Flin Flon, Manitoba
Vertical Seismic Profile (VSP) Survey 2006/2007:
First Processing of VSP Near Offset Dynamite Data

Barbara Dietiker and Don White

Geological Survey of Canada,
Seismology and Electromagnetism Section,
615 Booth St.,
Ottawa, Ontario K1A 0E2
Canada

Report May 2007

Abstract

As a part of the TGI-3 seismic program surface and VSP data were acquired in the Flin Flon VMS Mining Camp, Manitoba. The data serve as tests for future 3D seismic acquisition and as verification / reconnaissance of subsurface structures. Different seismic sources were tested and recorded both with downhole and surface geophones.

This report outlines the processing done on the vertical component of the VSP data acquired from near offset dynamite sources. In general DSISoft Software was used which offered a good variety of standard processing modules as well as flexibility and the option to change and write new modules.

On all three processed datasets reflections could be seen as expected from the rock property measurements and their resulting reflection coefficients. Generally, there is correlation between the strong reflections and some of the primary lithologic boundaries obtained from drill cores.

Contents

1 Introduction **1**

2 Borehole 4Q66W3 **3**
2.1 Processing Flow . . . 3
2.2 Processing Steps for Borehole 4Q66W3 . . . 3
2.2.1 Data Conversion . . . 3
2.2.2 Geometry: Correction of Receiver Coordinates . . . 3
2.2.3 Sorting . . . 6
2.2.4 DSISoft Module Update . . . 6
2.2.5 Trace Editing and Scaling . . . 6
2.2.6 Notch Filtering . . . 6
2.2.7 Median Filtering of Downgoing Energy . . . 7
2.2.8 F-k Filtering, Bandpass Filtering and Gain . . . 7
2.2.9 Corridor Stack . . . 8

3 Borehole FFM001 **17**
3.1 Processing Flow . . . 17
3.2 Processing Steps . . . 17
3.2.1 Data Conversion . . . 17
3.2.2 Geometry and Sorting . . . 19
3.2.3 Trace Editing . . . 19
3.2.4 Notch Filtering . . . 19
3.2.5 Removal of Downgoing P-Wave Energy . . . 20
3.2.6 Removal of Downgoing S-Wave Energy . . . 20
3.2.7 F-k filtering . . . 20
3.2.8 Shift to TWT . . . 20
3.2.9 Effect of the Deviated Borehole . . . 20
3.2.10 Corridor Stacks . . . 21
3.2.11 Synthetic Traces . . . 22

4 Borehole FFS039 **34**
4.1 Processing Flow . . . 34
4.2 Processing Steps for Borehole FFS039 . . . 34

4.2.1 Data Conversion . . . 34
4.2.2 Geometry: Correction of Receiver and Shot Coordinates . . . 34
4.2.3 Trace Editing and Scaling . . . 36
4.2.4 Notch Filtering . . . 36
4.2.5 Median Filtering of Downgoing Energy . . . 36
4.2.6 Bandpass and F-K Filtering . . . 42
4.2.7 Shift to TWT . . . 42
4.2.8 Effect of the Deviated Borehole . . . 42
4.2.9 Corridor Stacks . . . 48

A Enlarged Figures of 4Q66W3 near offset **53**
A.1 Removal of P-Wave Energy . . . 53
A.2 Removal of S-Wave Energy . . . 60
A.3 FK Filtering . . . 67
A.4 Bandpass Filtering . . . 72
A.5 Geological Logs . . . 79

B Enlarged Figures of FFM001 near offset **81**
B.1 Notch filtering . . . 81
B.2 Removal of P-Wave Energy . . . 85
B.3 Removal of S-Wave Energy . . . 89

C Enlarged Figures of FFS039 near offset **97**
C.1 Notch filtering . . . 97
C.2 Removal of P-Wave Energy . . . 100
C.3 Removal of S-Wave Energy . . . 107
C.4 Removal of Horizontal Energy . . . 113
C.5 Bandpass Filtering . . . 115

D New Modules for DSISoft **121**
D.1 time_scale . . . 121
D.2 cos_notch . . . 122
D.3 mute_Hann . . . 124
D.4 new_bandpass . . . 127
D.5 corrstack_t . . . 129
D.6 merge_traces . . . 130
D.7 window_mute4 . . . 131
D.8 norm_stack . . . 133
D.9 cos_notch_test1 . . . 134
D.10 txdirect1 . . . 136
D.11 txplane2 . . . 137
D.12 cos_notch_newtest . . . 138

E Processing Scripts 4Q66W3 near offset **141**
E.1 convert from .segy to .mat (sc_segy2mat.m) . . . 141
E.2 correct wrong trace headers (sc_write_th.m) . . . 142
E.3 sort data (sc_sort_many2.m) . . . 144
E.4 more_geometry (sc_geom.m) . . . 145
E.5 remove horizontal components and store P-times (sc_mod5.m) . . . 146
E.6 processing of the raw, sorted data (sc_all5.m) . . . 147
E.7 create 'near' corridor stack (sc_mute_corr10.m) . . . 150
E.8 create 'mid' corridor stack (sc_corr_mute12.m) . . . 151
E.9 create 'far' corridor stack (sc_corr_mute11.m) . . . 152
E.10 creates plots (sc_plotPremove.m) . . . 153
E.11 creates plot (sc_plot_FK.m) . . . 157
E.12 bandpass filter test (sc_bp_filter_Test.m) . . . 160
E.13 creates plot (sc_plot_BP.m) . . . 161
E.14 creates geometry (sc_plot_geom.m) . . . 162

F Processing Scripts FFM001 near offset **163**
F.1 convert from .seg2 to .mat (get_data.m) . . . 163
F.2 write primary trace headers and sort (load_data.m) . . . 164
F.3 write trace headers (sc_write_th.m) . . . 165
F.4 trace editing (sc_filt1.m) . . . 166
F.5 write missing source locations (sc_more_tha.m) . . . 167
F.6 processing applied to the preprocessed data(sc_new_all.m) . . . 168
F.7 create a corridor stack of the whole dataset(sc_corr_new_all.m) . . . 171
F.8 create corridor stack (sc_corr_40... .m) . . . 172
F.9 merges all corridor stacks into one file(sc_new_comp_corr2.m) . . . 173
F.10 creates plot (sc_plot_fi4) . . . 174
F.11 creates plot (sc_plot_remR.m) . . . 176
F.12 creates plot (sc_plot_remS1.m) . . . 178
F.13 displays the geometries (plot_geom1.m) . . . 181
F.14 calculate reflection point, traveltime and travelled distance (VSP_DSI2a.m) . . . 183
F.15 calculates the maximum traces within the threshold (find_dist1.m) . . . 184
F.16 calculates synthetic traces (sc_synTraces2.m) . . . 185

G Processing Scripts FFS039 near offset **187**
G.1 convert from .seg2 to .mat (sc_segy2mat.m) . . . 187
G.2 write trace headers (sc_write_th.m) . . . 188
G.3 more trace headers (sc_more_th3.m) . . . 189
G.4 plot geometry of FFS039 (plot_BH_FS39_cor5.m) . . . 190
G.5 remove H1 and H2 (sc_subset3.m) . . . 192
G.6 processing applied to the data (sc_all26.m) . . . 193
G.7 create a corridor stack of the whole dataset(sc_twt_new.m) . . . 196
G.8 create corridor stack (sc_corr_28... .m) . . . 197

G.9 merges all corridor stacks into one file(sc_new_comp_corr.m) 198
G.10 creates plot (sc_plot_all.m) 199
G.11 bandpass filter test (sc_bp_Test2.m) 204
G.12 calculate reflection point, traveltime and travelled distance (VSP_DSI2a.m) . . 206
G.13 calculate reflection point, traveltime and travelled distance (VSP_DSI1b.m) . . 207
G.14 calculates the maximum traces within the threshold (find_dist1.m) 208
G.15 plot geometry of the borehole and raypaths(plot_geom1.m) 209

Chapter 1

Introduction

Zero-offset dynamite data were acquired in boreholes 4Q66W3, FFM001 and FFS039 within the Flin Flon Mining Camp. An eight-level, three component, clamping geophone tool was lowered into the boreholes. The distance between the levels was 10 m. For an optimal borehole geophone coupling the geophone was pressed against the borehole wall by a clamping arm. The vertical geophones were aligned along the dip of the borehole whereas both horizontal receivers had arbitrary orientations and need to be aligned mathematically during the processing. After recording, the tool was pulled upward 5 or 75 m to obtain an overall receiver spacing of 5 m. The near offset sources were located at approximately 5 m depth and approximately 10 m offset from the boreholes. Figure 1.1 shows all borehole locations, source positions and surface profiles.
The following three chapters focus on each borehole. The data itself, the borehole geometry and the applied processing steps are described and resulting data sections are shown. Processing scripts of the DSISoft software and new modules can be found in the Appendices. Full page sized versions of the data figures are also included in the Appendices.

Generally, in the text to follow, *italic* typesetting refers to processing scripts, new modules and any MatLab files used. All these files can be found on the accompanying DVD (in the processing folder).
`These filenames` refer to additional Excel or text files, also put onto the DVD.
All small Figures from the text and large copies from the Appendix are found on the DVD in the specific folders or subfolders. All Figures, obtained directly from processing outputs, which were used as basis for Figures with accentuated areas are included as well.
Additionally, presented posters can be found on the DVD.

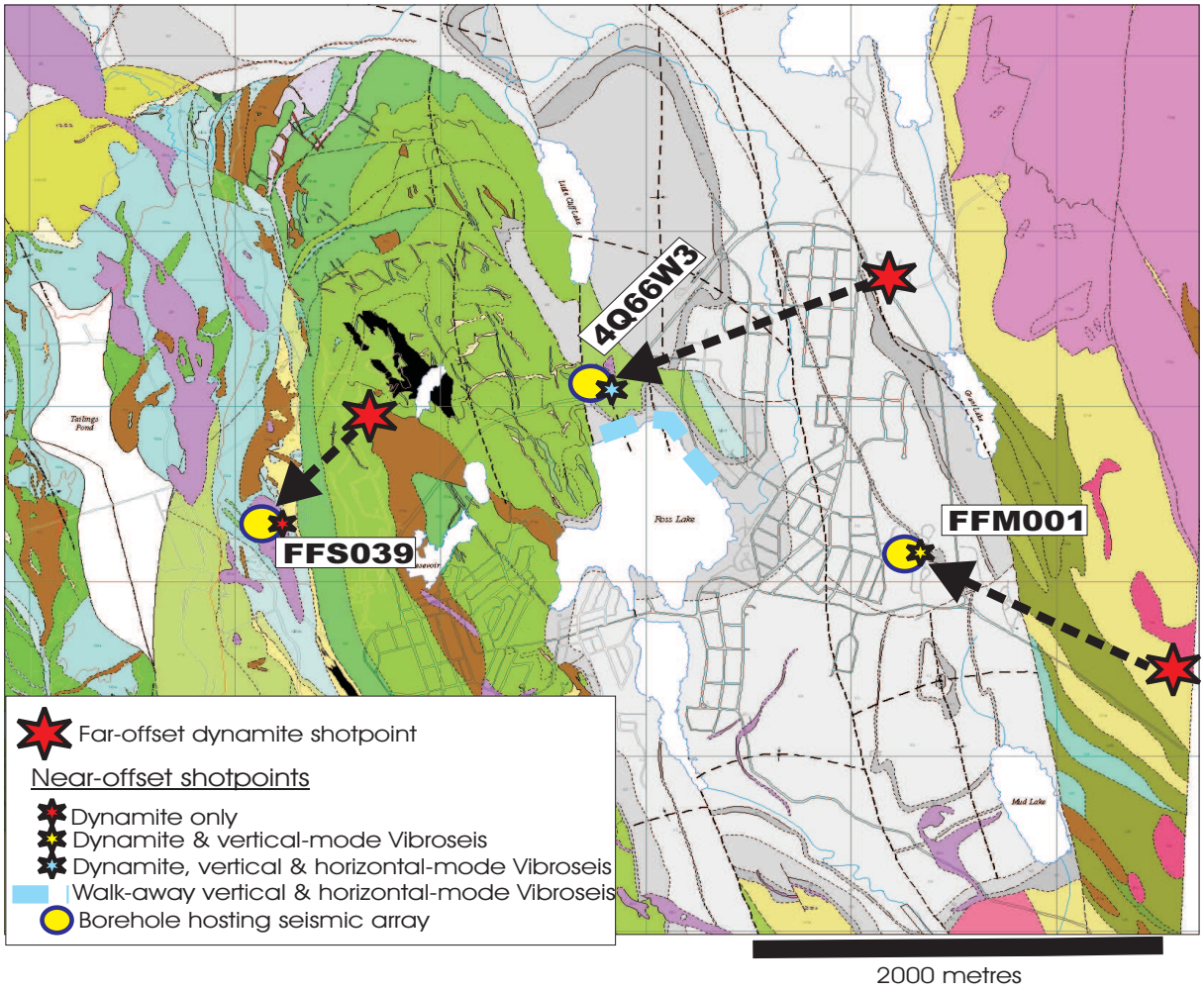

Figure 1.1: Geological map with borehole locations, near- and far-offset shot positions as well as the surface profiles and sources. The local road network in Flin Flon is shown as an overlay on the geology.

Chapter 2

Borehole 4Q66W3

Borehole 4Q66W3 is the northernmost of the three boreholes. It is located north of Ross Lake. It extends to 2130 m below surface whereas seismic data were recorded from 161.5 to 1116.5 m below surface, covering an overall distance of 955 m. The dataset consists of 768 traces, 192 vertical component, 192 horizontal component (H1), 192 horizontal component (H2) and 192 surface pilot traces. Here, we report on processing of the vertical component data only.

2.1 Processing Flow

Table 2.1 gives an overview of the relevant processing steps for the vertical component data from borehole 4Q66W3. Each step is described in more detail in the Section 2.2. All of the entire DSISoft processing scripts are shown in Appendix E.

2.2 Processing Steps for Borehole 4Q66W3

The following sections describe all applied processing steps in detail. The corresponding processing scripts are found in Appendix E.

2.2.1 Data Conversion

Since data processing steps was to be done with the DSISoft software, the data first needed to be converted into the specific format. The script *sc_segy2mat* was used (Appendix E.1). Problems arose reading the right header words into the file and trace headers. While solving these difficulties an in-depth check on all acquisition information was performed.

2.2.2 Geometry: Correction of Receiver Coordinates

The receiver coordinates in the data headers proved to be wrong. The observer log of the drillhole coordinates (`gems_drillhole_umt_trace.txt`) served as the basis for the corrected input data. According to the wireline depth of each receiver station the new coordinates were

Table 2.1: shows the applied processing flow.

Data Conversion
The dataset is converted from SEGY to the DSISoft format.
Geometry1
File and trace headers are written into the dataset. Incorrect headers are corrected.
Sort
The dataset is sorted into its components and according to wireline depth.
Geometry2
The shot to receiver azimuth and offset are calculated and written to the trace headers. Both horizontal components are removed from the dataset.
Picking of First Arrivals (P- and S-waves)
First arrival times of P- and S-waves are picked and transferred to the trace headers.
P-wave Energy Removal
P-wave energy is removed by muting the trace up to 5ms after the first arrival.
Electrical Noise Removal
For each frequency peak a notch filter is applied.
More Top Mute
More P-wave energy is removed by using a top mute.
Data Enhancement
A broad bandpass filter is applied.
S-wave Energy Removal
S-wave energy is removed by using an 11 traces wide median filter.
FK- and BP Filtering

interpolated and written to the trace header using the script *sc_write_th* (Appendix E.2). As a test, Figure 2.1 shows the interpolated coordinates (red) plotted on top of the drillhole coordinates. It also shows the deviation of the borehole. This plot was produced with the script *sc_plot_geom* found in Appendix E.14. Finally the corrected values were written to each trace header.
The geometry information now consists of wireline depth, receiver coordinates (elevation, northing, easting), source coordinates and recording components. All coordinates were converted into **Mine UTM**. Table 2.2 shows all header words that describe the geometry, including their location within the trace header. Additionally, the format and its Byte length are specified.
In a second step more geometry (Geometry2) was applied: *more_geometry* (Appendix E.4). This calculated shot-receiver offsets and angles.

Table 2.2: All assigned trace header values.

Headerword	**Description**	**Format**
1	original trace number	int32
2	shot number	int32
4	component	int32
5	rotation angle	int32
8	channel number	int16
9	shot receiver azimuth	int32
12	number of traces per record	int16
13	trace number within record	int16
15	first break pick time P	int32
16 - 18	different muting times	int32
19	first break pick time S	int32
26	shot point number	int16
27	rec station number	int16
28	old shot receiver offset	int32
29	source north	int32
30	time delay (in samples?)	int16
31	source east	int32
33	source elevation	int32
35	new rec north	int32
36	old rec north	int32
37	new rec east	int32
38	old rec east	int32
39	new rec elevation	int32
40	old rec elevation	int32
53	shot receiver offset	int32
54	shot depth	int32
56	wireline depth	int32

2.2.3 Sorting

The data were sorted by component and receiver elevation, wireline depth respectively using the script *sc_sort_many2* (Appendix E.3). In a first step only the vertical component was processed and therefore both horizontal components were removed: *sc_mod5.m* (Appendix E.5).

2.2.4 DSISoft Module Update

Some DSISoft modules didn't work properly and needed to be changed. The new modules can be found in the Appendix D.

2.2.5 Trace Editing and Scaling

Some extremely noisy traces were observed and marked for later muting. Due to geometrical spreading and attenuation the of the uncorrected amplitudes decay with time. Several correction functions were tested. Finally a power formula seemed most suitable. Each sample is multiplied by $t^{1.7}$. The DSISoft module written for this purpose is *time_scale* shown in Appendix D.1.
After this step the preprocessing was finished. The next three steps were tested individually, but for the final version they were combined in one step called *sc_all5*. The script can be found in Appendix E.6.

2.2.6 Notch Filtering

In order to apply appropriate filtering the frequency spectrum of the raw vertical component data was investigated. This was done with the Promax software after the data were converted into the SEGY format using the DSISoft module *mat2segy*. The file used for conversion of the headers is `new_cross3.crs`. Figure 2.2 shows the amplitude spectrum of the scaled raw data from different time windows in dashed lines. Solid lines mark the amplitudes after filtering. The first time window (0-70 ms) recorded the background noise before the first signals arrived. The spectra of the raw data (dashed lines) show extremely high frequency peaks, especially in the later time windows.
These frequency peaks were removed using one notch filter for each peak. The solid lines show that these noise peaks were removed effectively without removing too much data. The general power of the notch filtered curves is lower because the first P-wave arrivals were muted prior to notch filtering.
Very narrow filters with steep limits were used. To be able to use these filters without adding ringing to the data, cosine shaped tapers are introduced in the new function *cos_notch* (see Appendix D.2).

2.2.7 Median Filtering of Downgoing Energy

Since we are only interested in the upgoing reflection energy, downgoing P- and S-wave energy had to be removed.

P-Wave Energy
The P-wave energy was removed in two steps. First, the data were muted up 5 ms after the picked first arrival times. In a second step, the first breaks were aligned before the remaining P energy was muted. Figure 2.3 shows data after the different steps were applied. *Sc_plotPremove*, displayed in Appendix E.10, was to produce and display this Figure of removing the P-wave energy.
An average P-wave velocity could be calculated from the first breaks. This value of 6210 m/s ±100 m/s (assuming the picks are off 0.5 wavelength) matches the values obtained from the Flin Flon area at the Rock Properties Lab.

S-Wave Energy
The data were flattened according to the arrival times before a median filter was applied. The median filtered data were then subtracted from the flattened data, before the result was shifted back to normal travel times. Figure 2.4 shows the removal of the downgoing S-wave energy. First break arrival times display an average S-wave velocity of 3680 m/s ±200 m/s.

2.2.8 F-k Filtering, Bandpass Filtering and Gain

After removal of the downgoing energy, the data still show some multiples of P- and S-waves which were not completely removed. Use of an f-k filter therefore seemed appropriate. Different rejection polygons were tested and finally the smallest polygons that were effective were chosen. Figure 2.5 shows the f-k spectrum of the raw data on the left. The P-wave energy is overpowering. The middle panel displays the f-k spectrum before the S-wave energy is removed (after P-removal and notch filtering). The S-wave, dipping less than the P-wave, is barely visible. The 180 and 300 Hz peaks have obviously been removed. The right spectrum shows the data after S-wave energy removal, before the f-k filters were applied. The two rejecton polygons are marked.
Figure 2.6 displays the data before (upper left) and after f-k filtering (upper right). On the lower left panel a bandpass filter was applied on the f-k filtered data. The script *sc_plot_FK* (Appendix E.11) has been written to produce and display this Figure. The bandpass parameters were 50, 71, 150, 220 Hz. The lower right panel shows the data after a top mute. This is the data which was used as input for the corridor stacks.
After f-k filtering some reflections became clearer. For example the reflection around trace 100 and 0.4 s depth is more pronounced. Reflections around 0.55 s (around trace 100) have become more continuous.
The bandpass filter was applied for visualization purposes. Since in this case it is easier to recognize reflections in data with lower frequency content.
The limits for this bandpass filter were selected from a frequency analysis of the data after

scaling and P-energy removal done with the script *sc_bp_filter_Test* (Appendix E.12). Figure 2.7 shows the data after bandpass filter with different parameters applied (*sc_plot_BP* Appendix E.13). Those with the clearest upgoing reflections were chosen. For example some prominent reflections are visible at 0.3 s between trace 1 and 20.
Finally parameters of 50/71 for the lower and 150/220 Hz for the upper limit were chosen. A cosine taper fuction was applied in the frequency domain (see Appendix D.4 for the new module *new_bandpass*). An AGC of 80 ms was only applied for displaying the data. There is no AGC applied within the different processing steps.

2.2.9 Corridor Stack

To compare the VSP with geophysical logs a corridor stack was created. In a corridor stack, composite trace or vertical summation the upgoing energy (i.e., reflections) will be positively reinforeced provided that the reflectors are flat and horizontal. Only in this case, the reflections of the two-way-traveltime (TWT) shifted data line up horizontally and can be summed. The downgoing reflections are misaligned and therefore they will be destructively superposed.
Each data trace was shifted by its P-wave first arrival time to line up the upgoing reflections horizontally. The data was summed by the new function *corrstack_t* which doesn't convert time to depth (see Appendix D.5). To easier display data and corridor stack a new function was written to merge them into one dataset: *merge_traces* (Appendix D.6). But since the data also contains noisier areas which should not be summed, it needed to be restricted. The new function *window_mute4* (Appendix D.7) was written for this purpose. It also makes sure that the function *norm_stack* (Appendix D.8) normalizes the summed values by the number of traces summed over. Three different scripts were used to create the "near" (Appendix E.7), "mid" (Appendix E.8) and "far" stack (Appendix E.9).
Figure 2.8 shows the TWT shifted data and the different corridor stacks. The red corridor stack displays data from the vicinity of the borehole. Two prominent reflections are visible at 0.50 and 0.53 s. They were recorded in 1050 and 950 m depth. The same reflections are visible at slightly later times in the blue stack. They are recorded in 700-480 m and 730 - 550 m depth, resp. Further clear reflections are observable at 0.64, 0.68 and 0.74 s in the blue stack originating from depths larger than 1000 m. Probably the same reflections are visible at later times in the green stack. The mismatch of these three stacks can be explained by dipping discontinuities, the deviation of the borehole and the uncorrected normal move-out. For further comparisons the lithologic log is placed beside the stacks. A larger picture of the lithologic log and its legend can be found in Appendix A.5. First observations show that the stacks match the lithologic logs nicely. More precise conclusions can only be drawn after velocity and density logs are available. Then synthetic seismograms can be calculated and serve as further tests and comparisons.

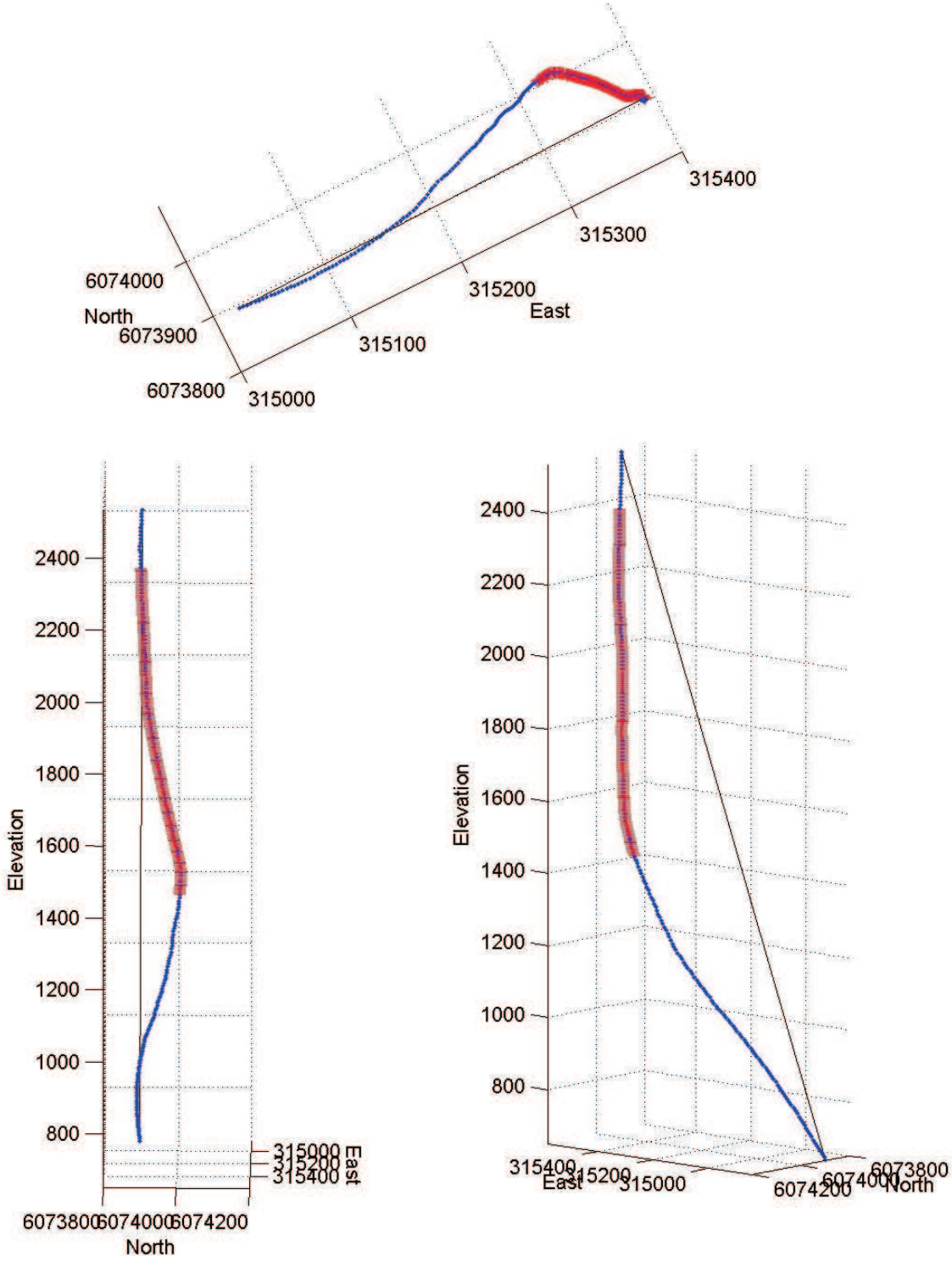

Figure 2.1: Borehole from different angles. Top: birdeye view; left: "in line" with the borehole plane; right: "orthogonal" to the borehole plane. The black line marks the direct connection from top to bottom. The blue line marks the borehole coordinates and red is the depth covered by the VSP.

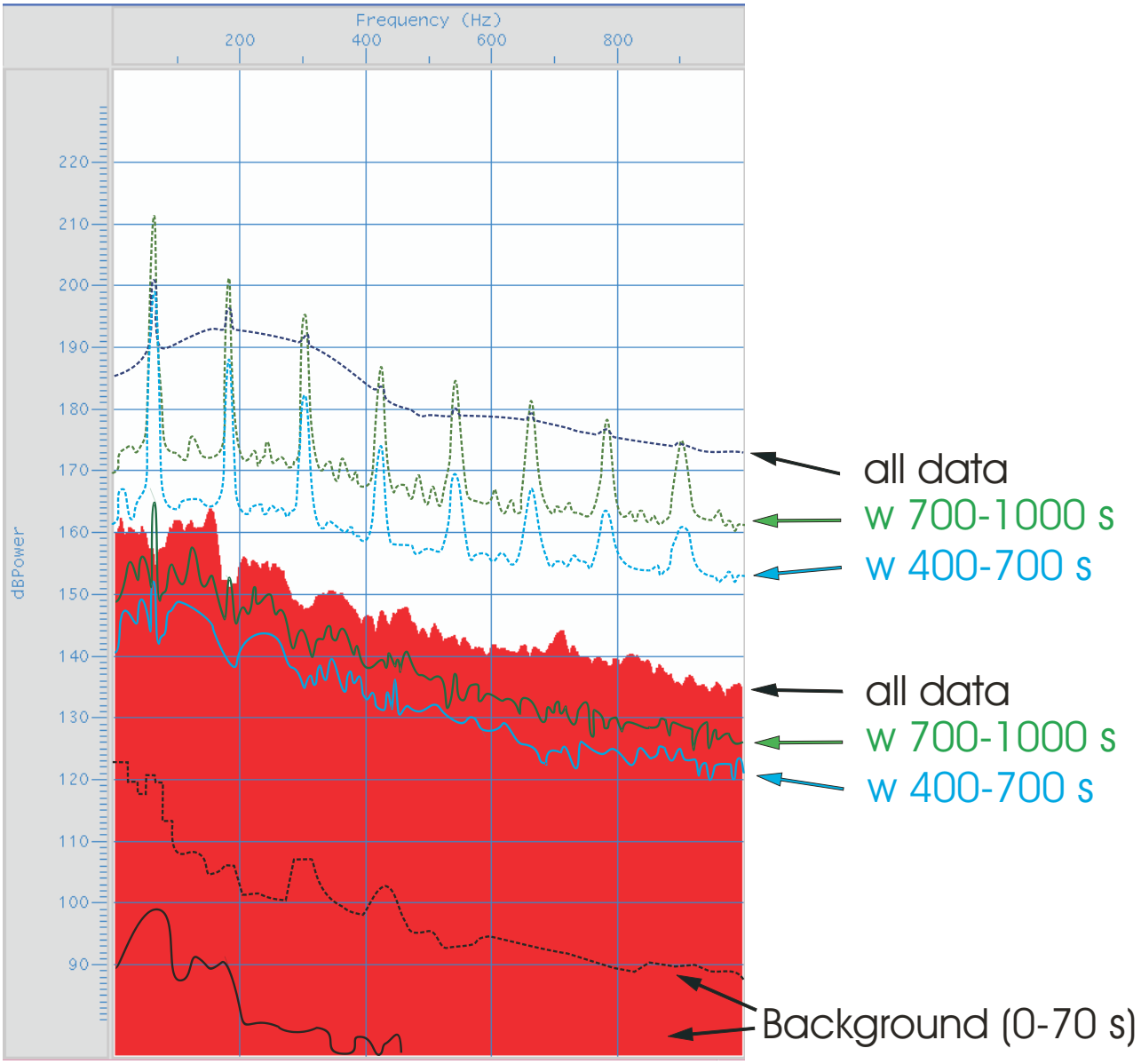

Figure 2.2: Amplitude frequency spectrum of the filtered, scaled dataset in red. The dashed lines mark the amplitudes of the raw, but scaled data from different time windows. The solid lines show the amplitudes of the filtered data from time windows of 0 to 70 ms, 400 to 700 ms and 700 to 1000 ms. The general power is lower because the P-wave energy was removed prior to notch filtering.

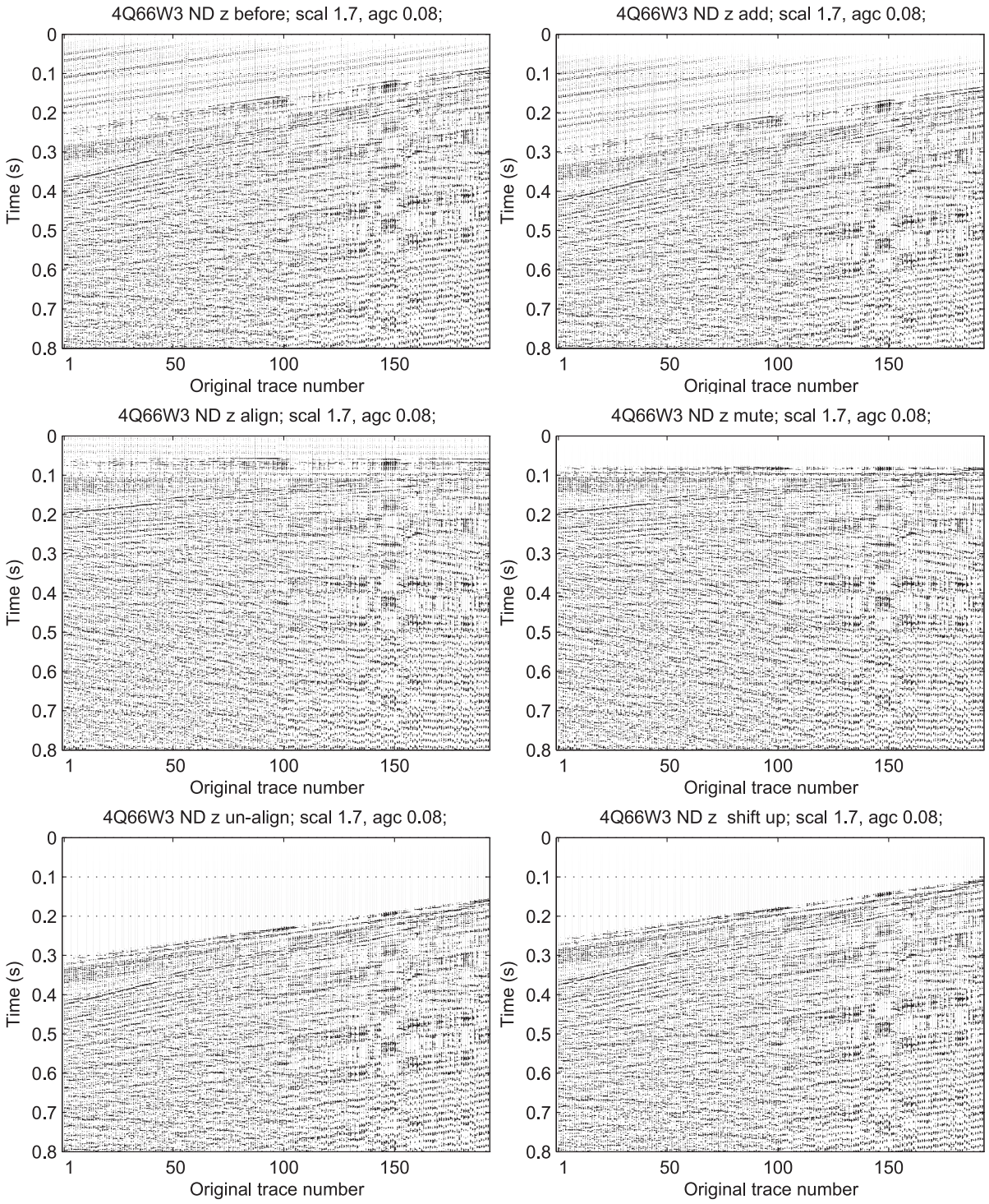

Figure 2.3: Removal of the P-wave energy. Figures from upper left to lower right: z-component before removal of P-wave energy; time is added to make sure no data is lost; data aligned at first break; data after top muting; data after shifting back of first break times; result after removing the added time. In Appendix A.1 the enlarged panels can be found.

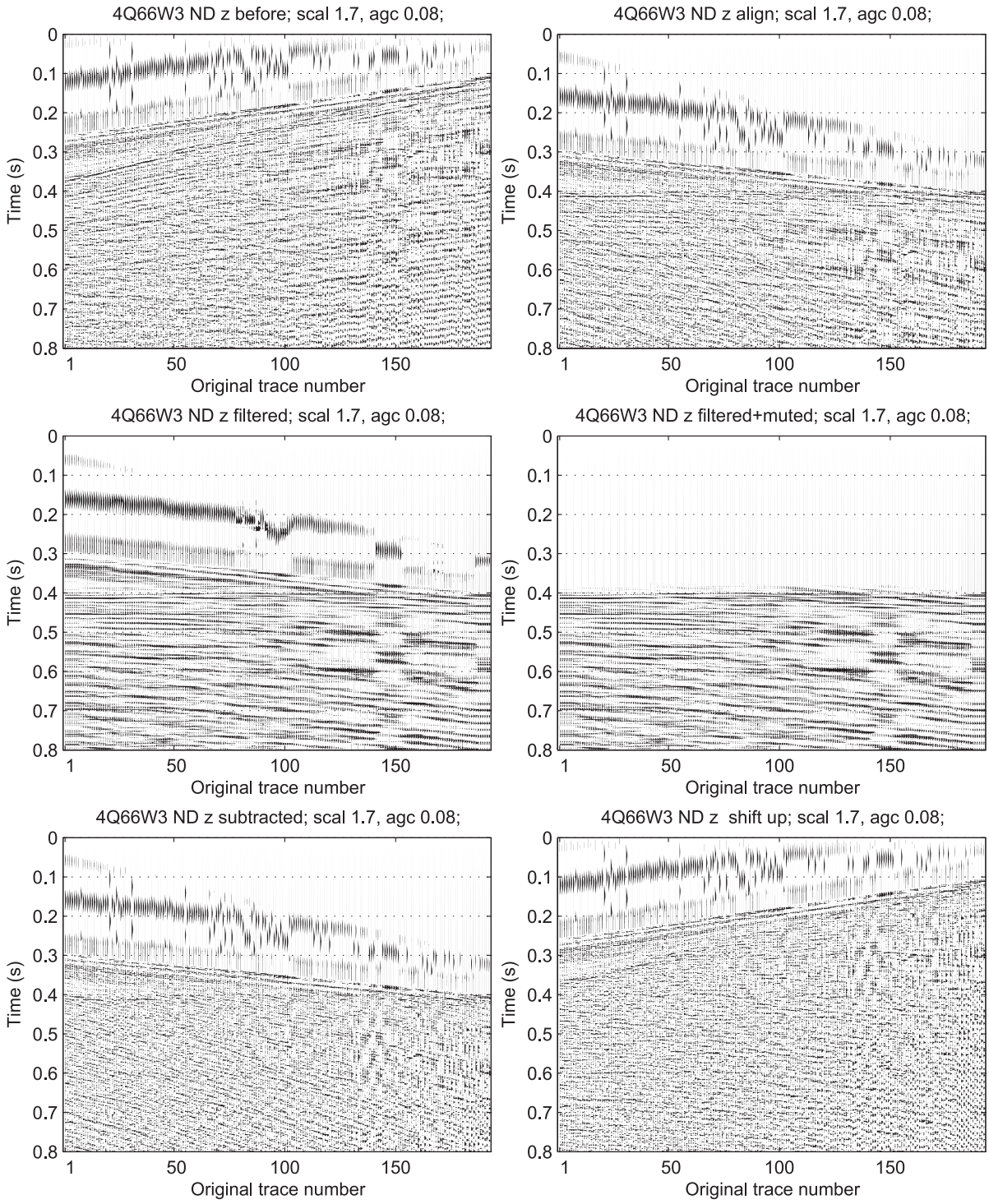

Figure 2.4: Removal of the S-wave energy. Figures from upper left to lower right: z-component before removal of S-wave energy; time is added to make sure no data are lost before they are shifted according the first arrival times; median filtered data; filtered data after muting; data after the filtered and muted dataset is subtracted; shifting back of first breaks and removal of the added time. In Appendix A.2 the enlarged panels can be found.

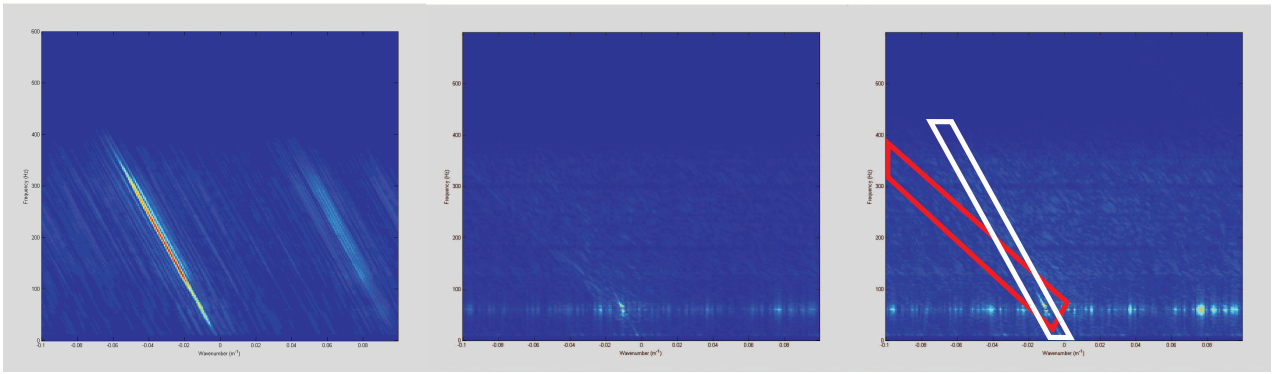

Figure 2.5: Different f-k spectra: On the left is the spectrum of the raw data. The middle panel displays the spectrum of the data after notch filtering, before removal of the S-wave energy. On the right is the spectrum of the data before f-k filtering was applied. The rejecting polygons are shown.

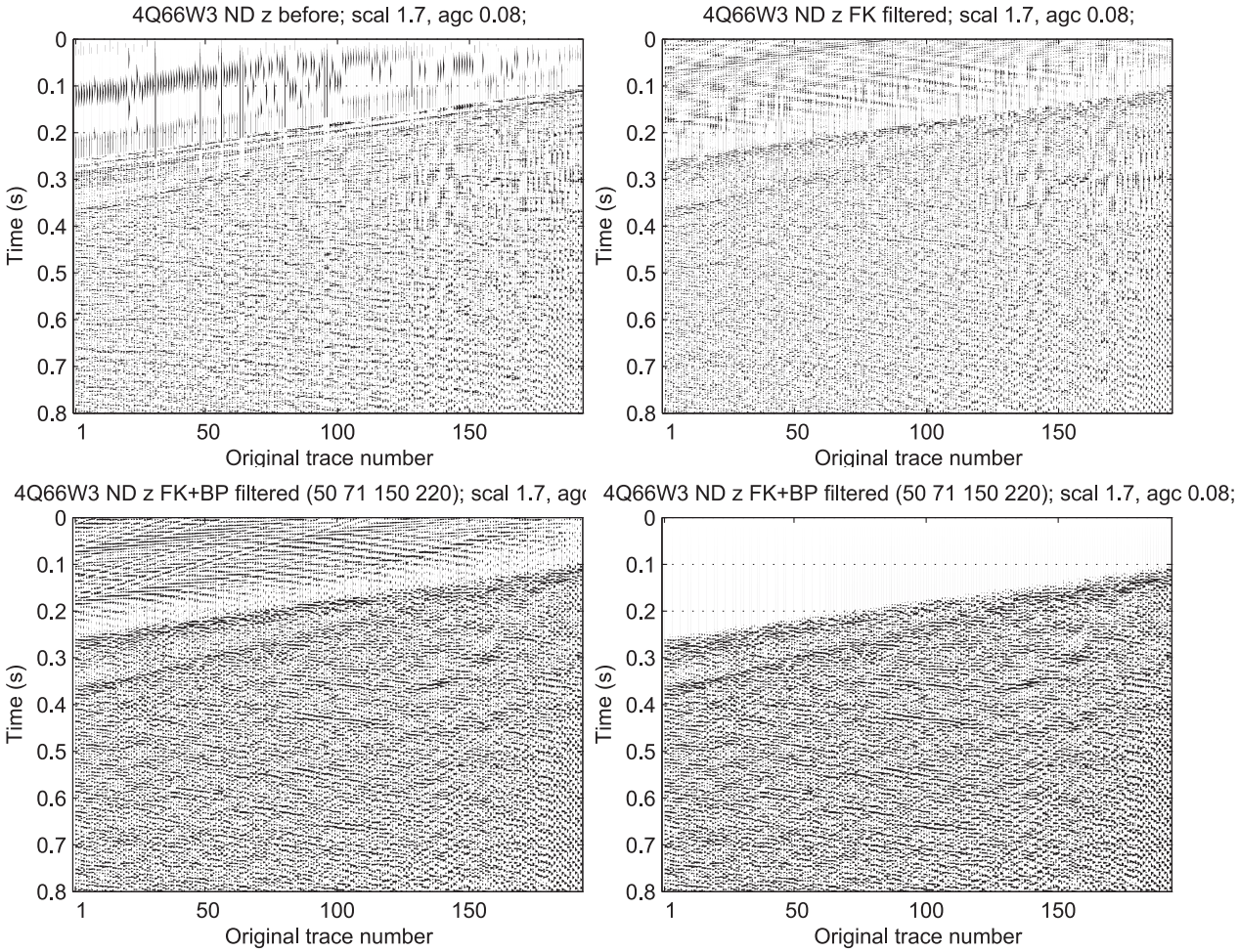

Figure 2.6: From upper left to lower right: Data before f-k filtering; f-k filtered data; f-k and bandpass filtered data; result after top muting. Note that all sections are scaled equally and have an AGC of 80 ms applied. In Appendix A.3 the enlarged panels can be found.

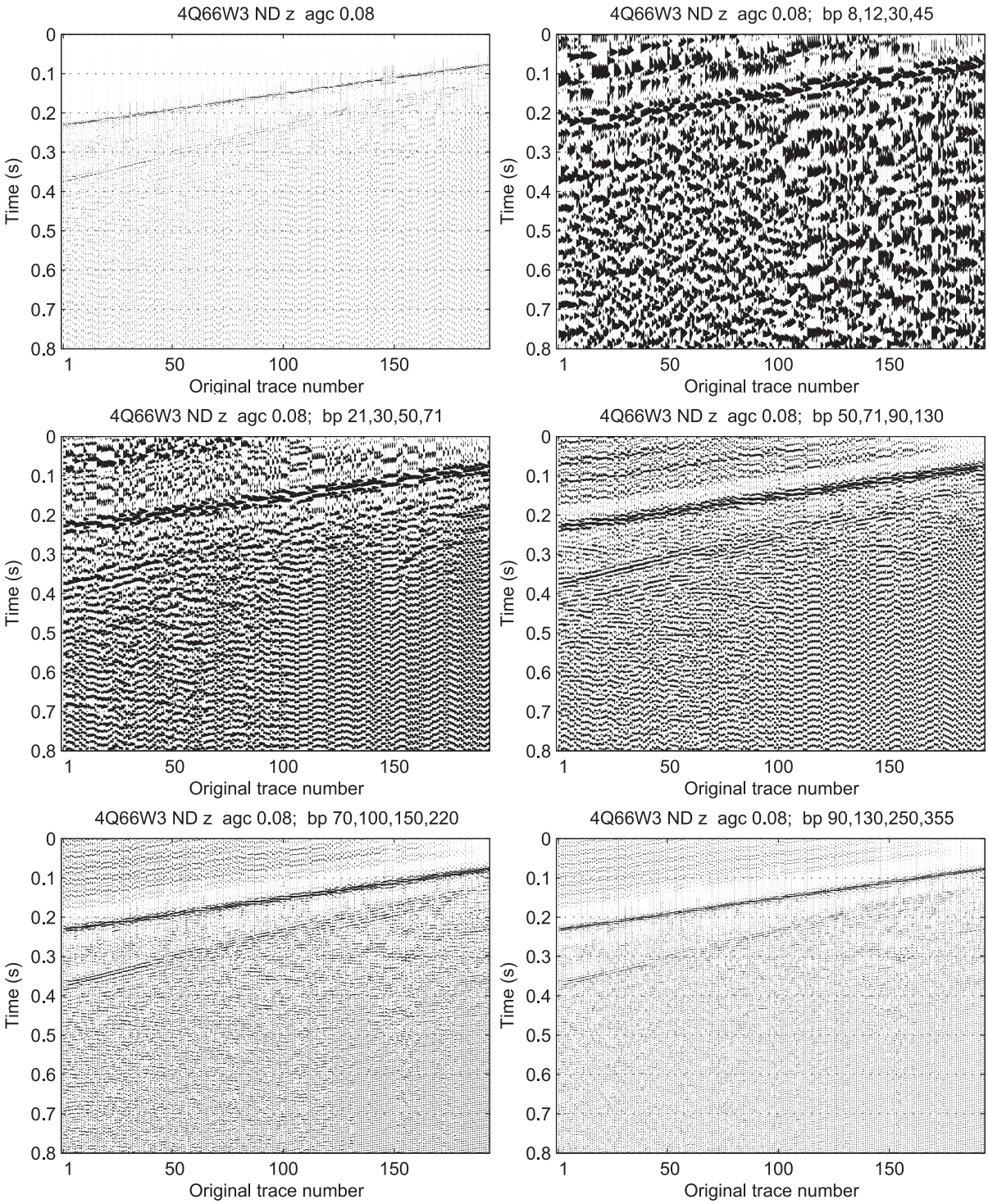

Figure 2.7: The first arrival muted, scaled data are shown after different bandpass were applied. Note: no notch filters have been applied to remove the electrical noise. The upper left figure shows the unfiltered data, figure captions specify the applied filters. An AGC of 80 ms was applied on each dataset. In Appendix A.4 the enlarged panels can be found.

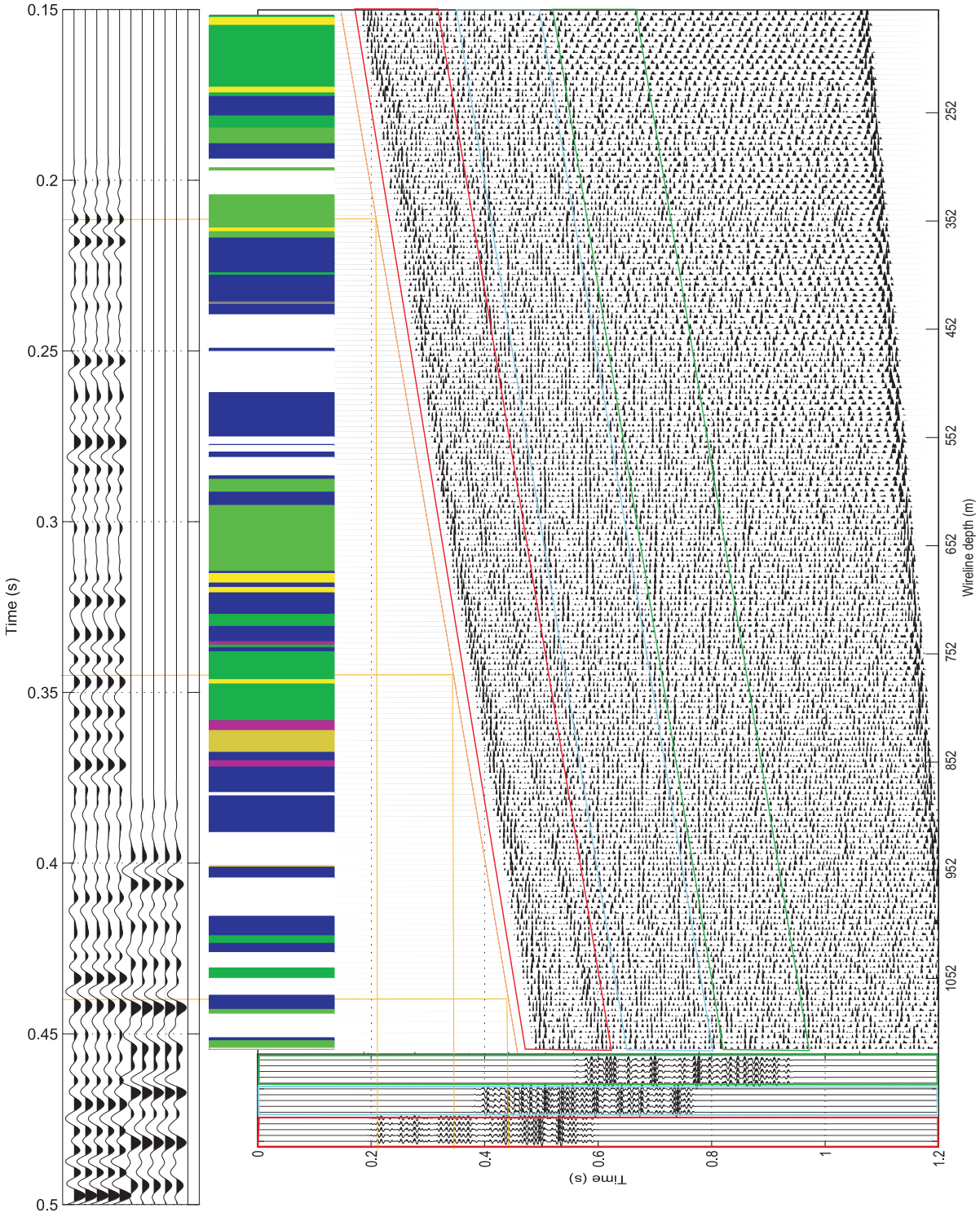

Figure 2.8: TWT shifted data where three corridor stack windows are marked. The summation traces are displayed with the same color frame on the bottom. On the left the depth converted stacks and the lithologic log are displayed. The orange lines show how the stacks are stretched in regard to the first breaks (orange diagonal). In Appendix A.5 the lithologic log and its legend are shown.

Chapter 3

Borehole FFM001

Borehole FFM001 is the easternmost borehole. It is located east of the town on meta-sedimentary rock. It is 1790 m deep. The VSP survey was conducted from 330 to 1765 m below surface leading to 288 receiver depth points. The whole dataset consists of 1152 traces: 288 vertical component, 288 horizontal component (H1), 288 horizontal component (H2) and 288 surface and pilot traces.

3.1 Processing Flow

Table 3.1 gives an overview of the relevant processing steps applied to the data of borehole FFM001. Generally, the processing flow is the same as for borehole 4Q66W3. Here are a few exceptions. 1) The starting point was single, near offset SEG2-files. Therefore sorting and geometry application was done differently. 2) Trace editing was necessary to restrict trace lengths to 1 s and to remove the low frequency tube waves. 3) No data enhancement was necessary since the low frequency has already been removed. Extremely noisy traces were marked but finally no traces were killed. 4) Filtering was done differently: no f-k and the bandpass filter restricts data to 100 to 200 Hz (71 to 150 Hz for 4Q66W3).

3.2 Processing Steps

This section describes each processing step for the vertical data component. In Appendix F the entire DSISoft processing scripts are shown.

3.2.1 Data Conversion

With the help of the observer's log (`Flin Flon Notes.pdf`) all near offset dynamite data files were identified. These files listed in `NeraDyn.txt` were converted from SEG2 into one single DSISoft format file using the script *get_data* found in Appendix F.1.

Table 3.1: The applied processing flow.

Data Conversion
The dataset is converted from SEG2 to the DSISoft format. `NeraDyn.txt` contains all file names with near dynamite data.
Geometry1
Primary file and trace headers are written into the dataset. Depth information is in the file `Depth.txt`.
Sort
The dataset is sorted into its components and according to wireline depth.
Geometry2
The receiver positions are copied from the file `Receivers.txt` into the trace headers.
Trace Editing
The trace length is reduced to 1 s. A first broad bandpass filter is applied to remove the powerful tube waves. All horizontal components are removed.
Picking of First Arrivals (P- and S-waves)
First arrival times of P- and S-waves are picked and transferred to the trace headers.
P-wave Energy Removal
P-wave energy is removed by muting the trace up to 5ms after the first arrival.
Electrical Noise Removal
For each frequency peak a notch filter is applied.
More Top Mute
More P-wave energy is removed by using a top mute.
S-wave Energy Removal
S-wave energy is removed by using an 11 traces wide median filter.
BP Filtering
TWT shift

3.2.2 Geometry and Sorting

The geometry was applied in two steps: First the primary headers containing receiver depths (`depths.txt`) and component information were written to the trace headers (*load_data* Appendix F.2). With these two headers the data were sorted according to component and receiver depth (wireline depth), respectively. In a second step all remaining geometry information from file `Receivers.txt` was written to the trace headers (*sc_write_th* Appendix F.3). Furthermore the missing source locations were included. For this purpose the script *sc_more_tha* found in Appendix F.5 was written. Figure 3.1 shows borehole FFM001 from different angles - the deviation becomes visible.

3.2.3 Trace Editing

The raw traces contain 3 s of data. The first 1.2 s of the dataset, added because of an unknown trigger delay, were removed. This trace length was then reduced to 1 s. To remove the disturbingly strong tube waves the data were filtered with a low-cut filter. Frequencies below 40 Hz were completely cut out. The two horizontal components were removed. To make up for attenuation and geometrical spreading the trace samples were multiplied by the time to the power of 1.7. This also allowed easier picking of P- and S-wave first breaks. All this is summarized in script *sc_filt1* in Appendix F.4.
The next three steps of the processing were tested individually before they were put together into one step called *sc_new_all* which can be found in Appendix F.6. Figure 3.2 shows the raw but scaled data. The P-wave is marked blue, the S-wave is orange and a prominent reflection is marked green. The tube waves are marked with red lines.

3.2.4 Notch Filtering

The frequency amplitude spectrum of the raw but scaled data was examined with Promax. Very high amplitude peaks were found between 18 and 42 Hz and spiking at 60, 180, 300, 361, 422, 541, 662, 719, 781 and 900 Hz. These frequencies were removed using a notch filter for each peak frequency. The frequency amplitude spectra of the trace edited and notch filtered data are shown in Figure 3.3.
After applying this first notch filter, some very noisy areas still remained in the dataset. Therefore the worst area between traces 207 and 240 (wireline depth 1525 and 1360 m in Figure 3.4, top panel) was filtered differently with a second, new notch filter (*cos_notch_test1* Appendix D.9). The following was done in this second notch filtering step: After the high cut filter removed frequencies above 300 Hz, a low cut filter (60 Hz) was applied. Notch filters have successfully removed peak frequencies of between 117 and 132 Hz and between 340 and 400 Hz. Figure 3.4 shows the data before and after the second, additional notch filtering step described above. The result is slightly better; extremely large amplitudes and low frequencies have been eliminated, the data look less disturbed and there is even a reflection visible within the noisy area.
These above described, additional notch filtering steps are carried out and displayed by the

script *sc_plot_fi4* (Appendix F.10). But for the final version only the first step of notch filtering is applied to the whole dataset. All applied processing is summarized in the DSISoft script *sc_new_all* shown in Appendix F.6.

3.2.5 Removal of Downgoing P-Wave Energy

To remove the direct P-wave energy the same two step processing was applied as for borehole 4Q66W3: The traces were muted until 5 ms after the first breaks and tapered in the following 5 ms.
After notch filtering the second step was carried out: The traces were shifted by their travel times. Then the remaining, aligned P-wave energy is muted in the observed time window. Afterward the traces are shifted back. Figure 3.5 shows these processing steps done by *sc_plot_remR* (Appendix F.11).
From the first breaks an average P-wave velocity could be calculated. The value of 6020 m/s ±80 m/s (assuming the picks are off 0.5 wavelength) matches the values obtained from the rock property measurements.

3.2.6 Removal of Downgoing S-Wave Energy

The S-wave energy again was removed using median filtering. On the shifted data which align S-wave first breaks horizontally different median filter parameters were tested. A filter over 11 traces worked best. The filtered data were subtracted from the aligned data before the traces were shifted back. Find the procedure carried out by *sc_plot_remS* (Appendix F.12) shown on Figure 3.6.
First break arrival times display an average S-wave velocity of 3560 m/s ±200 m/s.

3.2.7 F-k filtering

Different f-k filters have been tested but didn't improve the data. So finally there was no f-k filter applied to this dataset.

3.2.8 Shift to TWT

The dataset was shifted to the two-way-travel time according to the first P-wave arrivals for summation into corridor stacks.

3.2.9 Effect of the Deviated Borehole

Since the approach of summing traces to create a corridor stack only works for horizontal reflections, the effect of a crooked borehole needed to be tested. The goal was to find out how many traces could be summed without offending the "horizontal" reflection rule. First both geometries were compared and plotted with the script *plot_geom* (Appendix F.13). Figure 3.7 shows the receiver coordinates of the real borehole in green crosses. The blue triangles depict

the wireline depth of the receivers vertically below the borehole location at surface, resulting in a straight borehole. The reflection points on an assumed horizontal plane in 700 m depth are marked as black dots. For each borehole two reflected rays (from source to reflection plane and to the topmost and the lowermost receivers) are shown as light blue or violet lines. Then the travel path difference between the real and the straight borehole were calculated using *VSP_DSI...* (Appendix F.14). This calculation presented in Figure 3.8 shows, that the effect on the direct rays results in a difference of up to 60 m. The calculation was done by the adapted script *txdirect1* shown in Appendix D.10. The zigzagging of the difference around trace 20, 50 and 100 is due to the change of the source position before the tool was raised by 5 m. This didn't have a large influence on the difference of the reflected raypath. The travel path difference of the reflected rays had been calculated using the script *txplane2* (Appendix D.11). Here the differences between the reflected paths to the straight and the real borehole are considerably larger up to almost 600 m (compare Figure 3.9).
Finally, it had to be calculated how many traces could be summed. Assuming a central frequency of 100 Hz and a velocity of 6000 m/s the associated wavelength is 60 m. Assuming further, that there should be no more than 90 degrees phase shift for a good summation the threshold is 15 m.
For each trace the gradient of the difference between the reflected, travelled distances was calculated. Then an extrapolation over a certain amount of traces was calculated using the calculated gradient. This extrapolated value was subtracted from the real value (difference of reflected distances). The difference had to be smaller than 15 m and the amount of traces should be as large as possible. The script *find_dist1* found in Appendix F.15 was written for this purpose. A final visual check was needed to make sure the limit was met. This calculation showed that summing over 40 traces lies within the threshold of 15 m. Figure 3.10 shows the difference of the extrapolated and the real difference extrapolated over 40 traces.

3.2.10 Corridor Stacks

The shifted data were summed to create different corridor stacks: First, the entire dataset was summed by the script *sc_corr_new_all* (Appendix F.7). Then nine windows were generated. The first three time windows contain data form very early times - the corridor stacks showing reflections which are near the borehole. The first window starts 50 ms after the first arrivals. The second window sums data from 100 ms after the first arrivals and the third window starts at 150 ms after first arrivals. All windows were 40 traces wide - but this distance was converted into depths using the first arrival velocity. Like this windows of 0.0648 ms width resulted.
The next three windows show reflections from the "mid-range" distance from the borehole. They start at 200, 250 and 300 ms after the first breaks. The reflections from further away were summed from three windows which start at 400, 450 and 500 ms after first breaks. The script to calculate these windows is called *sc_corr40...* with the affix near, mid, far and 1 to 3. They can be found in Appendix F.8. These nine files are merged into one by the script *sc_new_comp_corr2* (Appendix F.9). The resulting 45 traces and the data are shown in Figure 3.11. The lithologic log is shown for comparison. A larger picture of the lithologic log including its legend is shown in Appendix A.5. Prominent reflections are marked by green

arrows, the first break is shown in orange.

3.2.11 Synthetic Traces

From velocity and density logging done by J. Mwenifumbo acoustic impedance and reflectivity were calculated. The reflectivity was convolved with a Ricker wavelet of 100 Hz central frequency. Three different traces were calculated in script *sc_synTraces2* (Appendix F.16) for comparison: All velocity and density data points were used resulting in 0.25 m depth spacing. Data points were averaged over 4 samples resulting in 1 m spacing. To get a 5 m interval every fifth data sample of the 1 m spaced data series was used. But this spacing was too large and was not used further. The two synthetic traces were compared with the corridor stacks and the lithologic log. Figure 3.12 shows the lithologic log on the left side, followed by the synthetic traces with 1 m depth interval. Next are the synthetic traces with 0.25 m depth interval. Note that due to the increased sampling interval of 1 m the convolution with the 100 Hz Ricker wavelet produced a trace with a four times lower frequency content. In Figure 3.12 on the right are the nine corridor stacks obtained from different time windows. Note the depth scales of the lithologic log and the synthetic traces are linear even though the velocity would need to increase with depths to make up for the NMO, for the longer travelled distances.
Some reflections of the corridor stacks match the synthetics nicely, for example at 0.4 s or 880 m or at 1645 m depth. Other reflections seem to be much broader like the prominent one at 1230 m. Some reflections are not visible at all like those between 1160 and 1210 m. There are different possible explanations: 1) The effect of the borehole geometry is larger than assumed and 40 traces summation width is still to large. A summation over 90 degree phase shifted wavelets leads to a broader wavelet. 2) The dataset contains very noisy areas which are also summed but should theoretically be cancelled. 3) The NMO has not been corrected yet. Later arrival times would need to be shifted upward or the depth scale would need to be adjusted accordingly resulting in a non-linear scale. 4) The most possible explanation is that the lithologic boundaries are not at all horizonal.
Still the first three "near" stacks actually depict the subsurface closely beside the borehole.

