Chapter 11
Spectral methods and advanced gaining methods for seismic data

There is a class of methods that are best called spectral methods because they modify the amplitude and/or phase spectrum of the data. There are several reasons for performing such operations. We may find that there is high or low frequency noise in the data that may be present in the original data, or which may be introduced by processing tools. The second is to suppress multiples, and the third is to sharpen the waveform to more clearly define reflection arrivals. There is, of course, the issue of correlating vibrator profile data with the vibroseis sweep as a prelude to further processing.

Some of these techniques are clearly filtering operations, wherein a particular filter is convolved with the data. Other techniques are best thought of as a deconvolutional processes, that is to say, a process by which data are “inverted” in some sense, to remove a particular response. We may think of both operations as being related, in that deconvolution is a convolution with an inverse of a signal. He we find that deconvolution is also a “filter.”

There are many such methods that have been developed over the decades since digital data processing was first introduced in seismic data processing in the mid 1950s, but we will discuss only a small subset of these, to give you a general idea of what to expect in a commercial environment.

We apply many of these operations prior to velocity analysis, or in conjunction with velocity analysis, or as a prelude to migrating the data.

11.1 Common assumptions of spectral method processing

Seismic data result from the introduction of seismic energy into the subsurface followed by the subsequent recording of reflections either on the surface of the earth or in a well bore. Such data have a natural frequency band and a natural phase spectrum. We may seek to change some of these characteristics as part of processing, but we have
common assumptions about our data that we make, or that we impose on the data as a precondition for further processing.

The seismic source may have a time history that makes it appear complicated. For example, a marine airgun signature may have a bubble pulse that follows some time after the main signal, see Figure 11.1. Because we may think of all resulting reflections as resulting from a convolutional process involving this source wavelet, complications in the source waveform may cause the reflections to appear unduly complicated.

Another source of waveform complication in ocean seismic surveys results from a phenomenon called ghosting. In addition to the direct arrival from the source, there is an additional arrival that originates from a reflection path that begins at the source, bounces off of the water surface, and then travels to the subsurface. Similarly, in addition to the direct reflection from the subsurface, the receiver may record an additional signal that has traveled to the ocean surface. These ghost reflections are also convolved with the reflectivity series.

Yet another source of data complication are multiples. These include reverberations in the water column, with waves bouncing between the sea surface and the sea bottom. This is not just the first reflection from the sea bottom, but can include large portions of the seismic arrivals. Again, these reverberations, called pegleg (named for a pirate’s artificial limb) multiples. While it may be tempting to think of multiples as being merely added to the data, these too act as secondary sources and are thus convolved with the
Finally, there may be ambient noise which is added to the data. This can include cultural noise, or natural noise from such sources as wind, or in ocean surveys noise from sea creatures or from ships, including the ship that is dragging the towed airgun/hydrophone array.

To deal with these issues, we apply deconvolutional methods. To apply such methods, we make some physically reasonable simplifying assumptions. These are causality and the minimum phase (delay) and white spectrum assumptions. Underlying all of this is the assumption that seismic wave propagation is a linear system.

### 11.1.1 Causality

If the source is an explosion or a pulse from an airgun, or even a sweep from a vibrator, the resulting data have the property that they are causal, which is to say that the wavelets have a definite beginning time. Causality means that there can be no signal before time zero, or in the case of propagating arrivals, there can be no arrivals before the shortest traveltime determined by the velocity function for the medium. This, of course, is a basic principle of physics and should not be a surprise.

Many processes that change the frequency spectrum also will tend to cause a distortion in the waveform resulting in signals that may appear later than the time predicted by the wavespeed. Sometimes we deliberately change the phase characteristics of the wavelets in the data as to make the wavelet symmetric about the expected arrival time, thus giving the appearance that energy is coming in a bit earlier than the predicted arrival time. In this case, we such signals are called zero-phase waveforms. If the data are then processed to appear to be similar to sinc functions, which is to say, symmetric zero-phase signals, then the reflectors will occur at times of the peaks of these “bandlimited delta functions”.

### 11.1.2 Minimum phase (aka minimum delay)

A signal which has a definite time of beginning and also has the majority of its energy in the beginning part of the waveform is called a minimum phase (aka minimum delay) wavelet. There is a precise mathematical definition, but physically we may consider any “front-loaded” signal to be or to approximately a minimum phase waveform. Again, many linear, or mildly nonlinear physical processes will produce signals that have this property. Errors that we see, then could be expected to result from a deviation from this frontloadedness.

### 11.1.3 White spectrum

The term “white spectrum” is an allusion to the notion of white light being composed of a full visual spectrum of frequencies. For seismic data, many processes that we apply would become unstable if amplitudes at particular frequencies were to be zero or near zero. For *deterministic processes* the instability comes from division by zero or division by a small number, as might be encountered by performing deconvolution in the Fourier
transform domain. For statistical processes which are viewed in some sense as matrix operations, the issue is to introduce a perturbation that moves the system away from its null space.

We know that all data are bandlimited, so the remedy is to include a small “white noise” term that will prevent this instability. The expense is that all such operators will introduce noise. Generally, frequency filtering is a remedy for this noise issue.

There are four operations that are the working tools of digital signal processing. These are convolution, cross-correlation, autocorrelation, and deconvolution.

11.1.4 Linear systems

Geophysicists have realized a tremendous benefit from a simple concept. That is the concept called the convolutional model of seismic wave propagation. Because the wave equation is a linear partial differential equation equation, the simplest way of looking at all processes involving the wave equation is to consider the processes as being a linear system.

The common metaphor is that of a “black box.” That is, we are assuming that a geophysical process is a linear system with an input and an output, but the only other thing that we know about the black box is that we assume it behaves in a linear fashion. That is, we assume that the output from a process depends linearly on the input.

A result known as “Green’s theorem” tells us that if we have a particular solution called the “Green’s function” of the linear system, which is to say the output of the linear system given the input of a Dirac delta function, we may form all possible solutions by the convolution of a given input with the Green’s function. The Green’s function is also known as the “impulse response” or the “transfer function.” (In modern mathematical usage the term ”fundamental solution” is often seen.) This is a feature of the principle of superposition.

