LatihanVSPsu

VSP Processing Exercise using Seismic Unix

fahdi@gm2001.net

February 2012

DESCRIPTION

Set of script to model and process simple geometry/simple vertical seismic profiling (VSP) data. This document also explained basic understanding about VSP (acquisition, processing, and application/interpretation).

REQUIREMENT

- Basic understanding of Unix/Linux command and shell scripting would be advantage in learning this.

- Seismic Unix (http://www.cwp.mines.edu/cwpcodes/)

- Ela2D (Get copy at http://www.gm2001.net/latihanvspsu/)

SOURCE CODE REPOSITORY

I put repository on github

https://www.github.com.fahdi104/latihanvspsu

If you have git account, you can fork this by typing

$git clone git@github.com:fahdi104/latihanvspsu.git

I will also publish to

http://www.gm2001.net/latihanvspsu

Please join me to develop this script or update/edit this document.

DATASET FOR EXERCISE

The dataset used for this exercise and displayed in this documents was using Naylor-1 well, available for research from Earth Resources, Dept. of Primary Industries, State Government of Victoria Australia.

(http://er-info.dpi.vic.gov.au/petroleum/assets/pe91072.htm#pe910726) or contact them directly for information of the data.

Document Version

Version 0.1 February 2012

Version 0.2 April 2013

VSP Overview

Vertical seismic profiling (VSP) or in general known as borehole seismic, is a kind of seismic survey where put the seismic receiver inside a wellbore (as opposed to surface seismic where we put the receiver on surface), and generated the source from surface (or from wellbore, depend on the configuration). This kind of survey is generally part of wellbore characterization (well-logging), combined with other logging methodology to extract information from a wellbore such as radioactive logging (gamma ray, density, and neutron), sonic logging etc.

Main purpose of running VSP survey is to get time and depth relation. Using wireline cable we can measure the depth, and by shooting seismic wave from surface we can measure what is the transit time of the seismic wave to the receiver at certain depth. This time-depth relation will be used to relate log data in depth with seismic data in time, and further to build velocity model for depth to time or time to depth conversion of subsurface model.

VSP survey normally defined by the type of source configuration that used in the acquisitions, because the receiver is always fixed inside the wellbore. For example if we are using single source with fixed offset close to well (receiver), this survey known as zero offset VSP. If we have fixed offset that relatively far away from wellbore, this can be near offset VSP survey or far offset VSP survey. In a case of multi offset VSP, where we shot directly above VSP receiver (particularly for deviated well), this survey known as walkabove VSP or vertical incidence VSP. And if we shot in multi offset within particular direction away from the wellbore, this is known as multi offset walkaway VSP. If we shot quite dense in 3D, this is known as 3D VSP.

Choices of which VSP survey/data that we want to acquire will be depend on what is the objective that we want to achive. If we only want to get time-depth relation and 1-D seismic image for correlation we can utilized zero offset VSP. In case we want to get structural image, we can used near offset, far offset, or multi offset walkaway VSP. If we required to measured anisotropic response in the borehole we need to run multi offset Walkaway VSP. Normally for complex VSP survey (more than zero offset VSP), it is required to run pre-survey modeling either by simple ray tracing to check illumination coverage or by doing full waveform modeling to see feasibility of recorded data to achieve the required objectives.

The processed full VSP waveforms can output 1-D or 2-D (or even 3-D) seismic image in the borehole. Theoretically VSP image is higher in resolution due to shorter raypath compare to surface seismic. And VSP imaging is controlled in depth, therefore provide more accurate depth image information. Furthermore than corridor stacks, depending on survey configuration, VSP can be utilized further for look ahead-depth predictions, structural imaging, attenuation factor estimations, and anisotropy parameter estimations.

Operational wise, VSP can be run on open hole or cased hole wellbore. In the acquisitions, normally VSP is part of logging sequence that came last (because it's can be run in open hole and cased hole), and it can be run for the whole sections. There are two main components for VSP logging operation, downhole tools that are the receiver (geophone), and surface equipment which consist of seismic source and surface seismic sensors to QC the source. The data quality will be depend on how good the wellbore condition, and the coupling of receiver with the formations. In cased hole (dual casing or even triple casing) environment this will also depend on the cement integrity between casing. And of course the data quality will also depend whether the source is working properly to generate seismic source or not.

Borehole seismic downhole tools (receiver/geophone) are generally a multi-component geophone, record data in horizontal (X-Y component) and vertical component (Z). The tools can be conveyed using wireline cable, tractor, or nowadays inside a drilling tools component for seismic while drilling operations. The source itself can be an airgun that generates pressure, in land operation this gun was put inside a gun pit filled with water, and can be attached in boat for offshore operations. Vibroseis is also common to used for VSP survey, especially in desert area.

In borehole seismic data, there are two important wave. These wave was divived by the way the recorded in receiver, the first wave are downgoing waves that is wave that travels directly from seismic source to the receiver. This wave measure direct P waves transit time which further more used for time-depth information. The second type of the wave is the upgoing wave or the reflection waves. These wave travels from source hitting formation, reflected back and recorded at the receiver. Depending on formation properties, and the type of source, downgoing and upgoing can be consist of compressional wave (P wave) and also shear waves (S wave) , that is downgoing P, downgoing S, upgoing P, and upgoing S. The reflection wave is the one that is required for subsurface imaging in borehole seismic. The exercise for modeling these various wavetype can be seen in the Synthetic Elastic Modeling chapter.

The understanding of various wave type within borehole seismic data is fundamental in order to be able to process, analyze, and interpret VSP data. For example, we need to know which one that we need for time-depth informations, and anisotropy parameter inversion. This is the direct wave in zero offset VSP for time depth informations, and direct wave in multi offset walkaway VSP survey for anisotropy parameter inversion. For structural imaging we required to identify and separate the reflection wave, and this is also depend whether we want to image using P wave or S wave. Besides the primary wave (primary downgoing, and primary reflections) it is also useful if we can estimate the multiple.

Just like any other seismic method, the way we processed the VSP data is more or less the same. At least there are five mandatory steps in processing borehole seismic data for any kind of survey. First step is to break time picking, because this is the basis of any kind of the subsequence processing. This is crucial steps, because from this step we define the velocity model for separation, imaging, etc. Second step is preconditioning the data; this is will be subject to tools and condition that we faced. In example when we processed vectorial data (X-Y-Z), we required to polarize and project the data to 2-D frame or 3-D frame to get true horizontal and true vertical data. And if we required compensating for spherical divergence or statistical variation because of tools in data or when we required removing unwanted frequency. Third steps is wavefield separations, this is also crucial that we extract the type of wave that we want for further analyzed. Normally we try to maximize in extracting the best possible of reflection wave from the total wavefield data. Next mandatory steps is deconvolutions, because we want to remove the multiple. The final steps is imaging, whether we are going to extract 1-D seismic profile for corridor stacks, or using CDP mapping or migration to get 2-D/3-D image.

Latihan VSP SU Scripts

The directory is numbered in a way to sequentially exercise VSP modeling and processing. You can start with directory number 1 (BuildFlatModel) if you want to start with synthetic data processing. Or if you already have field SEGY data, you can start with directory number 0 (ImportSEGY).

File IndexFile List

Here is a list of all documented files with brief descriptions:

0ImportSEGY/ImportSEGY.sh (Convert SEGY to SU format, setting required VSP headers) ??

1BuildFlatModel/CreateFlatModel.sh (Create Simple Flat Model) ??

2Ela2D/CreateElasticSynthetic.sh (Do elastic synthetic modeling using ela2d) ??

3BreakTimePick/FBAutoPick.sh (Automatic FB picking) ??

4Preprocessing/FrequencyAnalysis.sh (Create FZ Spectrum & FK-Spectrum) ??

4Preprocessing/Preprocessing.sh (Do Preprocessing: BPF, Normalize, TVG) ??

5Separation/Separation.sh (Run wavefield separation based on TT using median velocity filter) ??

6Deconvolution/ApplyDecon.sh (Apply PEF to upgoing wavefield) ??

6Deconvolution/CheckAutoCorrelation.sh (Check auto correlation to determine optimum lag and prediction on downgoing wavefield) ??

7CorrStack/CorridorStack.sh (Create corridor stack) ??

File DocumentationFile Documentation

0ImportSEGY/ImportSEGY.sh File Reference

20ImportSEGY/ImportSEGY.sh0ImportSEGY/ImportSEGY.sh

Convert SEGY to SU format, setting required VSP headers.

Detailed Description

The purposes of this script are to convert field VSP SEGY to SU format, and also set a few parameters that is required in further processing.

Currently it is tested for Schlumberger VSI* field SEGY.

Most important header for VSP is receiver depth. And if also possible other field recording information such as acquisition shot number, acquisition shot time, also field transit time picks.

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./ImportSEGY.sh

Output:

- SU File

- Transit Time Header and Receiver Depth in ASCII format (currently tested for SLB VSI SEGY)

Parameters:

i*nput*	Specify your input SEGY file
o*utput*	Specify your output SU file
t*min*	Minimum Output time
t*max*	Maximum Output time

Author:

fahdi@gm2001.net

Date:

February 2012

Script Source:

Filename: ImportSEGY.h

```
#!/bin/bash
# Input
input=VSI_007_A_gac_wavefield_z_resize.sgy
output=realZ.su
# set parameter
tmin=0 #start time
tmax=4 #output time
segyread tape=$input verbose=1 endian=0 | suwind tmin=$tmin tmax=$tmax > $output.tmp1
#housekeeping, just to make output SU available for XWIGB
#this is not a good practice, because I have to strip SEGY headers, will think about this later
sugethw < $output.tmp1 key=lagb,gelev | sed -e ’s/unscale=//’ -e ’s/gelev=//’ -e ’s/lagb=//’| sed ’/^$/d’ > tt-header.tmp
awk ’{ printf "%4f %d\n", $1/1000, $2/10000 }’ tt-header.tmp > tt-header.txt
awk ’{print $2}’ tt-header.txt | a2b > gelev.bin
sushw < $output.tmp1 key=d1,scalel,scalco > $output.tmp2
sushw < $output.tmp2 key=gelev infile=gelev.bin> $output
#display
nrec=($(wc -l tt-header.txt | awk ’{print $1}’))
suxwigb < $output title=$input perc=97 style=vsp key=gelev curve=tt-header.txt npair=$nrec,1 curvecolor=red &
#clean up
```

```
rm *.tmp*
rm *.bin

```

1BuildFlatModel/CreateFlatModel.sh File Reference

21BuildFlatModel/CreateFlatModel.sh1BuildFlatModel/CreateFlatModel.sh

Create Simple Flat Model.

