Advanced Offshore Multichannel Processing

(Revised 14.05.2025)

RADEXPRO SEISMIC SOFTWARE LLC
Tbilisi, Georgia
Belgrade, Serbia

t: +381 62 1494544
www.radexpro.com

E-mails:
support@radexpro.com
sales@radexpro.com

Content

Introduction

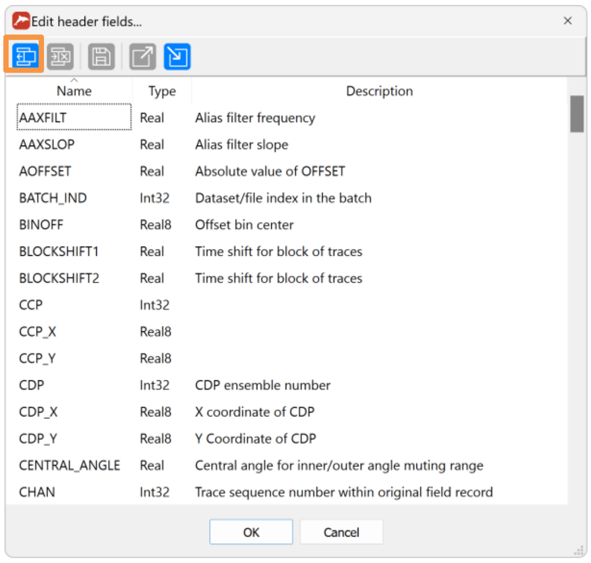
This tutorial provides a comprehensive overview of processing offshore multichannel high-resolution seismic data using **RadExPro**. It covers all the necessary steps to generate a migrated image from the seismic gathers recorded in the field. The processing workflow includes data input, geometry setup, preprocessing, demultiple, deghosting, deconvolution, and migration. The tutorial is accompanied by a **RadExPro** project where the explained steps are present as processing flows. In the accompanying project, all the datasets are emptied to save space, the input file for the project is located inside the project directory. To reproduce the results from this tutorial, one needs to simply open the project and run the flows one by one from top to bottom (there is only one interactive binning section, where one needs to apply binning to the dataset – refer to the Binning section of the tutorial for this).

Note that this tutorial assumes some prior knowledge of **RadExPro**, it does not delve into basic concepts such as project/flow creation, setup of data displays, etc. If you need an introduction to these topics, consider referring to the "How to Create a New **RadExPro** Processing Project and Load Input Data" tutorial and the basic multichannel processing tutorial available on our website.

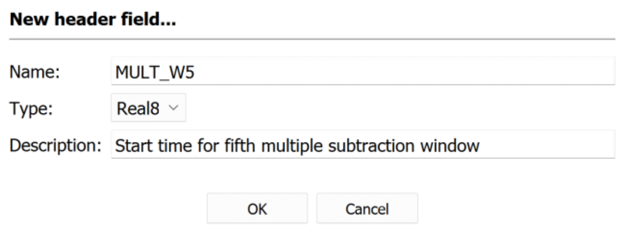
The input dataset for this tutorial was supplied by Applied Acoustics (L200 sparker, 48 channel streamer).

Header creation

Note that we use non-standard headers in several flows in the tutorial project (GUN_X, GUN_Y, SRME_GEOM, TOP_MUTE, SFLR_PICK, MODEL, MULT_W1, MULT_W2, MULT_W3, MULT_W4, etc.). These headers were created manually while working on this project. These headers are not present if a new project is created in **RadExPro**. To create a non-standard header, one needs to click *Database* => *Header fields list* => *Edit…* in the project window. A window will pop up. In this window, one needs to click the *Add* button (highlighted below), specify the name of the header, the data type (Int32, Real or Real8), and optional description, and click *OK* in both *New header field…* and *Edit header fields…* windows. This will create the new (empty) header with the specified name and type for all the datasets in the project.



Name	Type	Description
AAXFILT	Real	Alias filter frequency
AAXSLOP	Real	Alias filter slope
AOFFSET	Real	Absolute value of OFFSET
BATCH_IND	Int32	Dataset/file index in the batch
BINOFF	Real8	Offset bin center
BLOCKSHIFT1	Real	Time shift for block of traces
BLOCKSHIFT2	Real	Time shift for block of traces
CCP	Int32	
CCP_X	Real8	
CCP_Y	Real8	
CDP	Int32	CDP ensemble number
CDP_X	Real8	X coordinate of CDP
CDP_Y	Real8	Y Coordinate of CDP
CENTRAL_ANGLE	Real	Central angle for inner/outer angle muting range
CHAN	Int32	Trace sequence number within original field record



New header field...

Name: MULT_W5

Type: Real8

Description: Start time for fifth multiple subtraction window

Window for the editing of header fields with highlighted Add button (left) and New header field creation window (right)

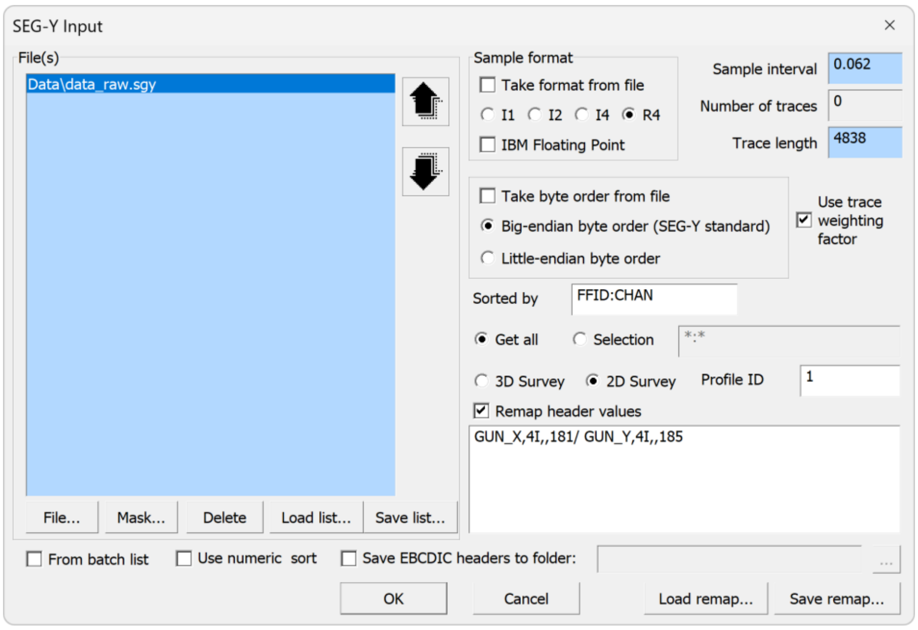
Data input

The flow **001-Data-Input** loads the data from a SEG-Y file into the project and saves it to a dataset.

```
SEG-Y Input <- data_raw.sgy
Trace Header Math
Header Enumerator -> TRACENO
Trace Output -> 001-raw
```

001-Data-Input flow

First, SEG-Y Input is used to input the *data_raw.sgy* file. SEG-Y Input detects the file format automatically, we specify the FFID:CHAN sorting and the *2D Survey* option. The *Remap header values* field is edited to load in the source coordinates to GUN_X and GUN_Y headers, which have non-standard locations.



SEG-Y Input

File(s): Data\data_raw.sgy

Sample format: [ ] Take format from file; I1 I2 I4 [✓ R4]; [ ] IBM Floating Point

Sample interval: 0.062
Number of traces: 0
Trace length: 4838

[ ] Take byte order from file
[✓ Big-endian byte order (SEG-Y standard)] Little-endian byte order

[✓] Use trace weighting factor

Sorted by: FFID:CHAN

[✓ Get all] Selection: *:*

3D Survey [✓ 2D Survey] Profile ID: 1

[✓] Remap header values
GUN_X,4I,,181/ GUN_Y,4I,,185

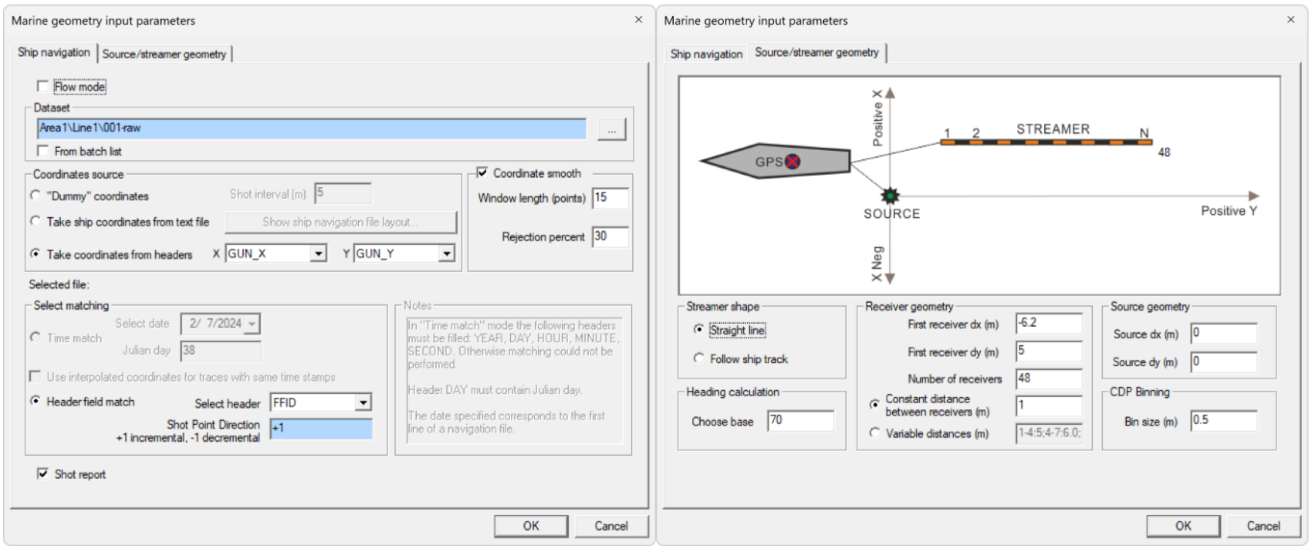
[ ] From batch list [ ] Use numeric sort [ ] Save EBCDIC headers to folder:

SEG-Y Input module parameters

In the following Trace Header Math module, we apply a scalar to the GUN_X and GUN_Y coordinates and set the line number in S_LINE header to 1. This is followed by the Header Enumerator, which simply numbers all the traces in the dataset sequentially and saves the resulting numbers to the TRACENO header. Finally, Trace Output saves the data to the *001-raw* dataset. Note that we use the Framed mode for most flows in this tutorial due to quite large data size.

Geometry

The geometry setup occurs in the **010-Geometry** flow. Only one standalone module, Marine Geometry Input*, is present in this flow. It computes the source and receiver coordinates from the information provided in the parameters and fills in the headers in the specified *Dataset* (here, we fill the headers in *001-raw*).



Marine geometry input parameters — Ship navigation

[ ] Flow mode
Dataset: Area1\Line1\001-raw
[ ] From batch list

Coordinates source:
"Dummy" coordinates — Shot interval (m): 5
Take ship coordinates from text file — Show ship navigation file layout...
[✓ Take coordinates from headers] X: GUN_X Y: GUN_Y

[✓] Coordinate smooth
Window length (points): 15
Rejection percent: 30

Selected file:

Select matching:
Time match — Select date: 2/ 7/2024; Julian day: 38
[ ] Use interpolated coordinates for traces with same time stamps
[✓ Header field match] Select header: FFID
Shot Point Direction +1 incremental, -1 decremental: +1

Notes: In "Time match" mode the following headers must be filled: YEAR, DAY, HOUR, MINUTE, SECOND. Otherwise matching could not be performed.
Header DAY must contain Julian day.
The date specified corresponds to the first line of a navigation file.

[✓] Shot report

Marine geometry input parameters — Source/streamer geometry

Streamer shape: [✓ Straight line] Follow ship track
Heading calculation: Choose base: 70

Receiver geometry:
First receiver dx (m): -6.2
First receiver dy (m): 5
Number of receivers: 48
[✓ Constant distance between receivers (m)]: 1
Variable distances (m): 1-4:5;4-7:6.0;

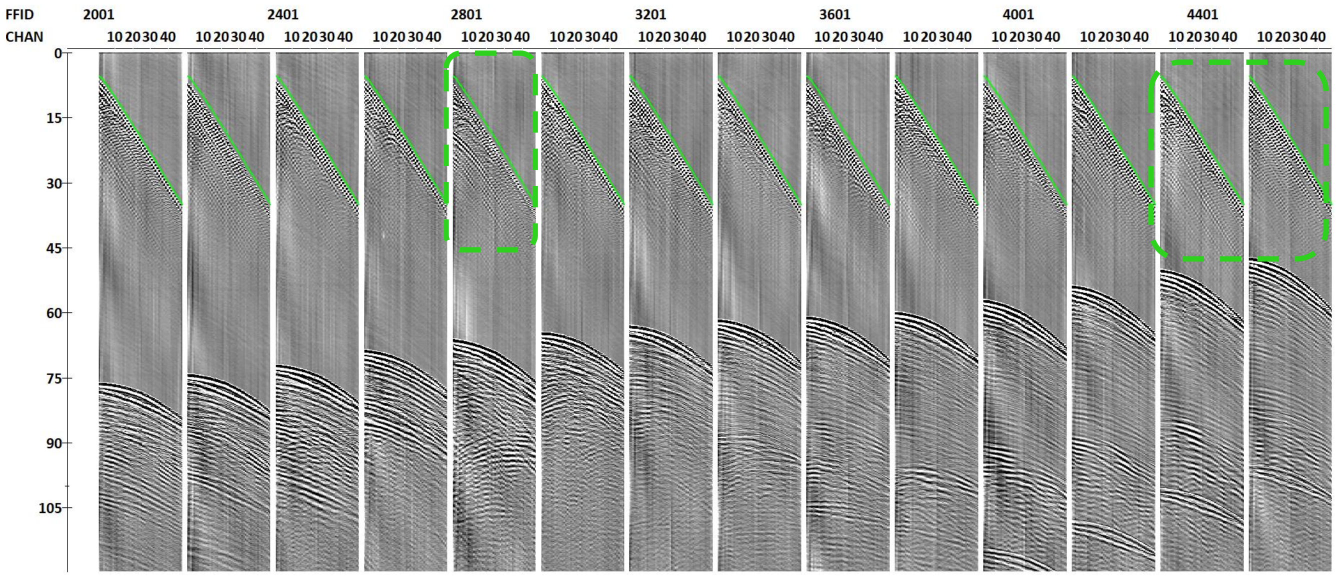
Source geometry:
Source dx (m): 0
Source dy (m): 0

CDP Binning: Bin size (m): 0.5

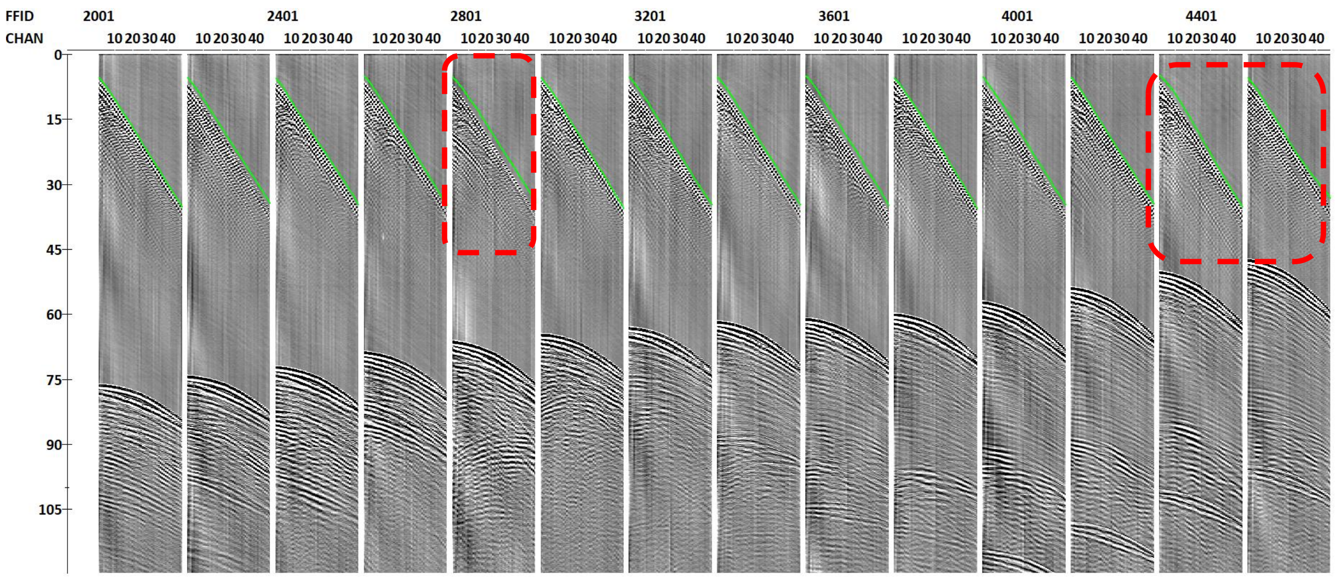
Marine Geometry Input module parameters

In this case, the source coordinates are present in the GUN_X and GUN_Y headers of the dataset, and the coordinates of the whole acquisition system are set relative to them. In the Ship navigation tab, *Take coordinates from headers* option is selected and GUN_X and GUN_Y headers are specified. We use the default *Coordinate smooth* parameters (these depend on the coordinate accuracy) and FFID as a header for matching. The *Shot Point Direction* set to +1 means that the shot numbers increase along this seismic line. On the *Source/streamer geometry* tab, the acquisition geometry is outlined. The *Streamer shape* is set to *Straight line*, as this option provides a better match with the direct wave arrivals later in geometry quality control for this dataset. We use 70 in *Choose base* to provide smoother streamer movement between the shots (which will be studied later using crossplots). Depending on the nature of the data, *Follow ship track* option may become preferable. The *Receiver geometry* section specifies the receiver setup of 48 receivers with 1 m distances between receivers. The offsets of the first receiver relative to the coordinates supplied in the previously specified GUN_X and GUN_Y headers are set in *First receiver dx (m)* (offset perpendicular to the ship track) and *First receiver dy (m)* (offset along the ship track). The source offsets relative to the coordinates supplied in GUN_X and GUN_Y headers are set in *Source dx (m)* and *Source dy (m)*. As GUN_X and GUN_Y store the exact source location, these offsets are equal to 0. The *Bin size* of this survey is 0.5 m. Upon launch, the flow writes the geometry headers to the dataset.

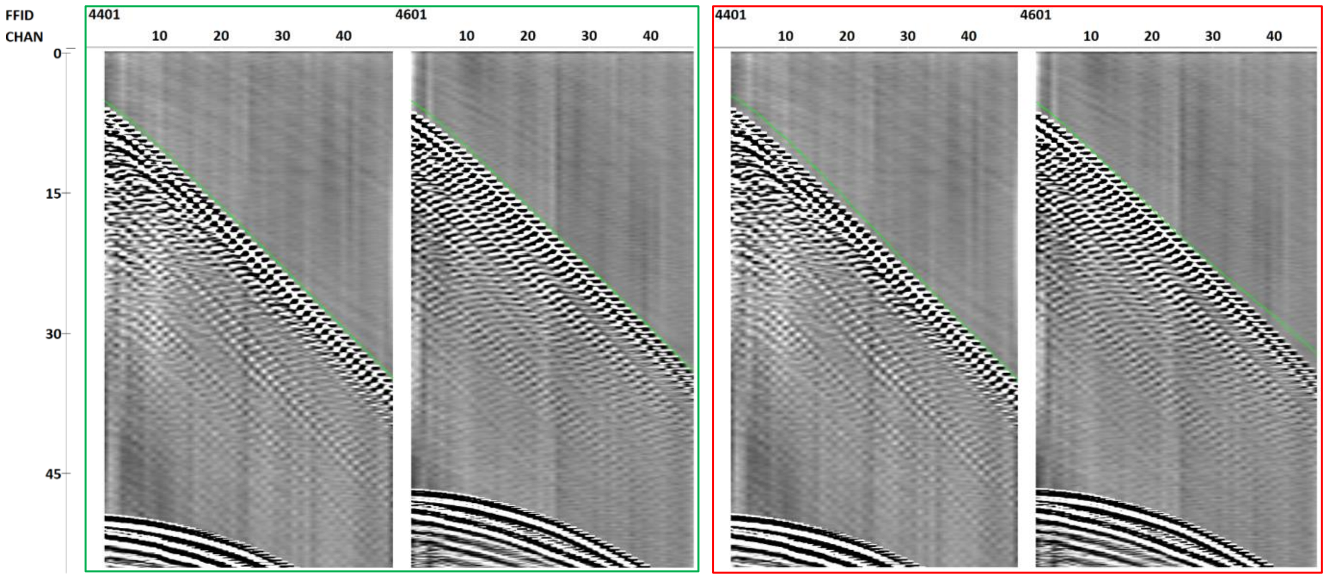
The quality control (QC) of the geometry setup occurs in the flow **012-Geometry-QC**. In this flow, we compute the direct wave traveltime in Trace Header Math and save it to AAXFILT as AAXFILT=OFFSET/1.49 (1.49 is the approximate water velocity). Then, these traveltimes are displayed over the seismic gathers – we can observe that in general the traveltimes are aligned with the direct wave on the gathers, which means the offsets are quite accurate. If we chose the *Follow ship track* option – the offsets would be incorrect in the areas where the ship track is curved (highlighted).



Direct wave traveltime overlay on seismic gathers



The same overlay, but with 'Follow ship track' option chosen



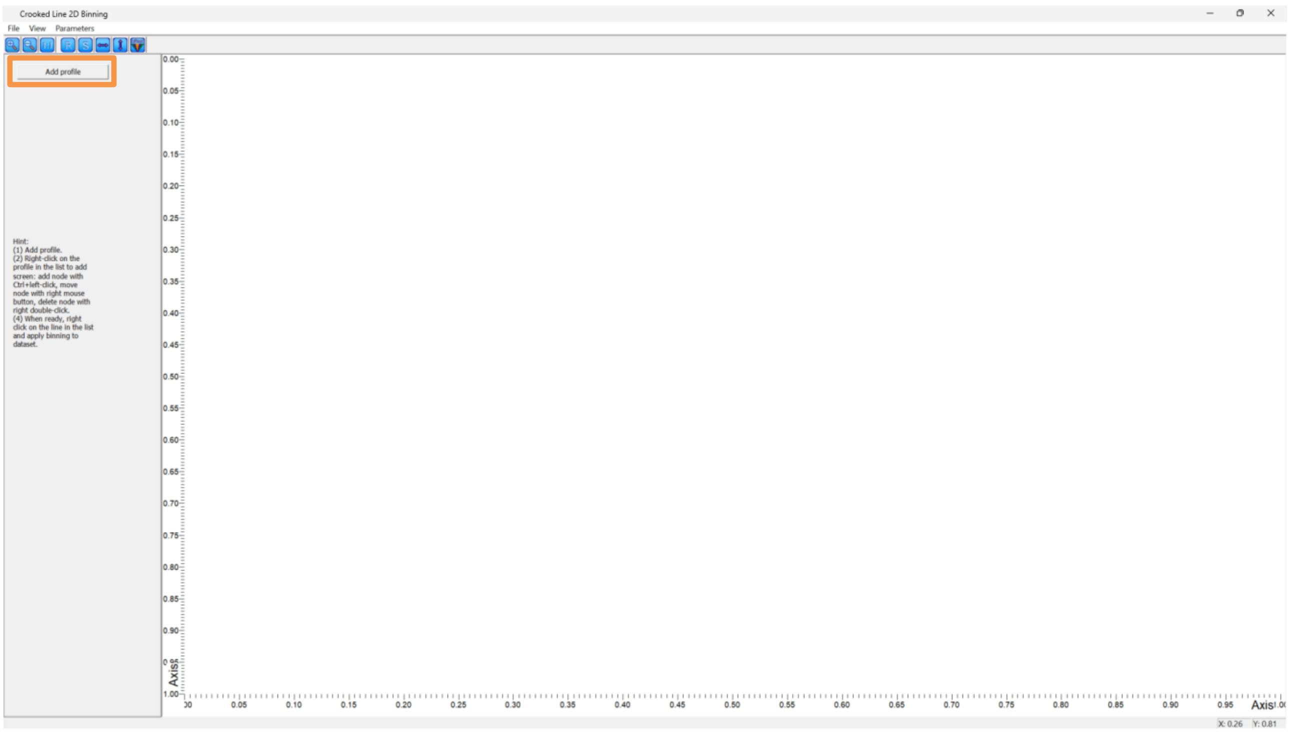
Zoom of two gathers with 'Straight line' (left) and 'Follow ship track' (right) geometries

Binning

The next step is CDP binning, which happens in the flow **015-Binning**. Marine Geometry Input* from the previous **010-Geometry** flow computes the midpoint coordinates of each trace in CDP_X and CDP_Y and supplies some bin numbers, which might work for straight seismic lines. In case the shot/receiver locations deviate from straight lines, accurate binning needs to be performed with Crooked Line 2D Binning*. This module creates a binning line which follows the true source and receiver locations and then writes the bin numbers to CDP and bin center coordinates to CDP_X and CDP_Y header of the chosen dataset.

The flow **015-Binning** contains only the interactive Crooked Line 2D Binning* module. The main module parameter is the scheme, which contains the created binning lines and some auxiliary data. Here, we created a new scheme titled *bin_scheme*. Other parameters are kept as defaults.

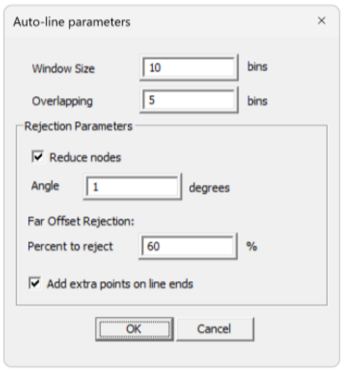
Next, we explain the steps required to create the binning line (in the supplied project, these steps are already completed). Upon launching the module window, one needs to click on the *Add profile* button and select the *001-raw* dataset.



Hint:
(1) Add profile.
(2) Right-click on the profile in the list to add screen: add node with Ctrl+left-click, move node with right mouse button, delete node with right double-click.
(4) When ready, right click on the line in the list and apply binning to dataset.

Crooked Line 2D Binning module window with highlighted Add profile button

Next, we add the automatically generated binning line. There is a set of parameters for automatic binning line generation in *Parameters => Auto-line parameters*. For the results shown in this tutorial, we increase the *Window Size* and *Overlapping* to 10 and 5 respectively to allow for a smoother binning line near the turns, as well as change *Far Offset Rejection* percent to 60%.