We have three mathematical systems that we will switch back and forth freely between. The first is the continuous representation. In this representation we write signals and filters as continuous functions of time, apply the Fourier transform, and perform computations in the frequency domain. While this formulation is straightforward and yields a great deal of insight into our problems, this is a more computationally expensive approach than to perform our operations in the time domain.

When we represent data as discrete samples in the time-domain, there is a representation known as the Z-transform, which allows digital interactions to be represented as the multiplication and division of signals and filters as polynomials.

Yet a third representation is to note that operations that are represented as multiplications of discretely represented time-domain signals may be represented in linear algebra form as matrix multiplications, or matrix-vector multiplications.

Each of these representations has its own merits in permitting insight into the processes being discussed.
What does “linear” mean and why is it good?

In the experience of mathematical physics, many processes can be described or may be approximated by linear ordinary or partial differential equations. The short answer of why linear is good, is that linear makes the mathematics easier.

If multiply the input to a linear system by a scalar then the output is scaled by the same value. If we shift the input to the black box, the output is shifted by the same amount. That’s it.

Furthermore, linear systems have a property called the principle of superposition. This means that new solutions of a linear system may be found by forming the linear combination of previously determined solutions. This fact allows transform theory to be applied. That is, we may use invertible mathematical transformations to decompose input functions into a family of simpler functions, apply the linear system to these simpler functions, and then add up the results to yield the output for the full function.

11.2 The three mathematical languages of signal processing

There are three formulations that are useful in signal processing. These are the continuous function, the Z polynomial, and the matrix and vector representations.

The continuous function representation is made via the Fourier transform. Though, formally we can write signals, wavelets, and filters as continuous functions, these are applied discretely as digital data equivalents.

The Z-transform representation replaces all functions with a polynomial representation that has many of the same properties of the frequency domain representation.

Finally, we can represent digital data operations as matrix-on-matrix, or matrix-on-vector multiplications.

11.2.1 The Forward and Inverse Fourier Transform

The most common decomposition is called the Fourier transform which decomposes a given signal into sines and cosines. The formal representation of the forward Fourier transform

\[ F(\omega) = \int_0^\infty f(t)e^{i\omega t}dt \]  

and the inverse Fourier transform

\[ f(t) = \int_{-\infty}^\infty f(\omega)e^{-i\omega t}d\omega. \]

Here \( \exp(i\omega t) = \cos(\omega t) + i\sin(\omega t) \). In the first expression, the function \( f(t) \) is decomposed into to the values of all of the sines and cosines of different frequencies. The inverse process collapses the sines and cosines, sums up the values for each frequency, yielding the original function \( f(t) \) back.
We can use this definition of the forward and inverse Fourier transforms to formulate convolution, cross correlation, and autocorrelation.

11.3 Convolution, cross-correlation, and autocorrelation

There are three related operations that are encountered in signal processing. These are convolution, cross-correlation, and autocorrelation.

11.3.1 Convolution

The formal definition of convolution is given by the continuous integral relation

\[ F(t) \ast G(t) = \int_{-\infty}^{\infty} d\tau F(\tau)G(t-\tau). \] (11.3.1)

This is not a “definition” but a relation that arises naturally when solving boundary value problems using Green’s theorem. If we replace \( F \) and \( G \) by their inverse Fourier transform definitions, we obtain the result that

\[ F(t) \ast G(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\omega)g(\omega) \exp(-i\omega t)d\omega, \] (11.3.2)

which means that convolution is multiplication in the frequency domain.

11.3.2 Lab Activity #18: Frequency filtering

The simplest, yet one of the most important spectral methods is simple frequency filtering. If we look at the spectra of the traces in CMP gather 265

\`
$ suspecfx < gain.jon=1.cdp=265.su \\
| susxgraph title="spectra" \\
| label1="frequency" \\
| label2="amplitude" &
\`

we see that the data have most of their frequency spectral values between 5 Hz and 80 Hz. The rest can be considered to be noise, which could be boosted and distorted by further processing steps. When engaging in signal processing the old computer science adage ”garbage in garbage out” it is often amplified to ”small garbage in, a lot of garbage out.”

Applying simple frequency filtering with \textbf{sufilter} we see in the frequency domain

\`
$ sufilter < gain.jon=1.cdp=265.su \\
f=0,5,70,80 amps=0,1,1,0 | suspecfx \\
| susxgraph title="spectra" \\
| label1="frequency" \\
| label2="amplitude" &
\`
Figure 11.2: a) Amplitude spectra of the traces in CMP=265, b) Amplitude spectra after filtering.
that the data are truncated. Note that it is up to the user to figure out the full range of frequencies in the data that are to be kept. It may take some experimentation with further processing steps to find the correct filter range.

The program `sufilter` applies only simple tapered zero-phase filters to the data. There is another class of filter known as “Butterworth” filters, which in SU may be performed via the program `subfilt`. Butterworth filters are described by a solution a class of ordinary differential equation. Such filters were originally applied as analog preprocessing during the time of data acquisition.

Are we done with frequency filtering? Often, not. Other spectrum modifying processes, as well as other transforms we may use, may introduce noise into the output. We may need to apply our simple bandpass filter again owing to these noise sources.

### 11.3.3 Lab Activity #19: Spectral whitening of the fake data

It may have occurred to the reader after seeing the spectra of the real data that it may be of benefit if the spectra of the traces were flat, instead of having the many peaks and valleys. This is the process known as spectral whitening. We expect that such an operation would tend to sharpen data, but with the caveat that we know it will sharpen the noise, as well, making everything more “spike-like.”

