Seismic Processing
Prac 2 - Processing a synthetic dataset.

ERTH3021

October 12, 2014

This is the second of two pracs on seismic data processing. In this prac we will be processing the dataset generated in prac 1.

If you do not have the dataset from prac 1 is it available for download at insert link here

As with prac 1 code templates have been provided. The prac can be downloaded from insert link here

1 Introduction

Seismic data processing involves taking raw seismic data and generating an image which can be related to the geology targetted by the seismic survey. Seismic processing usualling involves a number of steps, including

- velocity analysis
- geometric corrections (normal moveout, statics, migration)
- amplitude corrections (amplitude recovery, AGC)
- noise attenuation
- CMP stacking
- deconvolution

Although not neccessarily in that order.

For this prac we will use the following processing sequence.

1. load and QC (quality control) data
2. sort into CDP gathers
3. amplitude recovery
4. move out with constant velocity
5. field stack (aka brute stack)
6. first velocity analysis
7. first velocity stack
8. pre-stack noise attenuation
9. second velocity analysis
10. second velocity stack
11. post-stack noise attenuation

This is somewhat like a processing sequence used in a commercial processing house. Note the iterative nature of the sequence - this is a common trait of seismic processing.

image quality is influenced by the accuracy of the velocities and the parameters selected. inaccurate velocities can lead to imperfect moveout which can affect stack quality. poor selection of processing parameters can harm the data - for example a harsh noise attenuation routine might also attenuate the signal as well as the noise. finding optimum parameters usually involves significant amounts of testing.

In this prac you will be picking your own velocities and selecting your own processing parameters. You will be expected to test a range of parameters and discuss why you chose those parameters in your report.

Exercise 1: Load data and perform QC

In this exercise we will create an initialisation routine which will load our dataset into numpy, and set up some initial parameters. We will then have a look at some raw shot gathers to ensure the data is loaded correctly (QC).

Exercise 2: Sort into CMPs

CMP gathers collect all traces which pass through the same CMP into a single gather. CMP gathers are generally used for velocity analysis and stacking. The number of traces in a gather is refered to as the fold.

Sort the dataset into CMP gathers and perform QC. Compare the CMP gathers to the shot gathers from exercise 1 and discuss why they look different.

Extract one CDP gather for testing.

Exercise 3: Normal moveout correction

Notice in the raw CMP gathers the reflectors have a hyperbolic shape. In order to stack a CMP the reflectors need to be flat. The reason for the hyperbolic shape is normal moveout, and flattening reflectors is generally refered to as normal moveout correction. For a flat, horizontal reflector, the traveltime equation is

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

where

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

Write a python function which implements the normal moveout correction. That is, re-arrange the above equation so that we can calculate the zero offset traveltime t_0.

The function which applies your move out correction to the dataset has been supplied. Using your test CMP gather, try to find a single velocity (aka constant velocity) which flattens the reflectors the most. Show that gather before and after moveout correction. Discuss the issues associated with normal moveout with a constant velocity.

Note the code relating to "stretch". The amount of moveout correction is not linear, and thus shallow data tends to get stretched. Experiement with stretch values on your test CMP gather. Show a gather with and without stretch. Discuss the potential impact of stretched data on a stack.

Exercise 4: stacking

Stacking involves taking all of the traces in a single CMP gather and summing together into once trace. The amplitude of the trace is then normalised by the fold of the gather. Thus stacking can also be considered as finding the average trace at a given CMP location. When all CMP gathers are stacked, the resulting image has one trace at each CMP location. This is generally refered to as a stack or a stacked section.

Write a function which will stack any gather given to it. Note this can be done in 1 line. Hint: $mean = \bar{x} = \frac{x_1+x_2+...+x_n}{n}$

Exercise 5: Amplitude Recovery

Loss of amplitude due to spherical divergence and reflection/transmission coefficients are only part of the reason deeper signals are weaker. Spherical divergence and reflection/tranmission coefficients are often refered to as ”elastic”, because they conserve energy as a wave. But there are also anelastic effects, including fluidic movement and grain boundary friction which results in seismic energy being converted to heat.

We have already seen in the raw gathers that deeper signals are weaker and much more difficult to see. In the last prac we compensated for this using an AGC. An AGC normalises the amplitude in a sliding window over every trace, based upon the amplitudes in the window. Sometimes in seismic processing we use an AGC prior to stack, but sometimes we want to preserve the relative amplitude of samples between traces. We call this amplitude recovery.

In theory we could correct for spherical divergence, but it is much more challenging to correct for the remaining sources of attenuation. Thus a generic, tunable formula is generally, where testing is used to fine tune the result until it ”looks right”. One such formula is

$$R = \exp^{\gamma t}$$

,
where

- R is the amplitude correction
- γ is the tunable parameter
- t time

Write a python function which implements this formula, and apply to the pre-stack data. Test various values of $/gamma$ such that the amplitude of the bottom reflector in the stack is closer to the top reflectors. Show the stack before and after amplitude recovery.