

2-D Seismic Data Processing Exercise

Enter Name Here

ABSTRACT

In this exercise, we will process the 2-D Nankai dataset using the following seismic processing workflow:

1. Reading data in SU format and converting it to RSF format.
2. Frequency filtering to remove noise.
3. NMO and stack.
4. DMO and stack.
5. Post-stack Stolt migration.

The tutorial code will be available from the USB flash drive in the directory **nankai**.

GENERATING THIS DOCUMENT

At any point of doing this computational exercise, you can regenerate this document and display it on your screen.

1. Change directory to **nankai**:

```
cd $RSFSRC/book/data/nankai
```

2. Run

```
sftour scons lock  
scons read &
```

As the first step, open **nankai/paper.tex** file in your favorite editor and edit the first line to enter your name. Then run **scons read** again.

READING DATA

In this part of the tutorial, you will experiment with reading a seismic dataset in SU format, converting it to RSF format, examining its values, and displaying it on the screen. You are asked to answer a series of simple questions. Insert your answers by editing the **nankai/paper.tex** file.

1. Change directory to `nankai/data`.

2. Run

```
scons -c
```

to remove (clean) previously generated files.

3. Run

```
scons shots.rsfs
```

This command will find the SU file for the Nankai data and convert to two files: trace header data (`tshots.rsfs`) and trace data (`shots.rsfs`).

4. Examine the trace data file `shots.rsfs` by running

```
more shots.rsfs
```

What is the data format used in this file? How many traces does this file have (`n2=` parameter)? How many time samples per trace (`n1=` parameter)?

Answer:

Note that a more convenient way to check parameters is running a Madagascar program `sfin`:

```
sfin shots.rsfs
```

5. Examine the range of values in the trace data file by running

```
sfattr < shots.rsfs
```

What is the maximum value? How many zero samples are in this file?

Answer:

6. Run

```
scons shots2.view
```

to display all shot gathers in the Nankai dataset. Some traces are missing. To view the mask of the missing traces shots, run

```
scons smask.view
```

and observe where missing traces are located.

7. Examine the content of the trace header file `tshots.rsfs` by running

```
sfheaderattr < tshots.rsfs
```

To get a short description of different keys, you can also run

```
sfheaderattr < tshots.rsf desc=y
```

8. Run

```
scons shot.view
```

to window and display traces corresponding to the 1707th shot record (Figure 1).

9. Run

```
scons offset.rsf
```

to window values of offset (receiver-shot distance) for the 1707th shot record. Examine the values by running

```
sfdisfil < offset.rsf
```

What is the offset for the receiver closest to the source? What is the distance between receivers?

Answer:

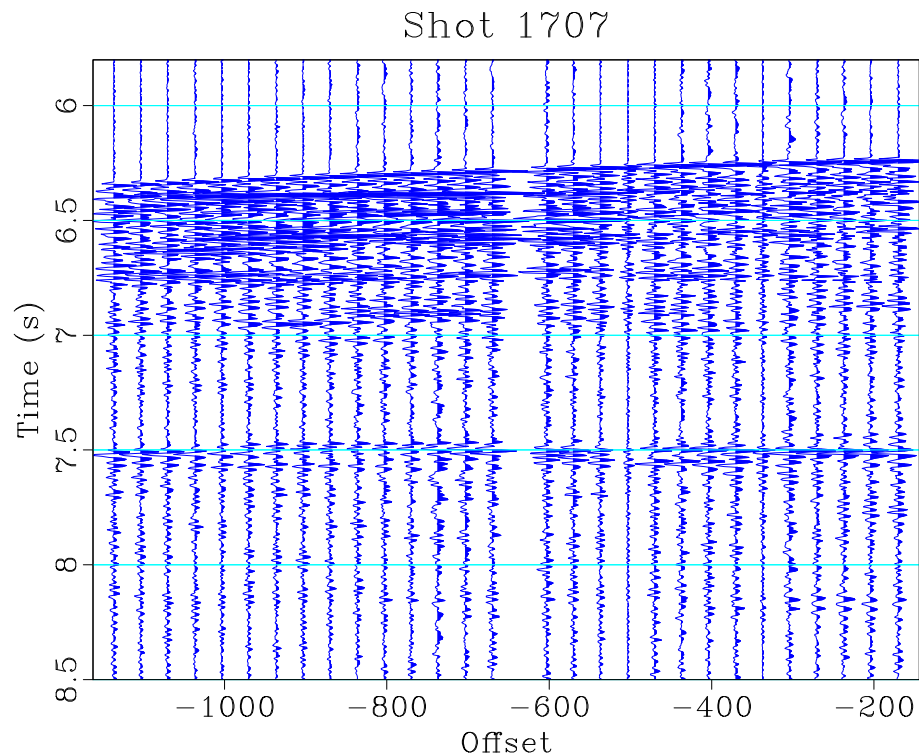


Figure 1: Wiggle plot of the 1707th shot record from the Nankai data. [data/shot](#)

10. Open the `SConstruct` file in your favorite editor and find the command which windowed the 1707th shot record. Change it to window different shot records. Run

```
scons shot.view
```

again to see the effect of the change. Edit the plot title to reflect the change.

FREQUENCY FILTERING

In this part of the tutorial, we will start processing the Nankai dataset by attenuating noise using frequency filtering.

1. We will design the noise attenuation filter as a muting function in the Fourier domain. Let us design the filter by focusing on one selected shot gather. The 2-D Fourier spectrum of the gather before and after filtering is shown in Figure 2. The corresponding separation of the gather into signal and noise is shown in Figure 3. Ideally, we would like to see no noise remaining in the estimated signal and no signal leaking into the estimated noise.

To reproduce the denoising result, run

```
scons signal.view
```

To display the Fourier-domain functions, run

```
scons fft.view
scons mute.view
scons filter.view
```

The mute function is implemented in the attached `filter.c` program and is designed as a triangular wedge controlled by two command-line parameters: velocities `v1` and `v2`. The filter passes seismic events with velocities higher than `v2`, attenuates those with velocities smaller than `v1`, and makes a smooth transition in between using a sine taper.

To run the noise attenuation filter on all shot gathers, run

```
scons fshots.view
```

2. Your task is to try improving the quality of the noise attenuation. You can do it by selecting better parameters or by modifying the `filter.c` program. Test your change not only on the selected shot gather but also on other gathers to make sure you achieve an improvement. Document your processing choices and create new figures if necessary.