The basic idea is simple. We take the data into the frequency domain, and consider the representation of the data $d(\omega)$ as a complex-valued function

$$d(\omega) = |d(\omega)|e^{i\phi(\omega)}.$$

We then multiply the amplitude $|d(\omega)|$ by a function $1/|d(\omega)|$, taking care not to divide by zero, so that $|d_{\text{new}}(\omega)| = 1$. This may be over the full range of 0 to the Nyquist frequency, or over some partial range. We then inverse Fourier transform the data. This operation is guaranteed to change the relationship between the amplitude and the phase function, as

$$d(\omega) = d_r + id_i.$$

where $d_r$ is the called the real part of $d$ and $d_i$ is the called the imaginary part of $d$. Here, the amplitude is given by the modulus of $d$ as

$$|d| = \sqrt{d_r^2 + d_i^2}$$

and the phase by

$$\phi = \arctan\left(\frac{d_i}{d_r}\right)$$

We can experiment with spectral whitening in SU using the command `suwfft`. This program gives the user the choice of whitening, from moderate to extreme. The plots labeled “traditional” use the default settings of the program, which are not really full spectrum whitening.

For example
Figure 11.3: a) Original fake data b) fake data with spectral whitening applied. Note that spectral whitening makes the random background noise bigger.
Another test is to apply spectral whitening to our CMP 265 data. Recall that frequency filtering may need to be applied before and after the whitening process. We see that the spectrum is an idealized white amplitude spectrum whose shape is the filter $\text{sufilter } f=0,5,80,90 < \text{gain.jon=1.cdp=265.su | suwfft w0=0 w1=1 w2=0 | suifft | sufitter f=0,5,80,90 | suspecfx | suxgraph title="Spectrum after whitening" &}$

and though it appears to be a single curve, these are really 55 identical spectra on top of the other, for each of the 55 traces in our gather.

The spectral whitening process involves the cascade of forward and inverse Fourier transforms. Owing to zero padding in these transforms, there may be more samples per trace on the output than on the input, so an extra step of windowing the data with $\text{suwind}$ in time is required $\text{sufilter f=0,5,80,90 < gain.jon=1.cdp=265.su | suwfft | suifft | sufitter f=0,5,80,90 | suwind itmin=1 itmax=1500 | suxwigb title="data after traditional spectral whitening" &}$

(Don’t forget the $\text{suiff}$ step!) The windowing passes samples from sample 1 through sample 1500 on each trace. As with our fake data, the real data have additional arrivals.

We can run $\text{Radon.test}$ on versions of the data after spectral whitening has been applied. Observe the changes in the semblance plots after spectral whitening. It may be that we would prefer to use spectral whitening after multiple suppression.

Should we run spectral whitening? The type of spectral whitening we discuss here is a brute force modification of amplitudes, which will certainly introduce noise into the output. Frequency filtering likely will be needed to remove frequency information that is totally fabricated by the spectral whitening process. We run spectral whitening (and spiking deconvolution) to improve resolution of velocity picks and to make reflectors sharper. To the end that the tools do that job, we may apply them. You definitely need to experiment with the operation to see if it helps.
11.4 The Discrete Representation of Seismic Data

In the past section, we discussed the application of filters that modify the spectrum of the input seismic signal. Though we are implementing the operations on sampled data, we use a continuous function representation of the processes that were being applied to formulate each technique.

We find, however, that the act of digitizing a signal introduces its own peculiarities.

11.4.1 The Forward and Inverse Z-transform

If we take the (causal) Fourier transform of our reflectivity series $R(t)$, we obtain a series in terms of shifted complex exponentials

$$\hat{R}(\omega) = \int_0^\infty R(t)e^{i\omega t} \, dt = \int_0^\infty \sum_{k=0}^N \mathcal{R}(t - \tau_k)e^{i\omega t} \, dt$$

$$= \sum_{k=0}^N \mathcal{R}_k e^{i\omega \tau_k} = \sum_{k=0}^N \mathcal{R}_k e^{i\omega k \Delta t}$$

where we note that the time sampling interval is a constant $\Delta t$ such that $\tau_k = k\Delta t$.

If we define $\phi = \Delta t$. The “Z” in Z-transform is this shifted complex exponential, $Z = \exp\{i\omega \Delta t\} = \exp\{i\omega \phi\}$. The ”transform” is the transformation of a this sampled data into a polynomial in $Z$. This polynomial is called the Z-transform representation of $R(t)$

$$R(Z) = \sum_{k=0}^N \mathcal{R}_k Z^k.$$ 

The advantage of this representation is that we have effectively taken the Fourier transform of our initial digitally sampled data by inspection!

All we have to do is to multiply the $k$-th term of our sequence digital value with $Z^k$ and add up the resulting terms to form the $k$-th order polynomial in $Z$. For a Z-transform representation of a finite number of terms this is all we need to know.

The Z-transform representation inherits the property that convolution and deconvolution of signals is represented as multiplication and division, respectively of the transformed data, as we see with Fourier transformed data.

More mathematics

For an infinite series, such as we might obtain through a Taylor expansion of a function or through a process of long division as in the case of the geometric series, we must also specify a region of convergence, which is a circle in the complex plane. Poles in the Z-transform representation will lie outside that circle of convergence. Zeros of the polynomial may lie inside the unit circle of convergence.
11.4.2 The inverse Z-transform

We can use the properties of the residue theorem from complex variables to get our original series back. The inverse Z-transform has to have the form

\[ R_k = \frac{1}{2\pi i} \int_C R(Z)Z^{k-1} dZ, \]  

for each term where the contour \( C \) encloses the origin.

We can see why this is so by considering that for \( Z = \exp\{i\omega \phi\} \) and for \( C \) being a circular contour enclosing the origin. Simply substituting for \( Z \) and noting that \( dZ = iZd\phi \)

\[ I = \int_C Z^{-1} dZ = i \int_0^{2\pi} e^{-i\omega \phi} e^{i\omega \phi} d\phi = 2\pi i, \]

which shows where the division by \( 2\pi i \) comes from.

The other possibility is to consider for \( n \geq 0 \)

\[ I = \int_C Z^n dZ = 0. \]

This integral vanishes by Cauchy’s theorem because the integrand is an analytic function of \( Z \). The contour integral in equation (11.4.1) sifts through the each term of the series representation of \( R(Z) \) and returns the original sequence of values as discrete values, giving us our original series of digital samples back.

The inverse Z-transform is effectively the inverse Fourier transform, as long as the contour \( C \) is the unit circle \( |Z| = 1 \).

