Seismic Processing
Prac 3 - Processing a real dataset.

ERTH3021

October 20, 2015

This is the last of three pracs on seismic data processing. In this prac we will complete the processing of the synthetic dataset from prac2. Once that is complete you will process a real dataset from scratch.

As with prac 1& 2 code templates have been provided. The prac can be downloaded from
https://github.com/stuliveshere/Seismic-Processing-Prac3

Introduction

At the end of last week we created a seismic section from the foybrook model synthetic dataset. The resulting stack was of reasonable quality, considering the tools available. However there are a number of processes missing which would normally be applied, including

- statics corrections
- wavelet deconvolution
- spatially variant velocity analysis
- spectral windowing
- spectral enhancement

Whilst statics corrections and wavelet deconvolution is outside the scope of this course, the remainder are not.

We are going to add 2 new processes to the work we did in prac 2. These processes are:

- spatially variant velocity analysis
- spectral windowing

Once you have had some experience with these processes, you will process a real seismic dataset from (almost) scratch.

Exercise 1: Spatially Varying Velocity Analysis

Why is performing velocity analysis on once CDP less than ideal? If the usual velocity analysis is performed every 100m (for shallow coal data) how many CDPS require velocity analysis?

Before we do velocity analysis/normal moveout correction we will need to initialise the dataset and perform a spherical divergence correction. Once that is done select a range of CDPS to perform velocity anaysis on.

The semblance analysis tool has been updated to make velocity anaysis easier. Use the new semblance tool to pick velocities for each CDP.

Build the velocity profile and display. Make any changes neccessary to improve the consistency of the model.

Stack the Foybrook model using the new velocity model. Compare to the stack generated last week with a single velocity point. Make note of any areas of improvement.

Exercise 2: Spectral Windowing

A seismic dataset contains many signal frequencies. Not all frequencies contain data. For example very low frequencies and very high frequencies generally do not contain reflection information. A frequency filter that cuts off low frequencies is referred to as a low-cut filter. A frequency filter that cuts off high frequencies is referred to as a high-cut filter. A frequency filter that cuts off both high and low is referred to as a bandpass filter.

A fairly sophisticated bandpass filter has been added to the toolbox. A spectral analysis tool, called an FX filter, has also been added. Load and display the stack from exercise 1 into the FX filter. Discuss the resulting image and the possible location of the data. Select a starting high and a low cut.

Apply the bandpass filter to the stack and compare to the stack with no filter. Make the bandpass filter wider and narrower, comparing the result each time. Select the best highcut and lowcut and apply to the data. This section is referred to as the "final stack" for the foybrook model. Compare with the stack generated in prac 2. Highlight areas of improvement.

DIY Seismic Processing

A dynamite survey has been supplied for you to process. There are some pros and cons with processing dynamite. Pros include high amplitude, high frequency signal. Cons include cost. Due to the cost, dynamite data is usually low fold. This data also has some significant statics. As such the data has been preprocessed to remove as much statics variation as possible. Deconvolution has also been applied to the dataset.

Exercises 3 - 9 contain the templates required to process the supplied dataset, 'al_dynamite.su'. Each student is expected to hand in a processing report and email their final stack to uqsflet4@uq.edu.au. The processing report should have the following format:

1. Introduction - one or two sentences explaining the data, including source type, source and reciever spacings, cdp spacings and fold.
2. Data Quality - discuss the data quality in a) shots, b) raw stack c) final stack d) spectral analysis
3. Processing
   - (a) Initialisation - quick description, screenshot of representitive shot record
   - (b) True Amplitude Recovery - describe TAR, screenshots of parameter testing (at least 3) and final parameter selected
   - (c) Velocity Analysis - Screencapture and description of a representive semblance plot. Discuss chosen CDP spacing. Screencapture of final velocity field.
   - (d) Stack - compare with stack generated with single velocity point. Highlight areas of improvement. Discuss other potential methods of velocity analysis.
   - (e) Bandpass Filter design - discuss the concept of spectral windowing. Show screencapture of FX design window. Discuss chosen highcut and lowcut thresholds. Apply thresholds, and compare to stack with no bandpass filter. Highlight areas of improvement. Vary the bandpass filter, and compare the results, with documentation. Select best thresholds.
   - (f) Velocity Scaling - discuss the aim of velocity scaling. Scale the velocities up and down and compare with the unscaled stack. Highlight areas of improvement. If neccessary, hand modify your velocity field. Document and select best velocity field/scaling.
   - (g) Trace Mix - discuss, test and apply pre-stack trace mix. Compare to stack without trace mix and highlight areas of improvement. Document testing and select best parameter.
   - (h) Final stack - generate final stack. include screencapture and table of key parameters. email stack to uqsflet4@uq.edu.au (it should be around 1.7Mb).

