Application of Digital Wiener Filtering in Seismic Data Processing

Pulak Biswas ***(18MC0014)***, **Payel Hazra** ***(18MC0023)***,
Sandip Kumar Rana ***(18MC0035)***, **Paran Sonowal** ***(18MC0004)***
Brajeswar Ghosh ***(18MC0045)***, **Saheli Banerjee** ***(18MC0036)***

Department of Applied Geophysics, Indian Institute of Technology (ISM) Dhanbad

Geophysical Signal Processing Project
3 Yr. M.Sc (Tech) AGP, 4th sem

Abstract: Seismic signals are usually transient waveforms radiated from a localized natural or manmade seismic source. They can be used to locate the source, to analyze source processes, and to study the structure of the medium of propagation. In contrast, the term "seismic noise" designates undesired components of ground motion that do not fit in our conceptual model of the signal under investigation. What we identify and treat as seismic noise depends on the available data, on the aim of our study and on the method of analysis. The conventional linear frequency filters like band-pass or low-pass filters are not sufficient if both seismic signal and noise possess the same frequency ranges. The Wiener filter is the optimal linear processing technique for minimizing, in the statistical sense, the mean square error value (MMSE) between the observed signal and original signal. Wiener filtering technique is one of the filtering techniques which filter a noise-corrupted signal by linear time invariant filtering and it can be used either in single-channel or multichannel processing. Spiking deconvolution, an important application of Wiener optimum filtering processes can be used to improve the seismic trace vertical resolution that might be lost due to previous linear filtering processes. On the other hand, predictive deconvolution, another application of Wiener filter, suppresses multiples or ghost noise types and optionally alters the spectrum of the input data to increase resolution. Wiener filter is an efficient algorithm which uses the concept of correlation between the noise and the noise corrupted data and predicts a system function such that we can get an estimate of the original signal.

Index Terms: Spiking deconvolution, predictive deconvolution, inverse filtering, spectral analysis.

Abbreviations

- AGC: Automatic Gain Control
- SNR: Signal to Noise Ratio
- LTI: Linear Time Invariant
- FIR: Finite Impulse Response
- PSD: Power Spectral Density
- OL: Operator Length

1. Introduction

In reflection seismology, the spectrum of noise may cover the same frequency range as the seismic signals. i.e., both the seismic signal and unwanted energy have similar frequency content. The use of conventional linear frequency filters like band-pass or low-pass filters may not be the best choice. Such filters may extract frequency components of large SNR when compared to other frequency ranges. On the other hand, the SNR may differ as well for different frequencies, when seismic and noise frequency ranges are similar. Furthermore, when using low-pass or band-pass filters to attenuate unwanted seismic energy from the seismic data records, the seismic vertical resolution will be affected. i.e., the high frequency components of seismic waves are attenuated. Therefore one needs to obtain a filter that does the best job of converting an input into a desired

output. Optimum Wiener filters, are considered to be that type of filters. They can be used to separate seismic signals and noise when, for example, both possess the same frequency ranges.

2. Concepts

During world war II, Norbert Wiener of MIT, one of the foremost mathematicians of this institute, was asked to develop a technique for extracting radar signal from noise. In formulating his filtering technique, Wiener assumed that an information trace, e.g., in a seismogram, consists of a desired signal immersed in noise. Criteria were developed for deriving a filter operator that when convolved with the recorded trace would yield an output as close as possible to the desired signal.

The Wiener filter problem has solutions for three possible cases: one where a noncausal filter is acceptable, the case where a causal filter is desired, and the finite impulse response case where only input data is used. In signal processing, the Wiener filter is, FIR type, used to produce an estimate of a desired or target random process by LTI filtering of an observed noisy process. This requires the following assumptions:

1) The noise is additive,
2) Signal and noise are stationary linear stochastic processes,
3) The seismic signal and noise must differ in some way, such as differences in their frequency or wavenumber spectra, velocity, and so on,
4) Spectral characteristics of signal and noise are known,
5) Reflectivity function is totally unpredictable i.e., it is white in nature.

Assumption 5 actually tells that our knowledge of the amplitudes and traveltimes of the first k reflections does not permit us to make any deterministic statement about the amplitude and traveltime of the (k + 1)th reflection. This also means that the locations of reflectors are uncorrelated.

2.1. Optimum Wiener Filter

Fig. 1 shows the model of Wiener optimum filter denoted by h(t). In order to derive the coefficients of h(t), we consider the signal x(t) being fed to a Wiener filter of length N and with coefficients (h_0, h_1, h_2, ... , h_N). The output of the filter is denoted y(t) which is given by the convolution of its input x(t) with coefficients h(t). That is,

$$y(t) = x(t) * h(t). \tag{1}$$

The residual error is denoted e(t) and is defined as e(t) = d(t) - y(t). Then the mean square error is given by:

$$E = \sum_t [d(t) - y(t)]^2 = \sum_t [d(t) - h(t) * x(t)]^2 \tag{2}$$

Wiener optimum filtering involves designing a filter h(t) so that the error E in Eq. 2. between the desired output d(t) and the actual output y(t) is minimum.

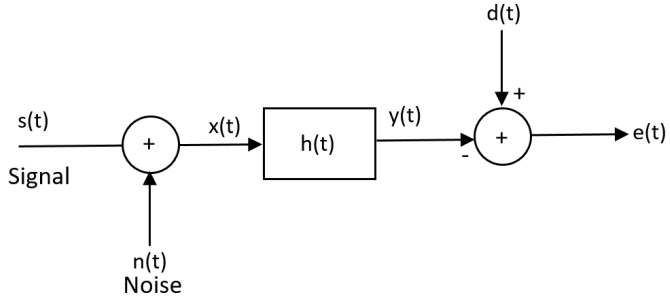


Fig. 1: The Wiener optimum filter model

This is the problem of least-squares and to attain the minimum error the partial derivative of E with respect to h(j) is set to zero:

$$\frac{\partial E}{\partial h(j)} = 0 \tag{3}$$

for j = 0, 1, 2, . . ., (N - 1). This set N of equations are called as normal eqations as the error is calculated normal to y(t). Applying Eq. 3. on 2. and simplifying, the result can be expressed as:

$$\sum_{t=0}^{N-1} r(j-t)h(t) = g(j) \tag{4}$$

for j = 0, 1, 2, . . ., (N - 1). This eqation can be explicitly represented as a matrix equation,

$$\begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{n-1} \\ r_1 & r_0 & r_1 & \cdots & r_{n-2} \\ r_2 & r_1 & r_0 & \cdots & r_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{n-2} & r_{n-3} & \cdots & r_0 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{N-1} \end{pmatrix} = \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{N-1} \end{pmatrix} \tag{5}$$

Here r_{j-i}, a_i and g_i, for i = 0,1,2,. . ., N-1 are the autocorrelation lags of the input wavelet, the Wiener filter coefficients, and the crosscorrelation lags of the desired output with the input wavelet, respectively.

In Eq. 5. g(j) is the j^{th} term of the crosscorrelation between d(n) and x(n):

$$g(j) = d(j) \star x(j) \tag{6}$$

for j = 0, 1, 2, . . . , (N-1).
r_{j-i} is the j^{th} term of the autocorrelation of x(t):

$$r(j) = x(j) \star x(j) \tag{7}$$

for j = 0, 1, 2, . . . , (N-1). The autocorrelation r(i) matrix of Eq. 5 is called a Toeplitz matrix and the set of N equations in Eq. 5. are known as the Wiener-Hopf equations. The solution for this equation gives the optimum filter coefficients.

