 Flow('fft','vdixz','transp | fft1 | fft3 axis=2 pad=1')
992 Flow('right left','vdixz fft',
993      ''',
994      transp | scale dscale=0.5 |
995      isolr2 seed=2016 dt=0.002 npk=50
996      fft=${SOURCES[1]} left=${TARGETS[1]}
997      ''')
998 Flow('rtm snaps','slice left right',
999      ''',
1000      put o2=0 d2=16.667 | spline n1=5500 o1=0 d1=0.002 |
1001      reverse which=1 |
1002      transp | fftexp0 mig=y snap=10 snaps=${TARGETS[1]}
1003      left=${SOURCES[1]} right=${SOURCES[2]}
1004      nz=2750 dz=5
1005      ''')
1006
1007 Result('rtm',
1008      ''',
1009      put d2=1 o2=900 unit1=m label1=Depth
1010      unit2= label2=CMP | window min1=4000 max1=7000 |
1011      grey title="Post-Stack Depth Migration"
1012      ''')
1013 Plot('snaps','grey title=Snapshots gainpanel=all unit2=m',view=1)
1014
1015 # Split-step
1016 Flow('slo','vdixz',
1017      ''',
1018      put d4=1 o4=1 d2=0.016667 d1=0.005 d3=0.016667 |
1019      transp | transp plane=23 |
1020      math output="1/(input/1000)"
1021      ''')
1022 Result('slo',
1023      ''',
1024      put label1=Distance unit1=km label2=
1025      unit2= unit3=km | byte |
1026      sfgrey3 color=j frame1=150 frame2=150 frame3=150
1027      flat=n title="Slowness"
1028      ''')
1029 Flow('fft2','slice',
1030      ''',
1031      put d2=0.016667 o2=0 d3=0.016667 | fft1 |
1032      transp plane=12 | transp plane=23
1033

```

```

1034      ''')
1035 Flow( 'mig' , 'fft2 slo' ,
1036      '',
1037      zomig3 omnth=1 mode=m --readwrite=y verb=y
1038      nrmax=10 slo=${SOURCES[1]}
1039      ''', split=[3, 'omp', [0]], reduce='add')
1040
1041 Result( 'mig' ,
1042      '',
1043      put d1=1 o1=900 d3=5 d2=16.667 unit3=m
1044      label3=Depth unit1= label1=CMP |
1045      transp plane=23 | transp plane=12 |
1046      window min1=4000 max1=7000 | grey
1047      title="Split-step migration"
1048      ''')
1049 #####
1050 # Zoom an interesting area
1051 #####
1052 #####
1053 min1,max1=6.0,6.8
1054 min2,max2=950,1050
1055
1056 for i in range ( 3):
1057     case=( 'slice' , 'mig2' , 'slice1')[ i ]
1058     zoom = case + '-zoom'
1059     Flow(zoom, case ,
1060           '',
1061           window min1=%g max1=%g min2=%g max2=%g
1062           ''' % (min1,max1,min2,max2))
1063     Plot(zoom, 'grey title=%s grid=y gridcol=9 label2=CMP unit2=%(%123'[ i ]))
1064 for i in range ( 5):
1065     case=( 'mstack' , 'mfrik' , 'image' , 'stack-shot' , 'kmig')[ i ]
1066     zoom2 = case + '-zoom'
1067     Flow(zoom2, case ,
1068           '',
1069           put d2=1 o2=900 | window min1=%g max1=%g min2=%g max2=%g
1070           ''' % (min1,max1,min2,max2))
1071     Plot(zoom2,
1072           '',
1073           grey title=%s grid=y gridcol=9
1074           label2=CMP unit2=
1075           ''' % ('45609'[ i ]))
1076     Plot( 'mig-zoom' , 'mig' ,
1077           '',
1078           put d1=1 o1=900 d3=5 d2=16.667
1079           unit3=m label3=Depth unit1= label1=CMP |
1080           transp plane=23 | transp plane=12 |
1081           window min1=4730 max1=5550 min2=950 max2=1050 |

```

```

1082     grey title=%s grid=y gridcol=9
1083     ''' % ('8'))
1084 Plot('rtm-zoom', 'rtm',
1085     '',
1086     put d2=1 o2=900 unit1=m label1=Depth unit2= label2=CMP |
1087     window min1=4730 max1=5550 min2=950 max2=1050 |
1088     grey title=%s grid=y gridcol=9
1089     ''' % ('7'))
1090 Result('zoom1', 'slice-zoom mig2-zoom slice1-zoom', 'SideBySideIso')
1091 Result('zoom2', 'mstack-zoom mfpik-zoom image-zoom', 'SideBySideIso')
1092 Result('zoom3', 'rtm-zoom mig-zoom kmig-zoom stack-shot-zoom',
1093     'SideBySideIso')