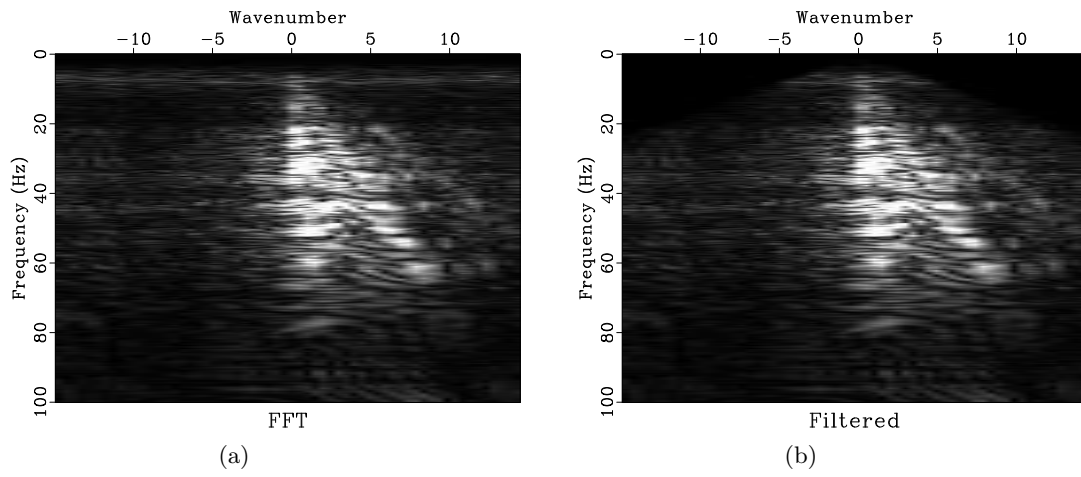


Figure 2: The 2-D Fourier spectrum of the Nankai shot before (a) and after (b) noise attenuation. [data/ fft,filter](#)

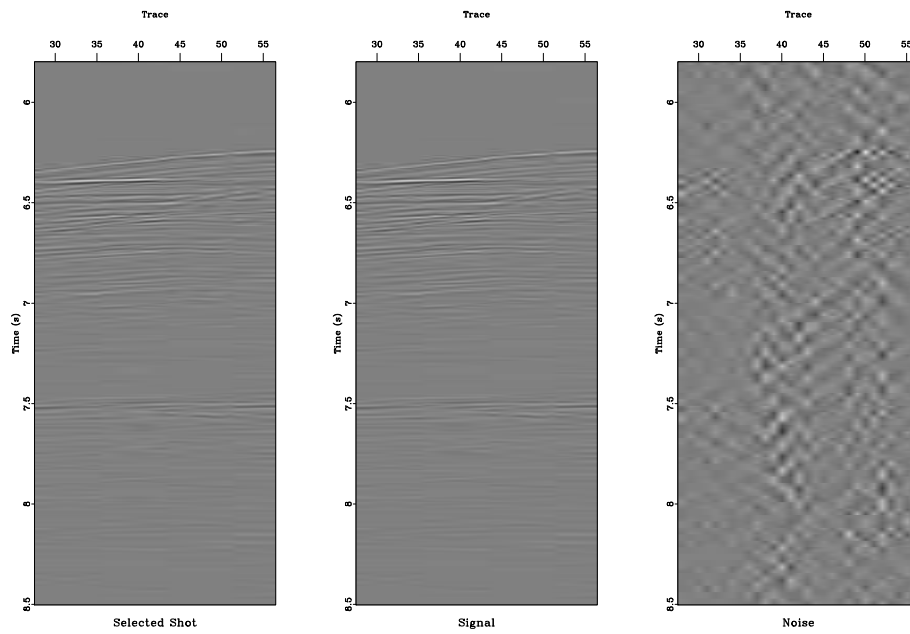


Figure 3: Shot gather separated into signal and noise. The noise section is gained to demonstrate that it contains no useful signal. [data/ signal](#)

NORMAL MOVEOUT (NMO)

The next step is to apply NMO-based velocity analysis to the Nankai dataset (Keys and Foster, 1998).

1. Run

```
scons cmps.view
```

to display the input data (Figure 4.)

2. What does “`pow pow1=2`” command do?

Hint: read the documentation for `sfpow` by running

```
sfpow
```

without arguments or by checking the “program of the month” blog post¹. Modify the `pow1=` parameter and rerun `scons` command to observe the change.

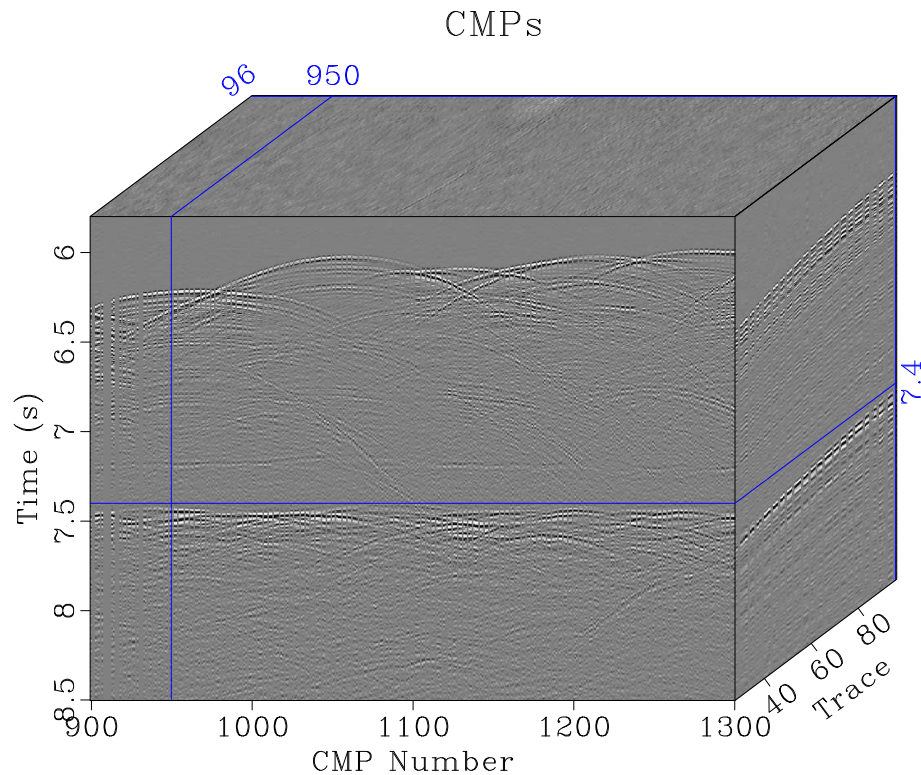


Figure 4: Nankai dataset after preprocessing and sorting into CMP gathers.