11.5 Deconvolution

We have another way of whitening the spectrum. This method is to deconvolve the data. This may be either a deterministic process, where an estimate of the wavelet is obtained, and is divided out of the data in the frequency domain. More commonly we apply deconvolution via a statistical estimate of the wavelet, which is based on the assumption that the data are minimum phase (aka minimum delay), under an error minimization criterion. Thus we assume that our entire data consists of spikes convolved with minimum phase (aka minimum delay) wavelets.

Before launching into the application of minimum phase (aka minimum delay) deconvolution, we discuss the operations that we will be applying to data in general mathematical terms.

11.5.1 Convolution of a wavelet with a reflectivity series

The simplest model of a seismic trace is to consider the notion of the reflectivity series. The idea is simple. The world is assumed, to first order, to consist of simple reflectors, each with its own arrival time and its own reflection coefficient \( R_k \) for the \( k \)-th reflector.
Seismic waves are represented as rays that travel from the source to each reflector and back, taking $\tau_k$ for the two-way traveltime. In this ideal world, we have only single scattering, so there are no multiples (yet).

The simplest seismogram that could be recorded would then be a collection of spikes of each of a respective height $R_k$, having values which could be positive or negative, arriving at the respective time $\tau_k$,

$$R(t) = \sum_{k=1}^{N} R_k \delta(t - \tau_k)$$

where $R_k$ is the reflection coefficient of the $k$-reflector and $\delta(t - \tau_k)$ is the Dirac delta. We may think of this Dirac delta as a spike that only “turns on” when $t - \tau_k = 0$ and is “turned off” (zero) the rest of the time.

This series is called the reflectivity series—a popular notion in exploration seismology. If we want to make a seismogram, then we would convolve a wavelet $W(t)$ with the reflectivity series to form a seismogram. **Note that when we write the reflection coefficient of the $k$-th reflector, we write that as $R_k$, but when we write the $k$-th sample of the digitized reflectivity series $R(t)$, we write $R_k$.**

A remarkable result of digital signal processing is that the process of digitizing a signal yields the $Z$-transform of a function. Simply stated, the $Z$-transform of a signal is polynomial representation of the discretely sampled signal with the $k$-th sample multiplied by the factor $Z^k$.

**Thus, without taking expensive Fourier transforms, we are able to perform convolution and deconvolution of digital data by serial multiplication of digital representations, often making digital data processing in the time domain inexpensive.** This “serial multiplication” is the multiplication of the $Z$-transform polynomial representation of the given functions.

The division of polynomial representations would then be deconvolution. As with division in the Fourier domain, the issue of avoiding division by zero also critical in the $Z$-transform representation of deconvolution.

**Minimum phase in $Z$-transforms**

In the world of digital signal processing in geophysics, we are dealing with causal functions. This means that there is a specific beginning time for signals. We also have the issue of where the energy is in the signal.

We call waveforms that are “front loaded” or contain most of their energy at the beginning of the wavelet “minimum phase” (aka minimum delay) signals,

Such a $Z$ polynomial could be of some degree $m$ but it might be that only the first few terms of the $Z$ polynomial representing the wavelet are of importance, the rest could be near zero, which is to say that the low degree terms in the $Z$-transform contribute the most.
11.5.2 Convolution with a wavelet

Our digital data are convolved with a wavelet given by \( W(t) \)

\[
D(t) = W(t) \ast R(t)
\]

\[
= \int_{-\infty}^{\infty} W(\tau)R(t - \tau)d\tau
\]

\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} s(\omega)w(\omega)e^{-i\omega t}d\omega.
\]

Thus, recorded data \( D(t) \) is the convolution of a wavelet \( W(t) \) with the reflectivity series \( R(t) \). The last line shows the Fourier transform domain form of convolution. Convolution is multiplication in the frequency domain.

Convolution of Z-transform representations

In the language of Z-transforms, convolution of two signals is the multiplication of the two polynomial representations in Z of the functions.

11.5.3 Deconvolution

Deconvolution then is the inverse process, which is to say, the process of removing the effect of a waveform, to produce a desired output.

Symbolically, we have recorded data \( D(t) \) with a particular waveform \( W(t) \) possibly distorting the arrival time of a given reflection. What we want ideally is to reconstruct the reflectivity series by applying the inverse process

\[
R(t) = W^{-1}(t) \ast D(t).
\]

We need only determine what the “inverse of \( W(t) \)” given by \( W^{-1}(t) \) is. If we write this out in the Fourier domain representation, then we see that

\[
R(t) = W^{-1} \ast D(t)
\]

\[
= \int_{-\infty}^{\infty} W^{-1}(t_1)D(t - t_1)dt_1
\]

\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{D(\omega)}{W(\omega)}e^{-i\omega t}d\omega.
\]

Thus, we see that deconvolution is division in the frequency domain.
11.5.4 Deconvolution of functions represented by their Z-transforms

In terms of Z-transforms, we would then be dividing the polynomial representation of the signal by the polynomial representation of the wavelet. The zeros of the Z-transform polynomial become poles in the deconvolution result, which, if were were performing the inversion by contour integration would be the contribution to the contour integral.

11.5.5 Division in the frequency domain - Deterministic deconvolution

There is a problem, however, when we consider the Fourier transform as a spectrum. If the function is zero over a range of values in the frequency domain (as opposed to isolated zeros in the Z-transform) in the Fourier transform form of the wavelet given by the function \( w(\omega) \), then deconvolution is unstable or undefined. This follows because division by a small number introduces computational instability, and of course, division by zero is not defined.