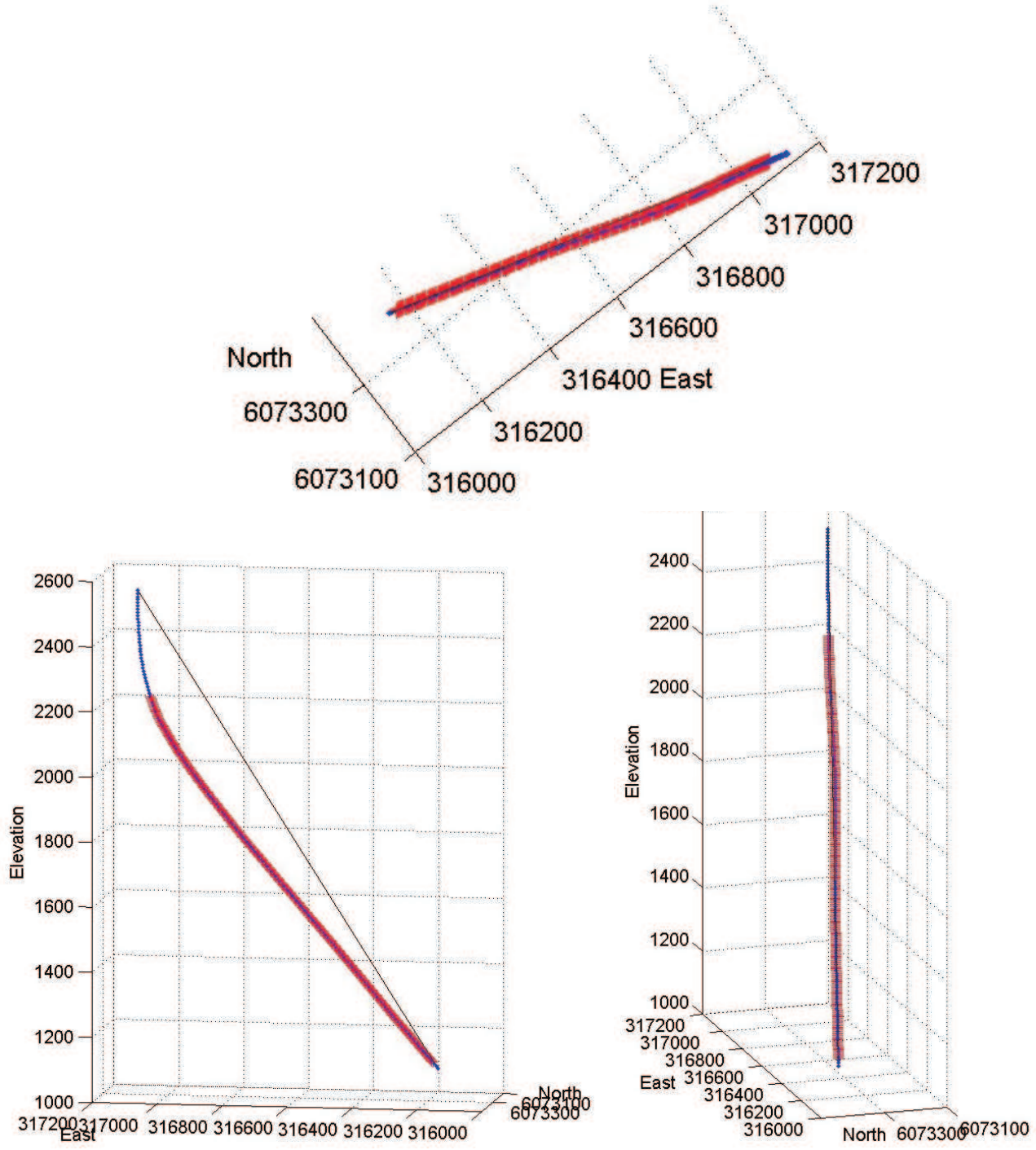

Figure 3.1: Drill coordinates of borehole FFM001 in blue dots. The red crosses mark the coordinates of each receiver position. The connection between the top and the bottom of the borehole is marked as a black line. Top: Birdeye view; left: orthogonal to the borehole plane; right "in line" with the borehole plane.

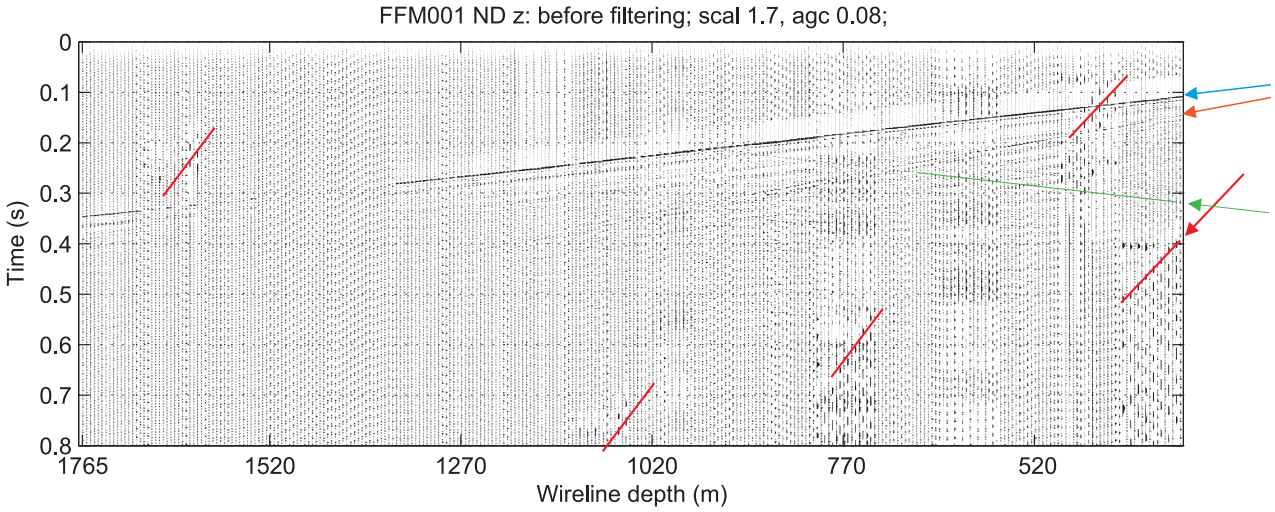

Figure 3.2: The raw but scaled data is shown with the P-wave marked in blue, the S-wave is orange and a prominent reflection is marked green. The tube waves are marked with red lines. In Appendix B.1 the enlarged Figure is shown.

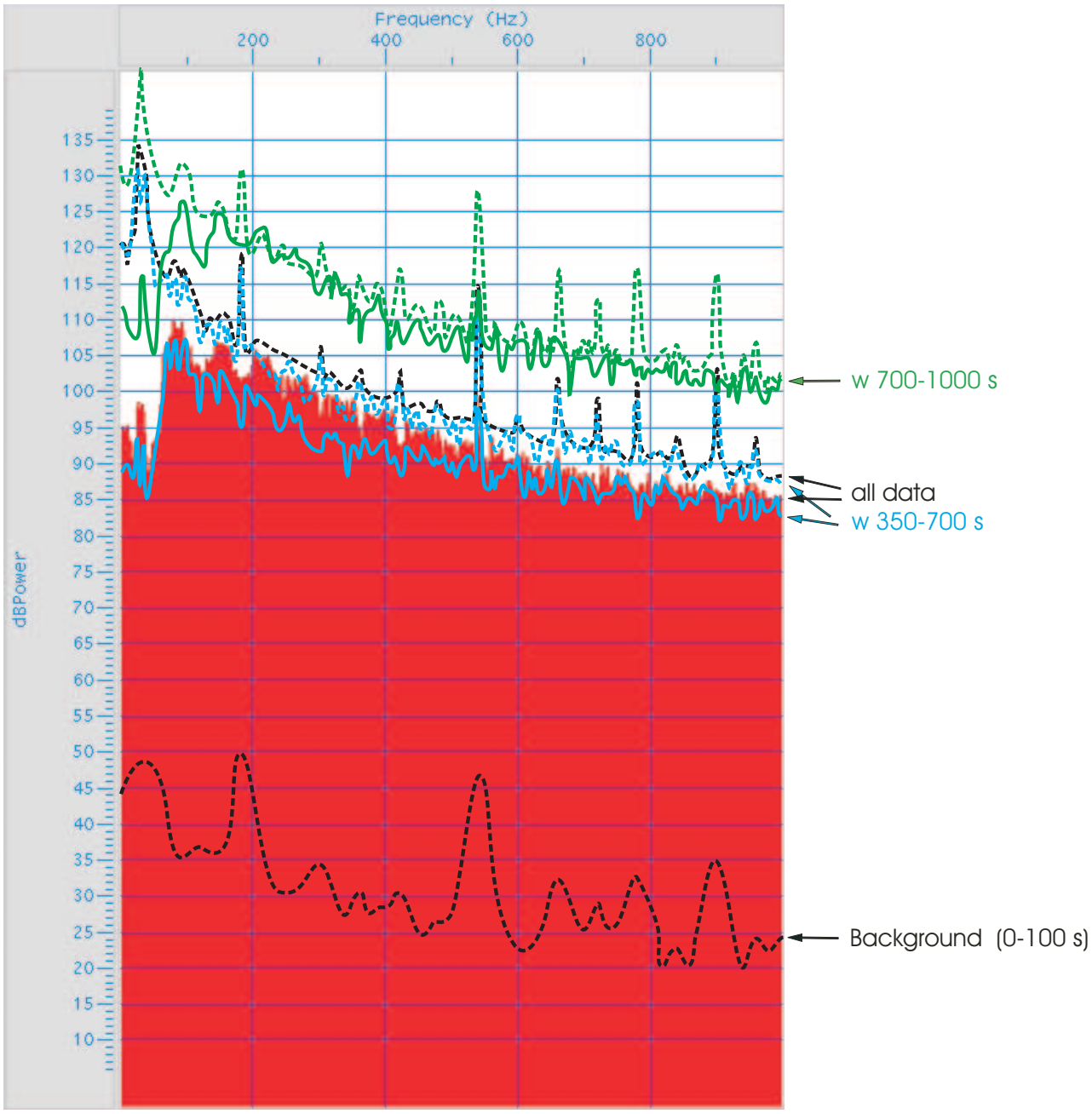

Figure 3.3: The frequency amplitude spectrum of the whole dataset after notch filtering is shown as solid red area. The amplitudes of different time windows of the notch-filtered data are marked with solid lines. The dashed lines show amplitudes of the raw data for different time windows.

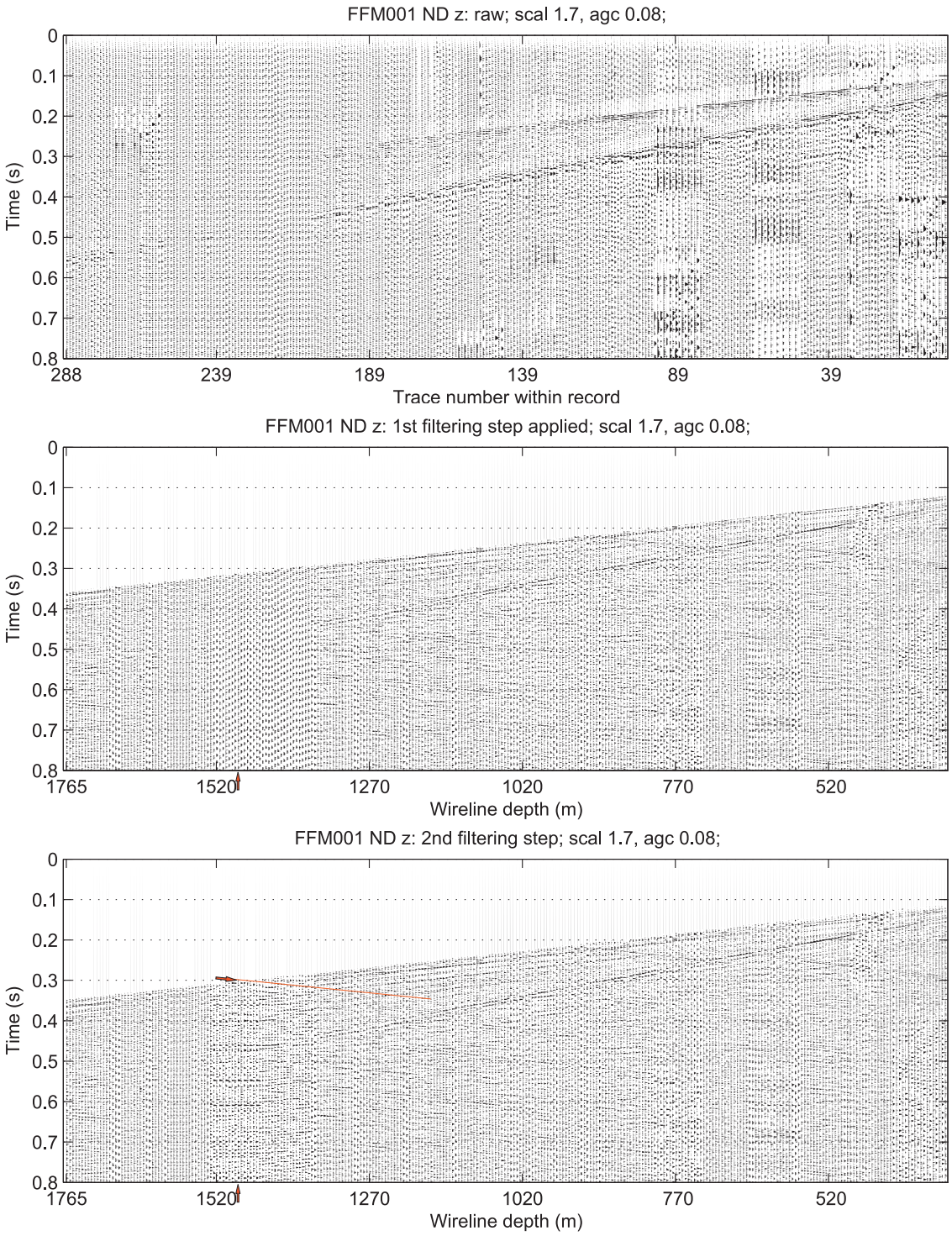

Figure 3.4: The raw, scaled data are shown on top. The data after the regular, first notch filtering are displayed in the middle panel. The same data are shown after the second, additional notch filtering step on the bottom panel. The arrows mark a reflection that was not visible in the middle panel before this special, second notch filtering step. In Appendix B.1 the enlarged panels can be found.

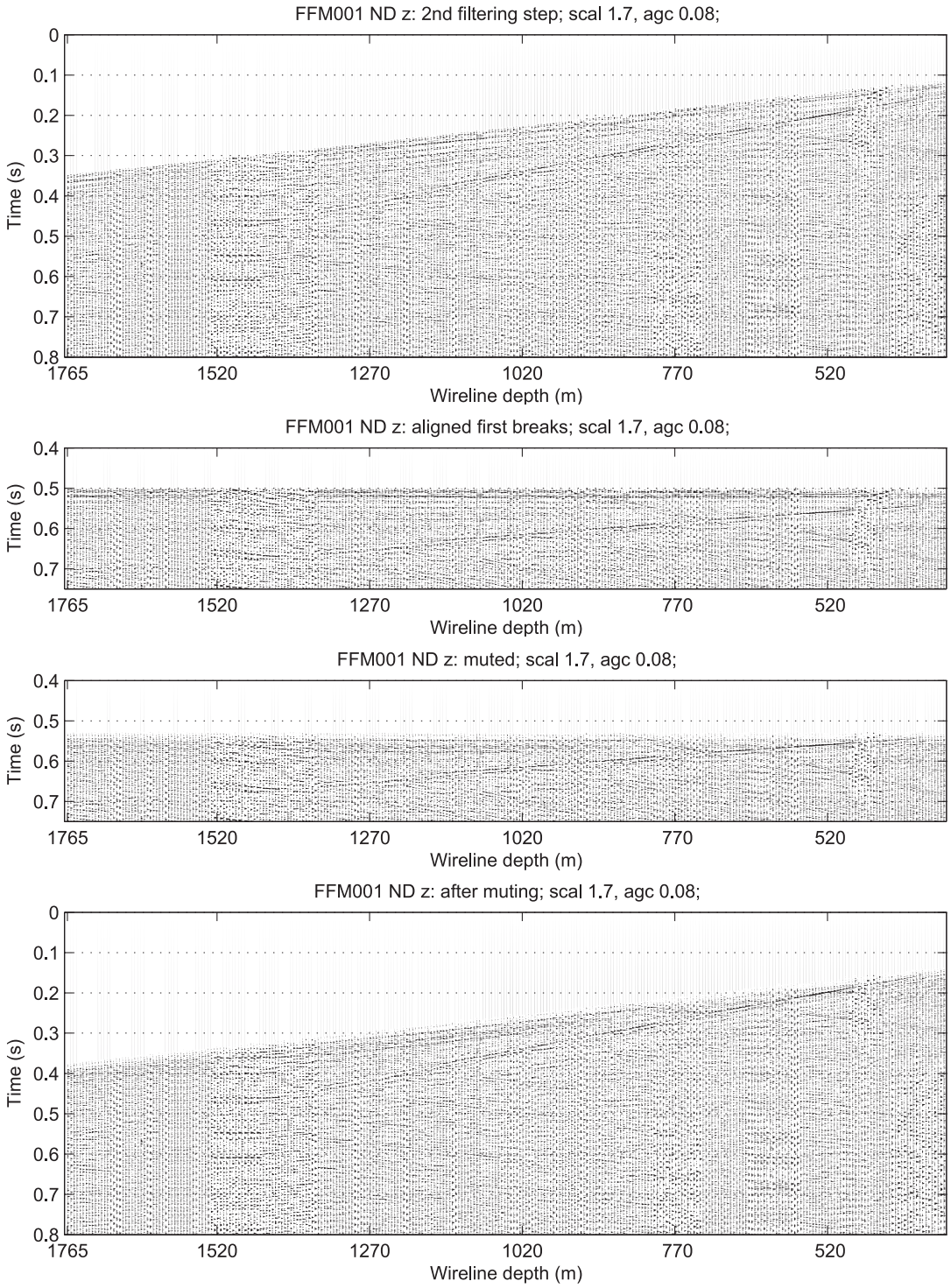

Figure 3.5: The second step of the P-wave energy removal is shown. Top: Data after first top mute and filtering; second from top: first break aligned data; muted data; bottom: after data is shifted back. The enlarged panels are shown in Appendix B.2.

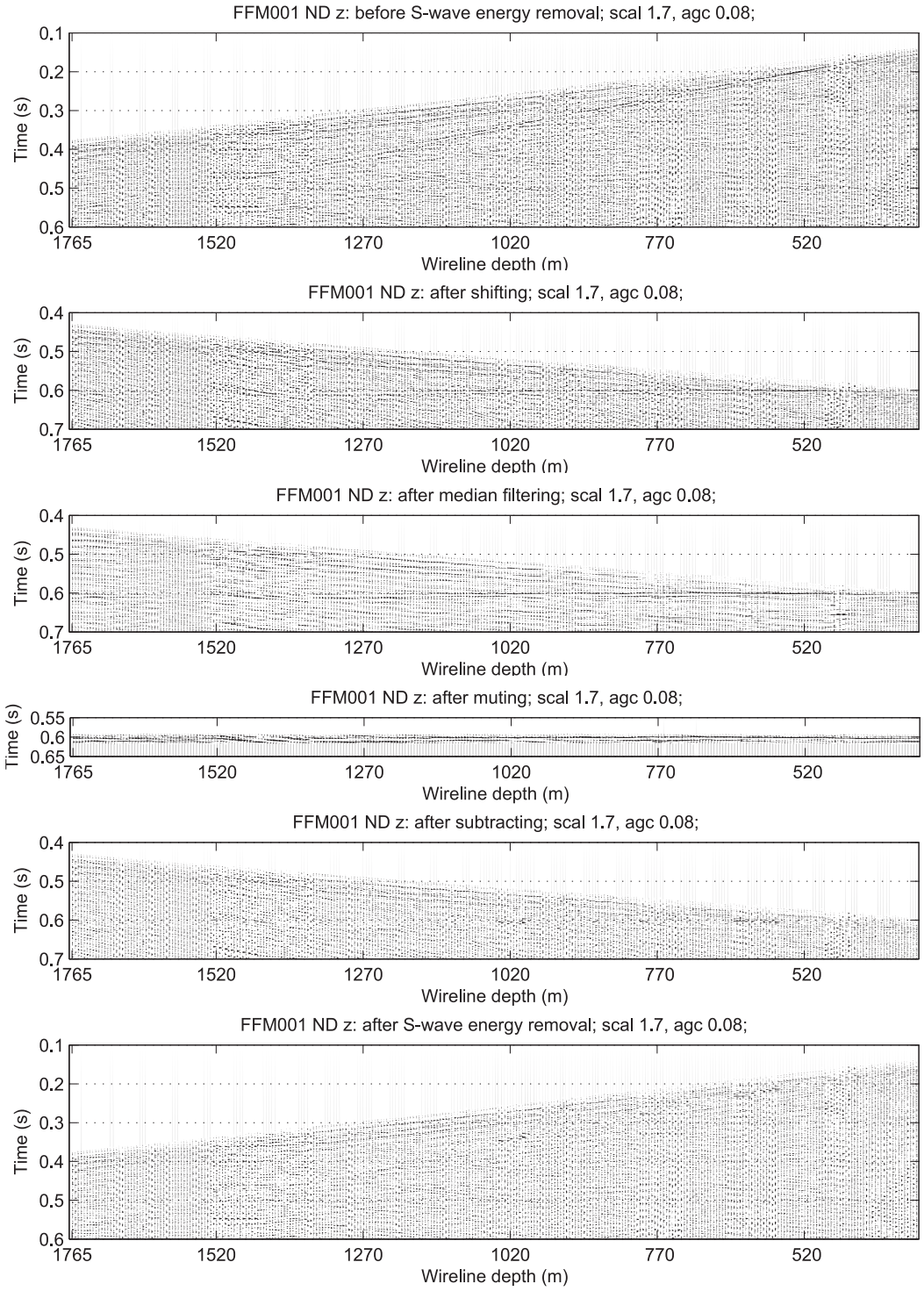

Figure 3.6: Removal of the S-wave energy. From top to bottom: The input data are shown; the data are shifted to align the first arrival times of the S-waves; median filtered data; except a narrow window all filtered data are muted; the small window is subtracted from the aligned data; the data are shifted back. In Appendix B.3 the enlarged panels can be found.

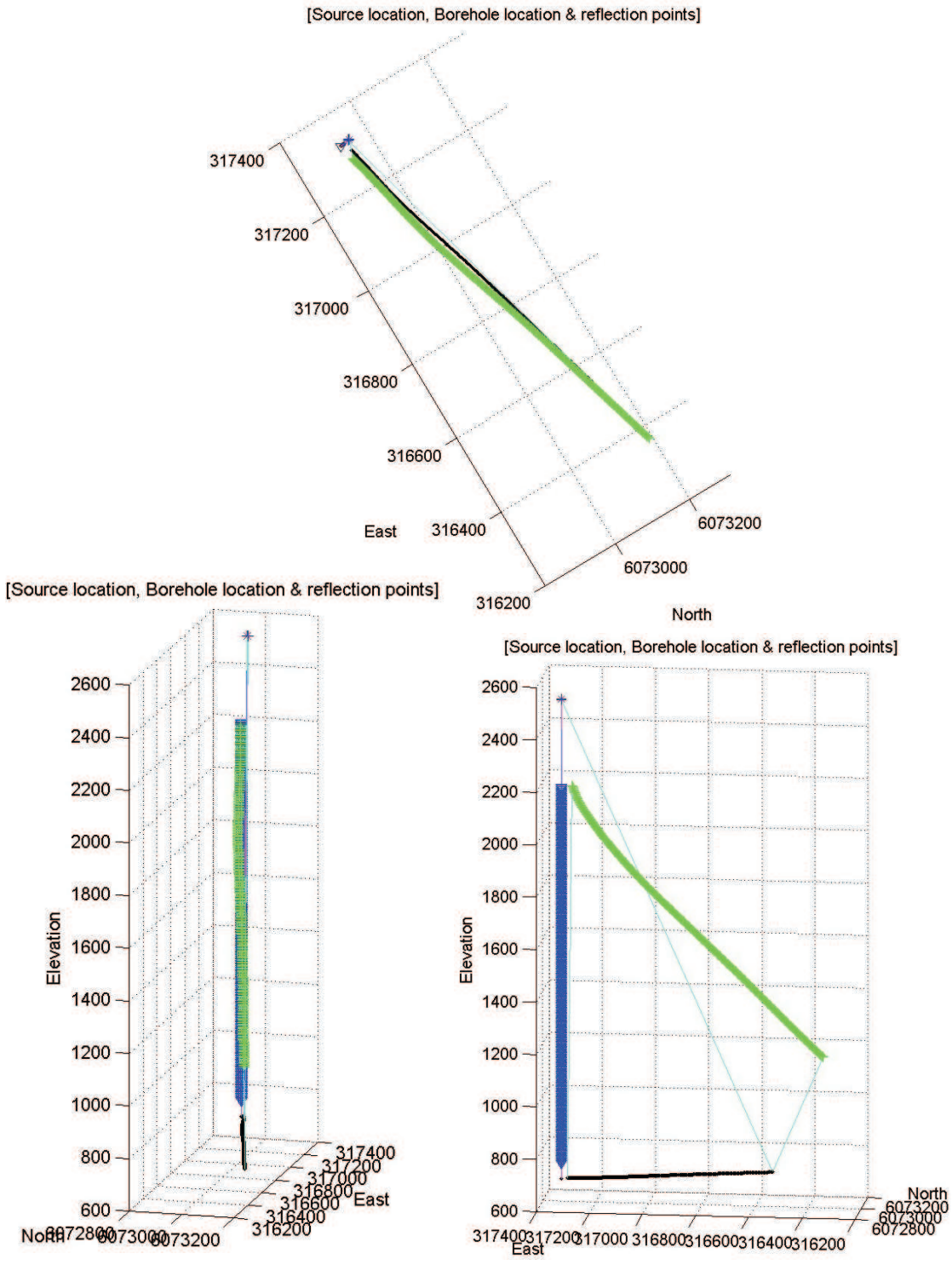

Figure 3.7: Geometry of the real (green) and the straight (blue) borehole are shown. Top: birdeye view; left: "in line" with the borehole plane; right: "orthogonal" to the borehole plane. Reflection points on an arbitrary horizontal plane in 700 m depth are marked in black. The top and lowermost rays are shown in light blue and violet.

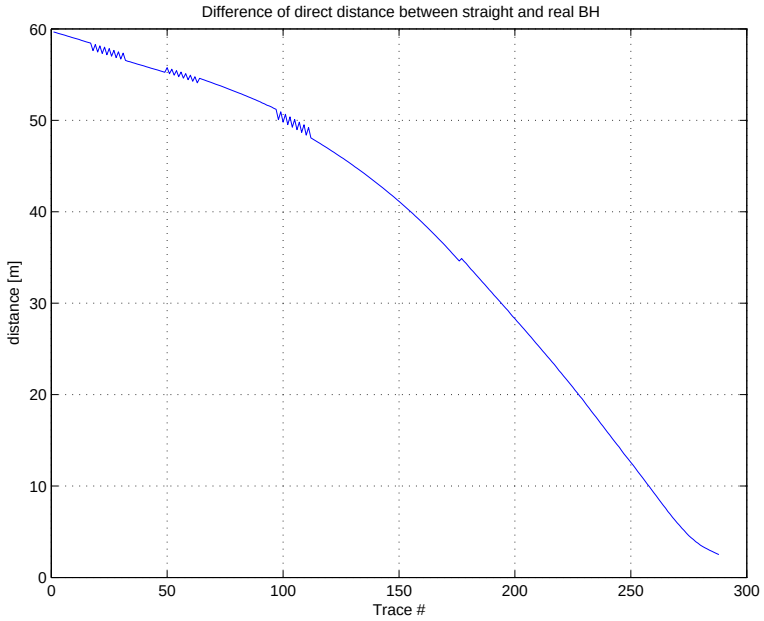

Figure 3.8: The travelled distance from source directly to the receiver (direct raypath) is calculated for the real and the straight borehole coordinates. The difference is plotted for each trace.

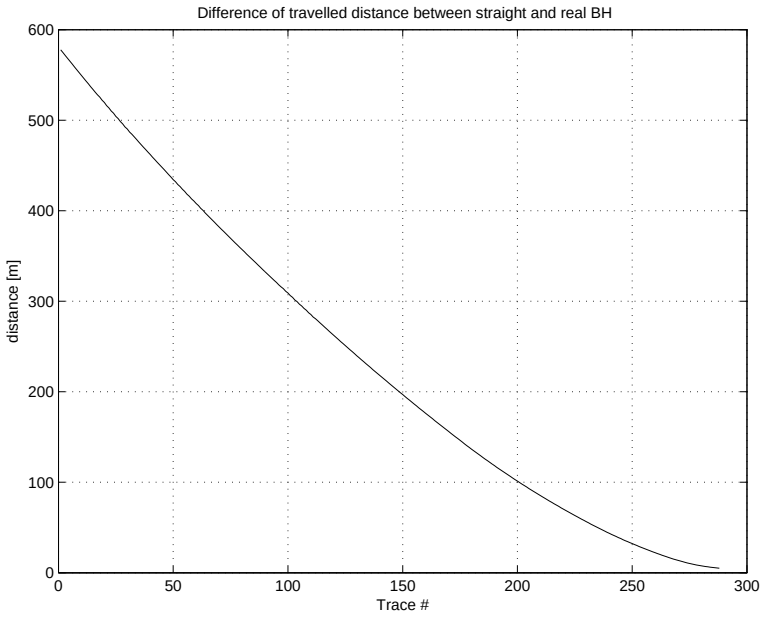

Figure 3.9: For a reflected ray the travelled distance is calculated for the real and the straight borehole coordinates. The difference is plotted for each trace.

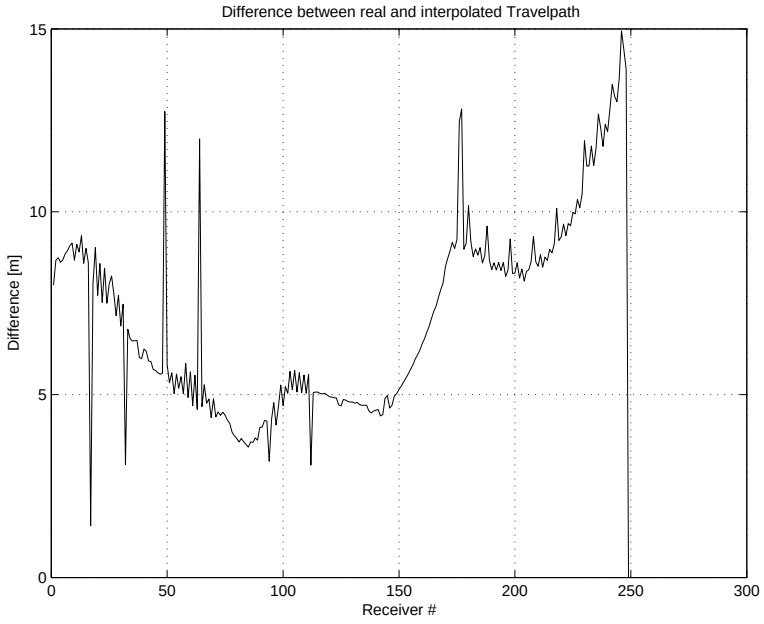

Figure 3.10: Difference of the extrapolated value (gradient of the difference over 40 traces) and the real difference.

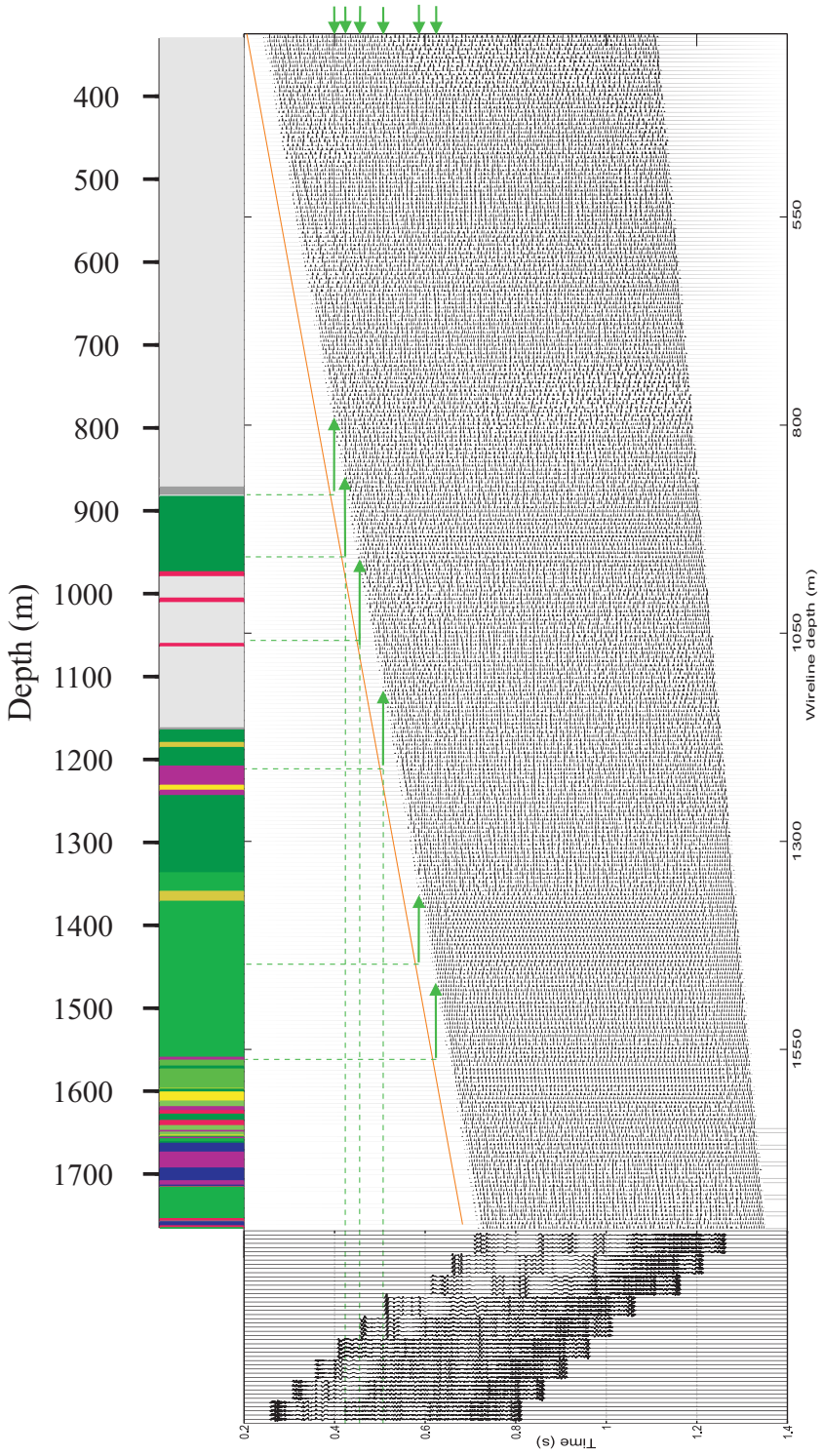

Figure 3.11: TWT shifted data and the corresponding summation traces are shown. The lithologic information is displayed on top matching the wireline depth (find the enlarged log and legend in Appendix A.5). The first arrivals are marked as an orange line. They are the link between reflections and lithology.

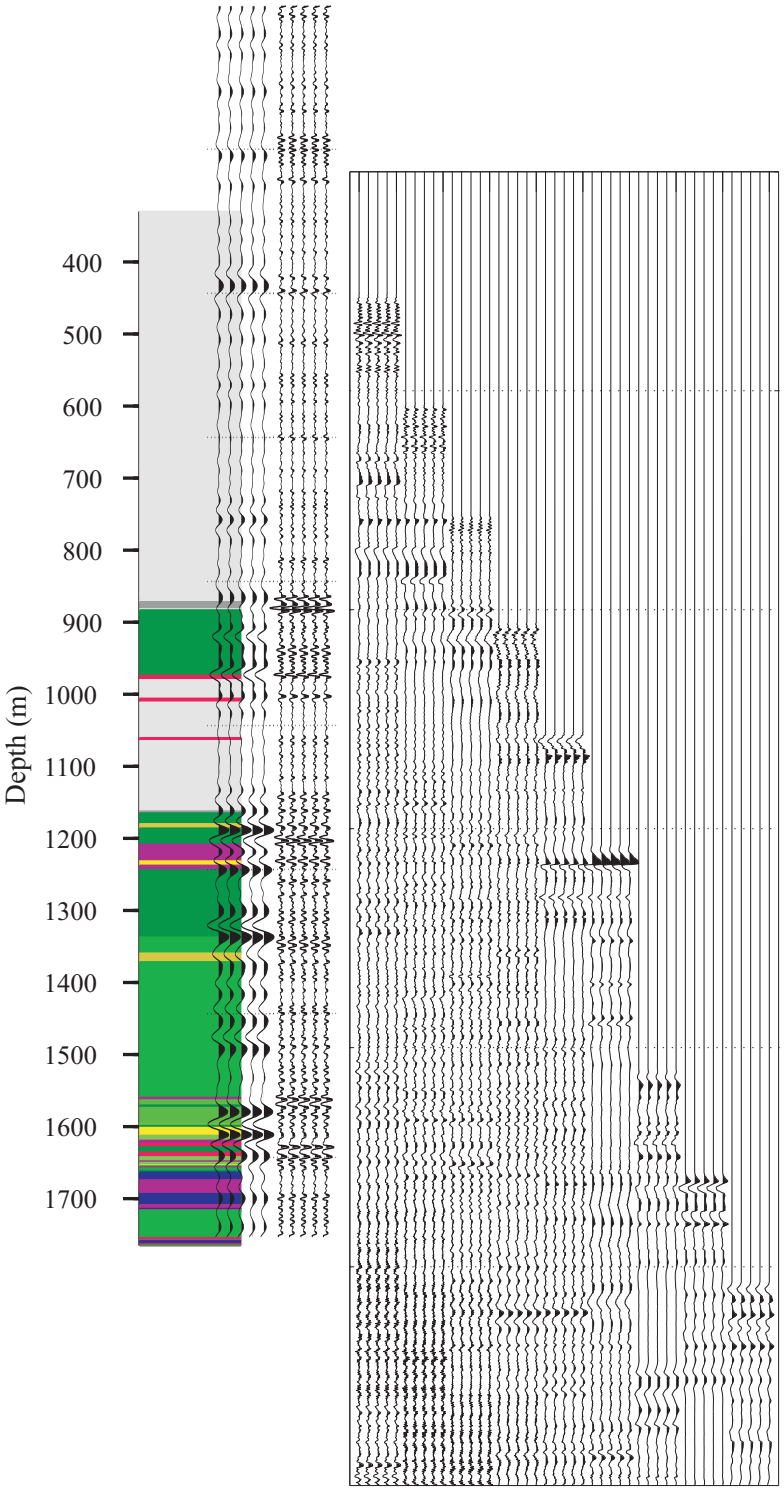

Figure 3.12: From left to right are: lithologic log (find the enlarged log and legend in Appendix A.5), synthetic seismic responses (for 1 m and 0.25 m sampling, respectively) and the corridor stacks from nine different windows.

Chapter 4

Borehole FFS039

Borehole FFS039 is the westernmost of the three boreholes. It is located west of Ross Lake. It reaches a depth of 1110 m below surface, whereas seismic data were recorded from 230 to 1105 m below surface, covering an overall distance of 875 m. The dataset consists of 704 traces; 176 vertical, 176 horizontal (H1), 176 horizontal (H2) and 176 surface pilot traces, but only the vertical traces of the near offset sources were processed.

4.1 Processing Flow

Table 4.1 gives an overview of the relevant processing steps for the vertical component data for borehole FFS039. Basically, the processing is the same as for 4Q66W3. These are major differences: 1) The electrical noise was removed prior to the P-wave energy removal. 2) The P-wave energy was removed by a median filter. 3) There was a horizontal event that was removed as well. 4) No f-k filter was applied and the bandpass limits were 71 to 250 Hz (compared to 71, 150 Hz for 4Q66W3 and 100, 200 Hz for FFM001)

4.2 Processing Steps for Borehole FFS039

The following sections describe the processing steps in more detail, and the corresponding processing scripts are found in Appendix G.

4.2.1 Data Conversion

The script *sc_segy2mat* (Appendix G.1) was used to convert the SEGY dataset into DSISoft format. The trace headers were not correct and needed to be changed.

4.2.2 Geometry: Correction of Receiver and Shot Coordinates

The drill log `gems_drillhole_umt_trace.txt` was used to correct the receiver coordinates. A summary for the FFS039 specific data is in `Drill1.txt`. While drilling the coordinates

Table 4.1: The applied processing steps.

Data Conversion
The dataset is converted from SEGY to the DSISoft format.
Geometry
File and trace headers are written into the dataset. Headers are corrected. Both horizontal components are removed from the dataset.
Trace Scaling
Picking of First Arrivals (P- and S-waves)
First arrival times of P- and S-waves are picked and transferred to the trace headers.
Electrical Noise Removal
For each frequency peak a notch filter is applied.
Direct P-wave Energy Removal
P-wave energy is removed with a 15 traces wide median filter.
Direct S-wave Energy Removal
The S-wave energy is removed using an
BP Filtering

were taken every 50 m along the wireline depth. To get 5 m intervals for the receivers a 3D interpolation was done. Calculations can be found in `calc3.xls`. Finally the interpolated coordinates were written to `Receivers4.txt`. The script *sc_write_th* (Appendix G.2) writes the corrected receivers into the trace headers.
The source locations were corrected using the script *sc_more_th3* (Appendix G.3). First, the source locations are read from `FFS039.xls` (from University of Alberta) into `shot_depths.xls` where they were changed into the Mine UTM coordinate system. The output was `Sources.txt` which was used by the script *sc_more_th3*. To inspect the coordinates once more Figure 4.1 was produced by the script *plot_BH_FS39_cor5* found in Appendix G.4.
Both horizontal components were removed using the script *sc_subset3* found in Appendix G.5.

4.2.3 Trace Editing and Scaling

Noisy traces were observed with extremely high amplitudes. They were marked for later muting. The amplitude decay due to geometrical spreading and attenuation was corrected using a power formula. Each sample was multiplied by $t^{1.7}$. The DSISoft module written for this purpose is *time_scale* shown in Appendix D.1.
This processing step and all the following ones were tested individually, but for the final version they were combined in one script called *sc_all26*. This script can be found in Appendix G.6.

4.2.4 Notch Filtering

Of all three processed boreholes this dataset suffered the most electrical and cultural noise. To investigate the frequency spectrum the whole dataset was converted into SEGY and analyzed with the Promax Software.
The unhealthy influence of the very high amplitudes on the noisy traces made it necessary to mute these traces first. Figure 4.2 shows the raw, but scaled amplitude spectrum on the right and the traces on the left. Traces 63, 120, 122, 124, 126 and 128 were muted, then the data were Fourier transformed. Figure 4.3 shows the amplitude spectrum of the muted, scaled data and the notch filtered data. The frequencies 60, 180 and 540 Hz show the highest amplitudes. For each of these amplitude peaks and for 120, 300, 380, 420, 660 and 780 a notch filter was applied. A new function *cos_notch_newtest* (Appendix D.12) was written, which applies all notch filters in one step, so that the Fourier transformation needs to be done only once.
In Figure 4.3 the background noise is marked as a black dashed line. Note that for the most time windows the amplitude peaks were successfully removed. Figure 4.4 shows the data before and after notch filtering.

4.2.5 Median Filtering of Downgoing Energy

Since we are only interested in the upgoing reflection energy, downgoing P- and S-wave energy had to be removed.

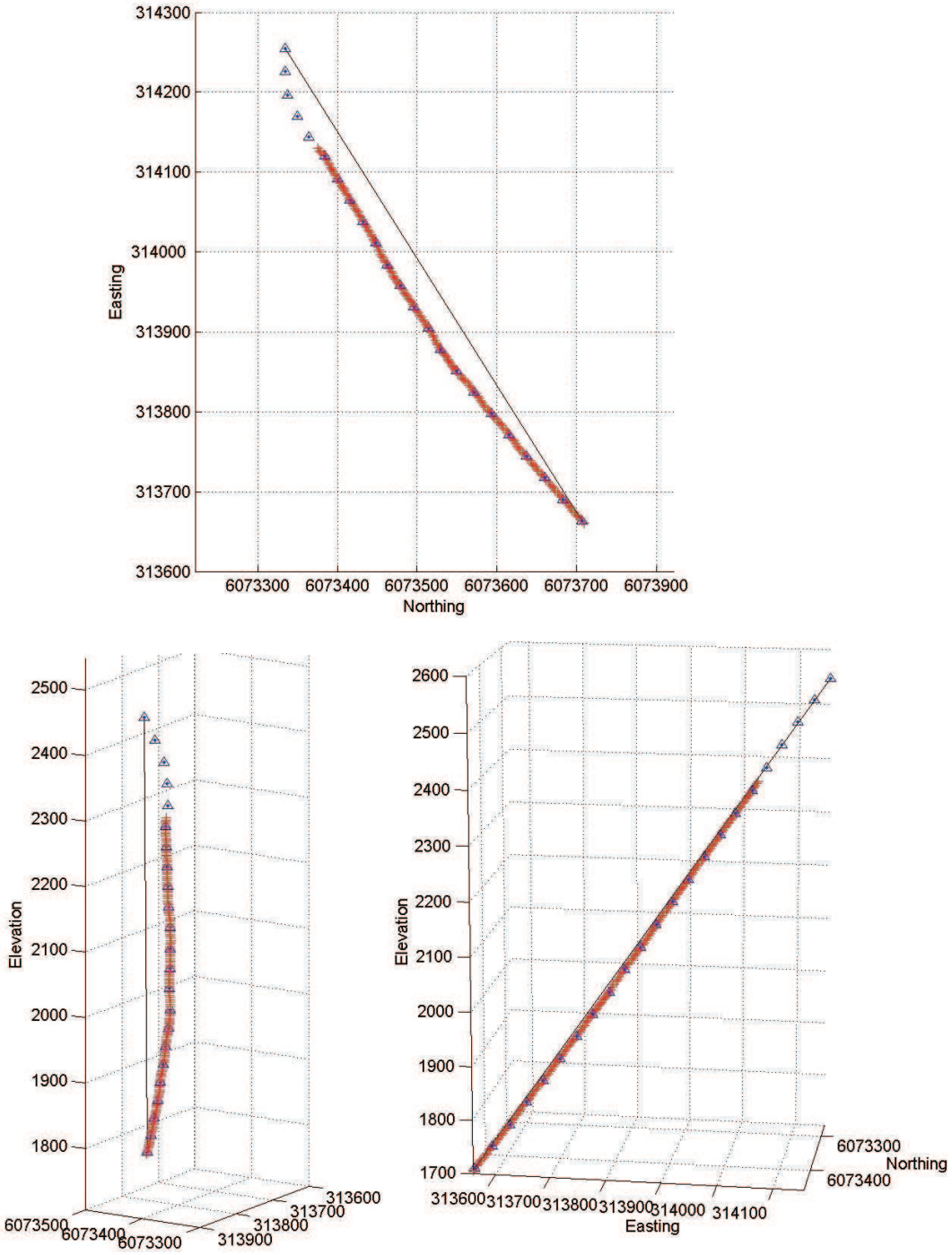

Figure 4.1: The surveyed coordinates of the borehole FFS039 are shown as blue triangles. The interpolated receiver coordinates are marked with red crosses. Top: Birdeye view; left: "in line" with the borehole plane; right: orthogonal to the borehole plane.

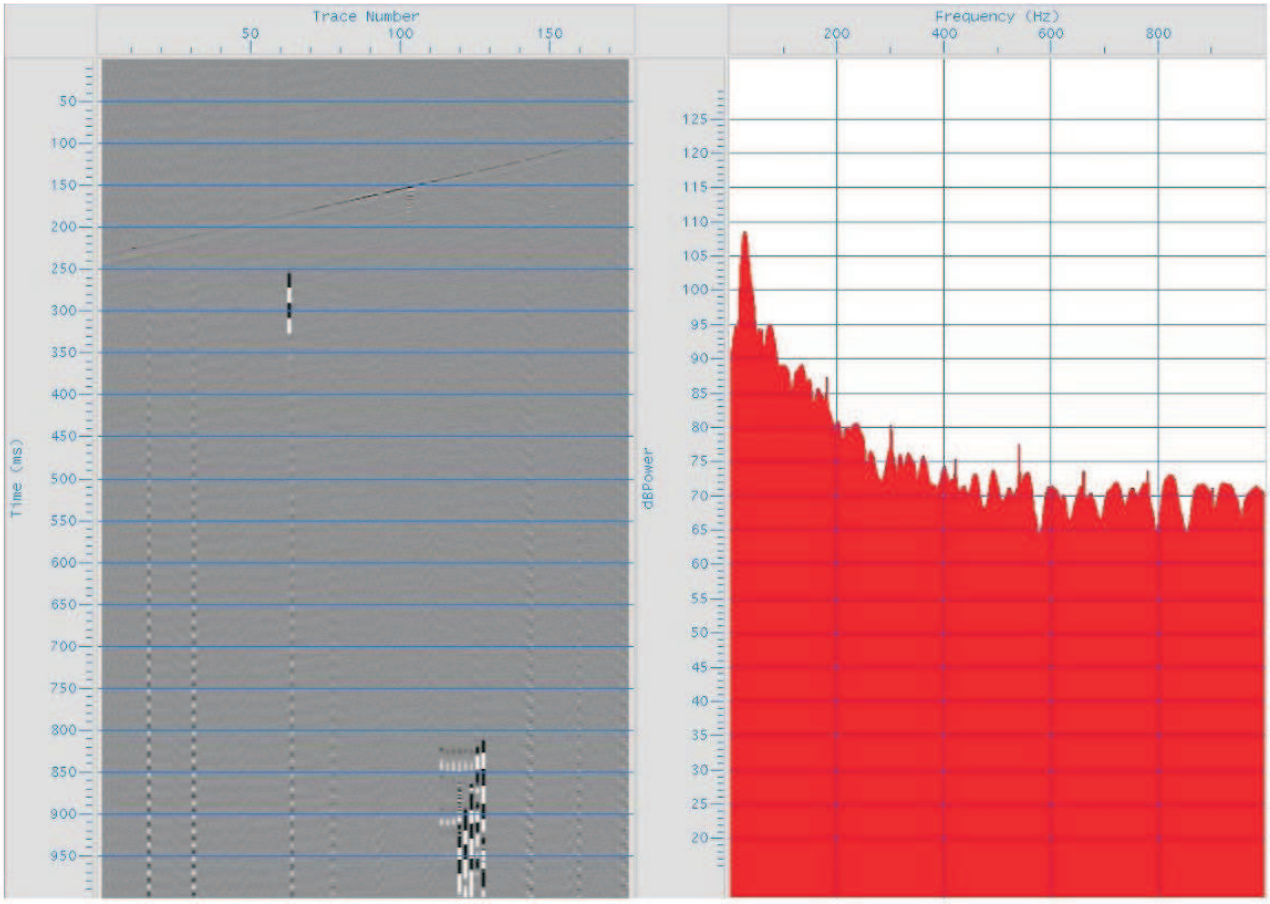

Figure 4.2: Very high amplitudes are visible on different traces on the left. The amplitude spectrum is dominated by these traces.

P-Wave Energy

The P-wave energy was removed using a median filter. Therefore the first breaks of the P-wave had to be aligned horizontally. This was done by shifting each trace upward by the picked P-arrival time. In order not to loose any data by the shifting process, 0.6 s were added at the beginning of each trace. This was done in the script *sc_all26* (Appendix G.6 in the P-mute section). The median filter was 15 traces wide. Finally each trace was shifted back and the added 0.6 s were removed. Figure 4.5 shows the data after each processing step.
From the first breaks an average P-wave velocity could be calculated. The value of 6090 m/s $\pm$120 m/s (assuming the picks are off 0.5 wavelength) matches the values obtained from the rock property measurements.

S-Wave Energy

The S-wave is hardly visible on the data acquired with the vertically oriented receivers. Its visibility is much better on H2 where the first S-arrivals were picked. First break arrival times

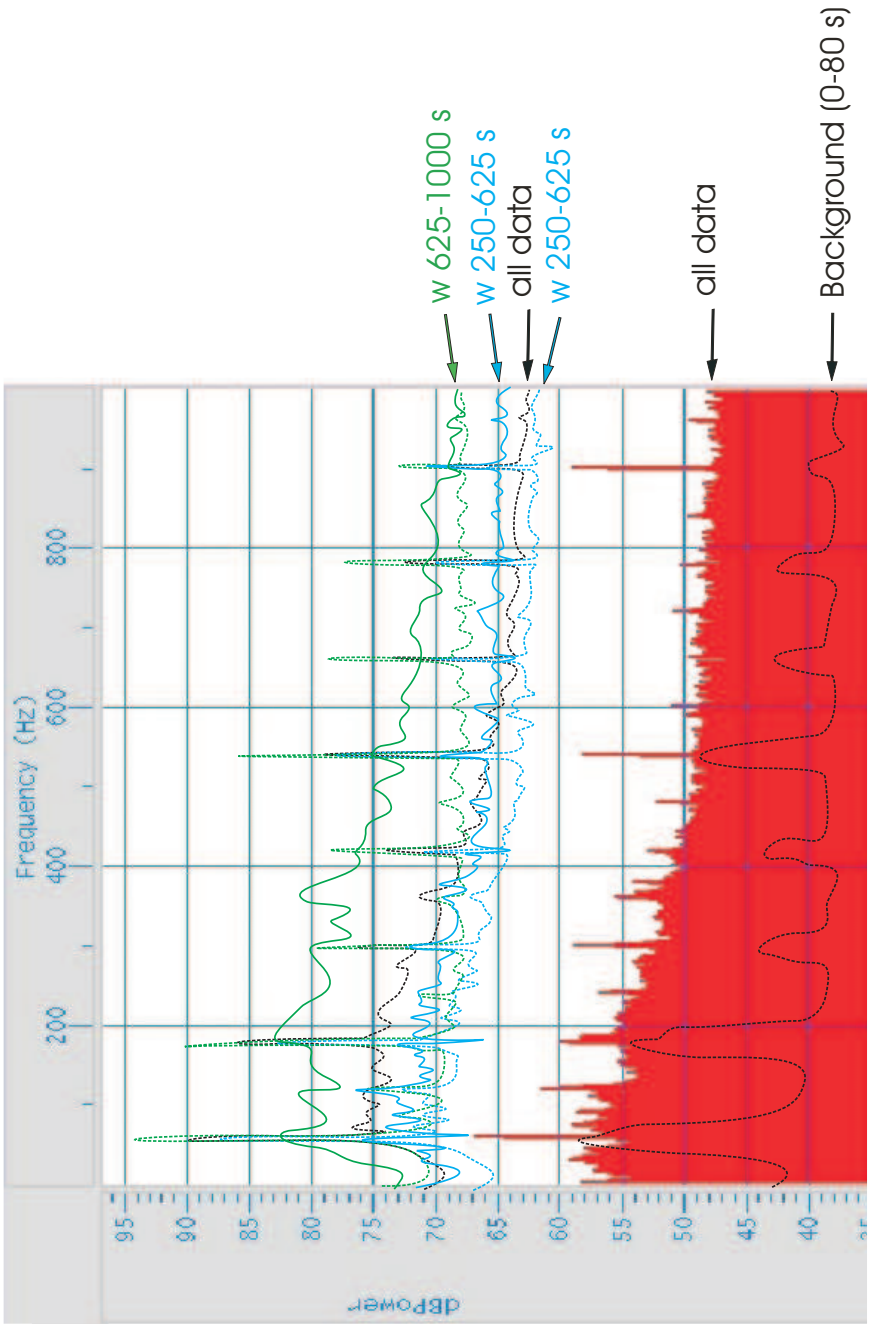

Figure 4.3: Amplitude spectra for different datasets and time windows are shown. All data with the notch filters applied is shown in red. All spectra of the muted and scaled data is marked in dashed lines. The solid lines show the spectra of the different notch filtered time windows. Blue is the time window 250 and 625 ms. The time window of 0.625 to 1 s is shown in green.

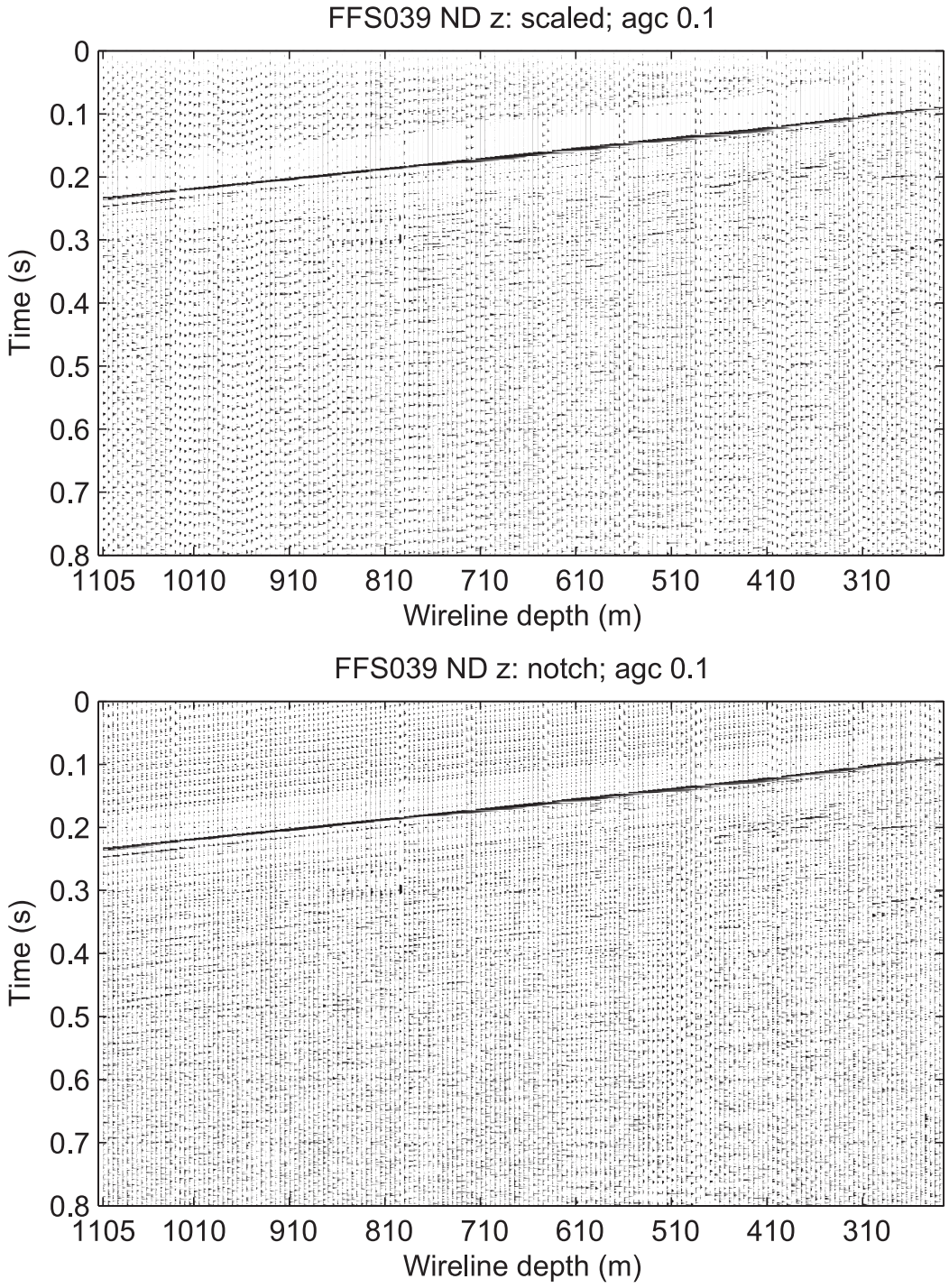

Figure 4.4: The raw, but scaled data are shown on top. Below is the scaled, notch filtered dataset. In Appendix C.1 the enlarged panels are shown.

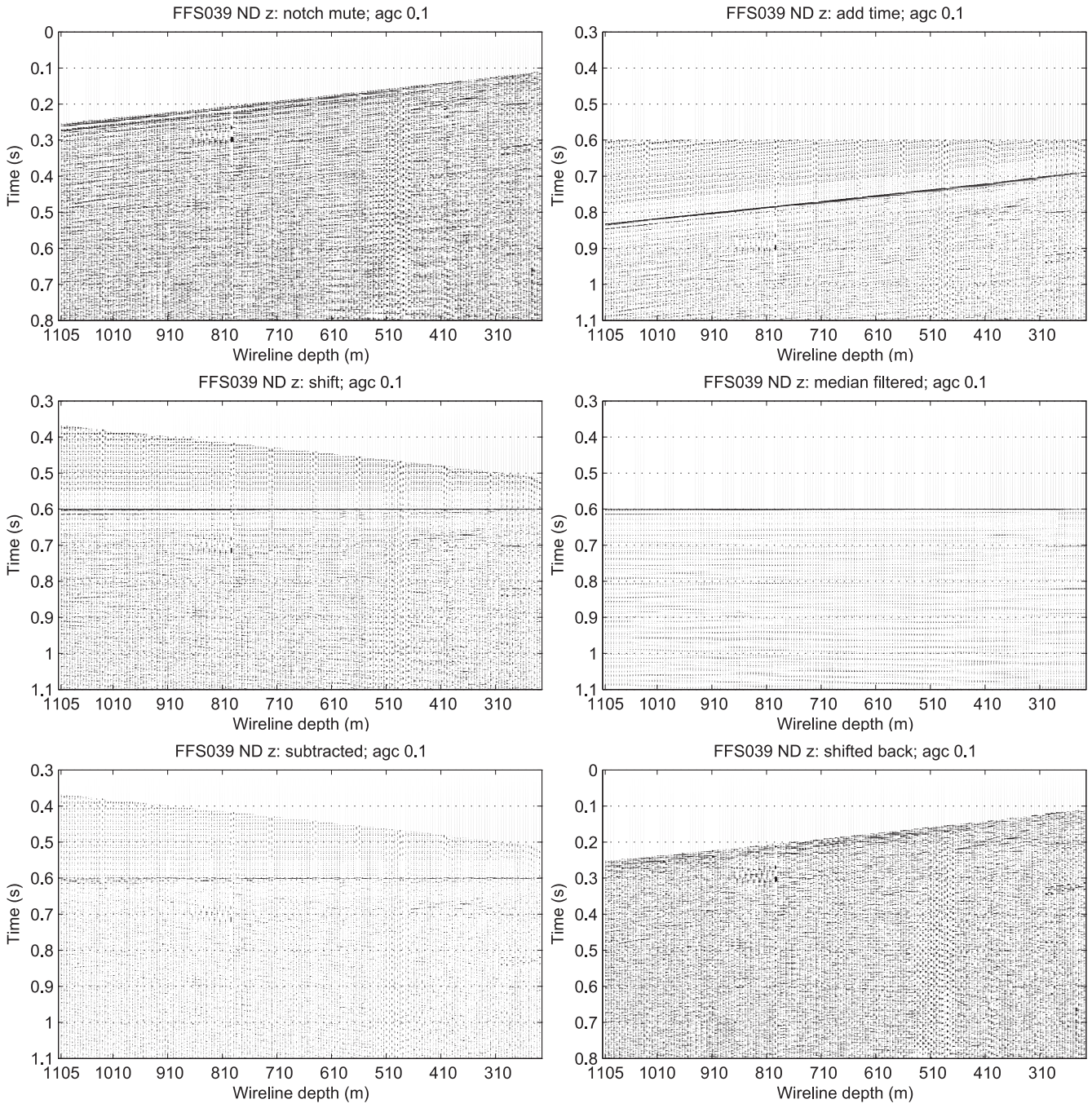

Figure 4.5: Removal of the downgoing P-wave energy. From top left to bottom right: notch filtered data; 0.6 s time added at each trace beginning; shifted by picked P-arrival times; median filtered data; data after subtraction of the median filtered data; shifted back. In Appendix C.2 the enlarge panels can be found.

display an average S-wave velocity of 3340 m/s ±200 m/s.
The following steps were applied to remove the S-wave energy: Add 0.6 s at each trace start; shift upward by the first arrival times, apply a 15 traces wide median filter; subtract the filtered from the shifted data; shift back and remove the added time. All these steps are shown in Figure 4.6.

H-Wave Energy
There are several horizontal events visible in the S-removed data at traces 800 to 750 for example. Therefore the data were median filtered (15 traces wide) before these filtered data were subtracted from the S-removed data. Figure 4.7 shows the data before and after the removal of the horizontal events.
All the above processing steps are summarized in the script *sc_all26* shown in Appendix G.6. All Figures shown were produce by the script *sc_plot_all* (Appendix G.10).

4.2.6 Bandpass and F-K Filtering

To further improve the dataset bandpass filters with different corner frequencies were tested on the dataset where all downgoing energy was removed. The script *sc_bp_Test2* shown in Appendix G.11 displays the data with different filters. These plots are shown in Figure 4.8. Finally, corner frequencies of 50, 71, 250, 400 Hz which seemed to provide the clearest image were applied to the data.
Different f-k filters were tested but didn't improve the data. So finally there was no f-k filter applied to this dataset.

4.2.7 Shift to TWT

The dataset was shifted to the two-way-travel time according to the first P-wave arrivals by the script *sc_twt_new* (Appendix G.7). Thus the reflections were aligned horizontally and ready for summation into corridor stacks. Figure 4.9 shows the whole dataset plus its corridor stack on the left.

4.2.8 Effect of the Deviated Borehole

Before the corridor stacks with the appropriate window sizes can be performed the effect of the deviated borehole had to be tested. Thus the best suited window size had to be found. In a first step both geometries were compared and plotted with script *plot_geom1* (Appendix G.15). Figure 4.10 shows the straight borehole with wireline depth vertically down as blue triangles whereas the real receiver coordinates are displayed as green crosses. All reflection points on an arbitrary plane in 700 m depth are marked with black dots. For each borehole the raypath to the highest and deepest receiver is shown in violet or light blue.

The difference of the directly travelled distance between the straight and the real borehole is not large, only about 5 m (Figure 4.11). In contrary the travelled distance of the reflected path (reflected at the arbitrary horizontal plane in 700 m depth) shows much larger differences

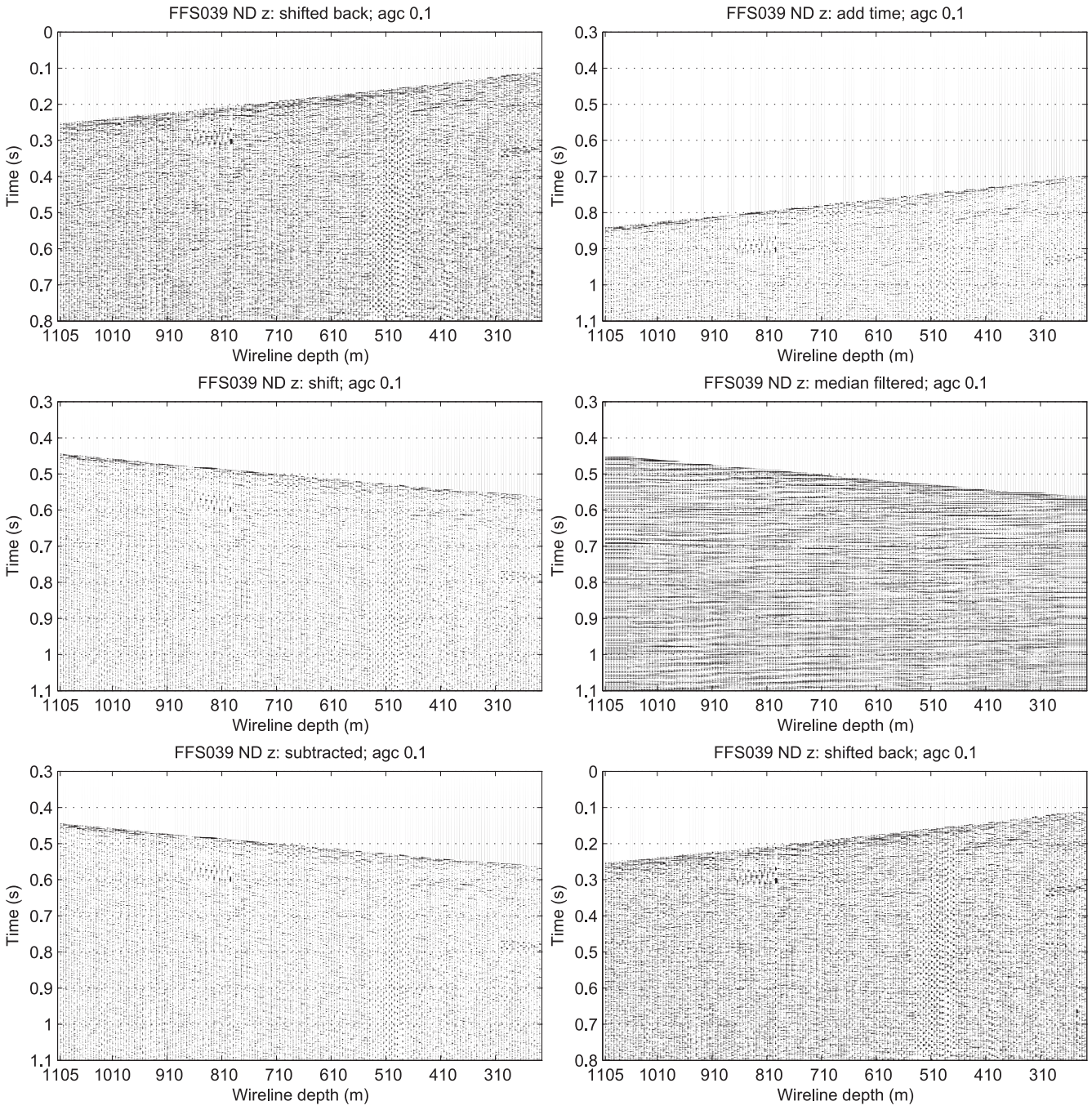

Figure 4.6: Removal of the downgoing S-wave energy. From top left to bottom right: P-removed data; 0.6 s time added at each trace beginning; shifted by picked S-arrival times; median filtered data; data after subtraction of the median filtered data; shifted back. In Appendix C.3 the enlarged panels are shown.