[data/ cmps](#)

3. We will test our velocity analysis first on one selected CMP gather. Run

```
scons cmp1.view
```

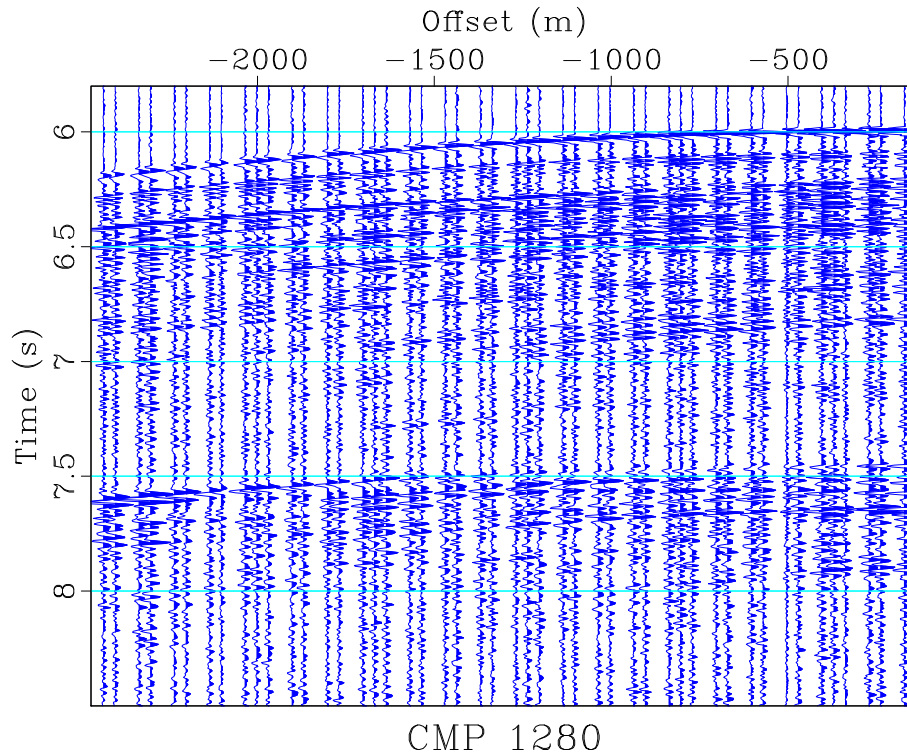


Figure 5: CMP gather selected for velocity analysis. [data/ cmp1](#)

to display the selected gather (Figure 5.)

4. The process of NMO-based velocity analysis consists of three steps:

- (a) Semblance scan.
- (b) Picking velocities.
- (c) Applying NMO correction.

The second step is typically done manually and consumes a significant portion of the processor's effort. To speed up the process, we will do it using an automatic picking algorithm.

Run

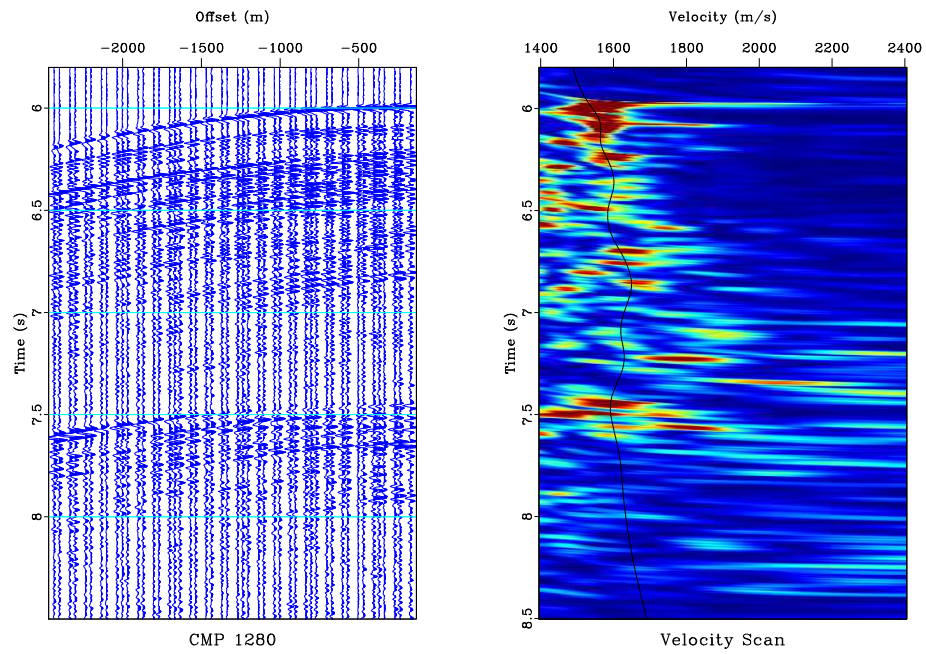
```
scons vscan.view
scons nmo.view
```

to observe the result of velocity analysis on the selected CMP gather using automatic picking (Figure 6.)

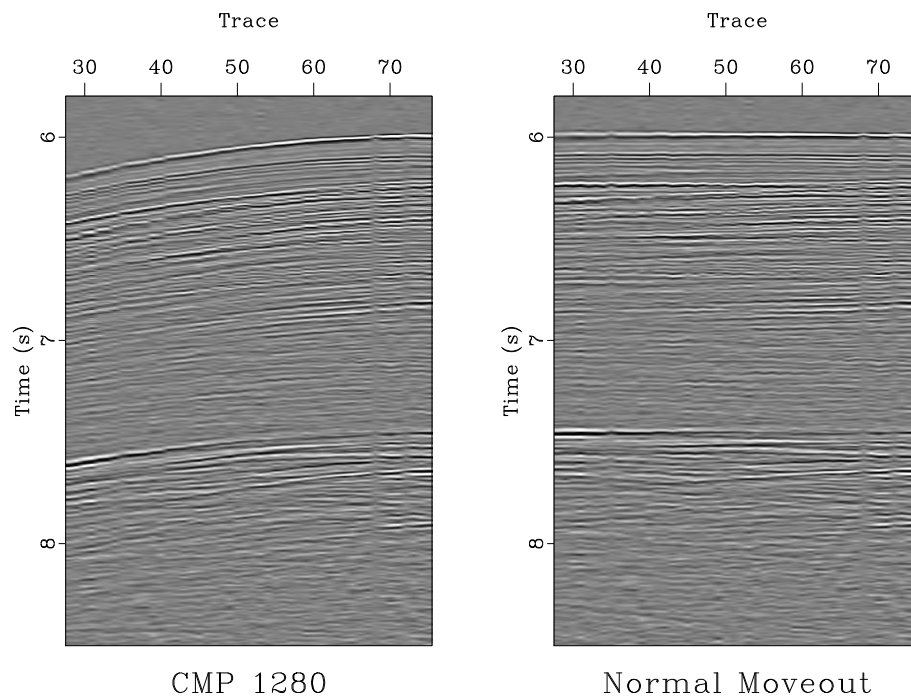
5. Once we are happy with our picking parameters, we can apply the procedure to the entire 2D line.