We recall that \( w(\omega) \) is a complex valued function, which may be written in complex exponential form as
\[
w(\omega) = |w(\omega)| e^{i\omega \phi(\omega)}
\]
or as the sum of real and imaginary parts as
\[
w(\omega) = w_r(\omega) + iw_i(\omega).
\]

We define the complex conjugate of \( w(\omega) \) as
\[
\bar{w}(\omega) = |w(\omega)| e^{-i\omega \phi(\omega)}
\]
or as the sum of real and imaginary parts as
\[
\bar{w}(\omega) = w_r(\omega) - iw_i(\omega).
\]

If we multiply \( w(\omega) \) by its complex conjugate, we have the square of the modulus of \( w(\omega) \) as above
\[
|w(\omega)|^2 = w(\omega)\bar{w}(\omega).
\]

Returning to our deconvolution problem, multiplying top and bottom of the integrand by \( \bar{w}(\omega) \), we have
\[
R(t) = W^{-1} \ast D(t)
\]
\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \bar{w}(\omega) d(\omega) e^{-i\omega t} d\omega.
\]
We still haven’t solved the problems of division by zero in $w(\omega)$ because if $w(\omega)$ has a zero, then so will $|w(\omega)|$. We solve this problem by adding a small number $\varepsilon$ to the denominator

$$R(t) = W^{-1} * D(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\bar{w}(\omega)d(\omega)}{|w(\omega)|^2 + \varepsilon} e^{-i\omega t} d\omega.$$ 

The quantity $\varepsilon$ is the noise or whitening or white noise parameter. This parameter is chosen to be small enough to stabilize the inverse, but not so big as to skew the results. Thus, formally we can define the inverse waveform $W^{-1}(t)$ by its Fourier transform representation

$$W^{-1}(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\bar{w}(\omega)}{|w(\omega)|^2 + \varepsilon} e^{-i\omega t} d\omega.$$ 

It is important to remember that no matter how a deconvolutional process is performed, we think of deconvolution as division in the frequency domain. All deconvolution schemes must then have the equivalent of a white noise parameter to stabilize the division process, by preventing division by a small number or by zero.

**Deterministic deconvolution in SU - sucddecon**

In SU the program sucddecon performs deconvolution by a direct division in the frequency domain, given an input waveform as the sufile=filename.

```
$ sucddecon
```

**SUCDDECON - DECONvolution with user-supplied filter by straightforward Complex Division in the frequency domain**

sucddecon <stdin >stdout [optional parameters]

Required parameters:
filter= ascii filter values separated by commas

...or...

sufile= file containing SU traces to use as filter
(must have same number of traces as input data for panel=1)

Optional parameters:
panel=0 use only the first trace of sufile as filter
=1 decon trace by trace an entire gather
pnoise=0.001 white noise factor for stabilizing results
For example, if we use the far field air gun signature in Fig 11.1 as our input waveform we can apply `sucddecon` to deconvolve a panel of our data with this waveform. Here we deconvolve cdp 265

```
$ sucddecon sufile=farfield-gun.su < gain.jon=1.cdp=265.su |
  suxwigb title="deterministic deconvolution" key=offset perc=99 &
```

which looks rather bad. We can see that there have been frequencies manufactured by the filtering process, so applying a bandpass filter

```
$ sucddecon sufile=farfield-gun.su < gain.jon=1.cdp=265.su |
  sufilter f=5,10,70,80 |
  suxwigb title="deterministic deconvolution" key=offset &
```

we obtain a more acceptable result.

Note also, that the value of the `pnoise` parameter can make a big difference. In fact the default value is low. Don’t be afraid to try numbers like `pnoise=1` or `pnoise=10`.

```
$ sucddecon sufile=farfield-gun.su pnoise=10 < gain.jon=1.cdp=265.su |
  sufilter f=5,10,70,80 |
  suxwigb title="deterministic deconvolution" key=offset &
```

### 11.5.6 Signature deconvolution using homomorphic wavelet estimation

If we had a far field gun signature for each shot, or some other estimate of the wavelet, then we could apply deterministic deconvolution to each shot. Such a shot by shot deconvolution is called “signature deconvolution.”

There are a number of ways we could proceed to estimate the wavelet. Each method is based on the following assumptions:

- the waveform is minimum phase
- the wavelet is the only part of the data that does not change
- the reflectivity series is random.

An airgun signature is made to approximate a minimum phase (delay) waveform, so for ocean data with an airgun source this may not be a bad assumption. Even if we had a farfield airgun signature for each shot, this would not take into account the loss of higher frequencies due to anelastic attenuation.
Figure 11.4: Deterministic decon of CDP 265 using the farfield airgun signature estimate from Fig 11.1
11.6 Cross- and auto-correlation

A related mathematical operation to convolution is the cross correlation. The cross-
correlation of two functions is the multiplication of one function by the complex
conjugate of the other in the frequency domain. Here we represent the cross-correlation by the symbol “xcor,” which for digital data is the serial multiplication
of the discrete representations of \( A(t) \) and \( B(t) \). We write this as in the Fourier domain as

\[
A(t) \text{xcor} B(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} a(\omega)\overline{b(\omega)} e^{-i\omega t} d\omega.
\]

We can see that the auto-correlation is the product of a function with its own complex conjugate in the frequency domain.

\[
A(t) \text{xcor} A(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} a(\omega)\overline{a(\omega)} e^{-i\omega t} d\omega.
\]

Thus the frequency domain representation of the autocorrelation of our waveform is
given by the \(|w(\omega)|^2\), which appears the denominator of the frequency domain form of
the deconvolution, and in the Fourier transform representation of inverse wavelet \( W(t) \).

Whether deconvolution is performed in the time-domain, or in the frequency domain,
the common elements of the auto-correlation, the \( \varepsilon \) noise or whitening parameter,
and the wavelet \( W(t) \) are present.

11.6.1 Z-transform view of cross-correlation

Given the Z-transform representations of two signals \( B(Z) \) and \( A(Z) \)

\[
A(Z) = \sum_{k=1}^{N} a_k Z^k \quad \text{and} \quad B(Z) = \sum_{l=1}^{N} b_l Z^l
\]

we represent the complex conjugate \( \overline{B(Z)} \) as the same series, but with terms represented
by the negative powers of \( Z \)

\[
\overline{B(Z)} = \sum_{l=1}^{N} b_l Z^{-l}.
\]

The cross-correlation of \( A(Z) \) and \( B(Z) \) is then the product of the polynomials

\[
A(Z)\overline{B(Z)} = \left( \sum_{k=1}^{N} a_k Z^k \right) \left( \sum_{l=1}^{N} b_l Z^{-l} \right).
\]

The effect is to flip the order of the \( B(z) \) series in the multiplication to the opposite order as would be done with convolution.
11.6.2 Cross correlation and auto correlation in SU suxcor and suacor

$ suxcor

SUXCOR - correlation with user-supplied filter

suxcor <stdin >stdout filter= [optional parameters]

Required parameters: ONE of
sufile= file containing SU traces to use as filter
filter= user-supplied correlation filter (ascii)

Optional parameters:
vibroseis=0 =nsout for correlating vibroseis data
first=1 supplied trace is default first element of
correlation. =0 for it to be second.
panel=0 use only the first trace of sufile as filter
 =1 xcor trace by trace an entire gather
ftwin=0 first sample on the first trace of the window
 (only with panel=1)
lwin=0 first sample on the last trace of the window
 (only with panel=1)
ntwin=nt number of samples in the correlation window
 (only with panel=1)
ntrc=48 number of traces on a gather

...