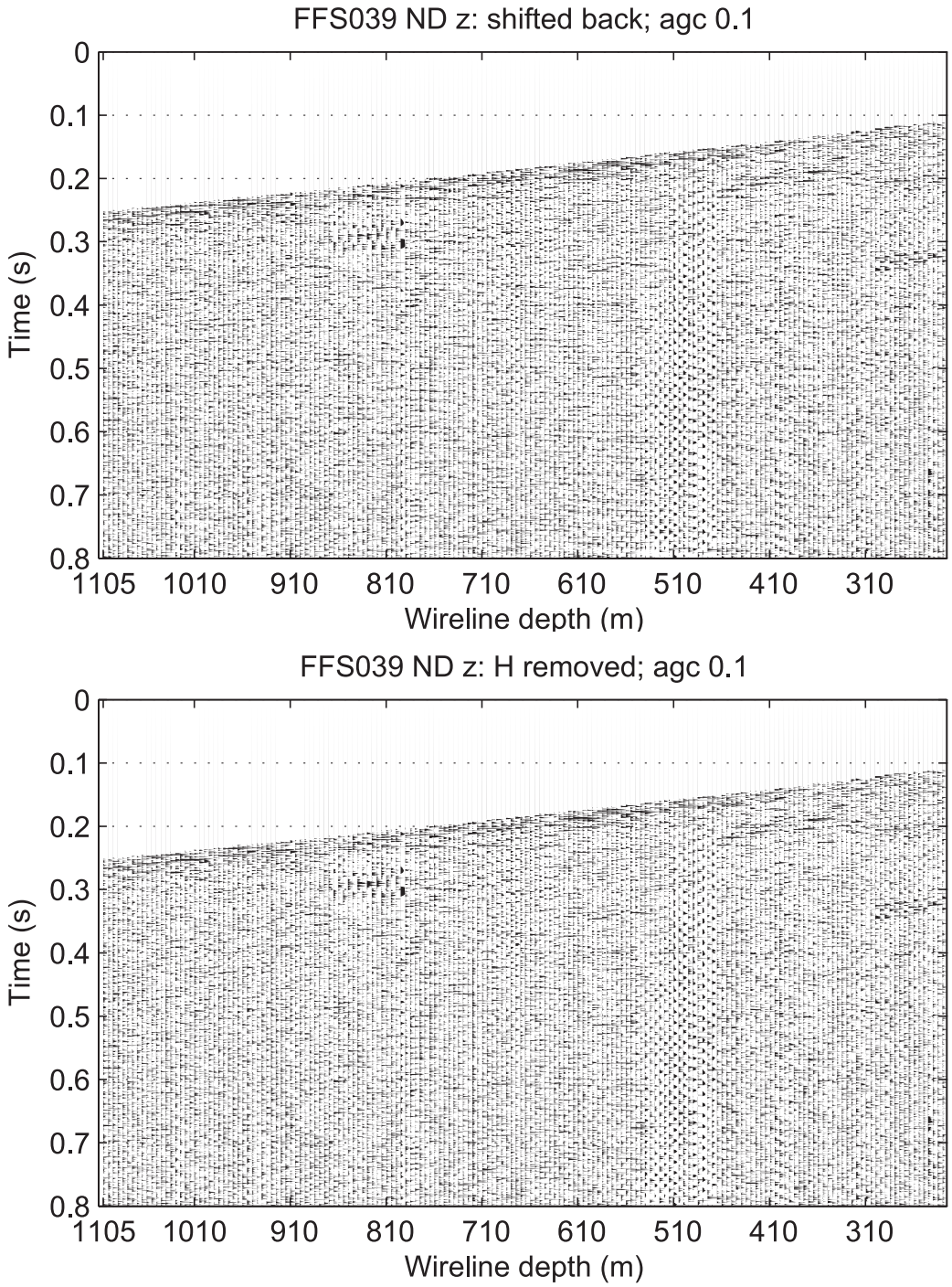

Figure 4.7: Removal of horizontal events. Before is shown on top, whereas the H-removed data is shown at the bottom. In Appendix C.4 the enlarged, lower panel can be found.

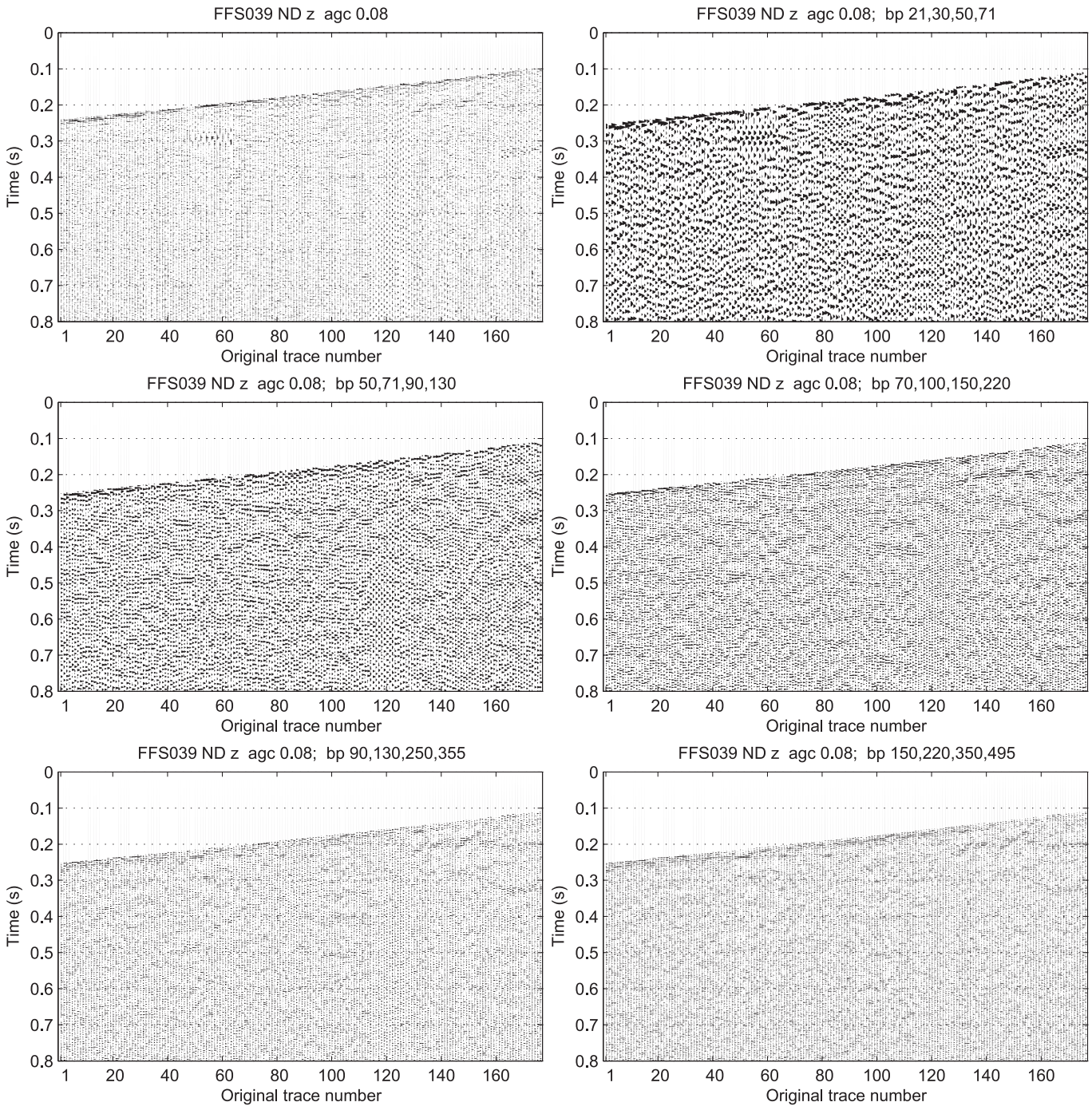

Figure 4.8: The scaled, filtered and downgoing energy removed data is shown with five different bandpass filters applied. The corner frequencies of the filters are shown in the Figure captions. In Appendix C.5 the enlarged panels can be found.

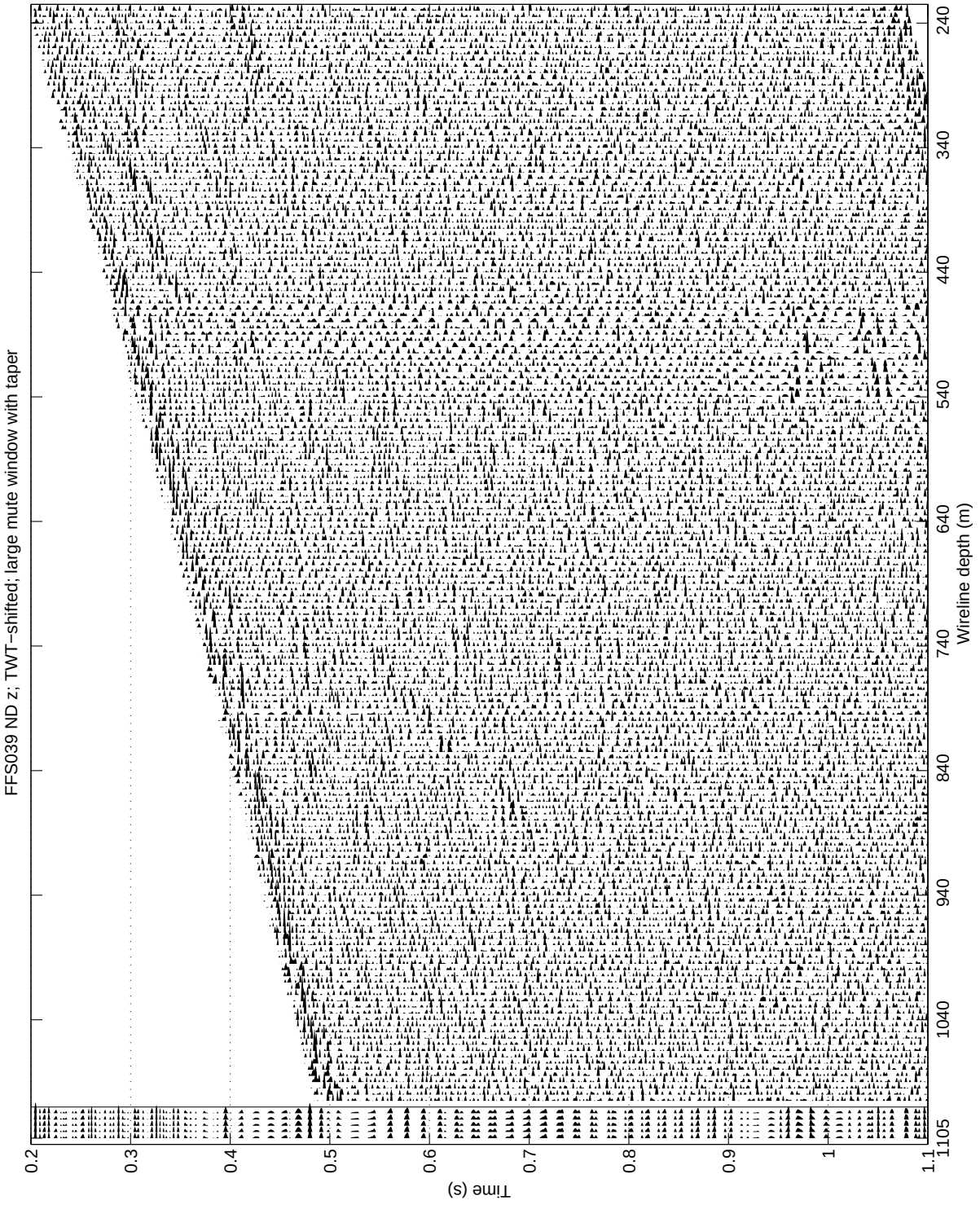

Figure 4.9: TWT shifted data and the corridor stack of the whole dataset on the left.

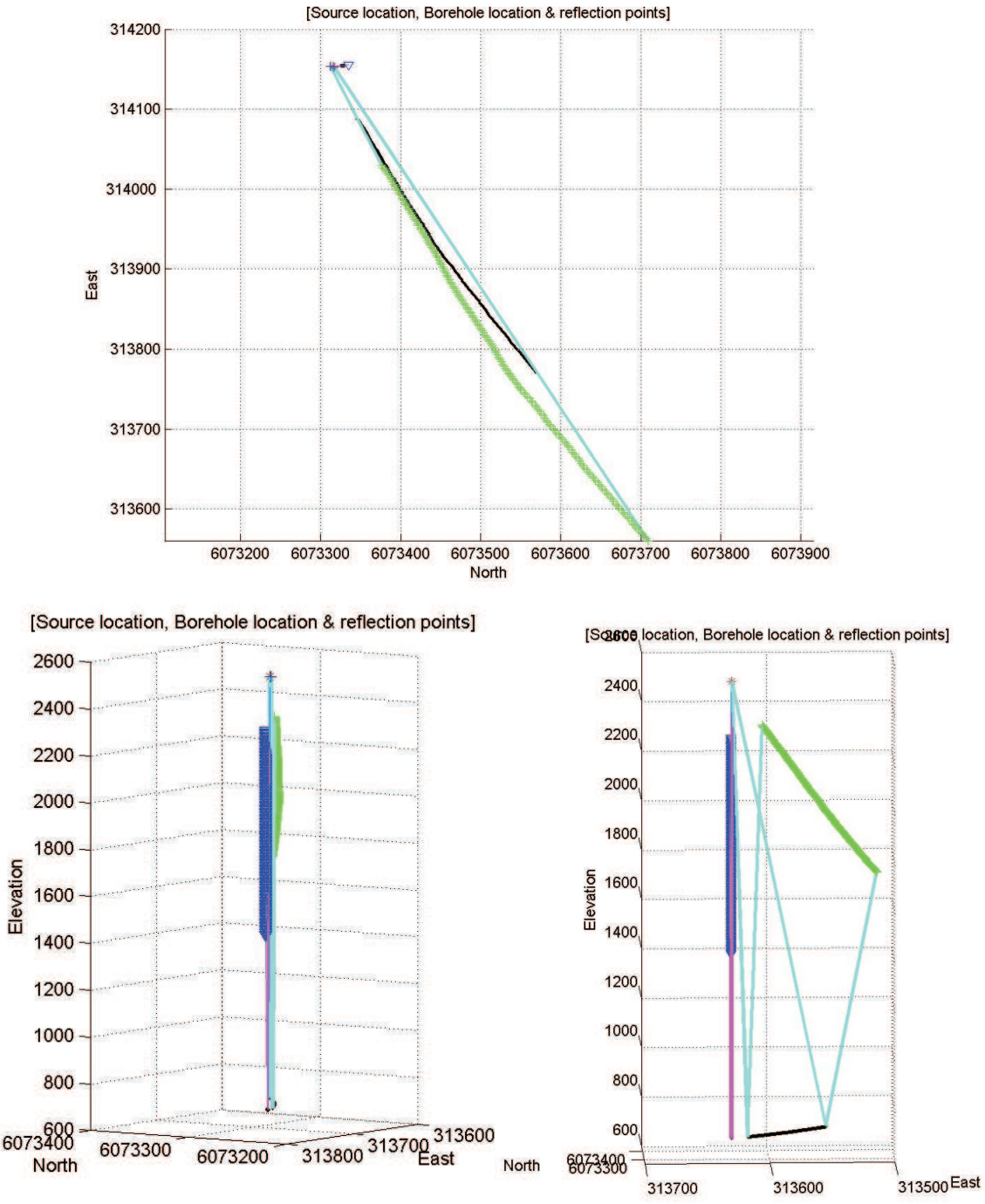

Figure 4.10: Geometry of the real and the straight borehole. Birdeye view on top; in plane with the borehole on the left and orthogonal on the right. The real receiver locations are shown in green crosses, whereas the receiver positions of the straight borehole are shown as blue triangles. The reflection points on the arbitrary horizontal plane in 700 m depth are marked in black. For each borehole the travelled reflection path to the topmost and the lowermost receivers are shown in violet and blue.

(Figure 4.12). This can easily be verified with Figure 4.10. These differences were calculated using the scripts *VSP_DSI2a* (Appendix G.12), *VSP_DSI1b* (Appendix G.13) and *find _dist1* (Appendix G.14). The latter also calculated the trace width over which the assumption of horizontal or straight reflection is still valid. This was done in the following manner: For each receiver-difference of reflected distance pair the gradient was calculated. This gradient was extrapolated over x traces and compared to the real value after x traces. This mismatch needs to be smaller than 15 m. This again represents the 90 degree phase shift of a 60 Hz main frequency wavelet. A width of only 20 traces fulfills this threshold. Since this is a very small window, a width of 28 traces was chosen where for only ten receiver locations this threshold is violated. This is shown in Figure 4.13.

4.2.9 Corridor Stacks

The shifted data were summed to create corridor stacks from nine different time windows. The width of 28 traces is translated into a time depth by the first arrival times. The script *sc_corr_28...* found in Appendix G.8 shows this translation on line 7. The window's start times change from 0.05 s after first arrival to 0.45 s increasing in 0.05 s increments. Thus stacks from different vicinities of the borehole are produced. The script *sc_new_comp_corr* (Appendix G.9) merges all corridor stacks into one file. Figure 4.14 shows these stacks merged into one file with and without an AGC applied. Note that only the six earlier stacks are shown. The five traces

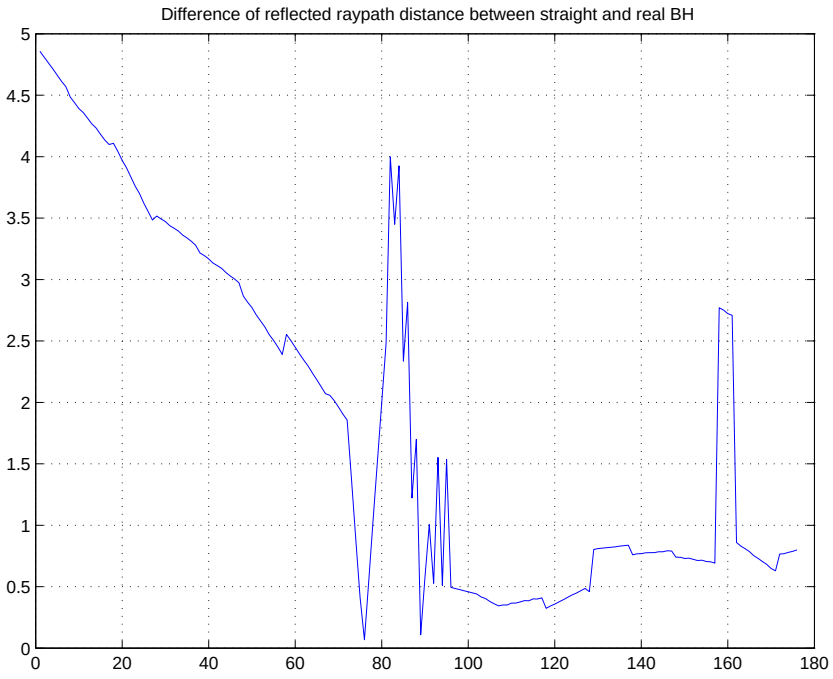

Figure 4.11: Difference between the travelled distance from source directly to the straight and to the real borehole.

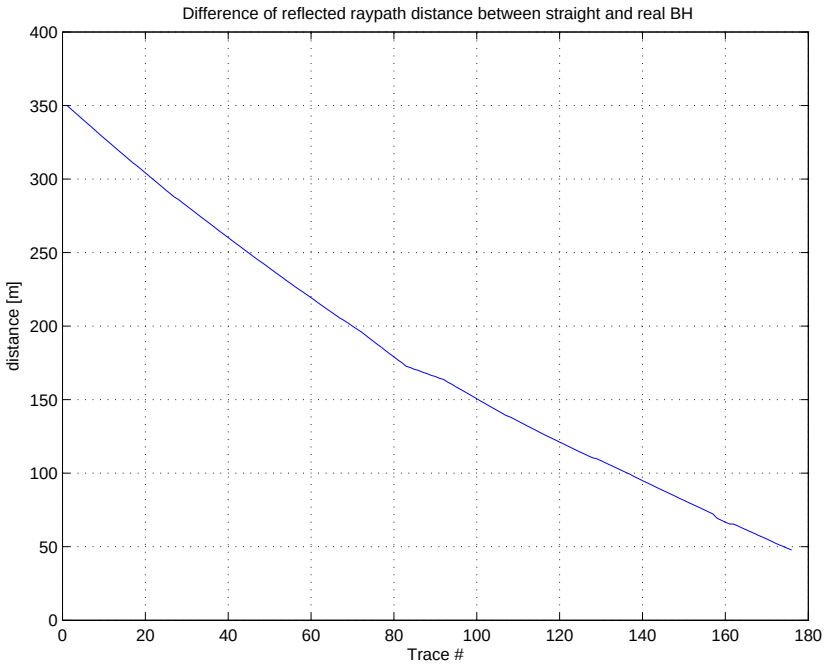

Figure 4.12: Travelled distances were calculated from source to reflection plane and the real or the straight borehole. These differences of the reflected travelled distance are shown.

on the left and in between are the stacked traces from the whole dataset - again with and without AGC. The lithologic log is shown on the right. An enlarged picture of the lithologic log including its legend is also shown in Appendix A.5. The lithologic log shown here is a simplification of the drilled core. Reflections within "one" unit could possibly be caused by a change of physical properties.

Figure 4.15 shows the TWT shifted data, the corridor stacks and the lithologic log for comparison. A larger picture of the lithologic log and the legend is shown in Appendix A.5. The most prominent reflection is at 0.45 s matching the lithologic boundary in 1035 m depth nicely.

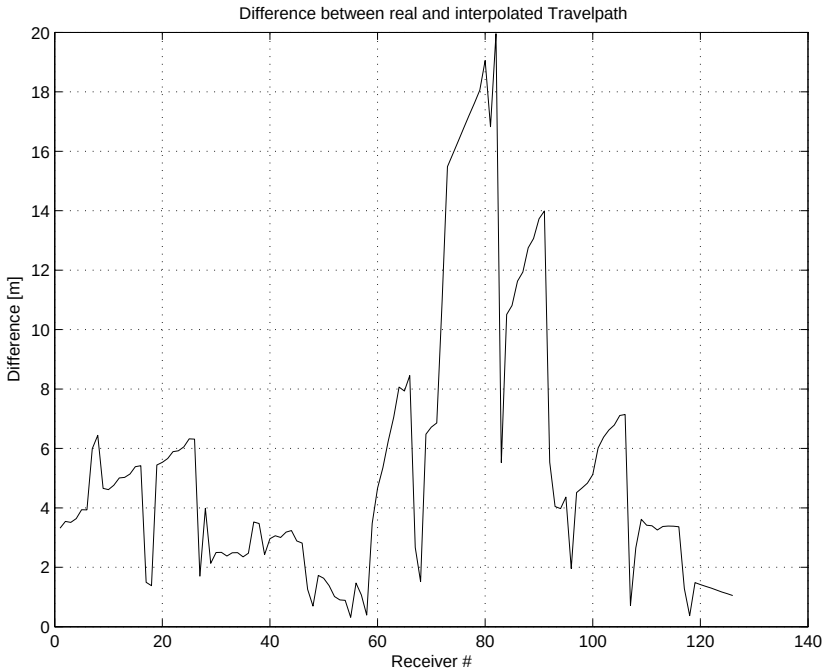

Figure 4.13: For each receiver the gradient of difference of the reflected travelpath is calculated and extrapolated over 28 traces. Here the difference of the real and the extrapolated value is shown.

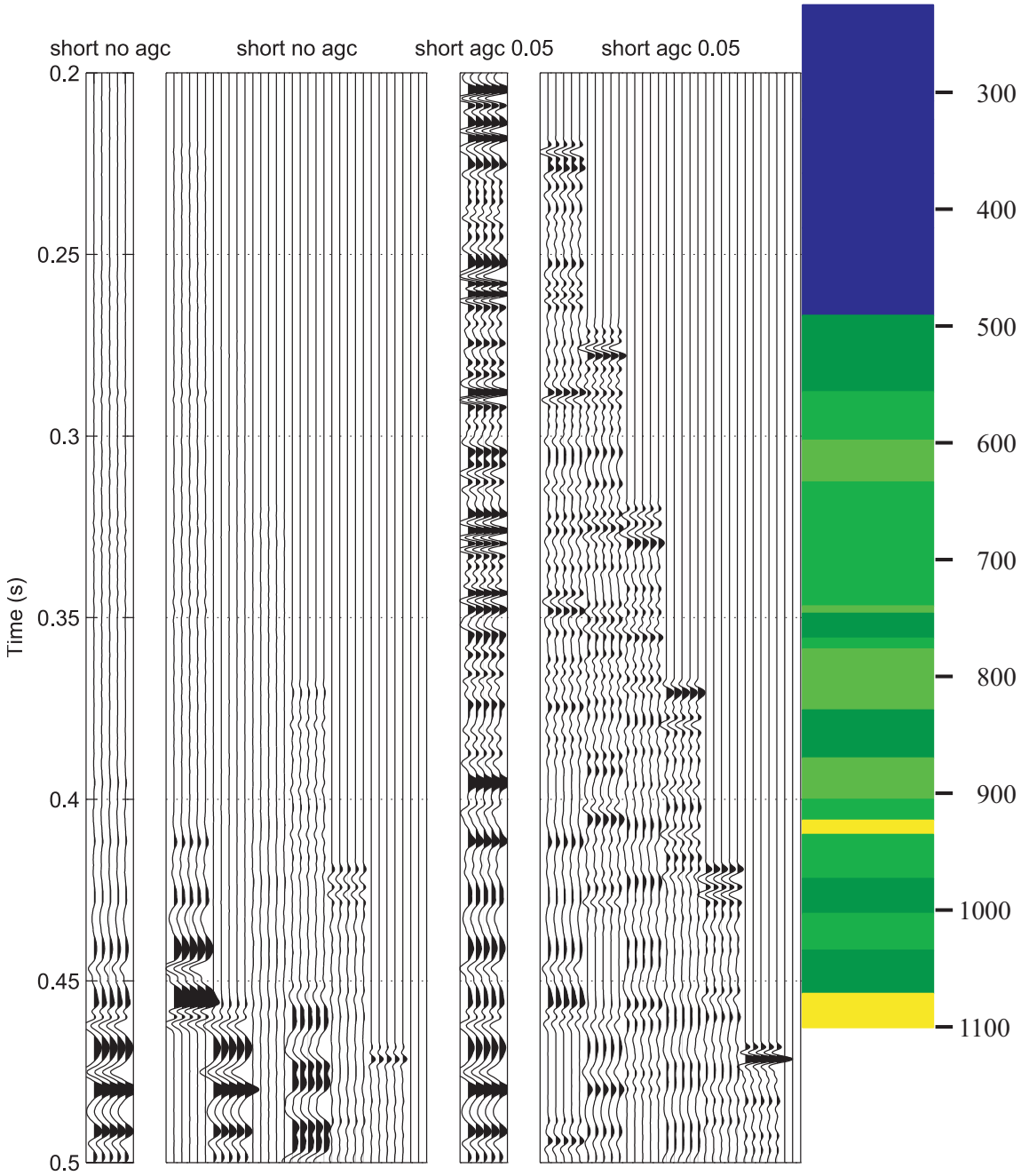

Figure 4.14: Corridor stacks and the lithologic log are shown. An enlarged picture of the lithologic log including its legend is also shown in Appendix A.5. From left to right: Stacked traces from the whole dataset, no AGC applied; stacked traces from the six early data windows, no AGC applied; whole dataset with an AGC of 0.05 s; early window with an AGC of 0.05 s.

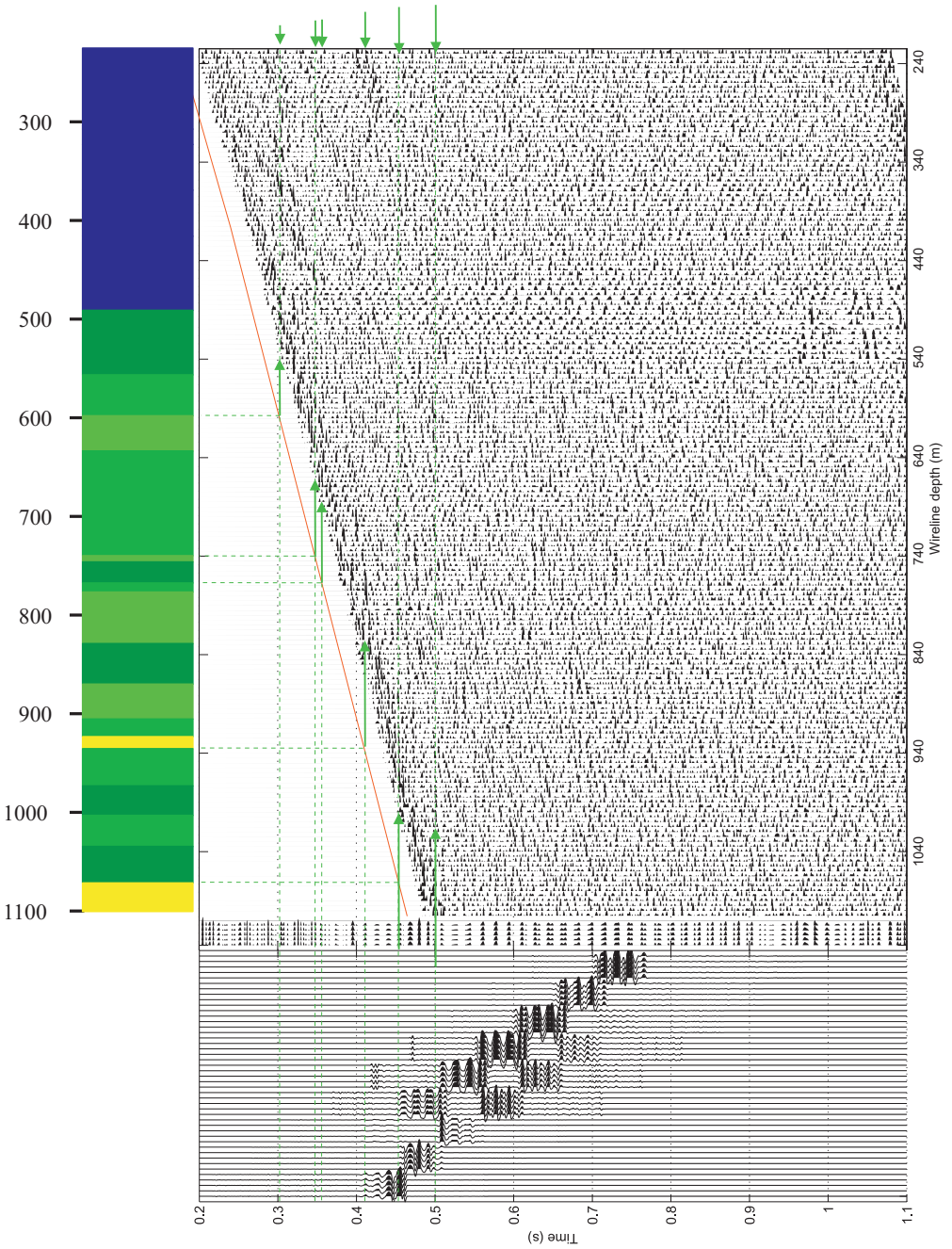

Figure 4.15: TWT shifted data are shown with the corridor stack of the whole dataset as well as the stacks from the nine different windows on the left. On top is the lithologic log (find the enlarged log and legend in Appendix A.5) matching the depth of the receivers and the first arrivals. Some reflections are highlighted in green, the first arrivals are marked orange.

Appendix A

Enlarged Figures of 4Q66W3 near offset

A.1 Removal of P-Wave Energy

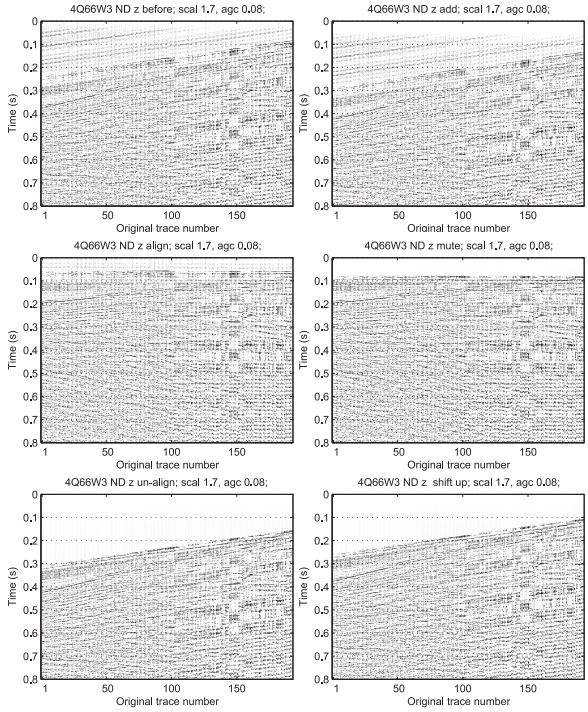

Figure A.1: Removal of the P-wave energy. Figures from upper left to lower right: z-component before removal of P-wave energy; time is added to make sure no data is lost; data aligned at first break; data after top muting; data after shifting back of first break times; result after removing the added time.

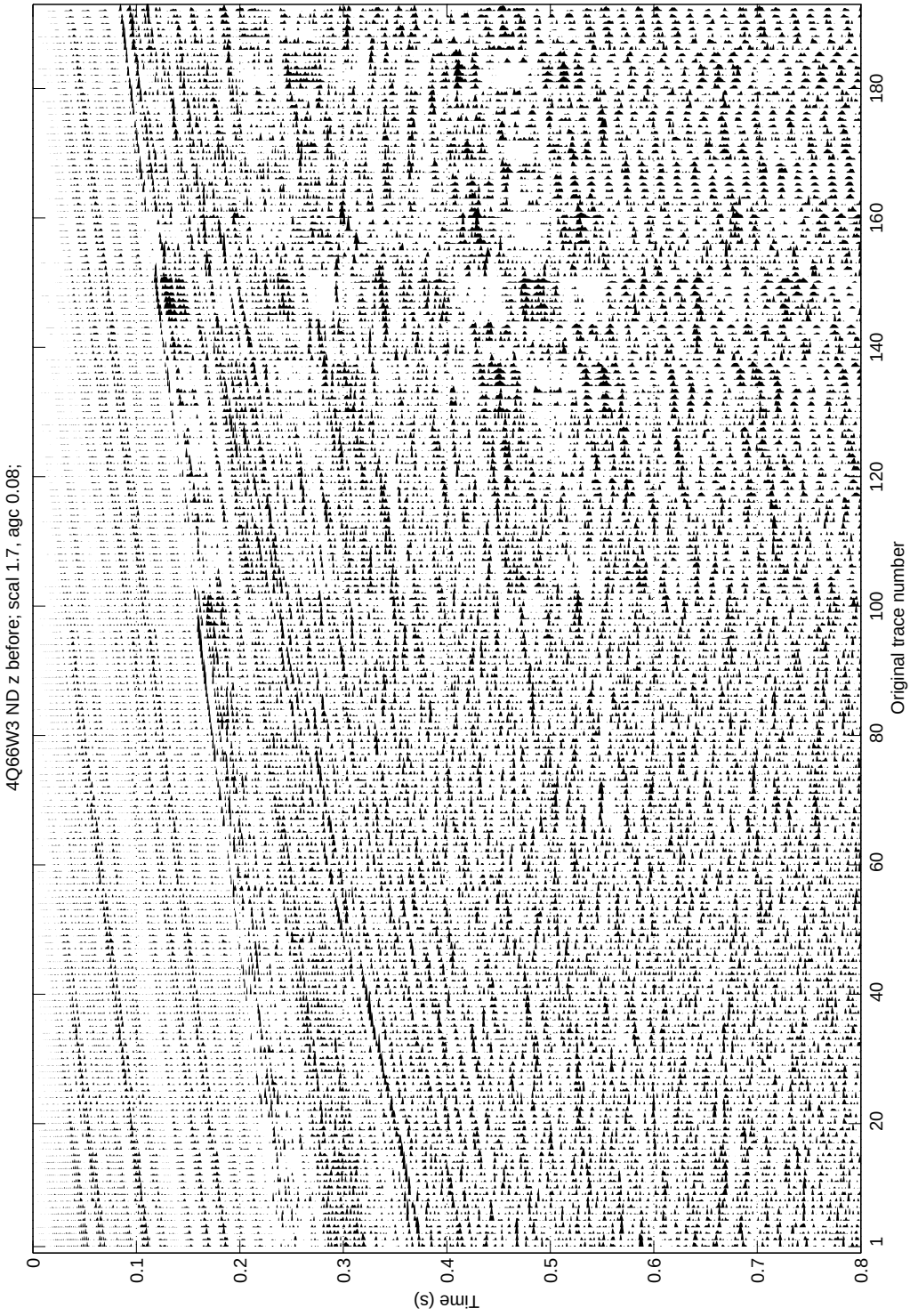

Figure A.2: Notch filtered and scaled data.

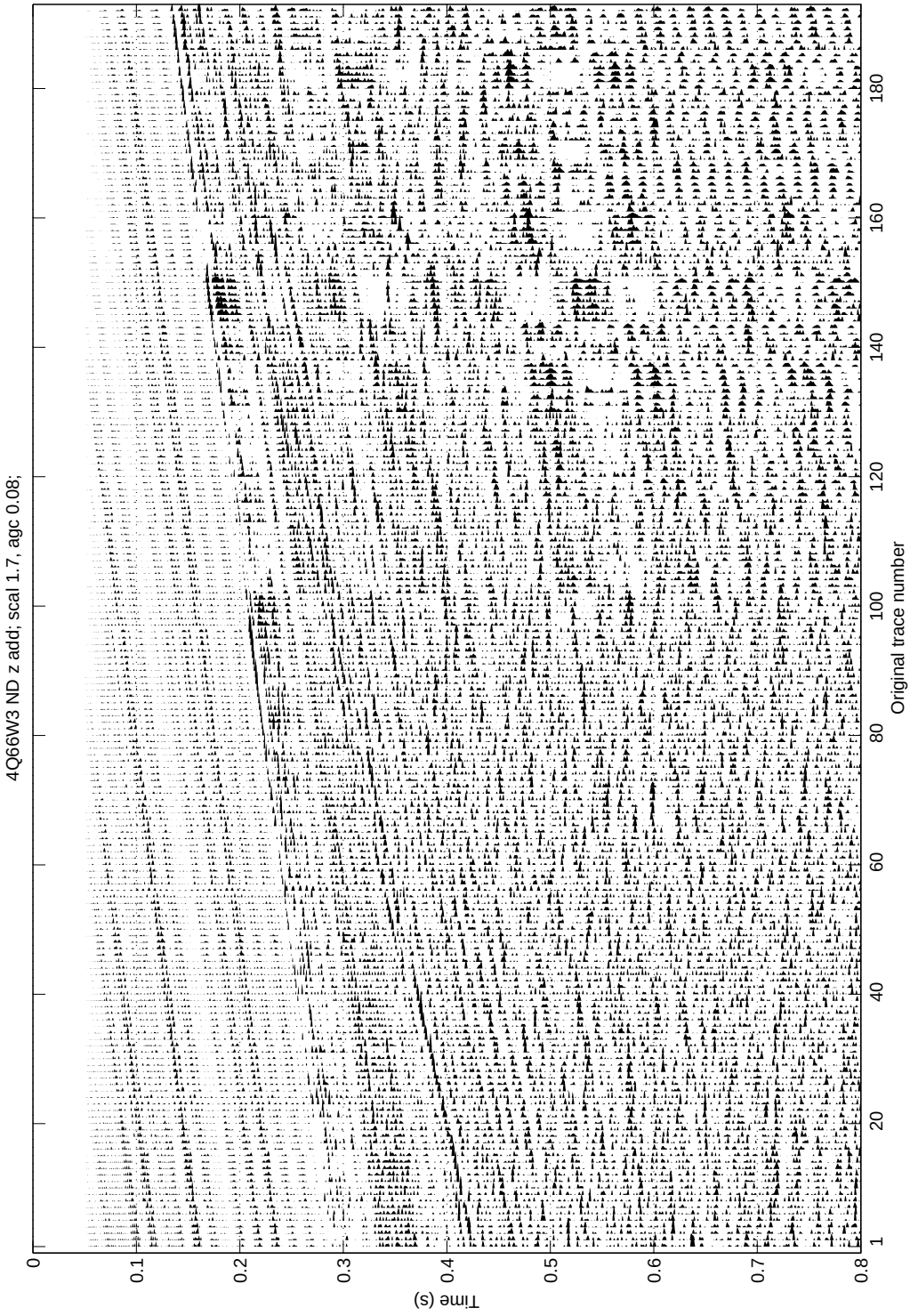

Figure A.3: Data with time added at the beginning of each trace.

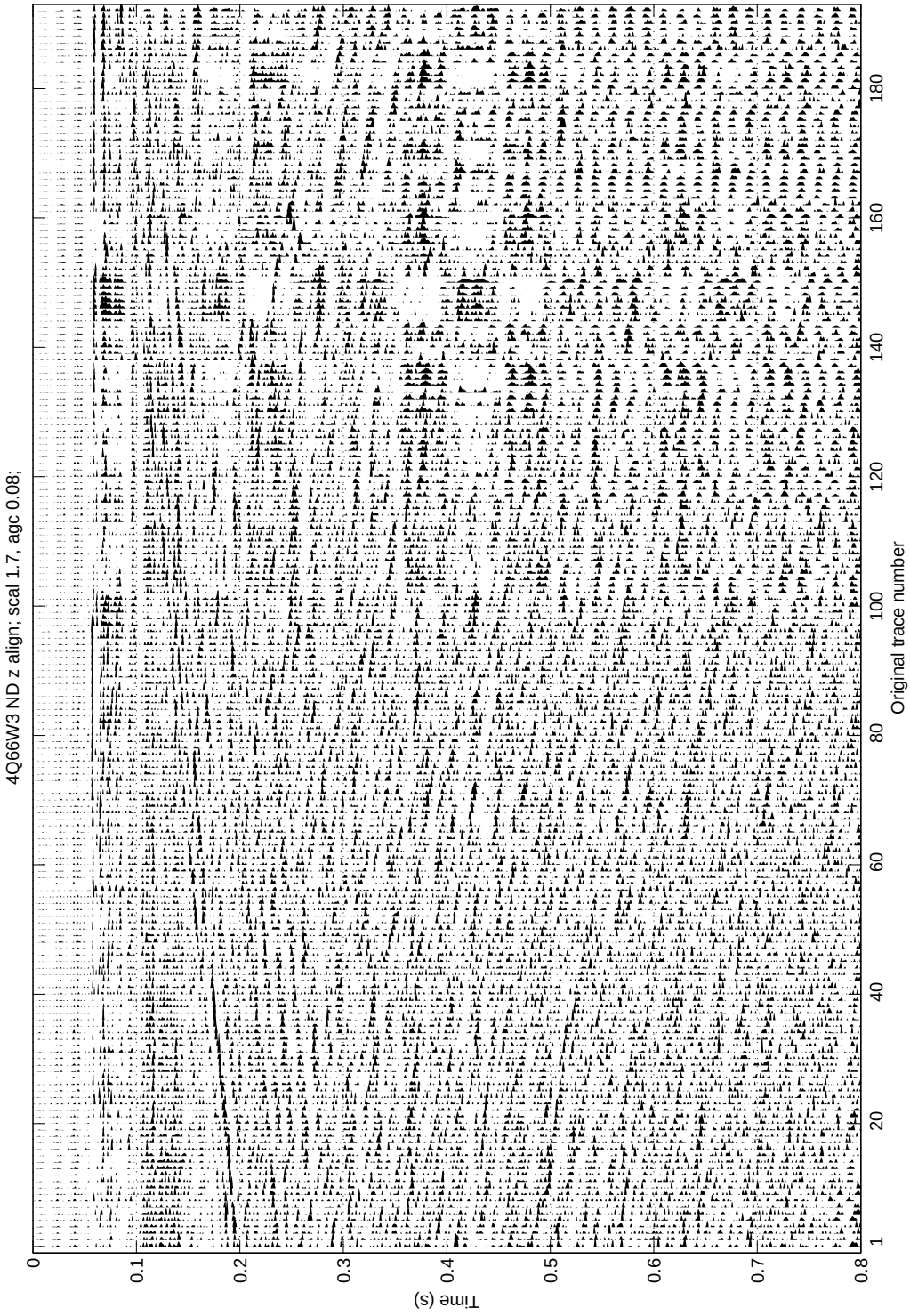

Figure A.4: P-wave arrival time is lined up horizontally.

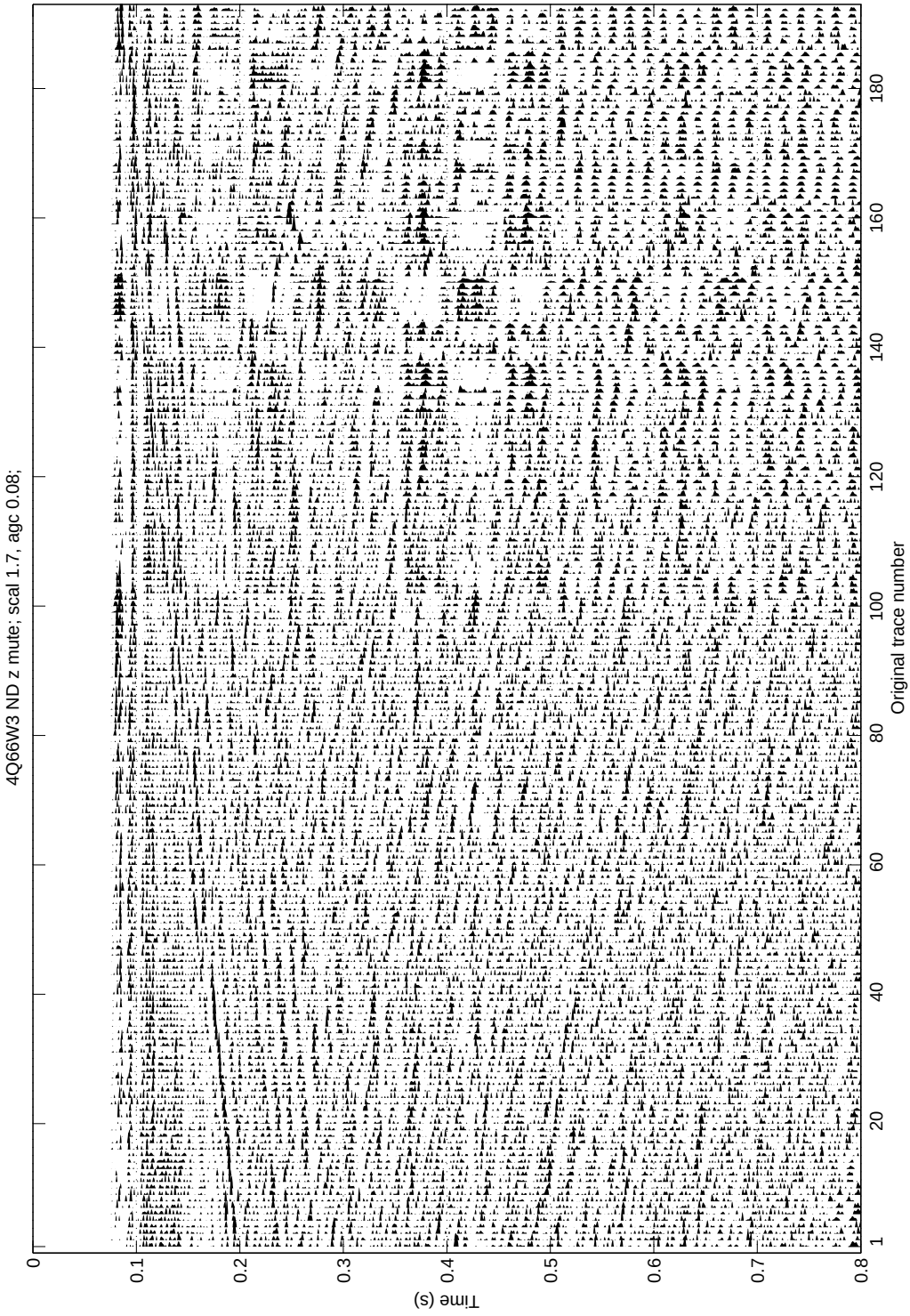

Figure A.5: First P arrivals are muted.

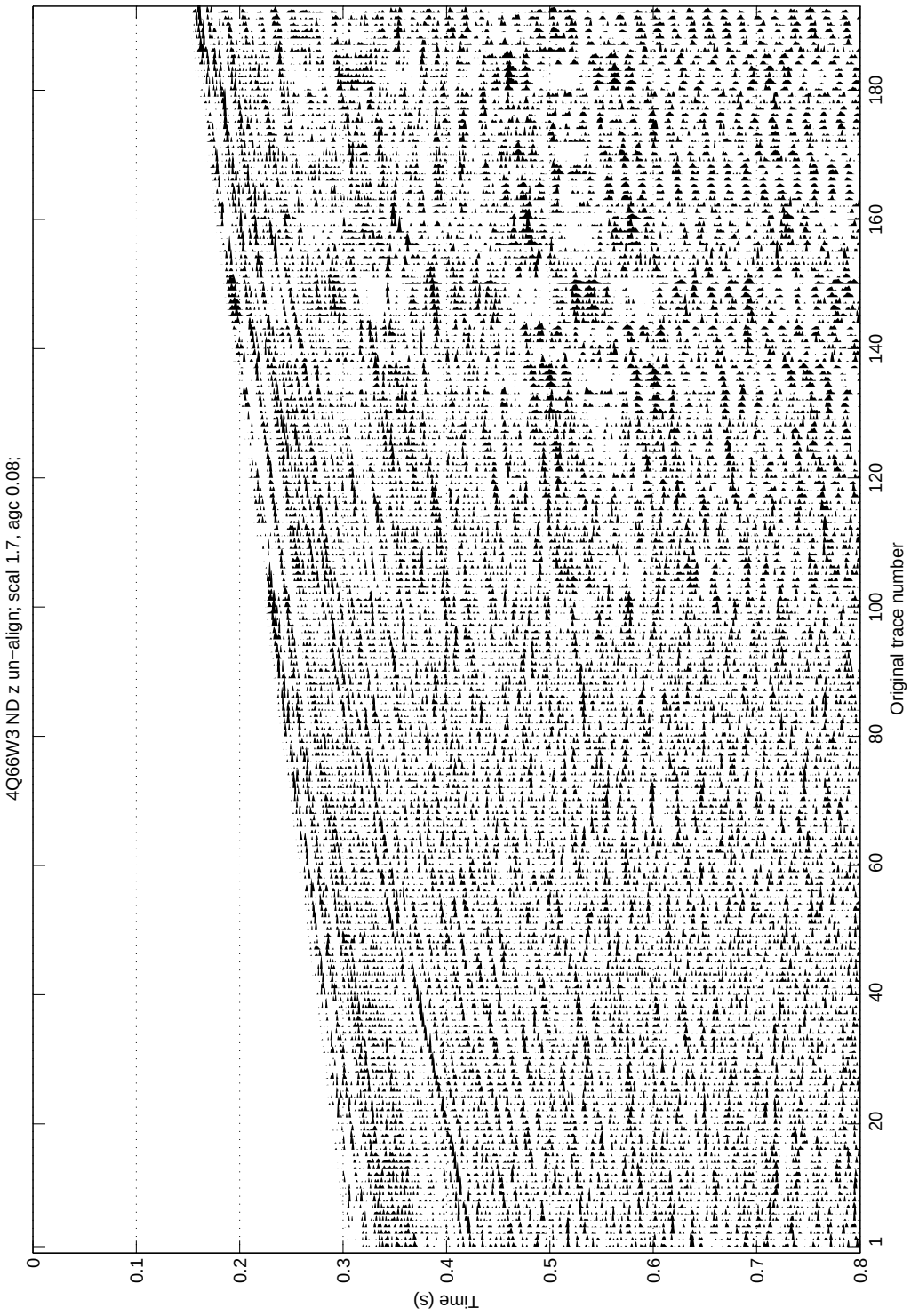

Figure A.6: Data is shifted back.

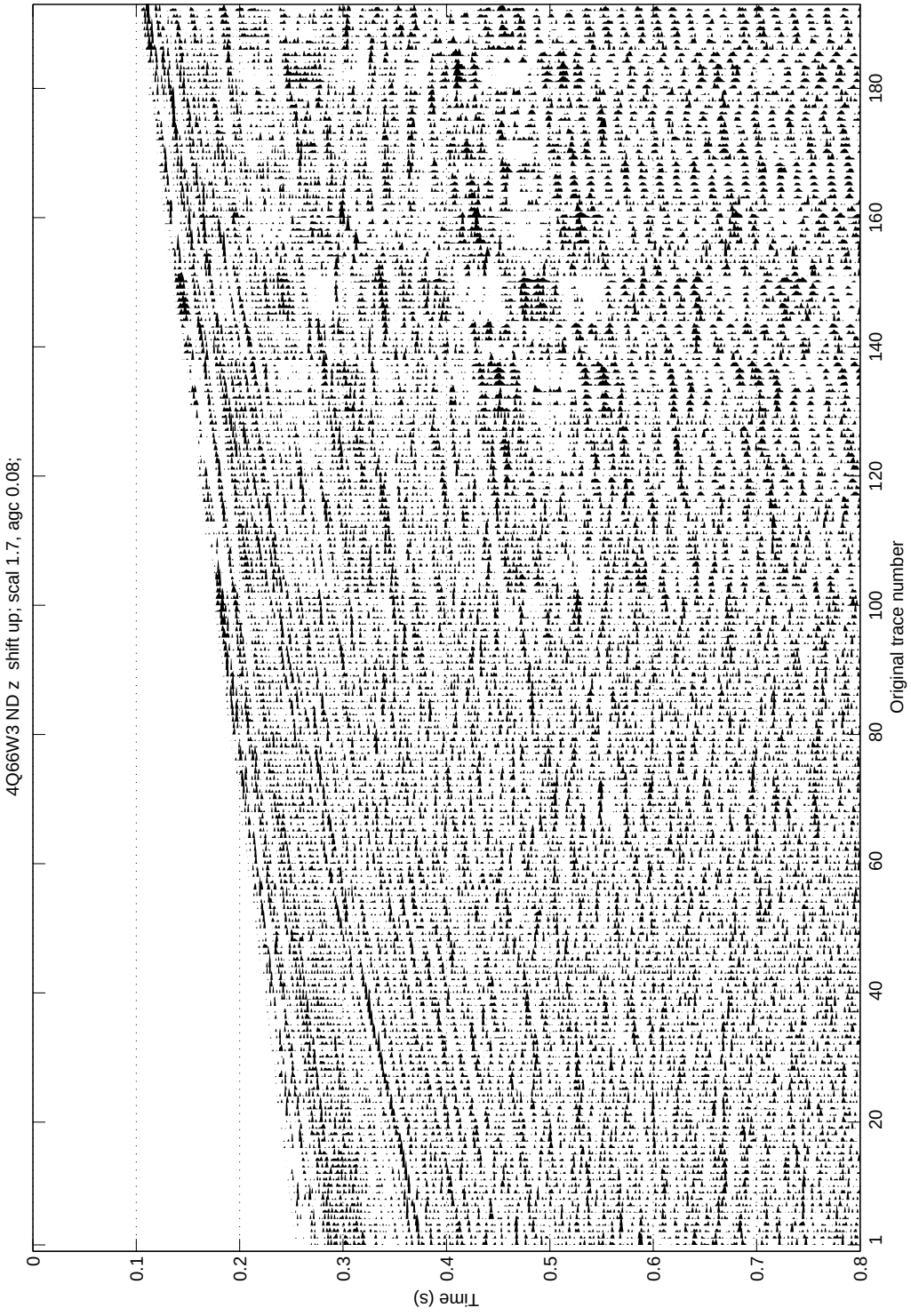

Figure A.7: The added time is removed.

A.2 Removal of S-Wave Energy

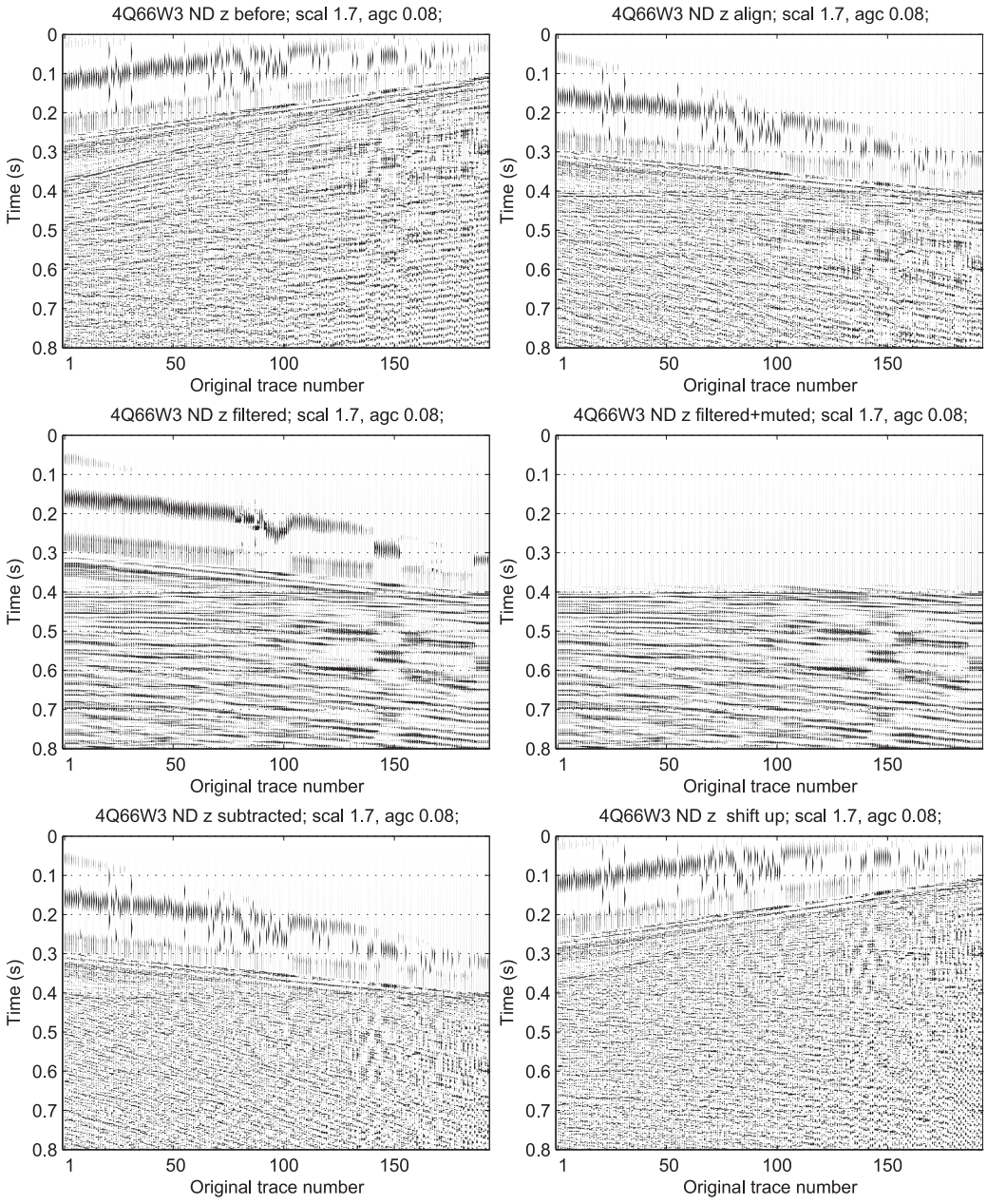

Figure A.8: Removal of the S-wave energy. Figures from upper left to lower right: z-component before removal of S-wave energy; time is added to make sure no data are lost before they are shifted according the first arrival times; median filtered data; filtered data after muting; data after the filtered and muted dataset is subtracted; shifting back of first breaks and removal of the added time.

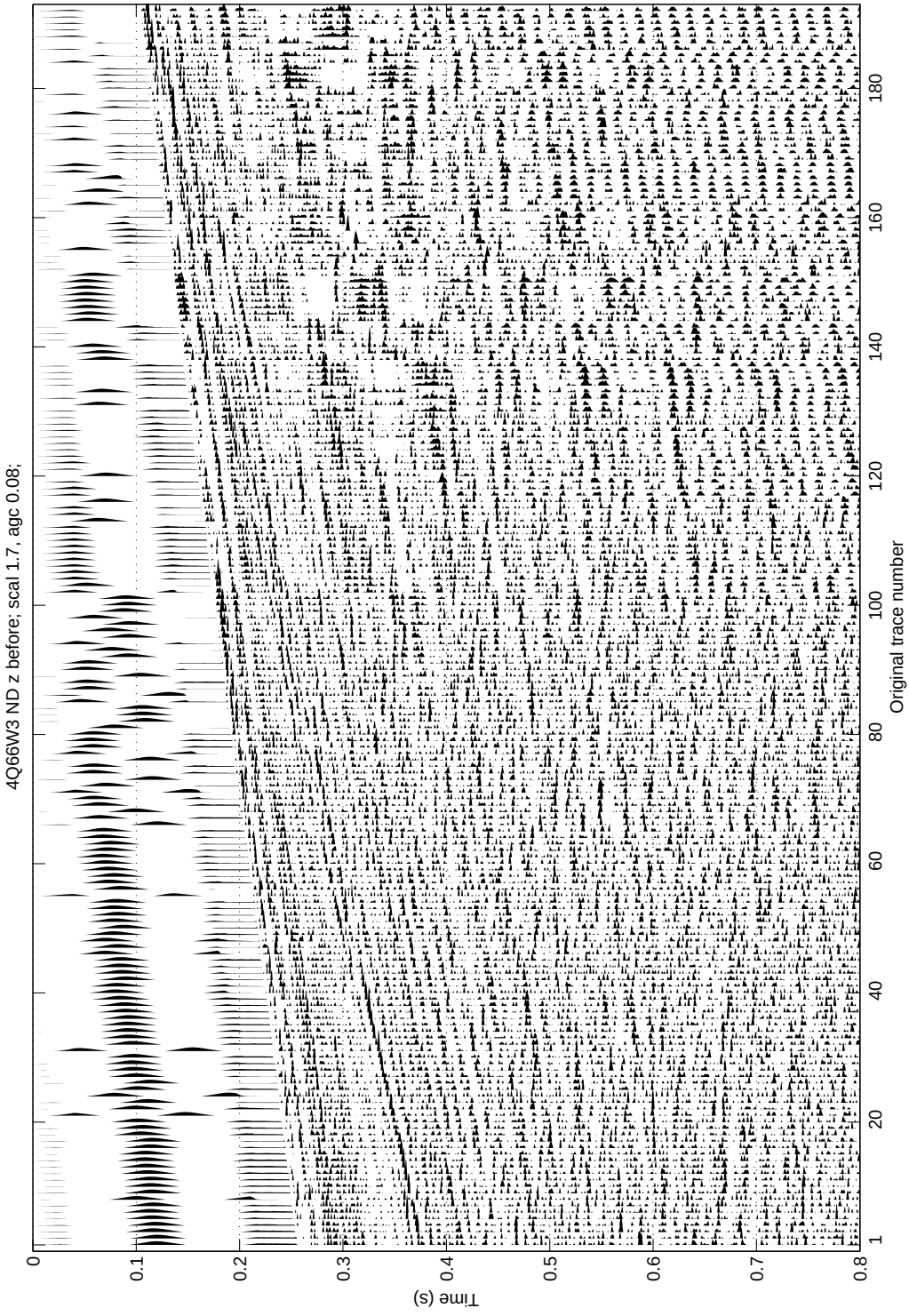

Figure A.9: P removed data.