Run

¹<http://www.ahay.org/blog/2013/03/10/program-of-the-month-sfpow/>



(a)



(b)

Figure 6: NMO velocity analysis with automatic picking applied to the selected CMP gather. `data/ vscan,nm01`

```
scons picks.view
```

to run the semblance analysis with automatic picking on every CMP gather. The result is shown in Figure 7.

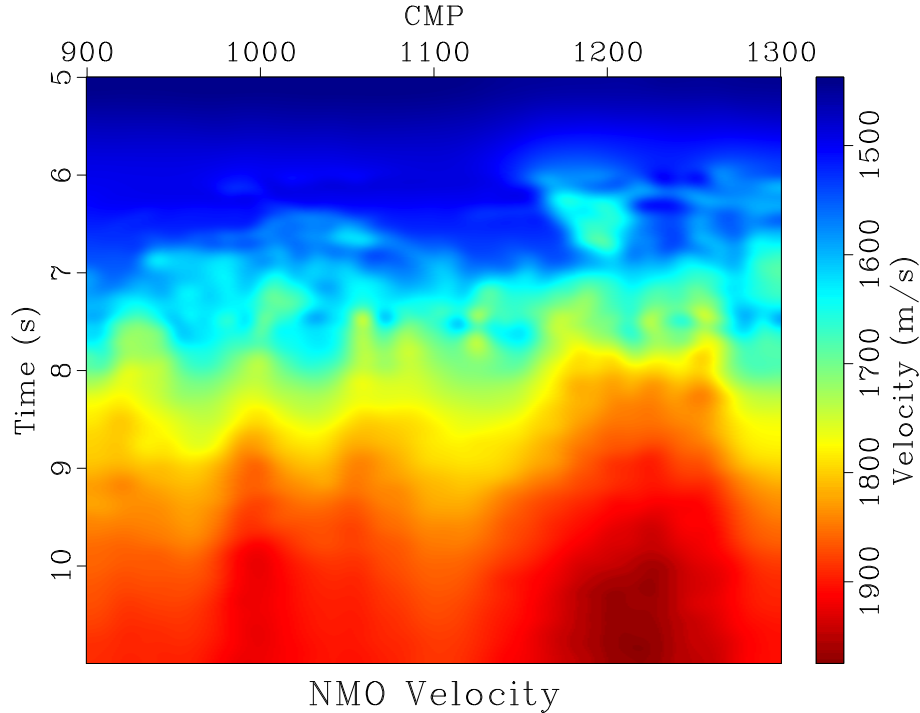


Figure 7: NMO stacking velocity picked automatically from the entire line.
[data/ picks](#)

6. Finally, the picked velocity can be applied to NMO and stacking.

Run

```
scons nmos.view
scons stack.view
```

to display the result (Figures 8 and 9.)

7. Your task is to try improving the quality of the stack (Figure 9) by improving the results of automatic picking. To achieve this goal, you can try changing parameters of the picking program `sfpick`: See <http://ahay.org/blog/2012/08/01/program-of-the-month-sfpick/> for more explanation.

DIP MOVEOUT (DMO)

In this section, we will apply an improved method of stacking (?) to the same dataset.

1. The method starts with generating an ensemble of NMO stacks with constant velocities (Figure 10.) To generate the stacks and display them on your screen, run

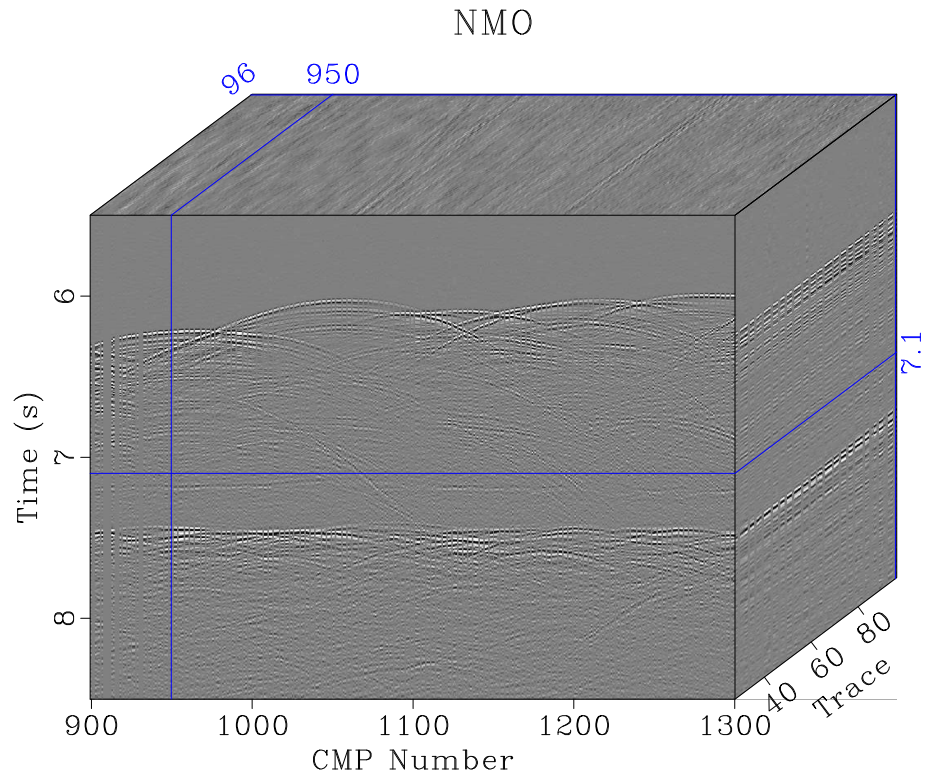


Figure 8: Nankai dataset after normal-moveout correction. [data/ nmos](#)