Detailed Description

In this folder we will exercise how to create flat model using unif2 from SU. For simple zero offset VSP, flat model should be sufficient. This scripting is also preparing the required file for Ela2D modeling, such as Vp, Vs, and density file in binary format.

Within this exercise we will also generate shooting geometry that we will use for next exercise in building synthetic data.Clean.sh can be used to clean up Model can be displayed by typing ./DisplayModel.sh

It's also good practice to see unif2 manual.

Clean.sh is used to delete exercise's output file.

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./CreateFlatModel.sh

Output:

- Vp, Vs, and Density in binary format (automatically copied to 2Ela2D directory)

- Source & Receiver Listing in ASCII format (table: depth - offset)

Parameters:

nx Number of model sample in X direction

dx	Model sampling rate in X direction
nz	Number of model sample in Z direction
dz	Model sampling rate in Z direction
whx	Wellhead location X
why	Wellhead location Y
rcv_x	Receiver location X
rcv_spacing	Receiver Spacing
rcv_z0	Depth of first receiver
nrcv	Number of Receiver
offset	Source Offset from wellhead
src_z	Depth of source
ninf	Number of interface in unif2 model
Vp	P velocity for each layers
Vs	S velocity for each layers
density	Density for each layers

Definition in file **CreateFlatModel.sh**.

Author:

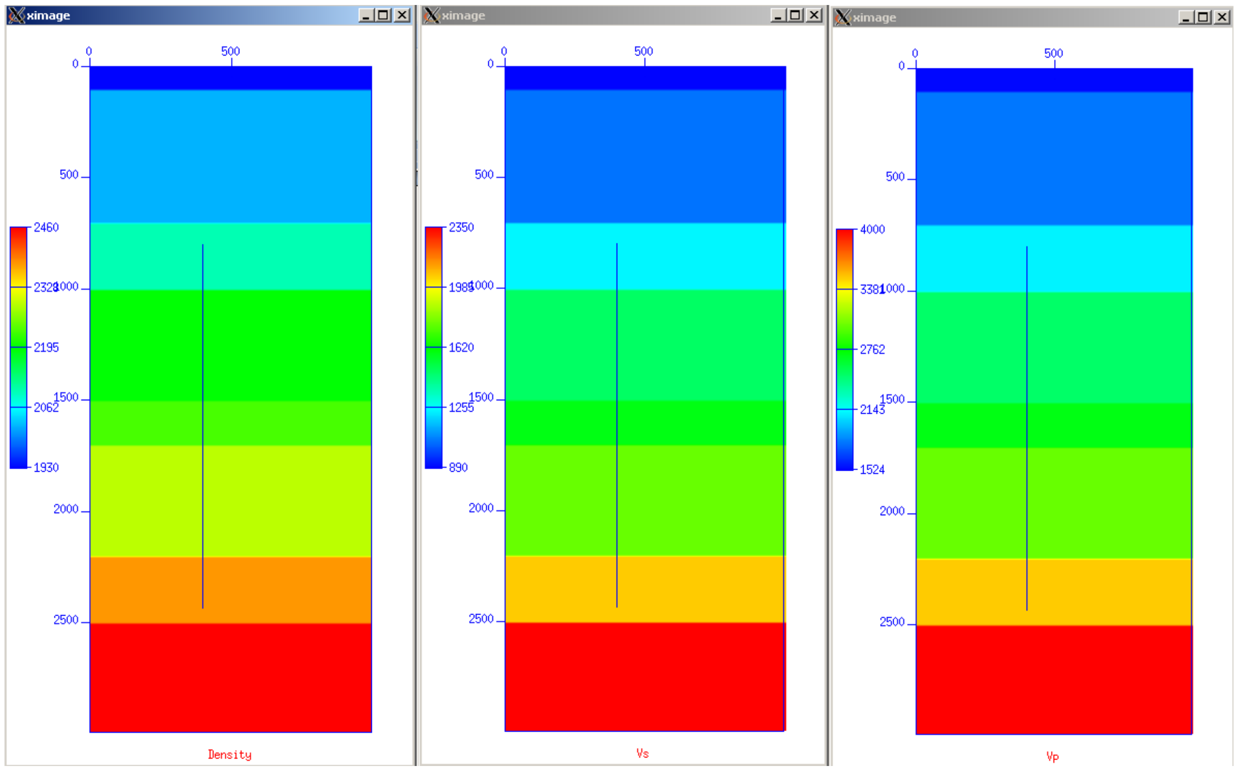
fahdi@gm2001.net

Date:

February 2012

Output Example:

$./DisplayModel.sh


ximage
0
500
0
500
1000
1500
2000
2500
2460
2328
2195
2062
1930
Density
ximage
2350
1985
1620
1255
890
Vs
ximage
4000
3381
2762
2143
1524
Vp

Script Source:

Filename: CreateFlatModel.sh

```
#!/bin/bash

#input
#define model dimension
nx=100
dx=10
nz=300
dz=10


#set parameter
#Define Shooting Geometry
#wellhead location which will be receiver location
whx=400
why=0

#receiver
rcv_x=$((whx))
rcv_spacing=15
rcv_z0=800
nrcv=110

#source, set by offset from wellhead
offset=100
src_z=50


#We will create model where interface defined by $input_model, contain 8 layer
#change the velocity for interface here
#To simply the exercise set Vp/Vs=2, and density is gardner
```

```
#output name is designed for Ela2d (vel_P.dat, vel_S.dat, density.dat)
#see SU documentation for unif2 to modify

input_model=model.unif2
ninf=8 #number of interface, check with $input_model
Vp=1524,1800,2100,2500,2700,3000,3500,4000 #m/s
Vs=890,1050,1235,1470,1588,1765,2058,2350 #m/s
density=1930,2020,2100,2190,2230,2290,2380,2460 #(g/m)?

# input stop here

#populate source
src_x=$((whx+offset))
printf "\t $src_z \t $src_x" > source_list.txt

#Create Elastic Flat Model
#CreateVP
unif2 < model.unif2 nx=$nx nz=$nz dx=$dx ninf=$ninf \
v00=$Vp dz=$dz > vel_P.dat

#CreateVs (Vp/Vs~1.7)
unif2 < model.unif2 nx=$nx nz=$nz dx=$dx ninf=$ninf \
v00=$Vs dz=$dz > vel_S.dat

#CreateDensity
unif2 < model.unif2 nx=$nx nz=$nz dx=$dx ninf=$ninf \
v00=$density dz=$dz > density.dat

#create receiver array
awk -v rcv_z0=${rcv_z0} -v rcv_spacing=${rcv_spacing} -v rcv_x=${rcv_x} -v nrcv=${nrcv} 'BEGIN {
rcv_z=rcv_z0
i=1
while (i <= nrcv) {
```

```
printf "\t %.2f \t %.2f \n",rcv_z,rcv_x
rcv_z=rcv_z+rcv_spacing
i++
}
}' > receiver_list.txt

#display, plot receiver as blue curve
ximage < vel_P.dat n1=$nz d1=$dz n2=$nx d2=$dx legend=1 cmap=hsv2
npair=$nrcv,1 \
curve=receiver_list.txt,source_list.txt title="Vp" &
ximage < vel_S.dat n1=$nz d1=$dz n2=$nx d2=$dx legend=1 cmap=hsv2
npair=$nrcv,1\
curve=receiver_list.txt,source_list.txt title="Vs" &
ximage < density.dat n1=$nz d1=$dz n2=$nx d2=$dx legend=1 cmap=hsv2
npair=$nrcv,1\
curve=receiver_list.txt,source_list.txt title="Density" &

#preparation for 2Ela2d
#copy velocity model
#@todo this should be gone
cp *.dat ../2Ela2d
cp *.txt ../2Ela2d
```

2Ela2D/CreateElasticSynthetic.sh File Reference

22Ela2D/CreateElasticSynthetic.sh2Ela2D/CreateElasticSynthetic.sh

Do elastic synthetic modeling using ela2d.

Detailed Description

In my current understanding, in seismic unix, there is no elastic modeling that can include source-receiver configuration. Suea2df and etc are outputting synthetic model with size of the input model. Therefore for the purposes of this exercise, we will be using Ela2D.
The input for this exercise is binary model (Vp,Vs,density) with [nx][nz], besides the flat model that generate earlier, you can make your own model. Output of this Ela2D modeling is horizontal and vertical component. You can edit the source-list.txt and receiver-list.txt as per your requirement. For source, I think we stick to use single source first.
Clean.sh can be used to clean up. Synthetic data can be displayed by typing ./DisplaySynthetic.sh

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./CreateElasticSynthetic.sh

Output:

- Vp, Vs, and Density in binary format (automatically copied to 2Ela2D directory)

- Source & Receiver Listing in ASCII format (table: depth - offset).

Parameters: * update with source listing

nx Number of model sample in X direction

d*x*	Model sampling rate in X direction
n*z*	Number of model sample in Z direction
d*z*	Model sampling rate in Z direction
o*utputX*	Output for horizontal component
o*utputZ*	Output for vertical component
s*ource_listing*	ASCII file for source listing (filename:source_list.txt, please do not change the filename)
r*eceiver_listing*	ASCII file for receiver listing (filename:receiver_list.txt, please do not change the filename)

Definition in file **CreateElasticSynthetic.sh**.