$ suacor

SUACOR - auto-correlation

suacor <stdin >stdout [optional parms]

Optional Parameters:
ntout=101 odd number of time samples output
norm=1 if non-zero, normalize maximum absolute output to 1
sym=1 if non-zero, produce a symmetric output from
 lag -(ntout-1)/2 to lag +(ntout-1)/2
11.7 Lab activity #20: Wiener (least-squares) filtering

There is a class of deconvolutional processes known as Wiener filters or prediction error filters, which have been found to be useful in exploration seismic methods. The method is called “predictive” because it assumes that the data have a specific character that allow later parts of the data to be predicted from earlier parts of the data.

Wiener filtering assumes that the data are minimum phase (aka minimum delay). While there is a requirement that the spectrum of the data is white, a small “noise” parameter is added or assumed in the algorithm to prevent division by zero. Physically, if a waveform is minimum phase (aka minimum delay) its energy is located in the front part of the waveform.

11.7.1 A matrix view of the convolution model

The convolutional model of seismic waves holds that the data $D(t)$ are formed by the convolution of a wavelet $W(t)$ with a reflectivity series $R(t)$. Symbolically this is represented as

$$D(t) = W(t) * R(t), \quad (11.7.1)$$

where

$$R(t) = \sum_{k=0}^{N} R_k \delta(t - \tau_k).$$

Here $\delta(t - \tau_k)$ is the Dirac delta function, which turns on only at time $\tau_k$, the two way traveltime to the $k$-th reflector, and $R_k$ is the reflection coefficient (either positive or negative) of the $k$-th reflector.

As an integral, the convolution of the reflectivity series $R(t)$ with the wavelet $W(t)$ is defined as

$$D(t) = W(t) * R(t) = \int_{-\infty}^{\infty} W(t - \tau) R(\tau) d\tau. \quad (11.7.2)$$

The discrete version of this operation can be written as

$$D_n = \sum_{k=0}^{N} W_{n-k} R_k.$$

Note how the integration variables in the continuous version correspond to the indexes in the discrete version.

We can write this numerically as the matrix multiplication

$$WR = D \quad (11.7.2)$$

where $W$ is a band matrix, whose rows are composed of shifted versions of a discrete representation of the wavelet $W(t) = \{w_0, w_1, w_2, \ldots\}$, the reflectivity series $R(t) = \{r_0, r_1, \ldots\}$. 
and the data $D(t) = \{d_0, d_1, \ldots\}$.

\[
\begin{bmatrix}
  w_0 & w_1 & w_2 & \ldots & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & w_0 & w_1 & w_2 & \ldots & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & w_0 & w_1 & w_2 & \ldots & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & w_0 & w_1 & w_2 & \ldots & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & \ldots & \ldots & \ldots & \ldots & 0 & 0 \\
  \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & w_0 & w_1 & w_2 & \ldots \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
\end{bmatrix}
\begin{bmatrix}
  r_0 \\
  r_1 \\
  r_3 \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  r_n
\end{bmatrix} =
\begin{bmatrix}
  d_0 \\
  d_1 \\
  d_3 \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
  d_m
\end{bmatrix}.
\] (11.7.3)

We want to solve for $R$, but $W$ is, in general, a non-square ($m \times n$) matrix. Our solution is the pseudoinverse or least-squares solution. We multiply by the transpose of $W$, which we write as $W^T$, we obtain

\[W^T W R = W^T D\] (11.7.4)

where ideally the solution for $D$ is given by taking the inverse of $W^T W$, yielding

\[R = (W^T W)^{-1} W^T D.\] (11.7.5)

For stability $\varepsilon I$ is added to the $W^T W$ to yield the final form

\[R = (W^T W + \varepsilon I)^{-1} W^T D.\] (11.7.6)

As written, this expression describes a mathematician’s view of the problem, but this form is not really practical to implement.

If we go back a step

\[W^T W R = W^T D\] (11.7.7)

The right hand side of the contains $W^T D$ which is the cross-correlation of the wavelet $W$ with the recorded data $D$. This is a spiked version of the data. The quantity $W^T W$ is the matrix of autocorrelations of the wavelet. The quantity $W^T D$ of bandlimited spikes, whose heights are proportional to the reflection coefficients. If the original data have no multiples, then the cross-correlation of the wavelet with the data will be a band-limited “spiked” version of the reflectivity series and would be a form of “spiking deconvolution”. Again, we are less satisfied with this because we do not have the wavelet $W$ needed so we can compute $W^T D$.

The left hand side of the expression contains $W^T W$, which is a matrix whose rows are composed of the autocorrelation of the wavelet, shifted successively by one sample.
on each row

\[
W^T W = \begin{bmatrix}
\phi_0 & \phi_1 & \phi_2 & \ldots & \phi_{n-1} \\
\phi_{-1} & \phi_0 & \phi_1 & \ldots & \phi_{n-2} \\
\phi_{-2} & \phi_{-1} & \phi_0 & \ldots & \phi_{n-3} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\phi_{n-1} & \ldots & \ldots & \phi_{-1} & \phi_0
\end{bmatrix}.
\] (11.7.8)

Here, \( \phi_{-(n-1)}, \ldots, \phi_{-1}, \phi_0, \phi_1, \ldots, \phi_{n-1} \) is the autocorrelation of the wavelet \( W(t) \). We note that this is symmetric such that \( \phi_{-k} = \phi_k \). If the reflectivity series is random, such that the later values of of \( R(t) \) cannot be predicted from earlier values, then the autocorrelation of the data \( D(t) \) is approximately the same as the autocorrelation of the wavelet \( W(t) \).