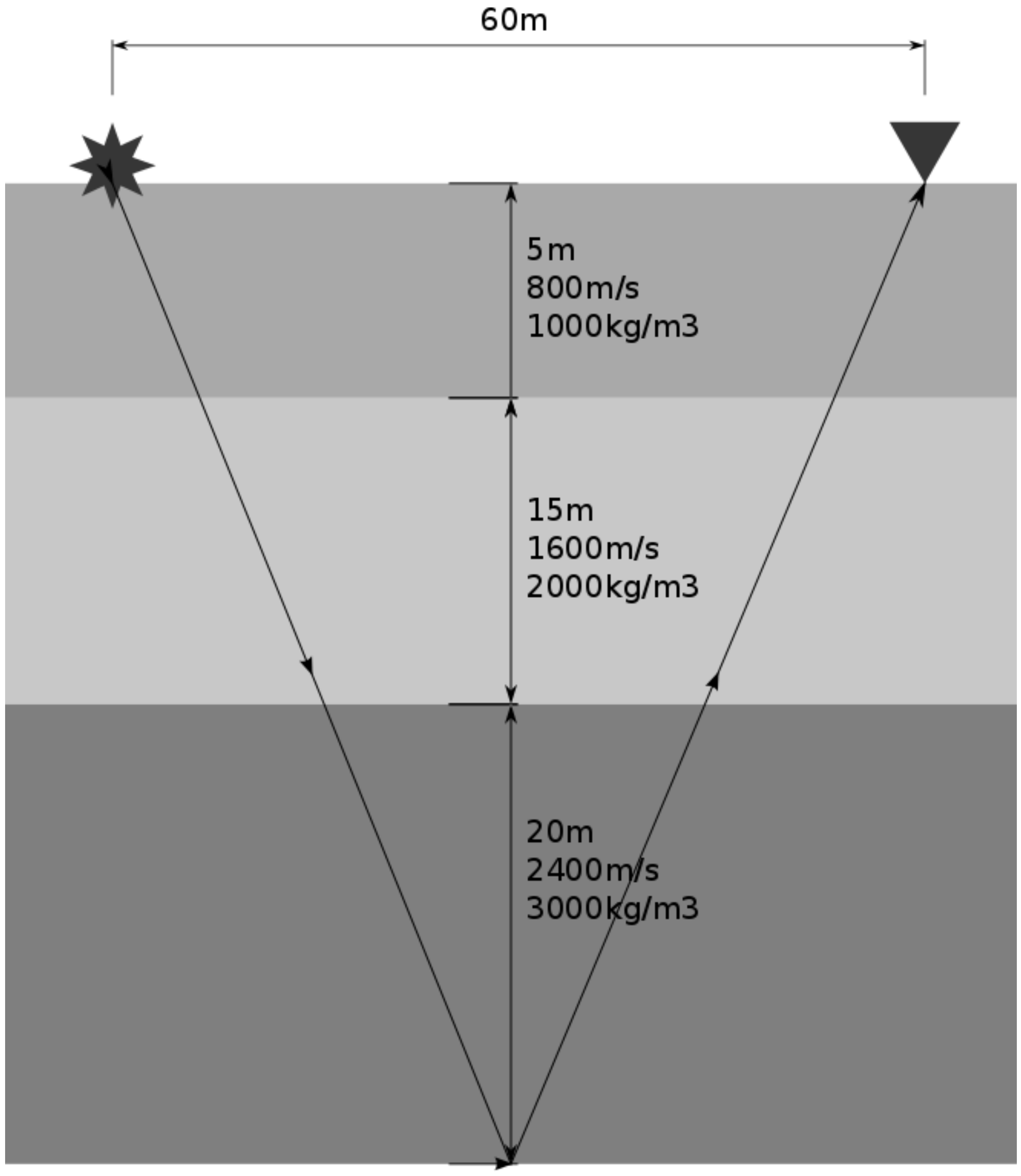

Figure 1: Seismic Model. The star is the source location and the triangle a geophone. This model assumes straight rays.