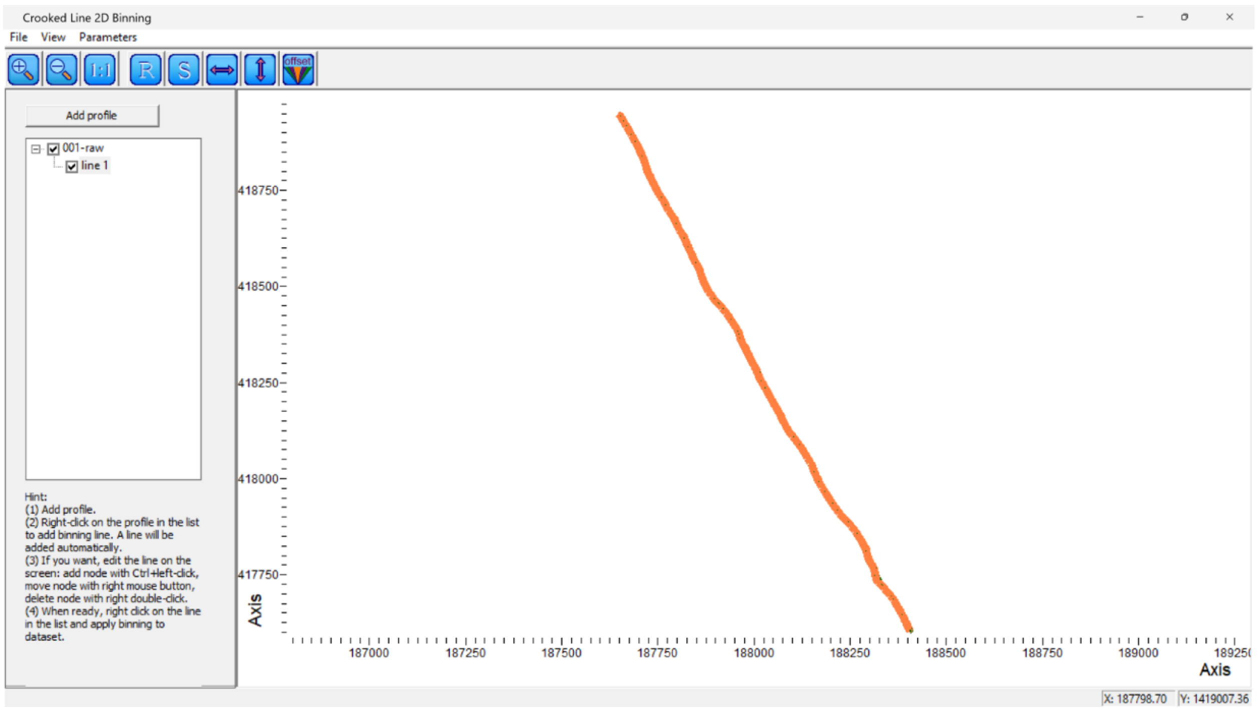


Window Size: 10 bins
Overlapping: 5 bins
Rejection Parameters
[✓] Reduce nodes
Angle: 1 degrees
Far Offset Rejection:
Percent to reject: 60 %
[✓] Add extra points on line ends

Auto-line parameters

Next, we right-click on the *001-raw* dataset which appears in the list on the left, select *Add binning line => Auto-line*. In the auto-line parameters, we set the *Bin size* to 0.5 m and *Swath range* to 10 m. Then, we specify the line starting point (as prompted) by clicking at the start of the seismic line (top left). Next, the binning line appears on the screen. To apply the binning provided by this line to the dataset and fill the headers, we right-click on the newly appearing *line 1* in the list on the left and select *Apply binning to dataset*.

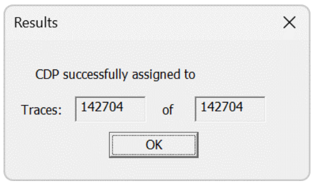
If you are running the project provided on our website, the only thing you need to do in this section to reproduce our results is right-click on *line 1* and select *Apply binning to dataset*, as the binning line is already there.



Hint:
(1) Add profile.
(2) Right-click on the profile in the list to add binning line. A line will be added automatically.
(3) If you want, edit the line on the screen: add node with Ctrl+left-click, move node with right mouse button, delete node with right double-click.
(4) When ready, right click on the line in the list and apply binning to dataset.

Crooked Line 2D Binning module window with created binning line

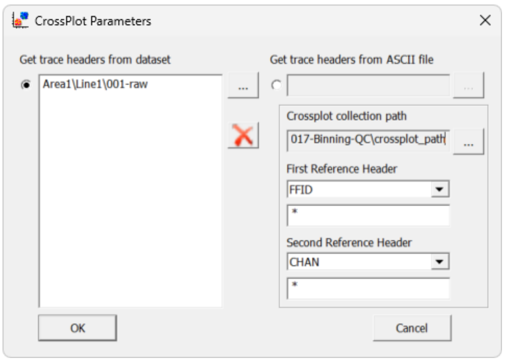
Upon successful completion, the following window with binning statistics appears. Next, we can save the progress (*File => Save scheme*) and exit Crooked Line 2D Binning*.



CDP successfully assigned to
Traces: 142704 of 142704

Statistics of CDP binning

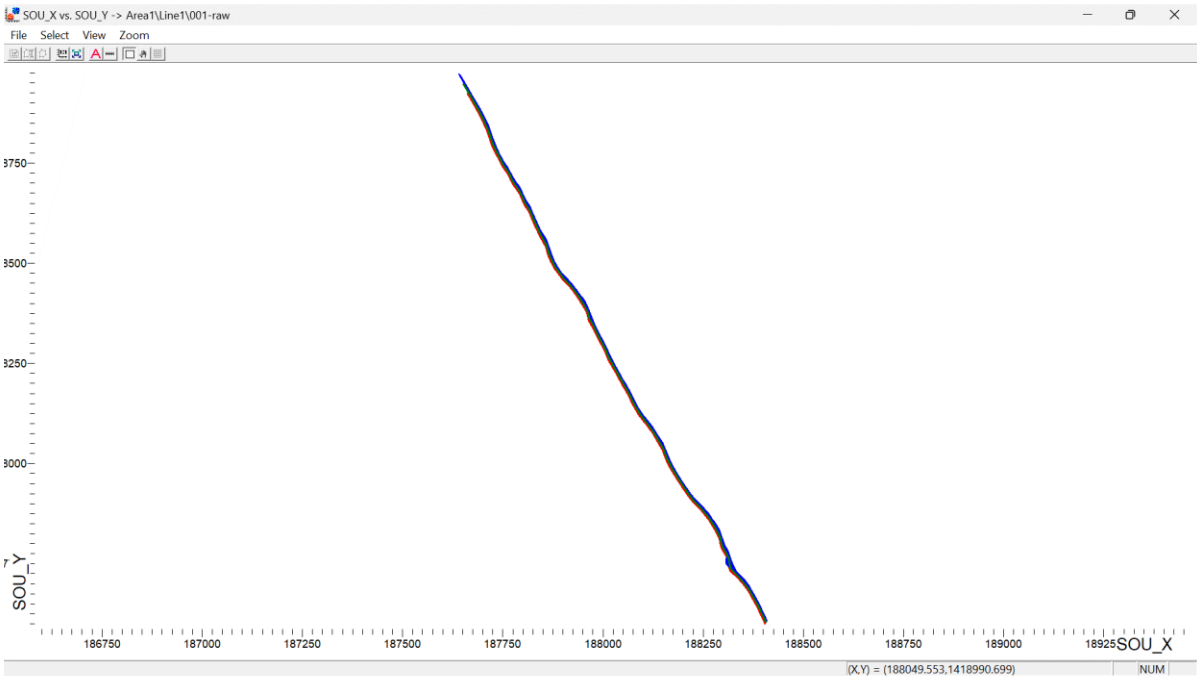
The quality control of the binning takes place in **017-Binning-QC**. In this flow, we use the standalone CrossPlot* module to plot sources, receivers and bin centers. In the provided project, the crossplot is already set up. Here is the workflow to set it up from the outset. In the module parameters, we choose the *001-raw* dataset for plotting.



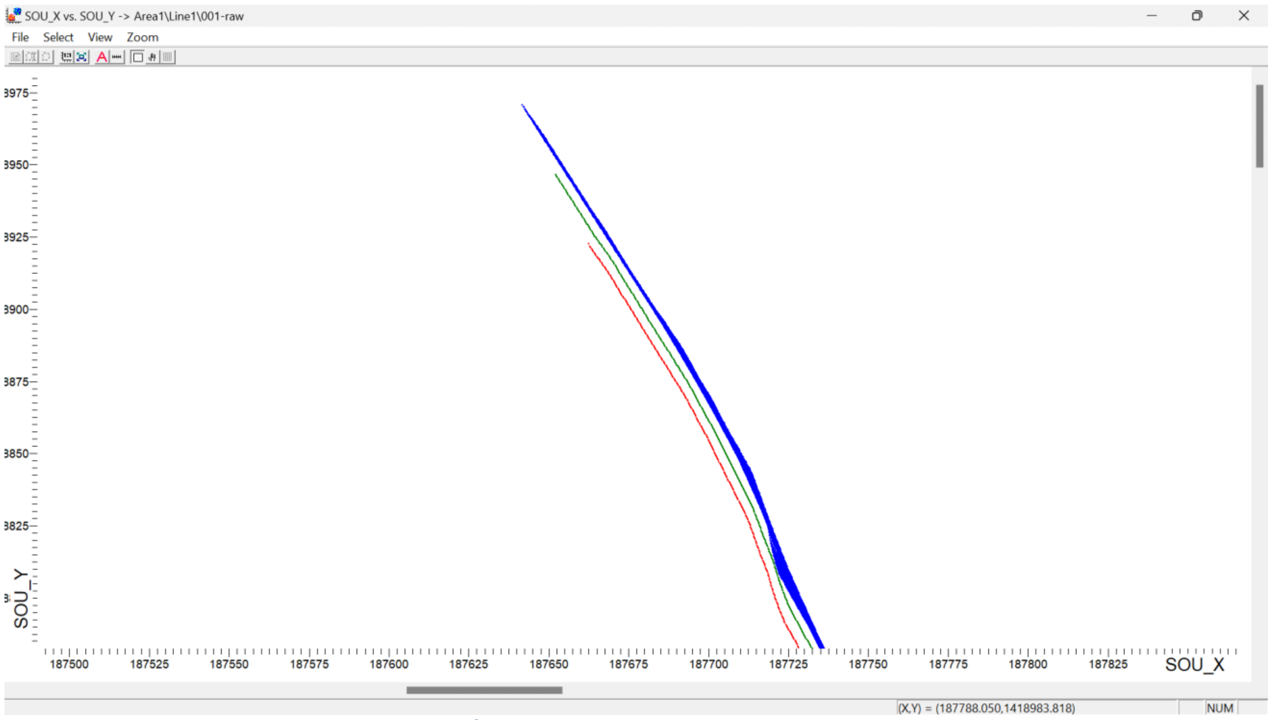
Get trace headers from dataset
● Area1\Line1\001-raw
Get trace headers from ASCII file
Crossplot collection path
017-Binning-QC\crossplot_path
First Reference Header
FFID
*
Second Reference Header
CHAN
*

CrossPlot module parameters

When launching the module, we set the new crossplot collection and add the new SOU_X – SOU_Y crossplot to display the sources (picking the red point color). In the crossplot window, by clicking *View => Extra headers*, we add the receiver points (REC_X – REC_Y) in blue and CDP points (CDP_X – CDP_Y) in green. When examining the resulting plot, we can observe than the CDPs are located between the source line and the receiver locations, we can also observe the streamer feathering.

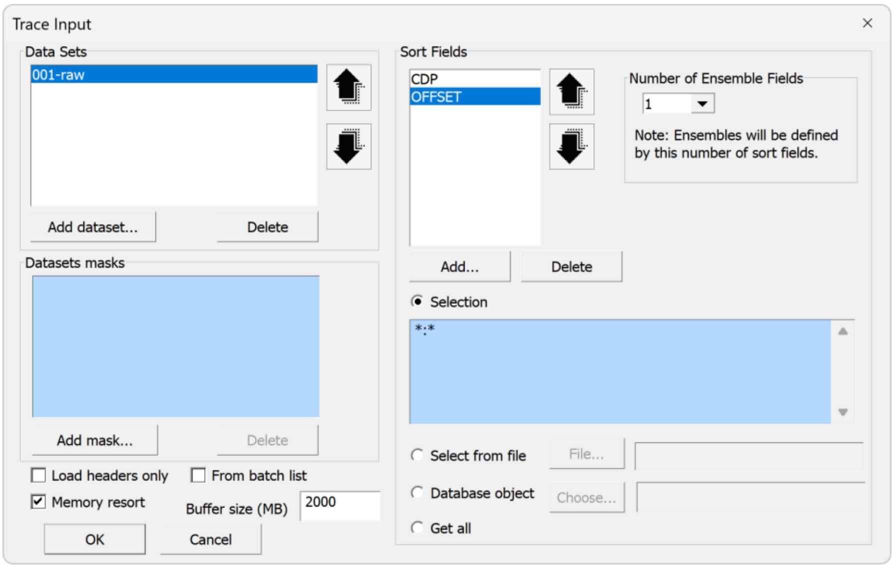


A plot of sources, receivers and CDPs



A plot of sources, receivers and CDPs (zoomed in)

Brute stack computation is the final step of the geometry QC. In the flow **019-Brute-stack**, the Trace Input module inputs the *001-raw* dataset into the flow in CDP:OFFSET sorting. Note that *Memory* *resort* is turned on for faster sorting. The *Buffer size* may be changed depending on the specifications of the computer running the software. In many cases, *Memory resort* increases the sorting speed significantly.


Trace Input

Data Sets: 001-raw

Add dataset... Delete

Datasets masks

Add mask... Delete

☐ Load headers only ☐ From batch list

☑ Memory resort Buffer size (MB) 2000

OK Cancel

Sort Fields: CDP, OFFSET

Add... Delete

Number of Ensemble Fields: 1

Note: Ensembles will be defined by this number of sort fields.

(●) Selection: *:*

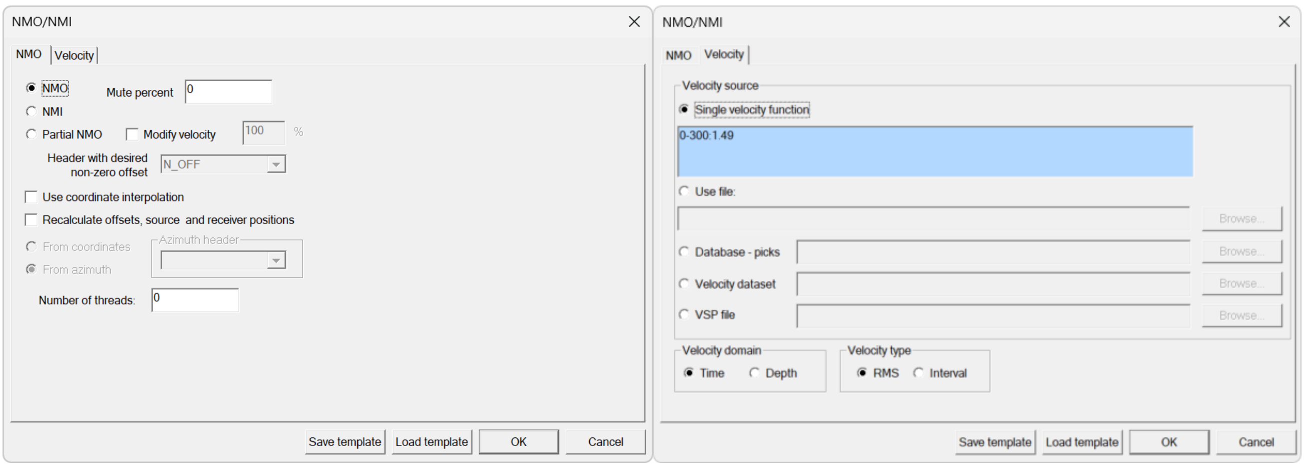
( ) Select from file File...

( ) Database object Choose...

( ) Get all

Trace Input parameters for brute stack computation

Next, the Data Filter module removes all the traces with CDP=-1. This is a safety measure. The Crooked Line 2D Binning* module assigns CDP=-1 to the traces with the midpoints which do not fit in any bin. In our case, all the traces have correct bin numbers, however, if we chose a narrower swath range when binning, a few traces could have missed the binning line, and we would need to remove such traces before stacking. Next, we apply the NMO/NMI module in NMO mode to introduce the normal moveout correction into the CDPs using the water velocity of 1.49 km/s. The traces in each CDP are then stacked with Ensemble Stack with default parameters and output to *001-raw-stack* dataset with Trace Output.


NMO/NMI

NMO | Velocity

(●) NMO Mute percent 0

( ) NMI

( ) Partial NMO ☐ Modify velocity 100 %

Header with desired non-zero offset N_OFF

☐ Use coordinate interpolation

☐ Recalculate offsets, source and receiver positions

( ) From coordinates (●) From azimuth Azimuth header

Number of threads: 0

Save template Load template OK Cancel

NMO/NMI

NMO | Velocity

Velocity source

(●) Single velocity function: 0-300:1.49

( ) Use file: Browse...

( ) Database - picks Browse...

( ) Velocity dataset Browse...

( ) VSP file Browse...

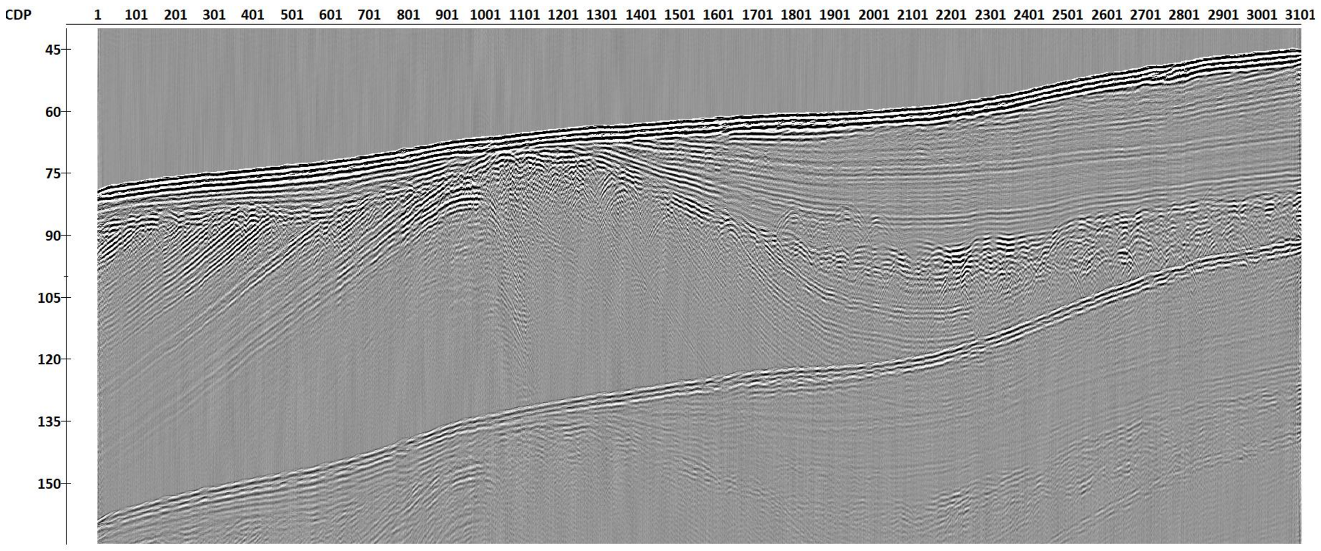
Velocity domain: (●) Time ( ) Depth

Velocity type: (●) RMS ( ) Interval

Save template Load template OK Cancel

NMO/NMI parameters

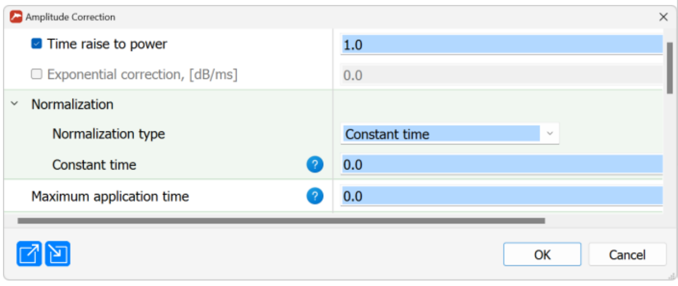
When plotting the dataset with Screen Display in **019-Brute-stack-QC**, we observe the coherent reflections, as expected. Later, we will use stacking flows similar to this one to compute stacks at other processing stages. At some point, we will replace the water velocities with the velocity analysis results.



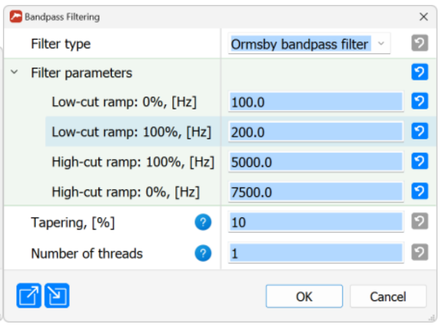
Brute stack

Preprocessing

Data preprocessing occurs in **020-Preprocessing**. For this dataset, data preprocessing consists of two procedures – Amplitude Correction and Bandpass Filtering. Amplitude Correction multiplies the traces' amplitudes by time, and Bandpass Filtering applies a trapezoidal (Ormsby) filter with characteristic frequencies of 100-200-5000-7500 Hz. To decide on the frequencies of the filter, one can use the spectrum estimation tool (,) in Screen Display or Seismic Display. The flow then outputs the preprocessed data to the dataset *020-preproc*.


Amplitude Correction

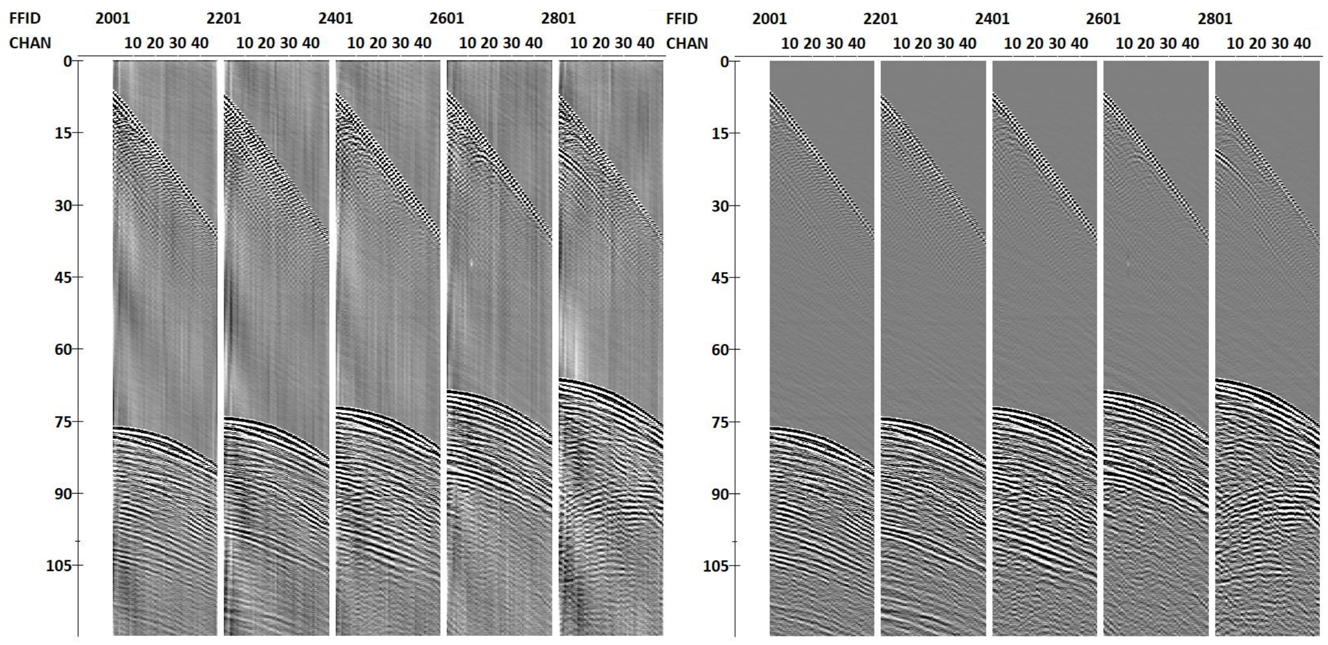
☑ Time raise to power	1.0
☐ Exponential correction, [dB/ms]	0.0
Normalization	
Normalization type	Constant time
Constant time	0.0
Maximum application time	0.0


Bandpass Filtering

Filter type	Ormsby bandpass filter
Filter parameters	
Low-cut ramp: 0%, [Hz]	100.0
Low-cut ramp: 100%, [Hz]	200.0
High-cut ramp: 100%, [Hz]	5000.0
High-cut ramp: 0%, [Hz]	7500.0
Tapering, [%]	10
Number of threads	1

Amplitude Correction (left) and Bandpass Filtering (right) parameters

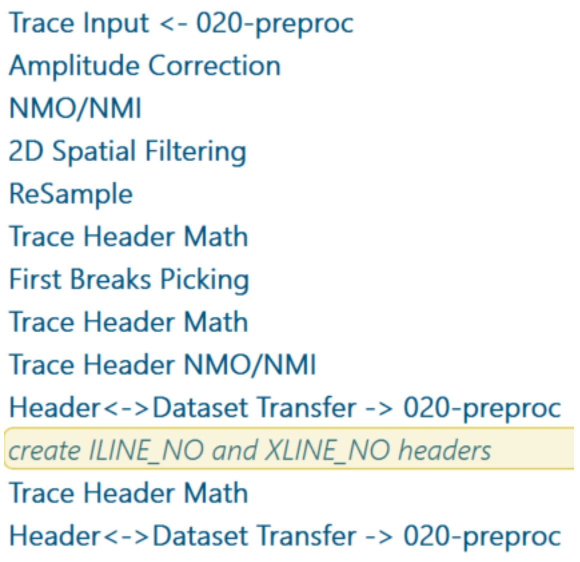
In **021-Preprocessing-QC**, one can plot and compare the shot gathers before and after preprocessing, and the flow **022-Preprocessing-stack** computes the stack after preprocessing.



Common-shot gathers before (left) and after (right) preprocessing

Seafloor reflection picking

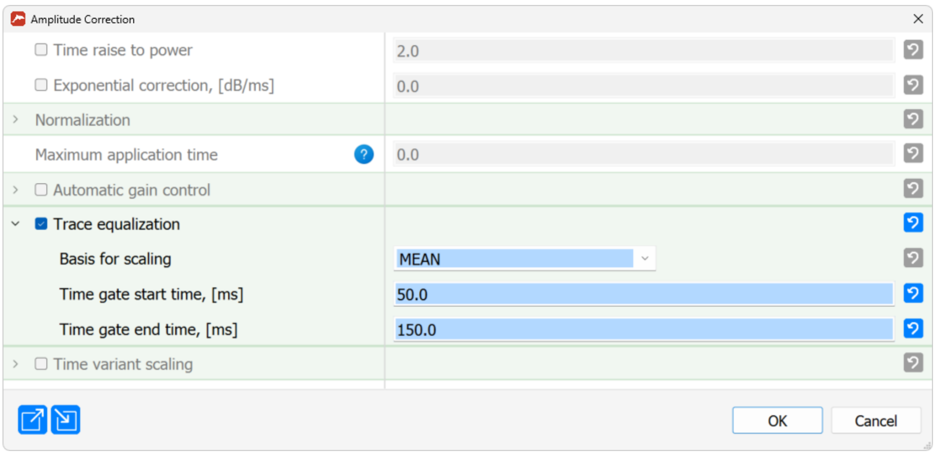
The next step of the workflow involves picking the seafloor reflection on all the seismic traces, which will be used in the high-resolution statics. The picking occurs automatically in the flow **023-Picking**.



Trace Input <- 020-preproc
Amplitude Correction
NMO/NMI
2D Spatial Filtering
ReSample
Trace Header Math
First Breaks Picking
Trace Header Math
Trace Header NMO/NMI
Header<->Dataset Transfer -> 020-preproc
create ILINE_NO and XLINE_NO headers
Trace Header Math
Header<->Dataset Transfer -> 020-preproc

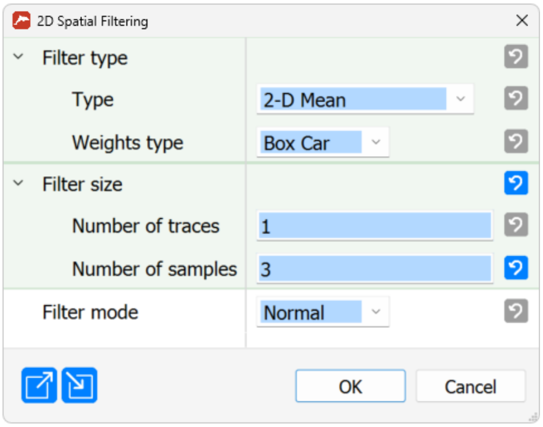
023-Picking flow

The flow first inputs the *020-preproc* dataset in CHAN:FFID sorting (as it is easier to pick the reflection along continuous common-channel gathers). Next, a trace equalization occurs in the Amplitude Correction module (note that this is a preconditioning step only for picking, the modified data will not be written anywhere), NMO/NMI introduces the normal moveout correction, and 2D Spatial Filtering smooths the data slightly along the trace axis (with a 3-point filter).