1094 #####
1095 # Zoom an interesting area
1096 #####
1097 min1,max1=5.9,6.8
1098 min2,max2=950,1200

1100
1101 for i in range(3):
1102     case=( 'slice' , 'mig2' , 'slice1')[ i ]
1103     zoom = case + '-zoom2'
1104     Flow(zoom, case,
1105           '',
1106           window min1=%g max1=%g min2=%g max2=%g
1107           ''' % (min1,max1,min2,max2))
1108     Plot(zoom, 'grey title=%s grid=y gridcol=9 label2=CMP unit2=%(%[123][i]))')
1109 for i in range(5):
1110     case=( 'mstack' , 'mfpik' , 'image' , 'stack-shot' , 'kmig')[ i ]
1111     zoom2 = case + '-zoom2'
1112     Flow(zoom2, case,
1113           '',
1114           put d2=1 o2=900 | window min1=%g max1=%g min2=%g max2=%g
1115           ''' % (min1,max1,min2,max2))
1116     Plot(zoom2,
1117           '',
1118           grey title=%s grid=y gridcol=9
1119           label2=CMP unit2=
1120           ''' % ('45609'[i]))
1121 Plot('mig-zoom2', 'mig',
1122     '',
1123     put d1=1 o1=900 d3=5 d2=16.667 unit3=m
1124     label3=Depth unit1= label1=CMP |
1125     transp plane=23 | transp plane=12 |
1126     window min1=4600 max1=5550 min2=950 max2=1200 |
1127     grey title=%s grid=y gridcol=9
1128     ''' % ('8'))
1129 Plot('rtm-zoom2', 'rtm',

```

```

1130      , ,
1131      put d2=1 o2=900 unit1=m label1=Depth unit2= label2=CMP |
1132      window min1=4600 max1=5550 min2=950 max2=1200 |
1133      grey title=%s grid=y gridcol=9
1134      ''' % ('7'))
1135 Result('zoom4', 'slice-zoom2 mig2-zoom2 slice1-zoom2', 'SideBySideIso')
1136 Result('zoom5', 'mstack-zoom2 mfpik-zoom2 image-zoom2', 'SideBySideIso')
1137 Result('zoom6', 'rtm-zoom2 mig-zoom2 kmig-zoom2 stack-shot-zoom2',
1138      'SideBySideIso')

1139 #####
1140 # Zoom an interesting area
1141 #####
1142 min1,max1=5.9,6.8
1144 min2,max2=1150,1250

1145
1146 for i in range(3):
1147     case=( 'slice' , 'mig2' , 'slice1' )[ i ]
1148     zoom = case + '-zoom3'
1149     Flow(zoom,case,
1150           '',
1151           window min1=%g max1=%g min2=%g max2=%g
1152           ''' % (min1,max1,min2,max2))
1153     Plot(zoom,
1154           '',
1155           grey title=%s grid=y gridcol=9 label2=CMP unit2=
1156           ''' % ('123'[ i ]))
1157 for i in range(5):
1158     case=( 'mstack' , 'mfpik' , 'image' , 'stack-shot' , 'kmig' )[ i ]
1159     zoom2 = case + '-zoom3'
1160     Flow(zoom2,case,
1161           '',
1162           put d2=1 o2=900 | window min1=%g max1=%g min2=%g max2=%g
1163           ''' % (min1,max1,min2,max2))
1164     Plot(zoom2,
1165           '',
1166           grey title=%s grid=y gridcol=9 label2=CMP unit2=
1167           ''' % ('45609'[ i ]))
1168 Plot('mig-zoom3', 'mig',
1169           '',
1170           put d1=1 o1=900 d3=5 d2=16.667 unit3=m
1171           label3=Depth unit1= label1=CMP |
1172           transp plane=23 | transp plane=12 |
1173           window min1=4600 max1=5550 min2=1150 max2=1250 |
1174           grey title=%s grid=y gridcol=9
1175           ''' % ('8'))
1176 Plot('rtm-zoom3', 'rtm',
1177           ''

```

```

1178 put d2=1 o2=900 unit1=m label1=Depth unit2=
1179 label2=CMP | window min1=4600 max1=5550
1180 min2=1150 max2=1250 | grey title="%s" grid=y gridcol=9
1181 ''' % ('7'))
1182 Result('zoom7', 'slice-zoom3 mig2-zoom3 slice1-zoom3', 'SideBySideIso')
1183 Result('zoom8', 'mstack-zoom3 mfpik-zoom3 image-zoom3', 'SideBySideIso')
1184 Result('zoom9', 'rtm-zoom3 mig-zoom3 kmig-zoom3 stack-shot-zoom3',
1185 'SideBySideIso')

1186 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. Obama, 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.