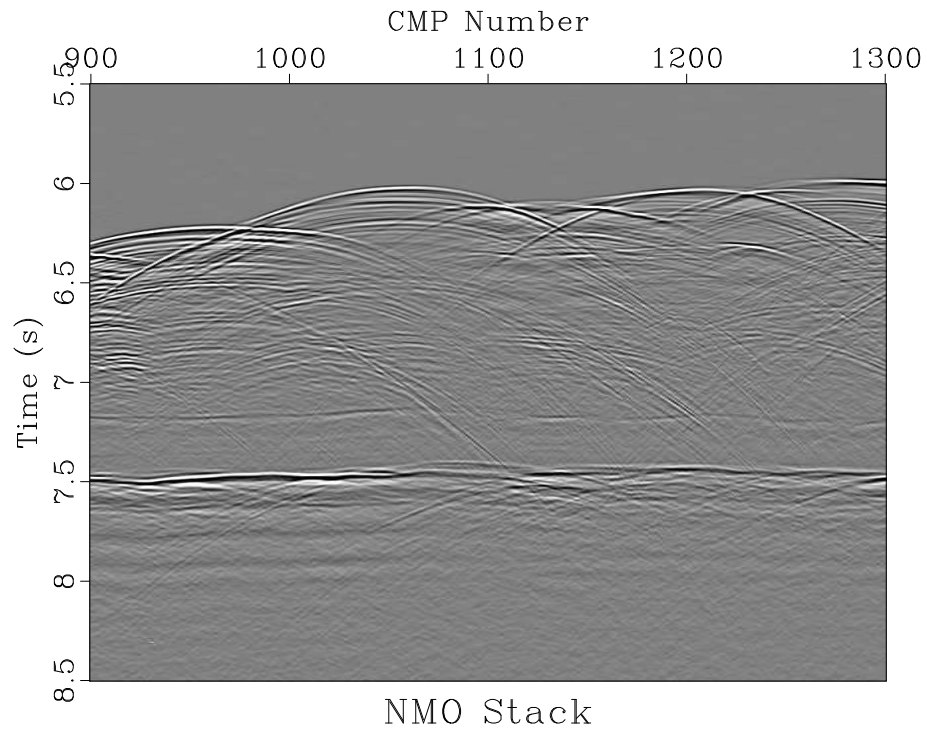


Figure 9: NMO stack of the Nankai dataset. [data/ stack](#)

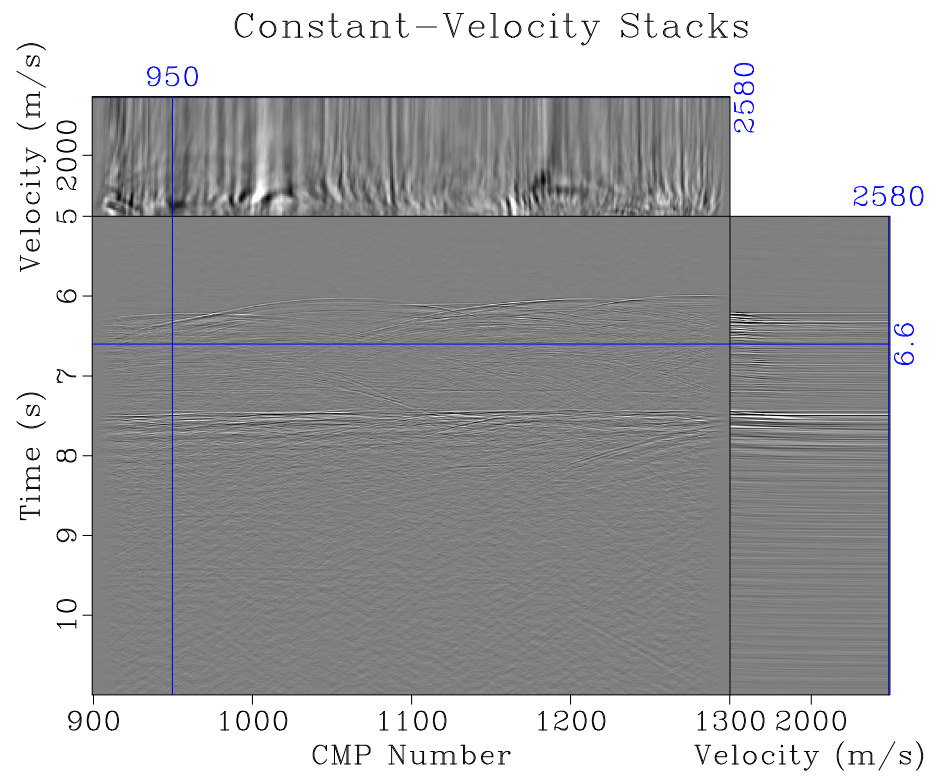


Figure 10: Nankai dataset after NMO stacking with an ensemble of constant velocities.

[data/ stacks](#)

```
scons stacks.view
```

2. Fowler's method works by transforming the stack volume into the frequency-wavenumber $\{\omega, k\}$ domain and applying the velocity mapping according to

$$v_0 = v \left(1 + \frac{k^2 v^2}{4 \omega^2} \right)^{-1/2}, \quad (1)$$

where v is NMO stacking velocity (dip-dependent) and v_0 is time-migration velocity (dip-independent).

3. To perform the DMO correction of the constant-velocity stacks (Figure 11), run

```
scons dmo.view
```

Compare the results by running

```
sfopen Fig/dmo.vpl Fig/stacks.vpl
```

Do you notice any improvements?

Answer:

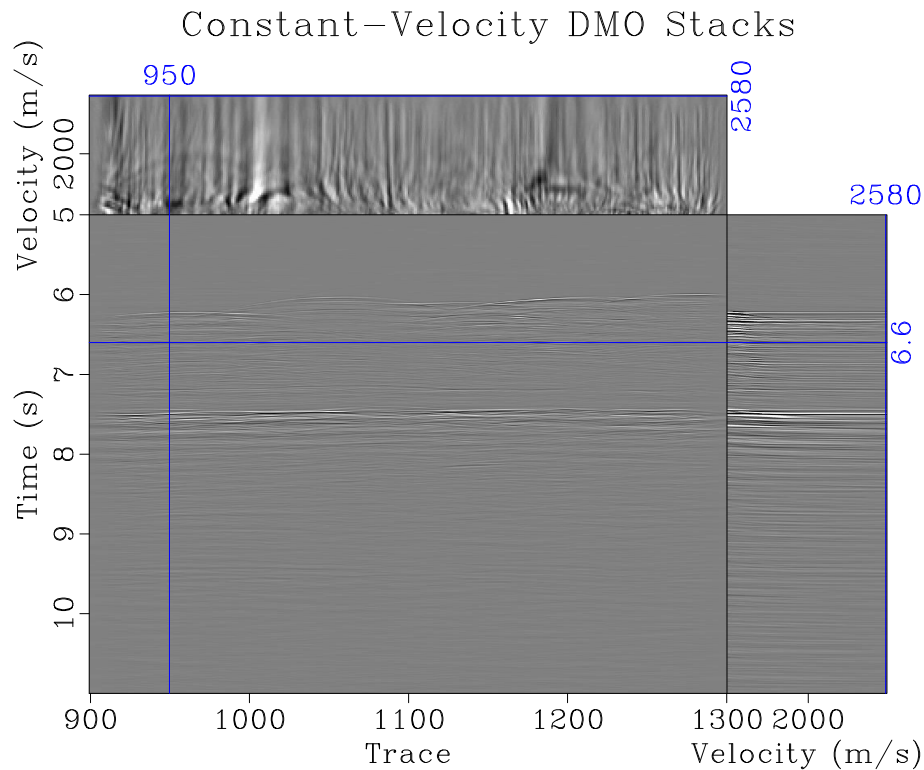


Figure 11: Nankai dataset after DMO stacking with an ensemble of constant velocities.