Revision

1 The Convolutional Model

1.1

Describe in words what convolution is, and how it differs from correlation. Give an example of where you might use a) convolution and b) correlation.

1.2

The convolutional model of the seismic trace states that the trace we record is the result of the earth's response convolved with the source wavelet and the recording system, with some additional noise. Show an equation which describes this, and label each term.

1.3

The earth's response consists of a range of different signals, including the direct wave, refracted wave and reflected wave. Modify the convolutional model equation to include these signals.

2 The Direct Wave

2.1

What is the direct wave?

2.2

Assuming a velocity of 400m/s, how long does it take the direct wave to travel from the source to receiver shown in Figure 1?

2.3

Attenuation of waves occurs for a number of reasons. One reason is that of spherical divergence. Describe the basic cause of amplitude loss due to spherical divergence.

2.4

Briefly discuss one other cause of wave attenuation in seismic data.

2.5

Assuming that the attenuation due to spherical divergence has a coefficient of two, and the source has unit amplitude, calculate the amplitude of the direct wave at the geophone in Figure 1.

3 The Refracted Wave

3.1

What is a refracted wave? What is one potential use of the refracted wave data?

3.2

What is the critical angle at the bottom of the first layer in the source-receiver geometry shown in Figure 1?

3.3

What is the travel time for the geometry shown in Figure 1?

4 The Reflected Wave

4.1

Calculate the travel time of the reflected wave for the model shown in Figure 1. Assume straight rays.

4.2

Discuss how you would analytically calculate the correct (non-straight) ray path.

4.3

Assuming a unit source amplitude, calculate the amplitude of the reflected wave at the geophone due to transmission and reflection coefficients. Assume zero offset coefficients.

Discuss when it might and might not be appropriate to use the zero offset assumption.

5 Seismic Gathers

5.1

What is the difference between a shot gather, a receiver gather and CMP gather? Draw a rough diagram of each.

5.2

What are CMP gathers useful for? Why?

5.3

What assumptions are made when CMP stacking? What happens when these assumptions break down?

5.4

What is fold?

6 Amplitude Recovery

6.1

Why is amplitude recovery used?

6.2

Why is using an AGC for amplitude recovery not ideal?

7 Normal Moveout

7.1

Draw a rough diagram showing why normal moveout is hyperbolic.

7.2

Calculate the zero offset travel-time for the reflection event in question 4.

7.3

Wavelet stretching is caused by the non-linear NMO correction. Discuss the impact of stretch and a potential mitigation strategy.

8 Stacking

8.1

What is stacking?

8.2

What is the signal to noise ratio improvement for a CMP gather with a fold of n?

9 Noise attenuation

9.1

The refracted wave calculated in question 3 is often considered noise. What is one method that can be used to remove the refracted wave? Discuss the steps required in this method.

9.2

Often we want to enhance coherent events whilst attenuating random events. Discuss one method for achieving this. Discuss the potential negatives associated with this method.

10 Spectral Analysis

Briefly discuss what a fourier transform does. Draw a rough diagram of the spectral analysis of two sine waves that have been mixed together. On the spectral diagram, draw a filter that could be used to separate the two signals. Discuss how this might be useful for seismic data. Discuss the spectral content of the 3 common seismic waves- ground roll, refractor and reflector.

11 Processing Flows

11.1

Suggest a simple processing flow for a seismic dataset, using the processes discussed from question 5 to question 9. State why you chose that particular processing sequence. State any key parameters which need to be tested.

12 Useful Formulas

$$R = \frac{1}{distance^2}$$

$$T = \frac{X}{V_1} + \frac{2z\cos i_c}{V_0}, \quad i_c = \sin^{-1}\frac{V_0}{V_1}$$