Author:

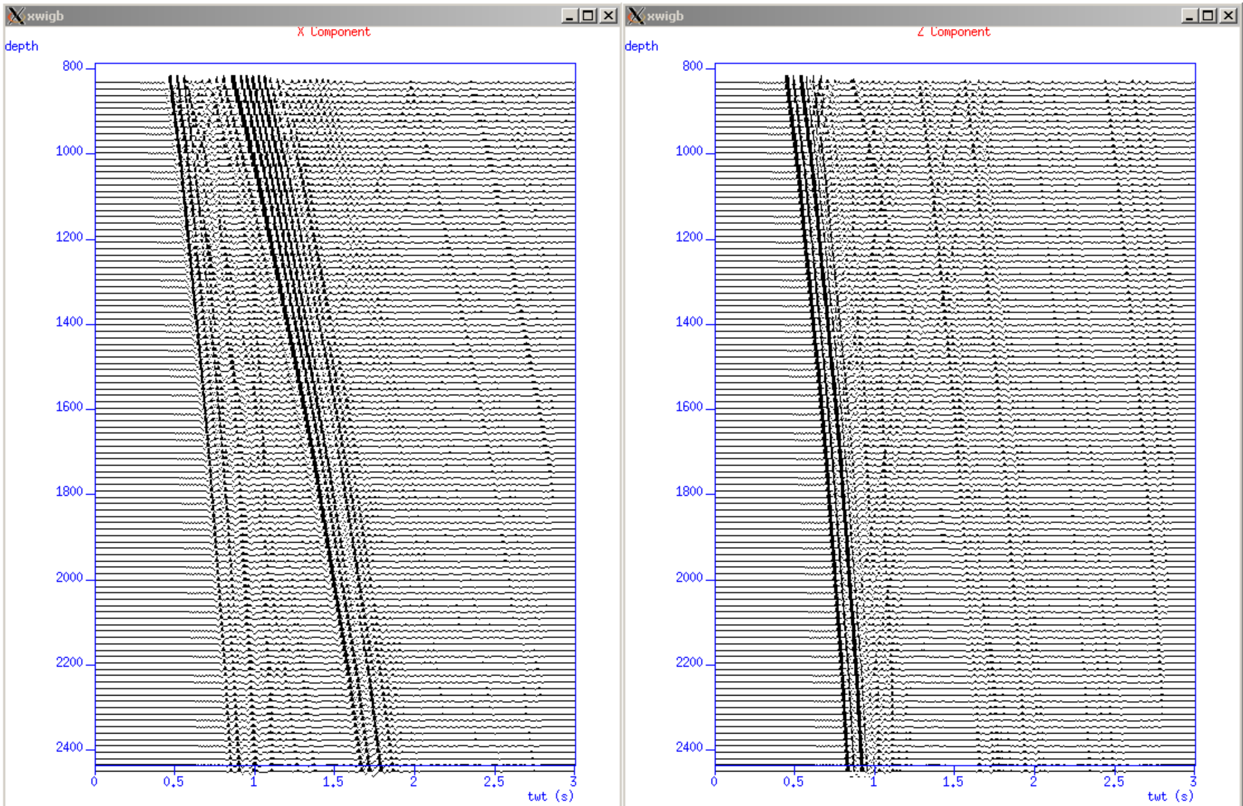
fahdi@gm2001.net

Date:

February 2012

Output Example:

$./DisplaySynthetic.sh



Synthetic Data is usable to identify VSP wave type

Script Source:

Filename: CreateElasticSynth.sh

```
#!/bin/bash

#input
#define model dimension
nx=100
dx=10
nz=300
dz=10
outputX=HMX.su #specify filename for horizontal component
outputZ=Z.su #specify filename for vertical component

#set parameter
#Ela2D FD setup
# * @todo this dt is for FD timestep modeling, should use appropriate dt for time step modeling
# * @todo implement resampling for final su data
dt=0.002 #set timestep and will also be samplin rate for out put data, @TODO CHECK!!
tmax=3 #maximum recording time
f_cent=30 #dominant frequency

#source
#setting source, use *.txt from 1BuildFlatModel, convert it to binary for SU header requirement
#DO NOT MODIFY LINE AFTER THIS, UNLESS YOU KNOW WHAT YOU ARE DOING

#copy receiver from 1BuildFlatModel
cp ../1BuildFlatModel/*.dat .
cp ../1BuildFlatModel/*.txt .

#flip receiver list for Ela2D requirement
awk '{print "\t", $2, "\t", $1}' receiver_list.txt > receiver_list.tmp
```

```
awk '{print "\t", $2, "\t", $1}' source_list.txt > source_list.tmp
mv receiver_list.tmp receiver_list.txt
mv source_list.tmp source_list.txt

a2b < source_list.txt > source_list.bin
src_x=($(awk '{print $1}' source_list.txt))
src_z=($(awk '{print $2}' source_list.txt))
#load receiver
a2b < receiver_list.txt > receiver_list.bin
nrec=($(wc -l receiver_list.txt | awk '{print $1}'))
ns=($(echo $tmax/$dt+1 |bc -l))

#substitute all input variables to Ela2D param & in file
sed -e "s/src_x/$src_x/" -e "s/src_z/$src_z/" \
-e "s/seddt/$dt/" -e "s/sedtmax/$tmax/" -e "s/sedf_cent/$f_cent/" \
-e "s/nrec/$nrec/" \
< ./template/ela2d.in.template > ela2d.in.tmp

sed -e "s/seddx/$dx/" -e "s/seddz/$dz/" \
-e "s/sednx/$nx/" -e "s/sednz/$nz/" \
< ./template/grid.param.template > grid.param

cat ela2d.in.tmp receiver_list.txt > ela2d.in

#Running Ela2D
echo "Start Ela2D"
ela2d

echo "Convert Ela2D output to SU Format"
dtms=($(echo $dt*1000000 |bc -l))
suaddhead < seismicX.H@ ns=$ns |
sushw key=dt a=$dtms |
sushw key=gx,gelev infile=receiver_list.bin |
sushw key=sx a=$src_x b=0 j=$nrec |
```

```
sushw key=selev a=$src_z b=0 j=$nrec > $outputX
suaddhead < seismicZ.H@ ns=$ns |
sushw key=dt a=$dtms |
sushw key=gx,gelev infile=receiver_list.bin |
sushw key=sx a=$src_x b=0 j=$nrec |
sushw key=selev a=$src_z b=0 j=$nrec > $outputZ
#clean Up
mv *.H* *.out ela2d_dir
rm *.tmp *.bin
echo "Ela2D Finished"
#display
suxwigb < $outputX title="X Component" perc=97 style=vsp key=gelev label2="depth" label1="twt (s)" &
suxwigb < $outputZ title="Z Component" perc=97 style=vsp key=gelev label2="depth" label1="twt (s)" &
```

3BreakTimePick/FBAutoPick.sh File Reference

23BreakTimePick/FBAutoPick.sh3BreakTimePick/FBAutoPick.sh

Automatic FB picking.

Detailed Description

Break time picking is the most crucial for every VSP job. Every VSP analysis is based and required break time pick. SU provides good tools but not yet sophisticated and user friendly to do manual break time pick. The script for manual picking using ximage and button [s] functionality is ***BreakTimePick.sh.***
We are going to use automatic break time picking from SU, and smooth it a little bit using octave script. A QC of breaktime picking, besides visual QC, is to compute interval velocity and compare it with sonic lo velocity; this will be done on DisplayPicks.sh.
The feature of transit time correction to seismic reference datum (SRD) through cosine correction has not been implemented.

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./CreateElasticSynthetic.sh

*** Before running the script, please manually copy the SU file that you want to pick to this directory , for example:

$cp ../2Ela2D/Z.su

.

Output:

- SU File after automatic picking, the breaktime pick saved in *unscale* header

- ASCII File containing Transit Time picking (format: [Transit Time][Depth]

Parameters:

input SU file to be picked

o*utput*	SU file after picked
o*utput_picks*	ASCII file for automatic Transit Time pick result

Definition in file **FBAutoPick.sh**.

Author:

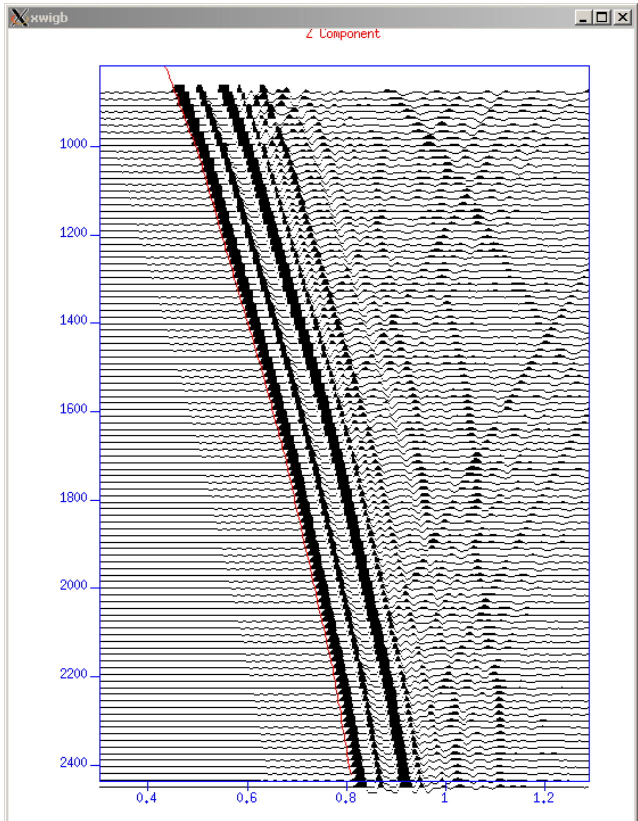
fahdi@gm2001.net

Date:

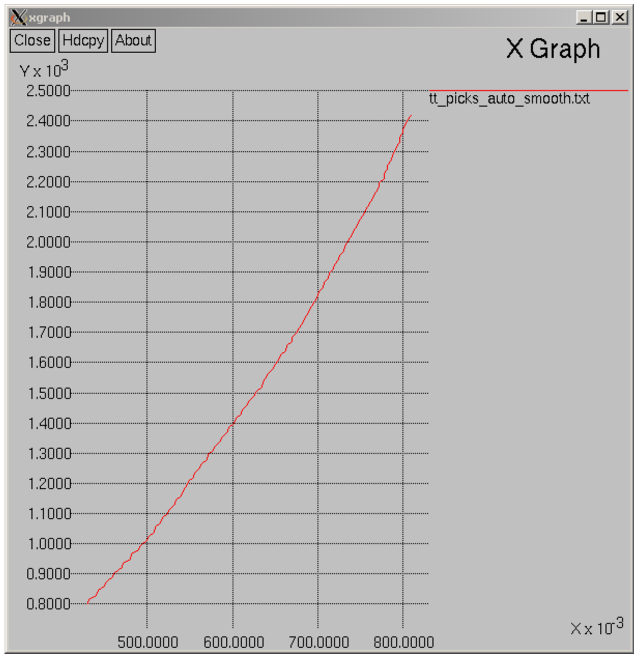
February 2012

Output Example:

$./DisplayPicks.sh



Breaktime pick displayed with red curve



Time-Depth Curve

Script Source:

Filename: FBAutoPick.sh

```
#!/bin/bash
#input
input=Z.su
output=Z_picked.su #breaktime will saved at unscale header in this file
output_picks=tt_picks_auto.txt #ASCII output of breaktime picking
#process
#do automatic picking
sufbpickw < $input > $output
#a little housekeeping
sugethw < $output key=unscale,gelev | sed -e ’s/unscale=//’ -e ’s/gelev=//’| sed ’/^$/d’ | sort -bn -k 2 | uniq > $output_picks
nrec=($(wc -l tt_picks_auto.txt | awk ’{print $1}’)) #calculate number of receiver
#display data and picks
suxwigb < $input title="Z Component" perc=97 style=vsp key=gelev curve=$output_picks npair=$nrec,1 curvecolor=red label2="depth" label1="twt (s)"&
#smoothing
octave –silent –eval "ttSmoothing(’$output_picks’)";
#clean up
rm test.su
```

4Preprocessing/FrequencyAnalysis.sh File Reference

24Preprocessing/FrequencyAnalysis.sh4Preprocessing/FrequencyAnalysis.sh

Create FZ Spectrum & FK-Spectrum.

Detailed Description

Analysis of frequency content in VSP is also significant process. If we compute frequency spectrum of each traces, we are going to have frequency profile each depth. These profiles will also describing quality factor/attenuation. FK Spectrum is also important to see how far the aliasing is.

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./FrequencyAnalysis.sh

*** Before running the script, please manually copy the SU file that you want to pick to this directory , for example:

$cp ../0ImportSEGY/realZ.su .

$cp ../0ImportSEGY/tt-header.txt .

Output:

- SU File for Frequency vs Depth & Frequency vs Wavenumber

Parameters:

input SU file to analyze

o*utput_fz*	Spectrum of Frequency vs Depth in SU format
o*utput_fk*	Spectrum of Frequency vs Wavenumber in SU format

Definition in file **FrequencyAnalysis.sh**.

Author:

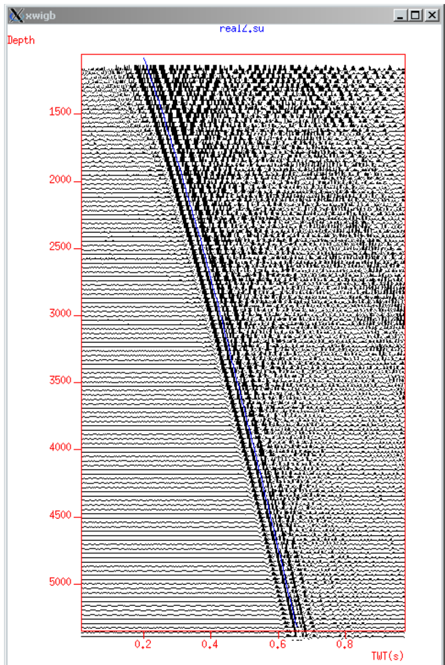
fahdi@gm2001.net

Date:

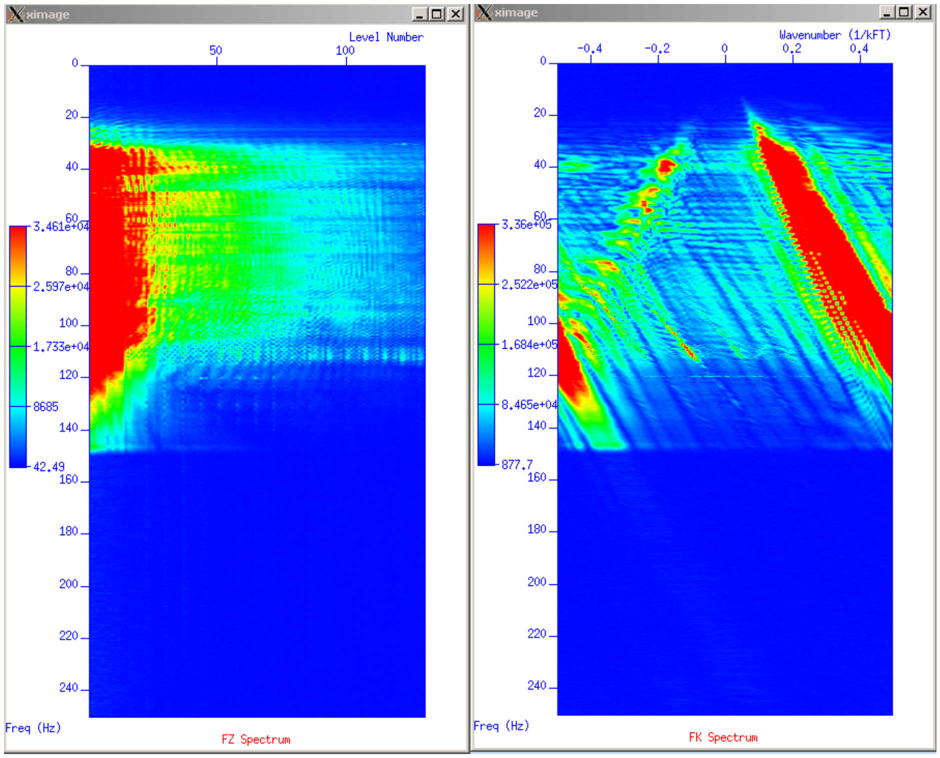
February 2012

Output Example:

$./FrequencyAnalysis.sh



Real Data- Note The break time on red curve.



The FZ and FK Spectrum

Script Source:

Filename: FBAutoPick.sh

```
#!/bin/bash

#input
input=realZ.su
output_fz=z-spec-fz.su
output_fk=z-spec-fk.su

#compute fz
suspecfx < $input > $output_fz

#compute fk
suspecfk < $input > $output_fk

#display
nrec=($(wc -l tt-header.txt | awk '{print $1}'))
suxwigb < $input title=$input perc=97 style=vsp key=gelev curve=tt-header.txt npair=$nrec,1 curvecolor=red label2="Depth" label1="TWT(s)"&

#displayfz
suximage < $output_fz cmap=hsv2 perc=95 legend=1 title='FZ Spectrum' label1='Freq (Hz)' label2='Level Number' &

#displayfk
suximage < $output_fk cmap=hsv2 perc=95 legend=1 title='FK Spectrum' label1='Freq (Hz)' label2='Wavenumber (1/kFT)' &
```

4Preprocessing/Preprocessing.sh File Reference

24Preprocessing/Preprocessing.sh4Preprocessing/Preprocessing.sh

Do Preprocessing: BPF, Normalize, TVG.

Detailed Description

This step is to perform preprocessing before we process VSP signal.
The BPF is required to eliminate noise based on their frequency content. The frequency of data needs to be analyzed first; you can use FrequencyAnalysis.sh script to do this
For some VSP tools, the coupling between tools and formation is not the same, and sometimes there is energy variation between shot to shot, which makes median filter not working properly. This is compensated by statistically normalize the data based on downgoing event. SU does not have the capability to do RMS calculation based on isolated downgoing event (say analyze for 200 ms after first break), but using sugain to normalize the data should be enough for this exercise.
The Time Varying Gain/Exponential Gain is required to correct the spherical divergence effect.
Workflow that has been set on Preprocessing.sh is just based on personal experience; you can modify it as necessary. The workflow is BPF + RMS Normalization + TimeVaryingGain

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./Preprocessing.sh

*** Before running the script, please manually copy the SU file that you want to pick to this directory , for example:

$cp ../0ImportSEGY/realZ.su

Output:

- SU file for each process (after BPF, after BPF+Normalization, after BPF+Normalization+TimeVaryingGain

- Final SU after preprocessing with proper filenaming

Parameters:

input SU file to process

output_bpf	output SU file after BPF filter
b*pf*	Four points of BPF filter
output_norm	output SU file after normalization (RMS whole window operation)
output_tvg	SU file after TimeVaryingGain (
t*pow*	TVG constant
final_output	Output after all preprocessing workflow in SU format

Definition in file **Preprocessing.sh**.