[data/ dmo](#)

4. Fowler's method does not generate semblance scans, but it is possible to pick velocities using the power of the stack or the envelope. To pick the DMO-corrected velocity automatically from the envelope (Figure 12), run

```
scons vpick.view
```

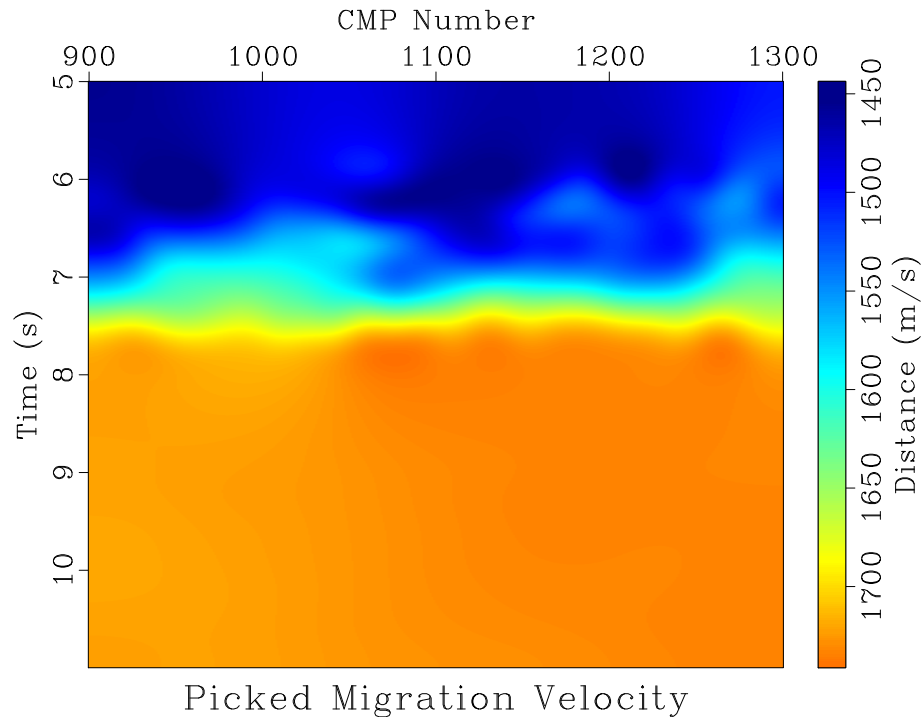


Figure 12: Migration velocity picked automatically from DMO stacks. [data/ vpick](#)

5. After the velocity is picked, we can generate a DMO stack by slicing through the velocity cube. Run

```
scons slice.view
```

to see the result (Figure 13.) To compare results, run

```
sfopen stack.vpl slice.vpl
```

and notice the changes brought by DMO.

6. To evaluate the velocity picking and to observe the change brought by DMO, we can examine a particular CMP location. Run

```
scons envelope.view
```

to display the result (Figure 14.) Modify the `SConstruct` file to try several other CMP locations and select one that shows the most interesting change.

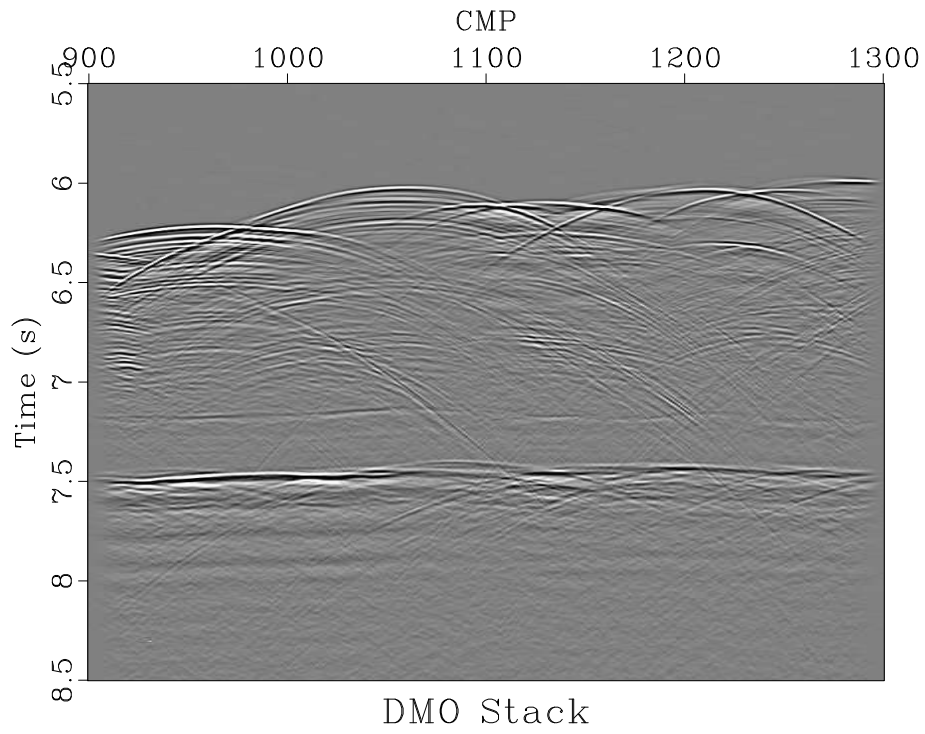


Figure 13: (a) DMO stack generated by slicing the constant-velocity stacks.
[data/ slice](#)

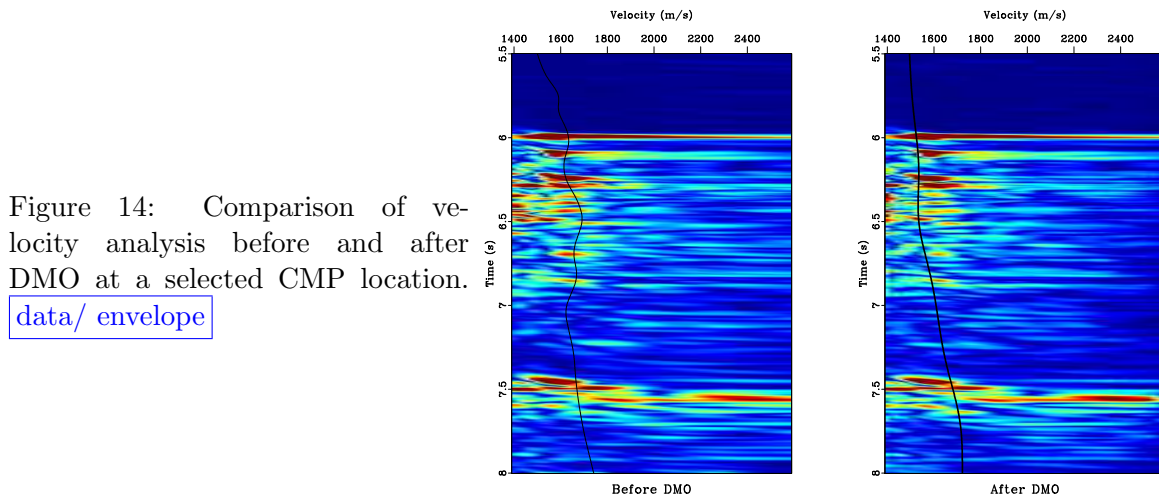


Figure 14: Comparison of velocity analysis before and after DMO at a selected CMP location.
[data/ envelope](#)

7. Similarly to the previous section, your creative task is to improve the stack (Figure 13) by improving the result of automatic picking. You can use any tools to find a better slice through the velocity cube. Document your work and include the results in this document.