At most, these discussions tell us about the problem of deconvolution, without giving us a practical solution to implement. We must look further.

### 11.7.2 Designing wavelet shaping filters – Wiener filtering

If the reflectivity series \( R \) is random, then the autocorrelation of the recorded data \( D \) is a good approximation to the autocorrelation of the wavelet \( W \), because at most the autocorrelation of a random sequence is a constant value, as matrices \( R^T R \) is the identity matrix \( I \).

We don’t want the entire autocorrelation, only the autocorrelation waveform, that is, the sinc-function like part that is in the middle of the autocorrelation output. We will call this waveform \( \Phi(t) \).

Suppose that we want to create a filter that will take data with given input wavelet \( W(t) \), and yield a desired output wavelet \( V(t) \). Then we want to design a filter \( F(t) \) such that the application of the filter to \( W(t) \) yields \( V(t) \),

\[
W(t) \ast F(t) = V(t).
\] (11.7.9)

Writing this convolution in the discrete representation, we have

\[
\sum_{k=0}^{n-1} w_{m-k} f_k = v_m.
\] (11.7.10)

Likely we cannot solve this problem exactly, so there will be an error vector \( E \) obtained by subtracting the right hand side from the left hand side

\[
E_m = \left( \sum_{k=0}^{n-1} w_{m-k} f_k - v_m \right).
\] (11.7.11)

### 11.7.3 Least-squares (Wiener) filter design

An approach developed independently by N. Wiener and A. N. Kolmogorov in the early 1940s is to apply least-squares optimization to the problem of designing the wavelet shaping filter. The filters that result from this approach are often called Wiener filters. Because noise is always present we cannot solve the system exactly.
The square of the error is given by

\[ E^2 = \sum_{j=0}^{m} \left( \sum_{k=0}^{n-1} w_{j-k} f_k - v_j \right) \left( \sum_{k=0}^{n-1} w_{j-k} f_k - v_j \right). \]  \hspace{1cm} (11.7.12)

The extra summation is required because the error is a vector, but the square of the error is taken as the dot product of the error vector with itself, which is a scalar. We much prefer dealing with a scalar quantity than a vector quantity in this context.

Minimization means taking the derivative and setting the result to zero. Which derivative? The thing that is varying is the filter, so we should be differentiating with respect to the filter values

\[ \frac{\partial}{\partial f_p} E^2 = \frac{\partial}{\partial f_p} \sum_{j=0}^{m} \left( \sum_{k=0}^{n-1} w_{j-k} f_k - v_j \right) \left( \sum_{k=0}^{n-1} w_{j-k} f_k - v_j \right). \]  \hspace{1cm} (11.7.13)

Applying the chain rule, and setting \( E^2 = 0 \)

\[ 0 = \sum_{j=0}^{m} \left( 2 \sum_{k=0}^{n-1} w_{j-k} \frac{\partial f_k}{\partial f_p} \left( \sum_{k=0}^{n-1} w_{m-k} f_k - v_m \right) \right). \]  \hspace{1cm} (11.7.14)

We note that \( \partial f_k / \partial f_p = 1 \) when \( k = p \) and is zero when \( k \neq p \), hence we have (canceling the factor of 2)

\[ 0 = \sum_{j=0}^{m} w_{j-p} \left( \sum_{k=0}^{n-1} w_{m-k} f_k - v_m \right). \]  \hspace{1cm} (11.7.15)

which may be rewritten as

\[ \sum_{j=0}^{m} w_{j-p} \sum_{k=0}^{n-1} w_{m-k} f_k = \sum_{j=0}^{m} w_{j-p} v_m. \]  \hspace{1cm} (11.7.16)

In matrix notation this is

\[ W^T W F = W^T V. \]  \hspace{1cm} (11.7.17)

The matrix \( W^T W \) is the matrix of autocorrelations of the wavelet, as before, and the right hand side is the crosscorrelation of the wavelet, with the desired output \( V(t) \). The matrix of autocorrelations is in a form called a Toeplitz matrix. There is a recursive method of solution of Toeplitz systems pioneered by N. Levinson in 1947, and improved by Durbin in 1960, and others since that time. The recursive method allows for the filter \( F \) to be solved for directly. This makes what might seem like a difficult process rather simple.

Several programs in the SU package use this method for performing spiking deconvolution, predictive or gapped-deconvolution, and wavelet shaping using these facts.
11.8 Spiking deconvolution

Suppose that we want to do spiking deconvolution. If the data were already spiked, then the autocorrelation would be a delta function at sample number zero, and the autocorrelation vector would consist of a single nonzero value \((\phi_0, 0, 0, \ldots)\). The value of \(\phi_0\) might be normalized to 1, or not.

Our issue is to solve the problem

\[
\begin{bmatrix}
\phi_0 & \phi_1 & \phi_2 & \ldots & \phi_{n-2} \\
\phi_{-1} & \phi_0 & \phi_1 & \ldots & \phi_{n-3} \\
\phi_{-2} & \phi_{-1} & \phi_0 & \ldots & \phi_{n-4} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\phi_{n-2} & \ldots & \ldots & \ldots & \phi_{-1} & \phi_0
\end{bmatrix}
\begin{bmatrix}
f_0 \\
f_1 \\
\vdots \\
f_n-2
\end{bmatrix}
= 
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\vdots \\
\phi_{n-1}
\end{bmatrix}.
\] (11.8.1)

Notice that the vector on the right beings with \(\phi_1\) not \(\phi_0\).

Thus we want to solve the Toeplitz system for the filter \(f_0, f_1, \ldots, f_{n-2}\). This allows us to compute the values of the autocorrelation after the first value. We then want to subtract this from our data.

Actually filtering and then subtracting is unnecessary. We can build the subtraction into the filter by writing \(1 - F = (1, -f_0, -f_1, \ldots, -f_{n-1})\). The only thing that we need to know is how many points long the filter should be. This is the \textit{maximum lag} of the spiking filter. \textbf{We take the length of the autocorrelation waveform as the value of the maximum lag for spiking deconvolution.}

The reader should be aware that this algorithm is somewhat insensitive to choice of the number of samples in the autocorrelation. We do not need to worry about picking an exact value, as there are a number of values that will work more or less equally well.