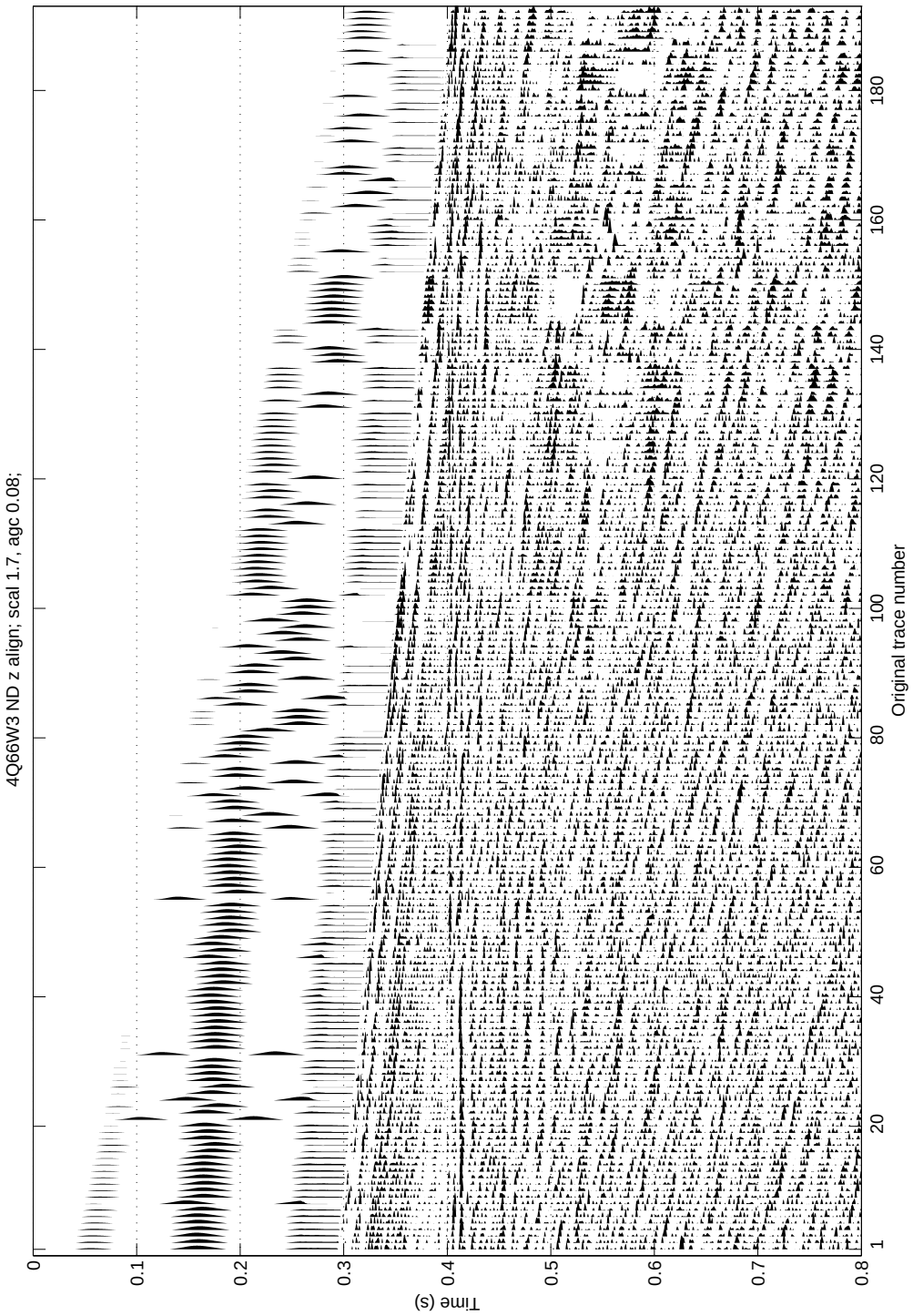

Figure A.10: Data with time added at the beginning of each trace and S-wave arrival time is lined up horizontally.

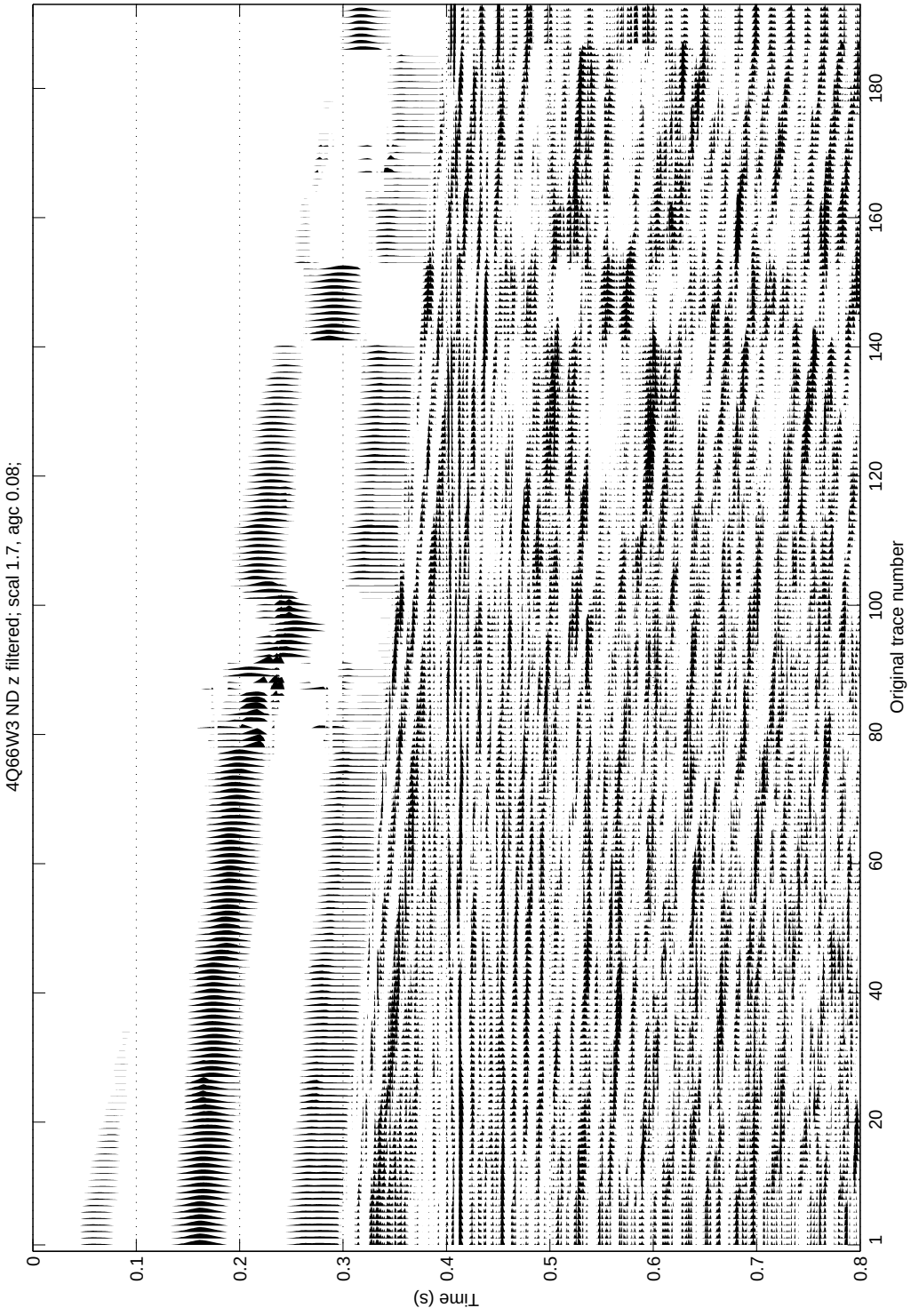

Figure A.11: A median filter of 11 traces width was applied on data.

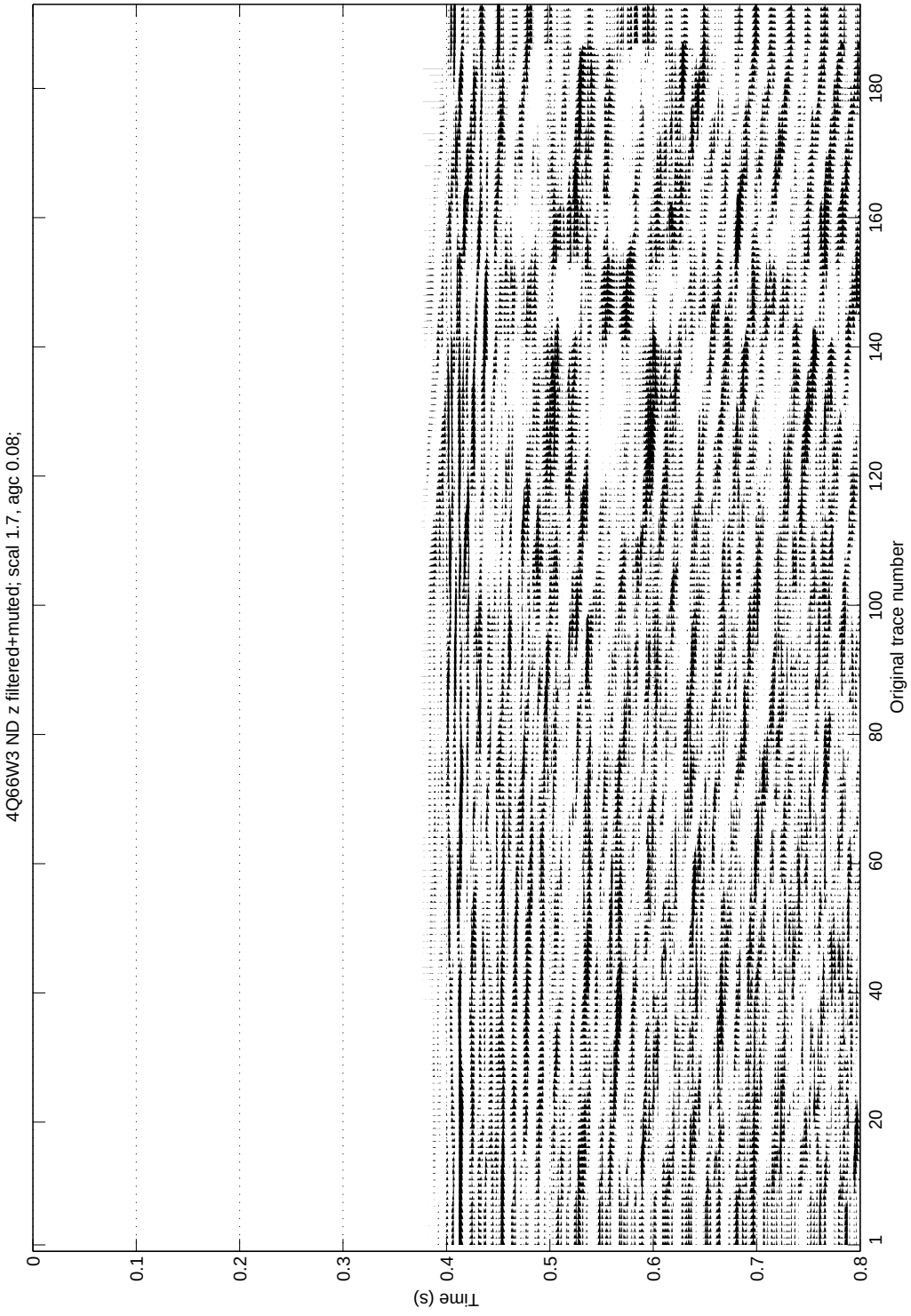

Figure A.12: Data up to the S arrival time are muted.

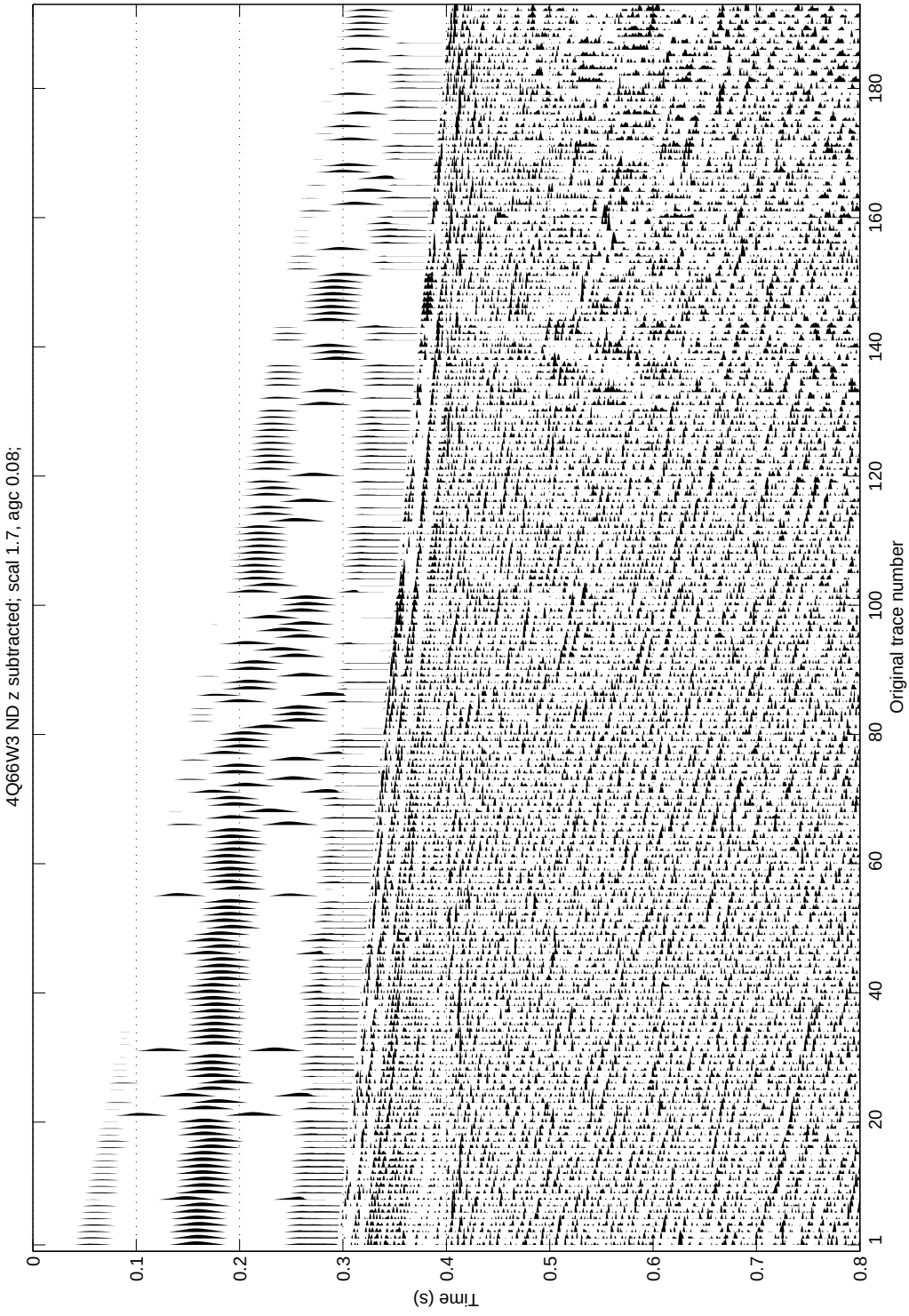

Figure A.13: The median filtered data is subtracted from the shifted data.

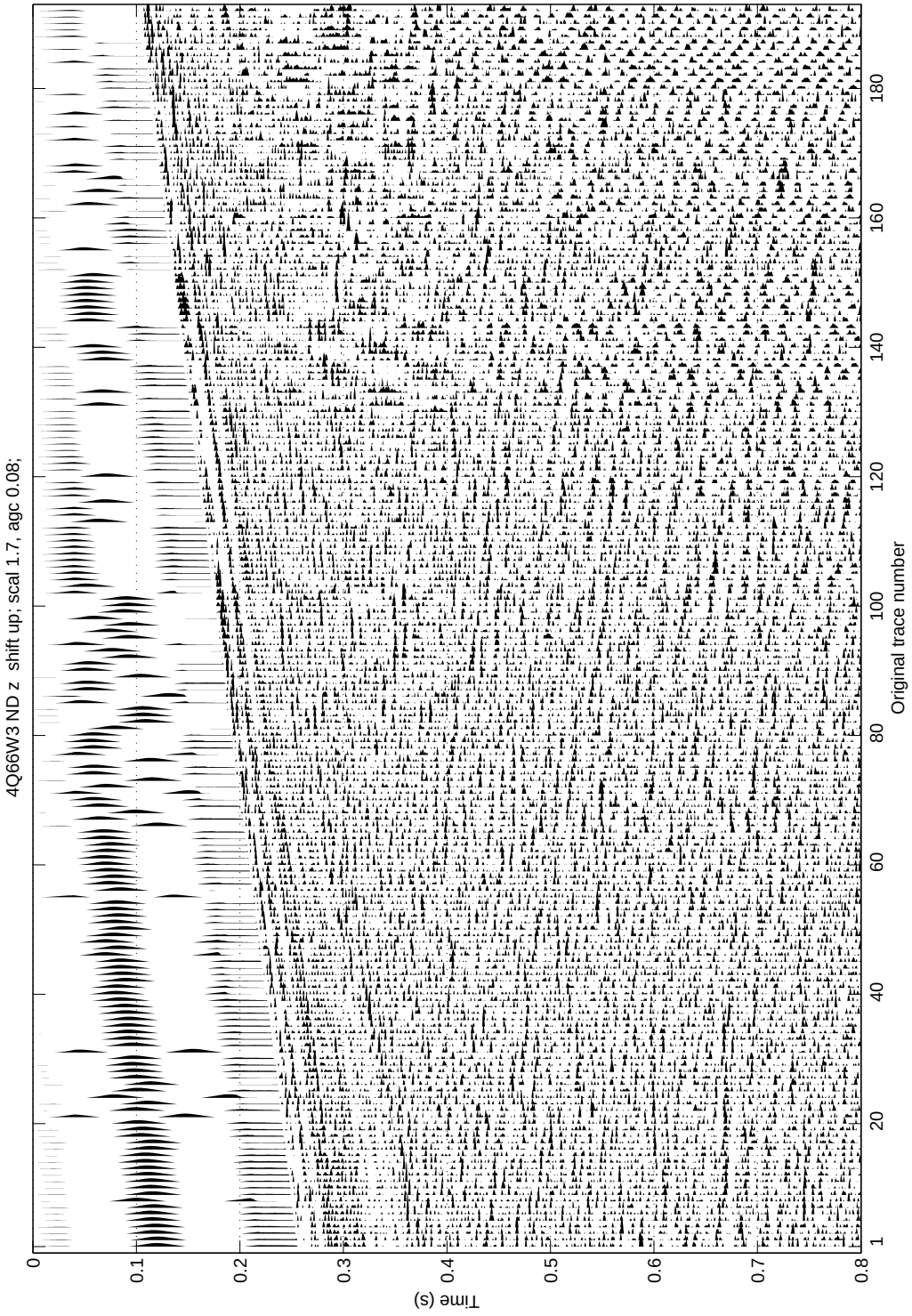

Figure A.14: The added time is removed and the data is shifted back.

A.3 FK Filtering

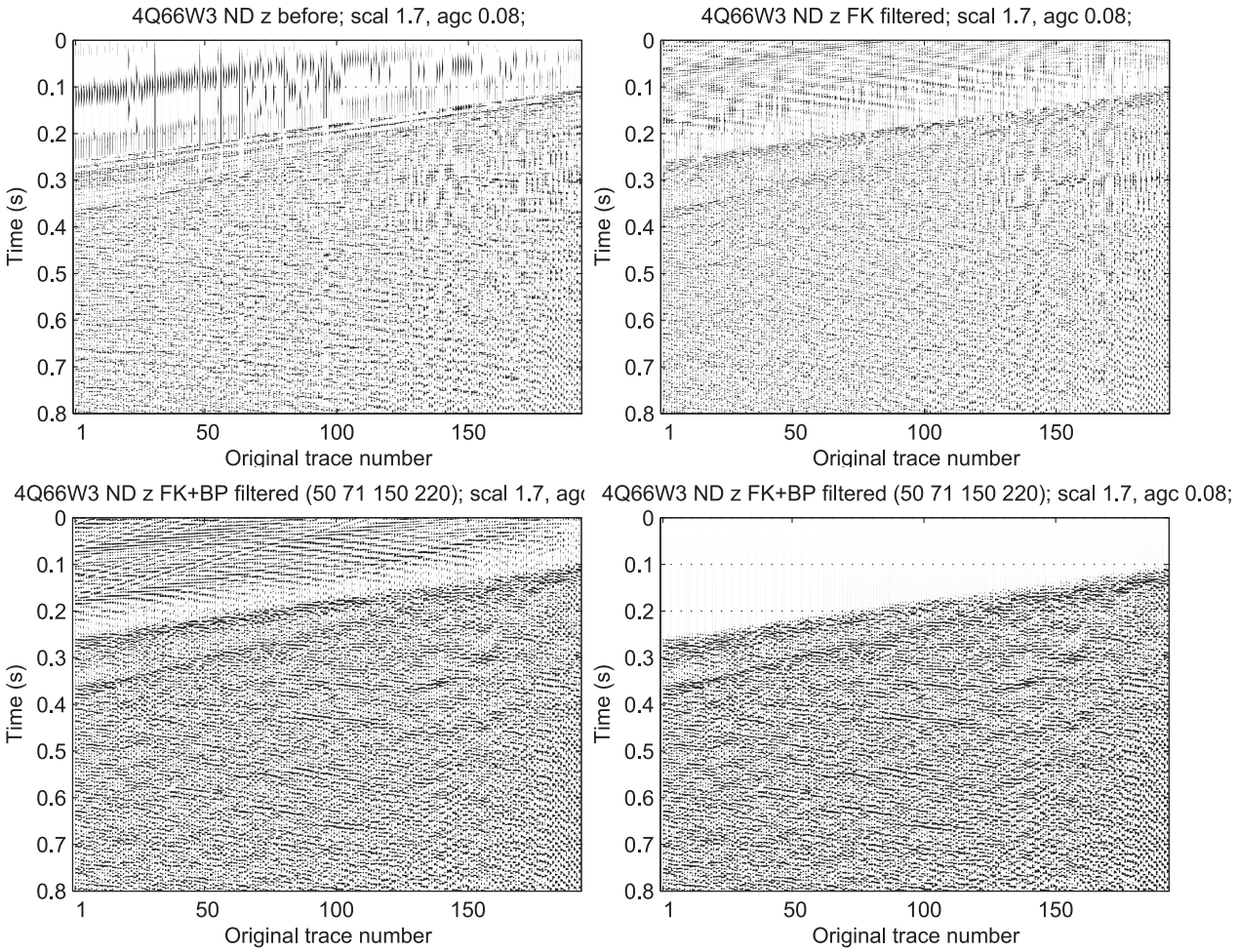

Figure A.15: From upper left to lower right: Data before f-k filtering; f-k filtered data; f-k and bandpass filtered data; result after top muting. Note that all sections are scaled equally and have an AGC of 80 ms applied.

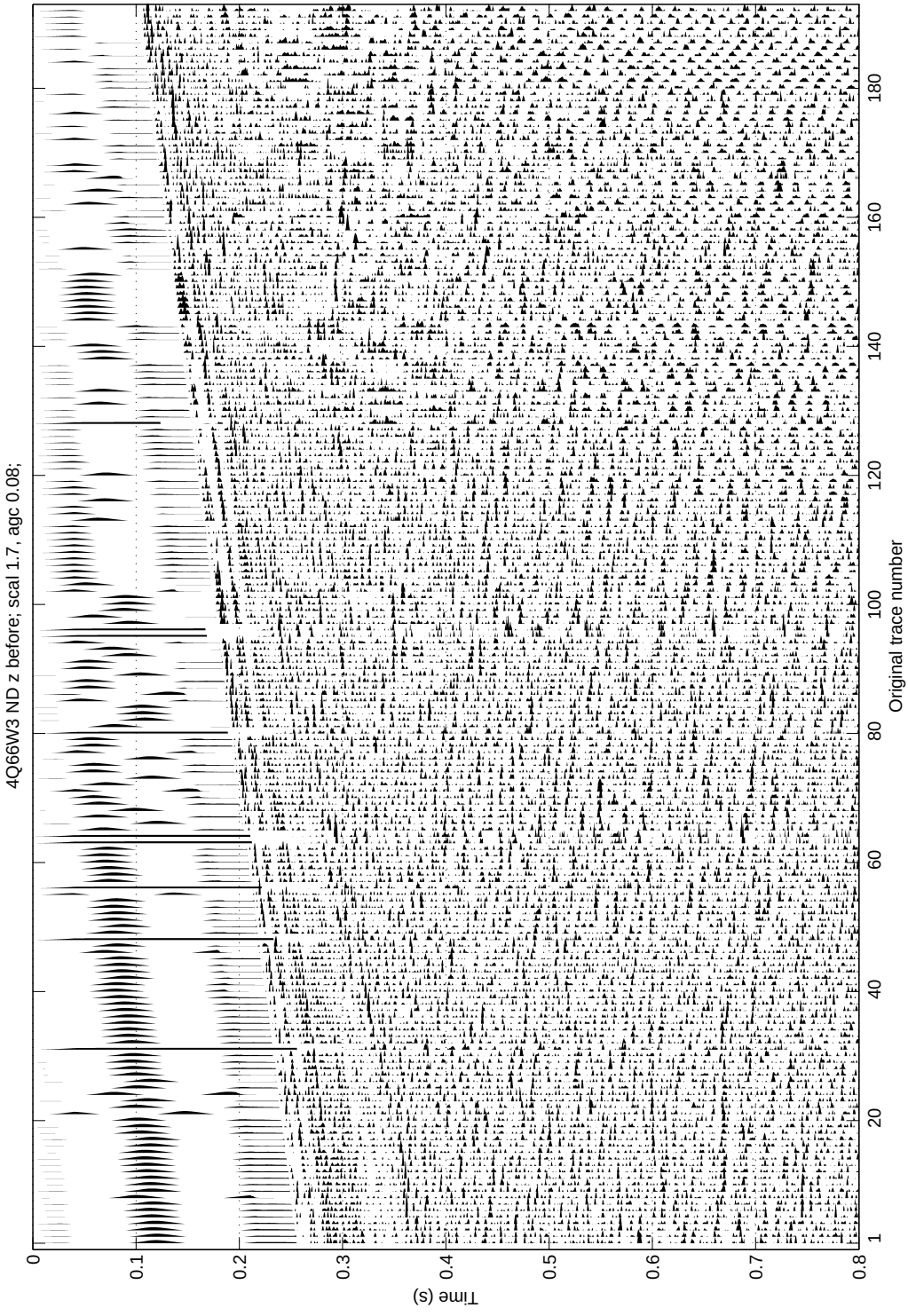

Figure A.16: Data after P and S energy removal.

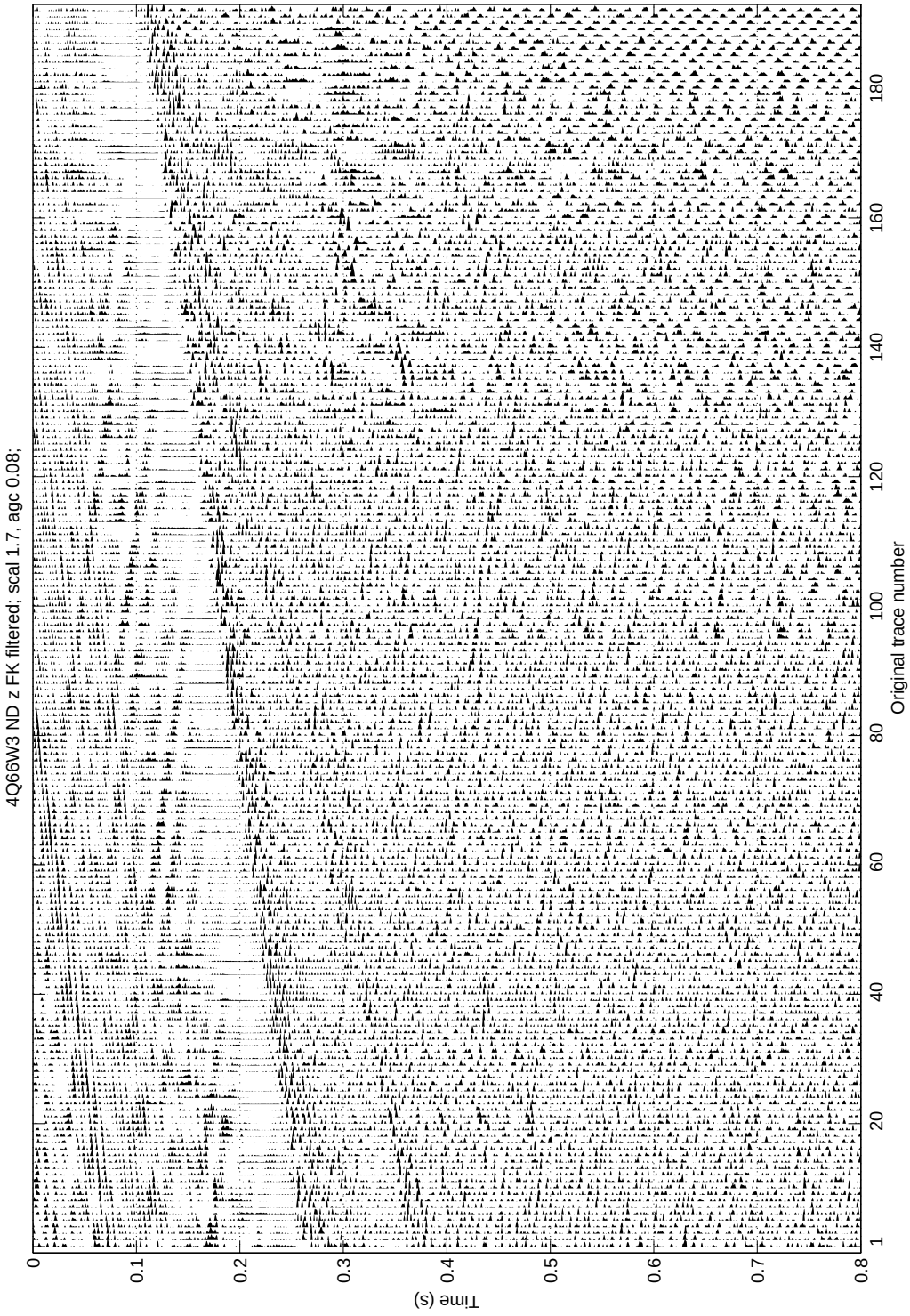

Figure A.17: FK filtered data.

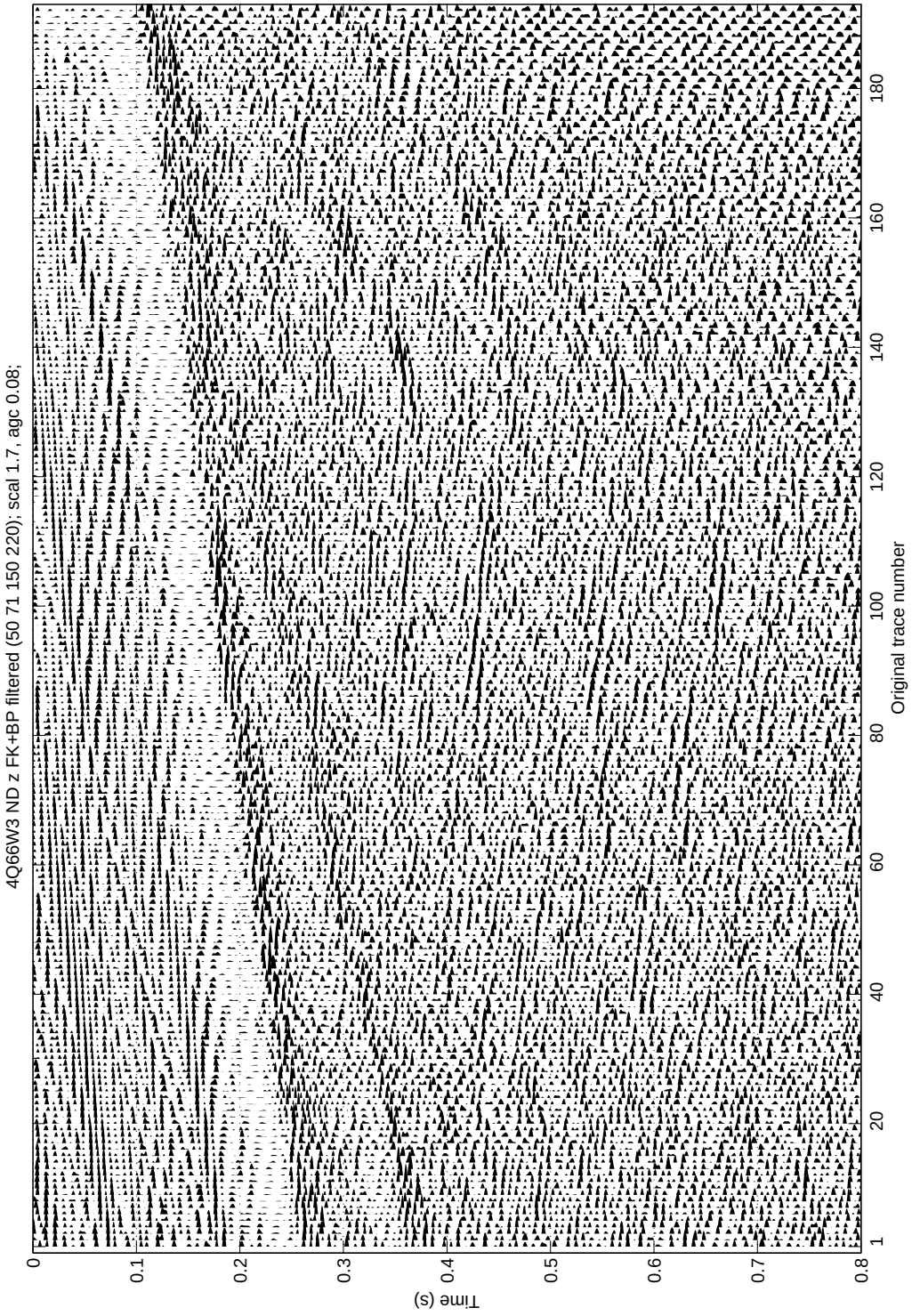

Figure A.18: On the FK filtered data a bandpass filter is applied.

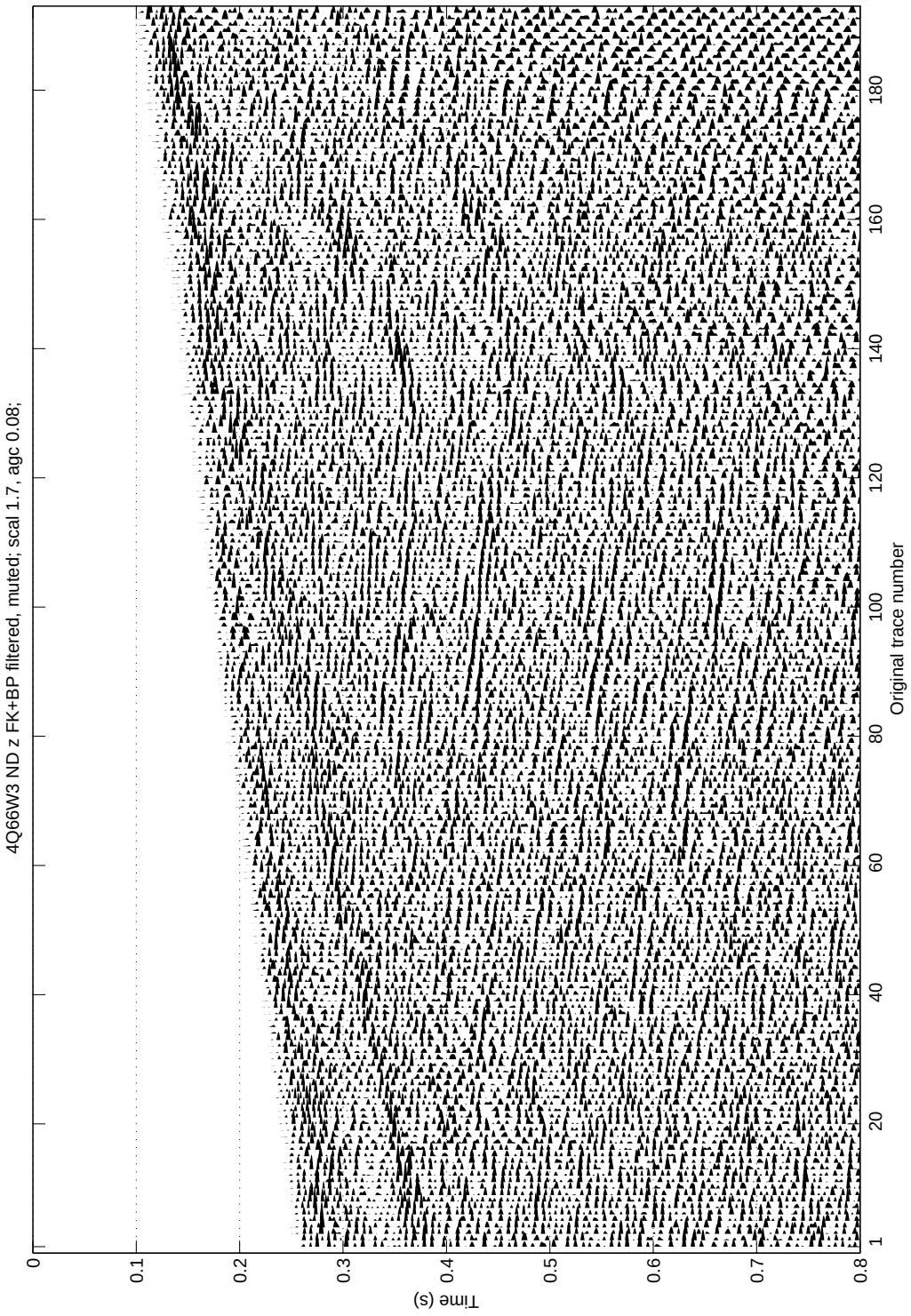

Figure A.19: Muting applied to the trace beginnings.

A.4 Bandpass Filtering

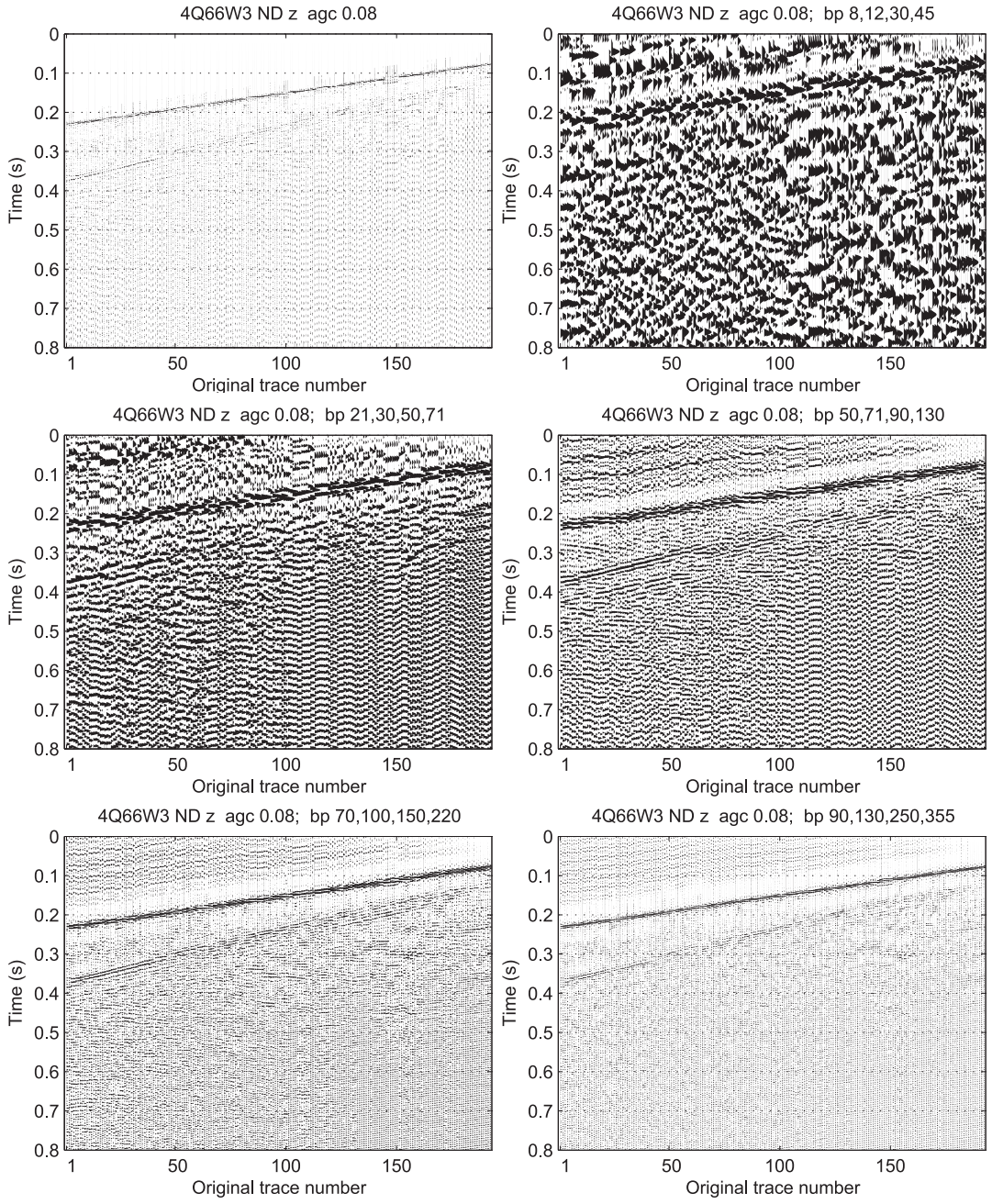

Figure A.20: shows the first arrival muted, scaled data after different bandpass were applied. The upper left figure shows the unfiltered data, figure captions specify the applied filters. An AGC of 80 ms was applied on each dataset. The enlarged panels can be found in Appendix A.4

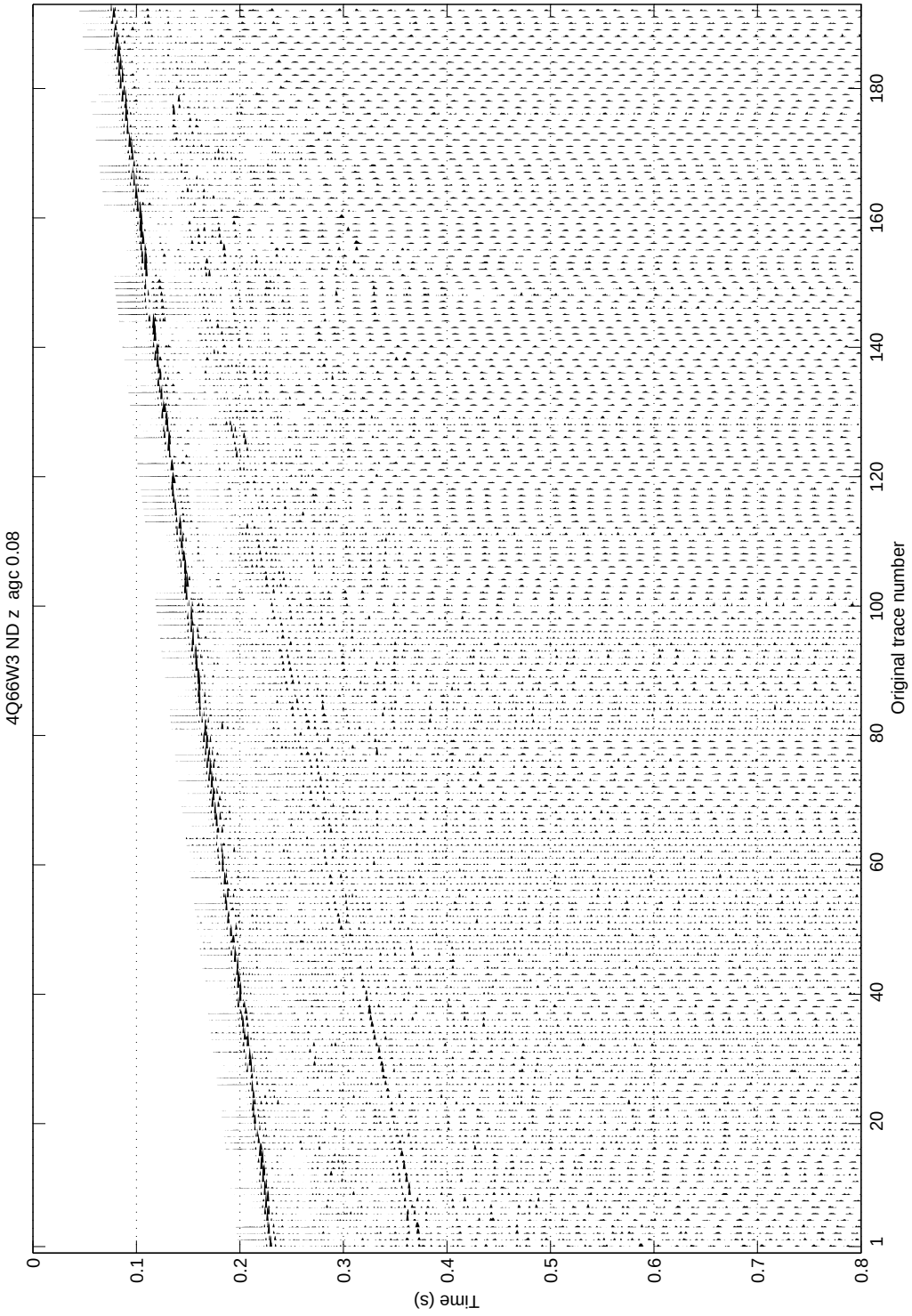

Figure A.21: Raw, but scaled data.

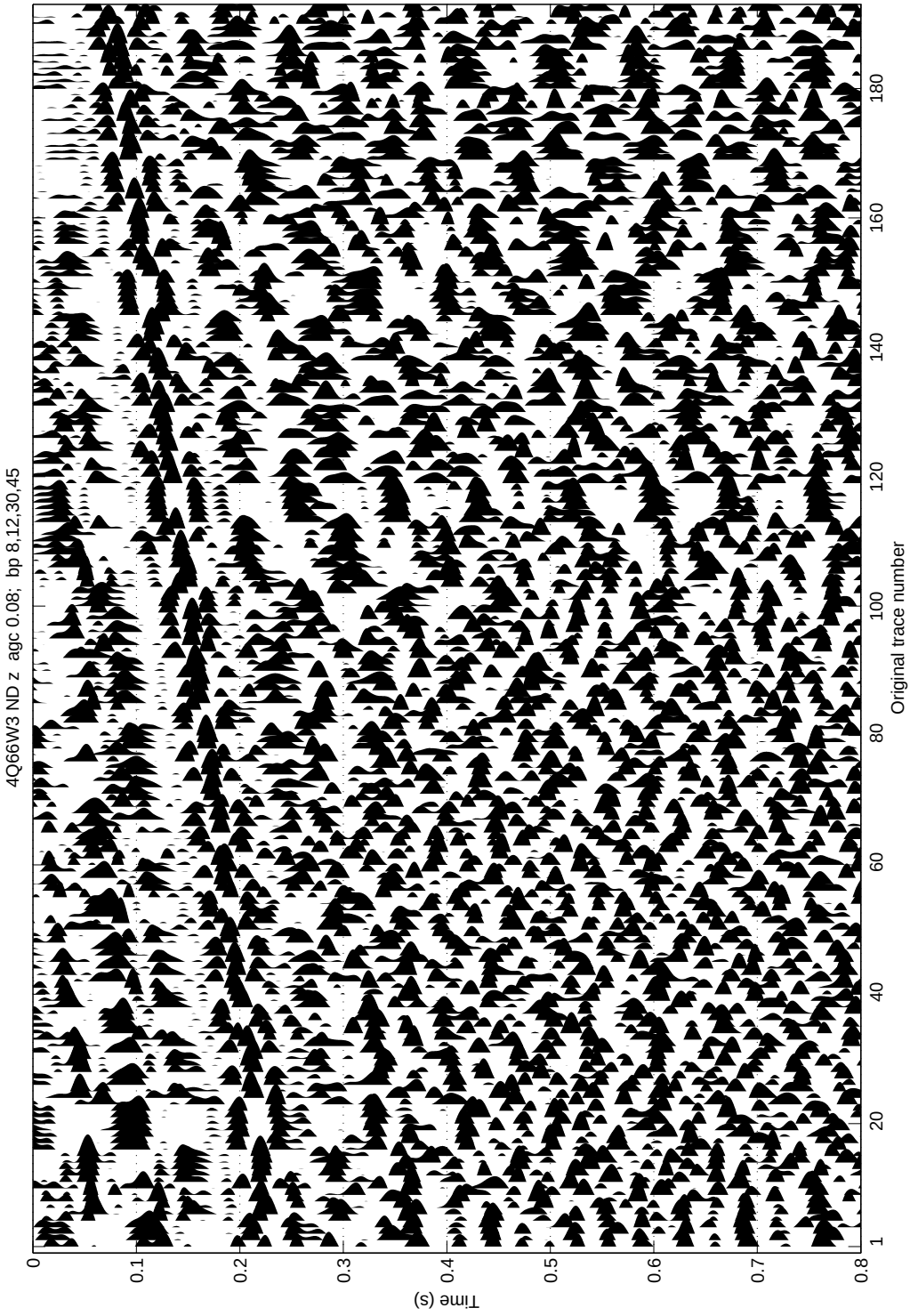

Figure A.22: A bandpass filter with the corner frequencies of 8, 12, 30, 45 Hz was applied

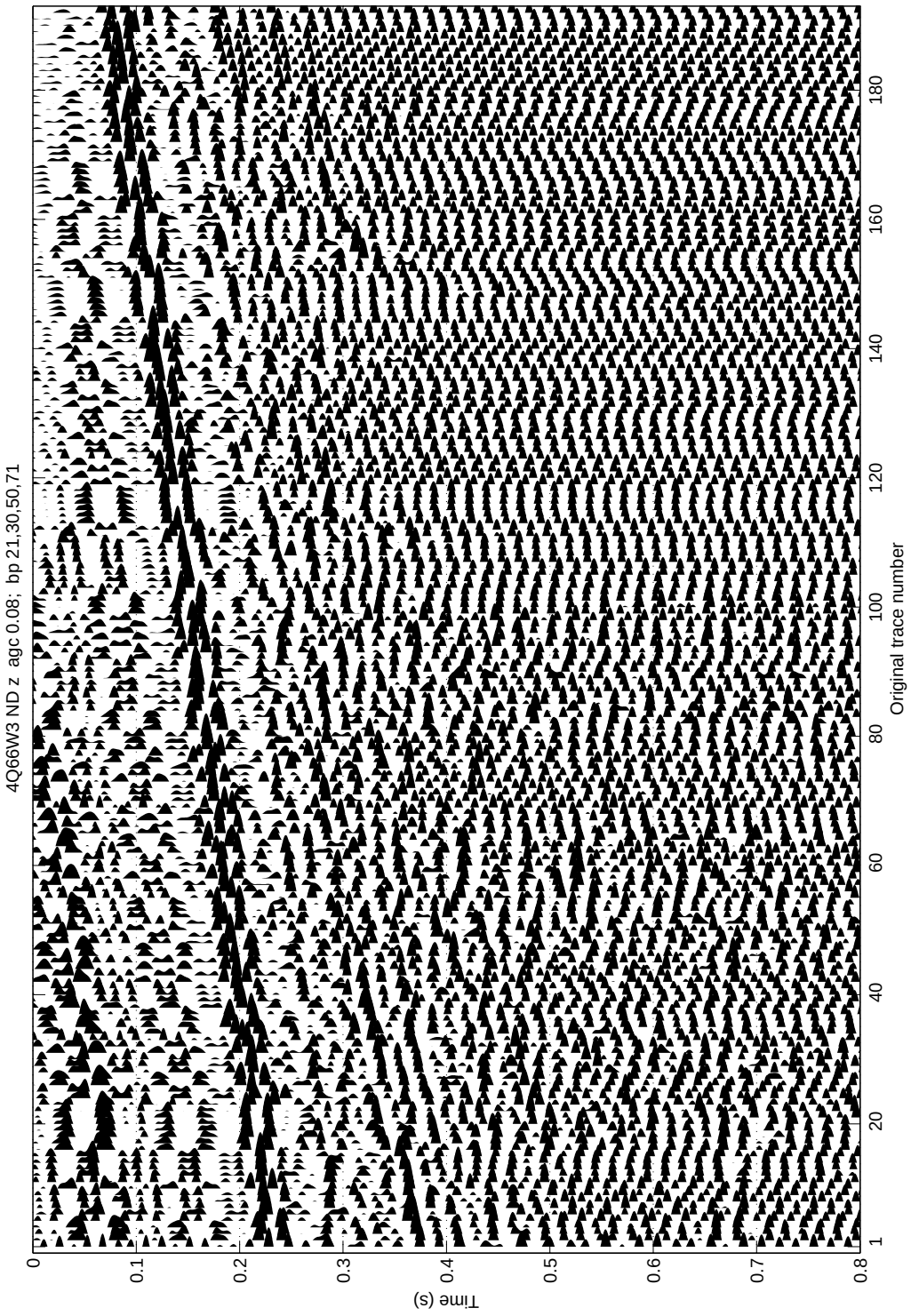

Figure A.23: A bandpass filter with the corner frequencies of 21, 30, 50, 71 Hz was applied.

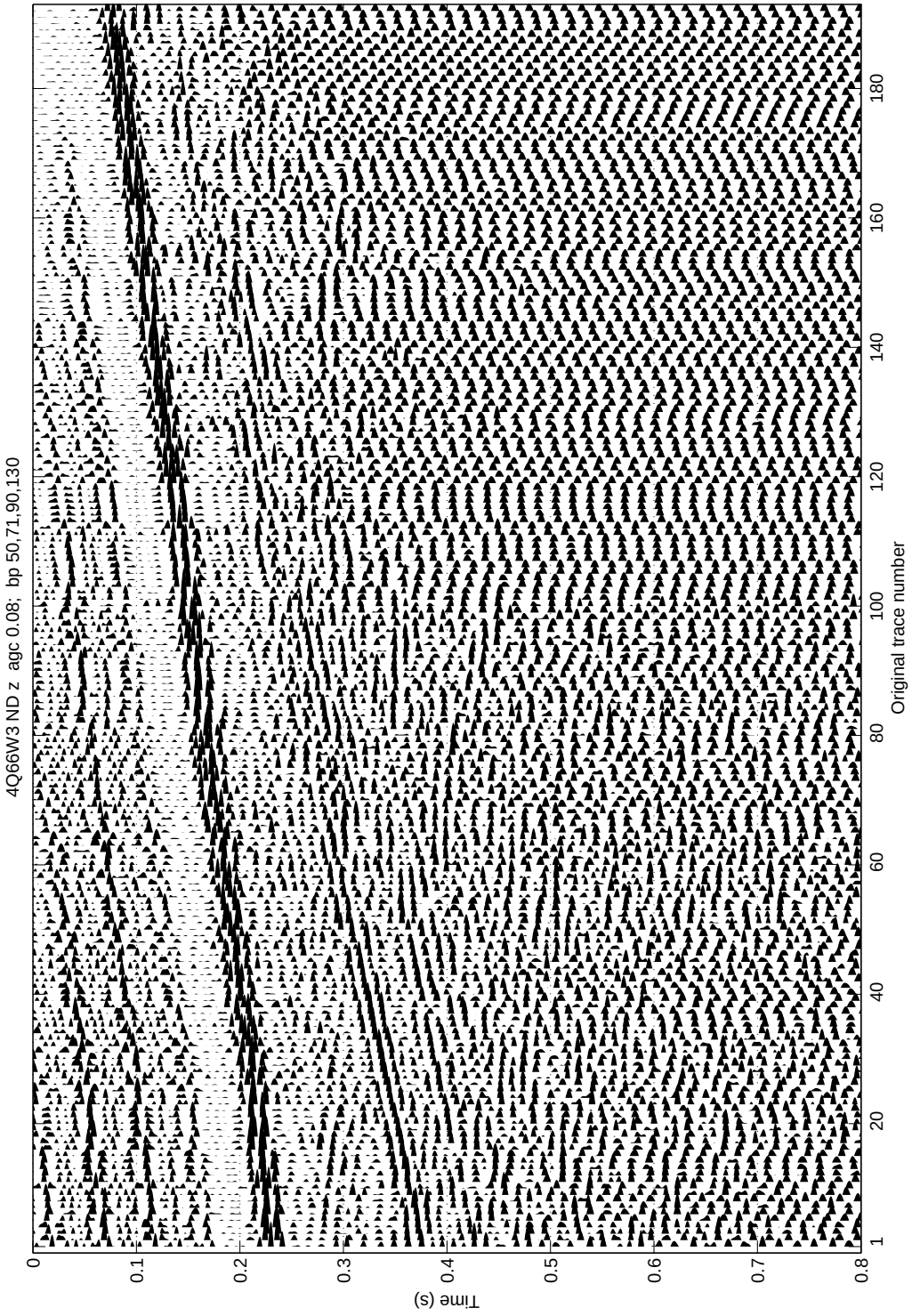

Figure A.24: A bandpass filter with the corner frequencies of 50, 71, 90, 130 Hz was applied.

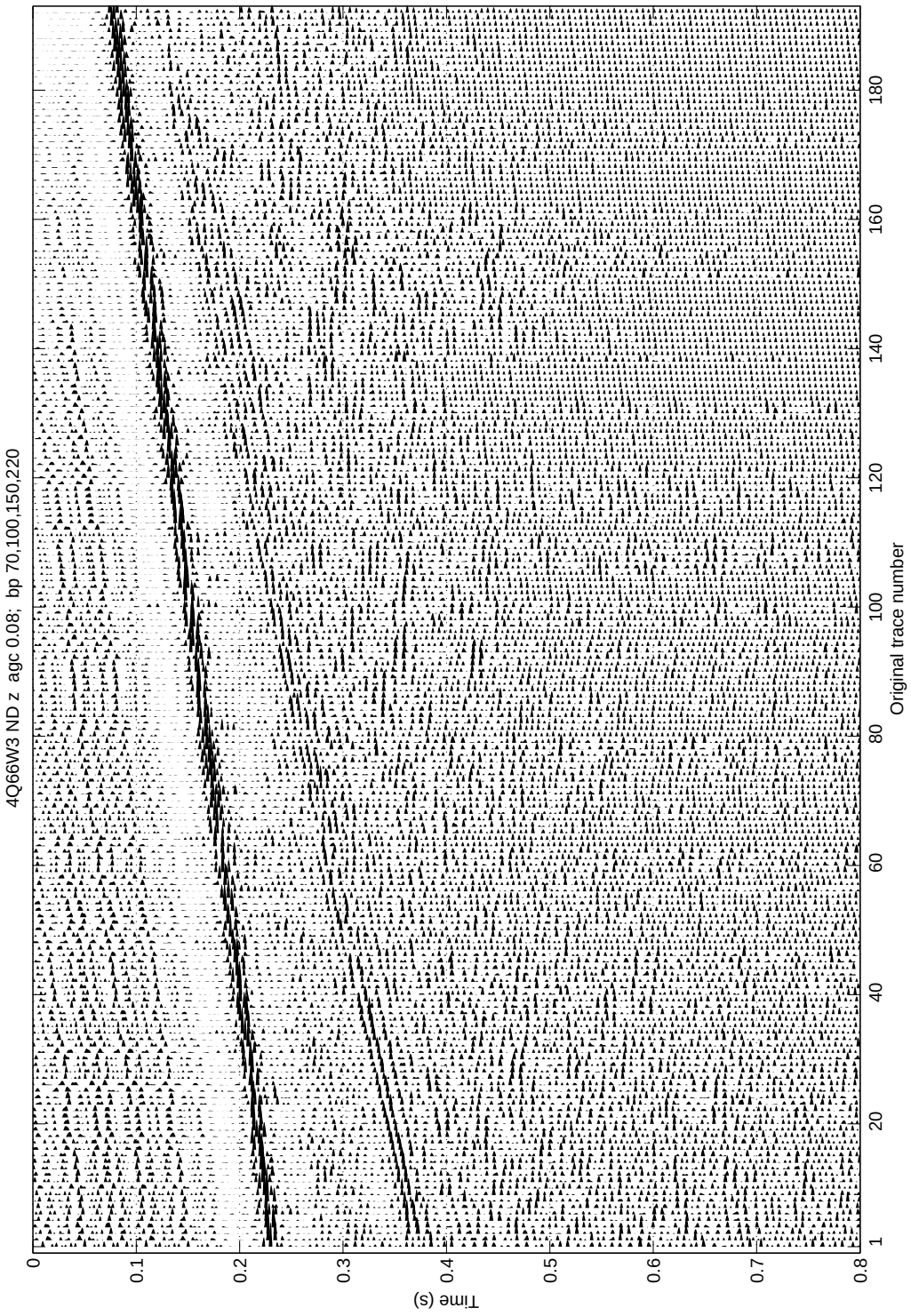

Figure A.25: A bandpass filter with the corner frequencies of 70, 100, 150, 220 Hz was applied.

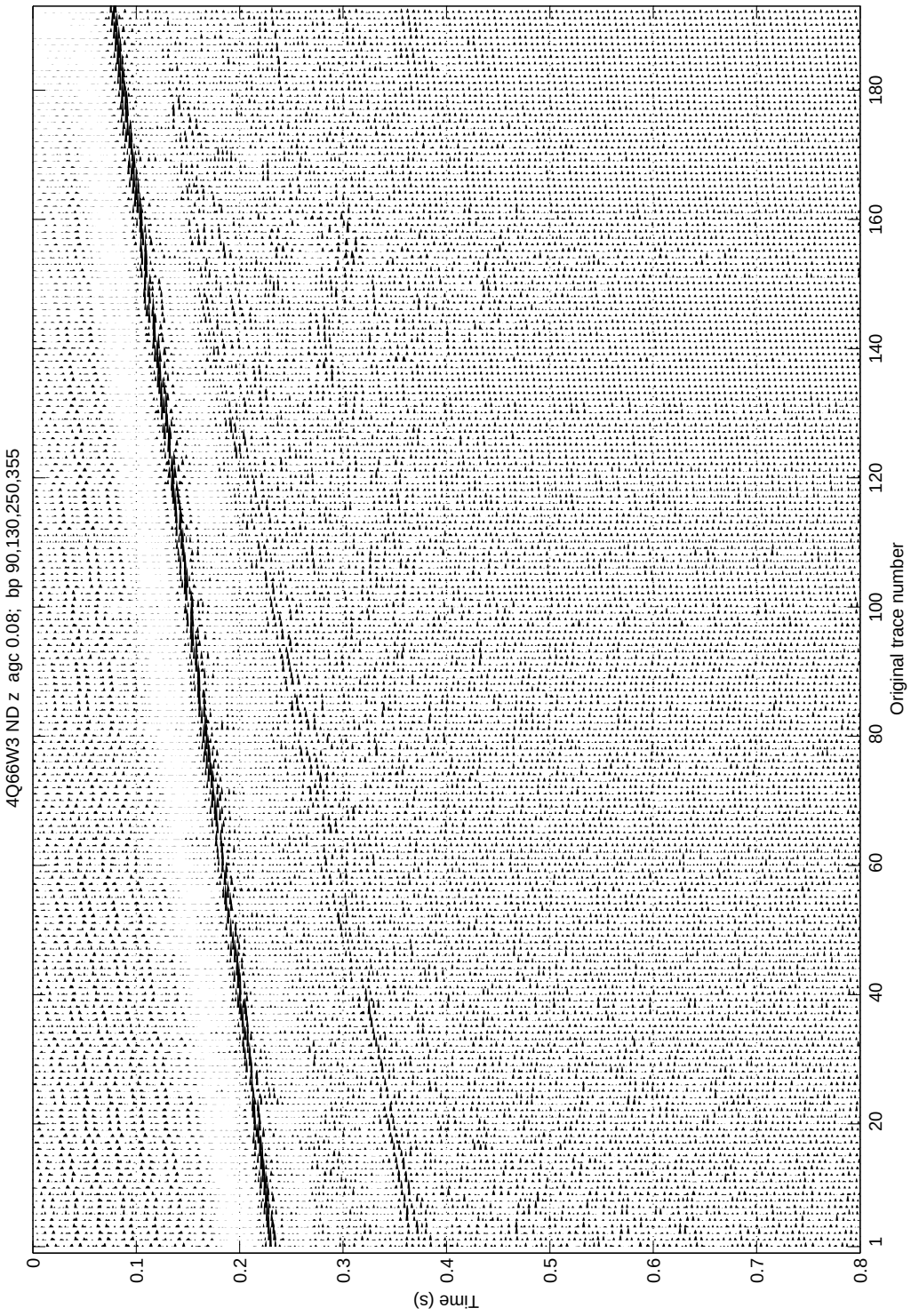

Figure A.26: A bandpass filter with the corner frequencies of 90, 130, 250, 355 Hz was applied.

A.5 Geological Logs

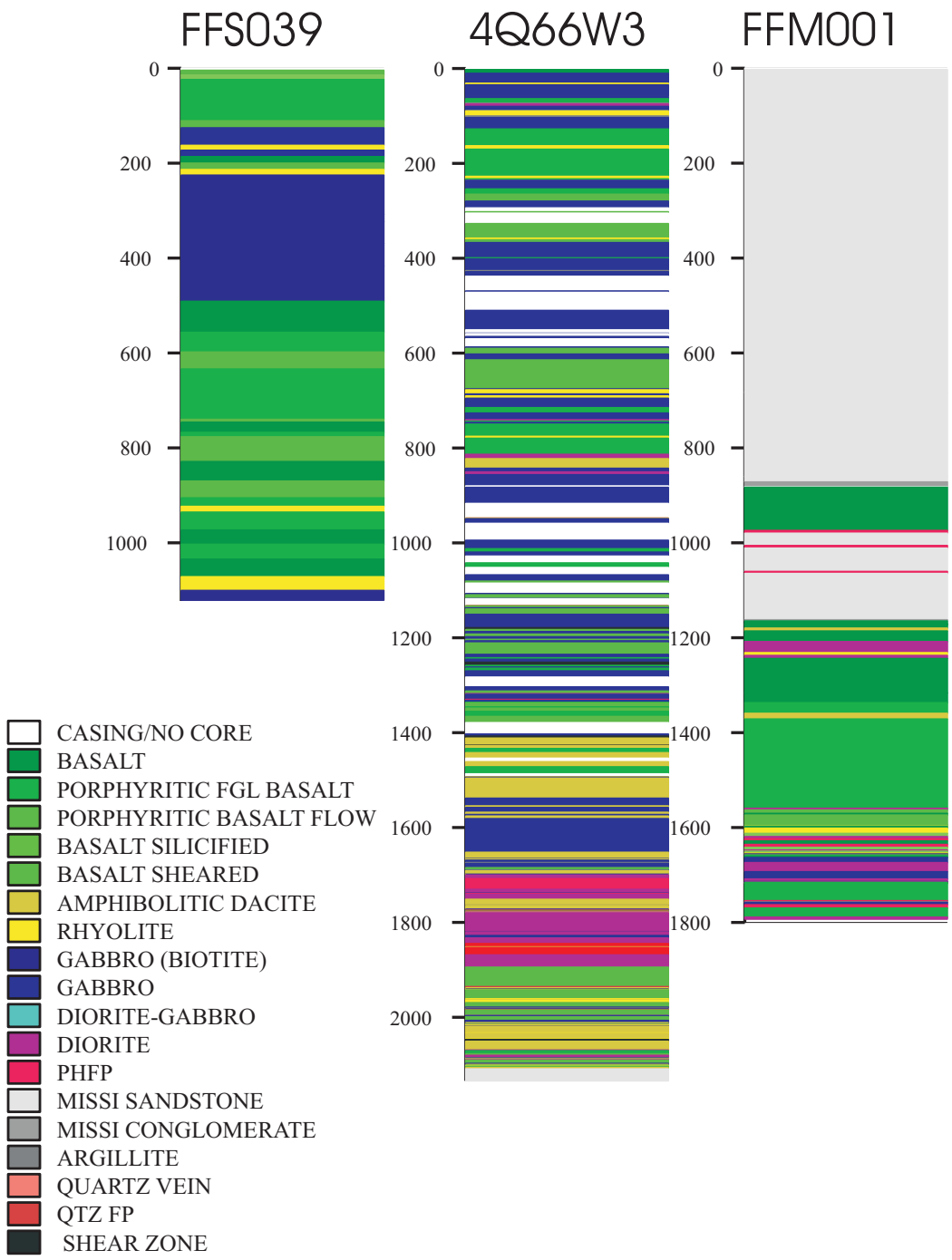

Figure A.27: Geological logs of all three boreholes.