Answer:

POST-STACK MIGRATION

In this part of the workflow, we will experiment with seismic migration applied to the previously generated DMO stack of the Nankai dataset. The particular migration algorithm that we will use is Stolt migration based on the Fourier transform (Stolt, 1985).

1. The input to migration is the DMO stacked section (Figure 13.)
2. Stolt migration amounts to a 2D Fourier transform from the time-space $\{t, y\}$ coordinates of the stack to the frequency-wavenumber coordinates $\{\omega, k\}$ followed by mapping from ω to ω_0 according to

$$\omega = \sqrt{\omega_0^2 + \frac{v^2}{4} k^2}, \quad (2)$$

where v is the migration velocity, and the inverse Fourier transform from $\{\omega_0, k\}$ to $\{t_0, x_0\}$ coordinates of the time-migrated image.

To compute and display the map from equation 2, run

```
scons map2.view
```

To display the 2-D Fourier transform (actually 2-D cosine transform) of the Nankai data before and after Stolt mapping with $v = 2$ km/s (Figure 15), run

```
scons cosft.view
scons cosft2.view
```

3. Run

```
scons mig2.view
```

to display a seismic image migrated with the velocity of 1500 m/s. To look at the change brought by migration (Figure 16), run

```
sfpn Fig/stack.vpl Fig/mig2.vpl
```

What changes do you notice?

Answer:

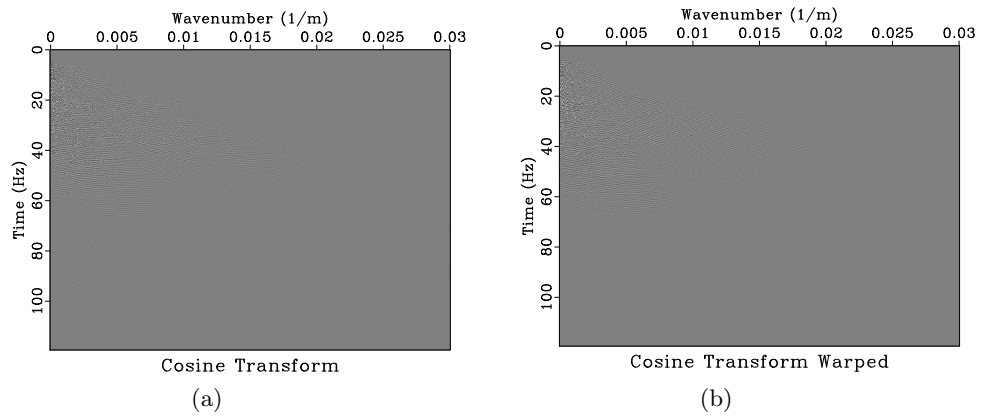


Figure 15: Nankai stack in the Cosine transform domain before (a) and after (b) Stolt migration with velocity 1500 m/s. [data/ cosft,cosft2](#)

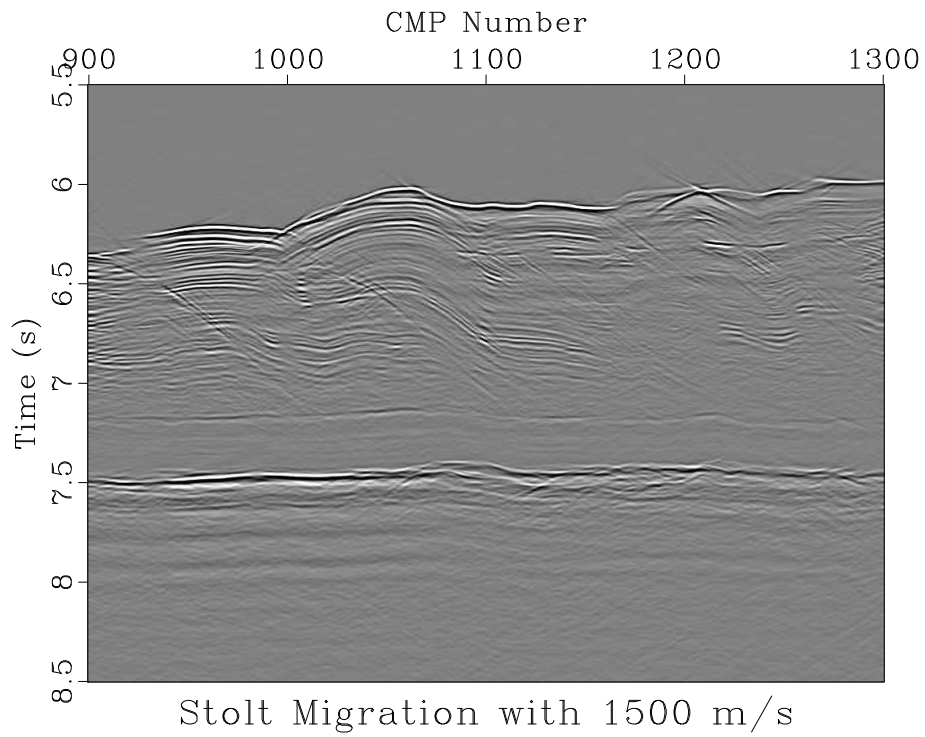


Figure 16: Nankai stack migrated with velocity of 1500 m/s. [data/ mig2](#)

4. Of course, the real velocity is not constant and is likely to change in space. To try a more realistic velocity distribution, we will start with a velocity that increases with vertical time (Figure 17.) The velocity starts with 1500 m/s (water velocity) at the surface and remains constant until the water bottom reflection around 6 s. The velocity then increases to 1750 m/s at the vertical time of 8.5 s. To perform migration with a variable velocity using the Stolt method, we will migrate with a number of constant velocities in the range from 1500 to 1750 m/s and then slice through this ensemble of migrations to create an image (Mikulich and Hale, 1992). Thanks to the speed of the FFT algorithm, the whole operation is reasonably fast.