Where

- X = lateral distance
- V_0 = velocity of weathering layer
- V_1 = velocity of sub-weathering layer
- z = thickness of weathering layer
- i_c = critical angle

$$R_r = \frac{z_1 - z_0}{z_1 + z_0}$$

$$R_t = \frac{2 * z_0}{z_1 + z_0}$$

where

- z_0 = acoustic contrast in layer 0, i.e. $\rho_0 v_0$
- z_1 = acoustic contrast in layer 1, i.e. $\rho_1 v_1$

$${t_x}^2 = {t_0}^2 + \frac{x^2}{v^2}$$

where

- t_x is the travel time at offset x
- t_0 is the travel-time at offset 0
- x is the offset
- v is the velocity

$$x_{rms} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}$$

where

- x_n is the sample number
- n is the number of samples

```
#last time we did a very simplistic velocity analysis.
#this time we want to do a few more locations

import toolbox
import numpy as np
import pylab

#-------------------------------------------------------
#       useful functions
#-------------------------------------------------------

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('foybrook.su')

        #apply TAR
        print "applying true amplitude recovery"
        params['gamma'] = 3
        toolbox.tar(workspace, None, **params)

        #lets see how many cdps there are
        print  np.unique(workspace['cdp'])[25::45].tolist()
        params['smoother'] = 5

        #copy your list of cdps here... it will make it easier later
        cdps = [219, 264, 309, 354, 399, 444, 489, 534, 579]

        #~ params['velocities'] = np.arange(2000,6000,50)

        #~ for cdp in cdps:
                #~ gather = workspace[workspace['cdp'] == cdp]
                #~ toolbox.agc(gather, None, **params)
                #~ toolbox.semb(gather, **params)

        vels = {}
        vels[219] =  (0.03, 2350.40) , (0.13, 2615.04) , (0.46, 3232.54) , (1.13, 4202.88) , (1.54,
    4545.94)
        vels[264] =  (0.05, 2840.48) , (0.10, 3203.13) , (0.15, 3918.64) , (0.56, 4398.91) , (1.14,
    4888.99)
        vels[309] =  (0.07, 2918.89) , (0.28, 3634.40) , (0.76, 4085.27) , (1.47, 4565.54)
        vels[354] =  (0.09, 2997.30) , (0.54, 3820.63) , (0.65, 3987.25) , (0.88, 4330.30) , (1.32,
    4692.96)
        vels[399] =  (0.07, 2938.49) , (0.31, 3516.78) , (0.71, 4389.11) , (1.38, 5055.61)
        vels[444] =  (0.07, 2869.88) , (0.26, 3242.34) , (0.45, 3712.81) , (0.75, 4006.85) , (1.11,
    5222.24)
        vels[489] =  (0.05, 2781.67) , (0.23, 3585.39) , (0.47, 4379.31) , (0.90, 4938.00) , (1.47,
    5437.87)
        vels[534] =  (0.05, 2673.85) , (0.14, 3212.93) , (0.24, 4193.08) , (0.76, 4741.97) , (1.22,
    5134.03)
        vels[579] =  (0.08, 2673.85) , (0.19, 3320.75) , (0.40, 4036.26) , (1.06, 4781.17)

        #build our 2D  velocity map
        print "building velocities"
        params['vels'] = toolbox.build_vels(vels, **params)

        #~ #view it
        pylab.imshow(params['vels'].T, aspect='auto')
        pylab.colorbar()
        pylab.show()

        #~ #Now we have a better velocity profile, we can use a better nmo to move it out
        #~ print "applying nmo correction"
        #~ workspace = toolbox.co_nmo(workspace, None, **params)

        #~ #apply AGC
        #~ print "applying AGC"
        #~ toolbox.agc(workspace, None, **params)

        #~ #trace mix
        #~ print "applying trace mix"
        #~ params['mix'] = 5
        #~ toolbox.trace_mix(workspace, None, **params)
        #~ #stack
        #~ print "stacking"
        #~ section = toolbox.stack(workspace, None, **params)

        #~ #display
        #~ #turn off gather sort
        #~ params['primary'] = None
        #~ print "displaying"
        #~ toolbox.display(section, None, **params)
        #~ toolbox.cp(section, 'stack.su', **params)
        #~ pylab.show()
```

../exercise1.py

```
#last time we did a very simplistic velocity analysis.
#this time we want to do a few more locations

import toolbox
import numpy as np
import pylab
pylab.rcParams['image.interpolation'] = 'sinc'

#—————————————————————————————————————————————
#       useful functions
#—————————————————————————————————————————————

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('stack.su')

        #turn off gather sort
        params['primary'] = None
        params['clip'] = 0.2
        #display check
        toolbox.display(workspace, None, **params)

        #do an fx spectrum, to get an idea of the frequency content
        #~ toolbox.fx(workspace, None, **params)

        #set bandpass
        params['lowcut'] = 20
        params['highcut'] = 100

        toolbox.bandpass(workspace, None, **params)
        toolbox.display(workspace, None, **params)

        pylab.show()
```