Appendix B

Enlarged Figures of FFM001 near offset

B.1 Notch filtering

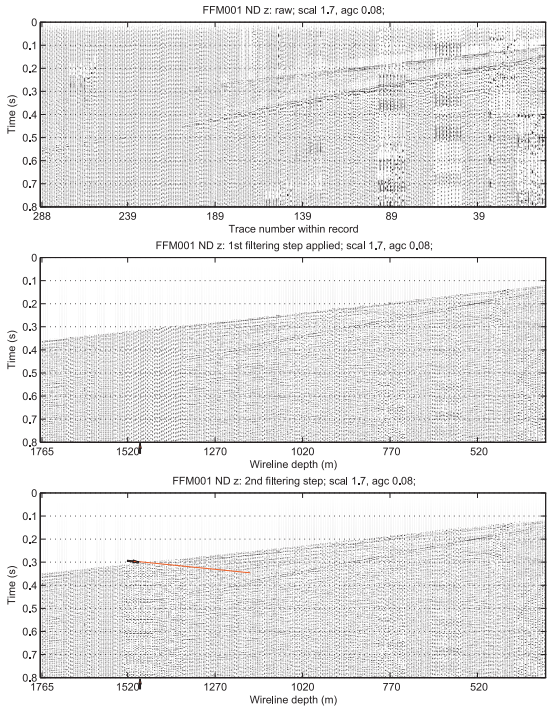

Figure B.1: The raw, scaled data are shown on top. The data after the regular, first notch filtering are displayed in the middle panel. The same data are shown after the second, additional notch filtering step on the bottom panel. The arrows mark a reflection that was not visible in the middle panel before this special, second notch filtering step.

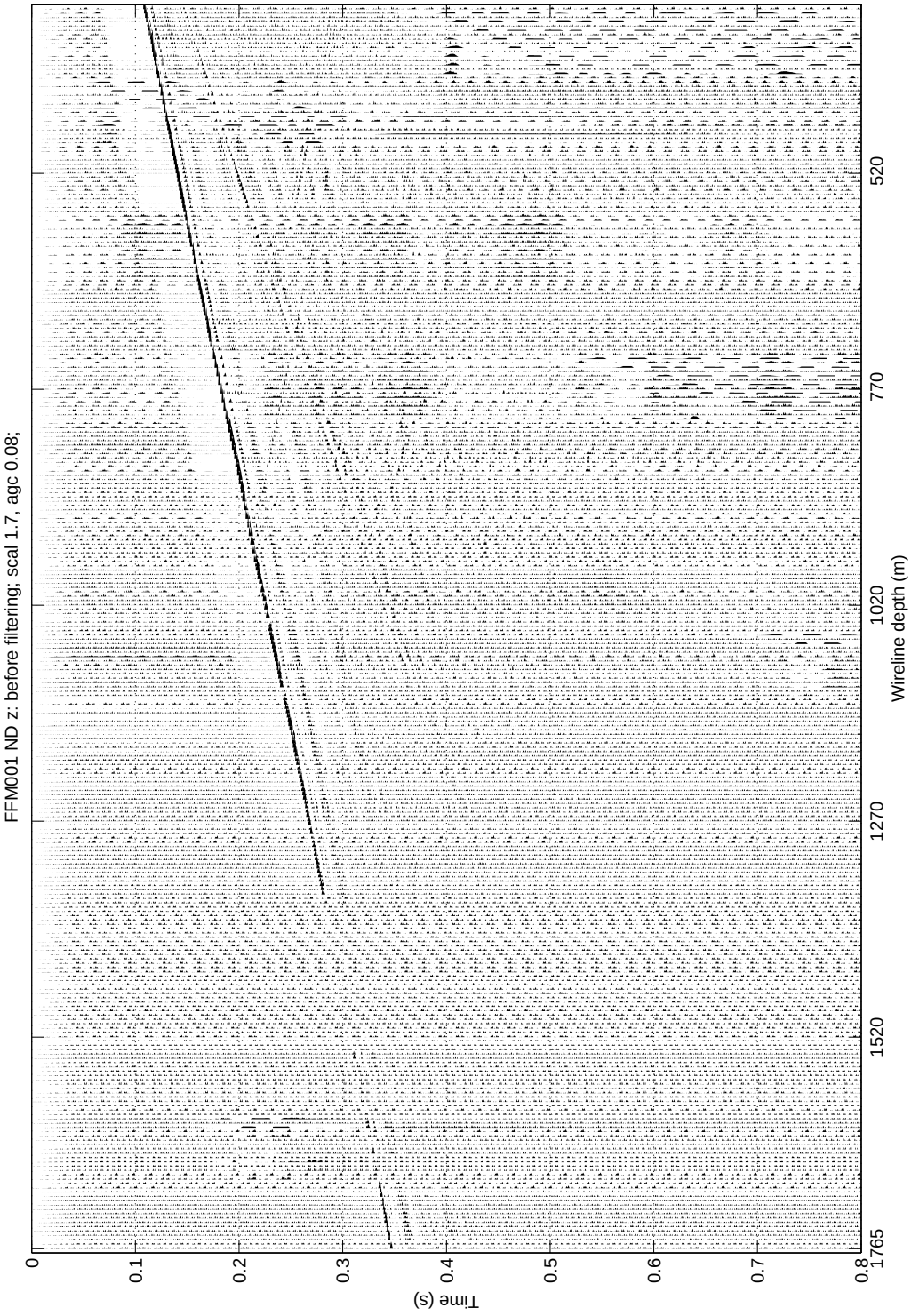

Figure B.2: Raw, but scaled data.

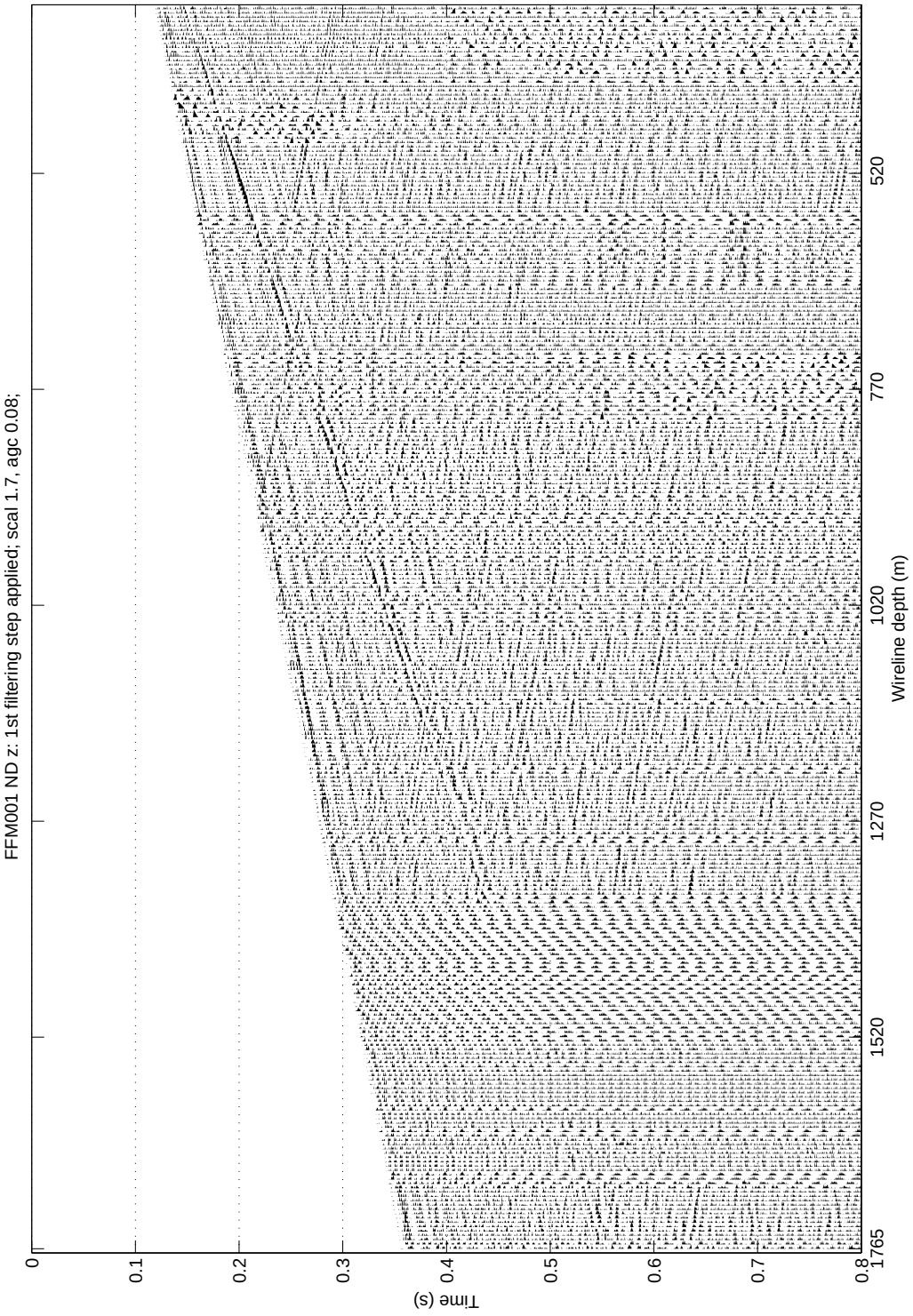

Figure B.3: Dta after the first notch filtering step applied.

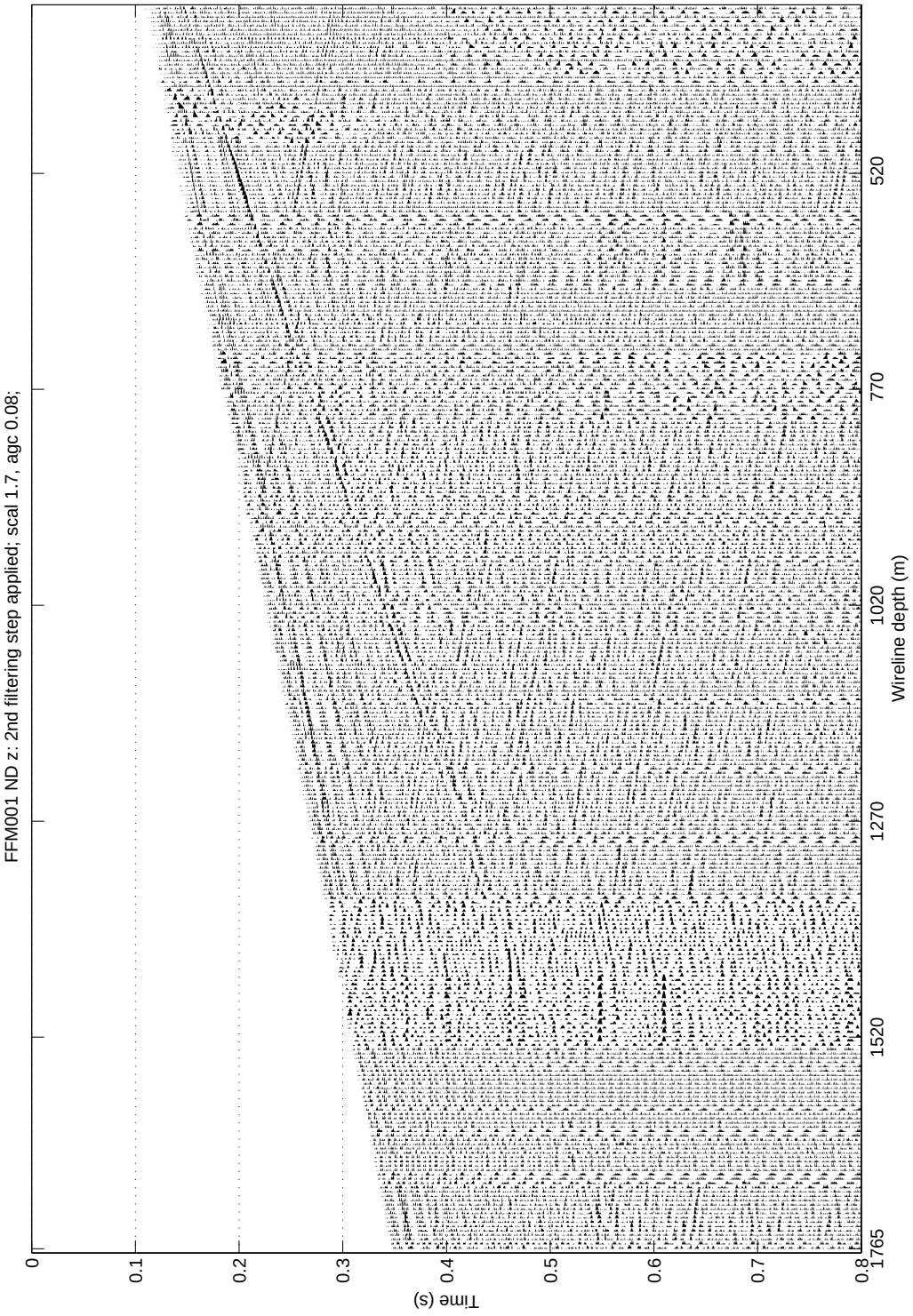

Figure B.4: Data after the second notch filtering step applied.

B.2 Removal of P-Wave Energy

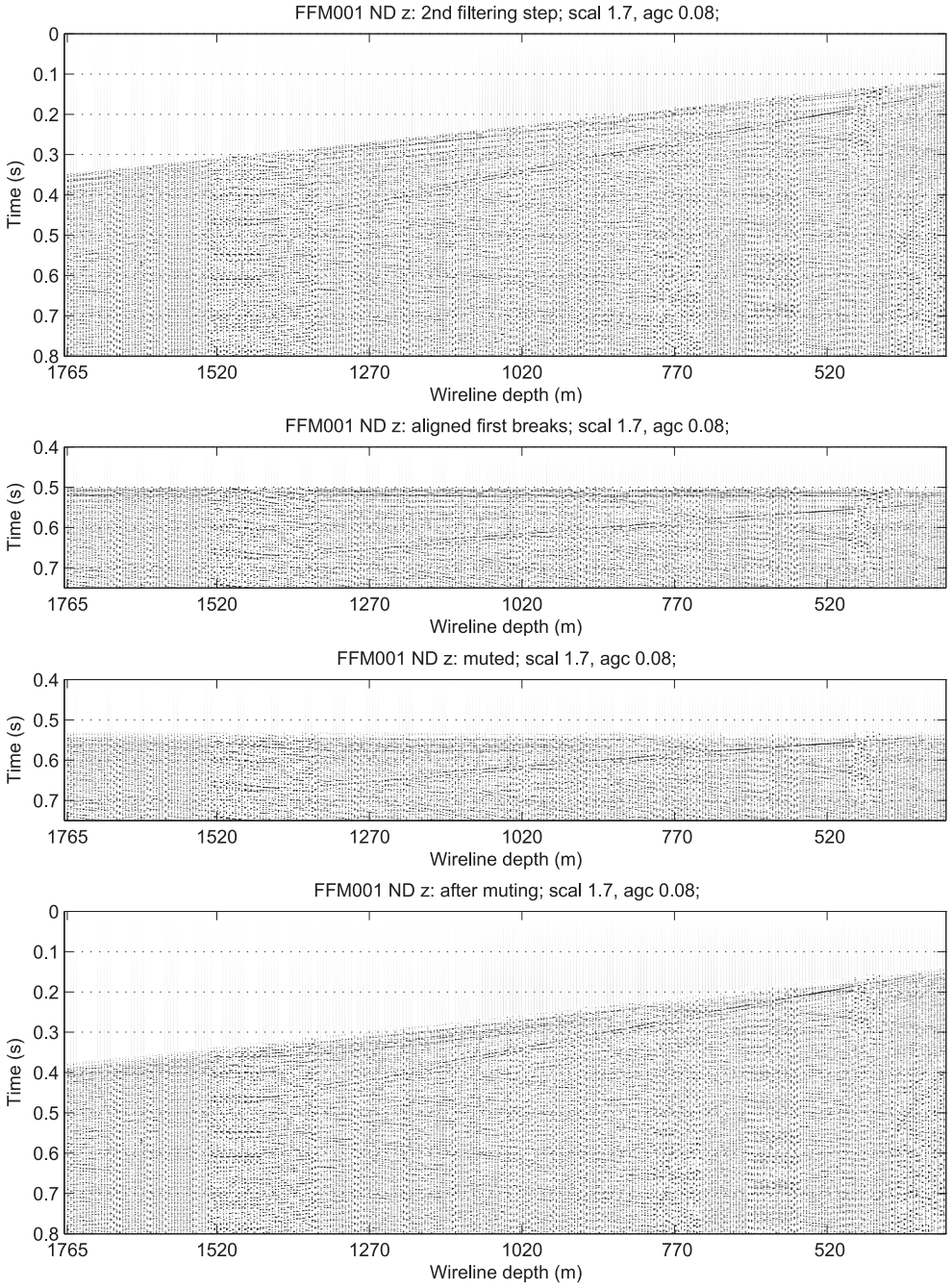

Figure B.5: The second step of the P-wave energy removal is shown. Top: Data after first top mute and filtering; second from top: first break aligned data; muted data; bottom: after data is shifted back.

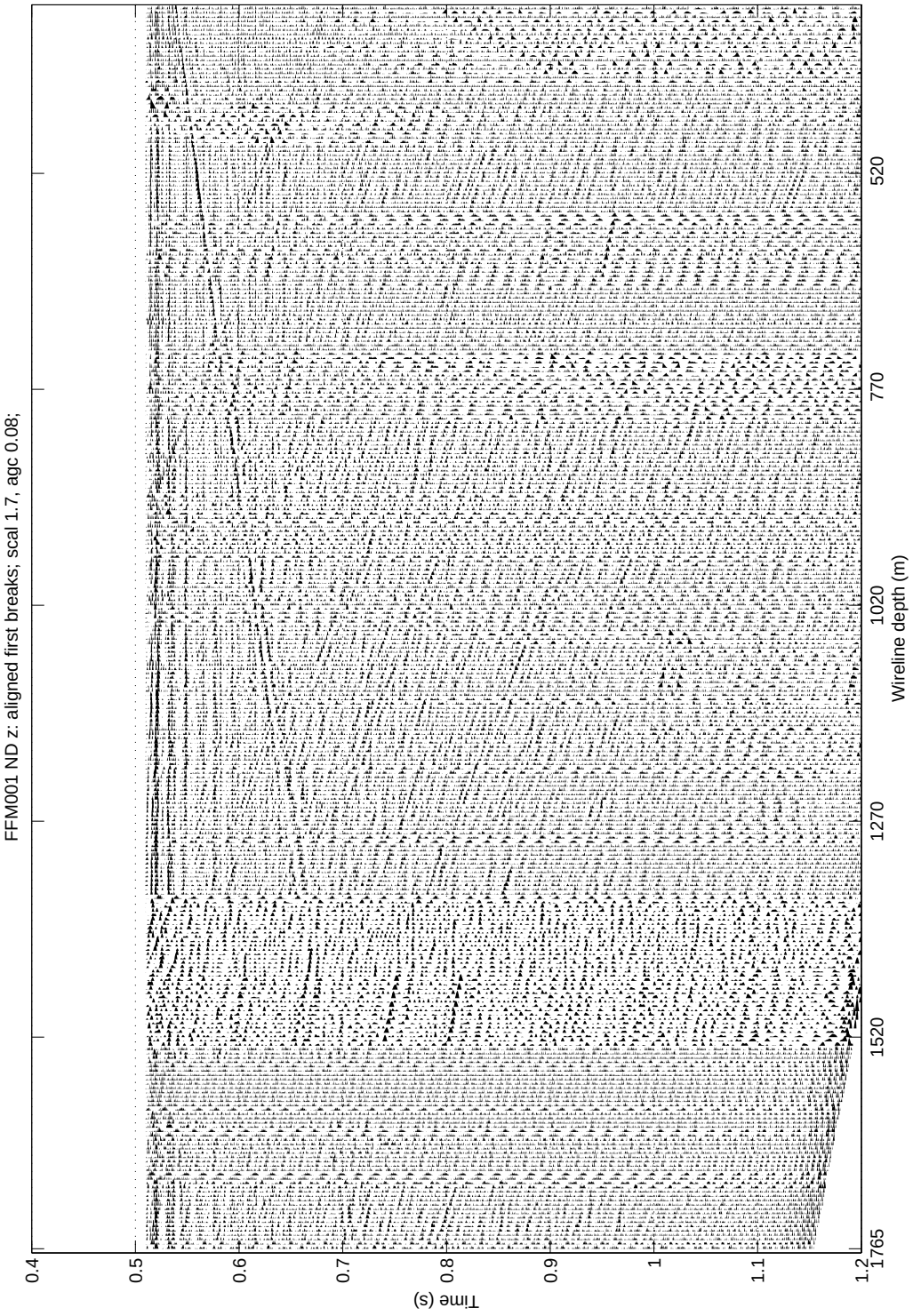

Figure B.6: Data aligned by the first P-wave arrivals.

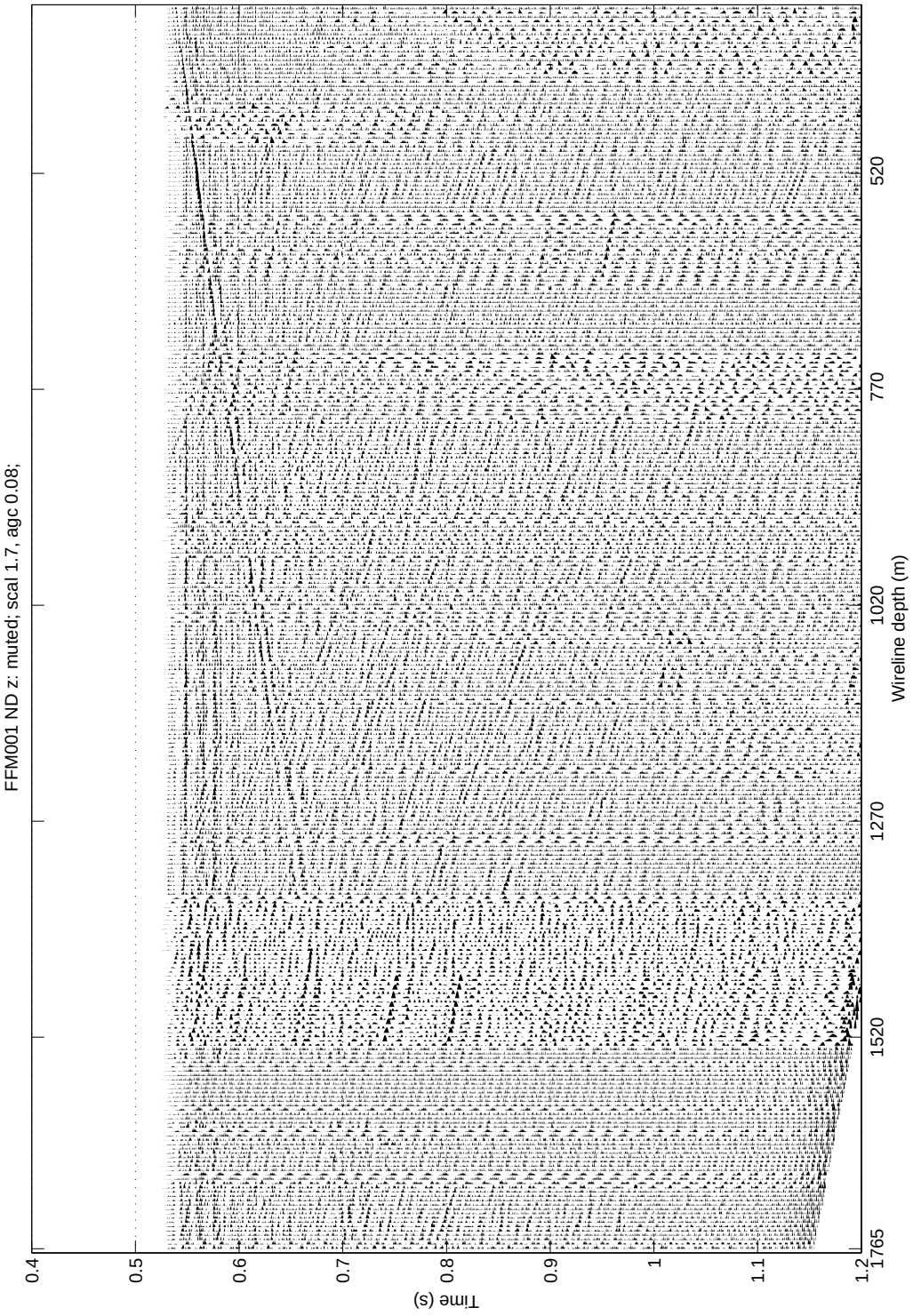

Figure B.7: Data after top muting.

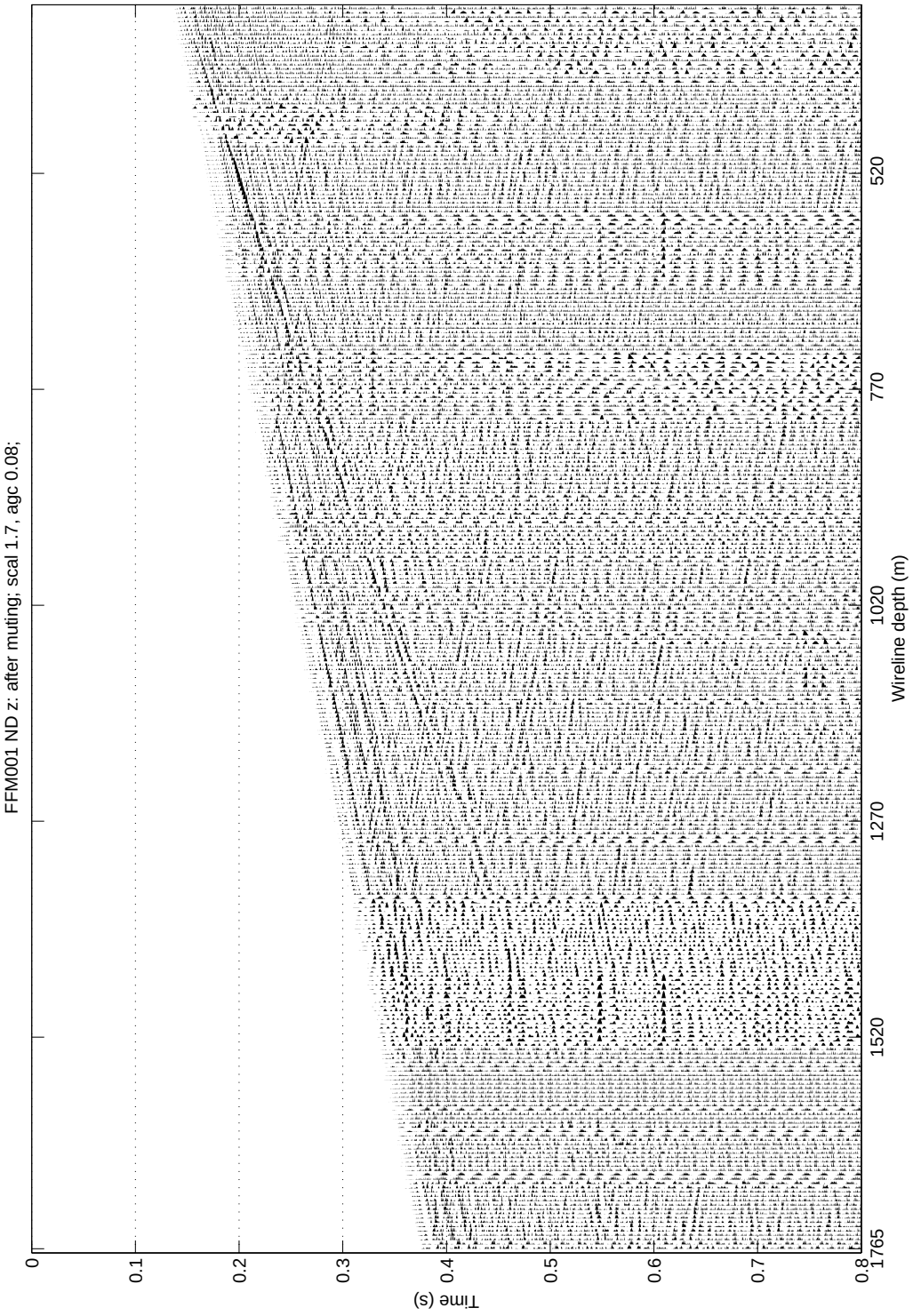

Figure B.8: Data is shifted back.

B.3 Removal of S-Wave Energy

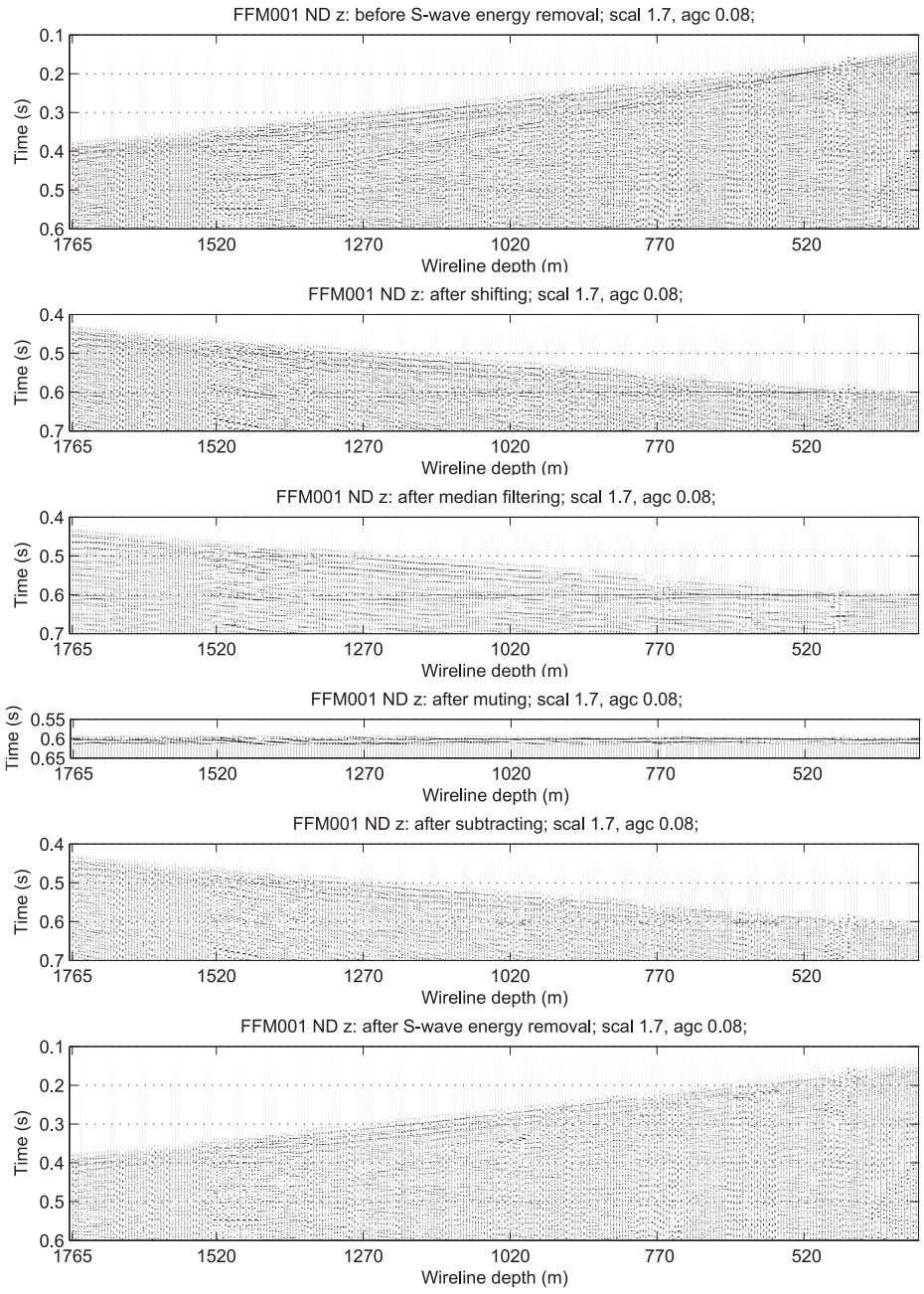

Figure B.9: Removal of the S-wave energy. From top to bottom: The input data are shown; the data are shifted to align the first arrival times of the S-waves; median filtered data; except a narrow window all filtered data are muted; the small window is subtracted from the aligned data; the data are shifted back.

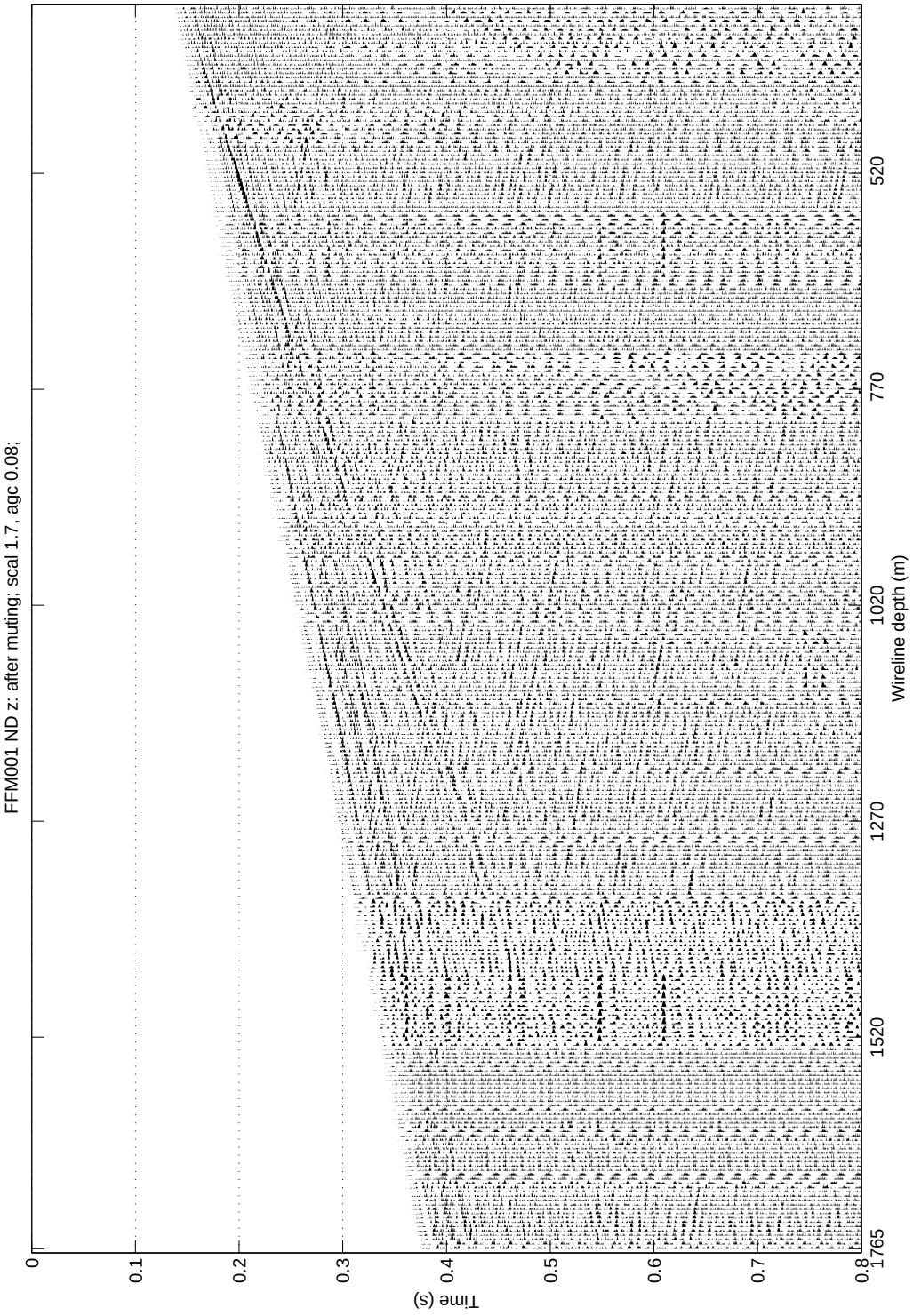

Figure B.10: P removed data.

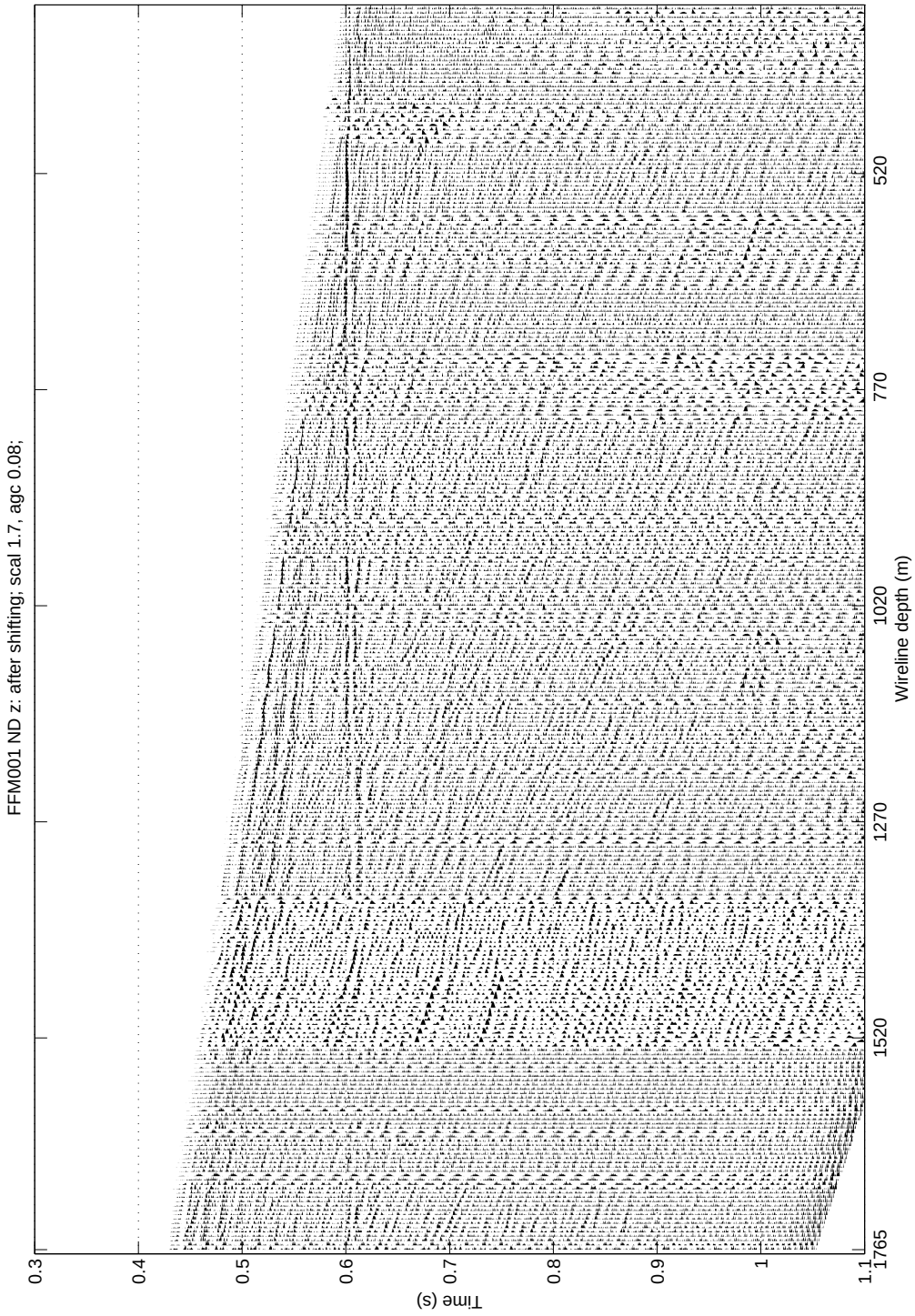

Figure B.11: Data with time added at the beginning of each trace and S-wave arrival time is lined up horizontally.

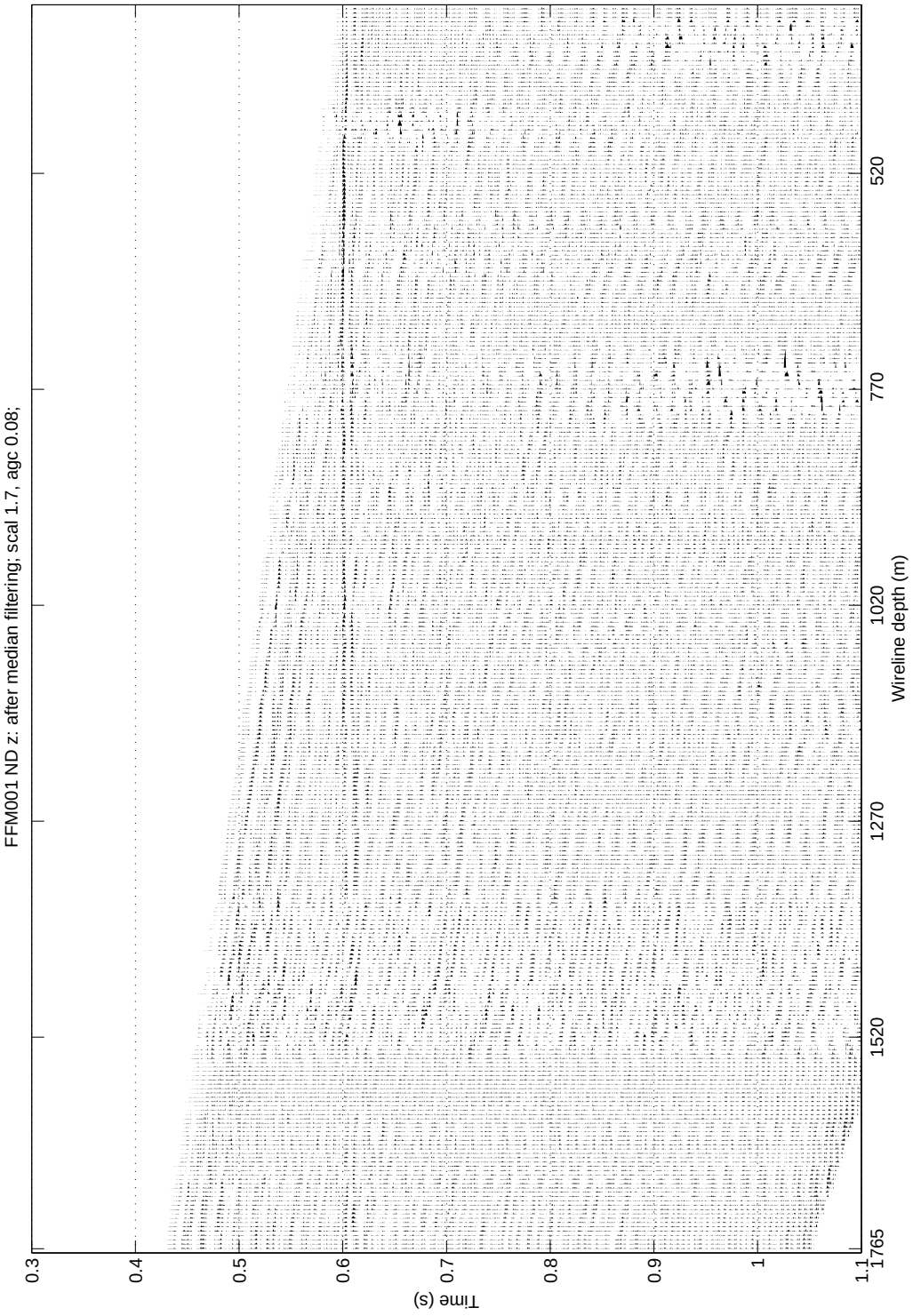

Figure B.12: A median filter of 11 traces width was applied on data.

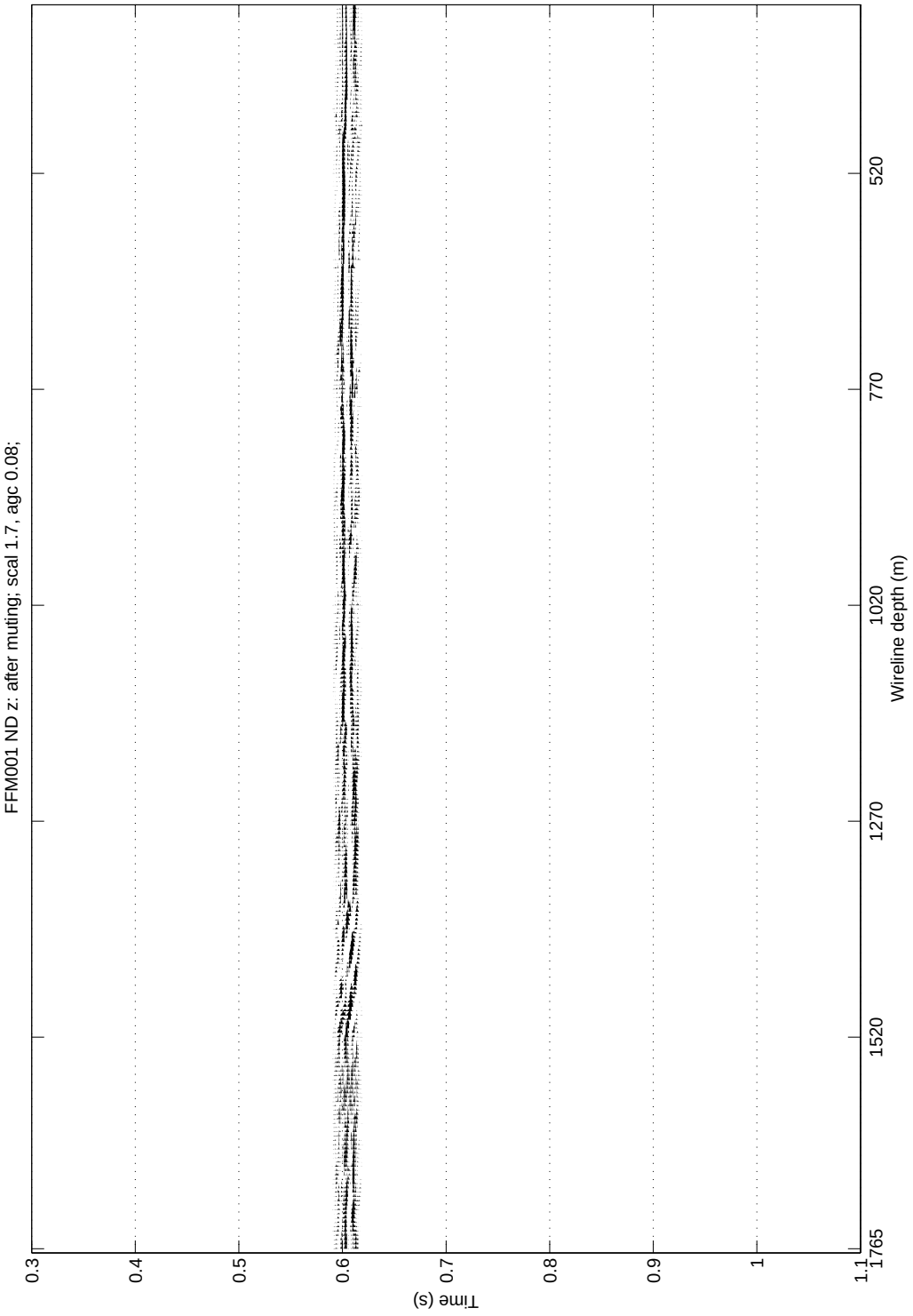

Figure B.13: Data up to the S arrival time are muted.

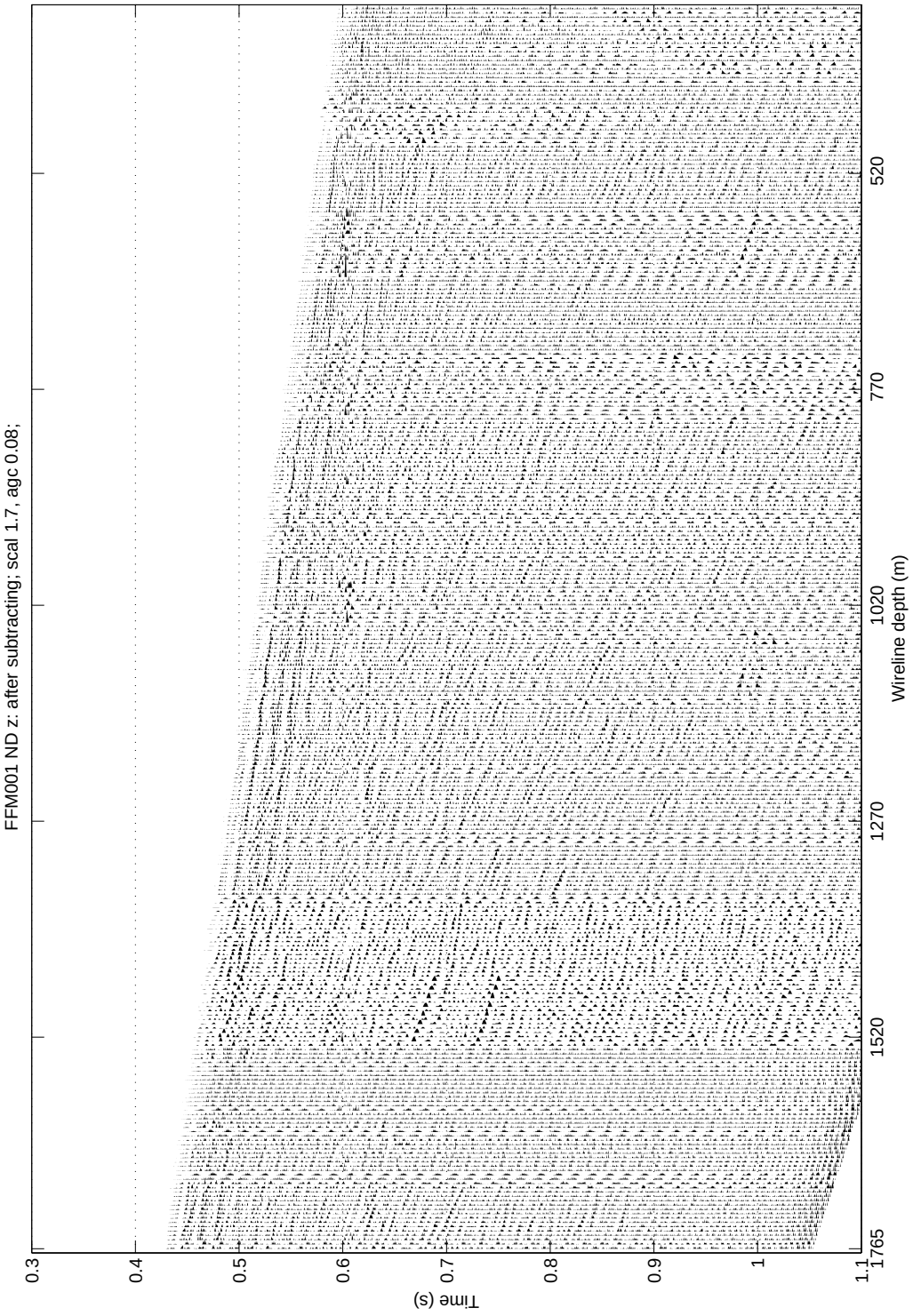

Figure B.14: The median filtered data is subtracted from the shifted data.

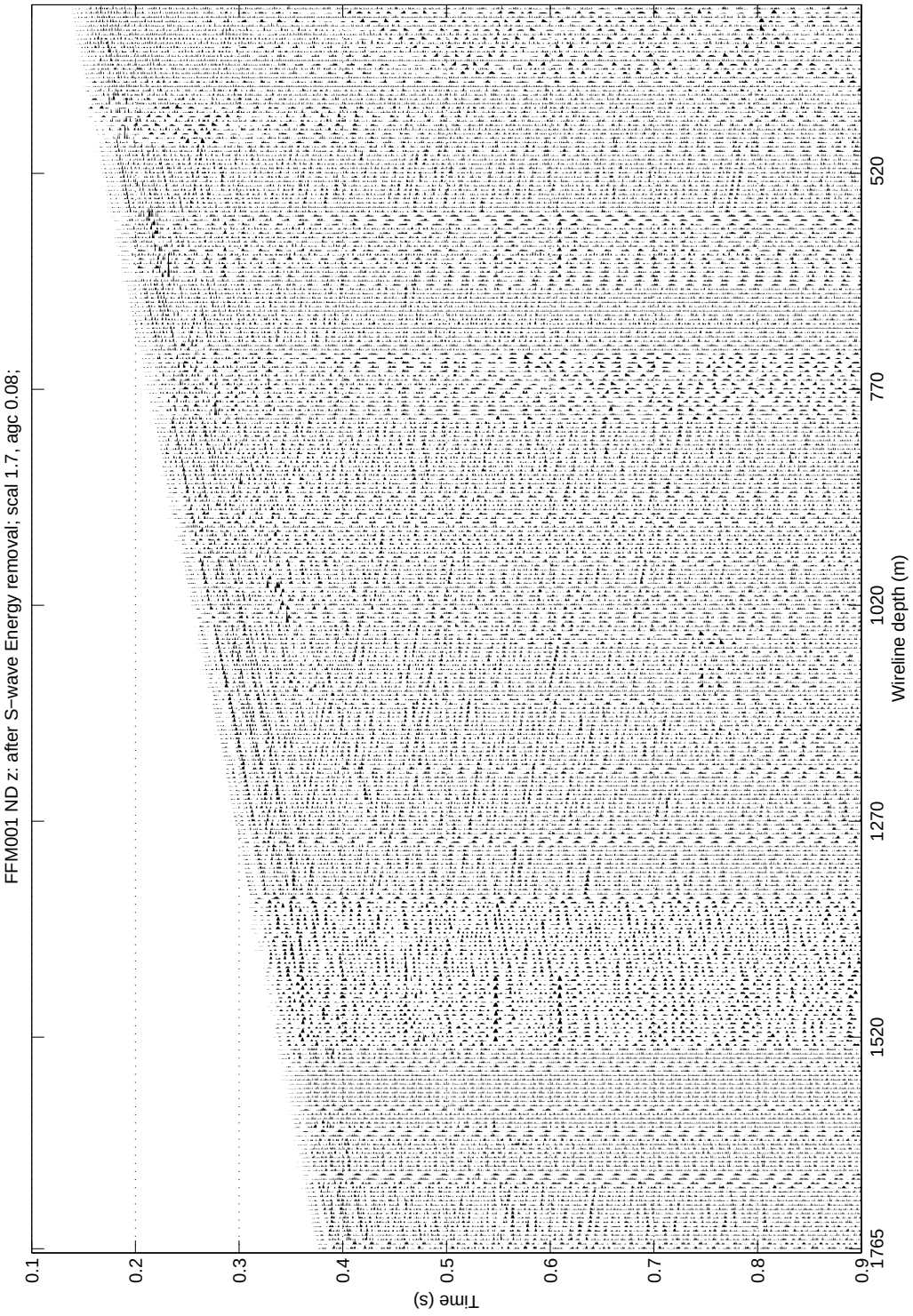

Figure B.15: The added time is removed and the data is shifted back.

Appendix C

Enlarged Figures of FFS039 near offset

C.1 Notch filtering

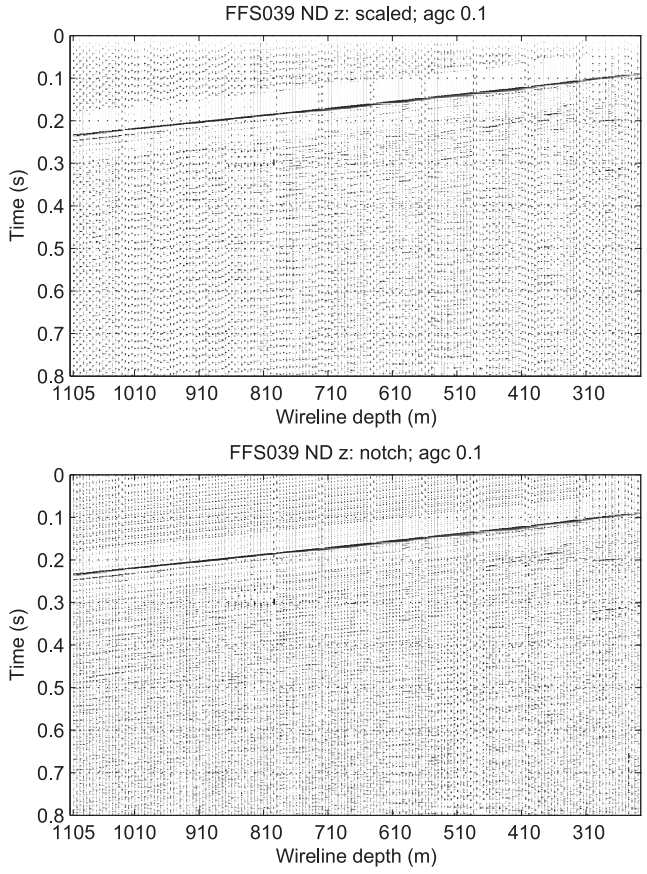

Figure C.1: The raw, but scaled data is shown on top. Below is the scaled, notch filtered dataset.

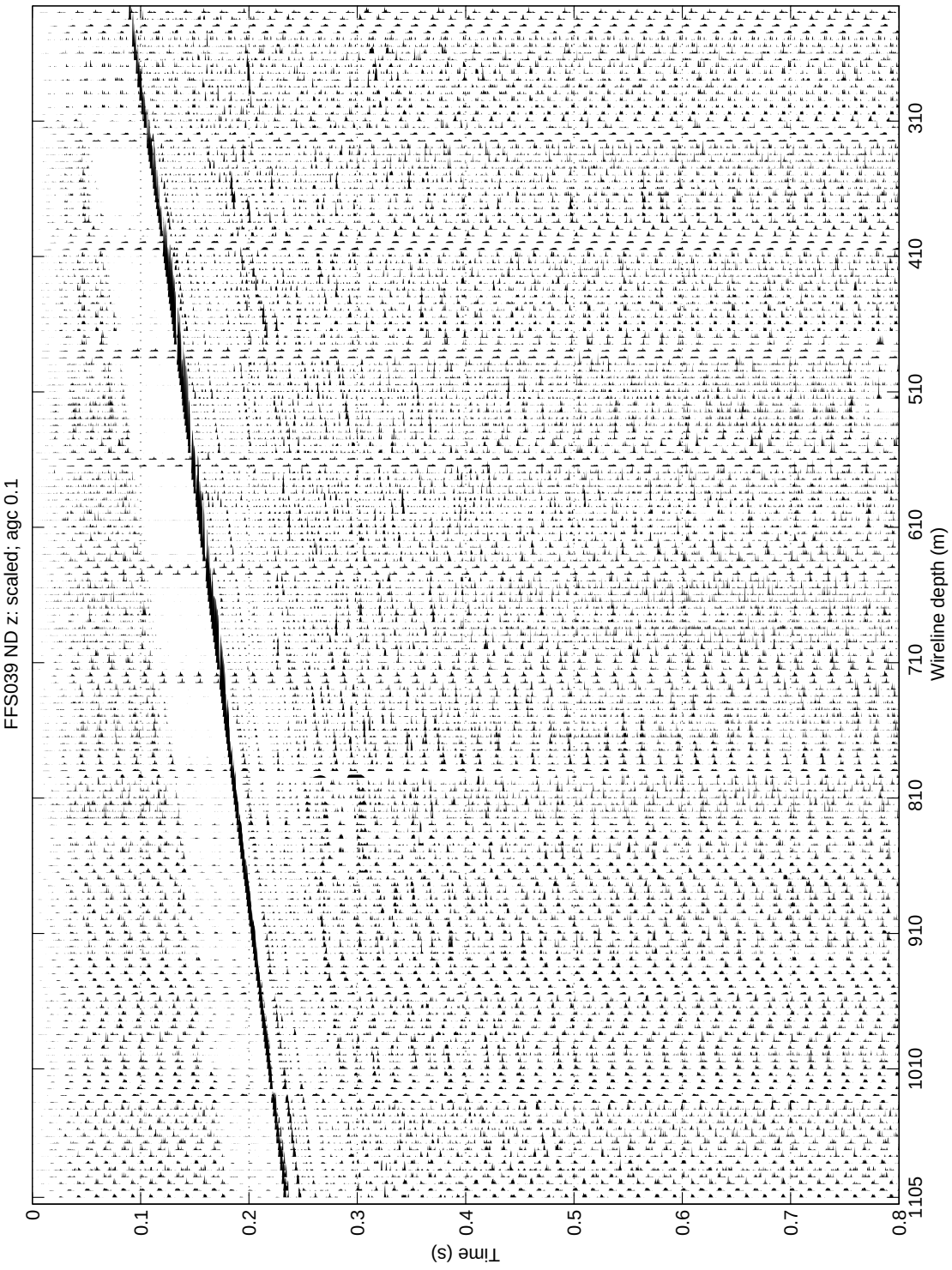

Figure C.2: Raw, but scaled data.

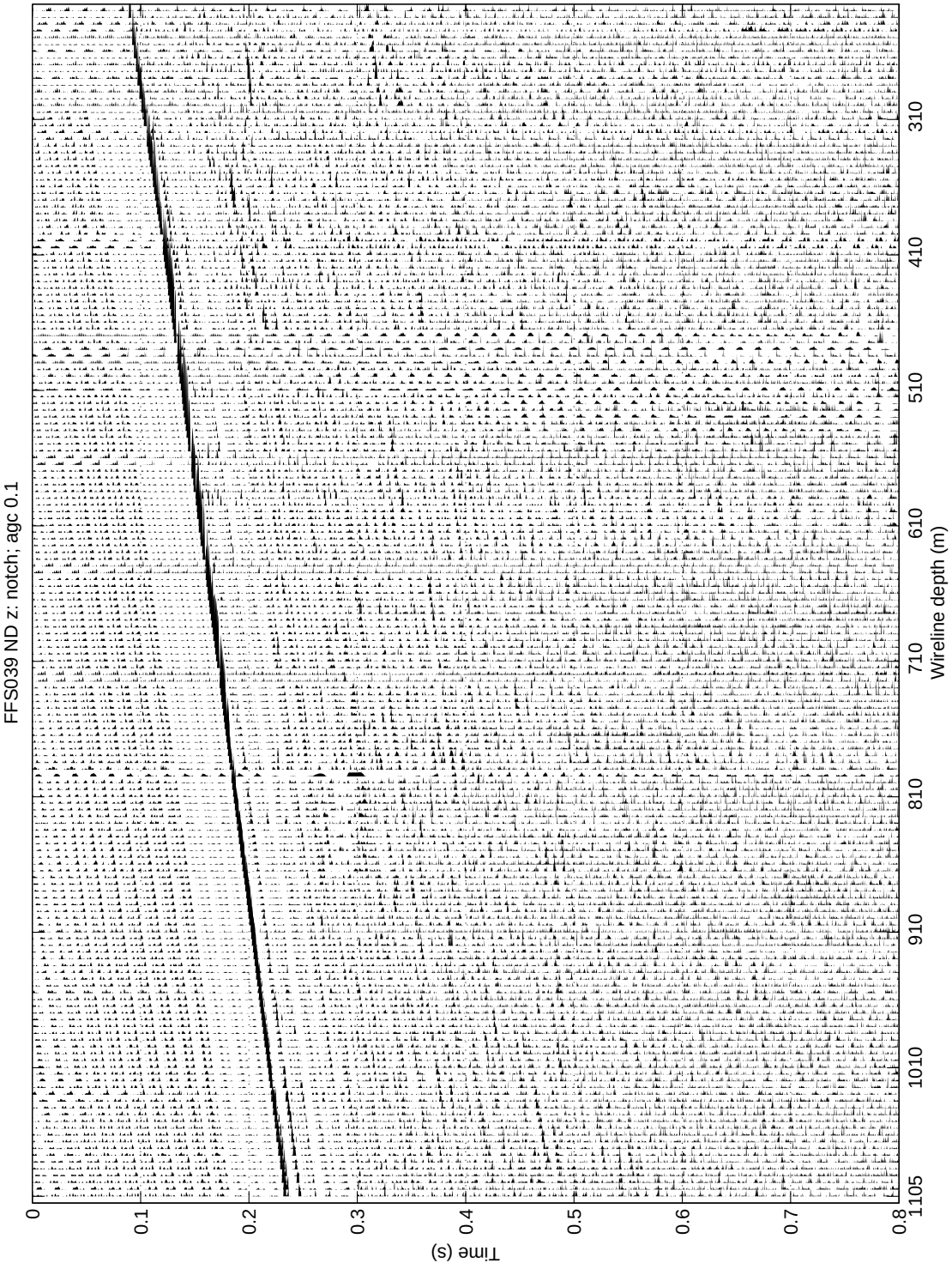

Figure C.3: Data after the notch filtering step applied.