The equation developed by Wiener and Hopf provides a convenient basis for determining the filter operators. This equation makes it possible to determine the filter coefficients from the autocorrelation function of the input signal and the cross-correlation function between the desired output and the input signal. The output function obtained by convolving the input with the optimum filter operator will not coincide exactly with the desired output, but the differences can be minimized in a least-squares sense.

Wiener filters can be used either in single-channel or multichannel processing. This filter applies to a large class of problems in which any desired output can be considered. Five choices for the desired output are:

- Type 1: Zero-lag spike,
- Type 2: Spike at arbitrary lag,
- Type 3: Time-advanced form of input series,
- Type 4: Zero-phase wavelet,
- Type 5: Any desired arbitrary shape.

The process with zero-lag spike as desired output is called spiking deconvolution while a time-advanced form of the input series, suggests a prediction process.

2.1.1. Spiking Deconvolution

The seismic trace vertical resolution might be lost due to previous linear filtering processes, which can be improved in order to better view and recognize important subsurface structures such as thin layers and faults using spiking deconvolution, an important application of Wiener optimum filtering processes.

The Wiener filter converts the seismic wavelet into any desired output. When the desired output is a zero-lag spike the process of filtering is called as spiking deconvolution. Crosscorrelation of the desired spike (1,0,0,.. .,0) with input wavelet $(x_0, x_1, x_2, ..., x_{n-1})$ yields the series $(x_0, 0, 0, ..., 0)$. The generalized form of the normal Eq. 5. takes the special form:

$$\begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{n-1} \\ r_1 & r_0 & r_1 & \cdots & r_{n-2} \\ r_2 & r_1 & r_0 & \cdots & r_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{n-2} & r_{n-3} & \cdots & r_0 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{n-1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{8}$$

Eq. 8. is scaled by $1/x_0$. To perform spiking deconvolution in practice, we need to set the following parameters:

- Autocorrelation window (w): This sets up the part of seismic trace from which we will select the elements of the autocorrelation matrix in the normal equations.
- Filter length (N): This sets up the length of the spiking filter h(t).
- Percent prewhitening (ε): This sets up the amount of white random noise we want to include into our autocorrelation matrix to stabilize the solution of the normal equations.

2.1.1.a. Autocorrelation Window

The autocorrelation of the seismic trace largely affects the choice of deconvolution parameters. Therefore, it is important to choose a suitable autocorrelation window that will be used to calculate the deconvolution parameters. The autocorrelation window should include the part of the record that contains useful reflection signal and should exclude coherent (e.g., ground roll) or incoherent noise (e.g., later parts of the record).The length of the autocorrelation window should be greater than eight times the largest operator length that will be used for that data set.

2.1.1.b. Filter Length

The filter length should be equal to the wavelet length. The first transient zone of the trace autocorrelation is the part that mostly represents the autocorrelation of the source wavelet.The first transient zone is the first part of the autocorrelation that contains high amplitudes.The operator length should be selected so that it is approximately equal to the length of the first transient zone of the trace autocorrelation. The optimum operator length should not leave considerable amount of energy in the trace autocorrelogram.

2.1.1.c. Prewhitening

The amplitude spectrum of the spiking deconvolution filter is the reciprocal of that of the source wavelet. If the amplitude spectrum of the source wavelet is zero at some frequencies, then the amplitude spectrum of inverse filter will be unstable (i. e., it will be infinite at these frequencies). A similar numerical instability might be encountered when inverting the trace autocorrelation matrix if the determinant of the autocorrelation matrix is zero. To ensure numerical stability, we introduce an artificial level of white random noise into the trace amplitude spectrum and autocorrelation before deconvolution. This process is called prewhitening.

The magnitude of prewhitening is measured as a percentage of the zero-lag autocorrelation value r(0). In practice, typically 0.1% to 0.3% prewhitening is standard in processing.

If the percent prewhitening is given by a scalar, $0 < \varepsilon < 1$, then the normal Eq. 8. are modified

as follows:

$$\begin{pmatrix} \beta r_0 & r_1 & r_2 & \cdots & r_{n-1} \\ r_1 & \beta r_0 & r_1 & \cdots & r_{n-2} \\ r_2 & r_1 & \beta r_0 & \cdots & r_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{n-2} & r_{n-3} & \cdots & \beta r_0 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{n-1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{9}$$

where $\beta = 1 + \varepsilon$. Adding a constant εr_0 to the zero lag of the autocorrelation function is the same as adding white noise to the spectrum, with its total energy equal to that constant.

2.1.2. Predictive Deconvolution

The most widespread and well-known use of Wiener filtering is in predictive deconvolution. Prediction error filters can greatly attenuate multiples or ghost noise types. When the desired output is a time-advanced form of the input series itself, the process of filtering is called as Predictive deconvolution.

Given the input x(t), we want to predict its value at some future time (t + a), where a is prediction lag. Since the desired output x(t + a) is the time-advanced version of the input x(t), we only need to specify the right side of Eq. 5. for the prediction problem and this is actually achieved by the crosscorrelation between the desired output x(t+a) and the input x(t). Eq. 5. for this case, can be rewritten as follows:

$$\begin{pmatrix} \beta r_0 & r_1 & r_2 & \cdots & r_{n-1} \\ r_1 & \beta r_0 & r_1 & \cdots & r_{n-2} \\ r_2 & r_1 & \beta r_0 & \cdots & r_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{n-2} & r_{n-3} & \cdots & \beta r_0 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{n-1} \end{pmatrix} = \begin{pmatrix} r_a \\ r_{a+1} \\ r_{a+2} \\ \vdots \\ r_{a+N-1} \end{pmatrix} \tag{10}$$

It is thus clear that the predictive deconvolution is a general process that reduces to spiking deconvolution when the prediction lag is unity.

3. Background

3.1. Concept of Random Signals

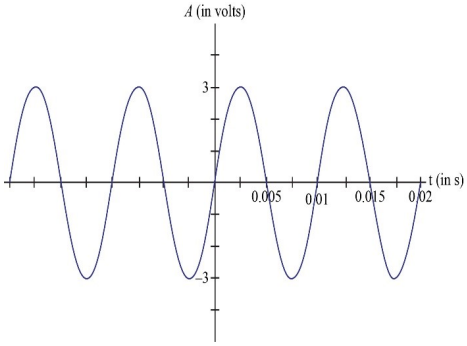
3.1.1. Stationary signal:

A signal is said to be stationary if its frequency or spectral components are not changing with respect to time. As an example we can say sine wave with a particular frequency is stationary because its frequency component remains same throughout time. It is to be mentioned that, with the progress of time the amplitude of sine wave may be degraded due to some frictional forces, but its frequency will remain unchanged. Now if we consider multitoned sine wave containing many frequencies, it will also be stationary if and only if its frequency components remain constant with time. So, stationarity is linked up only with the frequency components of a signal. Stationary signals can be further subdivided into deterministic and random signals. Deterministic signals are those that are predictable but the random signals are unpredictable at any instant of time.