../exercise2.py

```
#processing a real dataset
#step 1 - import dataset and check the gathers

import toolbox
import numpy as np
import pylab

if __name__ == "__main__":
        #import dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('al_dynamite.su')

        #set gather order to shot gather
        params['primary'] = 'sx'
        params['secondary'] = 'gx'

        #display
        toolbox.display(workspace, None, **params)
        pylab.show()
```

../exercise3.py

```
#spherical divergence correction

import toolbox
import numpy as np
import pylab

#-------------------------------------------------------------------
#       useful functions
#-------------------------------------------------------------------

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('foybrook.su')

        #find our test CDP
        #~ print np.unique(workspace['cdp'])

        #extract it
        cdp = workspace[workspace['cdp'] == 396]

        #display it
        #~ toolbox.display(cdp, None, **params)

        params['gamma'] = 6
        toolbox.tar(cdp, None, **params)

        #display it
        toolbox.display(cdp, None, **params)
        pylab.show()
```

../exercise4.py

```
#pick around 10 cdp locations and do a velocity analysis

import toolbox
import numpy as np
import pylab

#-------------------------------------------------------------------
#       useful functions
#-------------------------------------------------------------------

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('al_dynamite.su')

        #lets see how many cdps there are
        #~ print  np.unique(workspace['cdp'])[25::45].tolist()

        #store the cdps for reference
        cdps =[225, 270, 315, 360, 405, 450, 495, 540, 585]

        #define velocities
        params['velocities'] = np.arange(1000,6000,50)
        params['smoother'] = 5

        #iterate over list of cdps
        #~ for cdp in cdps:
                #~ params['smute'] = 30
                #~ inds = (workspace['cdp'] > cdp -2) * (workspace['cdp'] < cdp +2)
                #~ gather = workspace[inds]
                #~ toolbox.agc(gather, None, **params)
                #~ params['highcut'] = 120
                #~ params['lowcut'] = 30
                #~ toolbox.bandpass(gather, None, **params)
                #~ toolbox.semb(gather, **params)

        #set up vel dictionary for storing values
        vels = {}
        vels[225] =  (0.06, 1537.38) , (0.28, 2876.21) , (0.87, 4608.10)
        vels[270] =  (0.05, 1525.09) , (0.18, 2483.16) , (0.36, 3171.00) , (0.66, 4079.93) , (0.98,
    4816.90)
        vels[315] =  (0.04, 1365.42) , (0.14, 2728.82) , (0.22, 3134.15) , (0.57, 4116.78) , (0.74,
    4571.25) , (0.97, 5013.43)
        vels[360] =  (0.04, 1697.05) , (0.10, 2520.01) , (0.21, 2937.62) , (0.43, 3244.70) , (0.64,
    3981.67) , (0.98, 4239.61)
        vels[405] =  (0.06, 1439.11) , (0.27, 2753.38) , (0.49, 3957.10) , (0.97, 5381.92)
        vels[450] =  (0.06, 1340.85) , (0.41, 2741.10) , (0.52, 3625.47) , (0.02, 1144.32) , (0.29,
    3060.45) , (0.54, 3711.45) , (0.97, 4313.31)
        vels[495] =  (0.04, 1611.07) , (0.11, 3072.74) , (0.23, 3318.39) , (0.35, 3772.86) , (0.48,
    3981.67) , (0.94, 5099.41)
        vels[539] =  (0.04, 2028.69) , (0.11, 3072.74) , (0.32, 3883.41) , (0.51, 4485.27) , (0.96,
    5222.24)
        vels[584] =  (0.06, 1623.36) , (0.20, 2495.44) , (0.32, 3121.87) , (0.95, 4411.57)

        #build vels
        vels = toolbox.build_vels(vels, **params)

        #display vels
        pylab.imshow(vels.T, aspect='auto')
        pylab.colorbar()
        pylab.show()
```