C.2 Removal of P-Wave Energy

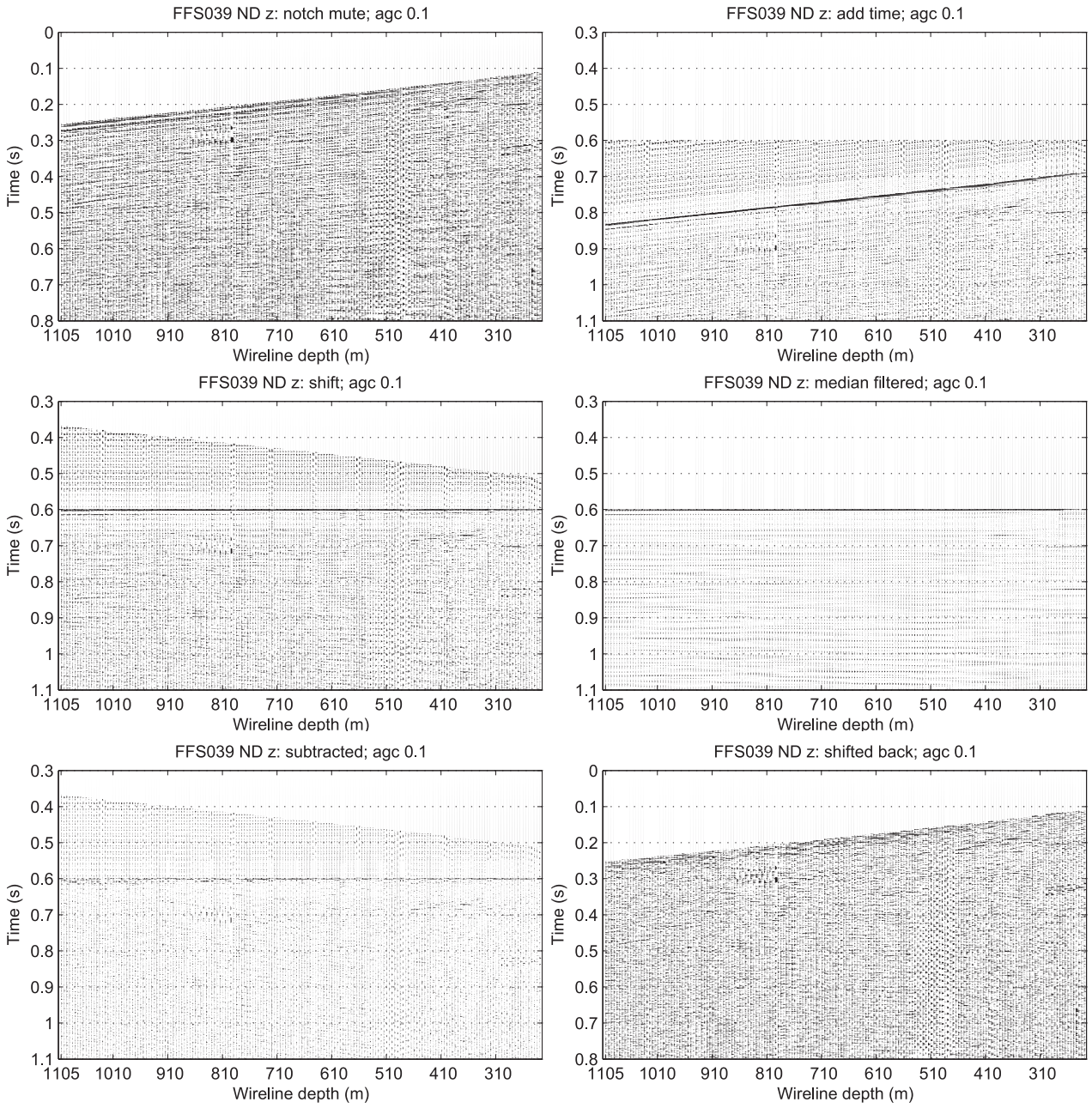

Figure C.4: Removal of the downgoing P-wave energy. From top left to bottom right: notch filtered data; 0.6 s time added at each trace beginning; shifted by picked P-arrival times; median filtered data; data after subtraction of the median filtered data; shifted back.

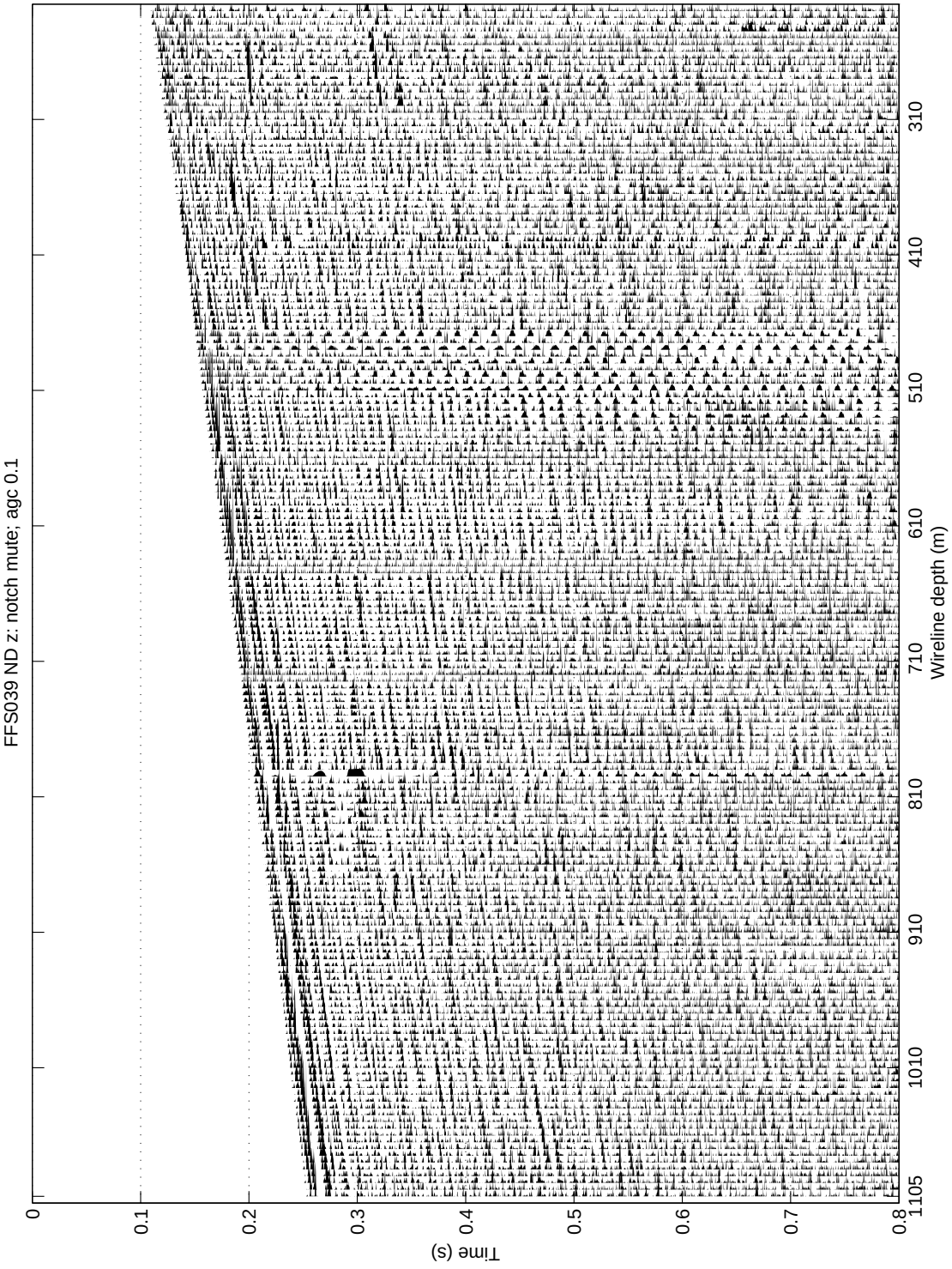

Figure C.5: Notch filtered and top muted data.

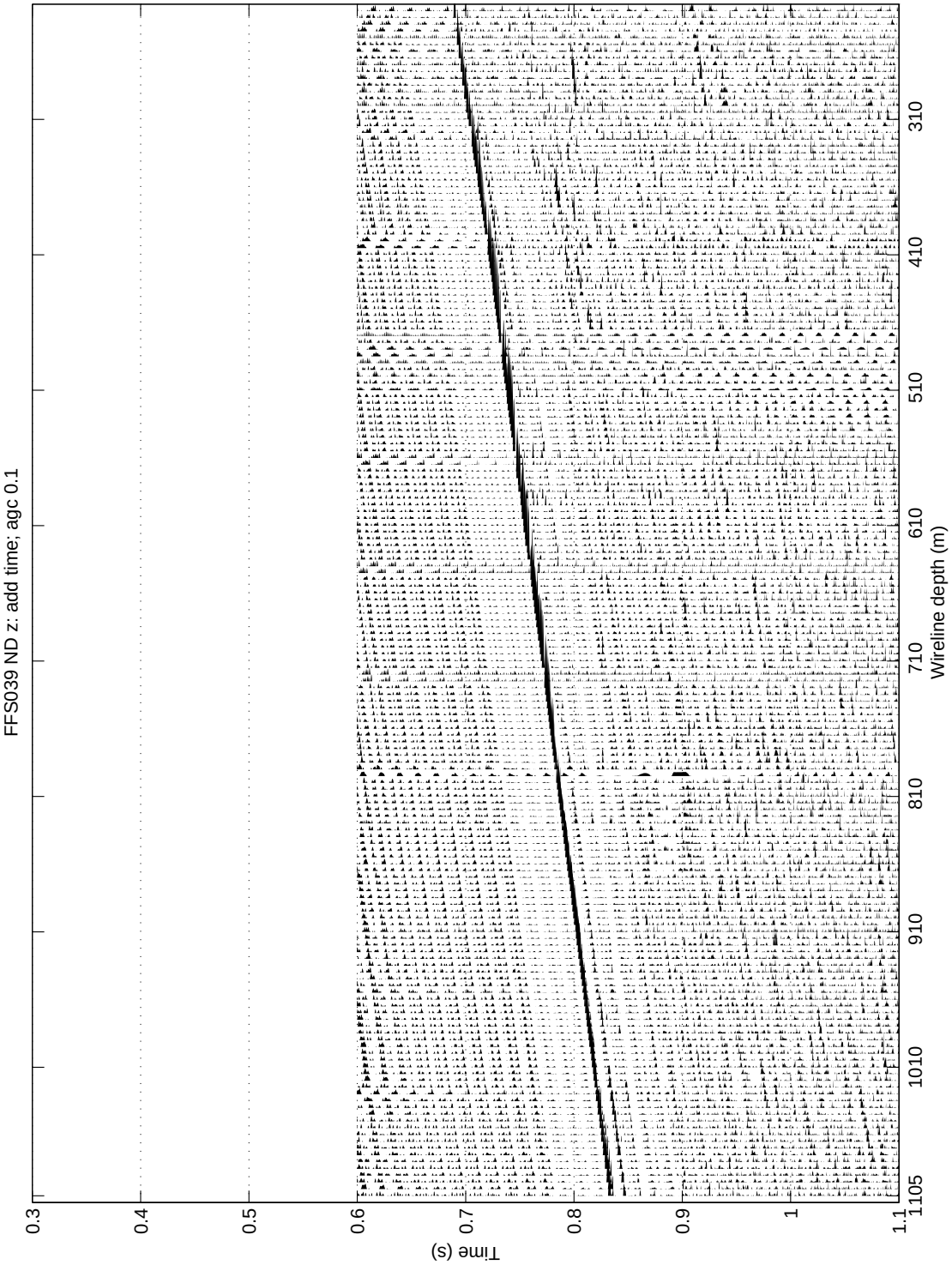

Figure C.6: Data with 0.6 s added to the trace beginnings.

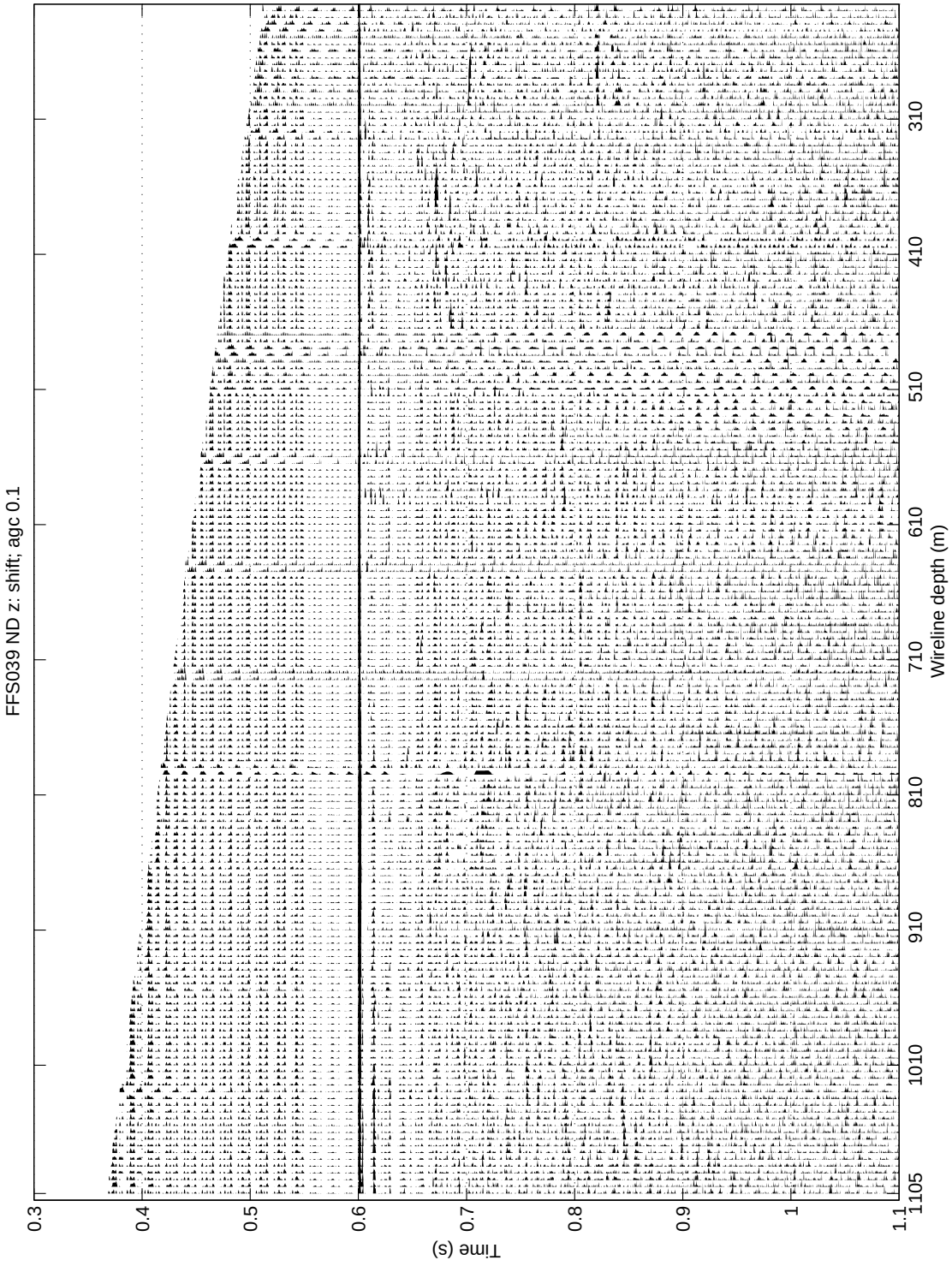

Figure C.7: Data aligned by the first P-wave arrivals.

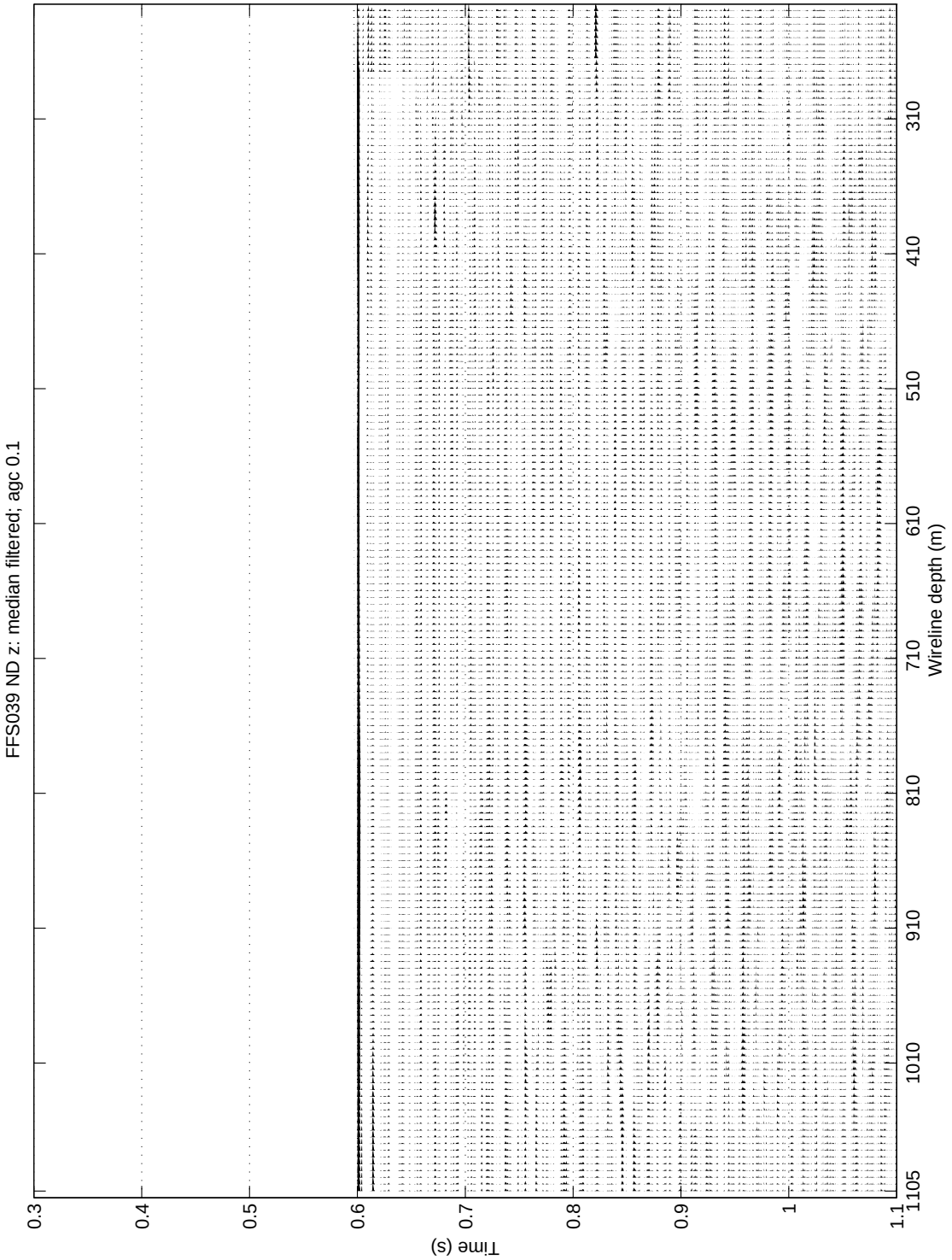

Figure C.8: Data after median filtering.

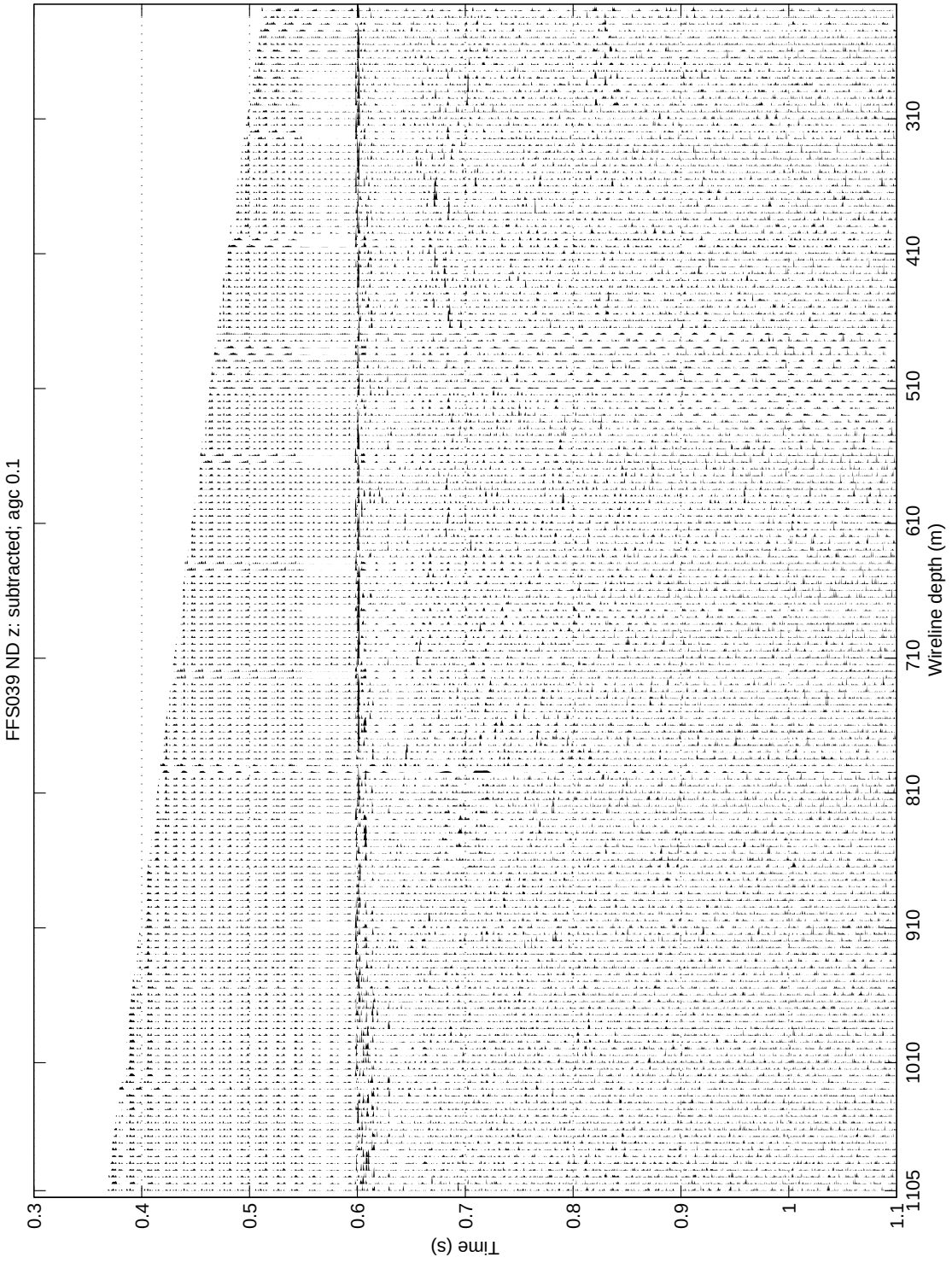

Figure C.9: Data after subtraction of the median filtered data from the shifted data.

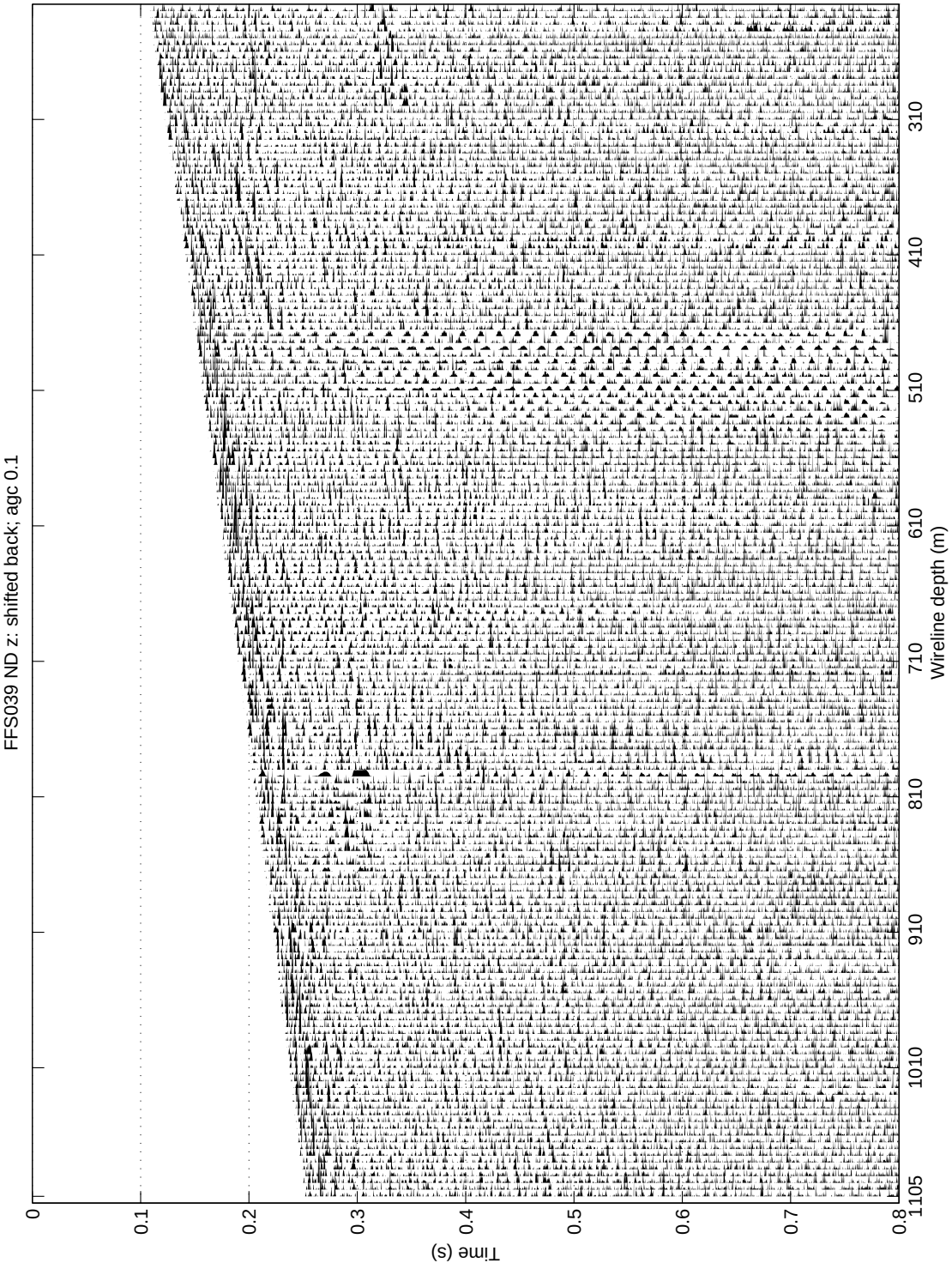

Figure C.10: Data is shifted back.

C.3 Removal of S-Wave Energy

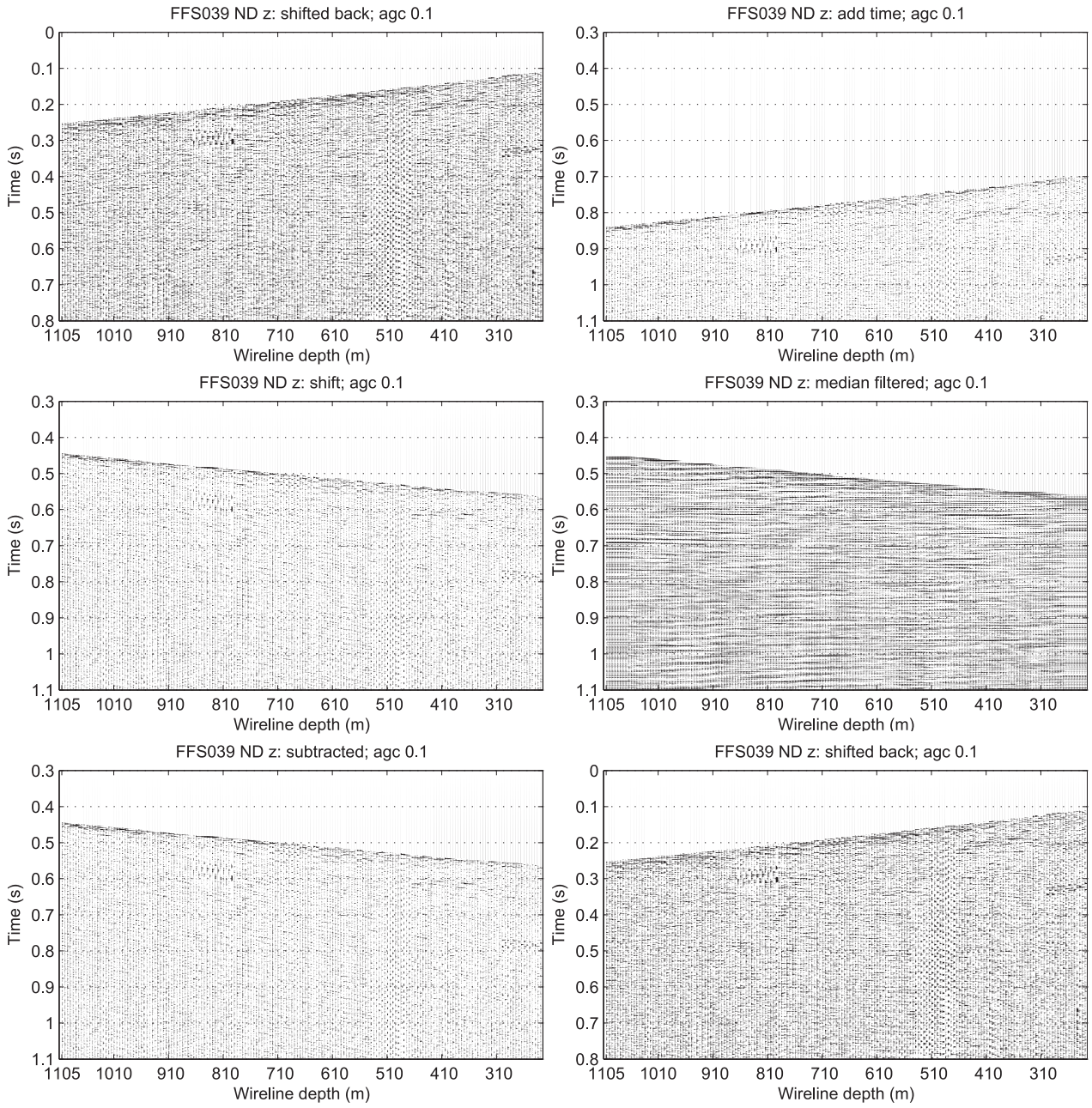

Figure C.11: Removal of the downgoing S-wave energy. From top left to bottom right: P-removed data; 0.6 s time added at each trace beginning; shifted by picked S-arrival times; median filtered data; data after subtraction of the median filtered data; shifted back.

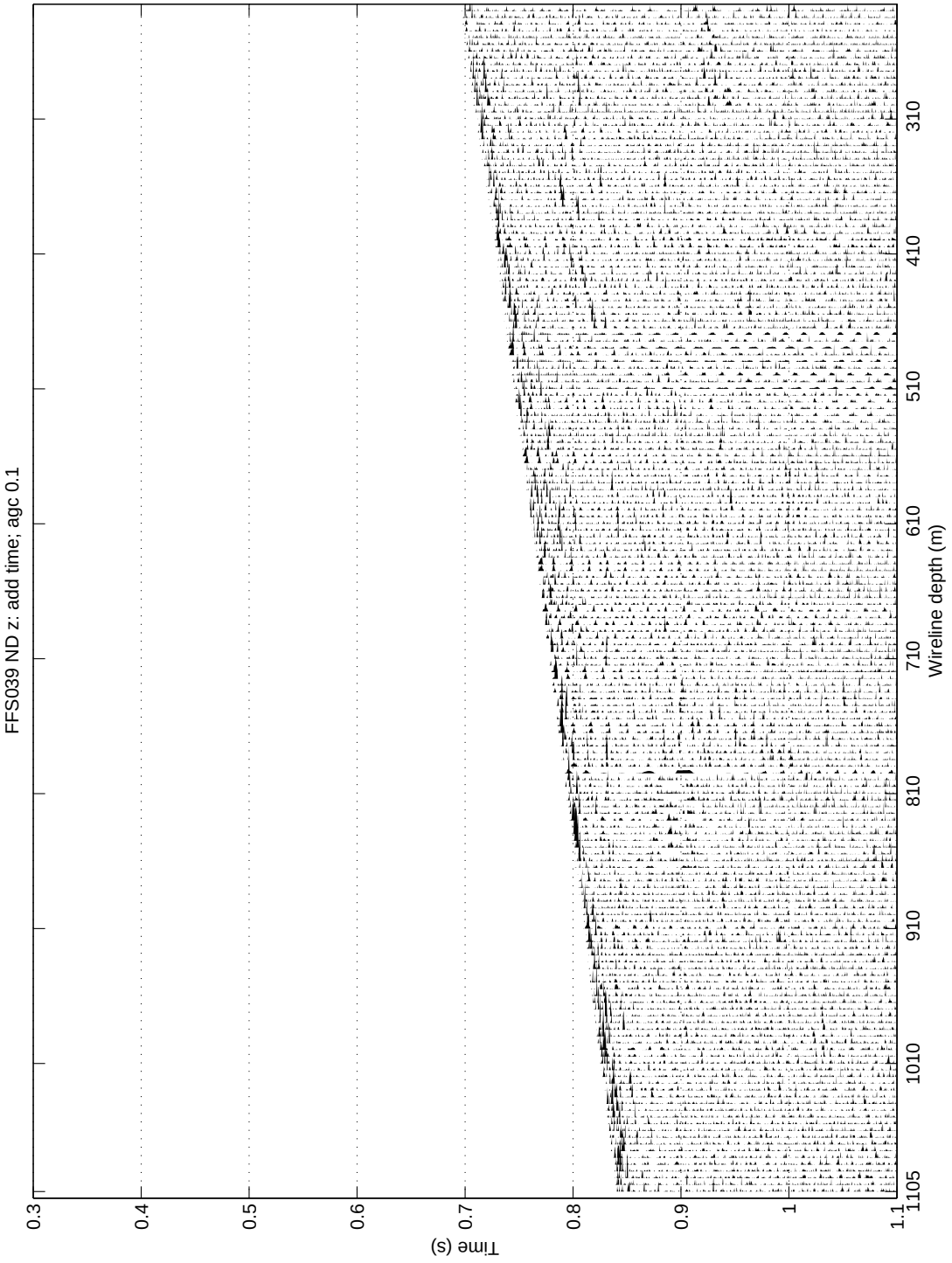

Figure C.12: Time is added at the beginning of each trace.

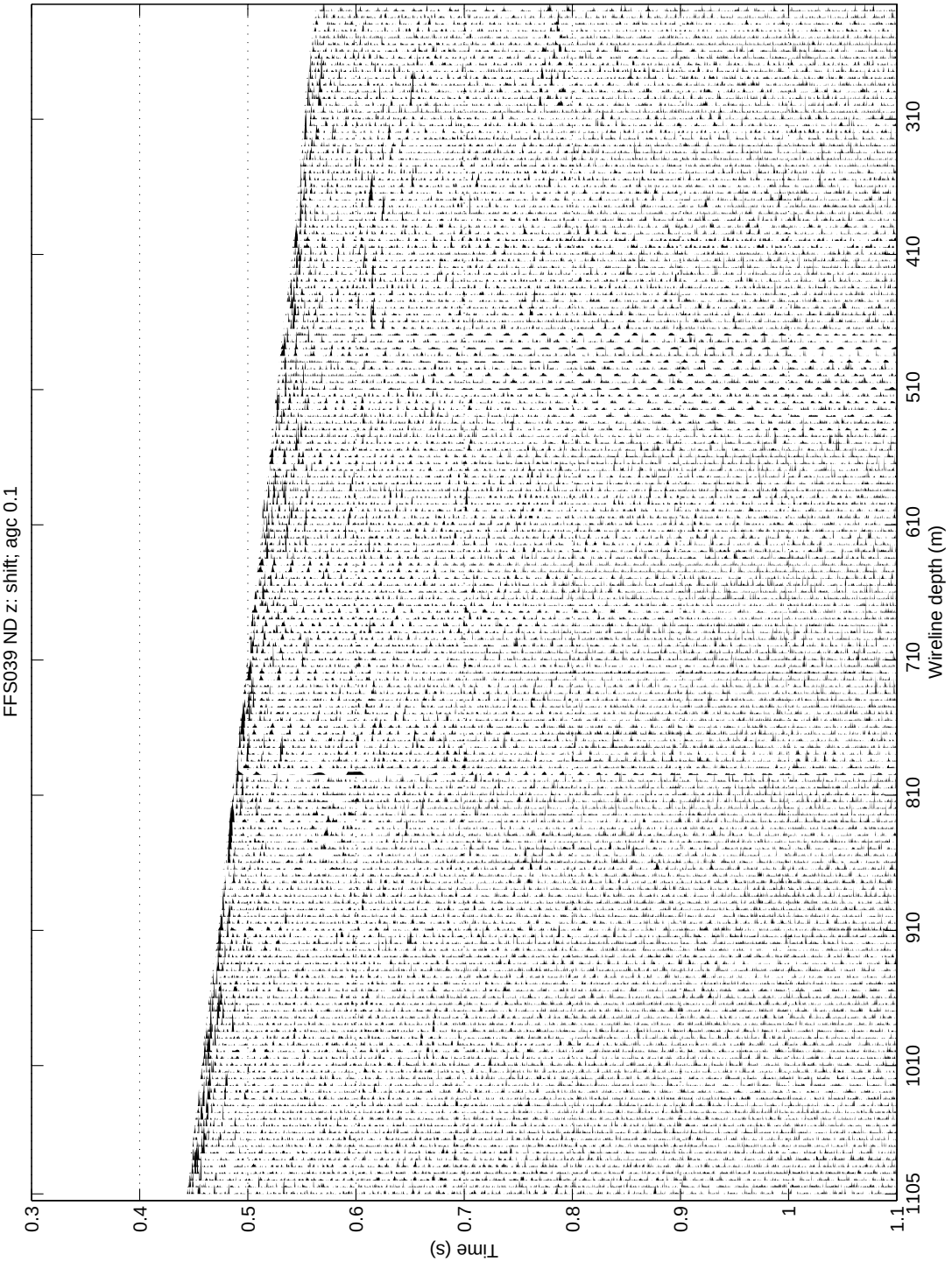

Figure C.13: Data with time added at the beginning of each trace. S-wave arrival time is lined up horizontally.

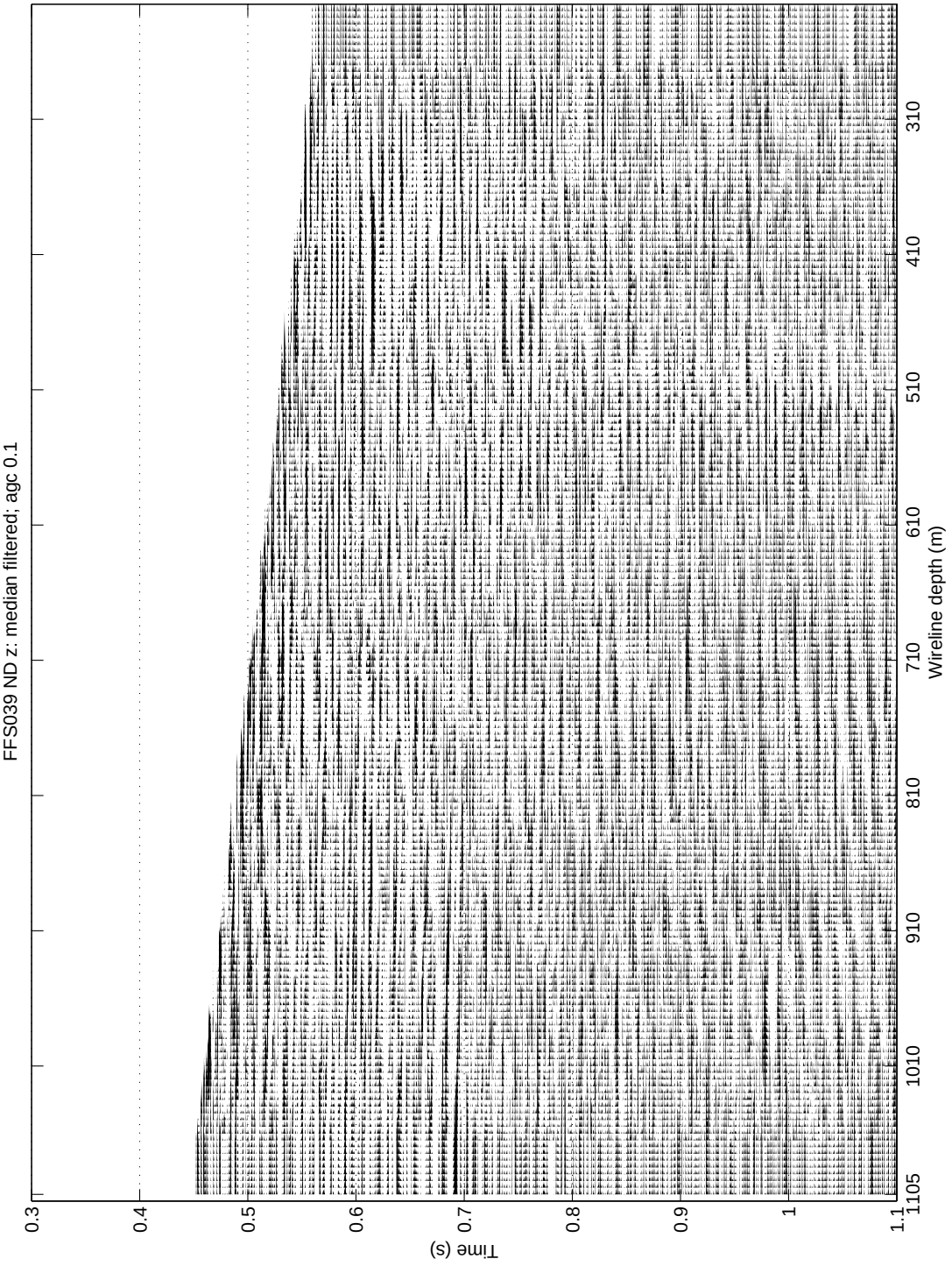

Figure C.14: A median filter of 11 traces width was applied on data.

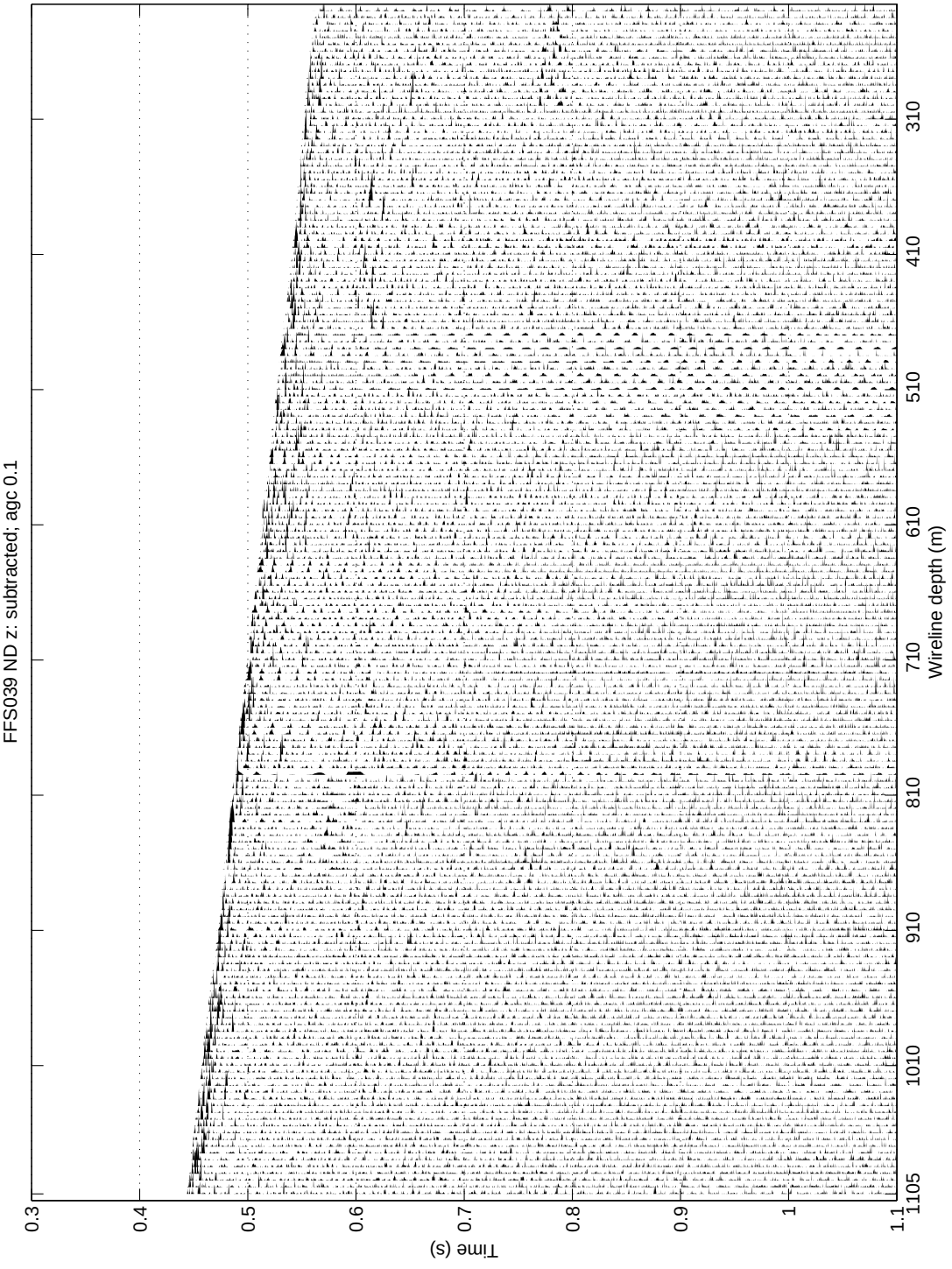

Figure C.15: The median filtered data is subtracted from the shifted data.

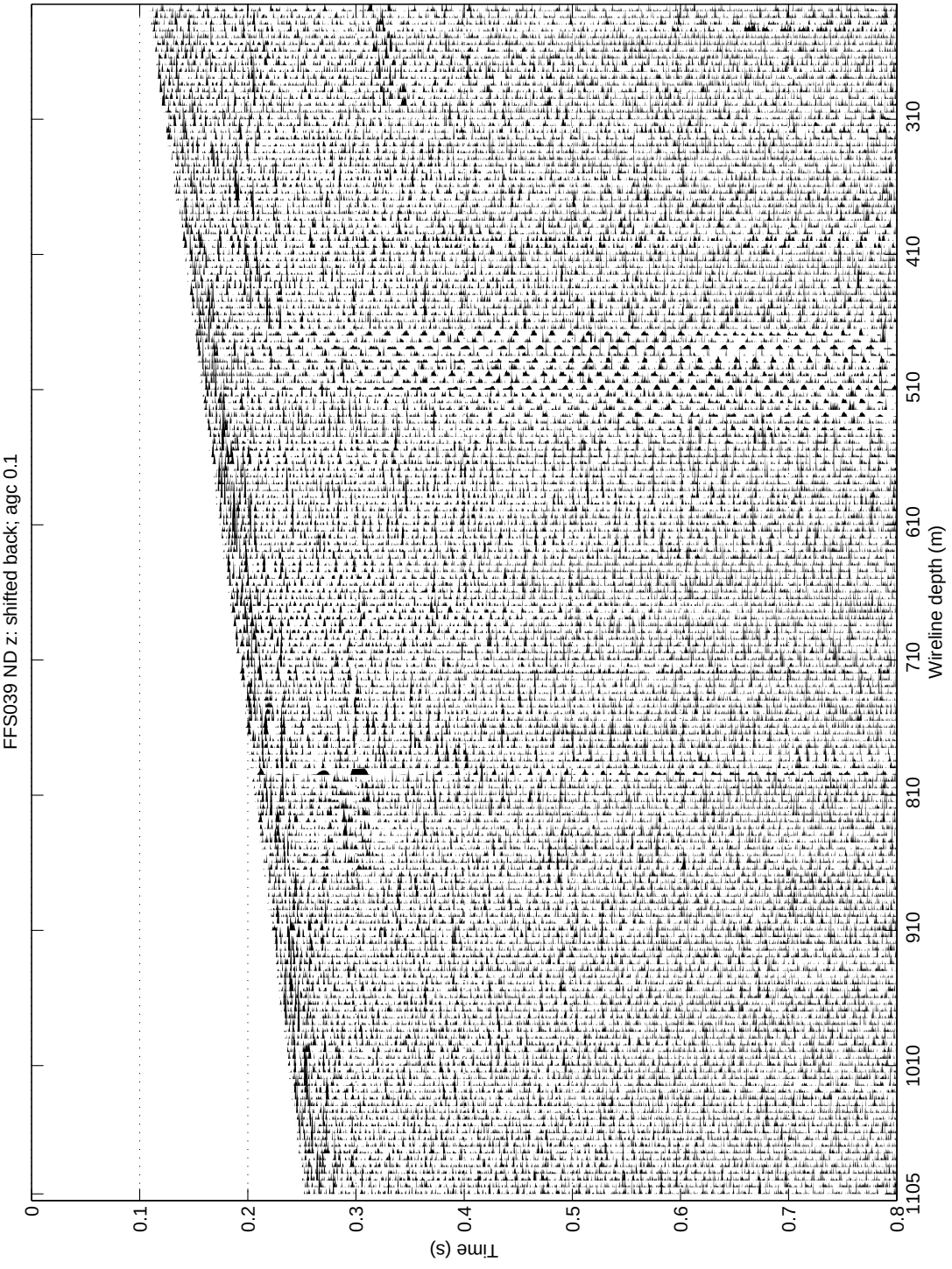

Figure C.16: The added time is removed and the data is shifted back.

C.4 Removal of Horizontal Energy

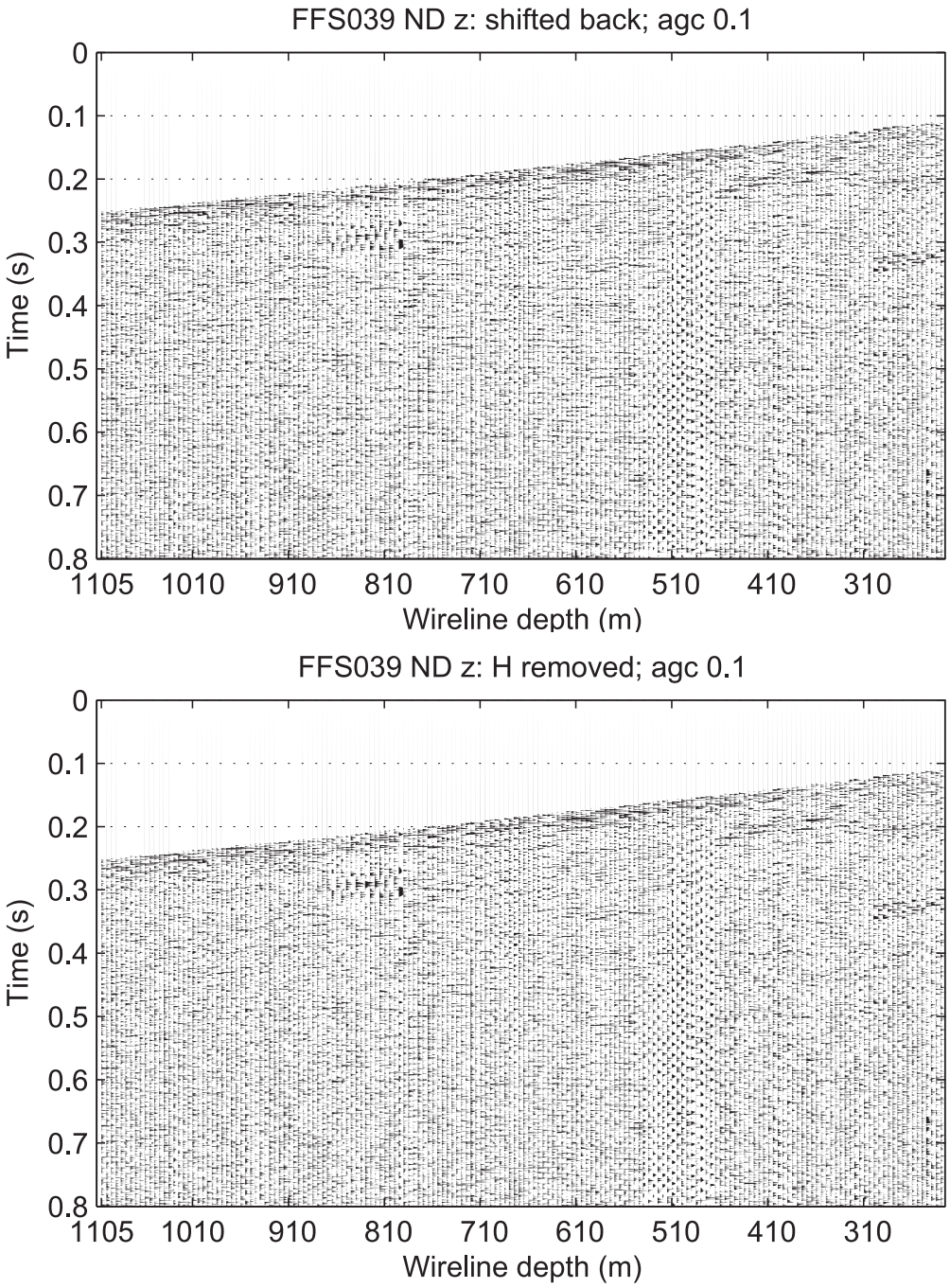

Figure C.17: Removal of horizontal events. Before is shown on top, whereas the H-removed data is shown at the bottom.

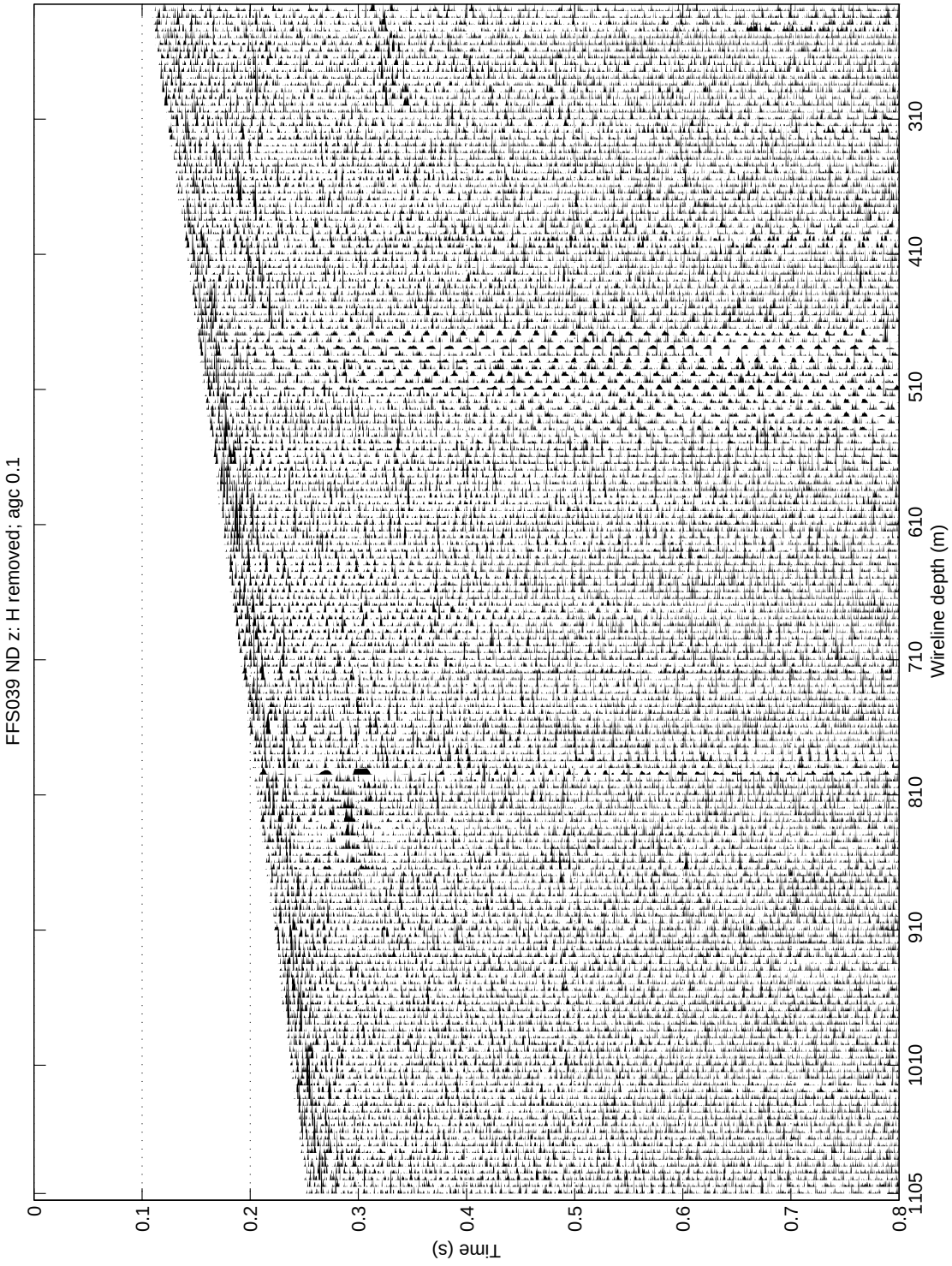

Figure C.18: Data after the horizontal energy is removed.

C.5 Bandpass Filtering

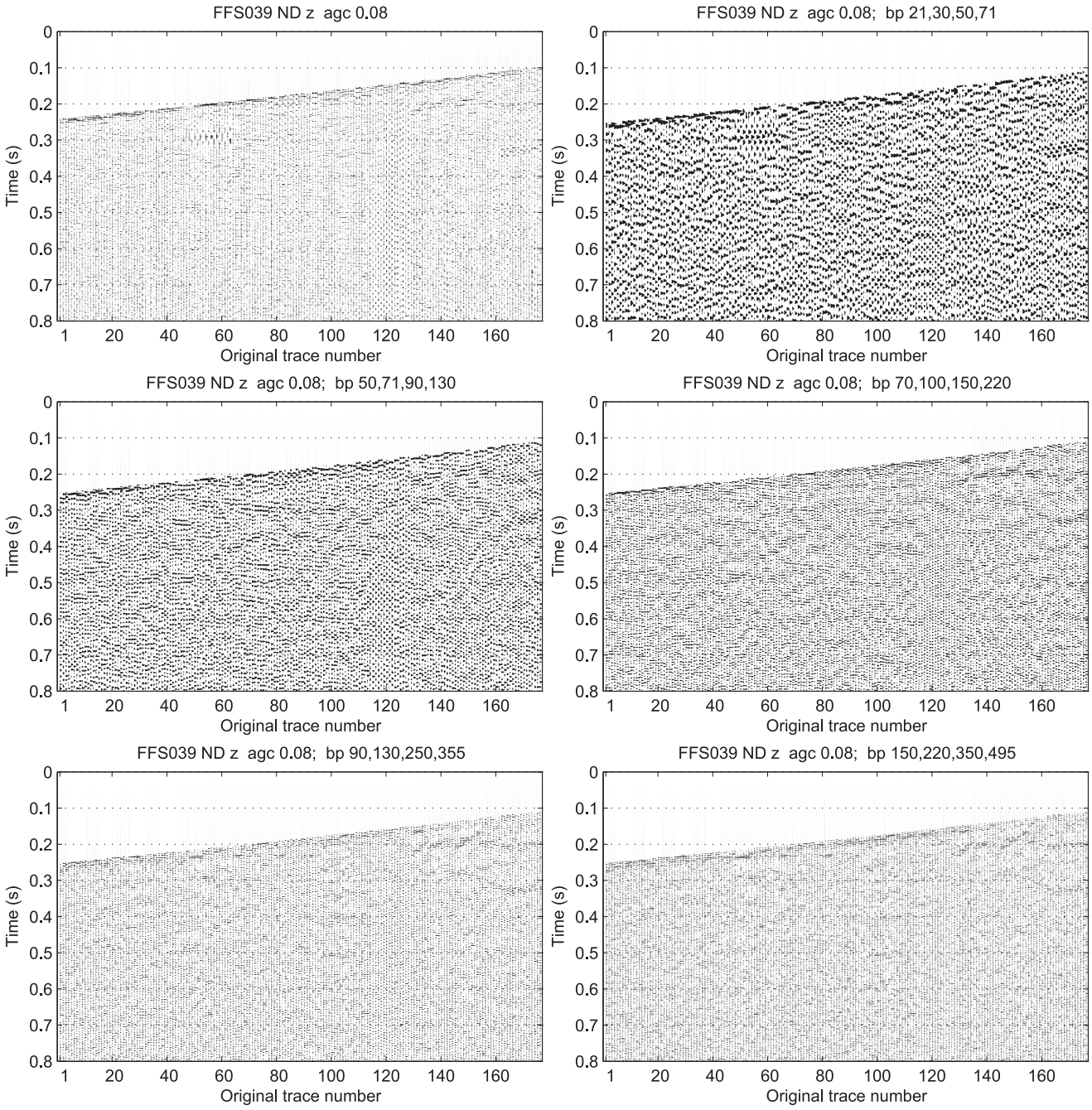

Figure C.19: The scaled, filtered and downgoing energy removed data is shown with five different bandpass filters applied. The corner frequencies of the filters are shown in the Figure captions.

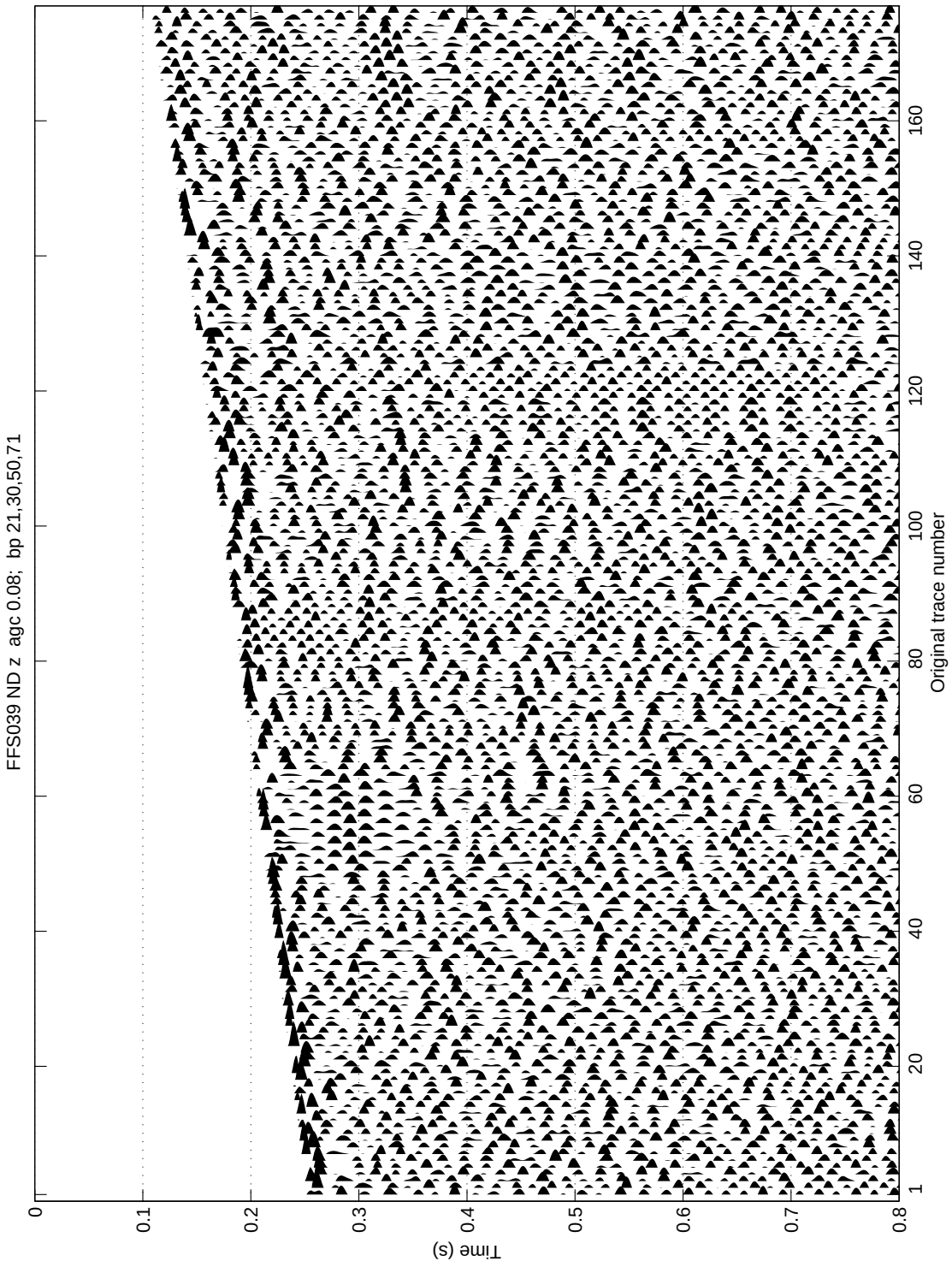

Figure C.20: A bandpass filter with the corner frequencies of 21, 30, 50, 71 Hz was applied.