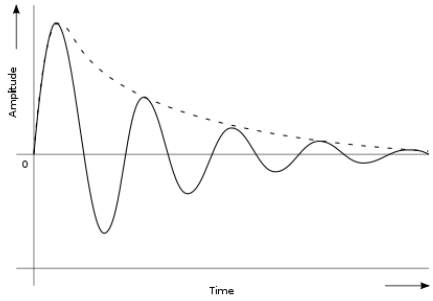
3.1.2. Transient signal:

Any sudden change in a signal is regarded as transient. A transient signal represents a short-burst of energy, caused by sudden change in state. Mathematically we say that a signal is said to contain transient whenever its Fourier expansion requires an infinite number of sinusoids. Basically transient signals exist for short duration of time. Examples of transient signals are-hammer excitation, explosion etc.

In a seismic section, each of the geophone traces represents the response of the source signal for the subsurface geology. Due to an infinite number of frequency content, seismic signals are



(a) Stationry signal



(b) Transient signal

Fig. 2: Stationary and transient signals

usually transient waveforms radiated from a localized natural or manmade seismic source. And the associated noise with the seismic signal is random in nature. The random process can be classified into two types one is stationary, and another is non-stationary. The noise of a seismic signal is stationary, i.e., the frequency contents of the noise are invariant with time whereas, for non-stationary signal the frequency content change with time.

3.2. Autocorrelation

Autocorrelation is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise, or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies. Autocorrelation is the correlation of a signal with a delayed copy of itself as a function of delay. It is the similarity between observations as a function of the time lag between them. It is a series of mathematical operations similar to those in convolution except that the two functions operating on each other are the same and one of them is not reversed with respect to the other.

Initially the two identical functions are aligned with each other as shown in Fig. 3. The ordinates of the respective curves are sampled at equal time intervals. The samples are multiplied at each of the ordinate positions, and the products are added to give what is defined as the autocorrelation function at zero time lag. The function has its highest value at this lag because all coincident ordinate values are equal. Sample by sample, the function is shifted relative to itself, the amount of shift being the lag, and the sum of the products being the output for that lag value.

3.3. Crosscorrelation

A similar mathematical procedure as autocorrelation, known as cross-correlation, is used to analyze the relationship between two different functions. Cross-correlation measures the similarity as a function of lag or time shift between two functions. At a lag T of zero, the two functions are juxtaposed so that their individual zero time axes are coincident. They are then progressively moved apart, and at each value of time lag T, the ordinate values sampled at closely spaced intervals are multiplied and the individual products added. For any values of the lag T at which the functions tend to have the same shape and thus seem to fit into each other or correlate, the cross-correlation function will have a local maximum.

3.4. Gain Application

Gain is a time-variant scaling in which the scaling function is based on a desired criterion. For instance, geometric spreading correction is applied to compensate for wavefront divergence early in processing, before deconvolution. Also before deconvolution, an exponential gain may be applied to compensate for attenuation losses. Often, gain is applied to seismic data for display. An automatic gain control (AGC) is applied to seismic data to bring up weak signals.

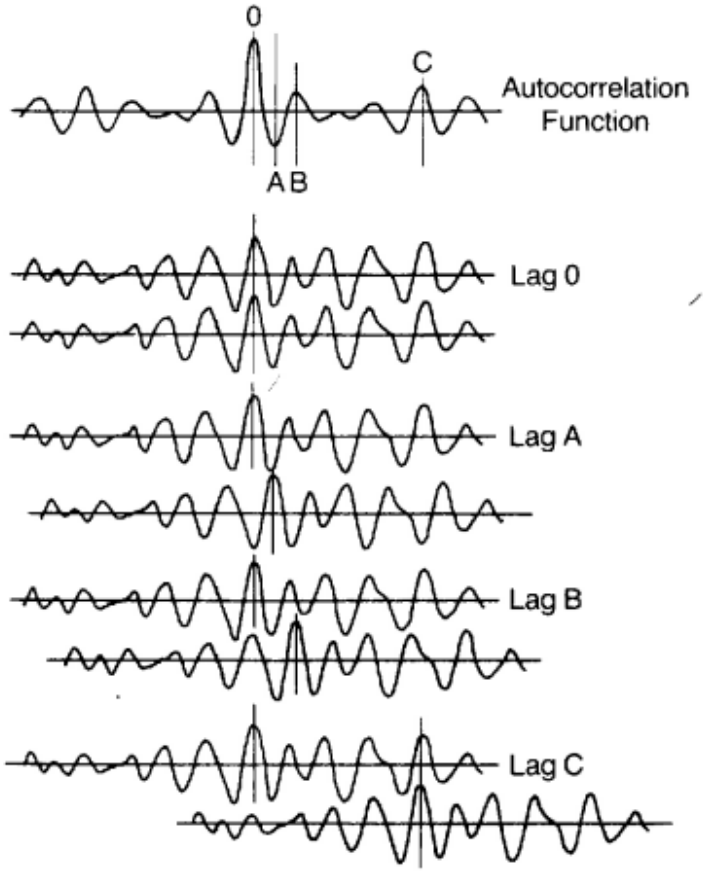


Fig. 3: Auto-correlation mechanism

3.4.1. RMS Amplitude AGC

The rms amplitude AGC function is based on the rms amplitude within a specified time gate on an input trace. This gain function is computed as follows. The input trace is subdivided into fixed time gates. First, the amplitude of each sample in a gate is squared. Second, the mean of these values is computed and its square root is taken. This is the rms amplitude over that gate. The ratio of a desired rms amplitude to the actual rms value is assigned as the value of the gain function at the center of the gate. Hence, the scaling function g(t) at the gate center is given by

$$g(t) = \frac{\text{desired rms}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}} \tag{11}$$

where x_i is the trace amplitude and N is the number of samples within the gate. Typically, we start out with a certain gate length at the shallow part of the trace. Gate length can be kept either constant or it can be increased systematically down the trace. At each gate center, the value of the gain function is computed as described above. Function g(t) then is interpolated between the gate centers.

3.5. PSD

For Wiener filtering we assumed the signal and noise as stationary stochastic processes and in the analysis of stationary noise we are normally not interested in the exact waveform or other

details that do not convey useful information. In most cases, we only want to know how strong the noise is, or whether it will prevent us from detecting a specific earth signal. Since with frequency filters we may be able to suppress noise in one part of the spectrum and still preserve signals in another part, it is necessary to consider the spectral distribution rather than the total strength of both signals.

As discussed in section 3.1 transient signals have a finite energy and zero power when averaged over all times, while stationary signals have an infinite energy but a finite power. For a quantification of stationary noise, the concept of Power Spectral Density (PSD) is therefore appropriate while transient signals are more adequately described by their Energy Spectral Density (ESD).

Any physical signal, for example, a time series x(t), can be decomposed into a number of discrete frequencies or a spectrum over a continuous range of frequencies. When the total energy of the signal is finite and concentrated around a finite time interval, we can compute its energy spectral density. But for continuous signals over all time i.e., for a stationary process the total energy of a signal over all time would generally be infinite. Summing or integrating the spectral components yields the total power for a physical process or variance for statistical process, i.e., identical with results obtained by integrating over the time domain (Parseval's theorem). In such cases we can not use energy spectral density and thus power spectral density is needed. So PSD can be defined as the spectral energy distribution that would be found per unit time.