Amplitude Correction

☐ Time raise to power	2.0
☐ Exponential correction, [dB/ms]	0.0
Normalization	
Maximum application time	0.0
☐ Automatic gain control	
☑ Trace equalization	
Basis for scaling	MEAN
Time gate start time, [ms]	50.0
Time gate end time, [ms]	150.0
☐ Time variant scaling	

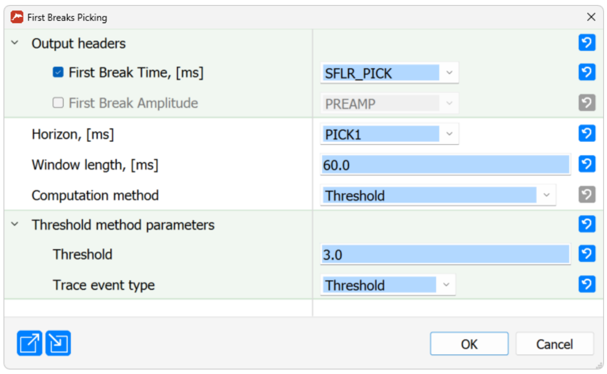
Amplitude Correction parameters for trace equalization


2D Spatial Filtering

Filter type	
Type	2-D Mean
Weights type	Box Car
Filter size	
Number of traces	1
Number of samples	3
Filter mode	Normal

2D Spatial Filtering parameters

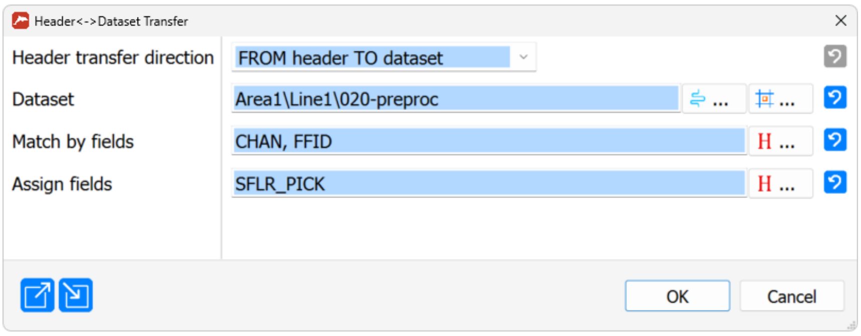
The data are then resampled to 0.01 ms with ReSample (this improves picking accuracy, as the following First Break Picking snaps the pick to the nearest sample). The First Break Picking module picks the first amplitude specified by the Threshold parameter in the 60 ms window starting from PICK1 header time (which was previously set to 40 ms in Trace Header Math) and saves its time to the SFLR_PICK header.


First Breaks Picking

Output headers	
☑ First Break Time, [ms]	SFLR_PICK
☐ First Break Amplitude	PREAMP
Horizon, [ms]	PICK1
Window length, [ms]	60.0
Computation method	Threshold
Threshold method parameters	
Threshold	3.0
Trace event type	Threshold

First Break Picking parameters

Next, the Trace Header Math module shifts the SFLR_PICK slightly to the top (to make sure it is located at the start of the wavelet). The Trace Header NMO/NMI introduces inverse moveout correction into the SFLR_PICK header with the same water velocity which was used previously. This finalizes the picking. The SFLR_PICK header containing the picked traveltimes is then saved to the *020-preproc* dataset with the Header <-> Dataset Transfer using the headers CHAN and FFID for matching.

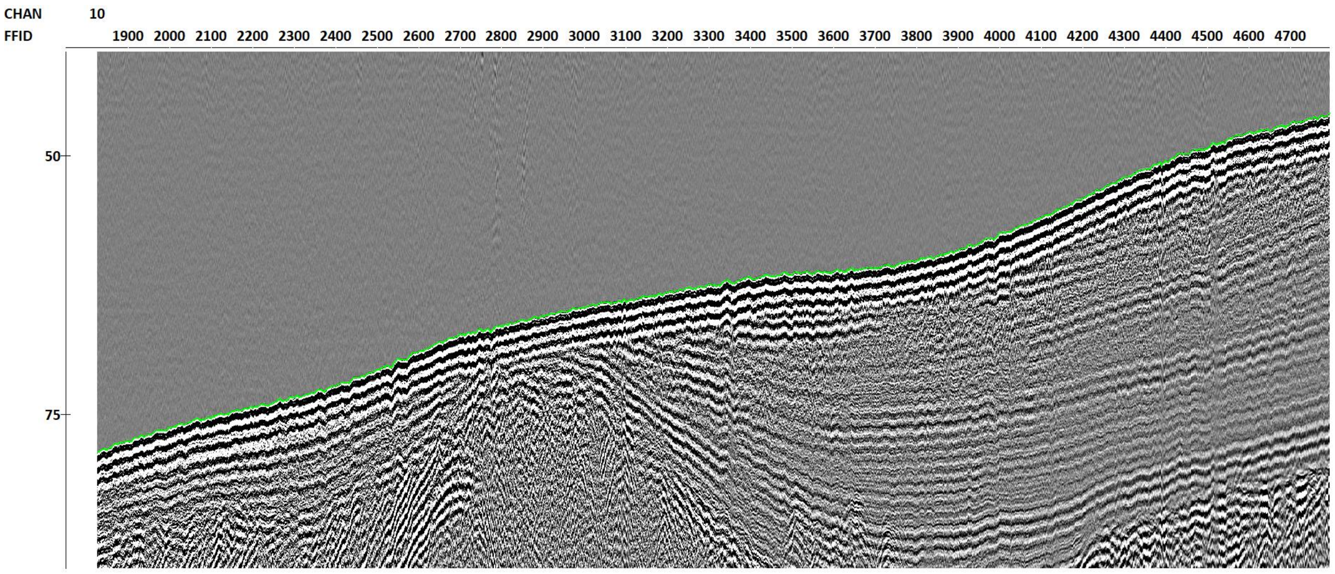

Header<->Dataset Transfer

Header transfer direction	FROM header TO dataset
Dataset	Area1\Line1\020-preproc
Match by fields	CHAN, FFID
Assign fields	SFLR_PICK

Header <-> Dataset Transfer parameters for saving the SFLR_PICK header to 020-preproc

In addition to this, the flow fills ILINE_NO=1 and XLINE_NO=CDP headers (used by some modules in the following flows) and transfers them to *020-preproc* in a similar way.

The quality control of the picking takes place in **024-Picking-QC**, where the picks are overlayed on common-channel gathers.

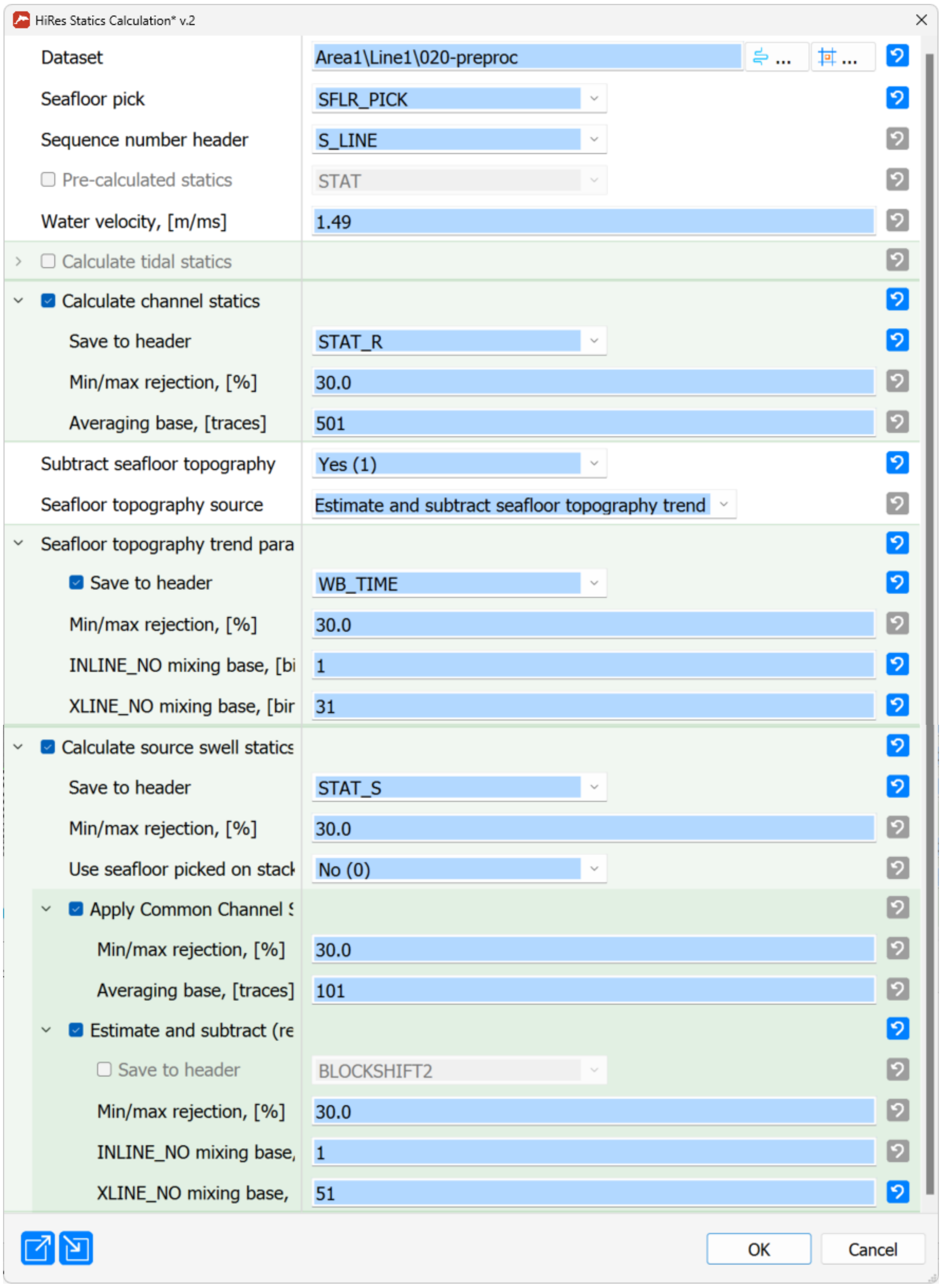


A common-channel gather with the picked seafloor reflection

High-Resolution Statics

Next, in the flow **025-Statics-Calc** we run the HiRes Statics Calculation* v.2 module, which uses the previously picked seafloor reflection to deduce the vertical movement of the sources and receivers and correct for that movement. The parameters of the module are shown below. The static corrections are inferred from the SFLR_PICK header in the *020-preproc* dataset and are written to STAT_R (for receivers) and STAT_S (for sources) headers in the same dataset.

Water velocity equal to 1.49 is used for the computations. The averaging/mixing parameters are mostly left as default or chosen with trial and error. The option *Subtract seafloor topography* is turned on with *Estimate and subtract seafloor topography trend* as *Seafloor topography source*. These options estimate the seafloor topography directly from the picked traveltimes and use it in the computation of static corrections. The seafloor trend is saved to WB_TIME and can later be used to set up processing windows, or as an approximation of the true seafloor shape, etc. The receiver statics are saved to STAT_R, the source statics are also computed and saved in STAT_S (the presence of source statics is the main advantage of the second version of the HiRes Statics Calculation* module). Launching this flow writes the corrections to the headers in *020-preproc*.



Dataset	Area1\Line1\020-preproc
Seafloor pick	SFLR_PICK
Sequence number header	S_LINE
☐ Pre-calculated statics	STAT
Water velocity, [m/ms]	1.49
☐ Calculate tidal statics	
☑ Calculate channel statics	
Save to header	STAT_R
Min/max rejection, [%]	30.0
Averaging base, [traces]	501
Subtract seafloor topography	Yes (1)
Seafloor topography source	Estimate and subtract seafloor topography trend
Seafloor topography trend para	
☑ Save to header	WB_TIME
Min/max rejection, [%]	30.0
INLINE_NO mixing base, [bi	1
XLINE_NO mixing base, [bir	31
☑ Calculate source swell statics	
Save to header	STAT_S
Min/max rejection, [%]	30.0
Use seafloor picked on stack	No (0)
☑ Apply Common Channel S	
Min/max rejection, [%]	30.0
Averaging base, [traces]	101
☑ Estimate and subtract (re	
☐ Save to header	BLOCKSHIFT2
Min/max rejection, [%]	30.0
INLINE_NO mixing base,	1
XLINE_NO mixing base,	51

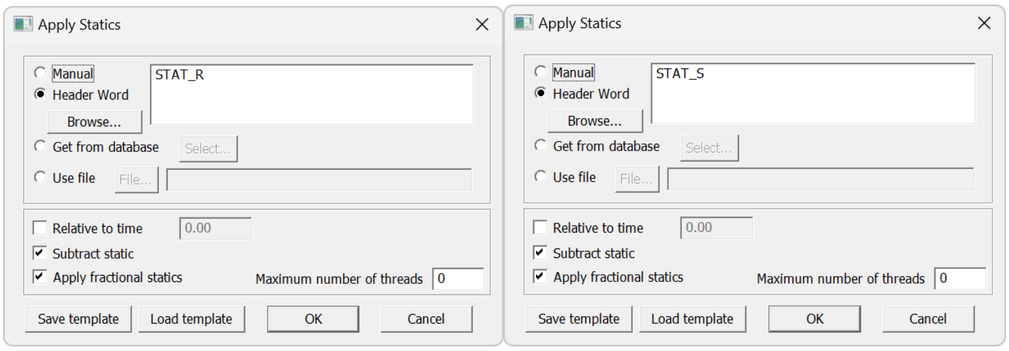
HiRes Statics Calculation v.2 parameters*

The application of the computed statics occurs in the **026-Statics-Apply** flow.

Trace Input <- 020-preproc
NMO/NMI
Apply Statics <- [STAT_R]
Apply Statics <- [STAT_S]
NMO/NMI
Trace Header NMO/NMI
Trace Header Math
Trace Header NMO/NMI
Trace Header Math
Trace Editing <- [TOP_MUTE]
Trace Output -> 026-stat

026-Statics-Apply flow

The *020-preproc* data are input in CDP:OFFSET sort (the sort does not matter here, but we use the opportunity to resort the data for the following velocity analysis and demultiple processing steps). The static corrections are to be applied with NMO correction introduced, so the NMO/NMI module first conducts the NMO before statics (with 1.49 km/s velocity) and then removes it after the static corrections are introduced. The Apply Statics modules introduce the previously computed corrections from STAT_R and STAT_S headers (*Subtract static* and *Apply fractional statics* are turned on).


Apply Statics

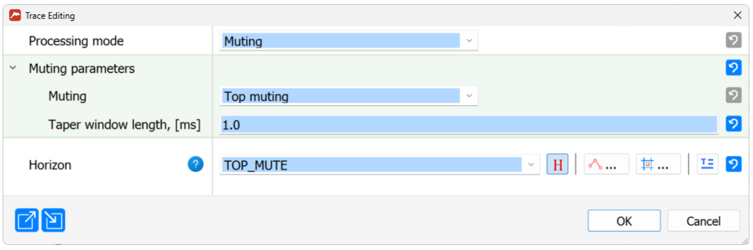
○ Manual
● Header Word: STAT_R
Browse...
○ Get from database — Select...
○ Use file — File...
☐ Relative to time 0.00
☑ Subtract static
☑ Apply fractional statics
Maximum number of threads 0

Apply Statics

○ Manual
● Header Word: STAT_S
Browse...
○ Get from database — Select...
○ Use file — File...
☐ Relative to time 0.00
☑ Subtract static
☑ Apply fractional statics
Maximum number of threads 0

Apply Statics parameters

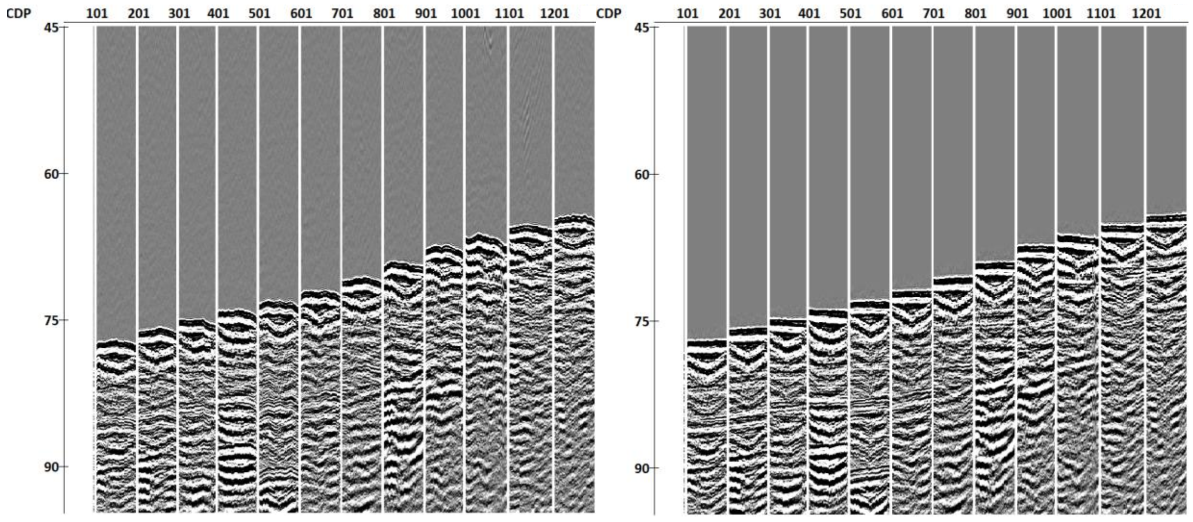
In the Trace Header Math module surrounded by two Trace Header NMO/NMI modules, we introduce the same corrections into the SFLR_PICK header (to avoid repeating the picking) as SFLR_PICK = SFLR_PICK - STAT_R - STAT_S. The next Trace Header Math computes the TOP_MUTE header as SFLR_PICK – 1.5, which is then used for top muting in Trace Editing (in order to remove the noise before the first arrivals). Finally, the corrected data are written to *026-stat*.


Trace Editing

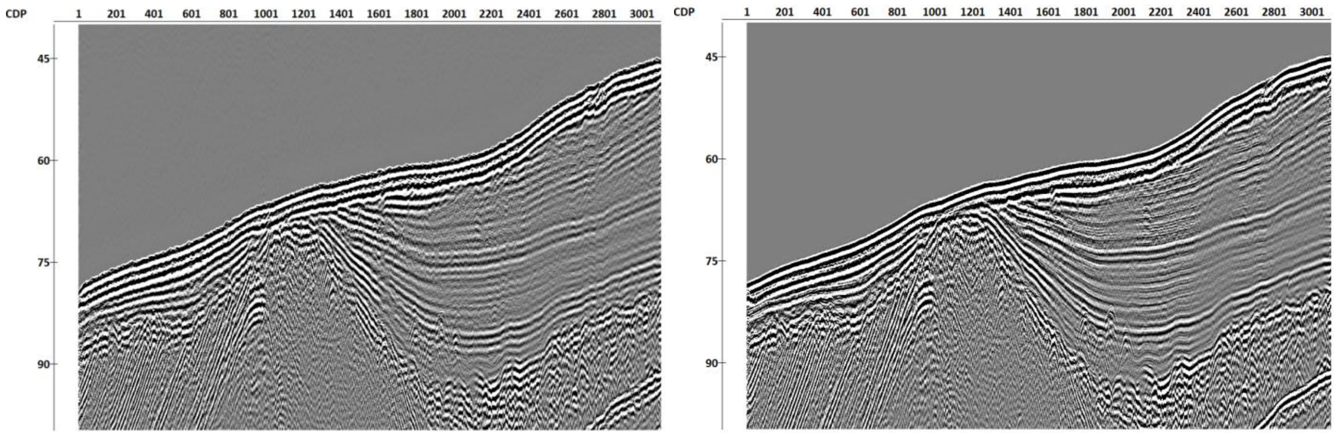
Processing mode	Muting
Muting parameters	
Muting	Top muting
Taper window length, [ms]	1.0
Horizon	TOP_MUTE

Trace Editing parameters for top muting

026-Statics-Apply-QC provides the quality control displays for statics by comparing the CDP gathers with introduced NMO corrections before and after statics. The flow **027-Statics-stack** computes the stack after statics computation, **028-Statics-stack-QC** plots the stacks before and after statics. On the CDP gathers, the variations in the seafloor reflection time are removed after statics. This leads to a more coherent and smooth seafloor reflection on the stack.



NMO-corrected CDP gathers before (left) and after (right) statics

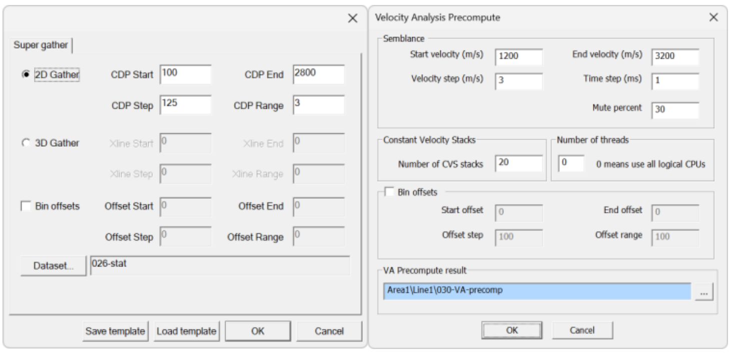


Stacks before (left) and after (right) statics

Velocity analysis

The following flows are dedicated to velocity analysis. The velocity analysis in the proposed workflow occurs two times – before demultiple and before final stacking/migration. We provide one (final) velocity model in the project and use it in all the relevant flows starting from this one.

A conventional way to conduct velocity analysis in **RadExPro** is first to precompute the semblance for all the relevant CDPs using Velocity Analysis Precompute and then use this precomputed data in the Interactive Velocity Analysis module. The flow **030-Velocity-Analysis-Precomp** precomputes the semblance. It consists of three modules: Super Gather, Amplitude Correction, and Velocity Analysis Precompute. Super Gather computes the CDP supergathers from the *026-stat* dataset with the intervals specified by *CDP Start*, *End*, *Step* and *Range*. Amplitude Correction introduces Automatic Gain Control (AGC) with 3 ms window to improve the computed semblances. Finally, Velocity Analysis Precompute computes the semblances and the CVS stacks in the specified velocity range and saves the result to the *030-VA-precomp* object.



Super gather

(● 2D Gather) CDP Start 100 CDP End 2800
CDP Step 125 CDP Range 3

○ 3D Gather Xline Start 0 Xline End 0
Xline Step 0 Xline Range 0

☐ Bin offsets Offset Start 0 Offset End 0
Offset Step 0 Offset Range 0

Dataset... 026-stat

Save template | Load template | OK | Cancel

Velocity Analysis Precompute

Semblance
Start velocity (m/s) 1200 End velocity (m/s) 3200
Velocity step (m/s) 3 Time step (ms) 1
Mute percent 30

Constant Velocity Stacks
Number of CVS stacks 20

Number of threads
0 0 means use all logical CPUs

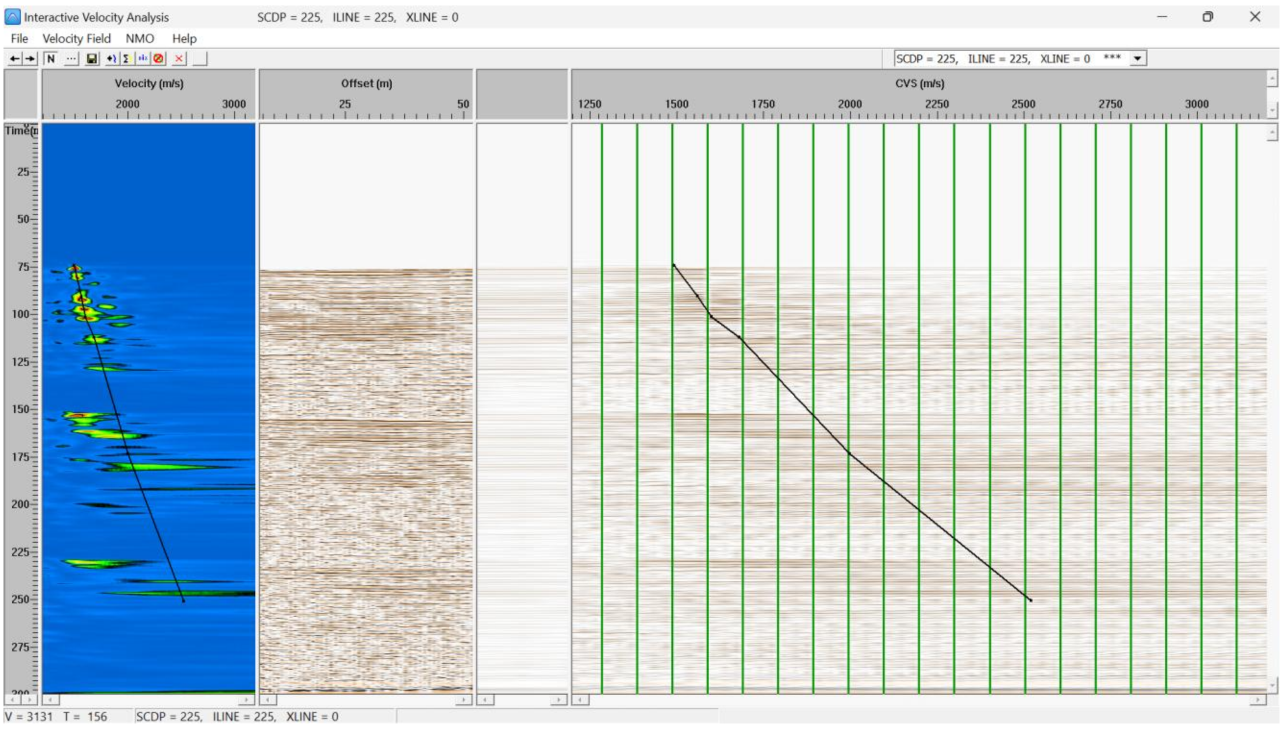
☐ Bin offsets
Start offset 0 End offset 0
Offset step 100 Offset range 100

VA Precompute result
Area1\Line1\030-VA-precomp

OK | Cancel

Super Gather and Velocity Analysis Precompute parameters

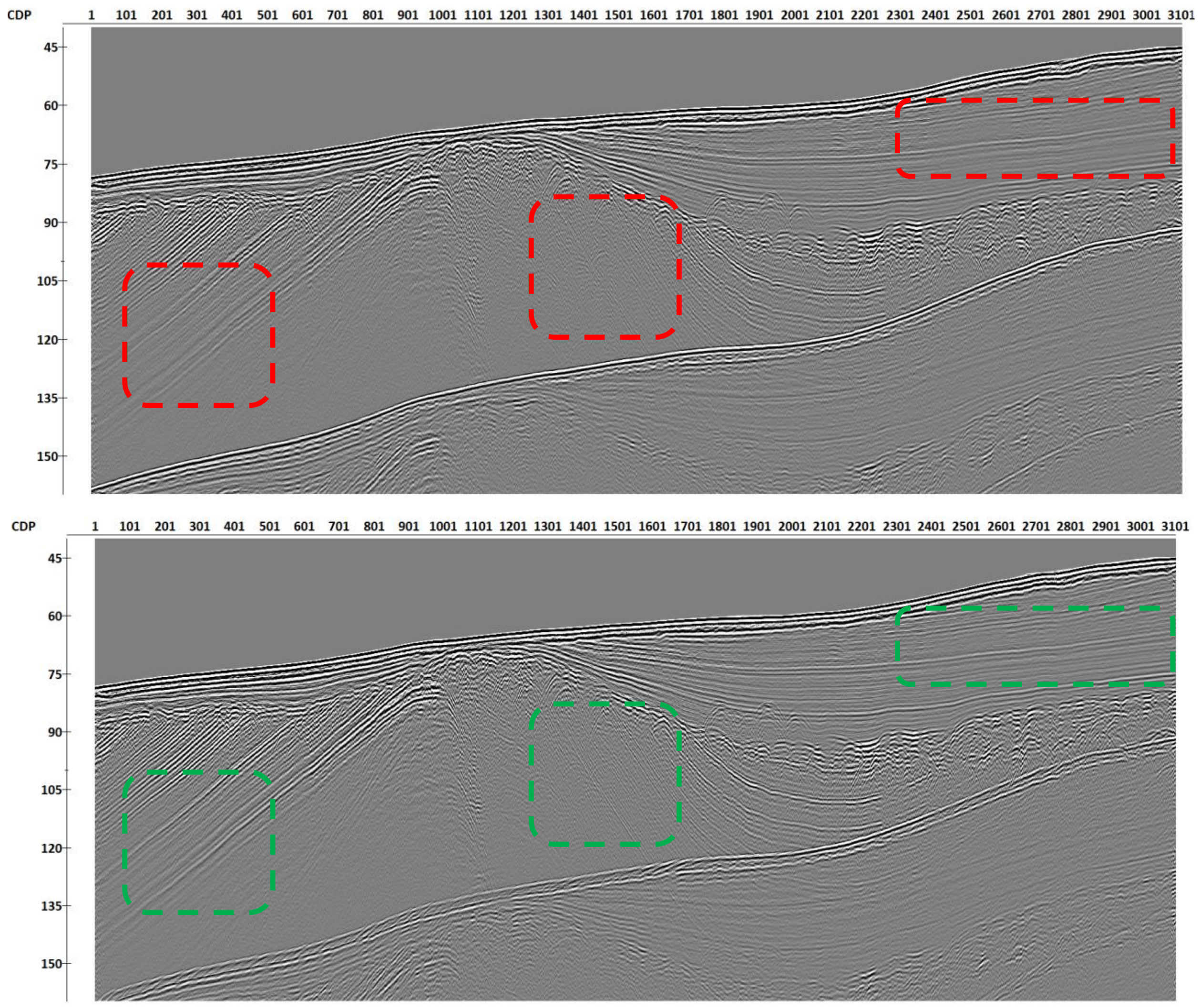
After running the precompute flow, one can start conducting the velocity analysis in **031-Velocity Analysis**. This flow runs the Interactive Velocity Analysis module in the standalone mode (the precomputed *030-VA-precomp* object is specified on the *Semblance* window of the parameters). For detailed explanation of the Interactive Velocity Analysis parameters, we refer to the manual of the module. In the interactive module window displayed below, one needs to pick the maxima by left-clicking on them on the semblance display on the left, while controlling the accuracy of NMO in the gather display and the stacking quality in the rightmost CVS window. This needs to be done for every supergather, which results in the RMS velocity model *vel_RMS* (final *vel_RMS* is already provided in the project, so there is no need to pick anything to reproduce the following results).