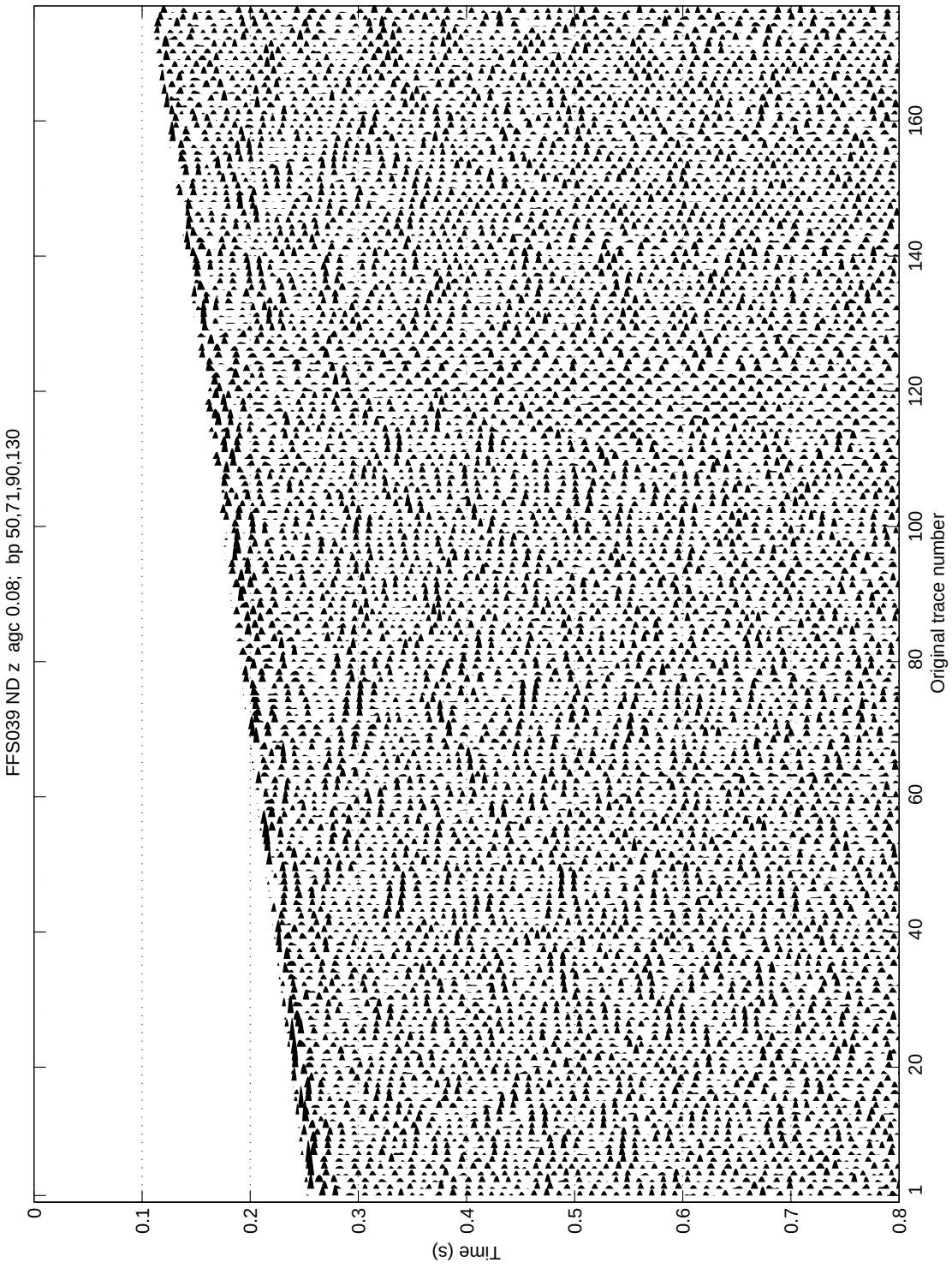

Figure C.21: A bandpass filter with the corner frequencies of 50, 71, 90, 130 Hz was applied

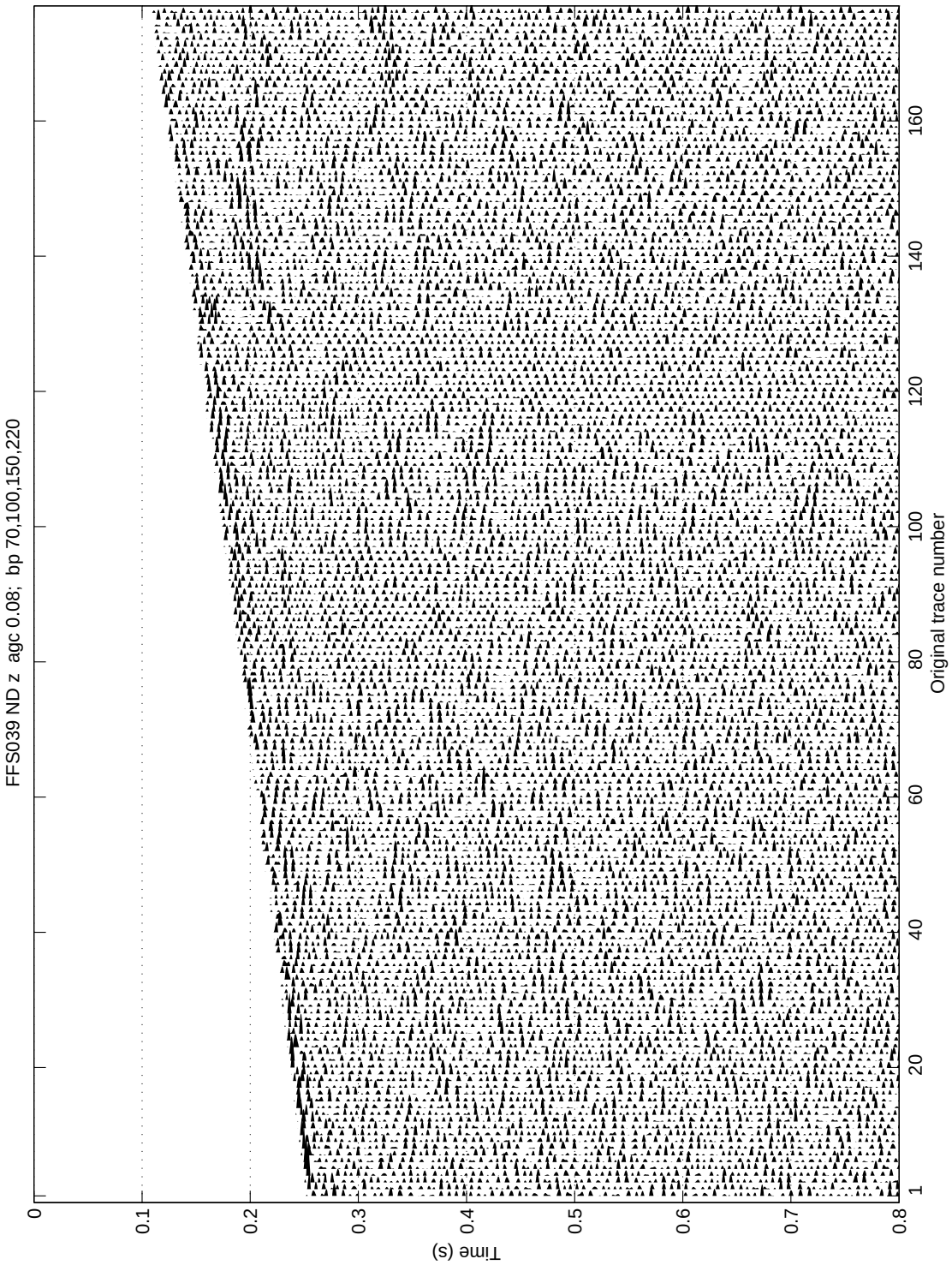

Figure C.22: A bandpass filter with the corner frequencies of 70, 100, 150, 220 Hz was applied.

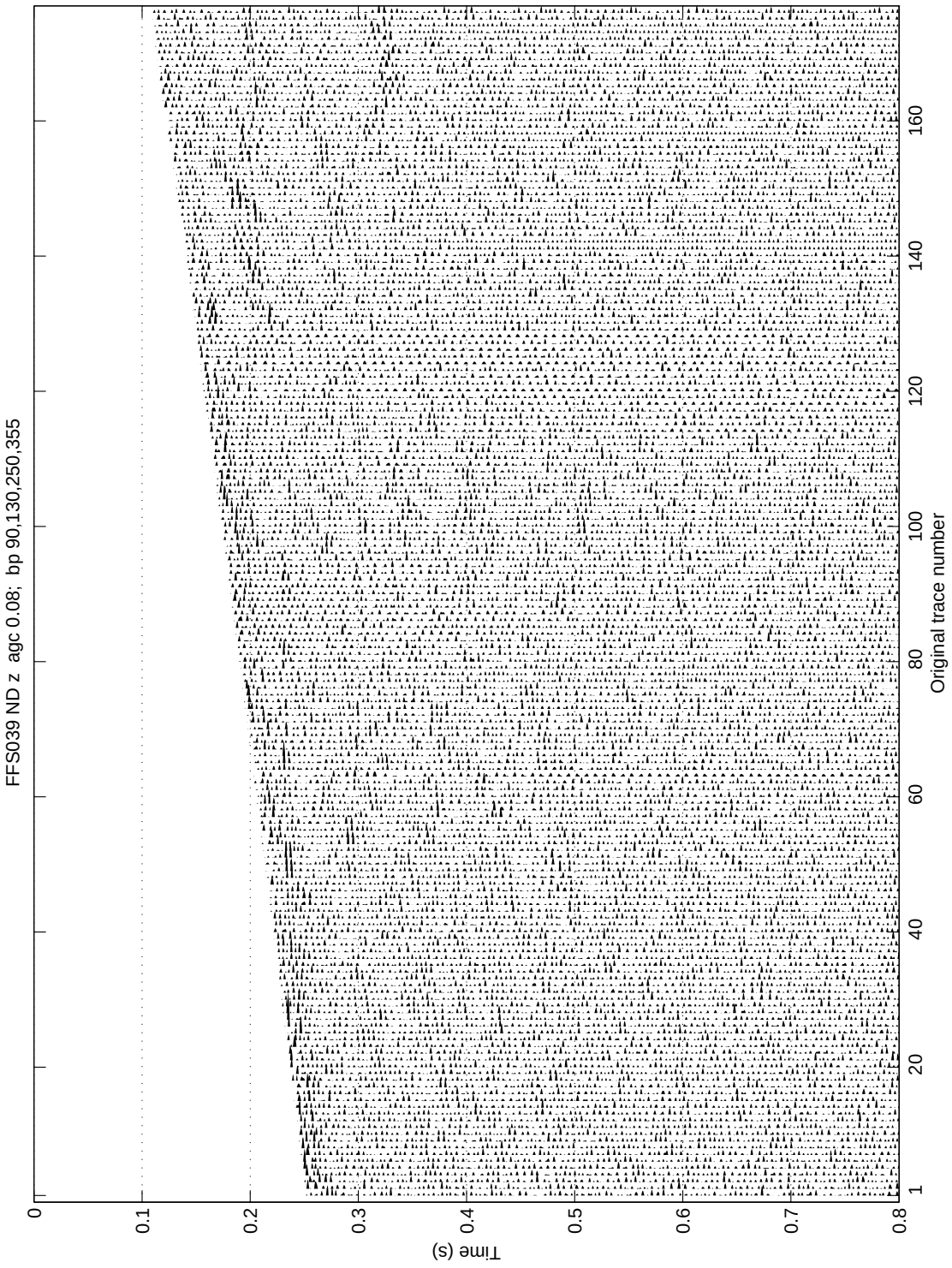

Figure C.23: A bandpass filter with the corner frequencies of 90, 130, 250, 355 Hz was applied.

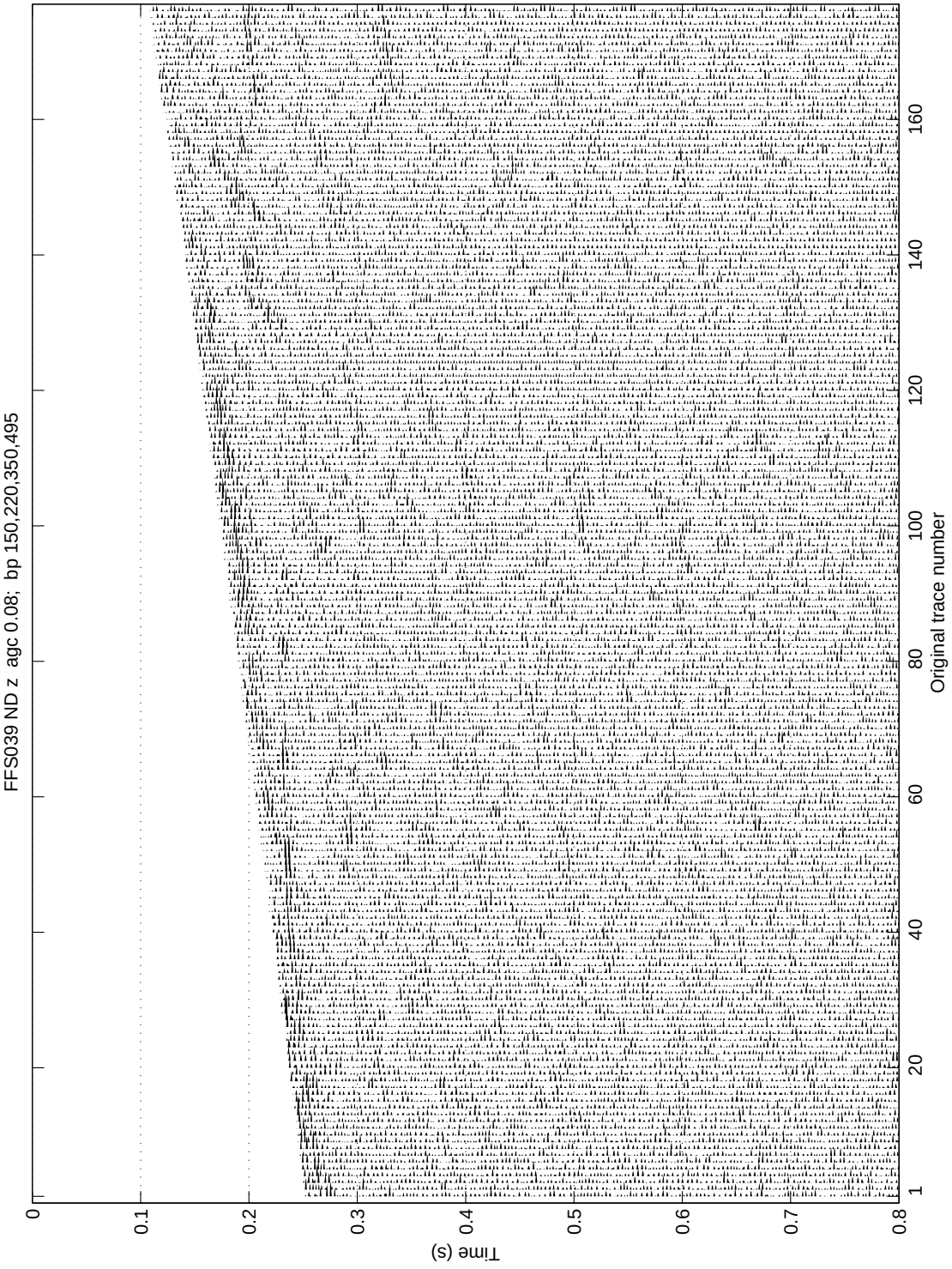

Figure C.24: A bandpass filter with the corner frequencies of 150, 220, 350, 495 Hz was applied.

Appendix D

New Modules for DSISoft

D.1 time_scale

```
% This function will do a time variant scaling: time to the power
% of fact (amp*t^fact).

function [dataout]=time_scale(datain,fact) dataout=datain;

tstart=datain.fh{9};                    % start time in seconds
int=datain.fh{8};                       % sampling interval in seconds
npts=datain.fh{7};                      % number of points per trace
tend=datain.fh{10};                     % end time in seconds

scale_vec = [tstart:int:tend]'.^fact;

nrec=datain.fh{12};                     % number of records in datain

for COUNT=1:nrec
     n=datain.th{COUNT}(12,1);          % number of traces in a record
for i=1:n
       dataout.dat{COUNT}(:,i)=datain.dat{COUNT}(:,i).*scale_vec;
    end                                 % loop over traces
end                                     % loop over records
```

D.2 cos_notch

```
% Notch filter using Hanning window
% Frequencies <=F1 and >=F4 are untouched
% Frequencies between F2 and F3 are zeroed
% Frequencies between F2 and F1 as well as between F3 and F4 are
% reduced by a Hanning window
% All frequencies are in Hertz (Hz)
%
function [dataout]=cos_notch(datain,F1,F2,F3,F4)


dataout=datain; nrec=datain.fh{12};
int=datain.fh{8};                       % sampling interval
Ny=1./(int.*2);                         % Nyquist frequency
npts=datain.fh{7};                      % number of points in each trace
N=2^(nextpow2(npts)+1);                 % number of points used in fft
f=2*Ny*(0:N/2-1)/N;                     % frequency vector

% create filter function in frequency domain => Hanning window
mask = ones(1, Ny+1);
vec=(0:1:Ny);
diff1=F2-F1;
diff2=F4-F3;
down=(0:1/diff1:1);
up=(0:1/diff2:1);
mask(F1:F2) = 0.5*(1+cos(pi*(down))); % (k,0.5*(1+cos(pi*k)))
mask(F3:F4) = 0.5*(1-cos(pi*(up)));   % 0.5*(1-cos(2*pi*f/Ny))
mask(F2:F3) = 0; xf=[0 F1 F2 F3 F4 f(N/2)];
mi=interp1(vec,mask,f);
filt2=zeros(N,1);
filt2(1:N/2)=mi;
mi2=fliplr(mi);
filt2(N/2+1)=1;
filt2(N/2+2:N)=mi2(1:length(mi2)-1);
% 'filt' is filter function in frequency domain

for COUNT=1:nrec                        % loop over records
    ntr=datain.th{COUNT}(12,1);         % # of traces in this record
    filtmat2=ndgrid(filt2,1:ntr);       % create filter matrix
    in_freq2=fft(datain.dat{COUNT},N);% perform N point fft in columns
    out_freq2=zeros(N,ntr);             % initialize
    out_freq2=in_freq2.*filtmat2;       % multiply in freq. domain
```

```
    out2=ifft(out_freq2,N);           % N points inverse fft
    dataout.dat{COUNT}=real(out2(1:npts,:));

    % error check
    re=real(out2(1:npts,1));
    im=imag(out2(1:npts,1));
        for i=npts:-1:1
         rat(i)=im(i)./re(i);
        end;
    x=find(rat>10^-6);
     if ~ isempty(x)
        error('imaginary part of inverse fft is too large');
    end                               % if
end                                   % loop over records
```

D.3 mute_Hann

```
% The mute let's you do a top, bottom and horizon mutes
% The mute function returns a muted data matrix
% Input arguments are:
%
% headw1=    header word containing the mute times in seconds
% headw2=    header word containing the mute times in seconds
% mute_flg:1=mutes everything above headw1
%   2=mutes everything below headw1
%   3=mutes everything above headw1 and everything below headw2
%   4=mutes everything between headw1 and headw2
% taper= taper length in s added before and after the mute times,
% cosine shaped

function [dataout]=mute_Hann(datain,mute_flg,headw1,headw2,taper)
if nargin==3
  headw2=headw1;
end

dataout=datain;
m=datain.fh{7};                         % number of points per trace
a=0;                                    % multiplier for muting data

int=round(taper/datain.fh{8});          % creates a taper vector
vec=(0:1/int:1);
tap_vec2 = (0.5*(1+cos(pi*(vec))))';  % (k,0.5*(1+cos(pi*k)))
tap_vec = (0.5*(1-cos(pi*(vec))))';
for COUNT=1:datain.fh{12}               % loop over records
    n=datain.th{COUNT}(12,1);           % number of traces in a record
                                        % if statement mute spec data
% mut_ind1 and mut_ind2 are the indexes (-1) corresponding to times in
% headw1 and %headw2
    if mute_flg==1                      % mutes data before headw1
     for i=1:n
       if datain.th{COUNT}(headw1,i)~=0
            tap_start=round((datain.th{COUNT}(headw1,i)-...
            datain.fh{9})/datain.fh{8});
            dataout.dat{COUNT}(1:tap_start,i)=datain.dat{COUNT}...
            (1:tap_start,i).*a;
% zeros data before headerword
            tap_end=tap_start+int;
            dataout.dat{COUNT}(tap_start:tap_end,i)=datain.dat...
```

```
            {COUNT}(tap_start:tap_end,i).*tap_vec;
% applies a taper
       end
      end

    elseif mute_flg==2               % mute data after time in headw1
      for i=1:n
       if datain.th{COUNT}(headw1,i)~=0
        tap_start=round((datain.th{COUNT}(headw1,i)-taper-...
        datain.fh{9})/datain.fh{8});
        tap_end=tap_start+int;
        dataout.dat{COUNT}(tap_start:tap_end,i)=datain.dat...
        {COUNT}(tap_start:tap_end,i).*tap_vec2;
% applies taper
        dataout.dat{COUNT}(tap_end:m,i)=datain.dat...
        {COUNT}(tap_end:m,i).*a;
       end                           % zeros data after headerword
      end

    elseif mute_flg==3      % mute data before headw1 and after headw2
     for i=1:n
     if (datain.th{COUNT}(headw1,i)~=0&datain.th{COUNT}(headw2,i)~=0)
            tap_start=round((datain.th{COUNT}(headw1,i)-...
            datain.fh{9})/datain.fh{8});
            dataout.dat{COUNT}(1:tap_start,i)=datain.dat...
            {COUNT}(1:tap_start,i).*a;
% zeros data before headerword
            tap_end=tap_start+int;    %applies a taper
            dataout.dat{COUNT}(tap_start:tap_end,i)=datain.dat...
            {COUNT}(tap_start:tap_end,i).*tap_vec;
            tap_start=round((datain.th{COUNT}(headw2,i)-taper-...
            datain.fh{9})/datain.fh{8});
            tap_end=tap_start+int;
            dataout.dat{COUNT}(tap_end:m,i)=datain.dat...
            {COUNT}(tap_end:m,i).*a;
% zeros data after headerword
            dataout.dat{COUNT}(tap_start:tap_end,i)=datain.dat...
            {COUNT}(tap_start:tap_end,i).*tap_vec2;
       end
      end

    else                          % mute data between headw1 and headw2
```

```
      for i=1:n
      if(datain.th{COUNT}(headw1,i)~=0&datain.th{COUNT}(headw2,i)~=0)
            tap_start1=round((datain.th{COUNT}(headw1,i)-taper-...
            datain.fh{9})/datain.fh{8});
            tap_end1=tap_start1+int;
            tap_start2=round((datain.th{COUNT}(headw2,i)-...
            datain.fh{9})/datain.fh{8});
            tap_end2=tap_start2+int;
            dataout.dat{COUNT}(tap_start1:tap_end1,i)=datain.dat...
            {COUNT}(tap_start1: tap_end1,i).*tap_vec2;
% applies taper
            dataout.dat{COUNT}(tap_end1:tap_start2,i)=datain.dat...
            {COUNT}(tap_end1: tap_start2,i).*a;
% zeros data between headw1 and headw2
            dataout.dat{COUNT}(tap_start2:tap_end2,i)=datain.dat...
            {COUNT}(tap_start2:tap_end2,i).*tap_vec;
% applies a taper
       end
      end
    end
end                                   % loop over records
```

D.4 new_bandpass

```
% Hanning window-shaped bandpass filter
% Frequencies <=F1 and >=F4 are zeroed
% Frequencies between F2 and F3 are untouched
% Frequencies between F2 and F1 as well as between F3 and F4 are
% reduced in with a Hanning window (cosine shaped).
% All frequencies is in Hertz (Hz).
%
function [dataout]=new_bandpass(datain,F1,F2,F3,F4)

dataout=datain; nrec=datain.fh{12};
int=datain.fh{8};                       % sampling interval
Ny=1./(int.*2);                         % Nyquist frequency
npts=datain.fh{7};                      % number of points in each trace
N=2^(nextpow2(npts)+1);                 % number of points used in fft
f=2*Ny*(0:N/2-1)/N;                     % frequency vector

mask = zeros(1, Ny+1);                  % shape for Hanning
vec=(0:1:Ny);
diff1=F2-F1;
diff2=F4-F3;
up=(0:1/diff1:1);
down=(0:1/diff2:1);
mask(F1:F2) = 0.5*(1-cos(pi*(up)));
mask(F3:F4) = 0.5*(1+cos(pi*(down)));
mask(F2:F3) = 1;

xf=[0 F1 F2 F3 F4 f(N/2)];              % x=[1 1 0 0 1 1];
mi=interp1(vec,mask,f);                 % plot(vec,mask,'o',f,m,'+')
filt2=zeros(N,1);
filt2(1:N/2)=mi;
mi2=fliplr(mi);
filt2(N/2+1)=1;
filt2(N/2+2:N)=mi2(1:length(mi2)-1);

% 'filt'=filter function in frequency domain

for COUNT=1:nrec                        % loop over records
    ntr=datain.th{COUNT}(12,1);         % number of traces in record
    filtmat=ndgrid(filt2,1:ntr);        % create filter matrix
    in_freq=fft(datain.dat{COUNT},N); % perform N point fft on columns
    out_freq=zeros(N,ntr);              % initialize
```

```
    out_freq=in_freq.*filtmat;        % multiply in freq. domain
    out=ifft(out_freq,N);             % N points inverse fft
    dataout.dat{COUNT}=real(out(1:npts,:));
% error check
    re=real(out(1:npts,1));
    im=imag(out(1:npts,1));
    for i=npts:-1:1
        rat(i)=im(i)./re(i);
    end;
    x=find(rat>10^-6);
    if ~ isempty(x)
        error('imaginary part of inverse fft is too large');
    end                               % if
end                                   % loop over records
```

D.5 corrstack_t

```
% creates a corridor stack (aka composite trace)
% ntrpr = number of traces per record to put in corridor stack output
%
function [dataout]=corrstack_t(datain,ntrpr) dt=datain.fh{8};
dataout = datain;
for COUNT =1: datain.fh{12}           % loop over records
% make some changes to file header
   dataout.fh=datain.fh;              % copy file header
   n=datain.fh{7};
   dataout.fh{1}=ntrpr;               % number of traces in dataset
   dataout.th{COUNT}=datain.th{COUNT};
   dataout.dat{COUNT}=zeros(n,ntrpr);
   for j=1:ntrpr
       dataout.th{COUNT}(12,j)=ntrpr; % number of traces in record
% put sum into dataout
       dataout.dat{COUNT}(:,j)=sum(datain.dat{COUNT}(:,:)')';
   end
end                                   % loop over records
```

D.6 merge_traces

```
% Function to merge traces from two diffrent files into one file,
% adds an empty trace between the two sets

function [dataout]=merge_traces2(datain1,datain2)
% check to make sure that files have same # of points, sampling rate

if ~datain1.fh{7}==datain2.fh{7} | ~datain1.fh{8}==datain2.fh{8}
error('Sampling rate + number of points per trace must be the same.')
end dataout.fh=datain1.fh;
dataout.fh{1}=datain1.fh{1}+datain2.fh{1}+1; %total # traces in file
dataout.fh{12}=datain1.fh{12};        % number of records in file
dataout.fh{13}=max([dataout.fh{1}]);  % max record fold
nrec1=datain1.fh{12}; nrec2=datain2.fh{12};


for rec=1:nrec1
   dataout.th{rec}=zeros(64,dataout.fh{1}); % initialize trace headers
   dataout.th{rec}(:,1:datain1.fh{1})=datain1.th{rec}(:,1:datain1.fh{1});
   % copy headers of first set of traces
   dataout.th{rec}(:,datain1.fh{1}+2:dataout.fh{1})=datain2.th{rec}(:,:);
   % copy headers of second set of traces
   dataout.dat{rec}=zeros(datain1.fh{7},dataout.fh{1});
   dataout.dat{rec}(:,1:datain1.fh{1})=datain1.dat{rec}(:,:);
   % copy data of first set of %traces
   dataout.dat{rec}(:,datain1.fh{1}+2:dataout.fh{1})=datain2.dat{rec}(:,:);
   % copy %data of second set of traces
end                                   % loop over files
```

D.7 window_mute4

```
% Mutes all data which don't lie in a specific window (poly=trace-time
% points), applies taper around the window. Does not loop over records.
% datain = input data in DSI format
% poly =   trace/time coordinates of the corners of window polygone
% taper = length in sec for tapering

function [dataout,win]=window_mute4(datain,poly1,taper,rec)
dataout=datain;
int=datain.fh{8};                    % sampling interval (s)

trac=datain.fh{1};                   % NUMBER OF TRACES
sta=datain.fh{9};                    % start time
nu=datain.fh{7};                     % sample points in y

poly2=poly1;
poly2(:,2)=poly2(:,2)+taper;
poly3=poly1;
poly3(:,2)=poly3(:,2)-taper;

datain=datain.dat{1};
[m,n]=size(datain);
window=zeros(m,n);
winlow=zeros(m,n);
wintop=zeros(m,n);                   % initializing filter matrix
x=[1:1:trac];                        % traces
y=[1:1:nu];                          % sample points
[xmat,ymat]=meshgrid(x,y);           % creating polymatrix

%--------finding points in polygon---------
inpass=inpolygon(xmat,ymat,poly1(:,1),round((poly1(:,2)-sta)/int));
in=inpass.*-1;
rejlow=inpolygon(xmat,ymat,poly2(:,1),round((poly2(:,2)-sta)/int));
rejtop=inpolygon(xmat,ymat,poly3(:,1),round((poly3(:,2)-sta)/int));

%-------------creating window--------------
% lower taper
winlow=in+rejlow; for i=1:trac
    [a] = find(winlow(:,i)==1);
    len=length(a);
    up=(1/len:1/len:1);
    winlow(a,i)=0.5*(1+cos(pi*(up')));
```

```
   for k=1:nu
        if winlow(k,i) < 0
                winlow(k,i) = 0;
        end                                % if
   end                                     % for k
end                                        % for i

% top taper
wintop=in+rejtop; for i=1:trac
    [a] = find(wintop(:,i)==1);
    len=length(a;
    up=(1/len:1/len:1);
    wintop(a,i)=0.5*(1-cos(pi*(up')));
    for k=1:nu
        if wintop(k,i) < 0
                wintop(k,i) = 0;
        end
   end
end

win=wintop+winlow+inpass;
%---------------apply window---------------
dataout=datain.*win;
```

D.8 norm_stack

```
% creates a corridor stack (aka composite trace) normalized by number
% of traces        => works only with window_mute4 !!!!!
% since window function is used for normalization
% scal = window matrix provided by window_mute4
% ntrpr = # traces per record to put in corridor stack output depth

dt=datain.fh{8};
corr_out = datain;
for COUNT =1: datain.fh{12}           % loop over records
            % make some changes to file header
   corr_out.fh=datain.fh;             % copy file header
   n=datain.fh{7};
   corr_out.fh{1}=ntrpr;              % number of traces in dataset
   corr_out.th{COUNT}=datain.th{COUNT};
   corr_out.dat{COUNT}=zeros(n,ntrpr);
   normal=ones(n,1);
   normal=sum(scal')'+1;
   for j=1:ntrpr
       corr_out.th{COUNT}(12,j)=ntrpr; % number of traces in record
       % put sum into dataout
       corr_out.dat{COUNT}(:,j)=sum(datain.dat{COUNT}(:,:)')';
       corr_out.dat{COUNT}(:,j)=corr_out.dat{COUNT}(:,j)./normal(:);
   end
   dataout=corr_out;
end                                   % loop over records
```

D.9 cos_notch_test1

```
function[dataout,fi_spec]=cos_notch(datain,F1,F2,F3,F4)

dataout=datain;
nrec=datain.fh{12};
int=datain.fh{8};                       % sampling interval
Ny=1./(int.*2);                         % Nyquist frequency
npts=datain.fh{7};                      % number of points in each trace
N=2^(nextpow2(npts)+1);                 % number of points used in fft

f=2*Ny*(0:N/2-1)/N;                     % frequency vector

% create filter function in frequency domain => Hanning window

diff1=F2-F1; diff2=F4-F3; down=(0:1/diff1:1); up=(0:1/diff2:1);
mask(F1:F2) = 0.5*(1+cos(pi*(down)));
mask(F3:F4) =0.5*(1-cos(pi*(up)));
mask(F2:F3) = 0;

mi=mask;

filt2=zeros(N,1);
filt2(1:N/2)=mi;
mi2=fliplr(mi);
filt2(N/2+1)=1;
filt2(N/2+2:N)=mi2(1:length(mi2)-1);
% 'filt' is filter function in frequency domain

for COUNT=1:nrec                        % loop over records
    ntr=datain.th{COUNT}(12,1);         % # traces in this record
    filtmat2=ndgrid(filt2,1:ntr);       % create filter matrix
    in_freq2=fft(datain.dat{COUNT},N);% perform N point fft on columns
    out_freq2=zeros(N,ntr);             % initialize
    out_freq2=in_freq2.*filtmat2;       % multiply in freq. domain
    fi_spec=real(out_freq2(1:600,:));
    out2=ifft(out_freq2,N);             %N points inverse fft
    dataout.dat{COUNT}=real(out2(1:npts,:));

% error check
    re=real(out2(1:npts,1));
    im=imag(out2(1:npts,1));
    for i=npts:-1:1
```

```
        rat(i)=im(i)./re(i);
    end;
    x=find(rat>10^-6);
    if ~ isempty(x)
        error('imaginary part of inverse fft is too large');
    end                                 % if
end                                     % loop over records
```

D.10 txdirect1

```
function [dist,t]=txdirect1(s,r,v)

% txdirect: function to calculate the traveltime in a medium with constant
% velocity v, given a source at s=[sx;sy;sz] and receiver at r=[rx;ry,rz].
% INPUT: s, r, v
% OUTPUT: travel distance and travel time.

dist=sqrt(dot ( (s-r),(s-r) ) );
t=dist/v;
```

D.11 txplane2

```
function [cdp,distref,tref]=txplane2(s,r,x,dip,strike,v)

% txplane: function to calculate reflection point,travelled distance and
% traveltime in a medium with constant velocity v, given a source at
% s=[sx;sy;sz] and receiver at r=[rx;ry;rz], with a reflection from a
% plane with given dip and strike, passing through the point x=[x;y;z].
% INPUT: s = [sx; sy; sz], source position
%        r = [rx; ry; rz], receiver position
%        x = [x; y; z], a point on the plane
%        v = velocity of medium (constant)
%        dip = dip of plane (horizontal -> dip=0 (degrees)
%        strike = strike of plane (clockwise from N in degrees)
% OUTPUT: reflection point position, travelled distance and traveltime

dip=dip*pi/180;
strike=strike*pi/180;
n=[0 0 1]; p=dot(n,x);

rimage = r+2*(p - dot(n,r) )*n;
distref=sqrt(dot((s-rimage),(s-rimage)));
tref=distref/v;
cdp=s+(p-dot(n,s))/(dot(n,rimage)-dot(n,s))*(rimage-s);
```

D.12 cos_notch_newtest

```
function [dataout]=cos_notch_newtest(datain,F)

% notch filter using Hanning window
% F is a matrix (x by 4) containing all necessary notch frequencies
% Frequencies <=F1 and >=F4 are untouched
% Frequencies between F2 and F3 are zeroed
% Frequencies between F2 and F1 as well as between F3 and F4 are
% reduced by a Hanning window
%
% Revision 4.0  2007/01/18 10:20:57 barbara

disp('[dataout]=cos_notch_newtest(datain,F1,F2,F3,F4)');

dataout=datain;
nrec=datain.fh{12};
int=datain.fh{8};                    % sampling interval
Ny=1./(int.*2);                      % Nyquist frequency
npts=datain.fh{7};                   % number of points in each trace
N=2^(nextpow2(npts)+1);              % number of points to be used in fft

f=2*Ny*(0:N/2-1)/N;                  % frequency vector

% create filter function in frequency domain => Hanning window

mask = ones(size(f));
[x,y]=size(F);
for i=1:x
    Fa=F(i,1);
    Fb=F(i,2);
    Fc=F(i,3);
    Fd=F(i,4);
    Fa=round(Fa*0.5*N/Ny);
    Fb=round(Fb*0.5*N/Ny);
    Fc=round(Fc*0.5*N/Ny);
    Fd=round(Fd*0.5*N/Ny);

    diff1=Fb-Fa;
    diff2=Fd-Fc;
    down=(0:1/diff1:1);
    up=(0:1/diff2:1);
    mask(Fa:Fb) = 0.5*(1+cos(pi*(down)));
```

```
    mask(Fc:Fd) = 0.5*(1-cos(pi*(up)));
    mask(Fb:Fc) = 0;
end

mi=mask;                                % plot(vec,mask,'o',f,m,'+')

filt2=zeros(N,1);
filt2(1:N/2)=mi;
mi2=fliplr(mi);
filt2(N/2+1)=1;
filt2(N/2+2:N)=mi2(1:length(mi2)-1);

% 'filt' is filter function in frequency domain

for COUNT=1:nrec                        % loop over records
    ntr=datain.th{COUNT}(12,1);         % number of traces in this record
    filtmat2=ndgrid(filt2,1:ntr);       % create filter matrix
    in_freq2=fft(datain.dat{COUNT},N); % performs N points fft on columns
    out_freq2=zeros(N,ntr);             % initialize
    out_freq2=in_freq2.*filtmat2;       % multiply in freq. domain
    out2=ifft(out_freq2,N);             % N points inverse fft
    dataout.dat{COUNT}=real(out2(1:npts,:));

% error check
    re=real(out2(1:npts,1));
    im=imag(out2(1:npts,1));
    for i=npts:-1:1
        rat(i)=im(i)./re(i);
    end;
    x=find(rat>10^-6);
    if ~ isempty(x)
        error('imaginary part of inverse fft is too large');
    end                                 % if
end                                     % loop over records
```

Appendix E

Processing Scripts 4Q66W3 near offset

E.1 convert from .segy to .mat (sc_segy2mat.m)

```
[data]=segy2mat('4Q66NDYN.sgy','cross2.crs','l');

save Q66NDYN.mat data
```

E.2 correct wrong trace headers (sc_write_th.m)

```
% Script to rewrite the wrong receiver coordinates using the file
% COR_REC.TXT where the new calculated coordinates are stored.
% Coordinates were interpolated from the survey-log
% (gems_drillhole_utm_trace.txt)

load cor_rec.txt
% it is an [192 x 5] matrix where rec station # are in colomn 1,
% wireline depth in colomn 2, UTM east in 3, UTM north in 4
% and elevation in colomn 5

load Q66NDYN.mat

A= cor_rec;
data.th{1}(36,:)=data.th{1}(35,:);
% stores the old rec north in the unused line 36, for surface too
data.th{1}(35,1:192)=A(:,4);
% stores the 192 north coordiantes in the 192 z-traces
data.th{1}(35,193:384)=A(:,4);
% stores the 192 north coordiantes in the 192 H1-traces
data.th{1}(35,385:576)=A(:,4);
% stores the 192 north coordiantes in the 192 H2-traces
data.th{1}(35,577:768)=data.th{1}(35,577:768)/100;
% stores the 192 surface coordiantes in the 192 surface traces

data.th{1}(38,:)=data.th{1}(37,:);
% stores the old rec east in the unused line 38, for surface, too
data.th{1}(37,1:192)=A(:,3);
% stores the 192 east coordiantes in the 192 z-traces
data.th{1}(37,193:384)=A(:,3);
% stores the 192 east coordiantes in the 192 H1-traces
data.th{1}(37,385:576)=A(:,3);
% stores the 192 east coordiantes in the 192 H2-traces
data.th{1}(37,577:768)=data.th{1}(37,577:768)/100;
% stores the 192 surface coordiantes in the 192 surface traces

data.th{1}(40,:)=data.th{1}(39,:);
% stores the old rec elev in the unused line 40, for surface, too
data.th{1}(39,1:192)=A(:,5);
% stores the 192 north coordiantes in the 192 z-traces
data.th{1}(39,193:384)=A(:,5);
% stores the 192 north coordiantes in the 192 H1-traces
```

```
data.th{1}(39,385:576)=A(:,5);
% stores the 192 north coordiantes in the 192 H2-traces
data.th{1}(39,577:768)=data.th{1}(39,577:768)/100;
% stores the 192 surface coordiantes in the 192 surface traces

data.th{1}(56,1:192)=A(:,2);
% stores the 192 wireline depth values in the 192 z-traces
data.th{1}(56,193:384)=A(:,2);
% stores the 192 wireline depth values in the 192 H1-traces
data.th{1}(56,385:576)=A(:,2);
% stores the wireline depth values in the 192 H2-traces

Q66W3ND_cor = data;
save Q66W3ND_cor.mat Q66W3ND_cor
```

E.3 sort data (sc_sort_many2.m)

```
% this script is written to sort the converted data (11=component,
% 39=new receiver elevation, 1=trace number)

load Q66W3ND_cor.mat

datain = Q66W3ND_cor;
sort_flg1=11;
sort_flg2=39;
sort_flg3=1;

[dataout]=sortrec_many(datain,sort_flg1,sort_flg2,sort_flg3)

Q66NDcor_sort = dataout;
save Q66NDcor_sort.mat Q66NDcor_sort
```

E.4 more_geometry (sc_geom.m)

```
% This program computes the shot-to-receiver azimuth angle
% and puts the corresponding angle (in degrees) in header word 9
% It also computes the shot-to-receiver offset (m) and puts the
% results in header word 53. This function implies that header
% words 29,31,33,35,37,39 are set before being used

load Q66NDcor_sort.mat

datain = Q66NDcor_sort;

[dataout]=s2r_geom2(datain)

Q66NDsort_geom = dataout;
save Q66NDsort_geom.mat Q66NDsort_geom
```

E.5 remove horizontal components and store P-times (sc_mod5.m)

```
load Q66NDsort_geom.mat;

datain = Q66NDsort_geom;

% make necessary changes to file header
dataout.fh=datain.fh;
dataout.fh{12}=1;
dataout.th{1}(:,:)=datain.th{2}(:,:);

% record {3} was the horizontal component (H1) is NEW record {1}
% stores the right header value 1 for h1 in header word 4

dataout.th{1}(15,:)=picks(1,:);
% stores the first break times

dataout.dat{1}(:,:)=datain.dat{2}(:,:);
dataout.th{1}(4,:)=1;

dataout=subset(dataout,1,192,0,.99975,1,1);

Q66ND_z_raw = dataout;
save Q66ND_z_raw.mat Q66ND_z_raw
```

E.6 processing of the raw, sorted data (sc_all5.m)

```
Note: AGC is applied for printing purposes only

% ALL PROC STEPS IN ONE SCRIPT
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load Q66ND_z_raw.mat;
load nicepicks.mat;
datain = Q66ND_z_raw;
datain.th{1}(15,:)=nicepicks(:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mute first P-arrival picks.mat
load Q66ND_z_raw.mat;
datain = Q66ND_z_raw;
datain.th{1}(16,:) = datain.th{1}(15,:)+0.005;
% makes sure P wave is muted
headw1=16;
headw2=17;
mute_flg=1;
taper = 0.005;
[dataout]=mute_taper(datain,mute_flg,headw1,headw2,taper)
Q66ND_wP_raw = dataout;
% [scal]=time_scale(dataout,1.7);
% [agc_mute]=agc(scal,0.08,1);
% vaplot(agc_mute,0.0,0.8,1,192,1,1,2,'k',1,20,8,1,0,'all2_top...
% Mute.ps','4Q66W3 ND z: P Top Mute; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% applies cos_notch5
[dataout] = cos_notch(Q66ND_wP_raw,50,60,60,70);
[dataout] = cos_notch(dataout,171,181,181,191);
[dataout] = cos_notch(dataout,291,301,301,311);
[dataout] = cos_notch(dataout,411,421,421,431);
[dataout] = cos_notch(dataout,531,541,541,541);
[dataout] = cos_notch(dataout,651,661,661,671);
[dataout] = cos_notch(dataout,770,780,780,790);
[dataout] = cos_notch(dataout,891,901,901,911);
Q66wP_cos_notch5 = dataout;
% [scal]=time_scale(dataout,1.7);
% [agc_mute]=agc(scal,0.08,1);
% vaplot(agc_mute,0.0,0.8,1,192,1,1,2,'k',1,20,8,1,0,'all2_cos_...
```

```
% notch5.ps','4Q66W3 ND z  cos notch5; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% applies top mute
datain = Q66wP_cos_notch5;
datain.th{1}(15,:)=nicepicks;
[add_P]=add(datain,0.05);
[align_P]=shft(add_P,-1,15);
align_P.th{1}(16,:)=0.075;
align_P.th{1}(17,:)=0.175;
[mute]=mute_Hann(align_P,1,16,16,0.01);
[un_align]=shft(mute,1,15);
[shiftup]=sft(un_align,1,192,-0.05);
Q66wP_top_mute=shiftup;
% [scal]=time_scale(Q66wP_top_mute,1.7);
% [agc_mute]=agc(scal,0.08,1);
% vaplot(agc_mute,0.0,0.8,1,192,1,1,2,'k',1,20,8,1,0,'all4_P_...
% removal.ps','4Q66W3ND z P removed; cos notch5; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% enhance data =>better4
datain = Q66wP_top_mute;
datain.th{1}(16,:)=datain.th{1}(15,:)+0.02;
[new_bp]=new_bandpass(datain,8,12,700,1000);
kword=1;
bad=[48 56];
[killed]=kill(new_bp,kword,bad);
flg=1;
[deleted]=pack_good(killed,flg);
[mute]=mute_Hann(deleted,1,16,16,0.05);
Q66wP_better4=mute;
[scaled]=time_scale(deleted,1.7);
Q66wP_sca_better4=scaled;
% [agc_mute]=agc(scaled,0.08,1);
% vaplot(agc_mute,0.0,0.8,1,192,1,1,2,'k',1,20,8,1,0,'all4_enhance.ps',...
% '4Q66W3 ND z  enhanced(BP 8 12 700 1000); P removed; cos notch5...
% scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% remove S
datain = Q66wP_sca_better4;
load s_pick.mat;
```

```
datain.th{1}(19,:)=s_pick(:);
datain.th{1}(17,:)=datain.th{1}(15,:)+0.02;
[plus_time]=add(datain,0.4);
[shift]=shft(plus_time,-1,19);
shift.th{1}(16,:)=0.375;
[filt]=medi_filt(shift,11);
[fimu]=mute_Hann(filt,1,16,16,0.05);
[sub]=subr(fimu,shift);
[un_align]=shft(sub,1,19);
[shiftup]=sft(un_align,1,192,-0.4);
[mute]=mute_Hann(shiftup,1,17,17,0.05);
[deleted]=pack_good(mute,1);
Q66wS_muteS = shiftup;
% [agc_mute]=agc(shiftup,0.08,1);
% vaplot(agc_mute,0.0,0.8,1,192,1,1,2,'k',1,20,8,1,0,'all4_S_...
%  removal.ps','4Q66W3 ND z S removed; enhanced; P removed; cos ...
% notch5; scal %1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FK and BP filter
load rejP.mat;
load rejS.mat;
datain=Q66wS_muteS;
datain.th{1}(17,:)=datain.th{1}(15,:)+0.02;
[filt2]=fkfilt(datain,rejP,600,24,1,-5,1);
[filt3]=fkfilt(filt2,rejS,600,24,1,-5,1);
[new_bp]=new_bandpass(filt3,50,71,150,220);
[mute]=mute_Hann(new_bp,1,17,17,0.05);
Q66wS_FKbp2 = filt3;
[agc_mute]=agc(mute,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,2,'k',1,20,8,1,0,'all5_FK.ps',...
'4Q66W3 ND z  scaled t 1.7; cos notch5; FK filtering(rejectS,P);...
S removed');
all5=mute;
save all5.mat all5
```

E.7 create 'near' corridor stack (sc_mute_corr10.m)

```
load all5.mat;
datain=all5;

poly2=[1 0.597 ;192 0.29; 192 0.19; 1 0.497];
[TWT]=shft(datain,1,15);
[wimu,scal]=window_mute4(TWT,poly2,0.03,1);

datain.dat{1} = wimu;

[corr_t1]=norm_stack(datain,scal,5);

Q66_corrstack10 = corr_t1;
save Q66_corrstack10.mat Q66_corrstack10;

[scal_dat]=scale_rec(datain);
[scal_corr]=scale_rec(corr_t1);
[dataout]=merge_traces2(scal_corr,scal_dat);

Q66a5_mute10= dataout;
save Q66a5_mute10.mat Q66a5_mute10;

% [agc_mute]=agc(dataout,0.08,1);
vaplot(dataout,0,1.3,1,198,1,1,2,'k',56,21,5,1,0,'Q66a5_mute10.ps',...
'4Q66W3 ND z; all5; TWT-shifted; BP 50 71 150 220; near BH (0.1s)');

vaplot(dataout,0,0.8,1,198,1,1,2,'k',56,21,7,1,0,'Q66a5_mute_bo10ps',...
'4Q66W3ND z; all5;TWT-shifted; BP 50 71 150 220;near BH (0.1s)');

vaplot(corr_t1,0,1.3,1,5,1,1,1,'k',1,5,7,1,0,'Q66_corrstack10.ps',...
'4Q66W3ND z; all5; TWT-shifted; BP 50 71 150 220; near BH (0.1s)');
```

E.8 create 'mid' corridor stack (sc_corr_mute12.m)

```
clear all;

load all5.mat;
datain=all5;
poly2=[1 0.775 ;192 0.475; 192 0.375; 1 0.675];
[TWT]=shft(datain,1,15);
[wimu,scal]=window_mute4(TWT,poly2,0.03,1);

datain.dat{1} = wimu;

[corr_t1]=norm_stack(datain,scal,5);

Q66_corrstack12 = corr_t1;
save Q66_corrstack12.mat Q66_corrstack12;

[scal_dat]=scale_rec(datain);
[scal_corr]=scale_rec(corr_t1);
[dataout]=merge_traces2(scal_corr,scal_dat);

Q66a5_mute12= dataout;
save Q66a5_mute12.mat Q66a5_mute12;
vaplot(dataout,0,1.2,1,198,1,1,2,'k',56,21,5,1,0,'Q66a5mute12.ps',...
'4Q66W3ND z; all5; TWT-shifted; BP 50 71 150 220; middle BH (0.1s)');

vaplot(dataout,0.2,1,1,198,1,1,2,'k',56,21,7,1,0,'Q66a5_mute_bo12.ps',...
'4Q66W3ND z; all5;TWT-shifted;BP 50 71 150 220;middle BH(0.1s)');

vaplot(corr_t1,0,1.3,1,5,1,1,1,'k',1,5,7,1,0,'Q66_corrstack12.ps',...
'4Q66W3ND z; all5; TWT-shifted; BP 50 71 150 220; middle BH (0.1s)');
```

E.9 create 'far' corridor stack (sc_corr_mute11.m)

```
clear all;

load all5.mat;
datain=all5;

dsi_start

% [mute]=mute_Hann(datain,1,17,17,0.05);
poly2=[1 0.95 ;192 0.65; 192 0.55; 1 0.85];
[TWT]=shft(datain,1,15);
[wimu,scal]=window_mute4(TWT,poly2,0.03,1);

datain.dat{1} = wimu;

[corr_t1]=norm_stack(datain,scal,5);

Q66_corrstack11 = corr_t1;
save Q66_corrstack11.mat Q66_corrstack11;

[scal_dat]=scale_rec(datain);
[scal_corr]=scale_rec(corr_t1);
[dataout]=merge_traces2(scal_corr,scal_dat);

Q66a5_mute11= dataout;
save Q66a5_mute11.mat Q66a5_mute11;

[agc_mute]=agc(TWT,0.08,1);
vaplot(dataout,0,1.3,1,198,1,1,2,'k',56,21,5,1,0,'Q66a5_mute11.ps',...
'4Q66W3ND z; all5; TWT-shifted; BP 50 71 150 220; far BH (0.1s)');

vaplot(dataout,0.4,1.2,1,198,1,1,2,'k',56,21,7,1,0,'Q66a5_mute_bo11.ps',...
'4Q66W3ND z; all5;TWT-shifted; BP 50 71 150 220; far BH (0.1s)');

vaplot(corr_t1,0,1.3,1,5,1,1,1,'k',1,5,7,1,0,'Q66_corrstack11.ps',...
'4Q66W3ND z; all5; TWT-shifted; BP 50 71 150 220; far BH (0.1s)');
```

E.10 creates plots (sc_plotPremove.m)

```
load Q66ND_z_raw.mat;
load nicepicks.mat;
datain = Q66ND_z_raw;
datain.th{1}(15,:)=nicepicks(:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mute first P-arrival picks.mat
load Q66ND_z_raw.mat;
datain = Q66ND_z_raw;
datain.th{1}(16,:) = datain.th{1}(15,:)+0.005;
% makes sure P wave is muted
headw1= 16;
headw2= 17;
mute_flg= 1;
taper = 0.005;
[dataout]=mute_taper(datain,mute_flg,headw1,headw2,taper)
Q66ND_wP_raw = dataout;
% save Q66ND_wP_raw.mat Q66ND_wP_raw

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% applies cos_notch5
[dataout] = cos_notch(Q66ND_wP_raw,50,60,60,70);
[dataout] = cos_notch(dataout,171,181,181,191);
[dataout] = cos_notch(dataout,291,301,301,311);
[dataout] = cos_notch(dataout,411,421,421,431);
[dataout] = cos_notch(dataout,531,541,541,541);
[dataout] = cos_notch(dataout,651,661,661,671);
[dataout] = cos_notch(dataout,770,780,780,790);
[dataout] = cos_notch(dataout, 891,901,901,911);
Q66wP_cos_notch5 = dataout;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% applies top mute
datain = Q66wP_cos_notch5;
datain.th{1}(15,:)=nicepicks;
[scal]=time_scale(Q66wP_cos_notch5,1.7);
[add_P]=add(scal,0.05); [align_P]=shft(add_P,-1,15);
align_P.th{1}(16,:)=0.075; align_P.th{1}(17,:)=0.175;
[mute]=mute_Hann(align_P,1,16,16,0.01);
[un_align]=shft(mute,1,15); [shiftup]=sft(un_align,1,192,-0.05);
[mut_top]=mute_Hann(shiftup,1,15,17,0.01);
```

```
[agc_mute]=agc(add_P,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'P_addP.ps',...
'4Q66W3ND z add; scal 1.7, agc 0.08;');

[agc_mute]=agc(align_P,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'P_alignP.ps',...
'4Q66W3ND z align; scal 1.7, agc 0.08;');

[agc_mute]=agc(mute,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'P_mute.ps',...
'4Q66W3ND z mute; scal 1.7, agc 0.08;');

[agc_mute]=agc(un_align,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'P_unalign.ps',...
'4Q66W3ND z un-align; scal 1.7, agc 0.08;');

[agc_mute]=agc(mut_top,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'P_mut_top.ps',...
'4Q66W3ND z  P removed; scal 1.7, agc 0.08;');

[agc_mute]=agc(shiftup,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'P_shiftup',...
'4Q66W3ND z  shift up; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sc_plotSremove
load Q66ND_z_raw.mat;
load nicepicks.mat;
datain = Q66ND_z_raw;
datain.th{1}(15,:)=nicepicks(:);
load Q66ND_z_raw.mat;
datain = Q66ND_z_raw;
datain.th{1}(16,:) = datain.th{1}(15,:)+0.005;
headw1= 16;
headw2= 17;
mute_flg= 1;
taper = 0.005;
[dataout]=mute_taper(datain,mute_flg,headw1,headw2,taper)
Q66ND_wP_raw = dataout;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% applies cos_notch5
[dataout] = cos_notch(Q66ND_wP_raw,50,60,60,70);
[dataout] = cos_notch(dataout,171,181,181,191);
[dataout] = cos_notch(dataout,291,301,301,311);
[dataout] = cos_notch(dataout,411,421,421,431);
[dataout] = cos_notch(dataout,531,541,541,541);
[dataout] = cos_notch(dataout,651,661,661,671);
[dataout] = cos_notch(dataout,770,780,780,790);
[dataout] = cos_notch(dataout, 891,901,901,911);
Q66wP_cos_notch5 = dataout;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% applies top mute
datain = Q66wP_cos_notch5;
 datain.th{1}(15,:)=nicepicks;
[add_P]=add(datain,0.05);
[align_P]=shft(add_P,-1,15);
align_P.th{1}(16,:)=0.075;
align_P.th{1}(17,:)=0.175;
[mute]=mute_Hann(align_P,1,16,16,0.01);
[un_align]=shft(mute,1,15);
[shiftup]=sft(un_align,1,192,-0.05);
Q66wP_top_mute=shiftup;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% enhance data =>better4
datain = Q66wP_top_mute;
datain.th{1}(16,:)=datain.th{1}(15,:)+0.02;
[new_bp]=new_bandpass(datain,8,12,700,1000);
[mute]=mute_Hann(new_bp,1,16,16,0.05);
Q66wP_better4=mute;
[scaled]=time_scale(new_bp,1.7);
Q66wP_sca_better4=scaled;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% remove S
datain = Q66wP_sca_better4;
load s_pick.mat;
datain.th{1}(19,:)=s_pick(:);
datain.th{1}(17,:)=datain.th{1}(15,:)+0.02;

[plus_time]=add(datain,0.4);
```

```
[shift]=shft(plus_time,-1,19);
shift.th{1}(16,:)=0.375;

[filt]=medi_filt(shift,11);
[fimu]=mute_Hann(filt,1,16,16,0.05);
[sub]=subr(fimu,shift);
[un_align]=shft(sub,1,19);
[shiftup]=sft(un_align,1,192,-0.4);
[mute]=mute_Hann(shiftup,1,17,17,0.05);
[deleted]=pack_good(mute,1);

[agc_mute]=agc(scaled,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_before.ps',...
'4Q66W3ND z before; scal 1.7, agc 0.08;');

[agc_mute]=agc(plus_time,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_addS.ps',...
'4Q66W3ND z add; scal 1.7, agc 0.08;');

[agc_mute]=agc(shift,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_shift.ps',...
'4Q66W3ND z align; scal 1.7, agc 0.08;');

[agc_mute]=agc(filt,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_filt.ps',...
'4Q66W3ND z filtered; scal 1.7, agc 0.08;');

[agc_mute]=agc(fimu,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_fimu.ps',...
'4Q66W3ND z filtered+muted; scal 1.7, agc 0.08;');

[agc_mute]=agc(sub,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_sub.ps',...
'4Q66W3ND z subtracted; scal 1.7, agc 0.08;');

[agc_mute]=agc(un_align,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_un_align.ps'...
,'4Q66W3ND z  un-align; scal 1.7, agc 0.08;');

[agc_mute]=agc(shiftup,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4.5,'k',1,59,3,1,0,'S_shiftup',...
'4Q66W3ND z  shift up; scal 1.7, agc 0.08;');
```

E.11 creates plot (sc_plot_FK.m)

```
 load Q66ND_z_raw.mat;
 load nicepicks.mat;
 datain = Q66ND_z_raw;
 datain.th{1}(15,:)=nicepicks(:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mute first P-arrival picks.mat
load Q66ND_z_raw.mat;
datain = Q66ND_z_raw;
datain.th{1}(16,:) = datain.th{1}(15,:)+0.005;
headw1= 16;
headw2= 17;
mute_flg= 1;
taper = 0.005;
[dataout]=mute_taper(datain,mute_flg,headw1,headw2,taper)
Q66ND_wP_raw = dataout;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% applies cos_notch5
[dataout] = cos_notch(Q66ND_wP_raw,50,60,60,70);
[dataout] = cos_notch(dataout,171,181,181,191);
[dataout] = cos_notch(dataout,291,301,301,311);
[dataout] = cos_notch(dataout,411,421,421,431);
[dataout] = cos_notch(dataout,531,541,541,541);
[dataout] = cos_notch(dataout,651,661,661,671);
[dataout] = cos_notch(dataout,770,780,780,790);
[dataout] = cos_notch(dataout, 891,901,901,911);
Q66wP_cos_notch5 = dataout;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% applies top mute
datain = Q66wP_cos_notch5;
datain.th{1}(15,:)=nicepicks;
[add_P]=add(datain,0.05);
[align_P]=shft(add_P,-1,15);
align_P.th{1}(16,:)=0.075; align_P.th{1}(17,:)=0.175;
[mute]=mute_Hann(align_P,1,16,16,0.01);
[un_align]=shft(mute,1,15);
[shiftup]=sft(un_align,1,192,-0.05);
Q66wP_top_mute=shiftup;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% enhance data =>better4
datain = Q66wP_top_mute;
datain.th{1}(16,:)=datain.th{1}(15,:)+0.02;
[new_bp]=new_bandpass(datain,8,12,700,1000);
kword=1;
bad=[48 56];
[killed]=kill(new_bp,kword,bad);
flg=1;
[deleted]=pack_good(killed,flg);
[mute]=mute_Hann(deleted,1,16,16,0.05);
Q66wP_better4=mute;
[scaled]=time_scale(deleted,1.7);
Q66wP_sca_better4=scaled;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% remove S
datain = Q66wP_sca_better4;
load s_pick.mat;
datain.th{1}(19,:)=s_pick(:);
datain.th{1}(17,:)=datain.th{1}(15,:)+0.02;
[plus_time]=add(datain,0.4);
[shift]=shft(plus_time,-1,19);
shift.th{1}(16,:)=0.375;
[filt]=medi_filt(shift,11);
[fimu]=mute_Hann(filt,1,16,16,0.05);
[sub]=subr(fimu,shift);
[un_align]=shft(sub,1,19);
[shiftup]=sft(un_align,1,192,-0.4);
[mute]=mute_Hann(shiftup,1,17,17,0.05);
[deleted]=pack_good(mute,1);
Q66wS_muteS = shiftup;

[agc_mute]=agc(shiftup,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4,'k',1,59,3,1,0,'FK_before.ps',...
'4Q66W3ND z before; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FK and BP filter
load rejP.mat;
load rejS.mat;
datain=Q66wS_muteS;
```

```
datain.th{1}(17,:)=datain.th{1}(15,:)+0.02;
[filt2]=fkfilt(datain,rejP,600,24,1,-5,1);
[filt3]=fkfilt(filt2,rejS,600,24,1,-5,1);

[new_bp]=new_bandpass(filt3,50,71,150,220);
[mute]=mute_Hann(new_bp,1,17,17,0.05);
Q66wS_FKbp2 = filt3;

[agc_mute]=agc(filt3,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4,'k',1,59,3,1,0,'filtFK.ps',...
'4Q66W3ND z FK filtered; scal 1.7, agc 0.08;');

[agc_mute]=agc(new_bp,0.08,1);
vaplot(agc_mute,0.0,0.8,1,192,1,1,4,'k',1,59,3,1,0,'filtBP.ps',...
'4Q66W3ND z FK+BP filtered (50 71 150 220); scal 1.7, agc 0.08;');
```

E.12 bandpass filter test (sc_bp_filter_Test.m)

```
clear all

load Q66ND_wP_z.mat;
datain = Q66ND_wP_z;

dsi_start

[bp0]=bandpass(datain,8,12,30,45);
[bp1]=bandpass(datain,8,12,50,71);
[bp2]=bandpass(datain,21,30,50,71);
[bp3]=bandpass(datain,21,30,70,100);
[bp4]=bandpass(datain,50,71,90,130);
[bp5]=bandpass(datain,70,100,150,220);
[bp6]=bandpass(datain,90,130,250,355);
[bp7]=bandpass(datain,150,220,350,495);
[bp8]=bandpass(datain,250,355,500,710);
[bp9]=bandpass(datain,350,495,650,920);
[bp10]=bandpass(datain,500,725,900,1272);
[bp11]=bandpass(datain,650,920,1100,1560);
[bp12]=bandpass(datain,900,1272,1500,2125);

[bp_test]=merge_files(bp0,bp1);
[bp_test]=merge_files(bp_test,bp2);
[bp_test]=merge_files(bp_test,bp3);
[bp_test]=merge_files(bp_test,bp4);
[bp_test]=merge_files(bp_test,bp5);
[bp_test]=merge_files(bp_test,bp6);
[bp_test]=merge_files(bp_test,bp7);
[bp_test]=merge_files(bp_test,bp8);
[bp_test]=merge_files(bp_test,bp9);
[bp_test]=merge_files(bp_test,bp10);
[bp_test]=merge_files(bp_test,bp11);
[bp_test]=merge_files(bp_test,bp12);

% Q66ND_z_bp = bp_test;
% save Q66ND_z_bp.mat Q66ND_z_bp
dsi_end
```

E.13 creates plot (sc_plot_BP.m)

```
load Q66ND_z_bp.mat;
load Q66ND_wP_z.mat;
[Q66ND_wP_z_agc]=agc(Q66ND_wP_z,0.08,1);
[Q66ND_zBP_agc]=agc(Q66ND_z_bp,0.08,1);

vaplot(Q66ND_wP_z_agc,0.0,0.8,1,192,1,1,4,'k',1,59,3,1,0,...
'Q66ND_before.ps','4Q66W3ND z  agc 0.08');

vaplot(Q66ND_zBP_agc,0.0,0.8,1,192,1,1,4,'k',1,59,3,1,0,...
'Q66ND_b0.ps','4Q66W3ND z  agc 0.08;  bp 8,12,30,45');

vaplot(Q66ND_zBP_agc,0.0,0.8,1,192,1,1,4,'k',1,59,3,3,0,...
'Q66ND_b2.ps','4Q66W3ND z  agc 0.08;  bp 21,30,50,71');

vaplot(Q66ND_zBP_agc,0.0,0.8,1,192,1,1,4,'k',1,59,3,5,0,...
'Q66ND_b4.ps','4Q66W3ND z  agc 0.08;  bp 50,71,90,130');

vaplot(Q66ND_zBP_agc,0.0,0.8,1,192,1,1,4,'k',1,59,3,6,0,...
'Q66ND_b5.ps','4Q66W3ND z  agc 0.08;  bp 70,100,150,220');

vaplot(Q66ND_zBP_agc,0.0,0.8,1,192,1,1,4,'k',1,59,3,7,0,...
'Q66ND_b6.ps','4Q66W3ND z  agc 0.08;  bp 90,130,250,355');
```

E.14 creates geometry (sc_plot_geom.m)

```
clear all
load test.txt;
A=test;
% Label=A(:,1);
East=A(:,4);
North=A(:,5);
Elev=A(:,6);
plot3(East,North,Elev,'b.'), hold on,

load cor_rec_copy.txt;
% xlsread cor_rec.xls
B=cor_rec_copy;
% Label=A(:,1);
E=B(:,3);
N=B(:,4);
El=B(:,5);
plot3(E,N,El,'r+'), hold on,

K=ones(2,6);
K(1,:)=test(1,:);
K(2,:)=test(214,:);
X=K(:,4);
Y=K(:,5); Z=K(:,6);
plot3(X,Y,Z,'k'), hold on,

grid on,

% title('BH 4Q66W3');
xlabel('Northing');
ylabel('Easting');
zlabel('Elevation');
hold off
set(gca,'CameraViewAngleMode','Manual')
```

Appendix F

Processing Scripts FFM001 near offset

F.1 convert from .seg2 to .mat (get_data.m)

```
vsp1=seg2mat2('NeraDyn.txt',1,49);
vsp1.fh{8}=0.00025;
vsp1.fh{10}=3;
raw=vsp1;
save raw.mat raw
```

F.2 write primary trace headers and sort (load_data.m)

```
clear all;
load depth.txt;
load raw.mat;
temp1=zeros(12001,24);
out.fh=raw.fh;
for k=1:raw.fh{12}
    out.th{k}=raw.th{k}(:,2:25);
    out.dat{k}=temp1;
end
out.fh{1}=864;
% assign original channel number and comp
% level 1 is on trace 22-24, level 2 is on 19-21 etc
orig=[8 8 8 7 7 7 6 6 6 5 5 5 4 4 4 3 3 3 2 2 2 1 1 1];
% comp # (3: vertical 1: 1st horizontal 2: 2nd horizontal)
comp=[3 1 2 3 1 2  3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2];
for jj=1:out.fh{12}
    out.dat{jj}=raw.dat{jj}(:,2:25);
    out.th{jj}(5,:)=orig;
    out.th{jj}(6,:)=comp;
end;
out.fh{13}=24; dl=depth;
for COUNT=1:out.fh{12}
    adl=dl(COUNT,2);
    wld1=[adl+70 adl+70 adl+70 adl+60 adl+60 adl+60 adl+50 adl+50];
    wld2=[adl+50 adl+40 adl+40 adl+40 adl+30 adl+30 adl+30 adl+20];
    wld3=[adl+20 adl+20 adl+10 adl+10 adl+10 adl adl adl];
    wld=[wld1'; wld2'; wld3'];
    for tr = 1:out.fh{13}
       out.th{COUNT}(56,tr)=wld(tr);
       out.th{COUNT}(10,tr)=wld(tr);
    end;
end;
FM1ND_raw=out;
save FM1ND_raw.mat FM1ND_raw

[xx]=sortrec(out,6,56);
FM1ND_sort=xx
save FM1ND_sort.mat FM1ND_sort
```