It can be mathematically expressed for a signal x(t) as,

$$S_x = |X(f)|^2 \tag{12}$$

where $X(f) = \int_{-\infty}^{\infty} \mathrm{e}^{-i\omega t} x(t) dt$
and $\omega = 2\pi f$, the angular frequency.

For stationary signals like seismic background noise, calculating the PSD of a long but finite section is the only possible form of a spectral analysis. The PSD is not equivalent to a Fourier transform; the signal cannot be reconstructed from it with an inverse transformation because the phase information is lost. For finite sections of a stationary signal we can of course use the discrete Fourier transformation if we later need a backward transformation. So fft is the best method for calculating PSD.

An alternative way of determining the spectral density of energy or power is to calculate the Fourier transformation of its autocorrelation. An autocorrelation can be defined for stationary signals that have no Fourier transform; the signal is then essentially defined via its autocorrelation, and needs not be explicitly known. The method can as well be applied to finite sections of a signal. The properly normalized Fourier transform is the PSD of the signal. A signal's autocorrelation and PSD are Fourier transform pairs. The method is now obsolete as a practical tool and it dates back to the time when the Fast Fourier Transformation was not available.

4. Objective

From the above discussion we are now familiar with the least square inverse filtering specially optimum Wiener filtering. We know that Wiener filter is based on a statistical approach where it tries to minimize the error between the desired output and the actual output in least square sense. So, as the objective of this project we are interested in applying the concept of Wiener filtering on a real field seismic data and to study the actual effect of this filtering. In order to apply this technique on seismic data we need to construct appropriate environment consistent with our purpose through coding. For this application we used one data set openly available online at `https://www.morganclaypool.com/page/mousa`.

5. Methodology

At begining, though we knew the fundamental concept of Wiener filtering, we were having trouble in deciding the desired wavelet as far as seismic signal has been considered. Later we came to know that in seismic data processing the actual application of Wiener filter is in deconvolution process. Then it becomes easy for visualization of the processes. So we found different deconvolution concepts and out of these, spiking and predictive deconvolution are the most important in seismic. Further predictive deconvolution is the most general deconvolution and we can return to the spiking deconvolution as a special case of this predictive deconvolution. We wrote two codes, each for spiking and predictive deconvolution and also few others in order to analyse their workings.

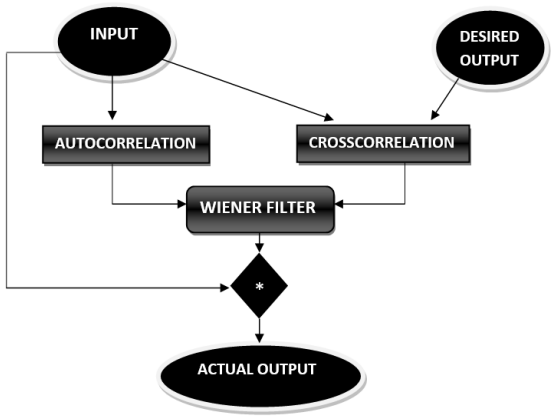


Fig. 4: Flowchart for Wiener filter design

5.1. Computational Algorithm

Fig. 4 shows the basic Wiener filter design strategy which is applicable for both spiking and predictive deconvolution. The computational mechanisms of the spiking and predictive deconvolution are briefly explained below:

5.1.1. Wiener Spiking Deconvolution Filter

To perform spiking deconvolution we wrote a Matlab function named WSDF representing Wiener Spiking Deconvolution Filter. The basic algorithm for this function assumes the simple matrix Eq. 9. where the concept of prewhitening is also considered. The important part in the function is the computaion of the auto-correlation and cross-correlation. We did not use the inbuilt matlab function 'xcorr' for these computation rather we follow the actual computational technique as discussed in sections 3.2 and 3.3. The important fact about the prewhitening is that we can simply add a constant ϵr_0 to the zero lag of the autocorrelation function to effect the same as adding white noise to the spectrum, with its total energy equal to that constant. What we found difficult initialy in constructing the desired signal becomes simply a zero lag spike in case of spiking deconvolution. Lastly to convert the autocorrelation vector to a Toeplitz matrix, the inbuit matlab function 'toeplitz' was used.

5.1.2. Wiener Predictive Deconvolution Filter

To perform predictive deconvolution we wrote another matlab function named WPDF representing Wiener Predictive Deconvolution Filter. The basic algorithm for this function assumes the simple matrix Eq. 10. which is very similar to what we did for spiking deconvolution except for the fact that the desired wavelet is the time-advanced version of the input series itself. As it is known, that for prediction lag of unity gives rise to the special case of spiking deconvolution out of predictive, we needed to consider this fact and also implement this in the function WPDF.

6. Results and Discussion

The Designed Wiener filters for spiking and predictive deconvolution are applied on a shot gather. This shot gather includes 33 spatial samples and 1501 temporal samples for each spatial point i.e., it is in demultiplexed format. From the data processing point of view, the next step to perform on this raw data set is the deconvolution process.

The output obtained from both type of filtering is discussed below. We have also checked the effects of different filter parameters like operator length, prewhitening and prediction lag on the data.

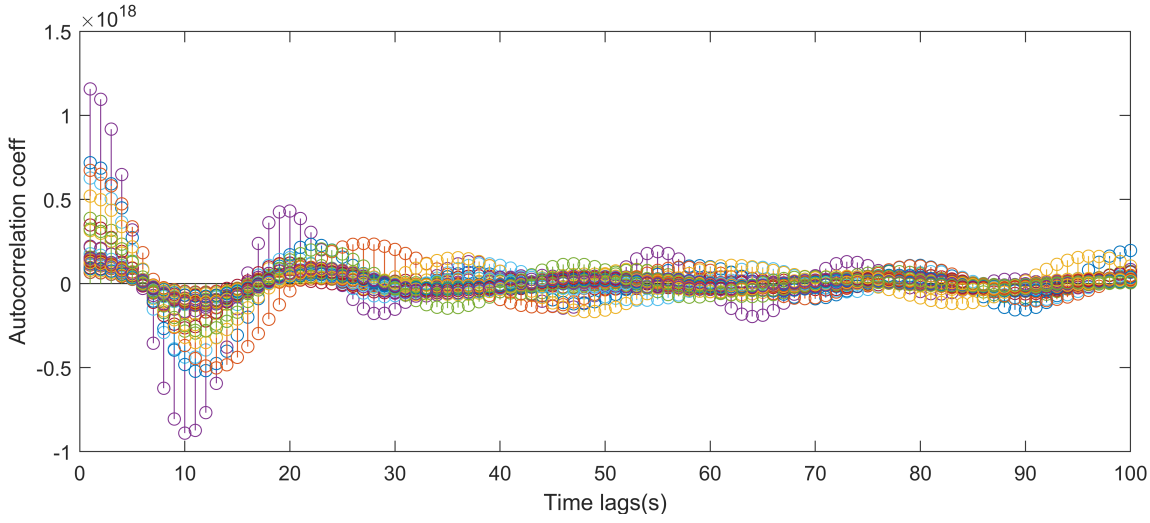


Fig. 5: Autocorrelation of seismic traces

Fig. 5 shows the autocorrelation of each trace in the shot gather. It can be seen from this figure that the autocorrelation of seismic traces possess maximum peak value at zero lag. At some time lags autocorrelation value are extremely large indicating presence of repetitive seismic events e.g., multiples, ghosts or reverberations. The usual plot of the autocorrelogram is shown in Fig. 6a. This autocorrelation function is critical in picking the deconvolution parameters of gap also called minimum autocorrelation lag and the operator length sometimes called maximum autocorrelation lag. Fig. 6b shows the plot of input shot gather without any kind of operation applied.