Interactive Velocity Analysis
SCDP = 225, ILINE = 225, XLINE = 0
File Velocity Field NMO Help
SCDP = 225, ILINE = 225, XLINE = 0 ***
Velocity (m/s)
2000
3000
Offset (m)
25
50
CVS (m/s)
1250
1500
1750
2000
2250
2500
2750
3000
Time
25
50
75
100
125
150
175
200
225
250
275
V = 3131 T = 156
SCDP = 225, ILINE = 225, XLINE = 0

Interactive Velocity Analysis module window

The flow **032-Velocity-Analysis-stack** computes the stack with the new velocities from the *026-stat* dataset. The only difference in this flow from previous stacking flows is the *vel_RMS* velocity model specified on the *Velocity* tab of the NMO/NMI module.

The flow **033-Velocity-Analysis-QC** compares the stacks computed with constant velocity and with the velocities from the velocity analysis. The improvement in the horizon continuity can be seen in the image below.



Stacks before (top) and after (bottom) velocity analysis

Multiple removal

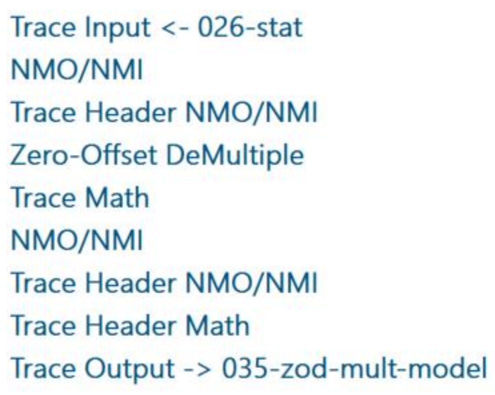
Multiple removal is one of the most computationally expensive steps of this workflow. There are two main methods for multiple subtraction in **RadExPro**: Zero-Offset DeMultiple and SRME. Depending on the dataset, one of these methods might be more successful at removing multiples than the other. In this tutorial, we explain both multiple removal techniques and discuss their differences. There is also a flow which demonstrates joint subtraction of the Zero-Offset DeMultiple and SRME models.

Note that the preferred order of multiple removal and deghosting procedures may change for other datasets. Here, we found that multiple removal before deghosting is preferable. However, the ghost waves complicate the SRME prediction, so, in other cases, deghosting before multiple removal may be viable.

Zero-Offset DeMultiple

The simplest multiple subtraction technique is Zero-Offset DeMultiple. It computes the model of the multiples by either shifting the original traces by a time taken from a specified header, or by auto-convolving the traces. The model can then be adaptively subtracted directly inside the Zero-Offset DeMultiple module or exported to a dataset for the later adaptive subtraction by Wave Field Subtraction. Here, we will first follow the second (slightly longer) approach and then present a way to obtain a similar result within one Zero-Offset DeMultiple module.

The flow **035-ZOD-model** creates a model of multiple waves using Zero-Offset DeMultiple. The flow for the creation of the model is shown below.

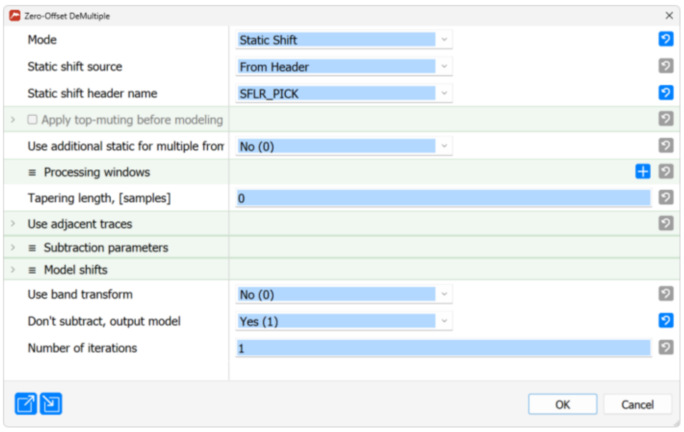


Trace Input <- 026-stat
NMO/NMI
Trace Header NMO/NMI
Zero-Offset DeMultiple
Trace Math
NMO/NMI
Trace Header NMO/NMI
Trace Header Math
Trace Output -> 035-zod-mult-model

035-ZOD-model flow

Trace Input loads the data in CDP:OFFSET sorting (sorting is not very important here, but the original data is in CDP:OFFSET, so we are not spending any extra time on sorting). The following NMO/NMI module introduces the normal moveout correction with the water velocity of 1.49 km/s, Trace Header NMO/NMI conducts the same for the picked seafloor SFLR_PICK header.

Next, the Zero-Offset DeMultiple conducts the modeling of multiples. The selected mode here is *Static Shift*, which uses the picked SFLR_PICK to shift the traces to the times of the multiple waves (depending on the accuracy of picking, it may be preferable to use the seafloor trend from static corrections WB_TIME for this). We do not need to specify any subtraction parameters, as the option *Don't subtract, output model* directly outputs the shifted traces into the flow without subtraction. Next, we multiply the traces by -1 in Trace Math to take into account the negative sign of the sea surface reflection coefficient and reverse the previously applied NMO/NMI and Trace Header NMO/NMI. Finally, the Trace Header Math provides an ID number to this model of multiples. This number is later used in subtraction for proper input data sorting. The model is saved to the dataset *035-zod-mult-model*.

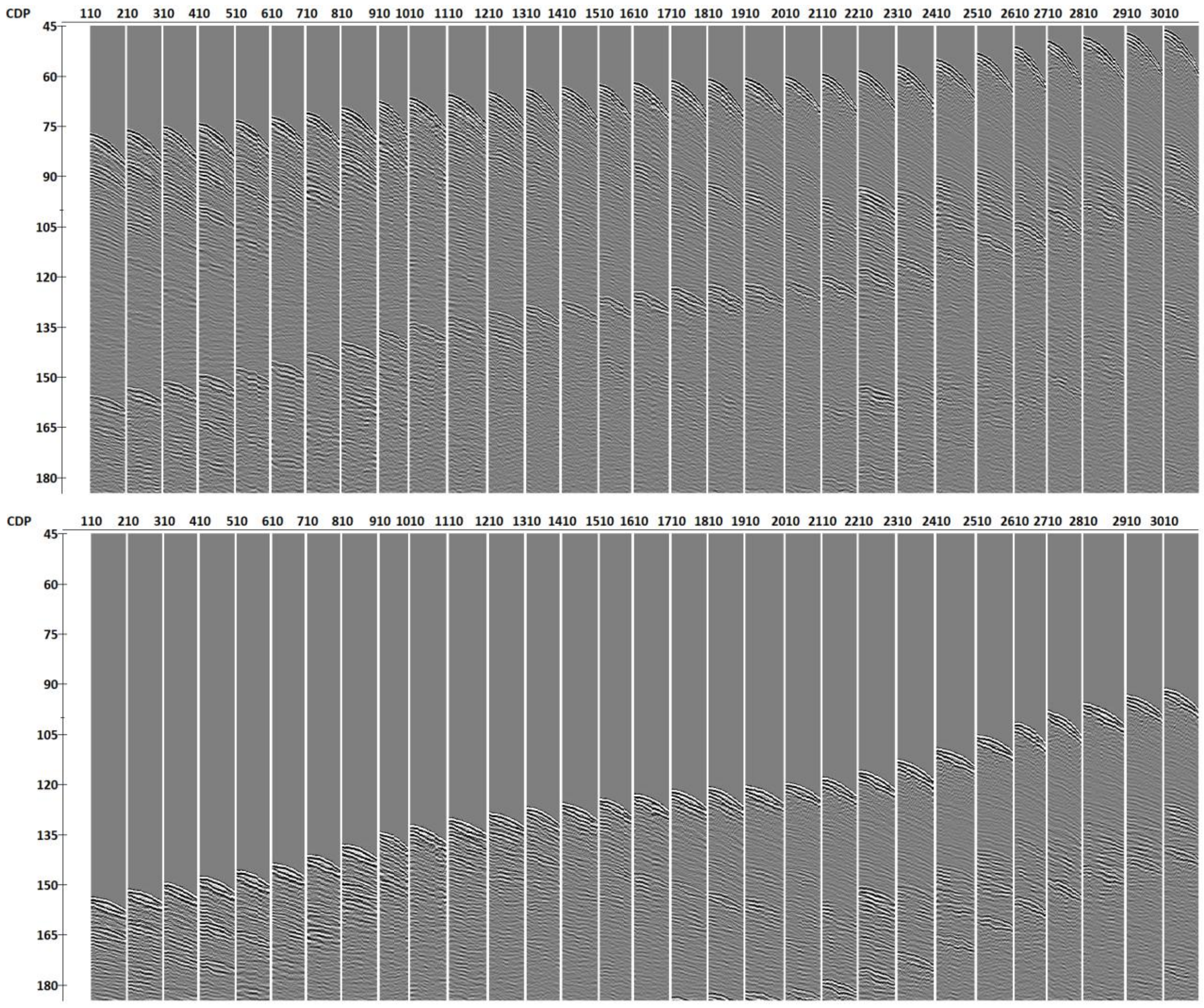


Zero-Offset DeMultiple

Mode	Static Shift
Static shift source	From Header
Static shift header name	SFLR_PICK
☐ Apply top-muting before modeling	
Use additional static for multiple fron	No (0)
≡ Processing windows	
Tapering length, [samples]	0
Use adjacent traces	
≡ Subtraction parameters	
≡ Model shifts	
Use band transform	No (0)
Don't subtract, output model	Yes (1)
Number of iterations	1

Zero-Offset DeMultiple parameters

The quality control of this model of multiples occurs in **035-ZOD-model-QC**, where the model of multiples is compared to the original CDP gathers. In the Figure below, one can observe that the model of multiples fits the multiples on the gathers quite well. There is a slight shift between the model and the true multiples. This shift occurs due to the inherent 1.5D nature of the method and will be taken into account during the adaptive subtraction.



Input CDP gathers (top) and the corresponding model of Zero Offset DeMultiple-predicted multiples (bottom)

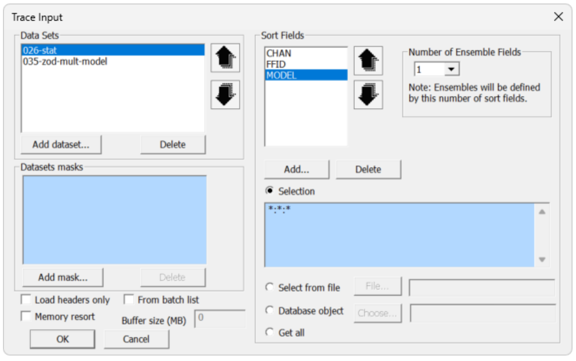
The next step in this multiple removal workflow is adaptive subtraction. It happens in the

036-ZOD-subtr flow.

Trace Input <- [multiple]
Trace Header Math
Trace Header NMO/NMI
Trace Header NMO/NMI
Trace Header NMO/NMI
Trace Header NMO/NMI
Wave Field Subtraction
Trace Header Math
Trace Output -> 036-subtr-zod

036-ZOD-subtr flow

The main module here is Wave Field Subtraction, which conducts adaptive subtraction. It is most effective to run the subtraction in common-channel domain. The requirement of the Wave Field Subtraction module is that the datasets are input as interleaved traces. If there is only one model of multiples to the subtracted, we need to input traces into the flow in the following order: data trace 1, model trace 1, data trace 2, model trace 2, etc. This is done using the Trace Input module with CHAN:FFID:MODEL sorting.



Trace Input

Data Sets: 026-stat (selected), 035-zod-mult-model

Sort Fields: CHAN, FFID, MODEL (selected)

Number of Ensemble Fields: 1

Note: Ensembles will be defined by this number of sort fields.

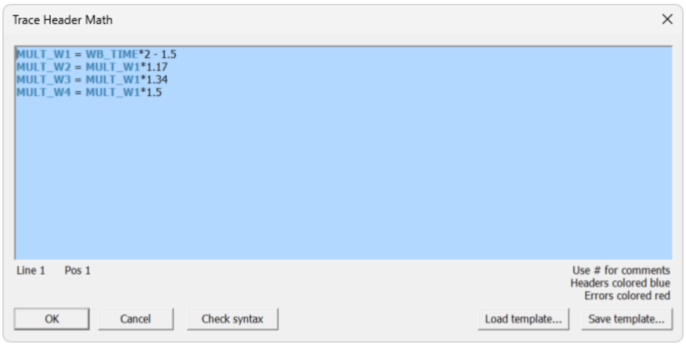
Selection (selected): *:*:*

Load headers only: unchecked
From batch list: unchecked
Memory resort: unchecked
Buffer size (MB): 0

Trace Input parameters suitable for subsequent Wave Field Subtraction

The following Trace Header Math module sets the windows for multiple subtraction. The windows will be displayed later. The header MULT_W1 is approximates the traveltimes of the first multiple at zero offset (as it is computed by multiplying the seafloor time in WB_TIME by 2) with a slight 1.5 ms shift. The header MULT_W4 approximates the traveltimes of the second-order seafloor multiple (WB_TIME*2*1.5= WB_TIME*3), and MULT_W2 and MULT_W3 divide the interval between the first and second-order seafloor multiples evenly. In some cases, one may decide to create more windows for subsequent multiple orders (third, fourth, etc.). We, however, subtract all the multiples starting from the second order seafloor multiple in one window.

The Trace Header NMO/NMI flows following this Trace Header Math introduce inverse NMO correction into the headers MULT_W1, MULT_W2, MULT_W3 and MULT_W4 with water velocity of 1.49 km/s, so that these windows follow the seafloor multiples' traveltimes at nonzero offsets.



Trace Header Math

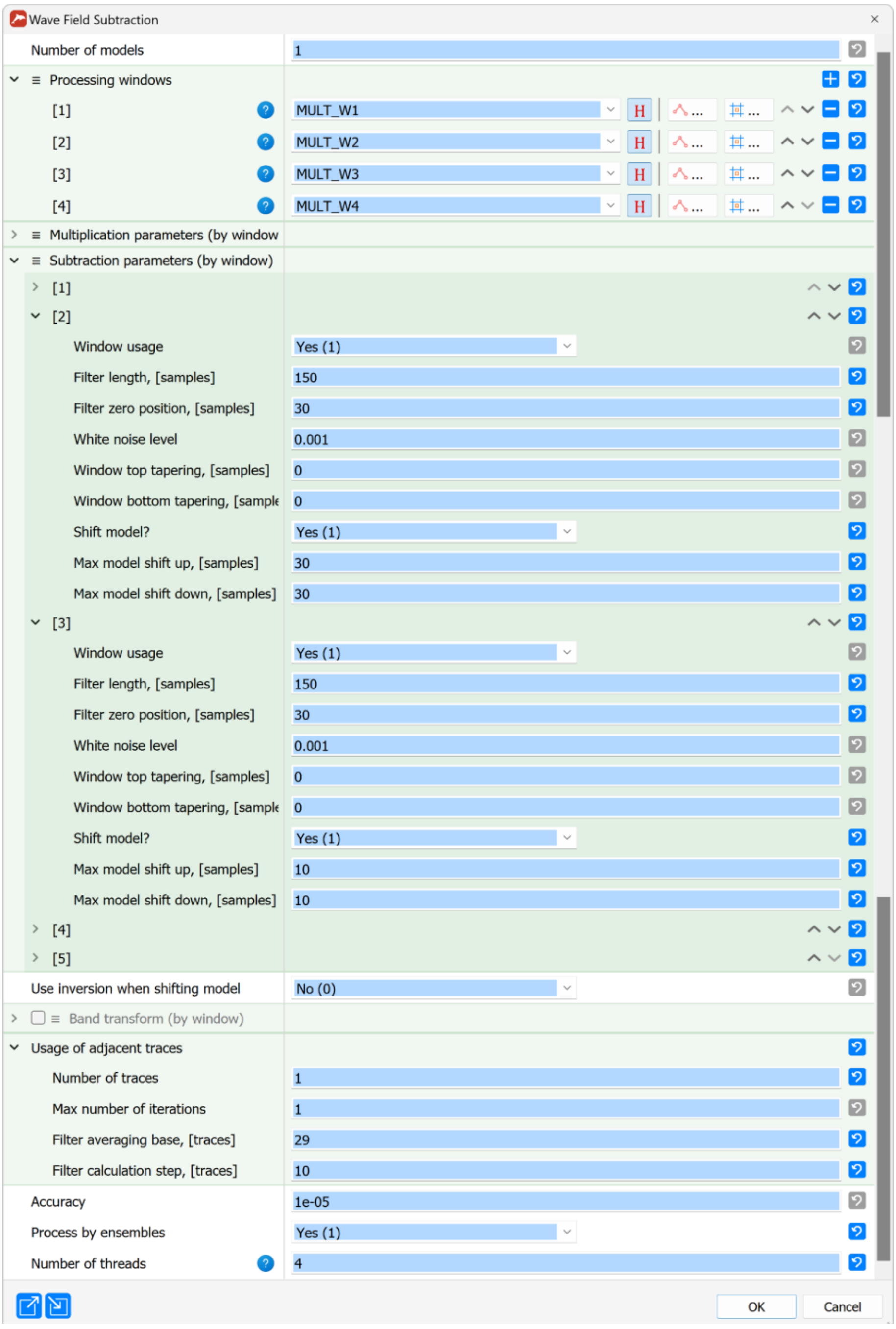
```
MULT_W1 = WB_TIME*2 - 1.5
MULT_W2 = MULT_W1*1.17
MULT_W3 = MULT_W1*1.34
MULT_W4 = MULT_W1*1.5
```

Line 1 Pos 1

Use # for comments
Headers colored blue
Errors colored red

Trace Header Math which sets up Wave Field Subtraction windows

The Wave Field Subtraction module conducts the subtraction. Below are its parameters.



Wave Field Subtraction

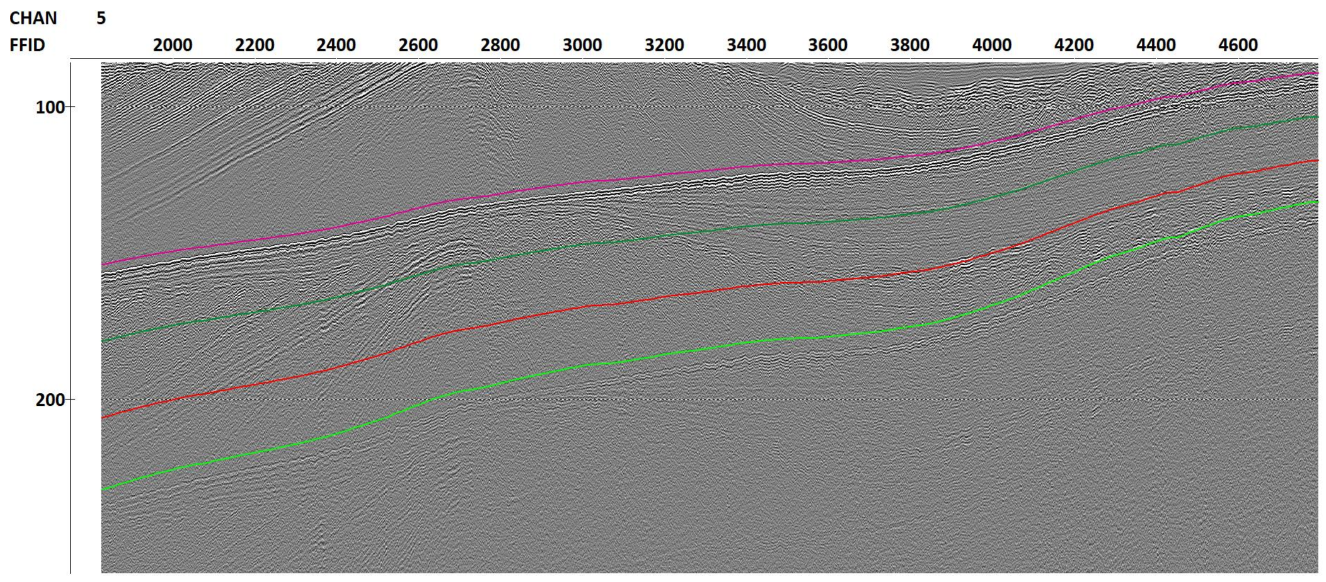
Number of models	1
Processing windows	
[1]	MULT_W1
[2]	MULT_W2
[3]	MULT_W3
[4]	MULT_W4
Multiplication parameters (by window	
Subtraction parameters (by window)	
[1]	
[2]	
Window usage	Yes (1)
Filter length, [samples]	150
Filter zero position, [samples]	30
White noise level	0.001
Window top tapering, [samples]	0
Window bottom tapering, [sample	0
Shift model?	Yes (1)
Max model shift up, [samples]	30
Max model shift down, [samples]	30
[3]	
Window usage	Yes (1)
Filter length, [samples]	150
Filter zero position, [samples]	30
White noise level	0.001
Window top tapering, [samples]	0
Window bottom tapering, [sample	0
Shift model?	Yes (1)
Max model shift up, [samples]	10
Max model shift down, [samples]	10
[4]	
[5]	
Use inversion when shifting model	No (0)
☐ Band transform (by window)	
Usage of adjacent traces	
Number of traces	1
Max number of iterations	1
Filter averaging base, [traces]	29
Filter calculation step, [traces]	10
Accuracy	1e-05
Process by ensembles	Yes (1)
Number of threads	4

Wave Field Subtraction parameters

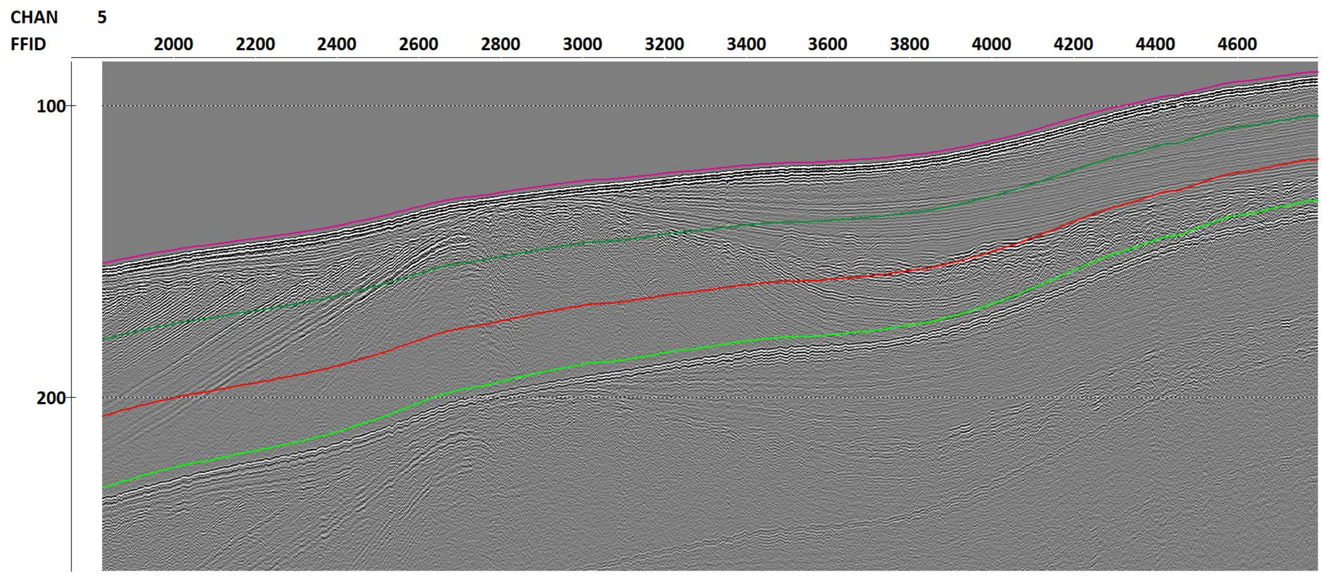
The *Number of models* is set to 1, as only one model (created by Zero-Offset DeMultiple) is being subtracted. Next, the headers MULT_W* created previously are specified in *Processing windows*. Doing this allows us to come up with a separate adaptive filter for each window. *Multiplication parameters* are set to defaults. In *Subtraction parameters*, we specify *Window usage* for window [1] as *No*. The first window is from time 0 to MULT_W1, the second window – from MULT_W1 to MULT_W2, etc. The first window does not contain any multiples. For the second window, by trial and error we set the *Filter length* to 150 and the Filter zero position to 30. We also allow the algorithm to attempt to shift the model by 30 samples down or up before adaptation. Windows [3], [4] and [5] all have the same parameters, the only change from window [2] is that the maximum shift is limited by 10.

In the *Use of adjacent traces* group of parameters, we specify the *Number of traces* to 1, which means that, when constructing the adaptive subtraction filter, we consider not only the current trace, but two adjacent traces (one on the left and one on the right) as well. This improves the subtraction quality, but may start subtracting the primary waves, if the *Number of traces* is set too high. *Filter averaging base* is equal to 29, which allows us to obtain the adaptation filters averaged in spatial domain. Increasing this parameter value helps to preserve the primaries if they are present in the subtraction window. *Filter calculation step* is set to 10. Setting this parameter to 1 means that we compute the filter for every trace, which is accurate, but computationally complex. Here, the data allows us to set it to 10, decreasing the computation time significantly without influencing the quality of subtraction. The filter is computed for every 10th trace and is interpolated in between. Max number of iterations is 1 due to almost no visible improvement for more iterations. We set *Process by ensembles* to *Yes* and *Number of threads* to 4 for faster computation. Multi-threading in Wave Field Subtraction occurs by ensemble, so we pick the frame size which fits at least 4 ensembles (ideally, the number of ensembles in the frame needs to be a multiple of the *Number of threads*). In the end, there is a Trace Header Math which sets the MODEL header equal to 1 to make the model creation for the following curvelet subtraction easier. This flow takes quite a while to run.