../exercise5.py

```
#stack up your velocities


import toolbox
import numpy as np
import pylab

#-------------------------------------------------------
#       useful functions
#------------------------------------------------------

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('al_dynamite.su')

        #apply tar
        params['gamma'] = 5
        toolbox.tar(workspace, None, **params)


        #copy vels from previous exercise
        vels = {}
        vels[225] =  (0.06, 1537.38) , (0.28, 2876.21) , (0.87, 4608.10)
        vels[270] =  (0.05, 1525.09) , (0.18, 2483.16) , (0.36, 3171.00) , (0.66, 4079.93) , (0.98,
    4816.90)
        vels[315] =  (0.04, 1365.42) , (0.14, 2728.82) , (0.22, 3134.15) , (0.57, 4116.78) , (0.74,
    4571.25) , (0.97, 5013.43)
        vels[360] =  (0.04, 1697.05) , (0.10, 2520.01) , (0.21, 2937.62) , (0.43, 3244.70) , (0.64,
    3981.67) , (0.98, 4239.61)
        vels[405] =  (0.06, 1439.11) , (0.27, 2753.38) , (0.49, 3957.10) , (0.97, 5381.92)
        vels[450] =  (0.06, 1340.85) , (0.41, 2741.10) , (0.52, 3625.47) , (0.02, 1144.32) , (0.29,
    3060.45) , (0.54, 3711.45) , (0.97, 4313.31)
        vels[495] =  (0.04, 1611.07) , (0.11, 3072.74) , (0.23, 3318.39) , (0.35, 3772.86) , (0.48,
    3981.67) , (0.94, 5099.41)
        vels[539] =  (0.04, 2028.69) , (0.11, 3072.74) , (0.32, 3883.41) , (0.51, 4485.27) , (0.96,
    5222.24)
        vels[584] =  (0.06, 1623.36) , (0.20, 2495.44) , (0.32, 3121.87) , (0.95, 4411.57)

        #build vels
        vels = toolbox.build_vels(vels, **params)

        params['primary'] = None
        params['smute'] = 30

        params['vels'] = vels
        v100 = toolbox.co_nmo(workspace, None, **params)
        toolbox.agc(v100, None, **params)
        section100 = toolbox.stack(v100, None, **params)
        toolbox.cp(section100, 'stack100.su', None)
        toolbox.display(section100, None, **params)

        pylab.show()
```

../exercise6.py

```
#spectral analysis, bandpass filters
#test a few filters to find the best

import toolbox
import numpy as np
import pylab

#-----------------------------------------------------------------
#       useful functions
#-----------------------------------------------------------------

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('stack100.su')
        params['primary'] = None

        #basic spectral analysis
        #~ toolbox.fx(workspace, None, **params)

        params['highcut'] = 100
        params['lowcut'] = 30

        toolbox.bandpass(workspace, None, **params)
        toolbox.display(workspace, None, **params)

        pylab.show()
```

../exercise7.py

```
#try varying your velocities
#pick your best scalar

import toolbox
import numpy as np
import pylab

#-----------------------------------------------------------------
#       useful functions
#-----------------------------------------------------------------

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('al_dynamite.su')

        #apply tar
        params['gamma'] = 5
        toolbox.tar(workspace, None, **params)


        #copy vels from previous exercise
        vels = {}
        vels[225] =  (0.06, 1537.38) , (0.28, 2876.21) , (0.87, 4608.10)
        vels[270] =  (0.05, 1525.09) , (0.18, 2483.16) , (0.36, 3171.00) , (0.66, 4079.93) , (0.98,
    4816.90)
        vels[315] =  (0.04, 1365.42) , (0.14, 2728.82) , (0.22, 3134.15) , (0.57, 4116.78) , (0.74,
    4571.25) , (0.97, 5013.43)
        vels[360] =  (0.04, 1697.05) , (0.10, 2520.01) , (0.21, 2937.62) , (0.43, 3244.70) , (0.64,
    3981.67) , (0.98, 4239.61)
        vels[405] =  (0.06, 1439.11) , (0.27, 2753.38) , (0.49, 3957.10) , (0.97, 5381.92)
        vels[450] =  (0.06, 1340.85) , (0.41, 2741.10) , (0.52, 3625.47) , (0.02, 1144.32) , (0.29,
    3060.45) , (0.54, 3711.45) , (0.97, 4313.31)
```

```
        vels[495] =  (0.04, 1611.07) , (0.11, 3072.74) , (0.23, 3318.39) , (0.35, 3772.86) , (0.48, 3981.67) , (0.94, 5099.41)
        vels[539] =  (0.04, 2028.69) , (0.11, 3072.74) , (0.32, 3883.41) , (0.51, 4485.27) , (0.96, 5222.24)
        vels[584] =  (0.06, 1623.36) , (0.20, 2495.44) , (0.32, 3121.87) , (0.95, 4411.57)

        #build vels
        vels = toolbox.build_vels(vels, **params)

        params['primary'] = None
        params['highcut'] = 100
        params['lowcut'] = 30
        params['smute'] = 30



        params['vels'] = vels
        v100 = toolbox.co_nmo(workspace, None, **params)
        toolbox.agc(v100, None, **params)
        section100 = toolbox.stack(v100, None, **params)
        toolbox.bandpass(section100, None, **params)
        toolbox.display(section100, None, **params)


        params['vels'] = vels * .9
        v90 = toolbox.co_nmo(workspace, None, **params)
        toolbox.agc(v90, None, **params)
        section90 = toolbox.stack(v90, None, **params)
        toolbox.bandpass(section90, None, **params)
        toolbox.display(section90, None, **params)

        params['vels'] = vels *1.1
        v110 = toolbox.co_nmo(workspace, None, **params)
        toolbox.agc(v110, None, **params)
        section110 = toolbox.stack(v110, None, **params)
        toolbox.bandpass(section110, None, **params)
        toolbox.display(section110, None, **params)

        pylab.show()
```