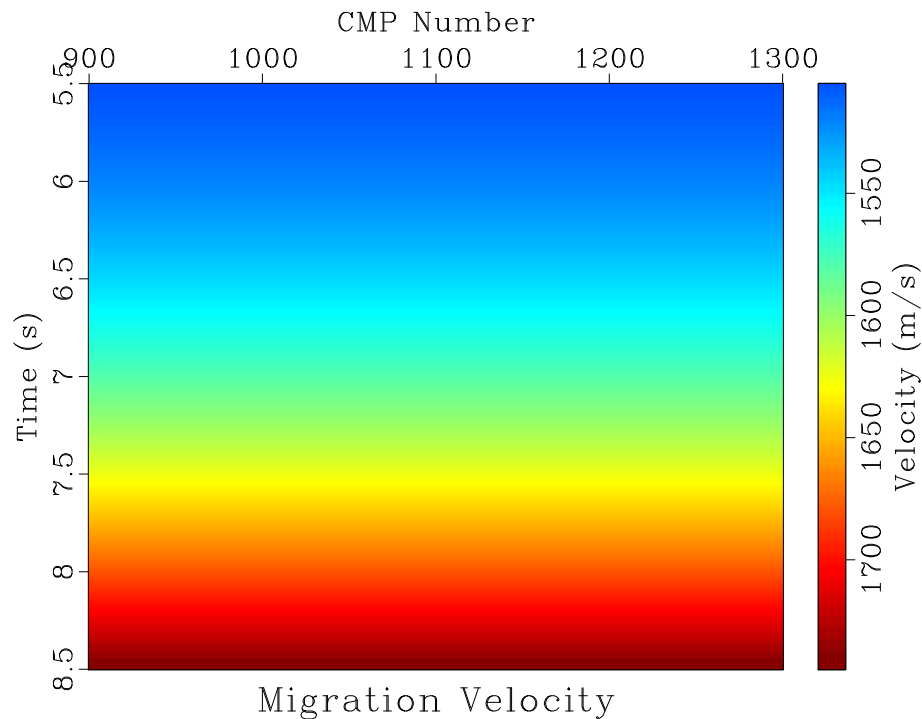


Figure 17: Variable migration velocity. [data/ vmig](#)

To generate an image using an ensemble of Stolt migrations (Figure 18), run

```
scons mig.vpl
scons mig.view
```

To look at the change brought by a variable velocity, run

```
sfopen Fig/mig2.vpl Fig/mig.vpl
```

Do you notice any interesting changes? Your task is to edit the **SConstruct** file to modify the migration velocity function used in order to improve the migration results.

Answer:

5. Let us select a small region of the image where a particularly interesting change occurs. One of such regions is shown in Figure 19. To display it on your screen, run

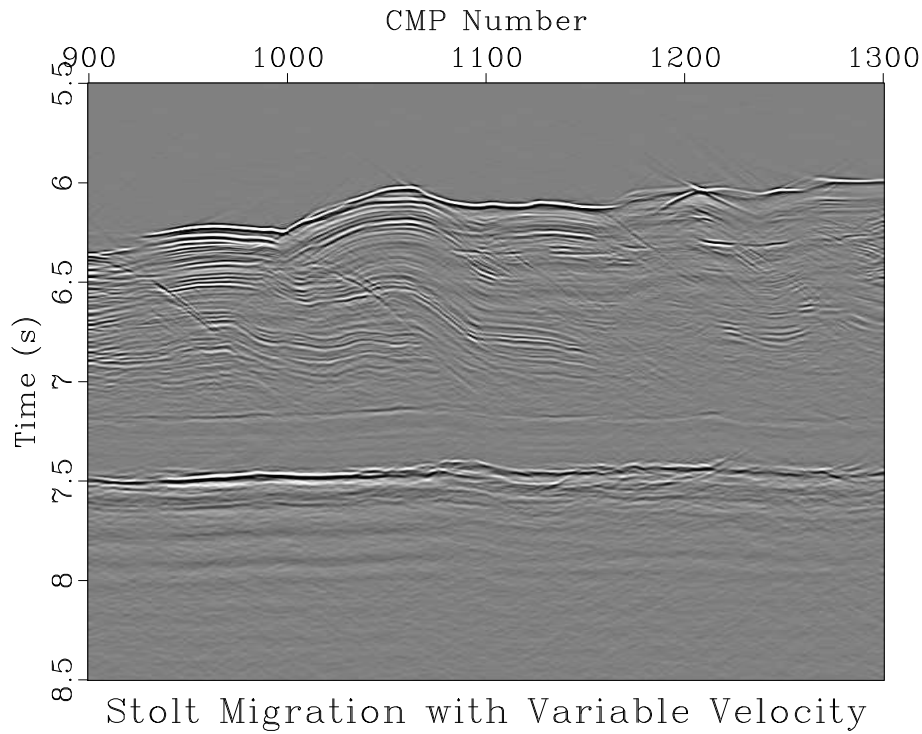


Figure 18: DMO stack migrated with variable velocity. [data/ mig](#)

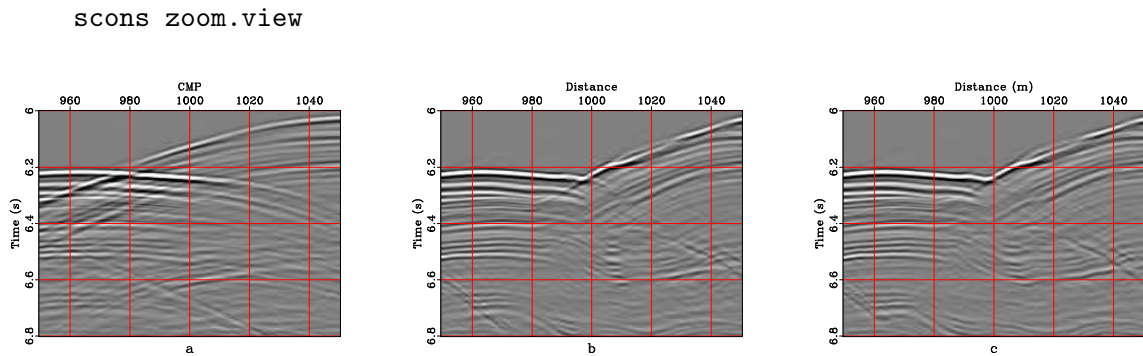


Figure 19: Zoomed comparison: (a) unmigrated DMO stack (b) migrated with 1500 m/s, and (c) migrated with variable velocity. [data/ zoom](#)

Edit and modify the `SConstruct` file to select a different region that you find interesting and modify the figure.

REFERENCES

- Keys, R. G., and D. J. Foster, eds., 1998, Comparison of Seismic Inversion Methods on a Single Real Data Set: Soc. of Expl. Geophys.
- Mikulich, W., and D. Hale, 1992, Steep-dip $v(z)$ imaging from an ensemble of Stolt-like migrations: *Geophysics*, **57**, 51–59.

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