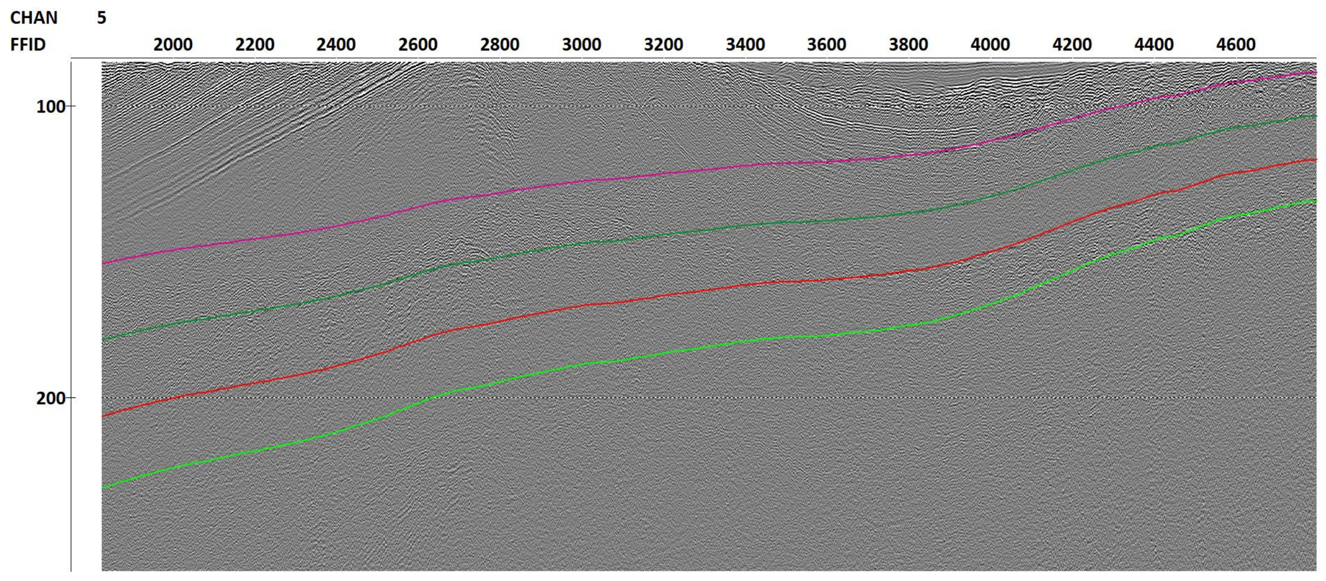
The quality control of the subtraction occurs in **036-ZOD-subtr-QC**. This flow displays channel 5 of three datasets – *026-stat*, *035-zod-mult-model*, and *036-subtr-zod* (the channel number can be changed in the Trace Inputs). It also contains a Trace Header Math and a set of Trace Header NMO/NMI modules – these are needed to display the subtraction windows. You can turn on the different Trace Input modules to display one of the three datasets. The subtraction of multiples can be clearly seen.



Common-channel gather before multiple subtraction



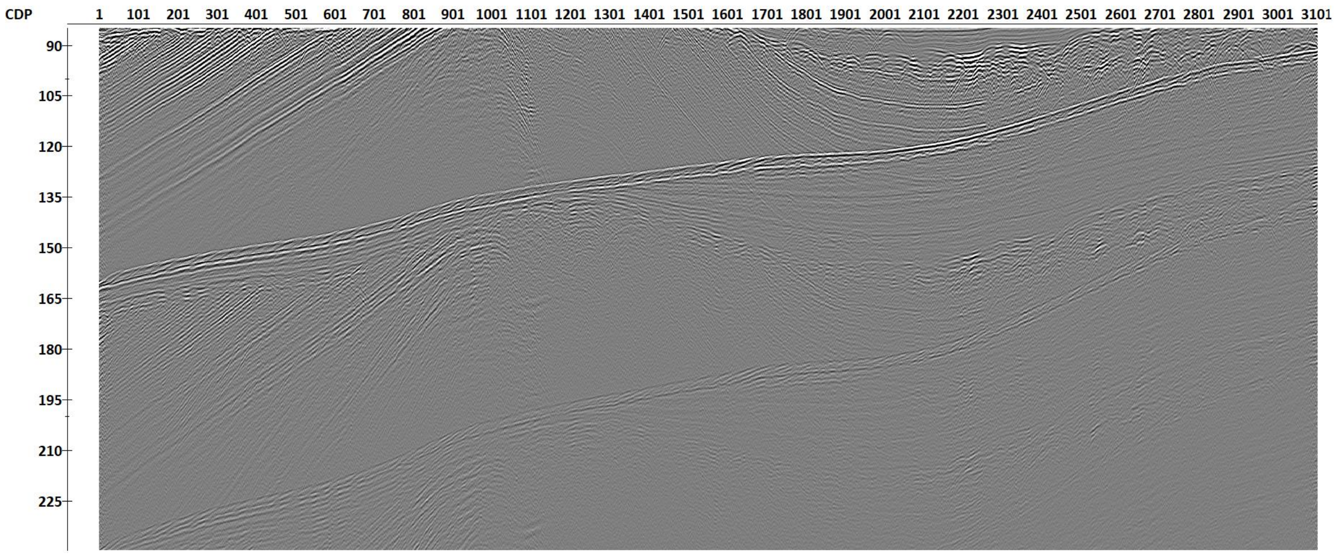
Common-channel gather of the model of multiples

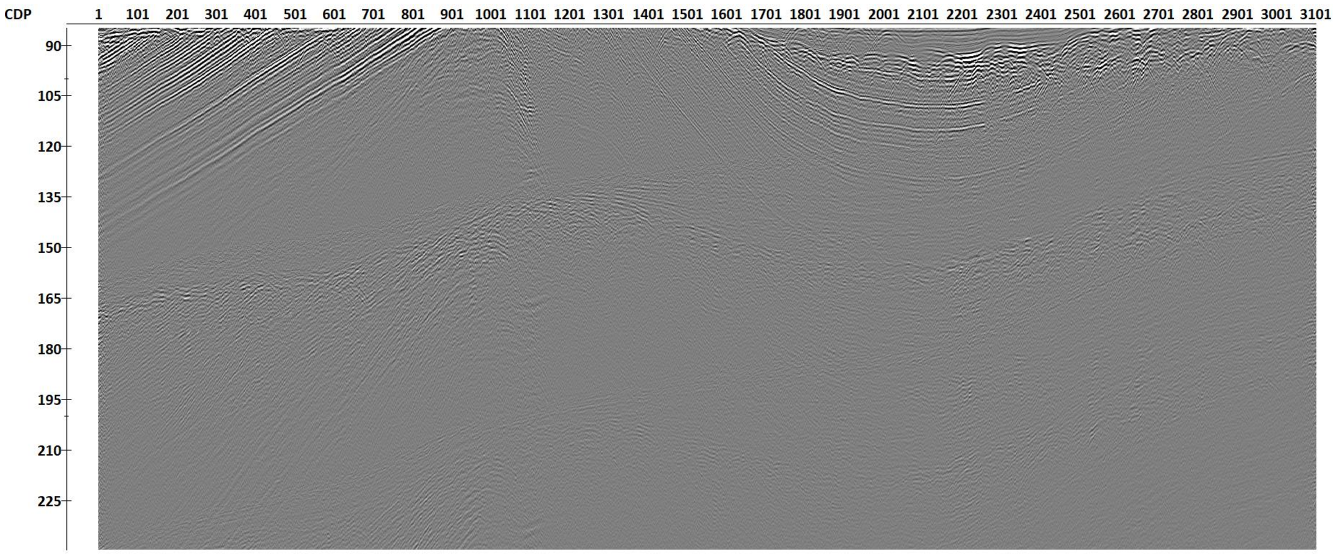


Common-channel gather after multiple subtraction

Flow **038-ZOD-stack** computes the stack after multiple subtraction. In **038-ZOD-stack-QC**, the stacks before and after multiple subtraction are compared. Some multiple energy remains after the subtraction, but the procedure is quite effective in general.

One can obtain the subtraction results in just one flow, using the subtraction options directly in the Zero-Offset DeMultiple module. We provide a flow for such subtraction in **037-ZOD-singleflow**. It provides a similar (slightly lower in quality) subtraction result. The differences occur mostly because in this single-flow case the subtraction takes place with NMO corrections applied.

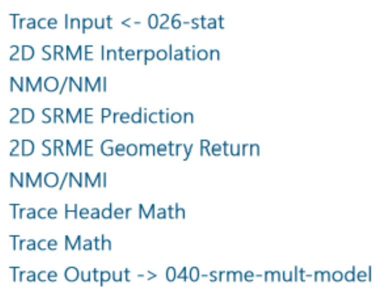




Stacks before (top) and after (bottom) Zero-Offset DeMultiple-based multiple subtraction

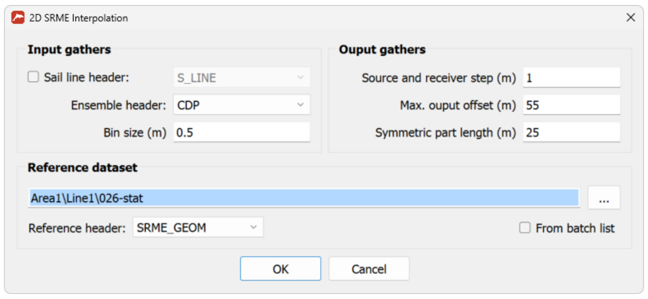
SRME

Now we repeat the multiple removal process using an alternative method, SRME. In RadExPro, three modules enable SRME: 2D SRME Interpolation, 2D SRME Prediction, 2D SRME Geometry Return. These modules are used in the **040-SRME-model** flow for the creation of the model of multiples. The flow looks as follows:


Trace Input <- 026-stat
2D SRME Interpolation
NMO/NMI
2D SRME Prediction
2D SRME Geometry Return
NMO/NMI
Trace Header Math
Trace Math
Trace Output -> 040-srme-mult-model

040-SRME-model flow

In this flow, the data are input in CDP:OFFSET sorting with Trace Input. Next, the module 2D SRME Interpolation conducts the interpolation of the data onto a regular grid required by the following 2D SRME Prediction. In 2D SRME Interpolation, we specify the *Bin size*, the *Source and receiver step* in the output gathers (here, we used the survey nominal receiver step of 1 m), the *Maximum output offset* in the interpolated data (needs to be larger than the maximum offset present in the survey) and the maximum negative offset (*Symmetric part length* parameter, a half of the maximum output offset is a suitable starting point). A reference dataset also needs to be set – this is the dataset where the reference header will be written. This reference header will later be used to extract the original survey geometry from the interpolated SRME gathers. In most cases, the input dataset to the SRME modeling flow is used as a reference dataset. Any integer-type header needs to be used in Reference header – here, we use SRME_GEOM, which was created specifically for this purpose.


2D SRME Interpolation

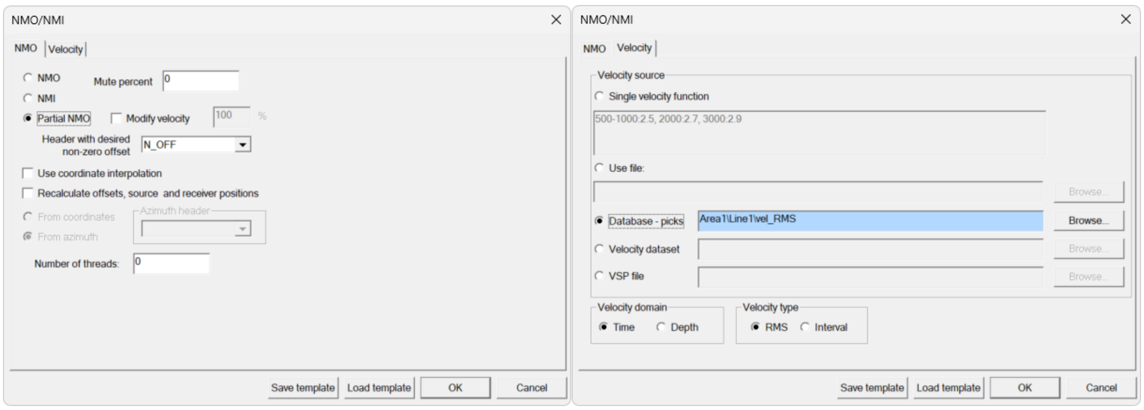
Input gathers
Sail line header: S_LINE
Ensemble header: CDP
Bin size (m) 0.5

Ouput gathers
Source and receiver step (m) 1
Max. ouput offset (m) 55
Symmetric part length (m) 25

Reference dataset
Area1\Line1\026-stat
Reference header: SRME_GEOM
From batch list

2D SRME Interpolation parameters

While 2D SRME Interpolation creates the interpolated traces, it does not transform them to the new offsets, the offset values where the traces need to be transformed to are written to the N_OFF header. To conduct the transformation, the NMO/NMI module is used in *Partial NMO* mode with the available velocity in *Velocity source* (here, we use the velocity analysis result *vel_RMS*, in some cases the water velocity is also suitable).



NMO/NMI

NMO | Velocity

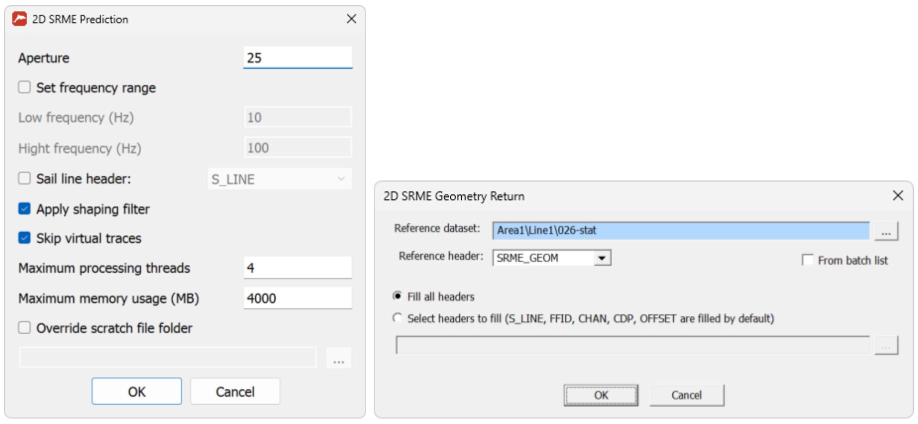
( ) NMO Mute percent: 0
( ) NMI
[✓ Partial NMO] [ ] Modify velocity: 100 %
Header with desired non-zero offset: N_OFF
[ ] Use coordinate interpolation
[ ] Recalculate offsets, source and receiver positions
( ) From coordinates [✓ From azimuth] Azimuth header:
Number of threads: 0

Velocity source:
( ) Single velocity function: 500-1000:2.5, 2000:2.7, 3000:2.9
( ) Use file:
[✓ Database - picks]: Area1\Line1\vel_RMS
( ) Velocity dataset
( ) VSP file
Velocity domain: [✓ Time] ( ) Depth
Velocity type: [✓ RMS] ( ) Interval

NMO/NMI parameters for partial NMO

After NMO/NMI, the multiple prediction takes place in 2D SRME Prediction. The main parameter here is the *Aperture* (equal to 25 in this case), which influences the accuracy of the prediction and potentially the presence of aliasing artefacts on modelled gathers. One can start with *Aperture* equal to the *Symmetric part length* in 2D SRME Interpolation and decrease it in case the results are not satisfactory (decreasing the aperture can make the aliasing weaker, but also may decrease the accuracy). We model the data in the full frequency band, so *Set frequency range* is turned off. Given the fact that the input data is one 2D line, *Sail line header* is also not needed. *Apply shaping filter* is useful, as it attempts to reconstruct the original signal waveform after prediction, and *Skip virtual traces* allows us to pass down the flow only those traces which are needed for the restoration of the original dataset's geometry.

Next, the 2D SRME Geometry Return module restores the original geometry and fills in the N_OFF field with the original offset. In the parameters of the module, one needs to fill in the same *Reference dataset* and *Reference header* which were used in 2D SRME Interpolation. Next, the NMO/NMI (with the same parameters as previously) applies partial NMO to shift the data to the original offset. The Trace Header Math provides an ID number to this model of multiples and writes it to the MODEL header. This number is later used in subtraction for proper input data sorting. Finally, Trace Math multiplies the prediction result by a small heuristic scalar number (this is only for convenience of plotting the prediction result along with the original data). The model is then saved to the dataset *040-SRME-mult-model*.


2D SRME Prediction

Aperture	25
☐ Set frequency range	
Low frequency (Hz)	10
Hight frequency (Hz)	100
☐ Sail line header:	S_LINE
☑ Apply shaping filter	
☑ Skip virtual traces	
Maximum processing threads	4
Maximum memory usage (MB)	4000
☐ Override scratch file folder	

2D SRME Geometry Return

Reference dataset: Area1\Line1\026-stat

Reference header: SRME_GEOM

☐ From batch list

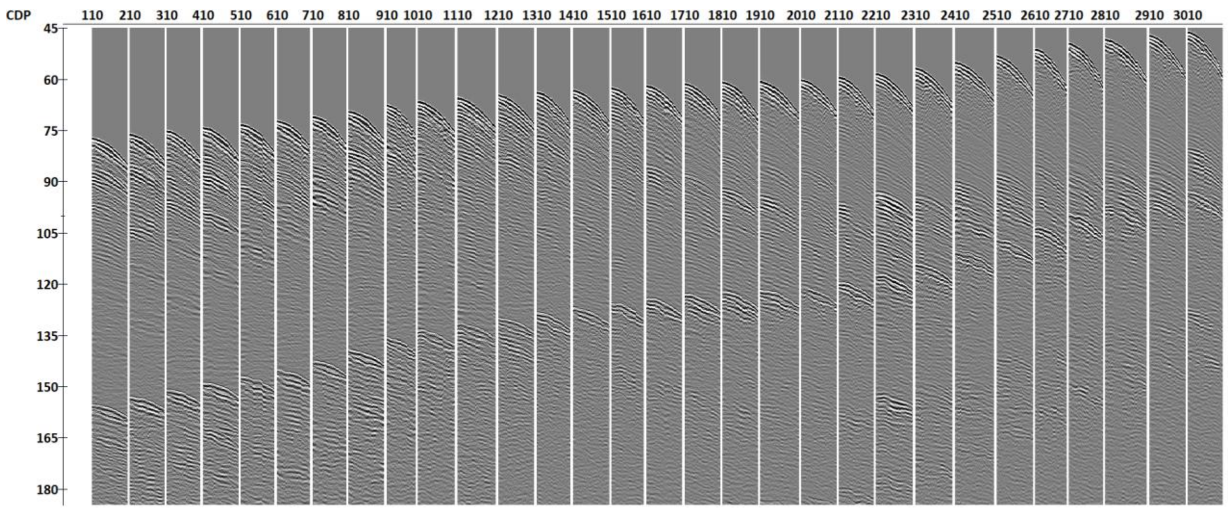
● Fill all headers

○ Select headers to fill (S_LINE, FFID, CHAN, CDP, OFFSET are filled by default)

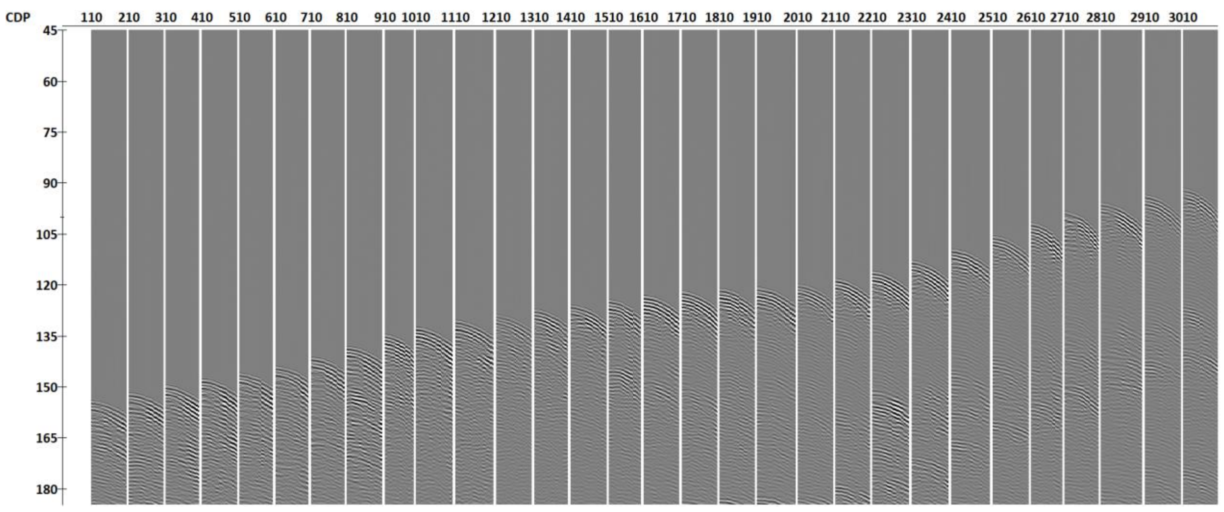
2D SRME Prediction and 2D SRME Geometry Return parameters

The flow **040-SRME-model-QC** plots the predicted CDP gather of the model versus the original data. We can observe the similarity between the predicted multiples and the multiples in the original data, although the model is not as accurate as the Zero-Offset DeMultiple model. The following flow, **041-SRME-subtr**, subtracts the SRME model from the original data. In terms of structure, the flow is the same as **036-ZOD-subtr**, the main difference is that the *035-zod-mult-model* model is replaced by *040-SRME-mult-model* in Trace Input. We also changed the Wave Field Subtraction parameters by making the filters slightly longer and by turning off the *Shift model* functionality in all the subtraction windows (this improves the SRME subtraction results in this case).

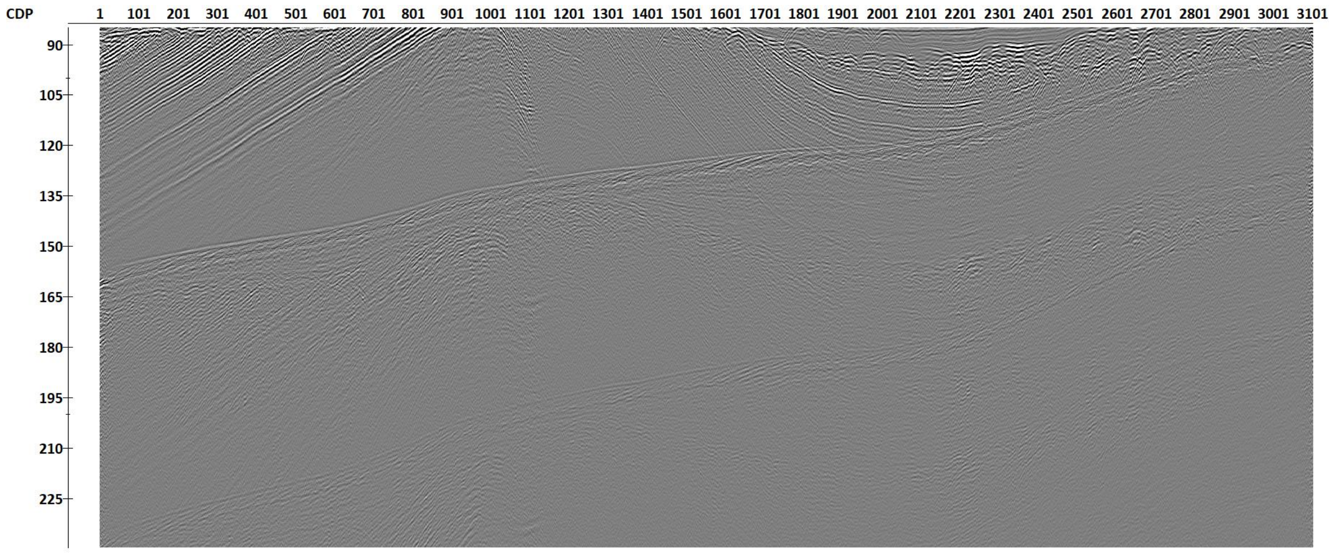
The quality of the subtraction can be studied in the flow **041-SRME-subtr-QC** by comparing the common-channel gathers of the data, the model, and the subtraction result. The flow **042-SRME-stack** computes the stack after the SRME subtraction, and the flow **042-SRME-stack-QC** displays it.



Input CDP gathers



Corresponding model of SRME-predicted multiples

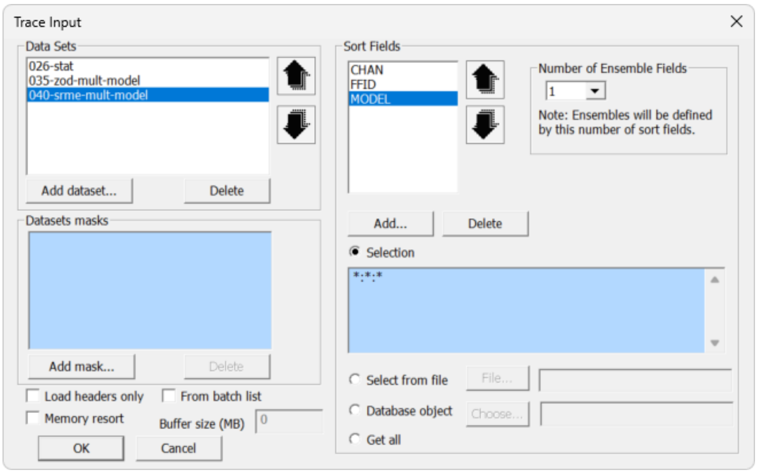


Stack after SRME

In general, we can observe that the subtraction is successful, but the quality of subtraction is lower than for Zero-Offset DeMultiple. So, we will use the Zero-Offset DeMultiple result for the next steps of the workflow.

Joint subtraction

In Wave Field Subtraction, one can input several models of multiples for the subtraction. In this case, we can supply both the Zero-Offset DeMultiple and SRME models and see if it improves the subtraction. We provide an example for such subtraction in the flow **045-joint-subtr**. The flow is almost the same as **036-ZOD-subtr**, with only a couple of changes. To subtract both models at once, we input three datasets in Trace Input (the data and two models) and change the *Number of models* in Wave Field Subtraction to 2. Note that in this case it is important that the model ID in the MODEL header is different for the data and two models (the data has MODEL=0, Zero-Offset DeMultiple has MODEL=1 and SRME has MODEL=2). One may also need to increase the frame size to accommodate the same number of ensembles. These are all the required changes. Joint subtraction takes a significant amount of time. We do not use the output of joint subtraction in the following steps of the workflow, but you can experiment and compare these results to simple Zero-Offset DeMultiple and SRME subtraction.



Trace Input

Data Sets: 026-stat, 035-zod-mult-model, 040-srme-mult-model (selected)

Sort Fields: CHAN, FFID, MODEL

Number of Ensemble Fields: 1

Note: Ensembles will be defined by this number of sort fields.

Selection: *:*:*

Buffer size (MB): 0

Trace Input parameters for joint subtraction

Curvelet-Domain Subtraction

Time-domain subtraction results obtained with Wave Field Subtraction can be further improved with the Curvelet-Domain Subtraction module. This module performs adaptive subtraction of multiples in the curvelet domain, which allows for more accurate matching and more effective subtraction. Similar to Wave Field Subtraction, this module takes interleaved data and model traces.

It is suggested to apply curvelet-domain matching after time-domain matching. This means that the time-domain-matched model of multiples needs to be obtained from Wave Field Subtraction. One can obtain such a model by specifying *Matched model* in the *Output type* of Wave Field Subtraction and saving this model to a separate dataset. Alternatively, Wave Field Subtraction and Curvelet-Domain Subtraction can be chained in a single flow. This is achieved by putting Curvelet-Domain Subtraction immediately after Wave Field Subtraction and specifying *Original data + matched model* in the *Output type* of Wave Field Subtraction. Finally, the time-domain matched model can be obtained by simply subtracting the Wave Field Subtraction result from the input dataset. The process is demonstrated in the flow **046-Curvelet-create-model**. In this flow, Trace Input inputs *026-stat* and *036-subtr-zod* datasets in CHAN-FFID-MODEL sort, which is followed by Trace Math in *Trace/Trace Mode* with *Subtract Traces* operation selected. Trace Header Math then sets the MODEL header to 2 for proper sorting in the subsequent flow. The model is then output to the *046-zod-mult-model* dataset.

The next flow, **047-Curvelet-subtr**, performs the subtraction. Trace Input loads the traces from the *026-stat* and *046-zod-mult-model* datasets in the required interleaved manner. This is followed by the Curvelet-Domain Subtraction module, the parameters of which are provided below.