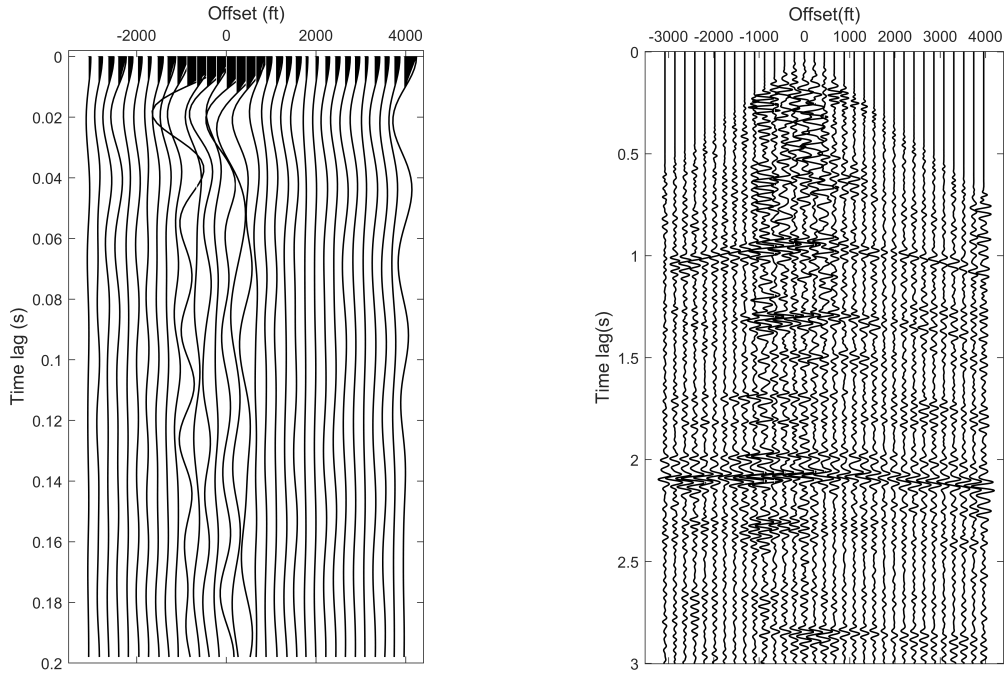
6.1. WSDF

Spiking deconvolution has been done with desired wavelet as a zero lag spike. Autocorrelation of input seismic trace and cross correlation of input seismic trace and desired zero lag spike have been determined to evaluate the wiener filter coefficients, which further convolved with the input traces to find the deconvolved output. As an example of the working of WSDF we used filter length of 100 with white noise percentage of 0.1% and applied on the input.

The result is shown in Fig. 7a and what we can see is that the seismic traces are vertically resolved.

6.2. WPDF

If the desired output is a time advanced form of the input series, we use Predictive deconvolution filter. It is the most useful deconvolution filter for seismic data processing. For WPDF we have used operator length of 100, a prewhitening percentage of 0.1% and prediction lag of 300. The result is shown in Fig. 7b and what we can see is that the repetitive part of the shot gather has been attenuated.



(a) Autocorrelogram (b) Input shot gather

Fig. 6: Plot of Autocorrelogram and input traces

Comparison between spiking and predictive deconvolution:
Fig. 7 shows the difference between the outputs after applying spiking and predictive deconvolution. From deconvolution theory, we can easily determine what is the reason behind this difference. Predictive deconvolution only predicts the multiples and attenuate them, but does not remove source wavelet while spiking deconvolution collapses wavelets and attenuate multiples. In other words, we can say spiking deconvolution is the simplified version of predictive deconvolution when the prediction lag equals unity.

6.3. Effect of Operator length and Prediction distance

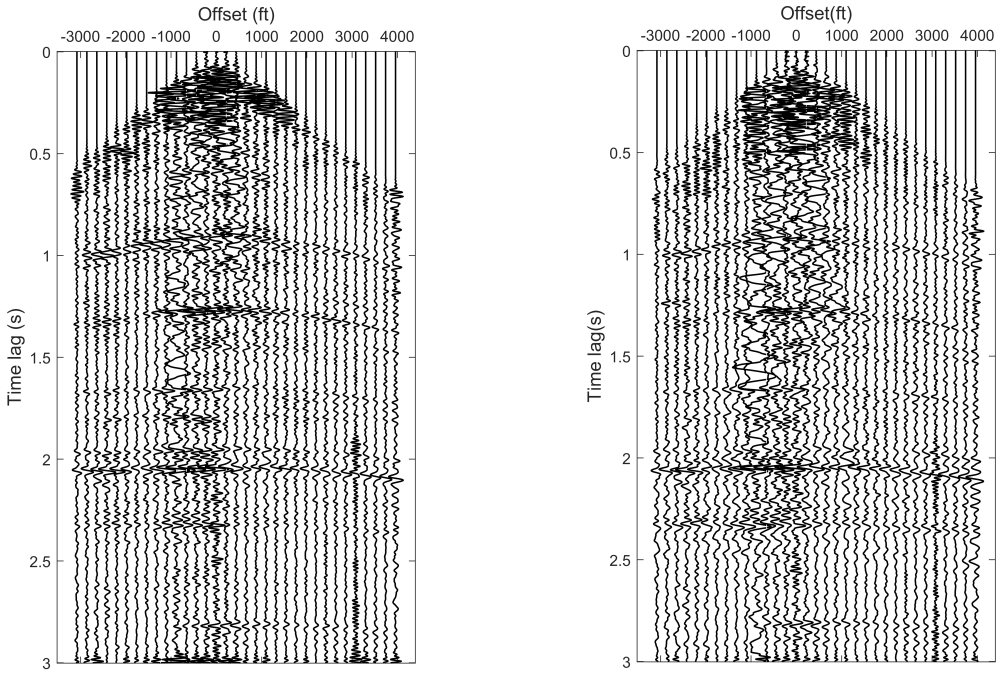
To analyze the effect of operator length, prewhitening and prediction lag we used different values of these parameters to better understand the working of spiking and predictive deconvolution.

6.3.1. On WSDF

We applied spiking deconvolution for different operator lengths of 10, 50, 100, 120, 150 for a constant whitening percentage of 0.1%. The result improves as the more number of coefficient be included in the filter operator. It is to be mentioned that to attenuate multiples the operator length must be greater than the period of multiples. But too large filter length may filter out valuable subsurface information. That is why different operator lengths have applied on seismic trace to decide which one is giving the optimum result. From the outputs of different operator lengths, we can see with operator length of 120 results the best output with most of the multiples have attenuated and having a better vertical resolution.