F.3 write trace headers (sc_write_th.m)

```
clear all;
load Receivers.txt;

% it is a matrix where rec station # are in colomn 1,
% wireline depth in colomn 2, UTM east in 7; UTM north in 8
% and elevation in colomn 9

load FM1ND_sort.mat;
data=FM1ND_sort;
A= Receivers;
for i=1:3,
    data.th{i}(35,:)=A(:,8);
    data.th{i}(4,:)=data.th{i}(6,:);
    data.th{i}(35,:)=A(:,8);
    data.th{i}(37,:)=A(:,7);
    data.th{i}(39,:)=A(:,9);
    data.th{i}(55,:)=A(:,2);
    data.th{i}(55,:)=A(:,1);
end
data.fh{8}=0.00025;
FM1NDsort_th = data;
save FM1NDsort_th.mat FM1NDsort_th
```

F.4 trace editing (sc_filt1.m)

```
clear all;
load FM1NDsort_th.mat;
datain = FM1NDsort_th
for i=1:3,
    datain.th{i}(17,:)=1.2;
    dataout.th{i}(:,:)=datain.th{i}(:,:);
end
[temp]=shft(datain,-1,17);
dataout.fh=datain.fh;
dataout.fh{9}=0;
dataout.fh{10}=1;
dataout.fh{7}=4000;
for i=1:3
    dataout.dat{i}(:,:)=temp.dat{i}(1:4000,:);
end

FM1ND_short1 = dataout;
save FM1ND_short1.mat FM1ND_short1

[filt] = new_bandpass(dataout,40,45,1000,2000);
FM1ND_filt = filt;
save FM1ND_filt.mat FM1ND_filt
```

F.5 write missing source locations (sc_more_tha.m)

```
clear all
dsi_start
load FM1ND_z_raw.mat;
% subset of FM1ND_filt, containing z-component only
datain=FM1ND_z_raw;

[sort1]=sortrec_many(datain,6,2);
load source_loc.txt
% it is an [5 x 3] matrix where UTM north is in column 1, UTM east in 2
% UTM elevation in column 3
A= source_loc;
dataout=sort1;
% shot 1
dataout.th{1}(29,1:112)=A(1,1);     % stores the north coordinates
dataout.th{1}(31,1:112)=A(1,2);     % stores the east coordinates
dataout.th{1}(33,1:112)=A(1,3);     % stores the elevation coordinates
% shot 2
dataout.th{1}(29,113:184)=A(2,1);   % stores the north coordinates
dataout.th{1}(31,113:184)=A(2,2);   % stores the east coordinates
dataout.th{1}(33,113:184)=A(2,3);   % stores the elevation coordinates
% shot 3
dataout.th{1}(29,185:232)=A(3,1);   % stores the north coordinates
dataout.th{1}(31,185:232)=A(3,2);   % stores the east coordinates
dataout.th{1}(33,185:232)=A(3,3);   % stores the elevation coordinates
% shot 4
dataout.th{1}(29,233:264)=A(4,1);   % stores the north coordinates
dataout.th{1}(31,233:264)=A(4,2);   % stores the east coordinates
dataout.th{1}(33,233:264)=A(4,3);   % stores the elevation coordinates
% shot 5
dataout.th{1}(29,265:288)=A(5,1);   % stores the north coordinates
dataout.th{1}(31,265:288)=A(5,2);   % stores the east coordinates
dataout.th{1}(33,265:288)=A(5,3);   % stores the elevation coordinates

[sort2]=sortrec_many(dataout,3,56);

FM1ND_z_cor = sort2;
save FM1ND_z_cor.mat FM1ND_z_cor
dsi_end
```

F.6 processing applied to the preprocessed data (sc_new_all.m)

```
clear all
dsi_start
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sc_top-mute
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load FM1ND_z_cor.mat;
load pickP.mat;
datain = FM1ND_z_cor;
[scal]=time_scale(datain,1.7);
[agc_mute]=agc(scal,0.08,1);
% vaplot(agc_mute,0.0,0.8,1,288,1,1,2,'k',13,32,8,1,0,'FM1ND_z_raw.ps',...
'FFM001 ND z: raw;
scal 1.7, agc 0.08;');
datain.th{1}(15,:)=pickP(:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mute first P-arrival
datain.th{1}(16,:) = datain.th{1}(15,:)+0.01; %makes sure P wave is muted
[dataout]=mute_taper(datain,1,16,16,0.005);
FM1ND_wP_raw =dataout;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sc_notch2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
datain = FM1ND_wP_raw;
[dataout] = new_bandpass(datain,15,20,1000,2000);
[dataout] = cos_notch(dataout,15,18,42,47);          % 18-42
[dataout] = cos_notch(dataout,50,60,60,70);          % 60
[dataout] = cos_notch(dataout,171,181,181,191);      % 180
[dataout] = cos_notch(dataout,291,301,301,311);      % 300
[dataout] = cos_notch(dataout,351,361,361,371);      % 361
[dataout] = cos_notch(dataout,411,421,422,432);      % 422
[dataout] = cos_notch(dataout,529,539,543,553);      % 541
[dataout] = cos_notch(dataout,652,662,662,672);      % 662
[dataout] = cos_notch(dataout,711,721,721,731);      % 719
[dataout] = cos_notch(dataout,771,781,781,791);      % 781
[dataout] = cos_notch(dataout, 891,901,901,911);     % 900
FM1ND_wP_notch2 = dataout;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sc_Pmute2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
datain = FM1ND_wP_notch2;
datain.th{1}(15,:)=pickP(:);
% applies top P-mute
[add_P]=add(datain,0.5);
[align_P]=shft(add_P,-1,15);
align_P.th{1}(16,:)=0.5;
align_P.th{1}(17,:)=0.525;
[fimu]=mute_Hann(align_P,1,17,17,0.05);
[un_align]=shft(fimu,1,15);
[shiftup]=shft(un_align,-1,16);
FM1NDw_Pmute2=shiftup;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sc_Smute
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
datain = FM1NDw_Pmute2;
datain.th{1}(15,:)=pickP(:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load pickS.mat;
datain.th{1}(16,:)=pickS(:);
datain.th{1}(17,:)=datain.th{1}(15,:)+0.02;
[plus_time]=add(datain,0.6);
[shift]=shft(plus_time,-1,16);
% shift.th{1}(19,:)=0.375;
% shift.th{1}(17,:)=0.475;
% [mutse]=mute_Hann(shift,3,16,17,0.01);
[filt]=medi_filt(shift,11);
[sca_fi]=time_scale(filt,1.7);
[fimu]=mute_Hann(filt,1,16,16,0.05);
[sca_mu]=time_scale(fimu,1.7);
[sub]=subr(fimu,shift);
sub.th{1}(19,:)=0.6;
[sca_su]=time_scale(sub,1.7);
[un_align]=shft(sub,1,16);
[shiftup]=shft(un_align,-1,19);
[mute]=mute_Hann(shiftup,1,17,17,0.05);
[deleted]=pack_good(mute,1);
FM1ND_muteS = shiftup;
FM1ND_new_all = shiftup;
save FM1ND_new_all.mat FM1ND_new_all

[scal]=time_scale(shiftup,1.7); [agc_mute]=agc(scal,0.02,1);
vaplot(agc_mute,0.0,0.8,1,288,1,1,2,'k',13,32,8,1,0,'FM1ND_new_all.ps',...
```

```
'FFM001 ND z: S mute; scal 1.7, agc 0.08;');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sc_BP_plot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
datain=FM1ND_muteS;
flip=datain;
datain.th{1}(16,:) = datain.th{1}(15,:)+0.02;
% match direction
ntra=datain.fh{1}; dataout=datain;
dataout.th{1}(:,1)=datain.th{1}(:,ntra);
dataout.th{1}(12,1)=datain.th{1}(12,1);
dataout.dat{1}(:,1)=datain.dat{1}(:,ntra);
for i=1:ntra-1
    dataout.th{1}(:,i+1)=datain.th{1}(:,ntra-i);
    dataout.dat{1}(:,i+1)=datain.dat{1}(:,ntra-i);
end
flip=dataout;
[dataout] = new_bandpass(dataout,70,100,200,300);
bp=dataout;
FM1ND_new_flip=flip;
save FM1ND_new_flip.mat FM1ND_new_flip;
[dataout]=mute_Hann(dataout,1,16,16,0.02);

[agc_mute]=agc(dataout,0.02,1);
vaplot(agc_mute,0,1,1,288,1,1,3,'k',56,32,7,1,0,'FM1ND_newa_BP3a.ps',...
'FFM001 ND z: BP(70 100 200 300); scal 1.7, agc 0.02;');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[TWT]=shft(flip,1,15);
FM1ND_new_TWT=TWT;
save FM1ND_new_TWT.mat FM1ND_new_TWT;
[TWT]=shft(dataout,1,15);
FM1ND_new_TWTBP=TWT;
save FM1ND_new_TWTBP.mat FM1ND_new_TWTBP;

[agc_mute]=agc(TWT,0.02,1);
vaplot(agc_mute,0.2,1.2,1,288,1,1,3,'k',56,32,7,1,0,'FM1ND_new_TWTBP.ps',...
'FFM001 ND z: BP(70 100 200 300); scal 1.7, agc 0.02;');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dsi_end
```

F.7 create a corridor stack of the whole dataset (sc_corr_new_all.m)

```
load FM1ND_new_TWTBP.mat
datain=FM1ND_new_TWTBP;

dsi_start

mute=datain;
% set window corners
poly=[1 0.69150+0.03; 1 1.3475; 288 1.1075; 288 0.215+0.03];
[wimu,scal]=window_mute4(datain,poly,0.01,1);

mute.dat{1} = wimu;

[corr_t1]=norm_stack(mute,scal,5);
[agc_corr]=agc(corr_t1,0.08,1);
cor_new_all = corr_t1;
save cor_new_all.mat cor_new_all;
cor_near = agc_corr;

[scal_dat]=scale_rec(mute);
[scal_corr]=scale_rec(corr_t1);
[dataout]=merge_traces2(scal_corr,scal_dat);

merge_near = dataout;

[agc_mute]=agc(dataout,0.02,1);
[agc_corr]=agc(corr_t1,0.08,1);
vaplot(agc_mute,0.2,1.4,1,293,1,1,2,'k',56,32,5.5,1,0,'corr_new_all.ps',...
'FFM001 ND z; BP');

dsi_end
```

F.8 create corridor stack (sc_corr_40... .m)

```
load FM1ND_new_flip.mat;
datain=FM1ND_new_flip;
win=abs((2*datain.th{1}(15,288)-2*datain.th{1}(15,1))/287*39);
a=0.05;
% dist from first breaks for near, 0.1 for near1, 0.15 for near2
% 0.2 for mid1, 0.25 for mid2, 0.3 for mid3,
% 0.4 for far1, 0.45 for far2, 0.5 for far3

dsi_start

[TWT]=shft(datain,1,15);
mute=TWT;
% set window corners
poly=[1 0.69150+a; 1 0.6915+a+win; 288 0.215+a+win; 288 0.215+a];
[wimu,scal]=window_mute4(TWT,poly,0.01,1);

mute.dat{1} = wimu;

[corr_t1]=norm_stack(mute,scal,5);
[agc_corr]=agc(corr_t1,0.08,1);
cor_new_near1 = corr_t1;
save cor_new_near1.mat cor_new_near1;
cor_near = agc_corr;
% save cor_near.mat cor_near;

[scal_dat]=scale_rec(mute);
[scal_corr]=scale_rec(corr_t1);
[dataout]=merge_traces2(scal_corr,scal_dat);

merge_near = dataout;

[agc_mute]=agc(dataout,0.08,1);
[agc_corr]=agc(corr_t1,0.08,1);
vaplot(agc_mute,0.2,1.4,1,293,1,1,4,'k',56,32,5.5,1,0,'merge_40.ps',...
'FFM001 ND z; "... borehole 40 traces"; no BP');

dsi_end
```

F.9 merges all corridor stacks into one file (sc_new_comp_corr2.m)

```
load cor_new_near1.mat;
load cor_new_near1a.mat;
load cor_new_near2.mat;
load cor_new_mid1.mat;
load cor_new_mid2.mat;
load cor_new_mid3.mat;
load cor_new_far1.mat;
load cor_new_far2.mat;
load cor_new_far3.mat;

dsi_start
[scal_1a]=(cor_new_near1);
[scal_1b]=(cor_new_near1a);
[scal_1c]=(cor_new_near2);

[scal_2a]=(cor_new_mid1);
[scal_2b]=(cor_new_mid2);
[scal_2c]=(cor_new_mid3);


[scal_3a]=(cor_new_far1);
[scal_3b]=(cor_new_far2);
[scal_3c]=(cor_new_far3);

[merge1]=merge_traces3(scal_1a,scal_1b);
[merge1]=merge_traces3(merge1,scal_1c);
[merge1]=merge_traces3(merge1,scal_2a);
[merge1]=merge_traces3(merge1,scal_2b);
[merge1]=merge_traces3(merge1,scal_2c);
[merge1]=merge_traces3(merge1,scal_3a);
[merge1]=merge_traces3(merge1,scal_3b);
[merge1]=merge_traces3(merge1,scal_3c);

New_comp_corr2= merge1;
save New_comp_corr2.mat New_comp_corr2;

vaplot(merge1,0.2,1.2,1,45,1,0,1,'k',1,10,8,1,0,'New_comp_corr2.ps',...
'FFM001 ND z; all stacks (0.1s)');
dsi_end
```

F.10 creates plot (sc_plot_fi4)

```
load FM1ND_z_raw;
load pickP.mat;
dsi_start

datain = FM1ND_z_raw;
datain.th{1}(15,:)=pickP(:);
datain.th{1}(16,:)=datain.th{1}(15,:)+0.01;
% match direction
ntra=288;
dataout=datain;
dataout.th{1}(:,1)=datain.th{1}(:,ntra);
dataout.th{1}(12,1)=datain.th{1}(12,1);
dataout.dat{1}(:,1)=datain.dat{1}(:,ntra);
for i=1:ntra-1
    dataout.th{1}(:,i+1)=datain.th{1}(:,ntra-i);
    dataout.dat{1}(:,i+1)=datain.dat{1}(:,ntra-i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mute first P-arrival picks.mat
% makes sure P wave is muted
[scal]=time_scale(dataout,1.7);
[agc_mute]=agc(scal,0.08,1);
[dataout]=mute_Hann(agc_mute,1,16,16,0.005);

vaplot(dataout,0.0,0.8,1,288,1,1,4.5,'k',56,44,3,1,0,'input41.ps',...
'FFM001 ND z: before filtering; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% notch5
[dataout] = new_bandpass(dataout,50,60,1000,2000);
[dataout] = cos_notch(dataout,15,18,50,60);         % 18-42
[dataout] = cos_notch(dataout,54,60,60,64);         % 60
[dataout] = cos_notch(dataout,177,181,181,185);     % 180
[dataout] = cos_notch(dataout,297,301,301,305);     % 300
[dataout] = cos_notch(dataout,357,361,361,365);     % 361
[dataout] = cos_notch(dataout,418,422,422,426);     % 422
[dataout] = cos_notch(dataout,537,541,542,546);     % 541
[dataout] = cos_notch(dataout,660,662,662,666);     % 662
[dataout] = cos_notch(dataout,717,721,721,725);     % 719
[dataout] = cos_notch(dataout,777,781,781,785);     % 781
```

```
[all] = cos_notch(dataout, 897,901,901,905);        % 900
[scal]=time_scale(all,1.7);

[agc_mute]=agc(scal,0.08,1);
[mut]=mute_Hann(agc_mute,1,16,16,0.005);
vaplot(mut,0.0,0.8,1,288,1,1,4.5,'k',56,44,3,1,0,'filt41.ps',...
'FFM001 ND z: 1st filtering step applied; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% extract noisy traces(207-240), filter differently
[noise]=subset(datain,48,81,0,1,1,1);
[dataout] = new_bandpass(noise,1,2,240,300);
[dataout] = new_bandpass(dataout,50,60,1000,2000);
[dataout,spec120] = cos_notch_test1(dataout,110,117,132,140);
[dataout,spec360] = cos_notch_test1(dataout,330,340,400,410);
[dataout] = new_bandpass(dataout,1,2,240,300);
all.dat{1}(:,48:81)=dataout.dat{1}(:,:);            % back together
[scal]=time_scale(all,1.7);

[agc_mute]=agc(scal,0.081,1);
[mut]=mute_Hann(agc_mute,1,15,15,0.005);
vaplot(mut,0.0,0.8,1,288,1,1,4.5,'k',56,44,3,1,0,'filt42.ps',...
'FFM001 ND z: 2nd filtering step; scal 1.7, agc 0.08;');
```

F.11 creates plot (sc_plot_remR.m)

```
clear all
load FM1ND_z_raw;
load pickP.mat;
dsi_start

datain = FM1ND_z_raw;
datain.th{1}(15,:)=pickP(:);
datain.th{1}(16,:)=datain.th{1}(15,:)+0.01;
%match direction
ntra=288; dataout=datain;
dataout.th{1}(:,1)=datain.th{1}(:,ntra);
dataout.th{1}(12,1)=datain.th{1}(12,1);
    dataout.dat{1}(:,1)=datain.dat{1}(:,ntra);
for i=1:ntra-1
    dataout.th{1}(:,i+1)=datain.th{1}(:,ntra-i);
    dataout.dat{1}(:,i+1)=datain.dat{1}(:,ntra-i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% notch5
[dataout] = new_bandpass(dataout,50,60,1000,2000);
[dataout] = cos_notch(dataout,15,18,50,60);         % 18-42
[dataout] = cos_notch(dataout,54,60,60,64);         % 60
[dataout] = cos_notch(dataout,177,181,181,185);     % 180
[dataout] = cos_notch(dataout,297,301,301,305);     % 300
[dataout] = cos_notch(dataout,357,361,361,365);     % 361
[dataout] = cos_notch(dataout,418,422,422,426);     % 422
[dataout] = cos_notch(dataout,537,541,542,546);     % 541
[dataout] = cos_notch(dataout,660,662,662,666);     % 662
[dataout] = cos_notch(dataout,717,721,721,725);     % 719
[dataout] = cos_notch(dataout,777,781,781,785);     % 781
[all] = cos_notch(dataout, 897,901,901,905);        % 900

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% extract noisy traces(207-240), filter differently
[noise]=subset(datain,48,81,0,1,1,1); [dataout] =
new_bandpass(noise,1,2,240,300); [dataout] =
new_bandpass(dataout,50,60,1000,2000);
[dataout,spec120] = cos_notch_test1(dataout,110,117,132,140);    % 120
[dataout,spec360] = cos_notch_test1(dataout,330,340,400,410);    % 370
[dataout] = new_bandpass(dataout,1,2,240,300);
all.dat{1}(:,48:81)=dataout.dat{1}(:,:);            % back together
```

```
[scal]=time_scale(all,1.7);

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% P-mute2
[mut]=mute_taper(scal,1,16,16,0.005);

[add_P]=add(mut,0.5);
[align_P]=shft(add_P,-1,15);

[agc_mute]=agc(align_P,0.08,1);
vaplot(agc_mute,0.4,0.75,1,288,1,1,4.5,'k',56,44,3,1,0,'P_align.ps',...
'FFM001 ND z: aligned first breaks; scal 1.7, agc 0.08;');

vaplot(agc_mute,0.4,1.2,1,288,1,1,4.5,'k',56,30,8,1,2,'P_align.eps',...
'FFM001 ND z: aligned first breaks; scal 1.7, agc 0.08;');

align_P.th{1}(16,:)=0.5;
align_P.th{1}(17,:)=0.525;
[fimu]=mute_Hann(align_P,1,17,17,0.05);

[agc_mute]=agc(fimu,0.08,1);
vaplot(agc_mute,0.4,0.75,1,288,1,1,4.5,'k',56,44,3,1,0,'P_fimu.ps',...
'FFM001 ND z: muted; scal 1.7, agc 0.08;');

vaplot(agc_mute,0.4,1.2,1,288,1,1,4.5,'k',56,30,8,1,2,'P_fimu.eps',...
'FFM001 ND z: muted; scal 1.7, agc 0.08;');

[un_align]=shft(fimu,1,15);
[shiftup]=shft(un_align,-1,16);
[agc_mute]=agc(shiftup,0.08,1);
vaplot(agc_mute,0.0,0.8,1,288,1,1,4.5,'k',56,44,3,1,0,'P_shiftup.ps',...
'FFM001 ND z: after muting; scal 1.7, agc 0.08;');

vaplot(agc_mute,0.0,0.8,1,288,1,1,4.5,'k',56,30,8,1,2,'P_shiftup.eps',...
'FFM001 ND z: after muting; scal 1.7, agc 0.08;');

dsi_end
```

F.12 creates plot (sc_plot_remS1.m)

```
clear all
load FM1ND_z_raw;
load pickP.mat;
dsi_start
datain = FM1ND_z_raw;
datain.th{1}(15,:)=pickP(:);
datain.th{1}(16,:)=datain.th{1}(15,:)+0.01;
load pickS.mat;
datain.th{1}(18,:)=pickS(:);
datain.th{1}(19,:)=datain.th{1}(18,:)-0.02;
% match direction
ntra=288;
dataout=datain;
dataout.th{1}(:,1)=datain.th{1}(:,ntra);
dataout.th{1}(12,1)=datain.th{1}(12,1);
dataout.dat{1}(:,1)=datain.dat{1}(:,ntra);
for i=1:ntra-1
    dataout.th{1}(:,i+1)=datain.th{1}(:,ntra-i);
    dataout.dat{1}(:,i+1)=datain.dat{1}(:,ntra-i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% notch5
[dataout] = new_bandpass(dataout,50,60,1000,2000);
[dataout] = cos_notch(dataout,15,18,50,60);         % 18-42
[dataout] = cos_notch(dataout,54,60,60,64);         % 60
[dataout] = cos_notch(dataout,177,181,181,185);     % 180
[dataout] = cos_notch(dataout,297,301,301,305);     % 300
[dataout] = cos_notch(dataout,357,361,361,365);     % 361
[dataout] = cos_notch(dataout,418,422,422,426);     % 422
[dataout] = cos_notch(dataout,537,541,542,546);     % 541
[dataout] = cos_notch(dataout,660,662,662,666);     % 662
[dataout] = cos_notch(dataout,717,721,721,725);     % 719
[dataout] = cos_notch(dataout,777,781,781,785);     % 781
[all] = cos_notch(dataout, 897,901,901,905);        % 900

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% extract noisy traces(207-240), filter differently
[noise]=subset(datain,48,81,0,1,1,1);
[dataout] = new_bandpass(noise,1,2,240,300);
[dataout] = new_bandpass(dataout,50,60,1000,2000);
[dataout,spec120] = cos_notch_test1(dataout,110,117,132,140);    % 120
```

```
[dataout,spec360] = cos_notch_test1(dataout,330,340,400,410);    % 370
[dataout] = new_bandpass(dataout,1,2,240,300);
all.dat{1}(:,48:81)=dataout.dat{1}(:,:);            % back together
[scal]=time_scale(all,1.7);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% P-mute2
[add_P]=add(scal,0.5); [align_P]=shft(add_P,-1,15);
align_P.th{1}(16,:)=0.5; align_P.th{1}(17,:)=0.525;
[fimu]=mute_Hann(align_P,1,17,17,0.05);
[un_align]=shft(fimu,1,15); [shiftup]=shft(un_align,-1,16);
[agc_mute]=agc(shiftup,0.08,1);

vaplot(agc_mute,0.1,0.6,1,288,1,1,4.5,'k',56,44,3,1,0,'S_input1.ps',...
'FFM001 ND z: before S-wave energy removal; scal 1.7, agc 0.08;');

vaplot(agc_mute,0.1,0.9,1,288,1,1,4.5,'k',56,20,8,1,2,'S_input1.eps',...
'FFM001 ND z: before S-wave energy removal; scal 1.7, agc 0.08;');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% S-mute
datain=shiftup;
[plus_time]=add(datain,0.6);
[shift]=shft(plus_time,-1,18);

[agc_mute]=agc(shift,0.08,1);
vaplot(agc_mute,0.4,0.7,1,288,1,1,4.5,'k',56,44,3,1,0,'S_shift1.ps',...
'FFM001 ND z: after shifting; scal 1.7, agc 0.08;');
vaplot(agc_mute,0.3,1.1,1,288,1,1,4.5,'k',56,30,8,1,2,'S_shift1.eps',...
'FFM001 ND z: after shifting; scal 1.7, agc 0.08;');

shift.th{1}(21,:)=0.59;
shift.th{1}(23,:)=0.62;
[filt]=medi_filt(shift,5);

[agc_mute]=agc(filt,0.08,1);
vaplot(agc_mute,0.4,0.7,1,288,1,1,4.5,'k',56,44,3,1,0,'S_filt1.ps',...
'FFM001 ND z: after median filtering; scal 1.7, agc 0.08;');

vaplot(agc_mute,0.3,1.1,1,288,1,1,4.5,'k',56,30,8,1,2,'S_filt1.eps',...
'FFM001 ND z: after median filtering; scal 1.7, agc 0.08;');
```

```
[mutse]=mute_Hann(filt,3,21,23,0.01);

[agc_mute]=agc(mutse,0.08,1);
vaplot(agc_mute,0.55,0.65,1,288,1,1,4.5,'k',56,44,3,1,0,'S_mutse1.ps',...
'FFM001 ND z: after muting; scal 1.7, agc 0.08;');
vaplot(agc_mute,0.3,1.1,1,288,1,1,4.5,'k',56,30,8,1,2,'S_mutse1.eps',...
'FFM001 ND z: after muting; scal 1.7, agc 0.08;');

[sub]=subr(mutse,shift);
sub.th{1}(21,:)=   0.6;

[agc_mute]=agc(sub,0.08,1);
vaplot(agc_mute,0.4,0.7,1,288,1,1,5.5,'k',56,44,3,1,0,'S_sub1.ps',...
'FFM001 ND z: after subtracting; scal 1.7, agc 0.08;');
vaplot(agc_mute,0.3,1.1,1,288,1,1,5.5,'k',56,30,8,1,2,'S_sub1.eps',...
'FFM001 ND z: after subtracting; scal 1.7, agc 0.08;');

[un_align]=shft(sub,1,18);
[shiftup]=shft(un_align,-1,21);

[agc_mute]=agc(shiftup,0.08,1);
vaplot(agc_mute,0.1,0.6,1,288,1,1,5.5,'k',56,44,3,1,0,'S_shiftup1.ps',...
'FFM001 ND z: after S-wave Energy removal; scal 1.7, agc 0.08;');
vaplot(agc_mute,0.1,0.9,1,288,1,1,5.5,'k',56,30,8,1,2,'S_shiftup1.eps',...
'FFM001 ND z: after S-wave Energy removal; scal 1.7, agc 0.08;');
dsi_end
```

F.13 displays the geometries (plot_geom1.m)

```
load FM1ND_z_cor.mat;
datain=FM1ND_z_cor;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%        all coordinates ARE IN MINE UTM!!!!!!
%
% calculate traveltimes
% straight BH

load source_loc.txt;
sBH=mean(source_loc);

BH=[6072927 317321  2525];        % BH at surface 317321   6072927  2525

% build the velocity model
zp=[0 2000];
vp=[6000 6500 ];

zrec=-1765:5:-330;                 % receiver - depth vector
zrec=zrec+2525;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

source=[FM1ND_z_cor.th{1}(29,:);
FM1ND_z_cor.th{1}(31,:);
FM1ND_z_cor.th{1}(33,:)]';
recev=[FM1ND_z_cor.th{1}(35,:);
       FM1ND_z_cor.th{1}(37,:);
       FM1ND_z_cor.th{1}(39,:)]';
level=1:1:length(recev);
for kk=1:length(recev)
    r=recev(kk,:);
    s=source(kk,:);
    x=[0 0 700];
    [refp(kk,:),reref(kk),treref(kk)]=txplane2(s,r,x,0,0,vp(1));
end
for kk=1:length(zrec);
    rBH=[BH(1) BH(2) zrec(kk)];
    x=[0 0 700];
    [refBH(kk,:),rerefBH(kk),trerefBH(kk)]=txplane2(sBH,rBH,x,0,0,vp(1));
end
```

```
ray1=[source(1,:); refp(1,:); recev(1,:)];
ray2=[source(288,:); refp(288,:); recev(288,:)];
ray3=[sBH; refBH(1,:); BH(1) BH(2) zrec(1)];
ray4=[sBH; refBH(288,:); BH(1)BH(2) zrec(288)];

figure;
hold on, plot3(ray1(:,1),ray1(:,2),ray1(:,3),'c')
hold on, plot3(ray2(:,1),ray2(:,2),ray2(:,3),'c')

hold on, plot3(recev(:,1),recev(:,2),recev(:,3),'g+')
hold on, plot3(refp(:,1),refp(:,2),refp(:,3),'k.')
hold on, plot3(source(:,1),source(:,2),source(:,3),'b+')

hold on, plot3(ray3(:,1),ray3(:,2),ray3(:,3),'m')
hold on, plot3(ray4(:,1),ray4(:,2),ray4(:,3),'m')

plot3(BH(1),BH(2),zrec,'bv'), hold on
plot3(sBH(1),sBH(2),sBH(3),'r*'), hold on,
plot3(refBH(:,1),refBH(:,2),refBH(:,3),'k.'), hold on

title('[Source location, Borehole location & reflection points]')
grid on;
xlabel('North');
ylabel('East');
zlabel('Elevation');
```

F.14 calculate reflection point, traveltime and travelled distance (VSP_DSI2a.m)

```
% for the straight borehole
% for the real, crooked borehole => DSI_VSP1b.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% build the velocity model
zp=[0 2000];
vp=[6000 6500];

zsrc=0;
zd=2000;
load FM1ND_z_cor.mat;
load source_loc.txt;
source=mean(source_loc);
BH=[6072927 317321  2525];
zrec=-1765:5:-330;          % receiver - depth vector
zrec=zrec+2525;
level=1:1:length(zrec);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% straight BH

% loop over receiver depth
for kk=1:length(zrec)
    r=[6072927  317321  zrec(kk)];
    s=source;
    x=[0 0 700];
    [sire(kk),stre(kk)]=txdirect1(s,r,vp(1));
    [sefp(kk,:),seref(kk),streref(kk)]=txplane2(s,r,x,0,0,vp(1));
end

save sire.mat sire;
save stre.mat stre;
save seref.mat seref;
save streref.mat streref;
save sefp.mat sefp;
```

F.15 calculates the maximum traces within the threshold (find_dist1.m)

```
load reref.mat;
load seref.mat;
diff=abs(seref-reref);
figure, plot(diff);
title('Difference of travelled distance between straight and real BH')
grid on;
xlabel('Trace #');
ylabel('distance [m]');
gra=gradient(diff);
for i=30:50
    for k=1:238
        test(i,k)=abs((diff(k)+gra(k)*i)-diff(k+i));
    end
end
[l,m]=find(test >= 14.90 & test <=14.99 );
figure, plot(test(min(l),:),'k');
title('Difference between real and interpolated Travelpath')
grid on;
xlabel('Receiver #');
ylabel('Difference [m]');
```

F.16 calculates synthetic traces (sc_synTraces2.m)

```
load FM1ND_z_cor.mat;
load FFM01_Density.aza;
load FFM01_Velocity.aza;
datain=FM1ND_z_cor;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% prepare output
dataout.fh=datain.fh;
dataout.fh{1}=5;                          % total number of traces in file
dataout.fh{12}=datain.fh{12};             % number of records in file
dataout.fh{13}=5;                         % max record fold
dataout.th{1}=zeros(64,5);                % initialize trace headers
dataout.th{1}(:,:)=datain.th{1}(:,1:5); % copy headers of first set of traces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% make a ricker wavelet
tw=-1/cf:datain.fh{8}:1/cf; arg=(pi*cf*tw).^2;
w=(1-2.*arg).*exp(-arg);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% all values
dens=FFM01_Density(:,2);
vel=FFM01_Velocity(:,2);
imp=dens.*vel; ref=ones(length(vel),1); for i=2:length(vel)
ref(i)=(imp(i)-imp(i-1))/(imp(i)+imp(i-1));
end ref(1)=ref(2);
dataout.fh{8}=0.25;                       % new sample interval = 0.25m
dataout.fh{7}=length(vel);
dataout.fh{10}=length(vel)*dataout.fh{8};
dataout.dat{1}=zeros(length(vel),dataout.fh{1});
all_lin=dataout;
s=conv(ref,w);
s=s(floor(length(tw)/2):floor(length(tw)/2)+length(vel)-1);
for
   i=1:5
   all_lin.dat{1}(:,i)=s;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% linearly interpolated per m
k=1;
for i=1:4:length(vel)-3
   dens1(k,:)=mean(FFM01_Density(i:i+3,:));
   velo1(k,:)=mean(FFM01_Velocity(i:i+3,:));
   k=k+1;
```

```
end
imp1=dens1(:,2).*velo1(:,2);
ref1=ones(length(velo1),1);
for i=2:length(velo1)
   ref1(i)=(imp1(i)-imp1(i-1))/(imp1(i)+imp1(i-1));
end
ref1(1)=ref1(2);
dataout.fh{8}=1;                          % new sample interval + 1m
dataout.fh{7}=length(velo1);
dataout.fh{10}=length(velo1)*dataout.fh{8};
dataout.dat{1}=zeros(length(velo1),dataout.fh{1}); lin_1=dataout;
s=conv(ref1,w);
s=s(floor(length(tw)/2):floor(length(tw)/2)+length(velo1)-1);
for i=1:5
   lin_1.dat{1}(:,i)=s;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% linearly interpolated(1m) picked every 5m
k=1;
for i=1:5:length(velo1)
   dens5(k,:)=dens1(i:i,:);
   velo5(k,:)=velo1(i:i,:);
   k=k+1;
end
imp5=dens5(:,2).*velo5(:,2);
ref5=ones(length(velo5),1);
for i=2:length(velo5)
   ref5(i)=(imp5(i)-imp5(i-1))/(imp5(i)+imp5(i-1));
end
ref5(1)=ref5(2);
dataout.fh{8}=5;                          % new sample interval + 5m
dataout.fh{7}=length(velo5);
dataout.fh{10}=length(velo5)*dataout.fh{8};
dataout.dat{1}=zeros(length(velo5),dataout.fh{1});
lin_5=dataout;
s=conv(ref5,w);
s=s(floor(length(tw)/2):floor(length(tw)/2)+length(velo5)-1);
for i=1:5
   lin_5.dat{1}(:,i)=s;
end
```

Appendix G

Processing Scripts FFS039 near offset

G.1 convert from .seg2 to .mat (sc_segy2mat.m)

```
% this script is written to convert segy to mat using segy2mat.m
dsi_start

[data]=segy2mat('FS39NDYN.sgy','cross3.crs','1');

save FS39NDYN.mat data
dsi_end
```

G.2 write trace headers (sc_write_th.m)

```
% Script to rewrite the wrong receiver coordinates using the file
% RECEIVERS4.TXT where the new calculated coordinates are stored.
% Coordinates were interpolated from the survey-log
%
% gems_drillhole_utm_trace.txt
%

clear all
load Receivers4.txt;

% it is an [176 x 5] matrix where rec station # are in colomn 1,
% wireline depth in colomn 2, UTM east in 3, UTM north in 4,
% and elevation in colomn 5

load FS39NDYN.mat                   % data

A= Receivers4;
data.th{1}(36,:)=data.th{1}(35,:);  % stores old rec north in unused line 36
data.th{1}(35,1:176)=A(:,4);        % stores north coordiantes in z
data.th{1}(35,177:352)=A(:,4);      % stores north coordiantes in H1
data.th{1}(35,353:528)=A(:,4);      % stores north coordiantes in H2

data.th{1}(38,:)=data.th{1}(37,:);  % stores the old rec east in th 38
data.th{1}(37,1:176)=A(:,3);        % stores east coordiantes in z
data.th{1}(37,177:352)=A(:,3);      % stores east coordiantes in H1
data.th{1}(37,353:528)=A(:,3);      % stores east coordiantes in H2

data.th{1}(40,:)=data.th{1}(39,:);  % stores old rec elev in th 40
data.th{1}(39,1:176)=A(:,5);        % stores north coordiantes in z
data.th{1}(39,177:352)=A(:,5);      % stores north coordiantes in H1
data.th{1}(39,353:528)=A(:,5);      % stores north coordiantes in H2

data.th{1}(56,1:176)=A(:,2);        % stores wireline depth in z
data.th{1}(56,177:352)=A(:,2);      % stores wireline depth in H1
data.th{1}(56,353:528)=A(:,2);      % stores wireline depth H2

FS39ND_cor = data;
save FS39ND_cor.mat FS39ND_cor
```

G.3 more trace headers (sc_more_th3.m)

```
dsi_start
load FS39ND_cor.mat;
datain=FS39ND_cor;

[dataout]=sortrec_many(datain,2,13);
load Sources.txt
% it is an [4 x 4] matrix where shot # are in colomn 1, UTM north in
% colomn 2, UTM east in 3, UTM elevation in colomn 4

A= Sources;

a=subset(dataout,1,32,0,0.99975,1,1);
b=subset(dataout,1,32,0,0.99975,2,2);
[all]=merge_traces5(a,b);
all1=all;
for i=3:22
    i=subset(dataout,1,32,0,0.99975,i,i);
    [all]=merge_traces5(all,i);
end

dataout=all;

dataout.th{1}(30,:)=dataout.th{1}(29,:);  % stores old source north in th 30
dataout.th{1}(29,1:352)=A(1,1);           % stores north coordiantes
dataout.th{1}(29,353:512)=A(2,1);
dataout.th{1}(29,513:704)=A(3,1);

dataout.th{1}(32,:)=dataout.th{1}(31,:);  % stores old source east in th 32
dataout.th{1}(31,1:352)=A(1,2);
dataout.th{1}(31,353:512)=A(2,2);         % stores east coordiantes
dataout.th{1}(31,513:704)=A(3,2);

dataout.th{1}(34,:)=dataout.th{1}(33,:);  % stores old source elev in th 34
dataout.th{1}(33,1:352)=A(1,3);
dataout.th{1}(33,353:512)=A(2,3);
dataout.th{1}(33,513:704)=A(3,3);         % stores the source elev coordiantes

[out]=sortrec_many(dataout,11,1);

FS39ND_cor2 = out;
save FS39ND_cor2.mat FS39ND_cor2
```

G.4 plot geometry of FFS039 (plot_BH_FS39_cor5.m)

```
clear all
load Drill1.txt;
load Receivers4.txt;
B=Receivers4;
load FS39_z_cor2.mat;
A=Drill1;

% plot surveyed coordinates from gems_drillhole_umt_trace.txt
East=A(:,4);
North=A(:,5);
Elev=A(:,6);

% plot receiver coordinates from trace headers
figure,
axis manual
axis equal
axis vis3d

plot3(North,East,Elev,'b^'), hold on,

E=FS39_z_cor2.th{1}(37,:);
N=FS39_z_cor2.th{1}(35,:);
Elv=FS39_z_cor2.th{1}(39,:);

% plot receivers from txt-file
Ea=B(:,3);
No=B(:,4);
Ev=B(:,5);
plot3(No,Ea,Ev,'g.'),

plot3(N,E,Elv,'r+'), hold on,
plot3(North,East,Elev,'b.'), hold on,
grid on,

K=ones(2,6);
K(1,:)=A(1,:);
K(2,:)=A(23,:);
X=K(:,4);
Y=K(:,5);
Z=K(:,6);
plot3(Y,X,Z,'k'), hold on,
```

```
% title('BH FFS039');
xlabel('Northing');
ylabel('Easting');
zlabel('Elevation');
hold off
set(gca,'CameraViewAngleMode','Manual')
```

G.5 remove H1 and H2 (sc_subset3.m)

```
dsi_start
load FS39ND_cor2.mat;
[z_comp]=subset(FS39ND_cor2,1,176,0,0.99975,2,2);
FS39_z_cor2=z_comp;
save FS39_z_cor2.mat FS39_z_cor2
dsi_end
```

G.6 processing applied to the data (sc_all26.m)

```
clear all
%%%%%%%%%%%%%%%%%
load FS39_z_cor2.mat;
load pickP.mat;
load pickS1.mat;

dsi_start
datain = FS39_z_cor2;
datain.th{1}(15,:)=pickP(:);
datain.th{1}(16,:) = datain.th{1}(15,:)+0.02;
datain.th{1}(17,:)=vbad1;
datain.th{1}(18,:)=vbad2;

[scal]=time_scale(datain,1.7);
bad_1=[63,120,122,124,126,128];
bad=[16,31,63,64,77,78,93,94,109,110,114,116,118,120,122,124,126,128,...
     143,144,157,160,175,176];

% [repl]=replace_bad(kill_bad);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% datain.th{1}(15,:)=pickP(:); already assigned
% datain.th{1}(16,:) = datain.th{1}(15,:)+0.002;
% 17 / 18 need to be overwritten
% [datain]=time_scale(datain,1.7); already done

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1st notch
F=[58,60,61,68;
118,120,120,124;
177,180,181,186;
298,301,301,304;
378,380,380,384;
417,420,420,426;
538,541,542,545;
657,660,660,664;
779,780,780,785;];

[new_no] = cos_notch_newtest(scal,F);
[mut]=mute_Hann(new_no,1,16,16,0.005);
newNo1=mut;
[N1ener]=agc(newNo1,0.2,1);
```

```
% vaplot(N1ener,0.0,0.8,1,176,1,1,1,'k',13,20,8,1,0,'FS39_all23_N1.ps',...
% 'FFS039 ND z: new_no; agc 0.2');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% P-mute
[plus_time]=add(new_no,0.6);
[shift]=shft(plus_time,-1,15);
shift.th{1}(21,:)=0.595;
[filt]=medi_filt(shift,15);
[mutse]=mute_Hann(filt,1,21,21,0.005);
[sub]=subr(mutse,shift);
sub.th{1}(23,:)= 0.603;
[mute]=mute_Hann(sub,1,23,23,0.005);
[un_align]=shft(mute,1,15);
[shiftup]=shft(un_align,-1,21);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% S-mute
shiftup.th{1}(19,:)=pickS1(:);
[mut]=mute_Hann(shiftup,1,15,15,0.00025);
[plus_time]=add(mut,0.6);
[shift]=shft(plus_time,-1,19);
[filt]=medi_filt(shift,15);
[sub]=subr(filt,shift);
sub.th{1}(21,:)= 0.6;
[un_align]=shft(sub,1,19);
[shiftup]=shft(un_align,-1,21);
outS=shiftup;

% H-mute
[filt]=medi_filt(shiftup,15);
[sub]=subr(filt,shiftup);
outH=sub;

[Hener]=agc(outH,0.2,1);
% vaplot(Hener,0.0,0.8,1,176,1,1,4,'k',13,20,8,1,0,'FS39_all23_H.ps',...
% 'FFS039 ND z: rem P,S,H; agc 0.2');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% make a nice plot
[bp] = bandpass(outH,50,71,250,400);
[mut]=mute_Hann(bp,1,16,16,0.01);
```

```
FS39_all26_fi=mut;
save FS39_all26_fi.mat FS39_all26_fi

% add first breaks peaks
headw1=15;
COUNT=1;        %loop over records
    for i=[1 140 141 176]
         peak=round((mut.th{COUNT}(headw1,i)-mut.fh{9})/mut.fh{8});
         mut.dat{COUNT}(peak,i)=50000;
     end
[BPener]=agc(mut,0.08,1); [kill_bad]=kill(BPener,1,bad_1);
[killed]=pack_good(kill_bad,1);
 vaplot(killed,0.0,0.8,1,176,1,1,1,'k',13,20,8,1,0,'FS39_all26.ps',...
 'FFS039 ND z: rem P,S,H; BP(50,71,250,400); agc 0.08');
 vaplot(killed,0.0,0.8,1,176,1,1,5,'k',13,20,8,1,0,'FS39_all26a.ps',...
 'FFS039 ND z: rem P,S,H; BP(50,71,250,400); agc 0.08');
 vaplot(killed,0.0,0.8,1,176,1,1,8,'k',13,20,8,1,0,'FS39_all26b.ps',...
 'FFS039 ND z: rem P,S,H; BP(50,71,250,400); agc 0.08');

FS39_all26_nofi=outH;
save FS39_all26_nofi.mat FS39_all26_nofi
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dsi_end
```

G.7 create a corridor stack of the whole dataset (sc_twt_new.m)

```
clear all;
load FS39_all26_fi.mat;
datain=FS39_all26_fi;

dsi_start
[TWT]=shft(datain,1,15);
mute=datain;

% set window corners
poly=[1 0.23175*2; 1 0.23175*2+0.9; 176 0.08825*2+0.9; 176 0.08825*2];
[wimu,scal]=window_mute4(TWT,poly,0.005,1);

mute.dat{1} = wimu;

[corr_t1,normal]=norm_stack_t(mute,scal,5);

[scal_dat]=scale_rec(mute);
[scal_corr]=scale_rec(corr_t1);
[dataout]=merge_traces2(scal_corr,scal_dat);

FS39_corr_Pfilt = dataout;
save FS39_corr_Pfilt.mat
FS39_corr_Pfilt;

[agc_mute]=agc(dataout,0.08,1); [agc_corr]=agc(corr_t1,0.08,1);
vaplot(agc_corr,0.0,0.9,1,5,1,1,1,'k',1,5,5.5,1,0,'ncorri_Pfilt.ps',...
'FFS039 ND z; TWT-shifted;large mute window with taper');
vaplot(agc_mute,0.2,1.1,1,182,1,1,5,'k',56,20,8,1,0,'FS39_ncorr_Pfilt.ps',...
'FFS039 ND z; TWT-shifted; large mute window with taper');
 dsi_end
```

G.8 create corridor stack (sc_corr_28... .m)

```
% this is the example for near 1

load FS39_all26_fi.mat;
datain=FS39_all26_fi;
a=0.05;

% dist from first breaks for near1, 0.1 for near2, 0.15 for near3
% 0.2 for mid1, 0.25 for mid2, 0.3 for mid3,
% 0.35 for far1, 0.4 for far2, 0.45 for far3

win=abs((2*datain.th{1}(15,176)-2*datain.th{1}(15,1))/175*27);
dsi_start

[TWT]=shft(datain,1,15);
mute=TWT;
% set window corners
poly=[[1 0.23175*2+a; 1 0.23175*2+a+win; 176 0.08825*2+a+win; 176 0.08825*2+a]];

[wimu,scal]=window_mute4(TWT,poly,0.01,1);

mute.dat{1} = wimu;

[corr_t1]=norm_stack(mute,scal,5);
[agc_corr]=agc(corr_t1,0.08,1);
cor_new_near1 = corr_t1;
save cor_new_near1.mat cor_new_near1;

[scal_dat]=scale_rec(mute);
[scal_corr]=scale_rec(corr_t1);
[dataout]=merge_traces2(scal_corr,scal_dat);

merge_near = dataout;

[agc_mute]=agc(dataout,0.08,1);
[agc_corr]=agc(corr_t1,0.08,1);

vaplot(agc_mute,0.2,1.2,1,182,1,1,4,'k',56,26,8,1,0,'merge_28near1.ps',...
'FFS039 ND z; "near borehole1 28 traces"');

dsi_end
```

G.9 merges all corridor stacks into one file (sc_new_comp_corr.m)

```
clear all;

load cor_new_near1.mat;
load cor_new_near2.mat;
load cor_new_near3.mat;
load cor_new_mid1.mat;
load cor_new_mid2.mat;
load cor_new_mid3.mat;
load cor_new_far1.mat;
load cor_new_far2.mat;
load cor_new_far3.mat;

dsi_start
[scal_1a]=scale_rec(cor_new_near1);
[scal_1b]=scale_rec(cor_new_near2);
[scal_1c]=scale_rec(cor_new_near3);
[scal_2a]=scale_rec(cor_new_mid1);
[scal_2b]=scale_rec(cor_new_mid2);
[scal_2c]=scale_rec(cor_new_mid3);
[scal_3a]=scale_rec(cor_new_far1);
[scal_3b]=scale_rec(cor_new_far2);
[scal_3c]=scale_rec(cor_new_far3);

[merge1]=merge_traces3(scal_1a,scal_1b);
[merge1]=merge_traces3(merge1,scal_1c);
[merge1]=merge_traces3(merge1,scal_2a);
[merge1]=merge_traces3(merge1,scal_2b);
[merge1]=merge_traces3(merge1,scal_2c);
[merge1]=merge_traces3(merge1,scal_3a);
[merge1]=merge_traces3(merge1,scal_3b);
[merge1]=merge_traces3(merge1,scal_3c);

New_comp_corr= merge1;
save New_comp_corr.mat New_comp_corr;

vaplot(merge1,0.2,1.2,1,45,1,0,2,'k',1,10,8,1,0,'New_comp_corr.ps',...
'FFS039 ND z; all stacks (0.1s)');
dsi_end
```

G.10 creates plot (sc_plot_all.m)

```
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load FS39_z_cor2.mat;

load pickP.mat;
load pickS1.mat;

dsi_start datain = FS39_z_cor2;
datain.th{1}(15,:)=pickP(:);
datain.th{1}(16,:) = datain.th{1}(15,:)+0.02;
datain.th{1}(17,:)=vbad1;
datain.th{1}(18,:)=vbad2;

[scal]=time_scale(datain,1.7); [ascal]=agc(scal,0.1,1);
vaplot(ascal,0.0,0.8,1,176,1,1,10,'k',56,44,3,1,0,'all26_scal.ps',...
'FFS039 ND z: scaled; agc 0.1');
vaplot(ascal,0.0,0.8,1,176,1,1,10,'k',56,20,8,1,2,'all26_scal.eps',...
'FFS039 ND z: scaled; agc 0.1');

bad_1=[63,120,122,124,126,128];
bad=[16,31,63,64,77,78,93,94,109,110,114,116,118,120,122,124,126,128,...
     143,144,157,160,175,176];


% [repl]=replace_bad(kill_bad);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% datain.th{1}(15,:)=pickP(:); already assigned
% datain.th{1}(16,:) = datain.th{1}(15,:)+0.002;
% 17 / 18 need to be overwritten
% [datain]=time_scale(datain,1.7); already done

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1st notch
F=[58,60,61,68;
   118,120,120,124;
   177,180,181,186;
   298,301,301,304;
   378,380,380,384;
```

```
   417,420,420,426;
   538,541,542,545;
   657,660,660,664;
   779,780,780,785;];

[new_no] = cos_notch_newtest(scal,F); [anew_no]=agc(new_no,0.1,1);
vaplot(anew_no,0.0,0.8,1,176,1,1,10,'k',56,44,3,1,0,'all26_fi.ps',...
'FFS039 ND z: notch; agc 0.1');
vaplot(anew_no,0.0,0.8,1,176,1,1,10,'k',56,20,8,1,2,'all26_fi.eps',...
'FFS039 ND z: notch; agc 0.1');
[mut]=mute_Hann(new_no,1,16,16,0.005); [amut]=agc(mut,0.1,1);
vaplot(amut,0.0,0.8,1,176,1,1,10,'k',56,44,3,1,0,'all26_fimu.ps',...
'FFS039 ND z: notch mute; agc 0.1');
vaplot(amut,0.0,0.8,1,176,1,1,10,'k',56,20,8,1,2,'all26_fimu.eps',...
'FFS039 ND z: notch mute; agc 0.1');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% P-mute
[plus_time]=add(new_no,0.6);

[aplus]=agc(plus_time,0.1,1);
vaplot(aplus,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Pplus.ps',...
'FFS039 ND z: add time; agc 0.1');
vaplot(aplus,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,2,'all26_Pplus.eps',...
'FFS039 ND z: add time; agc 0.1');

[shift]=shft(plus_time,-1,15);
shift.th{1}(21,:)=0.595;

[ashift]=agc(shift,0.1,1);
vaplot(ashift,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Pshift.ps',...
'FFS039 ND z: shift; agc 0.1');
vaplot(ashift,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,0,'all26_Pshift.eps',...
'FFS039 ND z: shift; agc 0.1');

[filt]=medi_filt(shift,15);
[mutse]=mute_Hann(filt,1,21,21,0.005);

[afilt]=agc(mutse,0.1,1);
vaplot(afilt,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Pfilt.ps',...
'FFS039 ND z: median filtered; agc 0.1');
vaplot(afilt,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,2,'all26_Pfilt.eps',...
```

```
'FFS039 ND z: median filtered; agc 0.1');

[sub]=subr(mutse,shift);
sub.th{1}(23,:)= 0.603;
[mute]=mute_Hann(sub,1,23,23,0.005);

[asub]=agc(sub,0.1,1);
vaplot(asub,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Psub.ps',...
'FFS039 ND z: subtracted; agc 0.1');
vaplot(asub,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,2,'all26_Psub.eps',...
'FFS039 ND z: subtracted; agc 0.1');

[un_align]=shft(mute,1,15);
[shiftup]=shft(un_align,-1,21);
[nmut]=mute_Hann(shiftup,1,16,16,0.005);

[ashiftup]=agc(nmut,0.1,1);
vaplot(ashiftup,0.0,0.8,1,176,1,1,10,'k',56,44,3,1,0,'all26_Pshiftup.ps',...
'FFS039 ND z: shifted back; agc 0.1');
vaplot(ashiftup,0.0,0.8,1,176,1,1,10,'k',56,20,8,1,2,'all26_Pshiftup.eps',...
'FFS039 ND z: shifted back; agc 0.1');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% S-mute
shiftup.th{1}(19,:)=pickS1(:);
[mut]=mute_Hann(shiftup,1,15,15,0.00025);
[Splus_time]=add(mut,0.6);

[aSplus_time]=agc(Splus_time,0.1,1);
vaplot(aSplus_time,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Sadd.ps',...
'FFS039 ND z: add time; agc 0.1');
vaplot(aSplus_time,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,2,'all26_Sadd.eps',...
'FFS039 ND z: add time; agc 0.1');

[Sshift]=shft(Splus_time,-1,19);

[aSshift]=agc(Sshift,0.1,1);
vaplot(aSshift,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Sshift.ps',...
'FFS039 ND z: shift; agc 0.1');
vaplot(aSshift,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,2,'all26_Sshift.eps',...
'FFS039 ND z: shift; agc 0.1');
```

```
[Sfilt]=medi_filt(Sshift,15);

[aSfilt]=agc(Sfilt,0.1,1);
vaplot(aSfilt,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Sfilt.ps',...
'FFS039 ND z: median filtered; agc 0.1');
vaplot(aSfilt,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,2,'all26_Sfilt.eps',...
'FFS039 ND z: median filtered; agc 0.1');

[Ssub]=subr(Sfilt,Sshift);
Ssub.th{1}(21,:)= 0.6;

[aSsub]=agc(Ssub,0.1,1);
vaplot(aSsub,0.3,1.1,1,176,1,1,10,'k',56,44,3,1,0,'all26_Ssub.ps',...
'FFS039 ND z: subtracted; agc 0.1');
vaplot(aSsub,0.3,1.1,1,176,1,1,10,'k',56,20,8,1,2,'all26_Ssub.eps',...
'FFS039 ND z: subtracted; agc 0.1');

[Sun_align]=shft(Ssub,1,19);
[Sshiftup]=shft(Sun_align,-1,21);
outS=Sshiftup;
[Smut]=mute_Hann(outS,1,16,16,0.005);

[aoutS]=agc(Smut,0.1,1);
vaplot(aoutS,0.0,0.8,1,176,1,1,10,'k',56,44,3,1,0,'all26_Sout.ps',...
'FFS039 ND z: shifted back; agc 0.1');
vaplot(aoutS,0.0,0.8,1,176,1,1,10,'k',56,20,8,1,2,'all26_Sout.eps',...
'FFS039 ND z: shifted back; agc 0.1');

% H-mute
[filt]=medi_filt(Sshiftup,15);
[sub]=subr(filt,Sshiftup);
outH=sub;
Hmut]=mute_Hann(outH,1,16,16,0.005);

[aoutH]=agc(Hmut,0.1,1);
vaplot(aoutH,0.0,0.8,1,176,1,1,10,'k',56,44,3,1,0,'all26_Hout.ps',...
'FFS039 ND z: H removed; agc 0.1');
vaplot(aoutH,0.0,0.8,1,176,1,1,10,'k',56,20,8,1,2,'all26_Hout.eps',...
'FFS039 ND z: H removed; agc 0.1');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% no more 2nd notch
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% make a nice plot
[bp] = bandpass(outH,50,71,250,400);
[mut]=mute_Hann(bp,1,16,16,0.01);

% add first breaks peaks
headw1=15;
COUNT=1;        %loop over records
    for i=[1 140 141 176]
         peak=round((mut.th{COUNT}(headw1,i)-mut.fh{9})/mut.fh{8});
         mut.dat{COUNT}(peak,i)=50000;
    end
[BPener]=agc(mut,0.08,1);
[kill_bad]=kill(BPener,1,bad_1);
[killed]=pack_good(kill_bad,1);
dsi_end
```

G.11 bandpass filter test (sc_bp_Test2.m)

```
clear all

load FS39_all26_nofi.mat;
datain = FS39_all26_nofi;
dsi_start
[bp0]=bandpass(datain,8,12,30,45);
[bp1]=bandpass(datain,8,12,50,71);
[bp2]=bandpass(datain,21,30,50,71);
[bp3]=bandpass(datain,21,30,70,100);
[bp4]=bandpass(datain,50,71,90,130);
[bp5]=bandpass(datain,70,100,150,220);
[bp6]=bandpass(datain,90,130,250,355);
[bp7]=bandpass(datain,150,220,350,495);
[bp8]=bandpass(datain,250,355,500,710);
[bp9]=bandpass(datain,350,495,650,920);
[bp10]=bandpass(datain,500,725,900,1272);
[bp11]=bandpass(datain,650,920,1100,1560);
[bp12]=bandpass(datain,900,1272,1500,2125);

[bp_test]=merge_files(bp0,bp1);
[bp_test]=merge_files(bp_test,bp2);
[bp_test]=merge_files(bp_test,bp3);
[bp_test]=merge_files(bp_test,bp4);
[bp_test]=merge_files(bp_test,bp5);
[bp_test]=merge_files(bp_test,bp6);
[bp_test]=merge_files(bp_test,bp7);
[bp_test]=merge_files(bp_test,bp8);
[bp_test]=merge_files(bp_test,bp9);
[bp_test]=merge_files(bp_test,bp10);
[bp_test]=merge_files(bp_test,bp11);
[bp_test]=merge_files(bp_test,bp12);

[scal]=time_scale(bp_test,1.7);
[mut]=mute_taper(scal,1,16,16,0.005);
[agc_mute]=agc(mut,0.08,1);
[a_mute]=agc(datain,0.08,1);
vaplot(a_mute,0.0,0.8,1,176,1,1,10,'k',1,44,3,1,0,'FS39ND_nbefore.ps',...
'FFS039 ND z  agc 0.08');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,1,0,'FS39ND_nb0.ps',...
'FFS039 ND z agc 0.08;  bp 8,12,30,45');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,3,0,'FS39ND_nb2.ps',...
```

```
'FFS039 ND z agc 0.08;  bp 21,30,50,71');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,5,0,'FS39ND_nb4.ps',...
'FFS039 ND z agc 0.08;  bp 50,71,90,130');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,6,0,'FS39ND_nb5.ps',...
'FFS039 ND z agc 0.08;  bp 70,100,150,220');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,7,0,'FS39ND_nb6.ps',...
'FFS039 ND z agc 0.08;  bp 90,130,250,355');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,8,0,'FS39ND_nb7.ps',...
'FFS039 ND z agc 0.08;  bp 150,220,350,495');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,9,0,'FS39ND_nb8.ps',...
'FFS039 ND z agc 0.08;  bp 250,355,500,710');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,11,0,'FS39ND_nb10.ps',...
'FFS039 ND z agc 0.08;  bp 500,725,900,1272');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,44,3,13,0,'FS39ND_nb12.ps',...
'FFS039 ND z agc 0.08;  bp 900,1272,1500,2125');

vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,1,2,'FS39ND_nb0.eps',...
'FFS039 ND z  agc 0.08;  bp 8,12,30,45');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,3,2,'FS39ND_nb2.eps',...
'FFS039 ND z  agc 0.08;  bp 21,30,50,71');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,5,2,'FS39ND_nb4.eps',...
'FFS039 ND z  agc 0.08;  bp 50,71,90,130');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,6,2,'FS39ND_nb5.eps',...
'FFS039 ND z  agc 0.08;  bp 70,100,150,220');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,7,2,'FS39ND_nb6.eps',...
'FFS039 ND z  agc 0.08;  bp 90,130,250,355');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,8,2,'FS39ND_nb7.eps',...
'FFS039 ND z  agc 0.08;  bp 150,220,350,495');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,9,2,'FS39ND_nb8.eps',...
'FFS039 ND z  agc 0.08;  bp 250,355,500,710');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,11,2,'FS39ND_nb10.eps',...
'FFS039 ND z  agc 0.08;  bp 500,725,900,1272');
vaplot(agc_mute,0.0,0.8,1,176,1,1,4,'k',1,20,8,13,2,'FS39ND_nb12.eps',...
'FFS039 ND z  agc 0.08;  bp 900,1272,1500,2125');

FS39ND_bp2 = bp_test;
save FS39ND_bp2.mat FS39ND_bp2

dsi_end
```

G.12 calculate reflection point, traveltime and travelled distance (VSP_DSI2a.m)

```
% build the velocity model
zp=[0 2000];
vp=[6000 6500];
% make s-wave model
zsrc=0;
zd=1400;
load Sources.txt;
source=mean(Sources);
BH=[6073335 314155  2547.71];
zrec=-1105:5:-230;                          % receiver - depth vector
zrec=zrec+2547.71;

level=1:1:length(zrec);
%%%%%%%%%%%%%%%%%%%%%%%%%%
% straight BH

% loop over receiver depth
for kk=1:length(zrec)
    r=[6073335  314155  zrec(kk)];
    s=source;
    x=[0 0 700];
  [sire(kk),stre(kk)]=txdirect1(s,r,vp(1));
  [sefp(kk,:),seref(kk),streref(kk)]=txplane2(s,r,x,0,0,vp(1));
end

save sire.mat sire;
save stre.mat stre;
save seref.mat seref;
save streref.mat streref;
save sefp.mat sefp;
```

G.13 calculate reflection point, traveltime and travelled distance (VSP_DSI1b.m)

```
% build the velocity model
zp=[0 2000];
vp=[6000 6500];
% make s-wave model
zsrc=0;
% zrec=350:50:550;
zd=2000;
refp=zeros(176,3);
load FS39_z_cor2.mat;
source=[FS39_z_cor2.th{1}(29,:);
        FS39_z_cor2.th{1}(31,:);
        FS39_z_cor2.th{1}(33,:)]';
recev=[FS39_z_cor2.th{1}(35,:);
       FS39_z_cor2.th{1}(37,:);
       FS39_z_cor2.th{1}(39,:)]';

level=1:1:length(recev);
%%%%%%%%%%%%%%%%%%%%%%%%%%
% crooked BH

% loop over receiver depth
for kk=1:length(recev)
    r=recev(kk,:);
    s=source(kk,:);
    x=[0 0 700];
    [dire(kk),tre(kk)]=txdirect1(s,r,vp(1));
    [refp(kk,:),reref(kk),treref(kk)]=txplane2(s,r,x,0,0,vp(1));
end

save dire.mat dire;
save tre.mat tre;
save refp.mat refp;
save reref.mat reref;
save treref.mat treref;
```

G.14 calculates the maximum traces within the threshold (find_dist1.m)

```
load reref.mat;
load seref.mat;
diff=abs(seref-reref);
figure, plot(diff);
title('Difference of reflected raypath distance
between straight and real BH')
grid on;
xlabel('Trace #');
ylabel('distance [m]');
gra=gradient(diff);
for i=20:50
    for k=1:126
        test(i,k)=abs((diff(k)+gra(k)*i)-diff(k+i));
    end
end
[l,m]=find(test >= 14 & test <=14.99 );
figure, plot(test(min(l),:),'k');
title('Difference between real and interpolated Travelpath') grid on;
xlabel('Receiver #');
ylabel('Difference [m]');
```

G.15 plot geometry of the borehole and raypaths (plot_geom1.m)

```
% calculates reflection points,travelled distance and traveltimes,
% for reflected rays straight BH and real BH using the module txplane2
% plots everything
load FS39_z_cor2.mat;
datain=FS39_z_cor2;

%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%        all coordinates ARE IN MINE UTM!!!!!!
%
load Sources.txt;
sBH=mean(Sources);

BH=[6073335 314155  2547.71];

% build the velocity model
zp=[0 2000];
vp=[6000 6500 ];

zrec=-1105:5:-230;                      % receiver - depth vector
zrec=zrec+2547.71;

%%%%%%%%%%%%%%%%%%%%%%%%
% real coordinates
source=[FS39_z_cor2.th{1}(29,:);
        FS39_z_cor2.th{1}(31,:);
        FS39_z_cor2.th{1}(33,:)]';
recev=[FS39_z_cor2.th{1}(35,:);
       FS39_z_cor2.th{1}(37,:);
       FS39_z_cor2.th{1}(39,:)]';

level=1:1:length(recev);

for kk=1:length(recev)
    r=recev(kk,:);
    s=source(kk,:);
    x=[0 0 700];
    [refp(kk,:),reref(kk),treref(kk)]=txplane2(s,r,x,0,0,vp(1));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%
% simple coordinates
for kk=1:length(zrec);
    rBH=[BH(1) BH(2) zrec(kk)];
    x=[0 0 700];
    [refBH(kk,:),rerefBH(kk),trerefBH(kk)]=txplane2(sBH,rBH,x,0,0,vp(1));
end

ray1=[source(1,:); refp(1,:); recev(1,:)];
ray2=[source(176,:); refp(176,:); recev(176,:)];
ray3=[sBH; refBH(1,:); BH(1) BH(2) zrec(1)];
ray4=[sBH; refBH(176,:); BH(1) BH(2) zrec(176)];


figure;
hold on, plot3(ray1(:,1),ray1(:,2),ray1(:,3),'c')
hold on, plot3(ray2(:,1),ray2(:,2),ray2(:,3),'c')

hold on, plot3(recev(:,1),recev(:,2),recev(:,3),'g+')
hold on, plot3(refp(:,1),refp(:,2),refp(:,3),'k.')
hold on, plot3(source(:,1),source(:,2),source(:,3),'b+')

hold on,plot3(ray3(:,1),ray3(:,2),ray3(:,3),'m')
hold on, plot3(ray4(:,1),ray4(:,2),ray4(:,3),'m')

plot3(BH(1),BH(2),zrec,'bv'), hold on
plot3(sBH(1),sBH(2),sBH(3),'r*'), hold on hold on,
plot3(refBH(:,1),refBH(:,2),refBH(:,3),'k.'), hold on

% legend('source','top BH','real sources','receiver')
title('[Source location, Borehole location & reflection points]') grid on;
xlabel('North');
ylabel('East');
zlabel('Elevation');
```