There are two options for the windowing for Curvelet-Domain Subtraction: sliding windows or horizon-based windows. Here, we opt for the sliding windows. It has been observed that decreasing the window size may make the subtraction much more effective, although making the window too small may result in signal subtraction. For this application, we use 300-trace and 10-ms windows with some overlap. The subtraction starting horizon must be provided to prevent this module from generating artifacts above the first multiple. In this case, this starting horizon is taken from the MULT_W1 header.

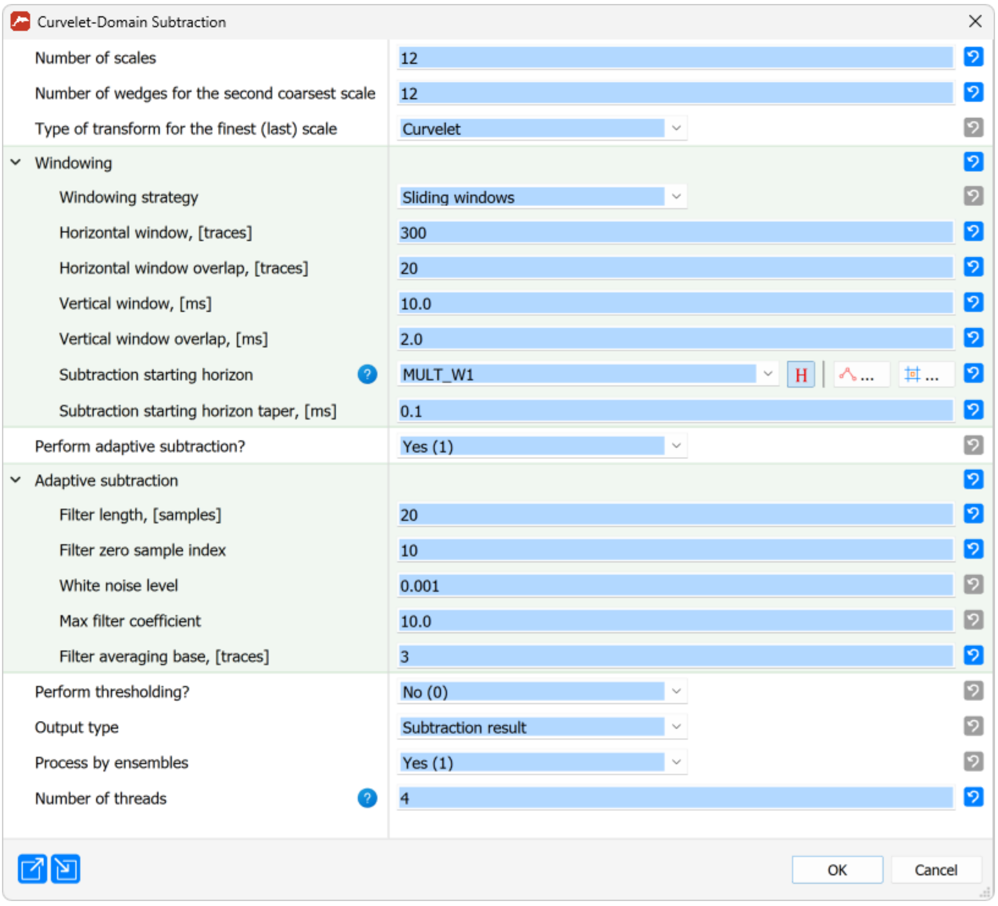
We set the number of scales and the number of wedges for the second coarsest scale to 12. Increasing these values provides a finer division of the 2D Fourier domain into wedges (the **RadExPro** manual contains a more detailed explanation), which should improve subtraction quality. However, if the model of multiples is inaccurate and the dips of multiples in the model and the data differ, the same multiple in the data and in the model can fall into different wedges, potentially decreasing subtraction quality. Furthermore, if the window is small and the numbers of scales and wedges are too large, the wedges will contain very small arrays, which can also influence the subtraction results. Ultimately, these parameters are chosen through trial and error.

Adaptive subtraction parameters are similar to those in Wave Field Subtraction, with the difference that the first two parameters are set in samples instead of milliseconds. A symmetric filter with a length similar to that of the wavelet is a good starting point when choosing the *Filter length* and *Filter zero sample index*. *White noise level*, *Max filter coefficient* and *Filter averaging base* can provide regularization if the adaptive subtraction is too aggressive and damages the signal. If the amplitudes in the model are accurate, decreasing the *Max filter coefficient* can be beneficial, as it will reduce the chance of the filter matching the multiple to a reflection that has a higher amplitude than the multiple. Increasing the *Filter averaging base* can also limit the filters' ability to subtract the signal.

For this dataset, we choose not to apply thresholding in curvelet domain. This is an additional nonlinear step that may improve subtraction quality in some instances. However, in many cases, it can lead to signal subtraction. It is suggested to first determine suitable adaptive subtraction parameters and then experiment with applying thresholding, using both hard and soft thresholds, starting with a *Subtraction strength* of 1. Increasing the *Subtraction strength* leads to more effective subtraction but also a higher risk of damaging the signal, and vice versa.

Subtraction result is chosen in the *Output type*, as we do not plan to conduct further subtraction steps (although chaining several instances of Curvelet-Domain Subtraction with different numbers of scales/wedges or window sizes can be useful).

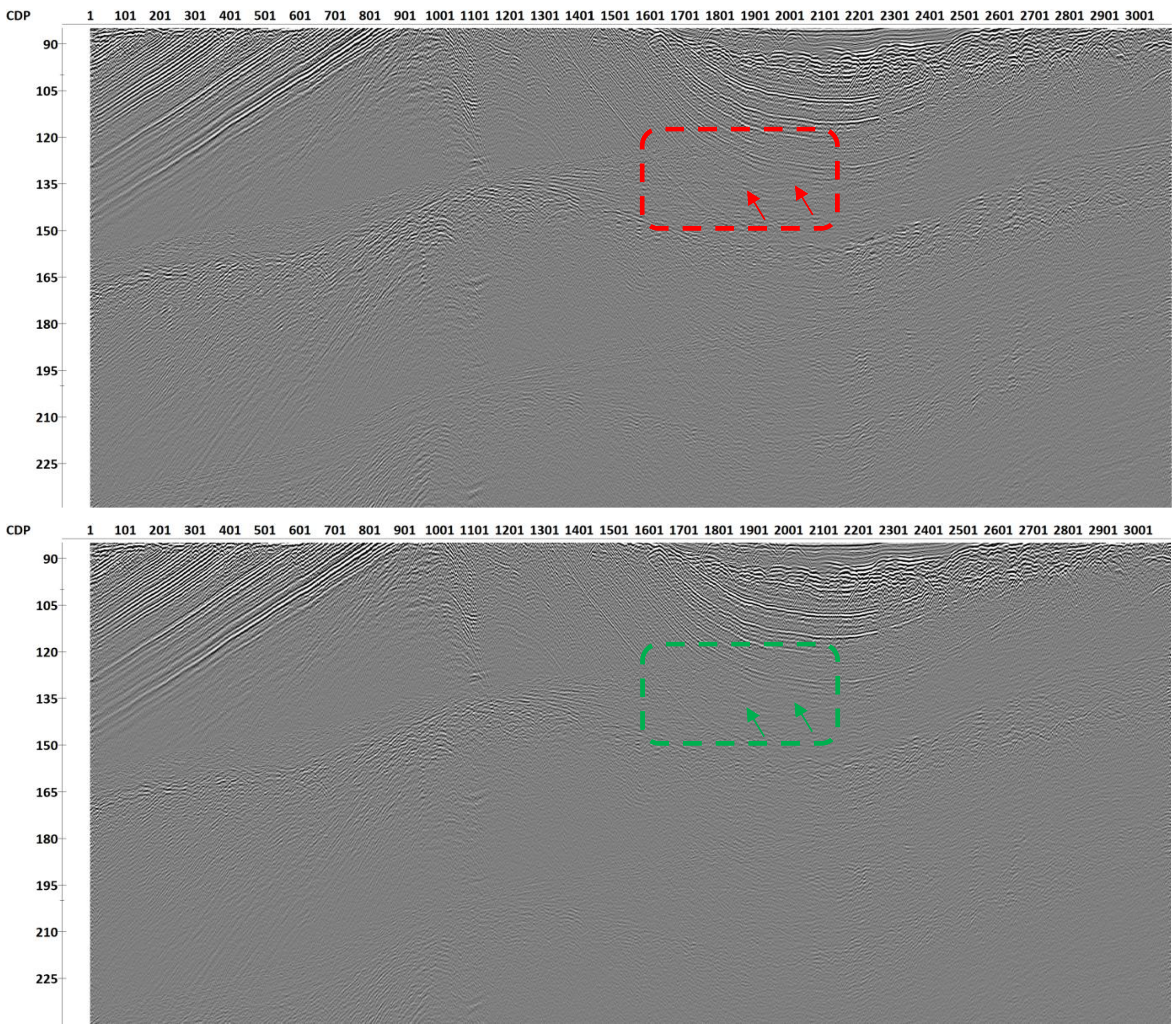
The data is processed by ensembles. Here, we are running parallel computations with 4 threads. In this configuration, different ensembles are processed in parallel, so one data frame needs to contain at least 4 common-channel gathers. Ideally, the number of such gathers in the frame needs to be a multiple of 4.


Curvelet-Domain Subtraction

Number of scales	12
Number of wedges for the second coarsest scale	12
Type of transform for the finest (last) scale	Curvelet
Windowing	
Windowing strategy	Sliding windows
Horizontal window, [traces]	300
Horizontal window overlap, [traces]	20
Vertical window, [ms]	10.0
Vertical window overlap, [ms]	2.0
Subtraction starting horizon	MULT_W1
Subtraction starting horizon taper, [ms]	0.1
Perform adaptive subtraction?	Yes (1)
Adaptive subtraction	
Filter length, [samples]	20
Filter zero sample index	10
White noise level	0.001
Max filter coefficient	10.0
Filter averaging base, [traces]	3
Perform thresholding?	No (0)
Output type	Subtraction result
Process by ensembles	Yes (1)
Number of threads	4

Curvelet-Domain Subtraction parameters

Flow **048-Curvelet-stack** computes the stack after Curvelet-Domain Subtraction, and **049-Curvelet-stack-QC** compares the stacks after time-domain subtraction and after curvelet-domain subtraction. The stacks are shown in the Figure below. One can observe how Curvelet-Domain Subtraction provides an additional improvement in subtraction quality, particularly in the highlighted zone where there is a difference in dip between the multiples and primaries (one almost horizontal multiple is highlighted with arrows).



A comparison between the time-domain and curvelet-domain subtraction results

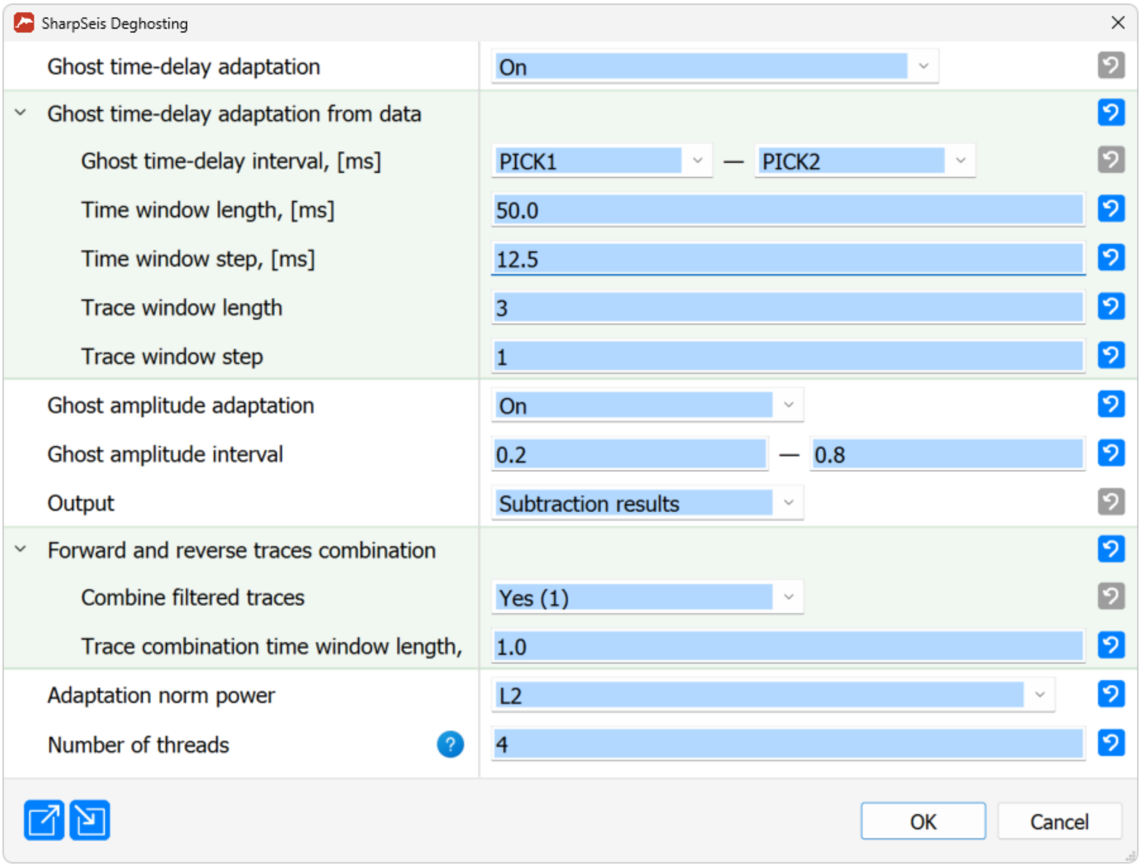
We do not use the output of Curvelet-Domain Subtraction in the following steps of the workflow, the users are encouraged to experiment with it by themselves.

Deghosting

Next, we take the Zero-Offset DeMultiple subtraction result and continue processing. Deghosting is the next step in the processing flow. For deghosting, we use a dedicated module titled SharpSeis Deghosting. This module constructs an adaptive filter for ghost wave removal. The deghosting operator is determined by the ghost wave delay and amplitude. The algorithm behind SharpSeis Deghosting selects the optimal values of these two parameters according to the user-defined constraints.

The deghosting occurs in the flow **050-Deghosting**. The parameter estimation occurs in finite windows along time and trace dimensions, so we input the *036-subtr-zod* dataset in Trace Input using CHAN:FFID sorting, as it allows us to keep the ghost time delay relatively constant within one window.

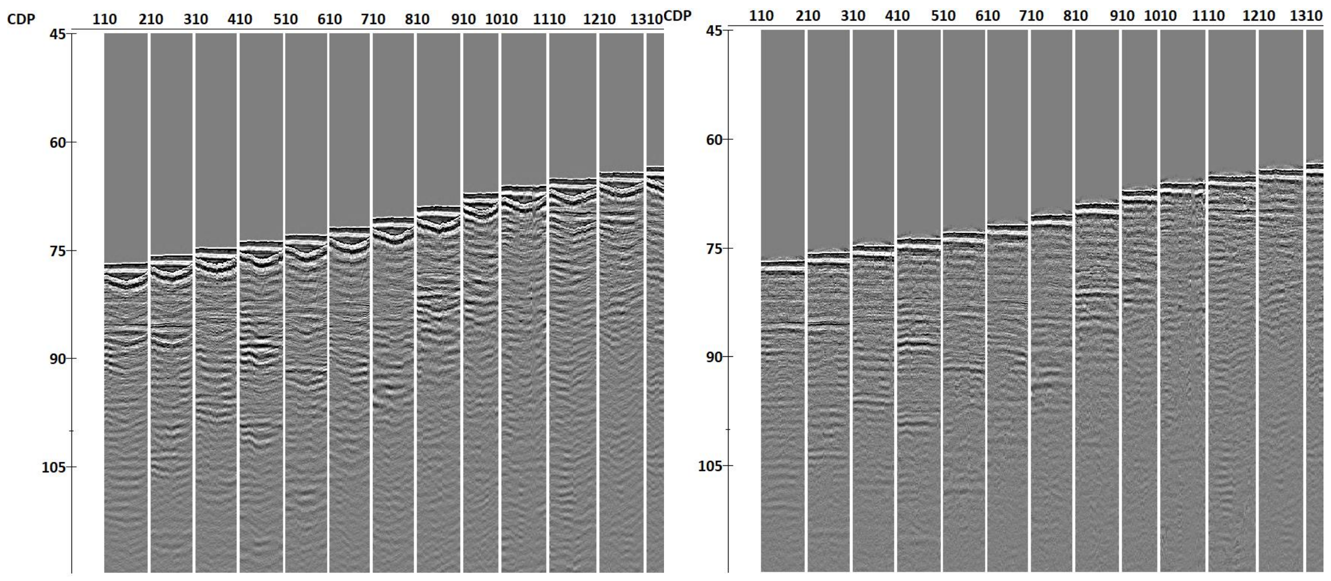
Next, Trace Header Math specifies the limits for ghost time delay in PICK1 (1 ms) and PICK2 (10 ms) headers. These limits depend on the range of time delays in the data – here, we deliberately set a wide range. Then, the deghosting occurs in SharpSeis Deghosting. This module adapts the time delay in the range specified by the headers and the ghost amplitude in the range from 0.2 to 0.8 (as specified in *Ghost amplitude interval*) relative to the amplitude of the signal. The deghosting window is 50 ms and 3 traces-long. Determining the window size is always a tradeoff – one needs to set the window small enough so that the ghost wave properties do not change inside it significantly, and big enough so that there is enough statistics for robust estimation of the parameters. We also combine the causal and non-causal deghosting results nonlinearly in 1 ms-long windows to attenuate the artefacts caused by the long impulse response of the deghosting operator. L2 norm is used for adaptation here, although other options may be preferable on other datasets. Then, the Trace Header Math prepares a header for top muting and Trace Editing applies top muting, as deghosting may cause some non-causal artefacts. The deghosted data are written to *050-deghost*.



Ghost time-delay adaptation	On
Ghost time-delay adaptation from data	
Ghost time-delay interval, [ms]	PICK1 — PICK2
Time window length, [ms]	50.0
Time window step, [ms]	12.5
Trace window length	3
Trace window step	1
Ghost amplitude adaptation	On
Ghost amplitude interval	0.2 — 0.8
Output	Subtraction results
Forward and reverse traces combination	
Combine filtered traces	Yes (1)
Trace combination time window length,	1.0
Adaptation norm power	L2
Number of threads	4

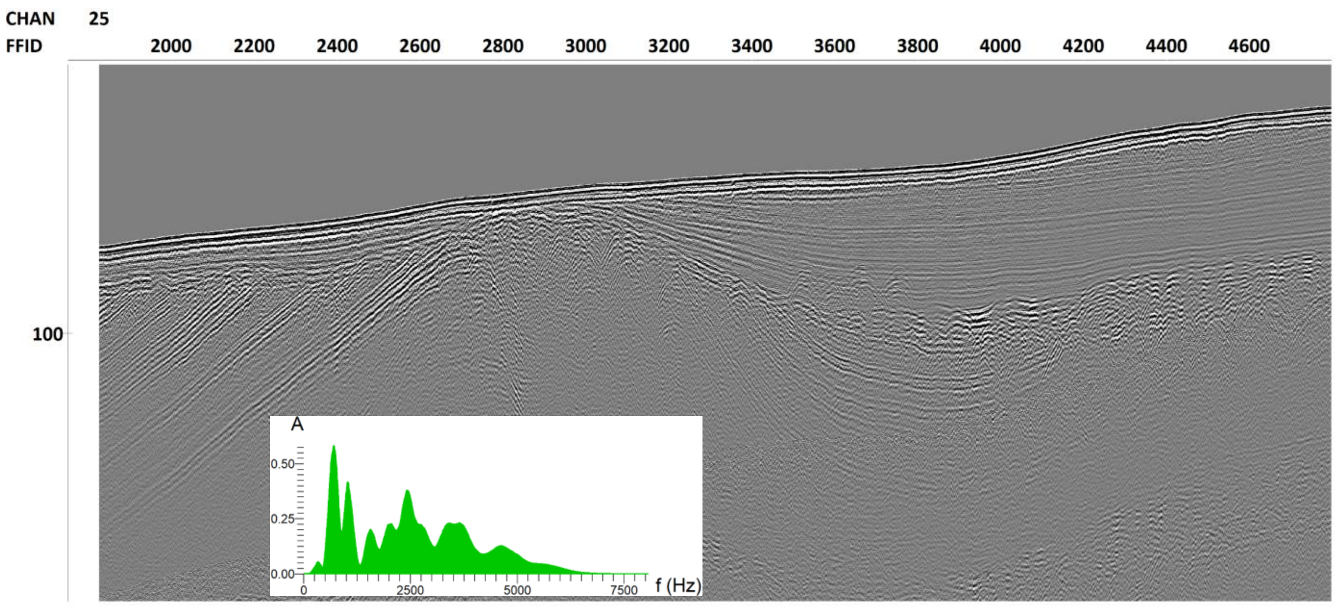
SharpSeis Deghosting parameters

Flow **051-Deghosting-QC** compares the common-channel gathers and the NMO-corrected CDP gathers before and after deghosting. The attenuation of the ghost wave is clearly visible on both.

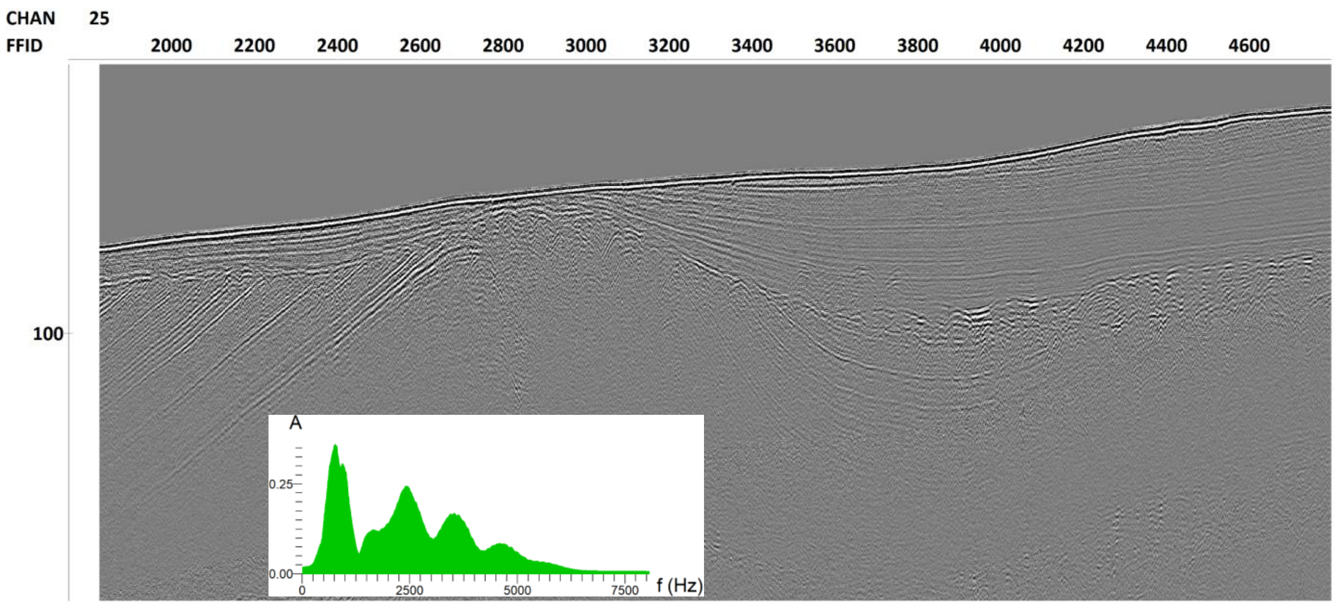


NMO-corrected CDP gathers before (left) and after (right) deghosting

In the figures below, we show the comparison of common-channel gathers and the estimated wavelet amplitude spectra for a channel with a rather deep tow. One can observe the successful removal of the ghost, which manifests itself in the smoothing of the amplitude spectrum (particularly visible for low frequencies in this figure). Note that the source ghost is still present in the data, and the original sparker waveform shape has some secondary pulses, so the spectrum is not perfectly smooth. The following deconvolution procedure handles these effects by compressing the wavelet into a very short pulse with a flat spectrum.

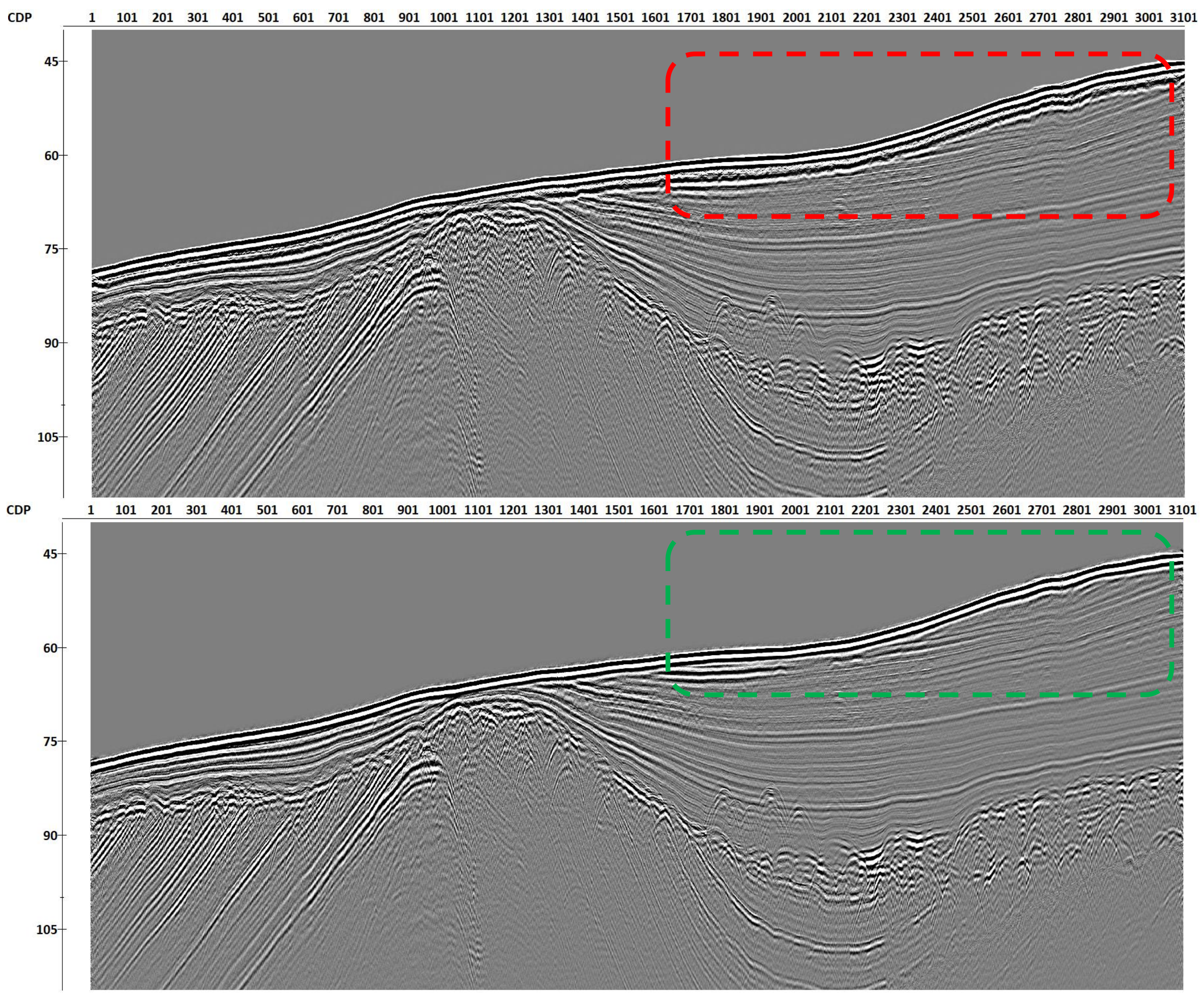


Common-channel gather and wavelet amplitude spectrum before deghosting



Common-channel gather and wavelet amplitude spectrum after deghosting

The flow **052-Deghosting-stack** computes the stack after deghosting. Although the difference is not that pronounced on the stack (due to the variation in streamer depth, the ghost wave is attenuated on stack even without deghosting), the difference is still visible in the figure below (these plots occur in the flow **053-Deghosting-stack-QC**).