6.3.2. On WPDF

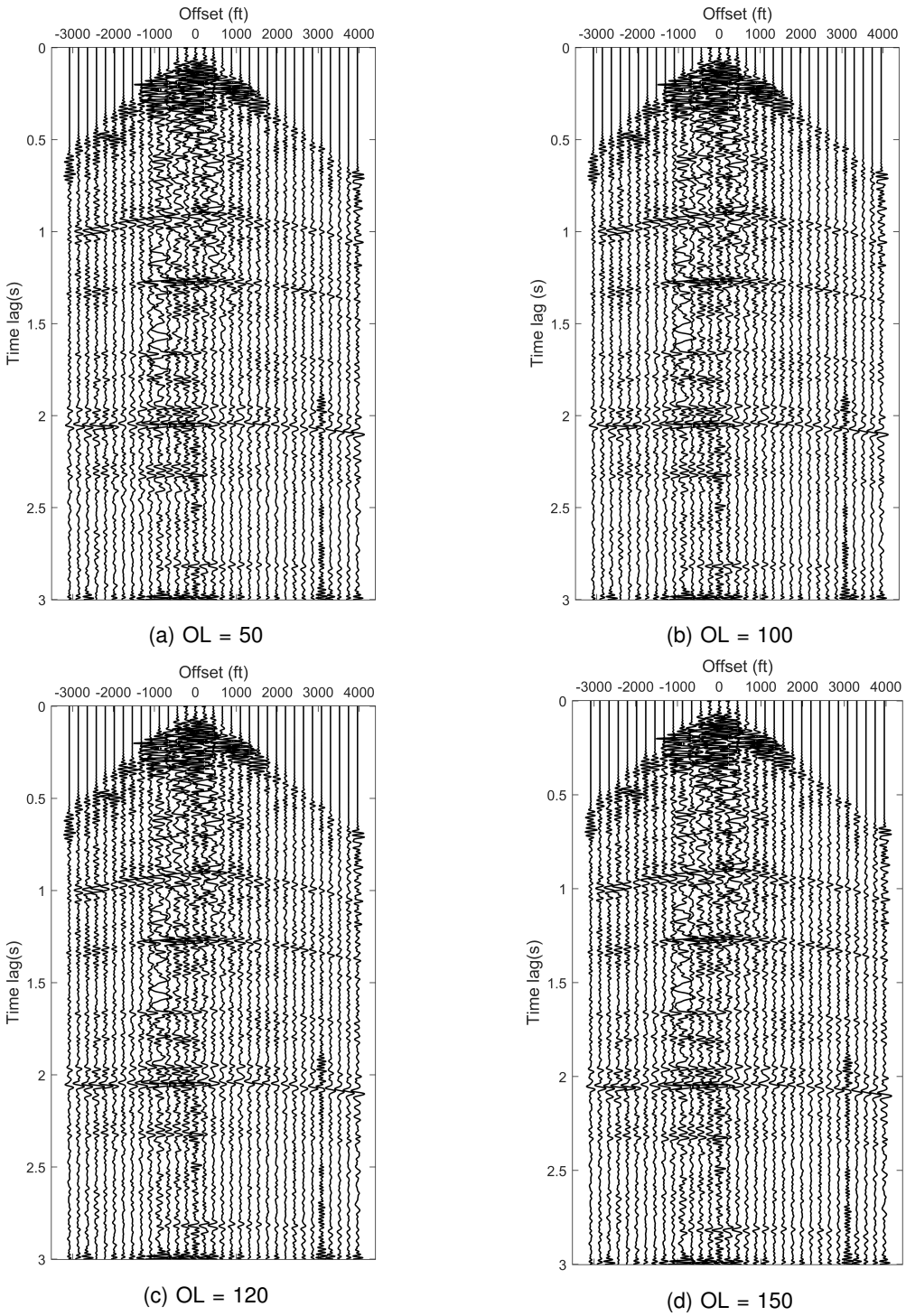
In order to examine the effect of the prediction lag parameter we apply WPDF of the input data set keeping the value of operator length and percent prewhitening constant at 100 and 0.1% while



(a) Spiking Deconvolved (b) Predictive Deconvolved

Fig. 7: Plot of Deconvolution outputs

prediction lag is varied. We took the lag's value to be 150, 200, 250 nad 300. At a 150 ms prediction lag, predictive deconvolution does nothing to the input wavelet because almost all the lags of its autocorrelation have been left untouched. This experiment has an important practical implication: Under the ideal, noise-free conditions, resolution on the output from predictive deconvolution can be controlled by adjusting the prediction lag. Unit prediction lag implies the highest resolution, while a larger prediction lag implies less than full resolution. However, in reality, these assessments are dictated by the signal-to-noise ratio. The deconvolved output using a unit prediction lag contains high frequencies; nevertheless, resolution may be degraded if the high-frequency energy is mostly noise, not signal.