../exercise8.py

```
#create final stack
#test stretch mute
#test trace mix
#email final stack to mail@stuliveshere.com

import toolbox
import numpy as np
import pylab

#-------------------------------------------------
#       useful functions
#-------------------------------------------------

None

if __name__ == "__main__":
        #initialise dataset
        print "initialising dataset"
        workspace, params = toolbox.initialise('al_dynamite.su')
        params['primary'] = None

        #apply tar
        params['gamma'] = 5
        toolbox.tar(workspace, None, **params)


        #copy vels from previous exercise
        vels = {}
        vels[225] =  (0.06, 1537.38) , (0.28, 2876.21) , (0.87, 4608.10)
        vels[270] =  (0.05, 1525.09) , (0.18, 2483.16) , (0.36, 3171.00) , (0.66, 4079.93) , (0.98, 4816.90)
        vels[315] =  (0.04, 1365.42) , (0.14, 2728.82) , (0.22, 3134.15) , (0.57, 4116.78) , (0.74, 4571.25) , (0.97, 5013.43)
        vels[360] =  (0.04, 1697.05) , (0.10, 2520.01) , (0.21, 2937.62) , (0.43, 3244.70) , (0.64,
    3981.67) , (0.98, 4239.61)
        vels[405] =  (0.06, 1439.11) , (0.27, 2753.38) , (0.49, 3957.10) , (0.97, 5381.92)
        vels[450] =  (0.06, 1340.85) , (0.41, 2741.10) , (0.52, 3625.47) , (0.02, 1144.32) , (0.29,
    3060.45) , (0.54, 3711.45) , (0.97, 4313.31)
        vels[495] =  (0.04, 1611.07) , (0.11, 3072.74) , (0.23, 3318.39) , (0.35, 3772.86) , (0.48,
    3981.67) , (0.94, 5099.41)
        vels[539] =  (0.04, 2028.69) , (0.11, 3072.74) , (0.32, 3883.41) , (0.51, 4485.27) , (0.96,
    5222.24)
        vels[584] =  (0.06, 1623.36) , (0.20, 2495.44) , (0.32, 3121.87) , (0.95, 4411.57)

        #build vels
        vels = toolbox.build_vels(vels, **params)

        #
        params['vels'] = vels
        params['smute'] = 30

        #normal moveout

        v100 = toolbox.co_nmo(workspace, None, **params)

        #agc
        toolbox.agc(v100, None, **params)

        #trace mix
        params['mix'] = 5
        toolbox.trace_mix(workspace, None, **params)

        #stack
        section100 = toolbox.stack(v100, None, **params)

        #bandpass filter
        params['lowcut'] = 30
        params['highcut'] = 100
        toolbox.bandpass(section100, None, **params)
        toolbox.cp(section100, 'final_stack.su', **params)
        #display
        toolbox.display(section100, None, **params)

        pylab.show()
```

../exercise9.py