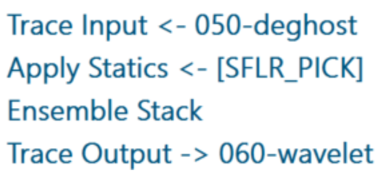


Seismic stacks before (top) and after (bottom) deghosting

Deconvolution

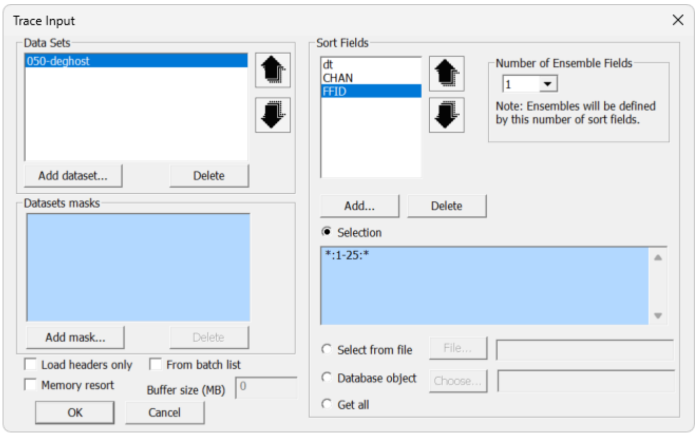
The next processing step is deconvolution. The proposed deconvolution workflow is deterministic – we estimate the seismic wavelet by stacking the seafloor reflection on a large portion of the data and dividing the spectrum of each data trace by the spectrum of this wavelet.

Wavelet estimation happens in the flow **060-Wavelet-estimation**. The flow looks as follows.


Trace Input <- 050-deghost
Apply Statics <- [SFLR_PICK]
Ensemble Stack
Trace Output -> 060-wavelet

060-Wavelet-estimation flow

Trace Input inputs the data in dt:CHAN:FFID sort with *Number of Ensemble Fields* equal to 1. This sorting is needed so that the whole dataset is input as a single ensemble (and will eventually be summed into one trace). We select a subset of channels (1-25) to maximize the accuracy of the wavelet estimate.


Trace Input

Data Sets: 050-deghost

Add dataset... Delete

Datasets masks

Add mask... Delete

☐ Load headers only ☐ From batch list

☐ Memory resort Buffer size (MB) 0

OK Cancel

Sort Fields: dt, CHAN, FFID

Number of Ensemble Fields: 1

Note: Ensembles will be defined by this number of sort fields.

Add... Delete

● Selection: *:1-25:*

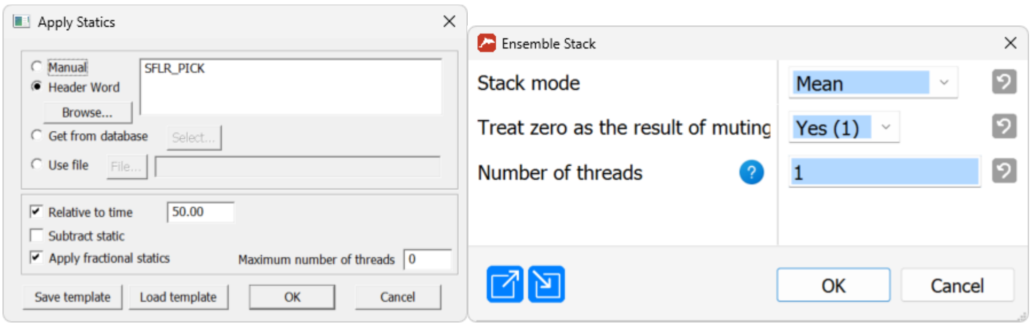
○ Select from file File...

○ Database object Choose...

○ Get all

Trace Input parameters for wavelet estimation

Next, we use the picked seafloor in SFLR_PICK to flatten the seafloor relative to 50 ms time by Apply Statics (50 ms is mostly there for plotting purposes, one could use the same flow with *Relative to time* turned off, and the wavelet would be located at zero time). After the seafloor is flattened, we stack the whole dataset by Ensemble Stack. This results in one trace, which is saved to *060-wavelet*.


Apply Statics

○ Manual

● Header Word: SFLR_PICK

Browse...

○ Get from database Select...

○ Use file File...

☑ Relative to time 50.00

☐ Subtract static

☑ Apply fractional statics Maximum number of threads 0

Save template Load template OK Cancel

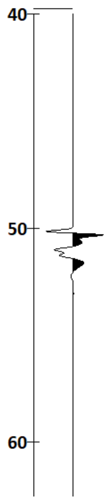
Ensemble Stack

Stack mode	Mean
Treat zero as the result of muting	Yes (1)
Number of threads	1

OK Cancel

Apply Statics and Ensemble Stack parameters for wavelet estimation

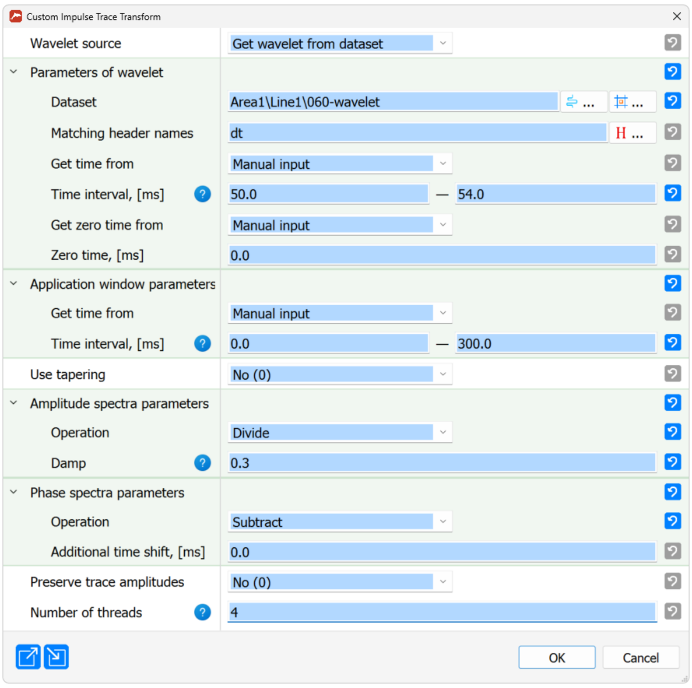
The flow **061-Wavelet-estimation-QC** plots the wavelet.



The estimated wavelet

The flow **062-Deconvolution** uses the estimated wavelet for the deconvolution of the data. There are two processing modules in the flow, Custom Impulse Trace Transform and Trace Editing. Custom Impulse Trace Transform conducts the deconvolution. It works in the *Get wavelet from dataset* mode.

We specify the *060-wavelet* dataset in the *Dataset* field and set the time interval from 50 to 54 ms. The start of the window is chosen due to the fact that we previously placed our wavelet at 50 ms by Apply Statics; the 54 ms end of the interval is set a result of the visual analysis of the wavelet shape. The application window is the whole trace, 0-300 ms. To enable deconvolution, we *Divide* the amplitude spectra of each trace by the amplitude spectrum of the wavelet (with *Damp* equal to 0.3) and *Subtract* the phase spectrum of the wavelet from the phase spectrum of each trace. Trace Editing then reapplies the top muting. The data is saved to *062-decon*.

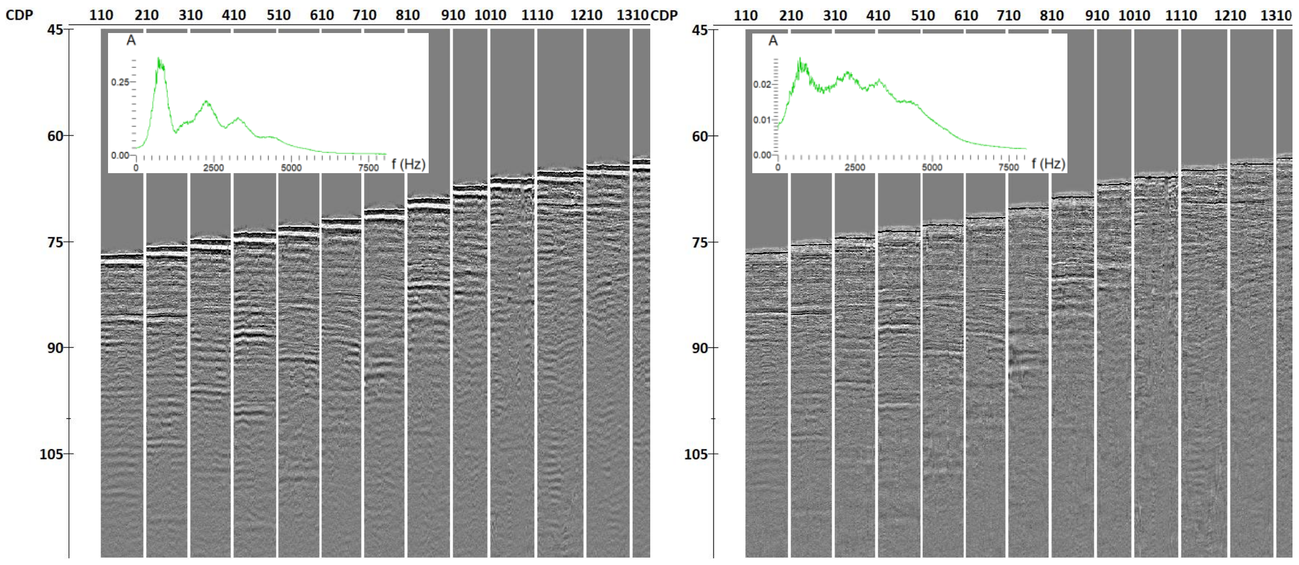

Custom Impulse Trace Transform

Wavelet source	Get wavelet from dataset
Parameters of wavelet	
Dataset	Area1\Line1\060-wavelet
Matching header names	dt
Get time from	Manual input
Time interval, [ms]	50.0 — 54.0
Get zero time from	Manual input
Zero time, [ms]	0.0
Application window parameters	
Get time from	Manual input
Time interval, [ms]	0.0 — 300.0
Use tapering	No (0)
Amplitude spectra parameters	
Operation	Divide
Damp	0.3
Phase spectra parameters	
Operation	Subtract
Additional time shift, [ms]	0.0
Preserve trace amplitudes	No (0)
Number of threads	4

OK Cancel

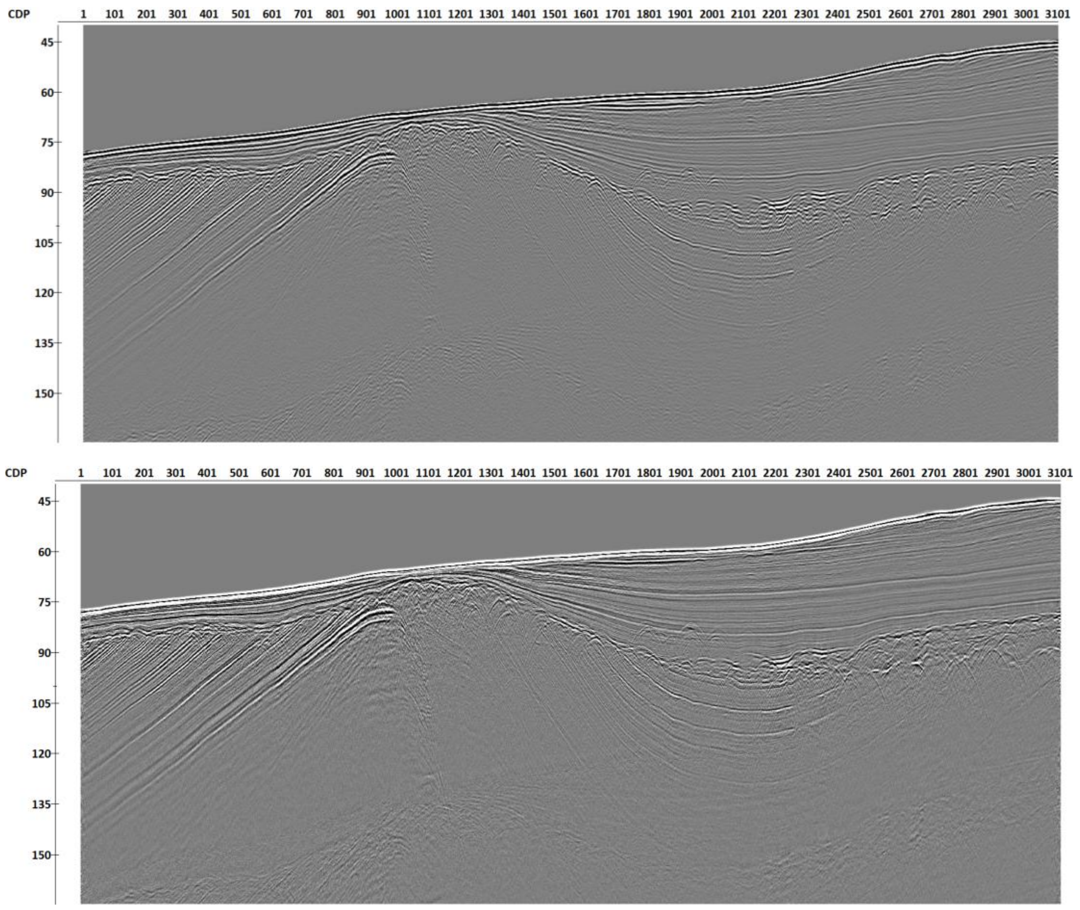
Custom Impulse Trace Transform parameters

The flow **063-Deconvolution-QC** plots the NMO-corrected CDP gathers. On them, one can observe that the seafloor reflection gets compressed into a very short pulse after deconvolution, which is accompanied by the flattening of the spectrum.



NMO-corrected CDP gathers and their spectra before (left) and after (right) deconvolution

Next, the stack after deconvolution is computed in **064-Deconvolution-stack**. It is then compared to the stack before deconvolution in the flow **065-Deconvolution-stack-QC**. The increase in resolution after deconvolution is clearly visible.

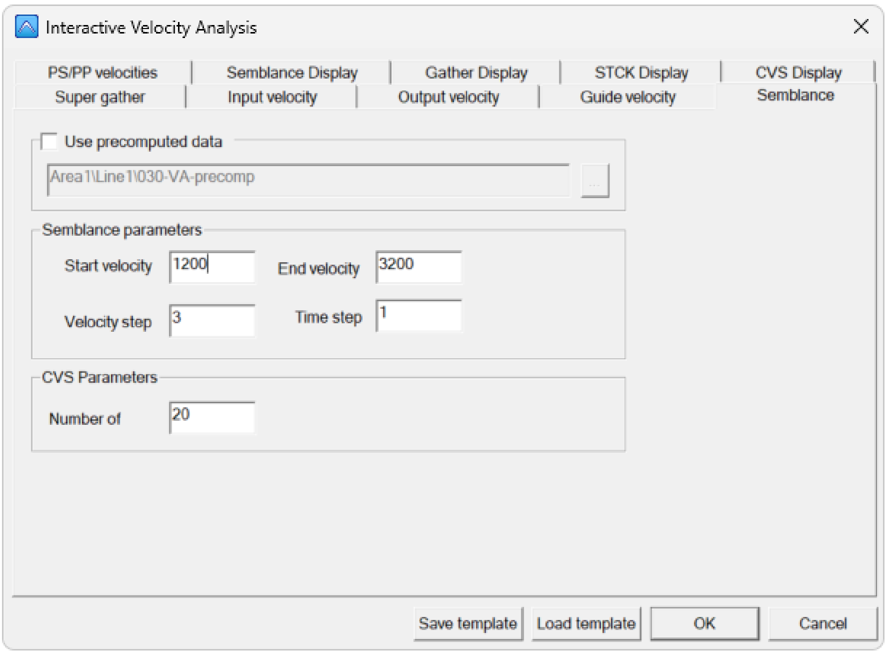


Seismic stacks before (top) and after (bottom) deconvolution

Here, deconvolution with a single wavelet is successful. Still, in many cases a channel-by-channel deconvolution is preferable. In order to extract a wavelet for each channel, one needs to make sure the seafloor pick is accurate for all channels, change the Trace Input sorting in **060-Wavelet-estimation** from dt:CHAN:FFID to CHAN:FFID and provide the whole dataset with *:* *Selection*. After this, the flow will export a wavelet for each channel to *060-wavelet* dataset (so, for this project, there will be 48 wavelets). Then, in **062-Deconvolution**, *Matching header names* field in Custom Impulse Trace Transform needs to be set to CHAN instead of dt. Optionally, one may also decide to set *Preserve trace amplitudes* to *Yes* – in this case, the energies of the wavelets will be normalized, which may provide (depending on the data) a more consistent amplitude distribution versus offset after channel-by-channel deconvolution.

Velocity analysis

After demultiple and deconvolution, it may be worth revising the velocity analysis. We provide the flow titled **070-Velocity-Analysis**, which is similar to the previously seen velocity analysis flows, but with the 062-decon dataset provided in Super Gather. One more difference is that in this case we do not run the Velocity Analysis Precompute module, we compute semblances on the fly. This is why the flow contains Super Gather -> Amplitude Correction -> Interactive Velocity Analysis. In Interactive Velocity Analysis, on the Semblance page, *Use precomputed data* is turned off and the parameters for semblance computation are specified. When launching the flow, we get a velocity analysis window similar to the one seen in the flow **031-Velocity-Analysis**. Note that the module is set so that it updates the previously used velocity *vel_RMS* (in the current project, we provide only one final velocity model, so no picking needs to happen).



Use precomputed data: ☐
Area1\Line1\030-VA-precomp

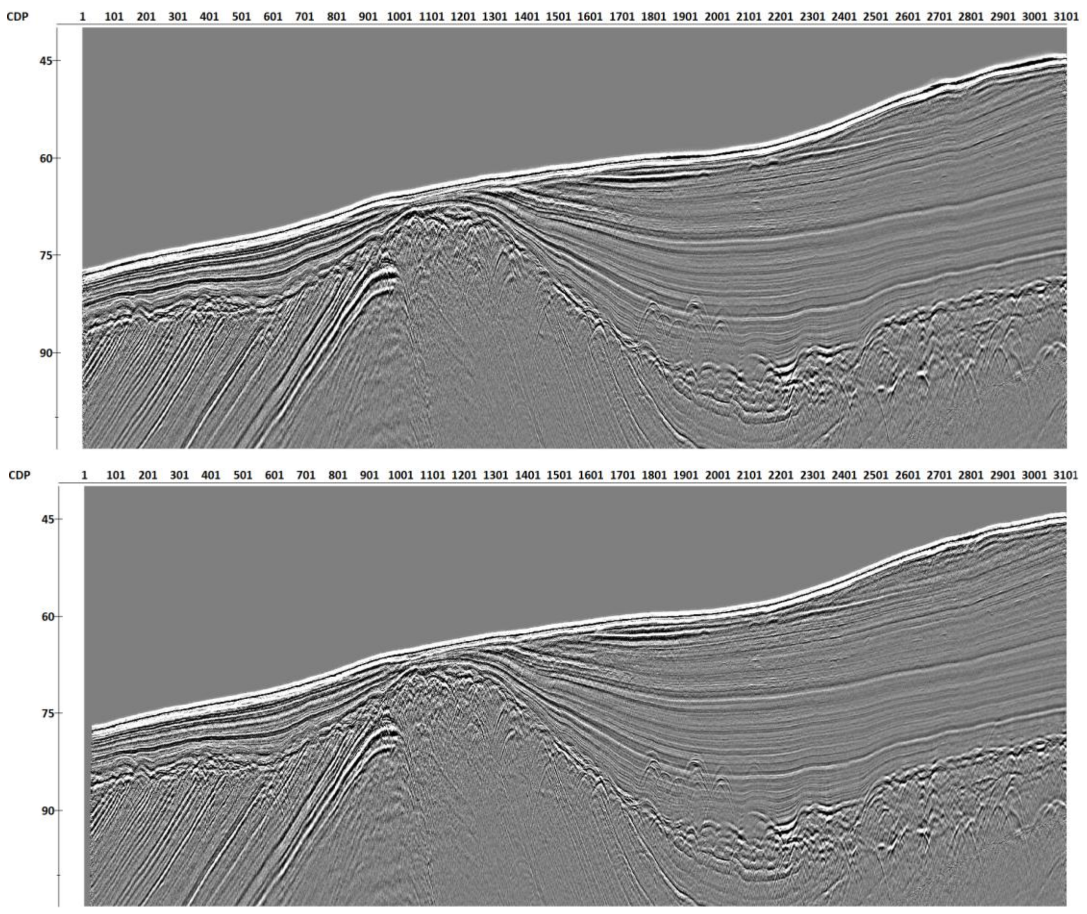
Semblance parameters
Start velocity: 1200
End velocity: 3200
Velocity step: 3
Time step: 1

CVS Parameters
Number of: 20

Semblance window in Interactive Velocity Analysis when precompute is not used

Final stacking

Now, we need to obtain the final stack from the 062-decon dataset, which will use the up-to-date velocities and, if needed, apply some additional processing, such as top muting. The flow **075-Final-stack** does exactly that and outputs the final stack to *075-decon-final-stack*. The only difference from the previous stacking flows is that we apply *Mute percent* equal to 5 in NMO/NMI, which will mute the far offsets and improve the stack in the near surface. The improvement can be observed in the flow **076-Final-stack-QC**, where the final stack is plotted versus the one obtained in **064-Deconvolution-stack**.



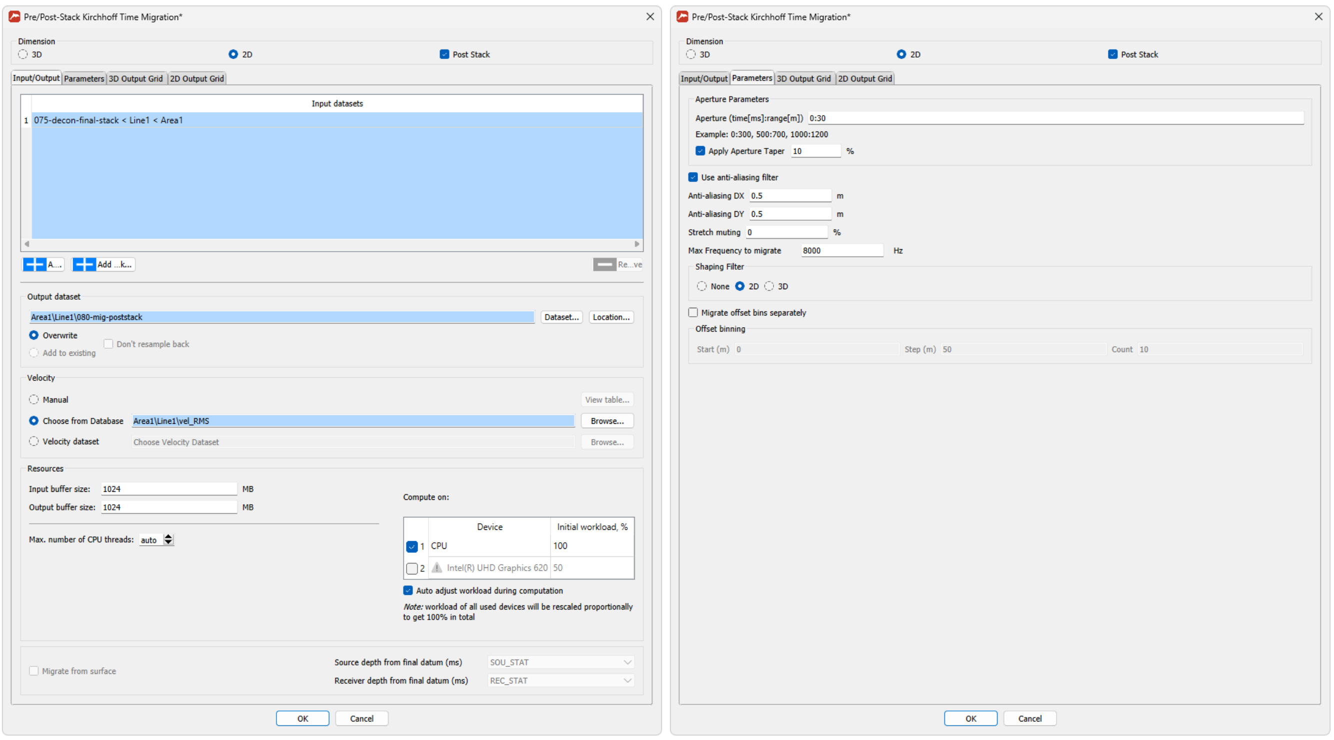
Stack without (above) and with (below) NMO stretch muting

Migration

Migration is the next step of the processing flow. We demonstrate how to run both post- and pre-stack migration using the final processed data. Both migrations are computed with the Pre/Post-Stack Kirchhoff Migration* module. This is a standalone module, it does not need other modules in the flow.

Post-stack Kirchhoff Migration

Post-stack migration occurs in the flow **080-Migration-poststack**. The parameters of the Pre/Post-Stack Kirchhoff Migration* are shown below. *Dimension* is set to *2D*, the *Post Stack* option is active. *075-decon-final-stack* is the only input dataset (note that this migration can migrate several datasets at once). The output dataset is *080-mig-poststack*. The final RMS velocities in *vel_RMS* are selected in *Velocity*. In this case, default buffer sizes are used, and migration is launched on CPU. However, if a dedicated GPU is available, we encourage you to try computing the migration using it. It can make the computations significantly faster (particularly for 3D). On the *Parameters* tab, we specify the *Aperture* (it can change in time and space, however here we use a constant 30-meter aperture) with 10% taper, use an anti-aliasing filter with 0.5 m *Anti-aliasing DX*. Note that this spacing influences only antialiasing, one can tweak it – it is fine if the chosen value does not coincide with the true trace spacing in the data. The stretch muting is set to 0 (which means that there is no stretch muting) and the Maximum frequency is set to 8000 Hz. This is close to the Nyquist frequency in this case. To save computation time, one may decide to limit the frequency range of the migration (although, in this case, it is advised to run preliminary filtering to make sure that the data does not contain frequencies higher than the one identified in this parameter window). Finally, *2D* shaping filter (rho-filter) is used.



Pre/Post-Stack Kirchhoff Time Migration*

Dimension: 3D [✓ 2D] — Post Stack ✓

Input/Output | Parameters | 3D Output Grid | 2D Output Grid

Input datasets
1 075-decon-final-stack < Line1 < Area1

Output dataset: Area1\Line1\080-mig-poststack
[✓ Overwrite] Add to existing — Don't resample back ☐

Velocity: Manual — [✓ Choose from Database] Area1\Line1\vel_RMS — Velocity dataset: Choose Velocity Dataset

Resources
Input buffer size: 1024 MB
Output buffer size: 1024 MB
Max. number of CPU threads: auto

Compute on:

	Device	Initial workload, %
✓ 1	CPU	100
☐ 2	Intel(R) UHD Graphics 620	50

✓ Auto adjust workload during computation
Note: workload of all used devices will be rescaled proportionally to get 100% in total

☐ Migrate from surface
Source depth from final datum (ms): SOU_STAT
Receiver depth from final datum (ms): REC_STAT

Pre/Post-Stack Kirchhoff Time Migration* — Parameters

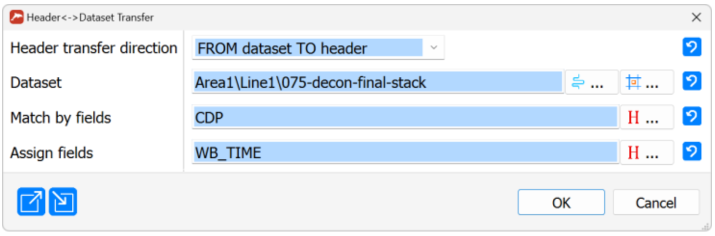
Aperture Parameters
Aperture (time[ms]:range[m]) 0:30
Example: 0:300, 500:700, 1000:1200
✓ Apply Aperture Taper 10 %

✓ Use anti-aliasing filter
Anti-aliasing DX 0.5 m
Anti-aliasing DY 0.5 m
Stretch muting 0 %
Max Frequency to migrate 8000 Hz
Shaping Filter: None [✓ 2D] 3D

☐ Migrate offset bins separately
Offset binning: Start (m) 0 — Step (m) 50 — Count 10

Pre/Post-Stack Kirchhoff Time Migration parameters for post-stack migration

Before looking at the migration result, we perform some postprocessing in the flow **081-Migration-poststack-postproc**. This flow takes the *080-mig-poststack* dataset and conducts top muting to remove the energy which the migration leaves above the seafloor. To run the muting, we need the seafloor estimate. It was created by static corrections in the WB_TIME header, however the header was rewritten during migration. We can take this header from the *075-decon-final-stack* dataset with the Header <-> Dataset Transfer module. This module can take a header from the dataset which is not present in the flow or write a header to such dataset. Here, we apply the direction *FROM dataset TO header*, taking the WB_TIME header from *075-decon-final-stack*, matching the two datasets by CDP. This takes WB_TIME from *075-decon-final-stack* and gives it to the traces in the current flow. Next, we use Trace Header Math to set TOP_MUTE=WB_TIME-1.5 and use Trace Editing for top muting (as it was done several times previously), outputting the result to *081-mig-poststack-postproc*.