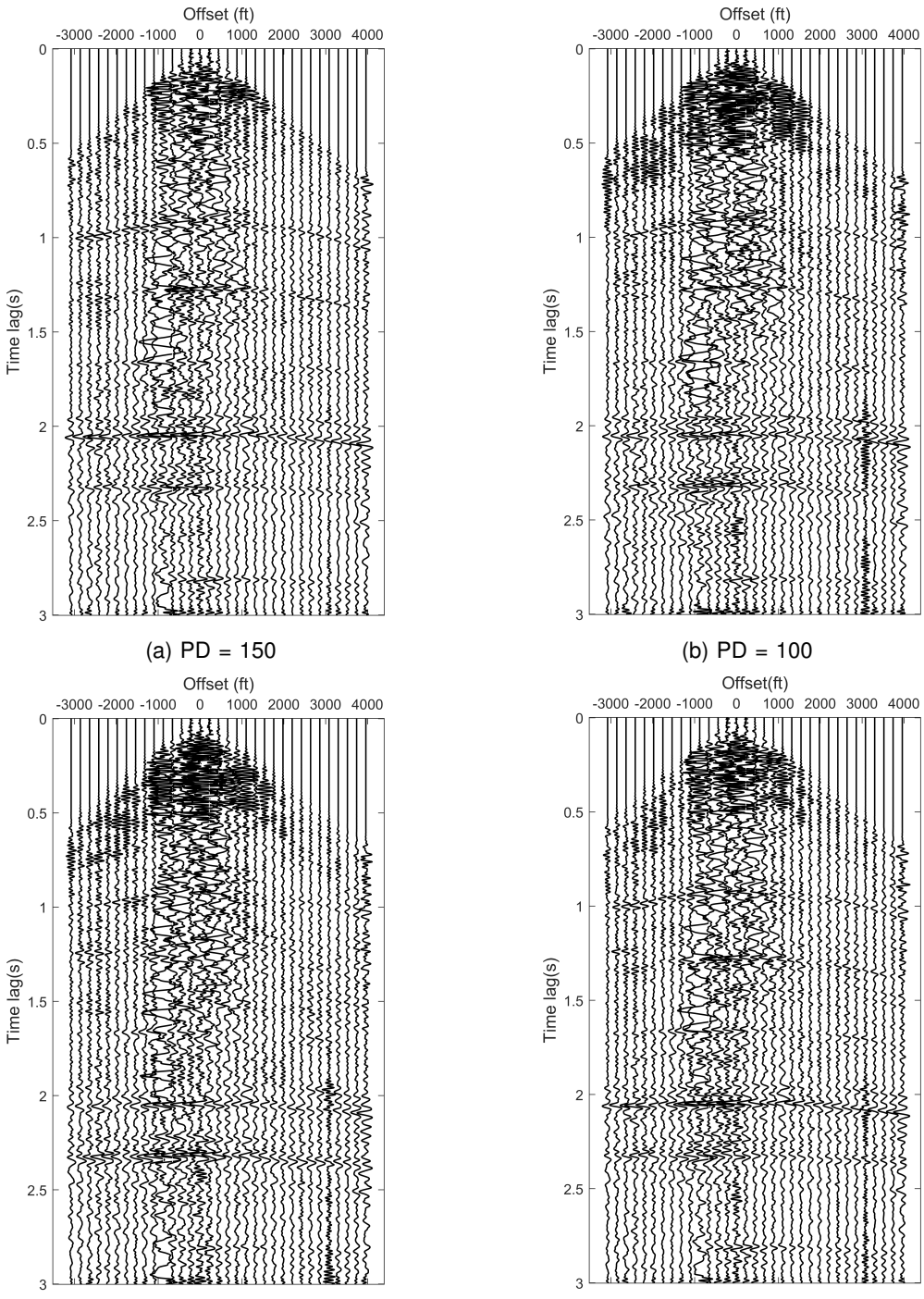
(a) OL = 50

(b) OL = 100

(c) OL = 120

(d) OL = 150

Fig. 8: Effect of OL on WSDF outputs



(a) PD = 150

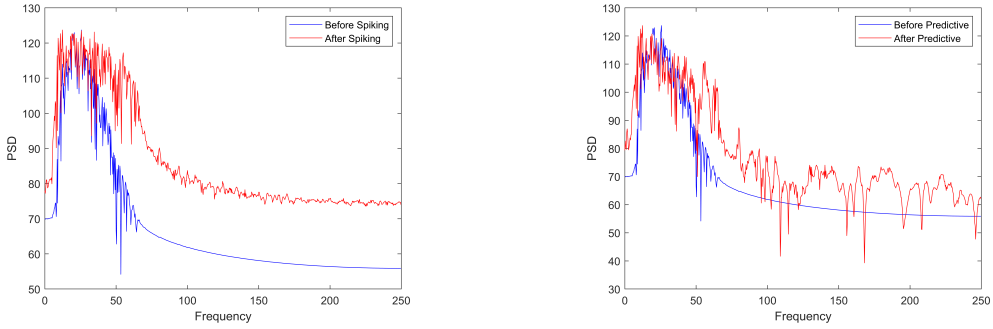
(b) PD = 100

(c) PD = 250

(d) PD = 300

Fig. 9: Effect of PD on WPDF outputs

6.4. Results from PSD



(a) PSD for spiking deconvolution

(b) PSD for predictive deconvolution

Fig. 10: Effect of deconvolution on PSD of the averaged trace grather

PSD measures the distribution of signal power over a range of frequency. We can determine the impact of filtering and modulation of power signals based on the analysis of PSD. As can be seen from the Fig. 10 PSD became more white after deconvolution. This supports the reliability of the deconvolution process for both spiking and predictive.

7. Conclusions

Deconvolution is an important processing step in seismic data processing to attenuate repetitive seismic events that can not be easily removed by move-out correction or velocity filtering. Out of different deconvolution processes spiking and predictive deconvolution are largely used in seismic data processing. The results we have got simply show that though spiking deconvolution removes source wavelet effect and attenuated multiples, predictive deconvolution does not remove the effect of source wavelet, rather tries to estimate and then remove the predictable parts of a seismic traces, usually multiples. Consequently, deconvolution results into shortening of wavelets, multiple free seismic trace, better vertical resolution and all of these help to resolve reflectivity series. It is to be mentioned that, wiener filter plays a significant role here which transforms the input to any desired output.

Supplimentary information

Used data and codes associated with this article can be found in the repository ***Wiener-Filter*** on GitHub. Link to this repository: `https://github.com/Pulak-Biswas/Wiener-Filter.git`

Acknowledgements

We would like to show our gratitude to thank Dr. Swarandeep Sahoo, and Dr. Srinivasa Rao Gangumalla for taking classes on signal processing which helped us a lot in dealing with some computational problems faced during this project.

We are also very grateful to Prof.Priya Ranjan Mohanty, for influencing us in gaining interest in seismic data processing.

Lastly, we would like to thank Wail A. Mousa and Abdullatif A. Al-Shuhail for keeping their used seismic data open to all, available online at: `https://www.morganclaypool.com/page/mousa`

References

[1] Bormann, P. and Wielandt, E. *Seismic Signals and Noise* In: Bormann, P. (Ed.), New Manual of Seismological Observatory Practice 2 (NMSOP2), Potsdam : Deutsches GeoForschungsZentrum GFZ, 1-62, 2013. https://doi.org/10.2312/GFZ.NMSOP-2_ch4

[2] Dobrin, M. B., *Introduction to geophysical prospecting*, McGraw-Hill Book Co, 1960.

[3] Mousa, Wail A and Al-Shuhail, Abdullatif., *Processing of seismic reflection data using MATLAB*. [San Rafael, Calif.], Morgan & Claypool., 2011 http://site.ebrary.com/id/10535298.

[4] Mousa, W., *Advanced Digital Signal Processing of Seismic Data*, Cambridge University Press, 2020. doi:10.1017/9781139626286

[5] Proakis, John G. and Manolakis, Dimitris G., *Digital Signal Processing*, Prentice Hall , Inc., 2007.

[6] Wikipedia: https://en.wikipedia.org/wiki/Wiener_filter

[7] Yilmaz, Öz., *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data*, Society of Exploration Geophysicists, 2001. https://doi.org/10.1190/1.9781560801580

The functions developed for this project may contain bugs and we will be working on these for their betterment.

Appendix A

Matlab code for Wiener Spiking Deconvolution Filter

```
function [Dac,lags,otpshot]=WSDF(inpshot,Flen,WN,ts)
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%
% % WIENER SPIKING DECONVOLUTION FILTER -%-  PULAK BISWAS    %
% % Example: [Dac,lags,otpshot]=WSDF(inpshot,Flen,WN,ts)     %
% %       INPUTS     inpshot: seismic shot gather(s)         %
% %                  Flen: Filter length                     %
% %                  WN: White noise percentage              %
% %                  ts: time sampling interval              %
% %      OUTPUTS     Dac: Autocorrelation fnc                %
% %                  lags: Autocorrelation lags              %
% %                  otpshot: actual output                  %
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%

box=size(inpshot);                % size of input data
Dwav=[1,zeros(1,box(1)-1)];       % desired wavelet

Dac=zeros(Flen,box(2));
Dcc=zeros(1,Flen);
for i=1:box(2)
    corr=zeros(1,Flen);
    for j=1:Flen
        for k=1:box(1)
            if k+j ≤ box(1)+1
                corr(j)=corr(j)+inpshot(k,i)*inpshot(k+j-1,i);
            end
        end
    end
    Dac(:,i)=corr;
    corr=zeros(1,Flen);
    for j=1:Flen
        for k=1:box(1)
            if k+j ≤ box(1)+1
                corr(j)=corr(j)+inpshot(k,i)*Dwav(k+j-1)';
            end
        end
    end
    Dcc=Dcc+corr;
end
lags=0:ts:(Flen-1)*ts;

Dcc=abs(Dcc);
Dac2=sum(Dac,2);
Dac2(1)=Dac2(1)+WN/100;
Tpltz=toeplitz(Dac2);

otpshot=zeros(box(1)+Flen-1,box(2));
hfilt=Dcc/(Tpltz);
for i=1:box(2)
    otpshot(:,i)=conv(inpshot(:,i),hfilt);
end

otpshot=otpshot(1:box(1),:);
```