Author:

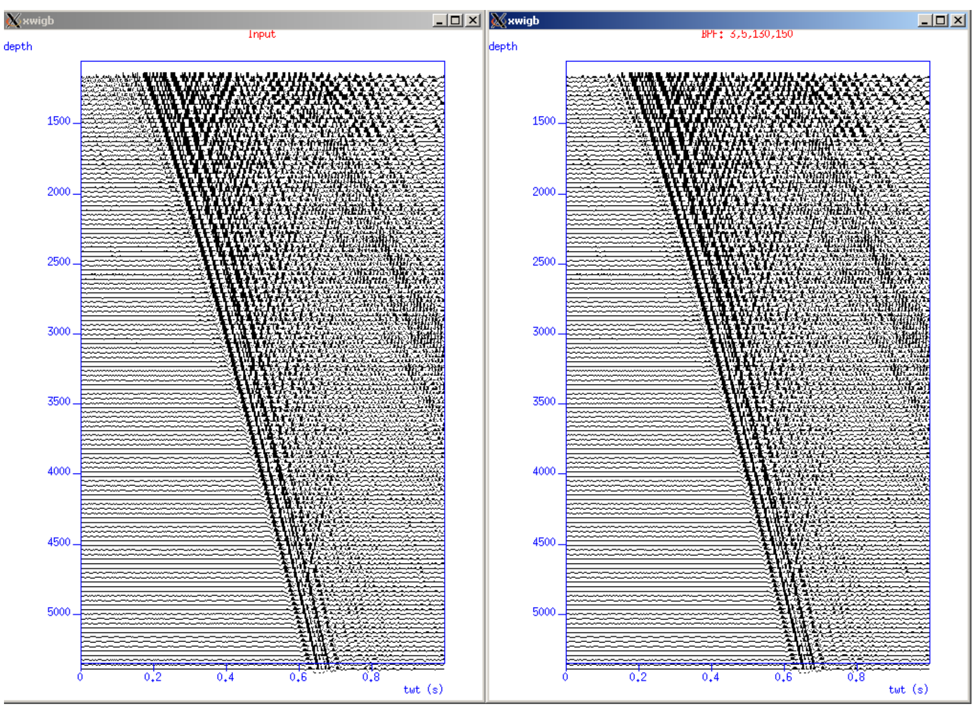
fahdi@gm2001.net

Date:

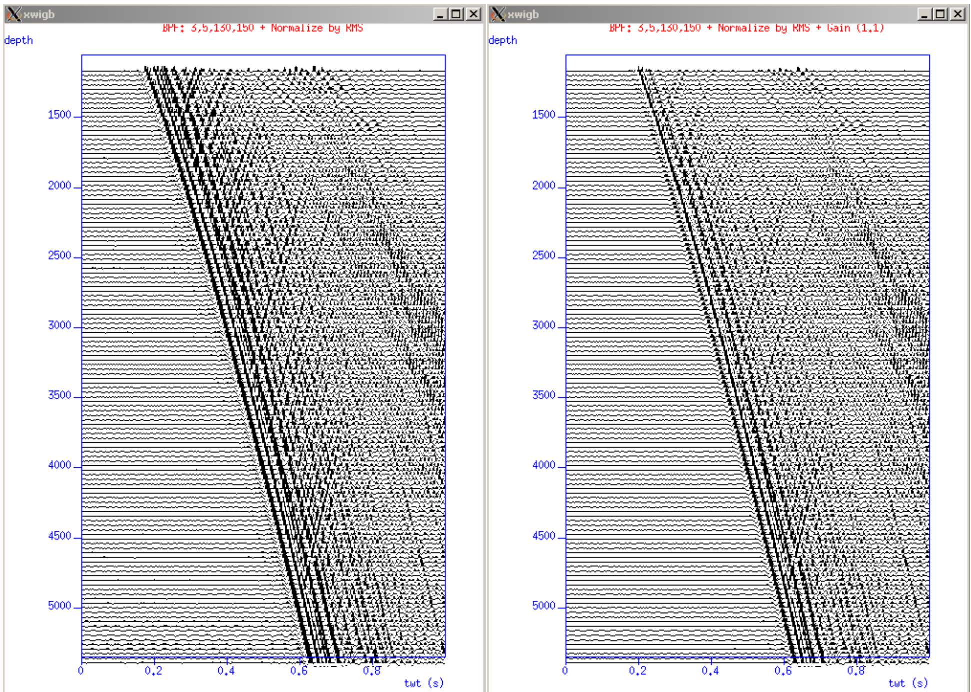
February 2012

Output Example:

$./Preprocessing.sh



Input (left), Input after BPF (right)



Input after BPF+Normalize (left) Input after BPF+Normalize+TimeVaryingGain (right)

Script Source:

Filename: Preprocessing.sh

```
#!/bin/bash

#input
input=realZ.su
output_bpf=Z_picked_bpf.su # output after BPF
output_norm=Z_picked_bpf_norm.su #output after BPF followed RMS
Normalization
```

```
output_tvg=Z_picked_bpf_norm_tvg.su #output after BPF followed RMS Normalization followed by TimeVaryingGain
final_output=Z_prepro.su #housekeeping to make naming convention
#set parameter
bpf=3,5,130,150 #4 points bandpass specification
tpow=1.1 #multiply data by t^tpow
#bpf
sufilter < $input f=$bpf > $output_bpf
#normalize by dividing with RMS
sugain < $output_bpf pbal=1 > $output_norm
#run exponential gain
sugain < $output_norm tpow=$tpow > $output_tvg
cp $output_tvg $final_output #housekeeping to make naming convention
#display
suxwigb < $input title="Input" perc=97 style=vsp key=gelev label2="depth" label1="twt (s)" x1beg=0.0 x1end=1.0 xbox=10 wbox=500&
suxwigb < $output_bpf title="BPF: $bpf" perc=97 style=vsp key=gelev label2="depth" label1="twt (s)" x1beg=0.0 x1end=1.0 xbox=520 wbox=500&
suxwigb < $output_norm title="BPF: $bpf + Normalize by RMS" perc=97 style=vsp key=gelev label2="depth" label1="twt (s)" x1beg=0.0 x1end=1.0 xbox=10 wbox=500&
suxwigb < $output_tvg title="BPF: $bpf + Normalize by RMS + Gain ($tpow)" perc=97 style=vsp key=gelev label2="depth" label1="twt (s)" x1beg=0.0 x1end=1.0 xbox=520 wbox=500&
```

5Separation/Separation.sh File Reference

25Separation/Separation.sh5Separation/Separation.sh

Run wavefield separation based on TT using median velocity filter.

Detailed Description

Wavefield separation is another crucial step in VSP processing. The process is to get the upgoing (reflection wave) from the total wavefield. As we have seen between downgoing and upgoing can be differentiate clearly by their slope, downgoing is positive towards depth, and upgoing is negative. Based on these two different slopes, we can separate this using velocity filter (slope ~ velocity), by aligning them along the slopes. After alignment, the signal that has same slopes will be coherent, and using median filter, you can get coherent signal, and the uncoherent signal will be the residual signal (subtracted from total signal).
For example, to extract downgoing, you will align the data by subracting using TT, the downgoing will be coherent along the TT, and you can run median filter to extract this downgoing. The uncoherent signal will be the residual.
Normally the workflow is you extract downgoing first, and subtract this downgoing from the total wavefield. You will get residual wavefield (residual1 = total-downgoing). And you run second velocity filter, to extract upgoing by aligning to the upgoing slope, the input for this is the residual wavefield.
Input for separation is normally wavefield after preprocessing.

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./Separation.sh

*** Before running the script, please manually copy the SU file that you want to pick to this directory , for example:

$cp ../4Preprocessing/Z_prepro.su .

$cp ../3BreakTimePick/tt-header.txt .

Output:

- SU file for each process (downgoing wavefield, residual after downgoing removal, upgoing wavefield, residual after after upgoing wavefield extraction

Parameters:

input SU file to separate after PreProcessing

output_dn	Downgoing wavefield
output_res	Residual Wavefield after downgoing wavefield subtraction
output_up	Upgoing wavefield (Enhanced Residual)
output_res2	2nd Residual after Upgoing extraction
timePicks	ASCII file containing transit time picking
level_down	number median level for downngoing extraction
level_up	number median level for upgoing extraction

Definition in file **Separation.sh**.

Author:

fahdi@gm2001.net

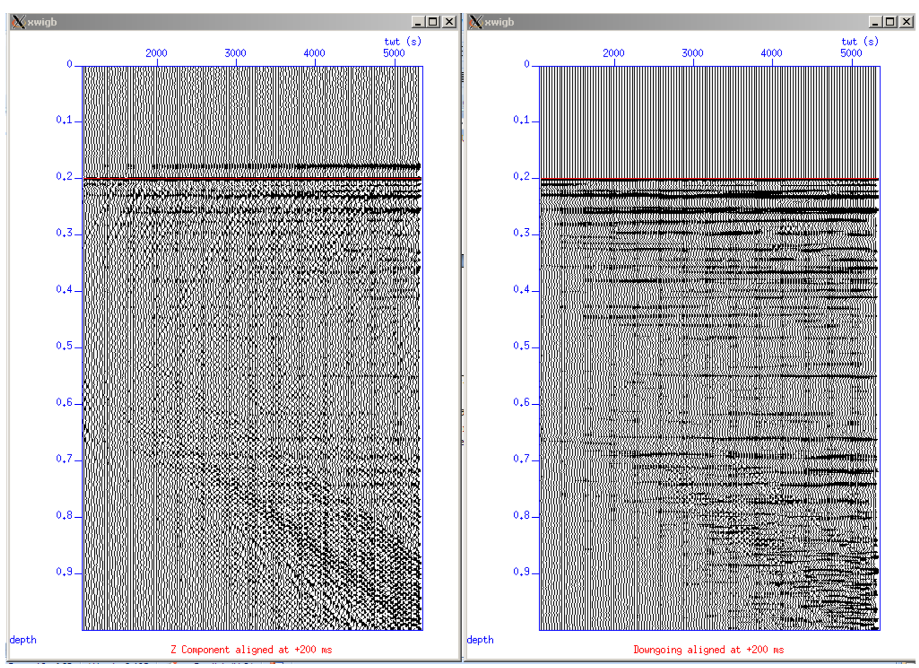
Date:

February 2012

Output Example:

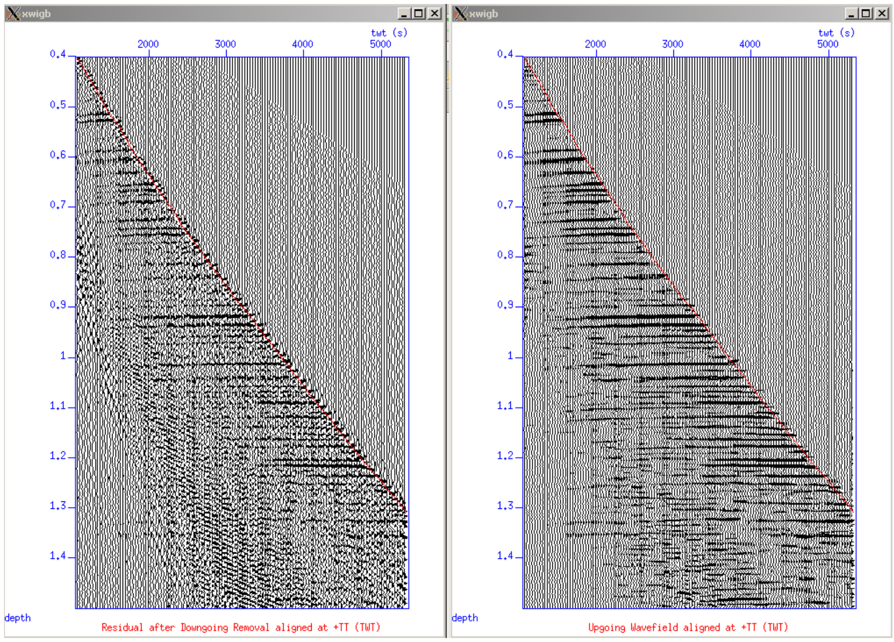
$./Separation.sh&

$./DisplaySeparation.sh &



Z component total wavefield aligned at -TT (left) and Downgoing Wavefield

(right)



Residual wavefield after downgoing wavefield removal aligned at twt (left) upgoing wavefield aligned at twt (right)

Script Source:

Filename: Separation.sh

```
#!/bin/bash
#input
input=Z_prepro.su
output_dn=velf_dn.su
output_res=velf_res.su
output_up=velf_up.su
output_res2=velf_res2.su
timePicks=tt-header.txt
#set parameter
level_down=9
level_up=7
#housekeeping
nrec=($(wc -l $timePicks | awk ’{print $1}’)) #housekeeping, check number of receiver
awk ’{print $2}’ $timePicks > xfile.tmp
a2b < xfile.tmp n1=$nrec> xfile.bin
awk ’{print $1}’ $timePicks > tfile.tmp
a2b < tfile.tmp n1=$nrec> tfile.bin
#processing
sumedian < $input xfile=xfile.bin tfile=tfile.bin key=gelev nshift=$nrec subtract=0 median=1 nmed=$level_down sign=-1> $output_dn
sumedian < $input xfile=xfile.bin tfile=tfile.bin key=gelev nshift=$nrec subtract=1 median=1 nmed=$level_down sign=-1> $output_res
sumedian < $output_res xfile=xfile.bin tfile=tfile.bin key=gelev
```

```
nshift=$nrec subtract=0 median=1 nmed=$level_up sign=+1> $output_up
sumedian < $output_res xfile=xfile.bin tfile=tfile.bin key=gelev
nshift=$nrec subtract=1 median=1 nmed=$level_up sign=+1> $output_res2

#suxwigb < HMX.su title="HMX" perc=97 style=vsp key=gelev
curve=$timePicks npair=$nrec,1 curvecolor=red label2="depth"
label1="twt (s)"&
suxwigb < $input title="Z" perc=97 style=vsp key=gelev curve=$timePicks
npair=$nrec,1 curvecolor=red label2="depth" label1="twt (s)"&
suxwigb < $output_dn title="Downgoing" perc=97 style=vsp key=gelev
curve=$timePicks npair=$nrec,1 curvecolor=red label2="depth"
label1="twt (s)"&
suxwigb < $output_res title="Residual-1" perc=97 style=vsp key=gelev
curve=$timePicks npair=$nrec,1 curvecolor=red label2="depth"
label1="twt (s)"&
suxwigb < $output_up title="Upgoing" perc=97 style=vsp key=gelev
curve=$timePicks npair=$nrec,1 curvecolor=red label2="depth"
label1="twt (s)"&
suxwigb < $output_res2 title="Residual-2" perc=97 style=vsp key=gelev
curve=$timePicks npair=$nrec,1 curvecolor=red label2="depth"
label1="twt (s)"&
```

6Deconvolution/CheckAutoCorrelation.sh File Reference

26Deconvolution/CheckAutoCorrelation.sh6Deconvolution/CheckAutoCorrelation.sh

Check auto correlation to determine optimum lag and prediction on downgoing wavefield.

Detailed Description

Deconvolution is needed to remove multiple. In VSP, we can design the decon operator based on downgoing wavefield. Because the reverberation of multiple was recorded on downgoing wavefield.

Within SU, we can try to use predictive decon (supef/PEF). Using downgoing wavefield autocorrelation, we estimated the multiple lag, and gap. After we satisfied with the parameter for downgoing wavefield, we applied to upgoing wavefield.
The decon in this exercise was divided into two scripts, the first script is CheckAutoCorrelation.sh, this is basically computing autocorrelation on downgoing wavefield and apply it on downgoing wavefield. If PEF result on downgoing is ok, we will use the same parameter on the upgoing decon.
[NEED QC]

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./CheckAutoCorrelation.sh

*** Before run, please manually copy the SU file that you want to process to this directory,

$cp ../5Separation/velf_up.su .

$cp ../5Separation/velf_down.su .

$cp ../0ImportSEGY/tt-header.txt .

Output:

- SU file for downgoing after PEF decon

Parameters:

input_dn Downgoing wavefield

t*t*	ASCII file of Transit Time picks data
o*utput_pef_dn*	Downgoing wavefield after predictive deconvolution
m*inlag*	First lag of prediction filter (sec)
m*axlag*	lag
p*noise*	relative additive noise level
b*pf*	Four points of BPF filter

Definition in file **CheckAutoCorrelation.sh**.

Author:

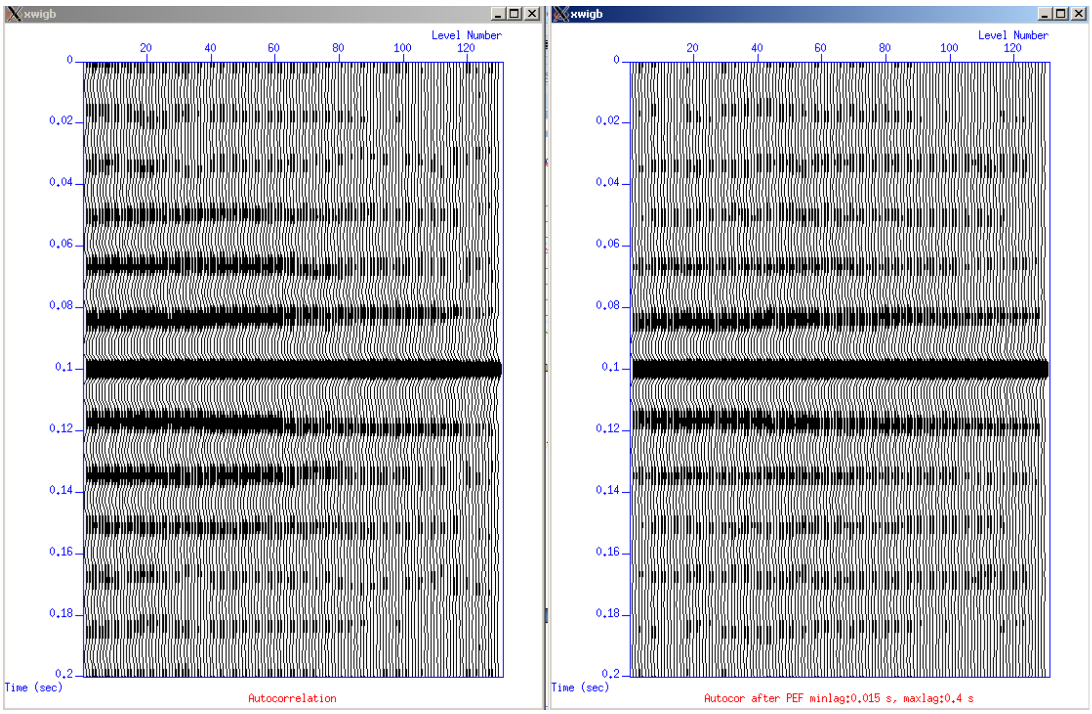
fahdi@gm2001.net

Date:

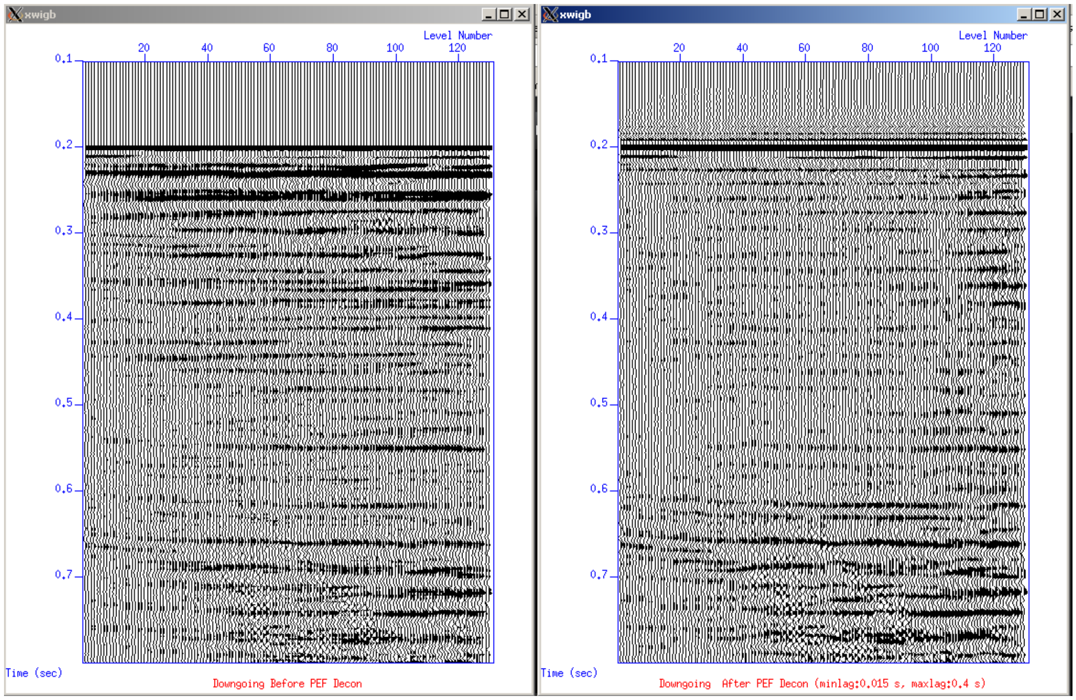
February 2012

Output Example:

$./CheckAutoCorrelation.sh &



Autocorrelation QC, before (left) and after PEF (right)



Downgoing wavefield before (left) and after (right) PEF decon

Script Source:

Filename: CheckAutoCorrelation.sh

```
#!/bin/bash

#input
input_dn=velf_dn.su
tt=tt-header.txt
output_pef_dn=pef_dn.su

#set parameter
minlag=0.015 #s
maxlag=0.4 #s
pnoise=0.0005
bpf=3,5,120,140 #4 points bandpass specification

#align downgoing
fac=-1 # (+1) aligned TWT upgoing , (-1) aligned downgoing
al=200 # 0 for TWT upgoing, 200 for aligned at 200 msec
tmin=0
tmax=5
awk '{print $1*1000}' $tt > tt.tmp
a2b <tt.tmp > tt_header.bin

#check autocorrelation before PEF
sugain < $input_dn tpow=1 | suacor sym=1 norm=1 \
| suxwigb perc=95 label1="Time (sec)" label2="Level Number" title="Autocorrelation" wbox=550 xbox=10&
#Apply to Attack Reveberations
supef < $input_dn minlag=$minlag maxlag=$maxlag pnoise=$pnoise | sufilter f=$bpf > $output_pef_dn
sugain < $output_pef_dn tpow=1 | suacor sym=1 norm=1 \
| suxwigb title="Autocor after PEF minlag:$minlag s, maxlag:$maxlag s" perc=95 label1="Time (sec)" label2="Level Number" wbox=550 xbox=570&
```

```
#display pef application on downgoing signal
sushw < $input_dn infile=tt_header.bin key=delrt | suchw key1=delrt key2=delrt a=$al b=$fac \
| sushift tmin=$tmin tmax=$tmax|suwind tmin=0.1 tmax=0.8> $input_dn.align.tmp
sushw < $output_pef_dn infile=tt_header.bin key=delrt | suchw key1=delrt key2=delrt a=$al b=$fac \
| sushift tmin=$tmin tmax=$tmax|suwind tmin=0.1 tmax=0.8> $output_pef_dn.align.tmp

suxwigb < $input_dn.align.tmp perc=95 title="Downgoing Before PEF Decon" label1="Time (sec)" label2="Level Number" wbox=550 xbox=10&
suxwigb < $output_pef_dn.align.tmp perc=95 title="Downgoing After PEF Decon (minlag:$minlag s, maxlag:$maxlag s)" label1="Time (sec)" label2="Level Number" wbox=550 xbox=570&
```

6Deconvolution/ApplyDecon.sh File Reference

Deconvolution/ApplyDecon.sh6Deconvolution/ApplyDecon.sh

Apply PEF to upgoing wavefield.

Detailed Description

This is sequential script after CheckAutoCorrelation.sh. In previous script we estimated the minlag and maxlag for downgoing wavefield. If the PEF decon is working as expected, we can collapse the reverberations, and the minlag & maxlag can also be applied for upgoing wavefield.
We can apply another median filter to remove noise after predictive decon.

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./ApplyDecon.sh

*** Before run, please manually copy the SU file that you want to process to this directory,

$cp ../5Separation/velf_up.su .

$cp ../5Separation/velf_down.su .

$cp ../0ImportSEGY/tt-header.txt .

Output:

- SU file for upgoing after PEF decon

- Enhanced upgoing SU file

Parameters:

input_up Upgoing wavefield to be deconvolved

tt	ASCII file of Transit Time picks data
output_pef_up	Upgoing Wavefield after predictive decon
output_pef_up_enh	Enhancement of Upgoing Wavefield after predictive decon
enhc_up	Number of median level for upgoing enhancement
minlag	First lag of prediction filter (sec)
maxlag	lag
pnoise	relative additive noise level
bpf	Four points of BPF filter

Definition in file **ApplyDecon.sh**.

Author:

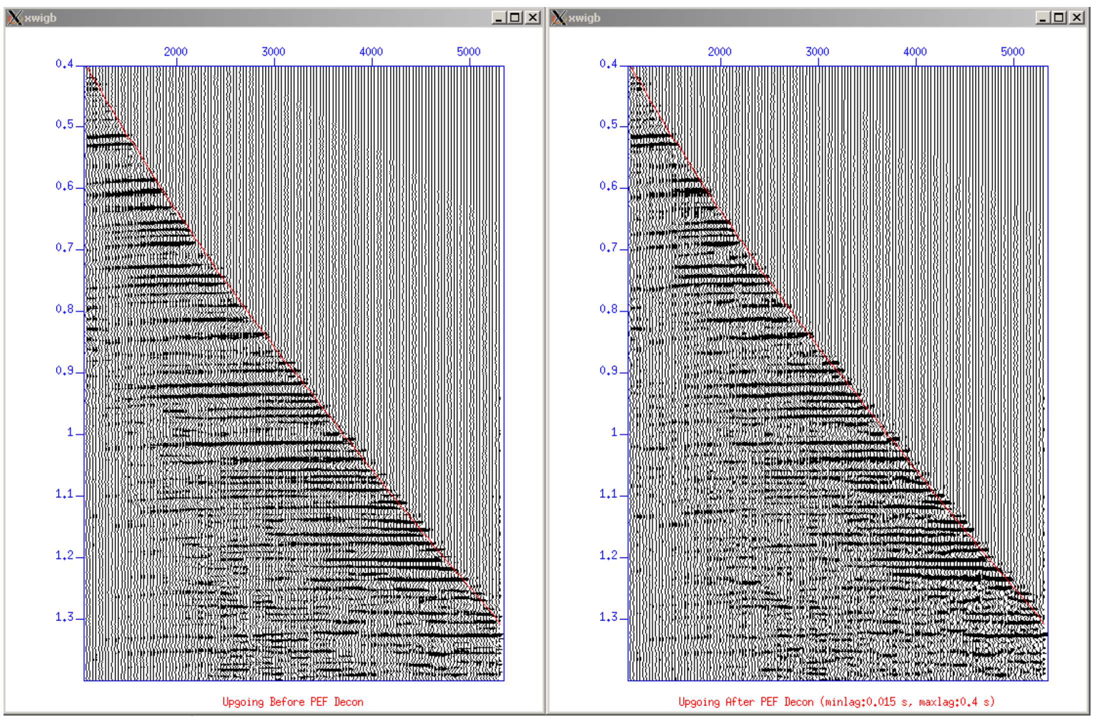
fahdi@gm2001.net

Date:

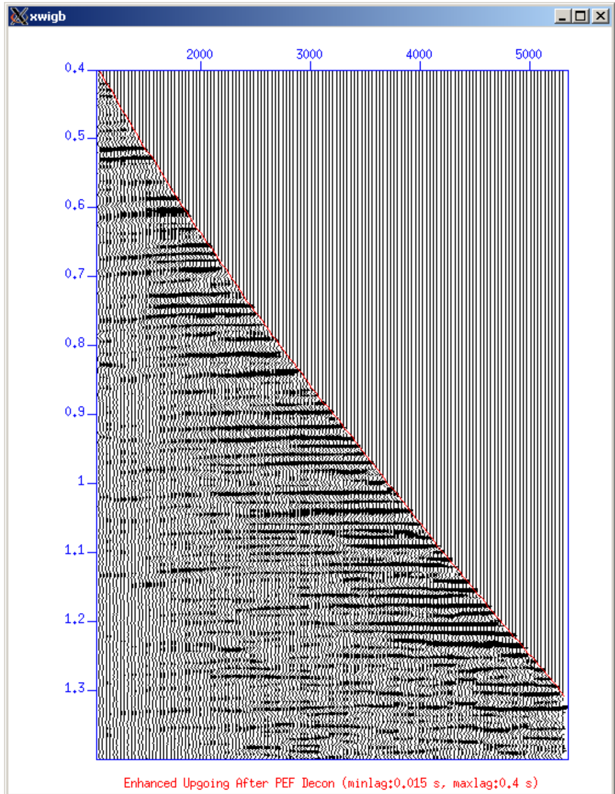
February 2012

Output Example:

$./ApplyDecon.sh &



Upgoing wavefield before decon (left) and after decon (right)



Enhanced Deconvolved upgoing wavefield

Script Source:

Filename: ApplyDecon.sh

```
#!/bin/bash
#input
input_up=velf_up.su
tt=tt-header.txt
output_pef_up=pef_up.su
output_pef_up_enh=pef_up_enh.su
enhc_up=5
#set parameter
minlag=0.015 #s
maxlag=0.4 #s
pnoise=0.0005
bpf=3,5,120,140 #4 points bandpass specification
#housekeeping-start
awk '{print $1*0+0.200,$2}' $tt > minustt.tmp
awk '{print $1*2,$2}' $tt > twt.tmp
#align
fac=1 # (+1) aligned TWT upgoing , (-1) aligned downgoing
al=0 # 0 for TWT upgoing, 200 for aligned at 200 msec
tmin=0
tmax=5
awk '{print $1*1000}' $tt > tt.tmp
a2b <tt.tmp > tt_header.bin
nrec=($(wc -l $tt | awk '{print $1}')) #housekeeping
#apply enhancement to pef up
awk '{print $2}' $tt > xfile.tmp
```

```
a2b < xfile.tmp n1=$nrec> xfile.bin
awk '{print $1}' $tt > tfile.tmp
a2b < tfile.tmp n1=$nrec> tfile.bin
#housekeeping-end
#check autocorrelation before PEF
#sugain < $input_up tpow=1 | suacor sym=1 norm=1 \
# | suxwigb perc=95 label1="Time (sec)" label2="Level Number" title="Autocorrelation" &
#Apply PEF
supef < $input_up minlag=$minlag maxlag=$maxlag pnoise=$pnoise | sufilter f=$bpf > $output_pef_up
#sugain < $output_pef_up tpow=1 | suacor sym=1 norm=1 \
# | suxwigb title="Autocor after PEF minlag:$minlag s, maxlag:$maxlag s" perc=95 &
sumedian < $output_pef_up xfile=xfile.bin tfile=tfile.bin key=gelev nshift=$nrec subtract=0 median=1 nmed=$level_up sign=+1 \
| sumute key=gelev nmute=$nrec xfile=xfile.bin tfile=tfile.bin mode=0 > $output_pef_up_enh
#display pef application on upgoing signal
sushw < $input_up infile=tt_header.bin key=delrt | suchw key1=delrt key2=delrt a=$al b=$fac \
| sushift tmin=$tmin tmax=$tmax|suwind tmin=0 tmax=$tmax> $input_up.align
sushw < $output_pef_up infile=tt_header.bin key=delrt | suchw key1=delrt key2=delrt a=$al b=$fac \
| sushift tmin=$tmin tmax=$tmax|suwind tmin=0 tmax=$tmax> $output_pef_up.align
sushw < $output_pef_up_enh infile=tt_header.bin key=delrt | suchw key1=delrt key2=delrt a=$al b=$fac \
```

```
| sushift tmin=$tmin tmax=$tmax|suwind tmin=0 tmax=$tmax>
    $output_pef_up_enh.align

suwind < $input_up.align tmin=0.4 tmax=1.4 | suxwigb perc=95
title="Upgoing Before PEF Decon" key=gelev curve=twt.tmp npair=$nrec,1
curvecolor=red &
suwind < $output_pef_up.align tmin=0.4 tmax=1.4 | suxwigb perc=95
    title="Upgoing After PEF Decon (minlag:$minlag s, maxlag:$maxlag s)"
    key=gelev curve=twt.tmp npair=$nrec,1 curvecolor=red &
    suwind < $output_pef_up_enh.align tmin=0.4 tmax=1.4 | suxwigb
perc=95
        title="Enhanced Upgoing After PEF Decon (minlag:$minlag s, \
maxlag:$maxlag s)" key=gelev curve=twt.tmp npair=$nrec,1
curvecolor=red &

#clean up
#rm *.bin
#rm *.tmp
```

7CorrStack/CorridorStack.sh File Reference

27CorrStack/CorridorStack.sh7CorrStack/CorridorStack.sh

Create corridor stack.

Detailed Description

The final process for Zero Offset VSP processing is creating corridor stack, that representing the 1-D seismic response in the borehole. This is done by taking a short window around transit time, and stacks it.
The corridor window don't be too short, because it may not capture the amplitude variation, but also not too long, because we can include multiple in the stacking process.
For reflection below TD, we include number of traces (defined by except_last_trace) to be stack for all window.

Usage:

Open the file in text editor, and edit required parameters as necessary. The parameter description is given here. Run the command from console.

$./ApplyDecon.sh

*** Before run, please manually copy the SU file that you want to process to this directory

$cp ../5Separation/pef_up_enh.su.align .

$cp ../0ImportSEGY/tt-header.txt .

Output:

- Corridor Stack

Parameters:

input_decon_enh_up Input is final upgoing wavefield (after decon)

o*utput_cstack*	Corridor Stack Output
t*t*	ASCII file of Transit Time picks data

Definition in file **CorridorStack.sh**.

Author:

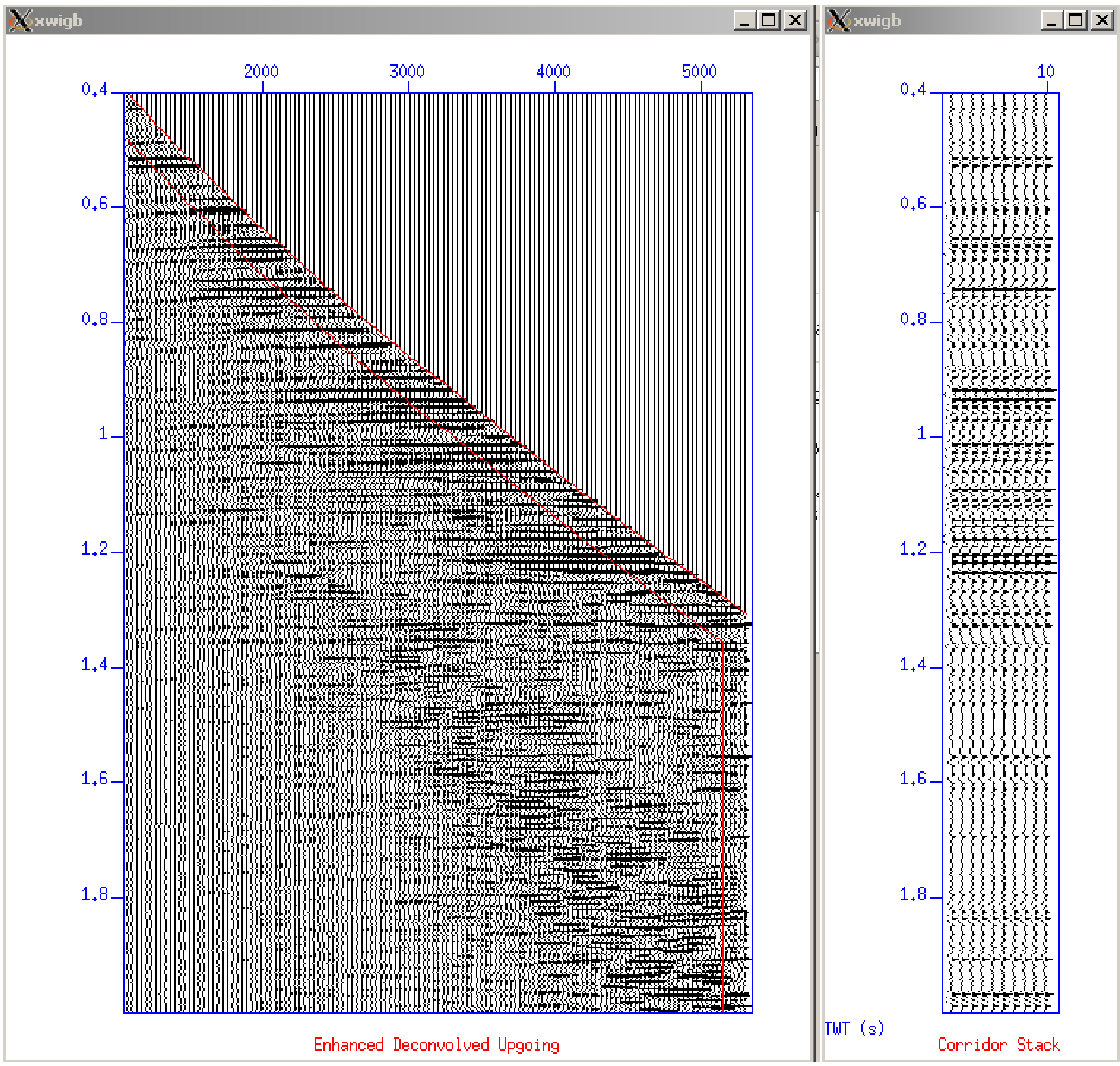
fahdi@gm2001.net

Date:

February 2012

Output Example:

$./CorridorStack.sh &



Upgoing wavefield and 1-D corridor stack repeated 12 times

Script Source:

Filename: ApplyDecon.sh

```
#!/bin/bash

#input
input_decon_enh_up=pef_up_enh.su.align
output_cstack=corr_stack.su
tt=tt-header.txt

#set parameter
corridor=0.08 #s
except_last_trace=5;

#housekeeping
nrec=($(wc -l $tt | awk '{print $1}')) #housekeeping
last_rec=$(($nrec - $except_last_trace))

awk -v corridor=$corridor -v last_rec=$last_rec \
'BEGIN {i=1}{if (i<=last_rec){print (($1*2)+corridor), $2} else {print 9,$2} i++}' $tt \
| awk '{ printf "%4f %d\n", $1, $2 }' > $tt.inside

awk '{ printf "%4f %d\n", $1*2, $2 }' $tt > $tt.outside

#apply enhancement to pef up
awk '{print $2}' $tt > xfile.tmp
a2b < xfile.tmp n1=$nrec> xfile.bin

awk '{print $1}' $tt.inside > tfile.inside.tmp
a2b < tfile.inside.tmp n1=$nrec> tfile.inside.bin

awk '{print $1}' $tt.outside > tfile.outside.tmp
a2b < tfile.outside.tmp n1=$nrec> tfile.outside.bin

```

```
#housekeeping-end

#display corridor window
suwind < $input_decon_enh_up tmin=0.4 tmax=2.0 | suxwigb perc=95 title="Enhanced Deconvolved Upgoing" key=gelev curve=$tt.inside,$tt.outside npair=$nrec,$nrec curvecolor=red &

#mute, change word, stack
sumute < $input_decon_enh_up key=gelev nmute=$nrec xfile=xfile.bin tfile=tfile.inside.bin mode=1 \
| sustack repeat=12 normpow=1.0 > $output_cstack
#display
suwind < $output_cstack tmin=0.4 tmax=2.0 \
| suxwigb perc=99 xbox=610 wbox=200 label1='TWT (s)' title='Corridor Stack'&

#clean up
rm *.tmp

```

Reference

Seismic Unix CWP

Ela2D

Acknowledgment

Thanks to Sunawar Kunaifi (sunawar.kunaifi@gmail.com) for interesting discussion SeismicUnix application for VSP.

Thanks to DCS Schlumberger Indonesia for help and support.