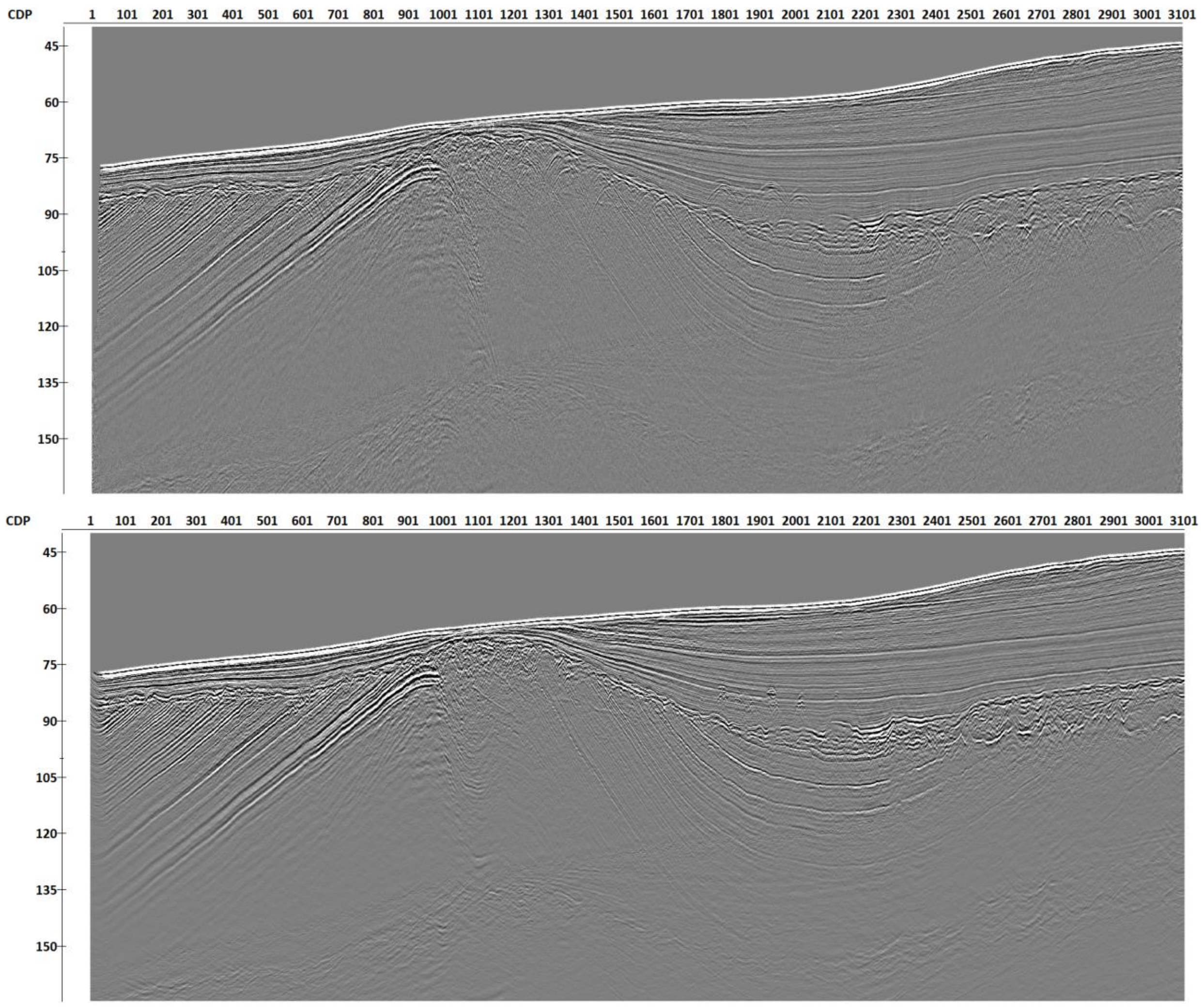


Header<->Dataset Transfer

Header transfer direction	FROM dataset TO header
Dataset	Area1\Line1\075-decon-final-stack
Match by fields	CDP
Assign fields	WB_TIME

Header <-> Dataset Transfer parameters

Finally, the flow **082-Migration-poststack-QC** compares the migration result to the final stack. We can see that the migration successfully reconstructs the complex geometry of the subsurface.

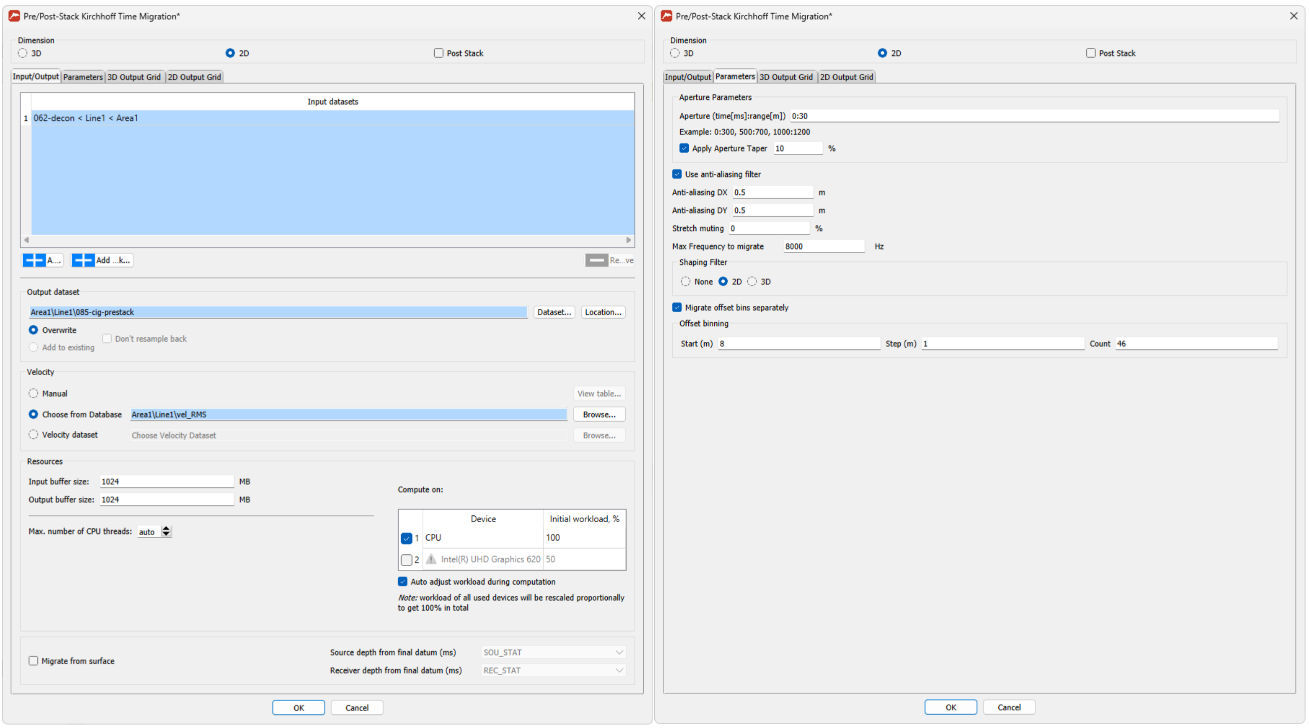


Final stack (above) compared to the migration result (below)

Pre-stack Kirchhoff Migration

To run the pre-stack migration (flow **085-Migration-prestack)**, a few parameters need to be changed in Pre/Post-Stack Kirchhoff Migration*. The *input dataset* is replaced by the prestack data *062-decon*, the output dataset is *085-cig-prestack*, the *Post Stack* option is turned off. We also choose to *Migrate offset bins separately*, with *Start (m)* set to 8 m, 1 m *Step (m)* and 46 bin *Count*. The offset binning depends on the survey parameters – here, we just select the smallest offset as the center of the first offset bin, keep the step between the bins equal to the receiver step in the survey, and the count is set so that the last bin is populated with quite many traces.

The migration outputs common image gathers, which are plotted in **085-Migration-prestack-QC**. They are relatively flat, which suggests that the velocities are quite accurate.



Pre/Post-Stack Kirchhoff Time Migration*

Dimension: ( ) 3D [✓ 2D] [ ] Post Stack

Input/Output | Parameters | 3D Output Grid | 2D Output Grid

Input datasets
1 062-decon < Line1 < Area1

Output dataset: Area1\Line1\085-cig-prestack
[✓ Overwrite] ( ) Add to existing [ ] Don't resample back

Velocity
( ) Manual
[✓ Choose from Database] Area1\Line1\vel_RMS
( ) Velocity dataset: Choose Velocity Dataset

Resources
Input buffer size: 1024 MB
Output buffer size: 1024 MB
Max. number of CPU threads: auto

Compute on:

	Device	Initial workload, %
✓ 1	CPU	100
2	Intel(R) UHD Graphics 620	50

[✓] Auto adjust workload during computation
Note: workload of all used devices will be rescaled proportionally to get 100% in total

[ ] Migrate from surface
Source depth from final datum (ms): SOU_STAT
Receiver depth from final datum (ms): REC_STAT

Pre/Post-Stack Kirchhoff Time Migration* — Parameters

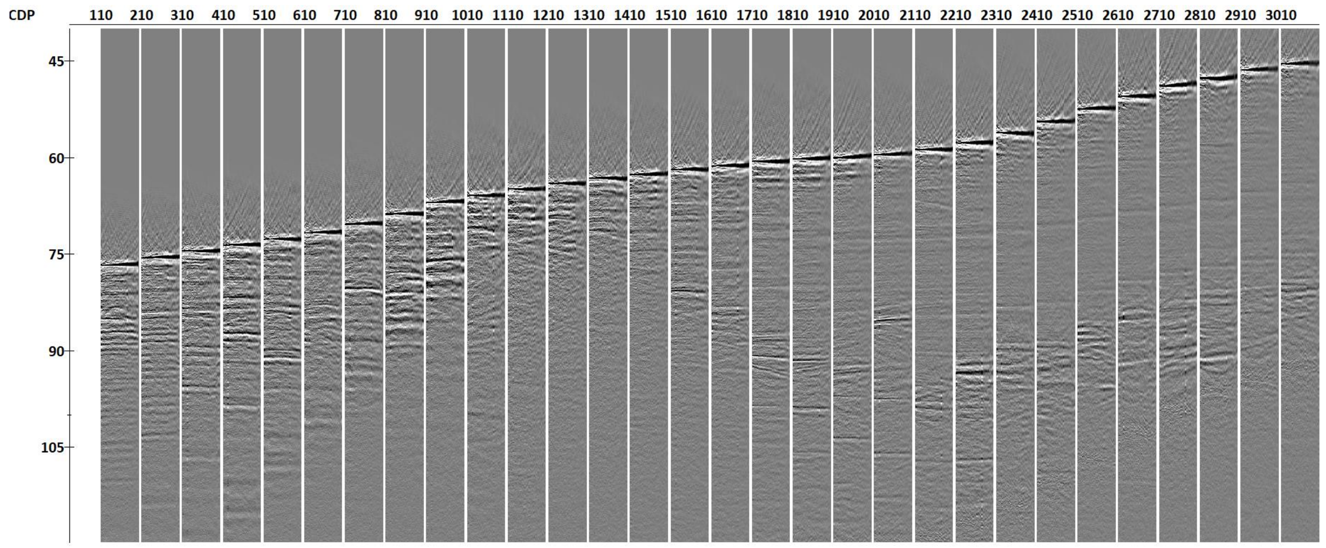
Aperture Parameters
Aperture (time[ms]:range[m]) 0:30
Example: 0:300, 500:700, 1000:1200
[✓] Apply Aperture Taper 10 %

[✓] Use anti-aliasing filter
Anti-aliasing DX 0.5 m
Anti-aliasing DY 0.5 m
Stretch muting 0 %
Max Frequency to migrate 8000 Hz

Shaping Filter: ( ) None [✓ 2D] ( ) 3D

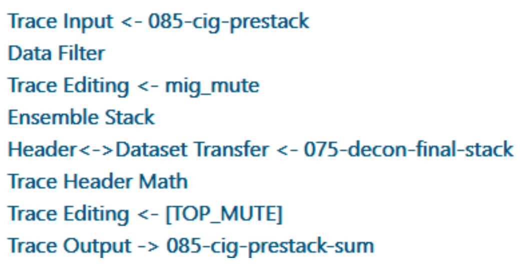
[✓] Migrate offset bins separately
Offset binning: Start (m) 8 Step (m) 1 Count 46

Pre/Post-Stack Kirchhoff Time Migration parameters for prestack migration



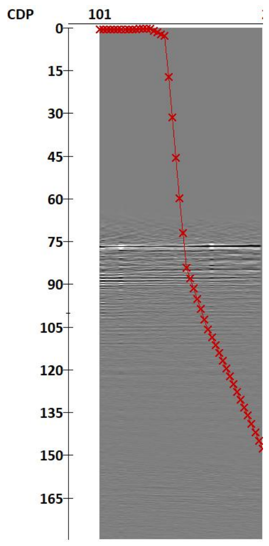
Common image gathers computed by prestack migration

Finally, we need to stack the common image gathers to obtain the final image. This happens in the flow **086-Migration-prestack-sum**. We first input the *085-cig-prestack* dataset and remove the CDP=-1 by Data Filter. Then, we conduct top muting in Trace Editing with a manually picked muting curve *mig_mute*. The muting curve was manually picked on one of the gathers in the Screen Display tool of the flow **085-Migration-prestack-QC** by selecting *Tools -> Pick -> New pick* (or just pressing the *N* hotkey). The pick was saved (*Ctrl-S*) with specified OFFSET-OFFSET headers (so that it can be applied to all CDPs). Next, the traces are stacked with Ensemble Stack and the data above the seafloor is muted like it was done in **081-Migration-poststack-postproc**. The image is written to *085-cig-prestack-sum*.



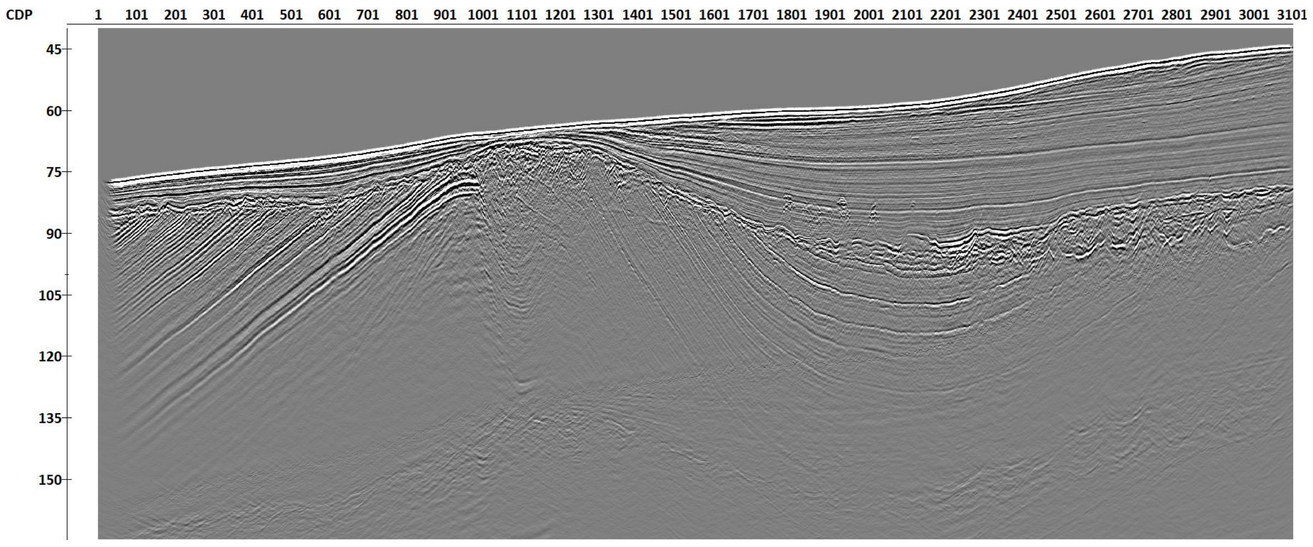
```
Trace Input <- 085-cig-prestack
Data Filter
Trace Editing <- mig_mute
Ensemble Stack
Header<->Dataset Transfer <- 075-decon-final-stack
Trace Header Math
Trace Editing <- [TOP_MUTE]
Trace Output -> 085-cig-prestack-sum
```

086-Migration-prestack-sum flow



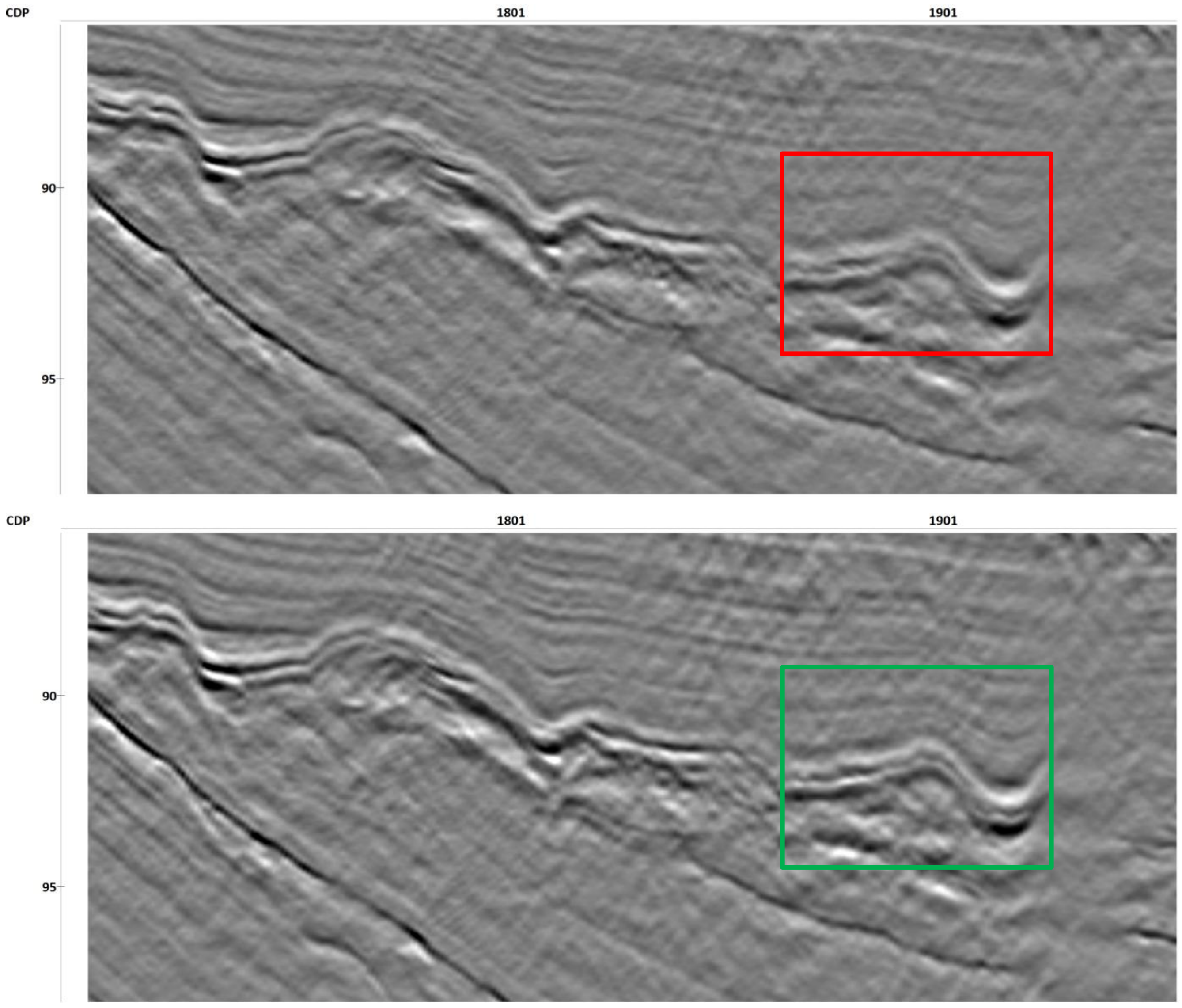
Picked muting curve

The flow **086-Migration-prestack-sum-QC** compares the pre- and post-stack migration images. In this case, they are very similar.



Pre-stack migration image

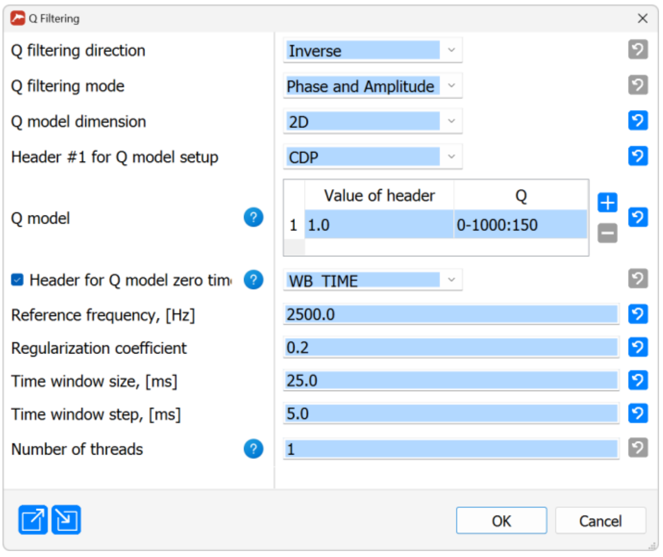
Still, for some parts of the seismic image (in particular, the zones with complex structure) the pre-stack migration can provide a superior result. In the figure below, we show an example where the pre-stack migration provides a slightly better-defined structure.



Post-stack (top) and pre-stack (bottom) migration images (zoom)

Q Filtering

As a final processing step, we introduce Q compensation with the Q Filtering module in the flow **090-Q-compensation**. Q filtering direction is set to *Inverse*, the Q filtering mode is *Phase and Amplitude* so that both phase and amplitude spectra are corrected. We use a one-dimensional Q model which starts from the seafloor. This can be obtained by using a *2D Q model dimension* with CDP as *Header for Q model setup*. *Q model* in our case is constant (150) below the seafloor, which is set by WB_TIME header. Above the seafloor, the medium (water) is considered to be non-attenuating. *Reference frequency, [Hz]* is chosen to be close to the dominant frequency of the wavelet spectrum to avoid shifting the events in the image because of Q compensation. The regularization equal to 0.2 allows us to avoid the overamplification of the high-frequency noise at later times (at the cost of a slightly lower compensation accuracy). The *Time window size* is chosen so that it can fit several wavelets, and the *step* is chosen so that is provides high enough accuracy of the compensation and fast enough computation (the most accurate computation occurs when *step* is equal to the sampling interval of the data, increasing the step from there makes the algorithm much quicker).



Q filtering direction	Inverse
Q filtering mode	Phase and Amplitude
Q model dimension	2D
Header #1 for Q model setup	CDP
Q model	
☑ Header for Q model zero tim	WB_TIME
Reference frequency, [Hz]	2500.0
Regularization coefficient	0.2
Time window size, [ms]	25.0
Time window step, [ms]	5.0
Number of threads	1

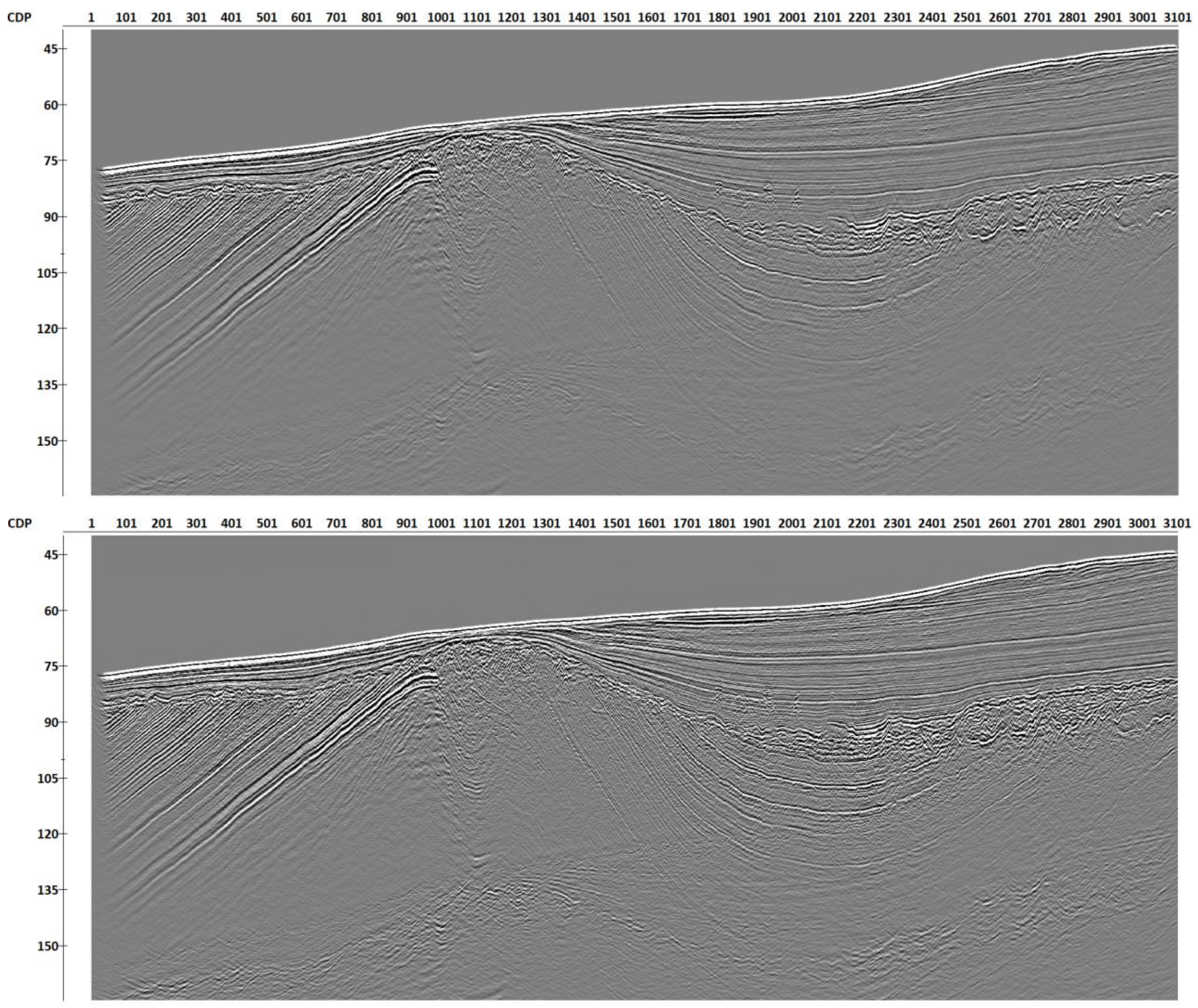
Q model:

	Value of header	Q
1	1.0	0-1000:150

Q Filtering parameters

Conclusion

In this tutorial, we have demonstrated the main processing procedures in a typical high-resolution offshore processing flow. Of course, additional procedures may be required depending on the data type. For example, some noise removal procedures may be needed (e.g., TFD Noise Attenuation) if the input data is noisy. Also, one may decide to conduct additional processing of common image gathers after migration (for long-offset data, one may remove the remaining multiple energy with 2D filters such as F-K Filter or Sparse Radon Filter). Still, we hope that the provided workflow is a suitable starting point.



Seismic images before (above) and after (below) Q compensation