Appendix B

Matlab code for Wiener Predictive Deconvolution Filter

```
function [Dac,lags,otpshot]=WPDF(inpshot,Flen,WN,ts,PD)
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%
% % WIENER PREDICTIVE DECONVOLUTION FILTER -%-  PULAK BISWAS    %
% % Example: [Dac,lags,otpshot]=WPDF(inpshot,Flen,WN,ts)        %
% %       INPUTS     inpshot: seismic trace or wavelet          %
% %                  Flen: Filter length                        %
% %                  WN: White noise percentage                 %
% %                  ts: time sampling interval                 %
% %                  PD:  Prediction distance                   %
% %                       = 1 for Spiking                       %
% %      OUTPUTS     Dac: Autocorrelation fnc                   %
% %                  lags: Autocorrelation lags                 %
% %                  otpshot: actual output                     %
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%

box=size(inpshot);                                          % size of input data
Olen = box(1)+Flen-1;

if PD == 1
    Dwav=[PD,zeros(1,box(1)-1)];                             % desired wavelet
else
    Dwav = [inpshot(PD+1:box(1)),zeros(1,Olen-box(1)+PD)]';  % desired wavelet
end

Dac=zeros(Flen,box(2));
Dcc=zeros(1,Flen);
for i=1:box(2)
    corr=zeros(1,Flen);
    for j=1:Flen
        for k=1:box(1)
            if k+j ≤ box(1)+1
                corr(j)=corr(j)+inpshot(k,i)*inpshot(k+j-1,i);
            end
        end
    end
    Dac(:,i)=corr;
    corr=zeros(1,Flen);
    for j=1:Flen
        for k=1:box(1)
            if k+j ≤ box(1)+1
                corr(j)=corr(j)+inpshot(k,i)*Dwav(k+j-1)';
            end
        end
    end
    Dcc=Dcc+corr;
end
lags=0:ts:(Flen-1)*ts;

Dcc=abs(Dcc);
Dac2=sum(Dac,2);
Dac2(1)=Dac2(1)+WN/100;
Tpltz=toeplitz(Dac2);

otpshot=zeros(Olen,box(2));
hfilt=Dcc/(Tpltz);
for i=1:box(2)
    otpshot(:,i)=conv(inpshot(:,i),hfilt);
end

otpshot=otpshot(1:box(1),:);
```

Appendix C

Matlab code for the calculation PSD.

```
function[sx,fr]=PSD(input,ts)
box = size(input);
if rem(box(1),2)==0
    N=box(1);
else
    N=box(1)-1;
end
Fs=1/ts;
freq = Fs*(-N/2:N/2-1)/N;
inpft=fft(mean(input'),N);
sx = (1/(Fs*N)) * (abs(inpft(1:N/2)).^2);
sx=10*log10(sx(1:N/2));
fr=freq(N/2+1:end);
end
```

Appendix D

Matlab code for RMS Amplitude AGC

```
function [gnout]=rmsGain(data,ts,glen)
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%
% %             RMS Amplitude AGC -%- PAYEL HAZRA                 %
% %  Example: [output]=rmsGain(data,ts,glen)                      %
% %   INPUTS     data: traces                                     %
% %              dt: sampling interval in sec                     %
% %              glen: length of the agc gate in secs             %
% %  OUTPUTS   gnout: gain applied traces                         %
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%

box=size(data);

% AGC_gate fnc
 L = glen/ts+1;
 L = floor(L/2);
 h = triang(2*L+1);                    %gate function

 for i = 1:box(2)
   aux =  data(:,i);
   e = aux.^2;                         %squaring of trace amplitude
   rms = sqrt(conv2(e,h,'same'));      %conv bwn trace and gate fnc
   epsi = 1.e-10*max(rms);
   op = rms./(rms.^2+epsi);
   gnout(:,i) = data(:,i).*op;
 end

 % RMS Amplitude AGC
for j = 1:box(2)
    aux = gnout(:,j);
    rmsa=sqrt(sum(aux.^2)/box(1));
    gnout(:,j)=gnout(:,j)/rmsa;      %normalized by rms amplitude
end
```

Appendix E

For plotting the shot gather as wiggle we used wigb function from SeismicLab package with little modification.

```
function wigb(tres,scal,offset,time)
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%
% %  WIGB: Plot seismic data using wiggles.                                    %
% %  Example: WIGB(tres,scal,offset,time)                                      %
% %  INPUTS  tres: seismic traces as columns                                   %
% %          scal: multiple data by scal                                       %
% %          offset: horizontal offset axis                                    %
% %          time: vertical time axis                                          %
% %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%
box=size(tres);
trmx= max(abs(tres));
amx=mean(trmx);

if box(1) ≤ 1; disp('ERROOR: No of tress has to be more than 1'); end

% take the average as dx
dx1 = abs(offset(2:box(2))-offset(1:box(2)-1));
dx = median(dx1);

dz=time(2)-time(1);

if scal == 0; scal=1; end
tres = tres * dx /amx;
tres = tres * scal;

% set display range

x1=min(offset)-2.0*dx; x2=max(offset)+2.0*dx;
z1=min(time)-dz; z2=max(time)+dz;

set(gca,'NextPlot','add','Box','on', ...
    'XLim', [x1 x2], ...
    'YDir','reverse', ...
    'YLim',[z1 z2]);
fillcolor = [0 0 0];
linecolor = [0 0 0];
linewidth = 1.;

time=time';     % input as row vector
zstart=time(1);
zend  =time(box(1));

for i=1:box(2)
    if trmx(i) ≠ 0     % skip the zero traces
        tr=tres(:,i);       % --- one scale for all section
        s = sign(tr) ;
        i1= find( s(1:box(1)-1) ≠ s(2:box(1))); % zero crossing points
        zadd = i1 + tr(i1) ./ (tr(i1) - tr(i1+1));  %locations with 0 amplitudes
        aadd = zeros(size(zadd));
        pos = find(tr >0);
        [zz,iz] = sort([pos(1); zadd]);     % indices of zero point plus positives
        aa = [tr(pos(1)); aadd];
        aa = aa(iz);

    if tr(1)>0
        a0=0; z0=1.00;
    else
        a0=0; z0=zadd(1);
    end
    if tr(box(1))>0
        a1=0; z1=box(1);
    else
```

```
63          a1=0; z1=max(zadd);
64      end
65
66          zz = [z0; zz; z1; z0];
67          aa = [a0; aa; a1; a0];
68          zzz = zstart + zz*dz -dz;
69
70          patch( aa+offset(i) , zzz,  fillcolor);
71          line( 'Color',[1 1 1],'EraseMode','background',  ...
72          'Xdata', offset(i)+[0 0], 'Ydata',[zstart zend]); % remove zero line
73          line( 'Color',linecolor,'EraseMode','background',  ...
74          'LineWidth',linewidth, ...
75          'Xdata', tr+offset(i), 'Ydata',time);    % negatives line
76      else % zeros trace
77          line( 'Color',linecolor,'EraseMode','background',  ...
78          'LineWidth',linewidth, ...
79          'Xdata', [offset(i) offset(i)], 'Ydata',[zstart zend]);
80      end
81  end
82
83  x=500;y=20;
84  w=400;h=650;
85  set(gcf,'position',[x y w h]);
86  set(gca,'xaxislocation','top')
```