\textbf{What does “lag” mean?}

If a time delay is in seconds, or other time units, we call that difference in time simply a \textit{delay}. If we express a time delay in samples, then it is the convention to call that a \textit{lag}. To make matters more confusing, in the SU programs that have “lags” those lags are expressed in seconds.

11.8.1 Spiking Deconvolution in SU

In SU the program \texttt{supef} may be used to perform spiking deconvolution

\$ \texttt{supef}\$

\texttt{SUPEF - Wiener predictive error filtering}\n
\texttt{supef <stdin >stdout [optional parameters]}\n
Required parameters:
\texttt{dt} is mandatory if not set in header
Figure 11.5: a) Autocorrelation waveforms of the `fake.su` data b) Autocorrelation waveforms of the same data after predictive (spiking) decon.
Optional parameters:

- **cdp=** CDPs for which minlag, maxlag, pnoise, mincorr, maxcorr are set (see Notes)
- **minlag=dt** first lag of prediction filter (sec)
- **maxlag=last** lag default is \((t_{\max}-t_{\min})/20\)
- **pnoise=0.001** relative additive noise level

Note also, that the value of the `pnoise=` parameter can make a big difference in the output.

If our data consist of a wavelet \(W\) convolved with the reflectivity series \(R\), further convolved with multiples \(M\), then our model in the previous section is not quite right. The autocorrelation will contain repetitions due to the multiples.

If the wavelet is minimum phase (aka minimum delay), then most of the energy will be located at the beginning of the waveform, and the autocorrelation of the data will produce a sinc-like autocorrelation waveform that is localized to the values near the center of the output of the autocorrelation. This is the diagonal region of the autocorrelation represented as a matrix.

If we select this window in the autocorrelation for processing, then spiking decon filter will be generated approximately correctly.

In `supef` the value of `maxlag =` is set to the width of the autocorrelation waveform, which we must determine by taking the autocorrelation using `suacor`. (Remember that a “lag” is just a time delay.)

For example, consider the `fake.su` data

```bash
$ suacor < fake.su ntout=101 | suxwigb perc=90.
```

Another possibility is to stack the autocorrelations

```bash
$ suacor < fake.su ntout=101
| sustack key=dt | suxgraph style=normal f1=-.2
```

which yields a sinc-like waveform.

The choice of `key=dt` was to ensure that the traces were all stacked with respect to a header field that does not change in the gather and the choice of `f1=-.2` is so that the peak of the autocorrelation is at zero lag. The choice of `ntout=101` means that we want 101 samples on the resulting traces. This number is chosen to be sufficiently large to capture the side lobes of the wavelet that appears in the center of each of the resulting traces.

This waveform is the *autocorrelation waveform*. For data that are dominated by spikes, or are spectrally white, the autocorrelation waveform would also be a spike.

We pick the width of the autocorrelation waveform. In the case of our example, this is between approximately 0.0667 and 0.1340 seconds, making the width of the autocorrelation waveform approximately .0673 seconds. We apply `supef` setting this value as the value of `maxlag`
The data are made more spike-like by the operation. Try different values of \texttt{maxlag=} to see what the effect of changing this parameter is.

Also you might want to view autocorrelation waveform of the deconvolved data to see what happens

\begin{verbatim}
$ supef < fake.su maxlag=.0673 | suxwigb perc=99
\end{verbatim}

\begin{verbatim}
$ supef < fake.su maxlag=.0673
    | suacor ntout=101 | suxwigb perc=90.
\end{verbatim}

or

\begin{verbatim}
$ supef < fake.su maxlag=.0673
    | suacor ntout=101 | sustack key=dt | suxgraph style=normal
\end{verbatim}

The effect

We apply the same operations on CDP 265

\begin{verbatim}
$ suacor ntout=101 < gain.jon=1.cdp=265.su
    | suxwigb perc=99
\end{verbatim}

The autocorrelation waveform is a sinc-like function. We define the “width” of this waveform to be the window of time just large enough to include the side lobes on each side of the main lobe. If we measure the time from the beginning to the end of the autocorrelation waveform, which is to say about .169 seconds to about .247 seconds see that the width is about .078 seconds. Your values may differ.

This is the value of \texttt{maxlag} that we will set in \texttt{supef}

\begin{verbatim}
$ supef < gain.jon=1.cdp=265.su maxlag=.078 | suxwigb xcur=3
\end{verbatim}

To see how well this has spiked the data, we may view the autocorrelation waveform with \texttt{suacor}

\begin{verbatim}
$ supef < gain.jon=1.cdp=265.su maxlag=.078
    | suacor ntout=101 | suxwigb perc=90
\end{verbatim}

which should show that the autocorrelation waveform is now a spike. Again, we may vary the value of \texttt{maxlag=} to see the effect of changing this parameter.

\subsection{11.8.2 Multiple suppression by Wiener filtering—Gapped prediction error filtering.}

We now seek to eliminate multiples by \textit{prediction error filtering} also known as \textit{predictive deconvolution}. Predictive decon relies on the minimum phase (aka minimum delay) assumption and the notion that the data contain repetitions owing to the series of multiples $M$. 
One of the reasons that multiples are damaging to processing and hard to eliminate is that multiples are not merely added to the data, they are \textit{convolved} with the reflectivity series \( R \), which is in turn convolved with the wavelet \( W \)

\[
D = W \ast M \ast R \\
D_n = \sum_{n=0}^{N} M_{n-m} \sum_{k=0}^{K} W_{l-k} R_k, \quad (11.8.2)
\]

which we could write as a cascaded matrix multiplications on to the vector \( R \)

\[
D = MW R \quad (11.8.3)
\]

where both \( M \) and \( W \) matrices composed of shifted versions of the \( M \) and \( W \) series, respectively. Not also, that there is a noise vector \( N \) that is added, as well

\[
D = W \ast M \ast R + N \quad (11.8.4)
\]

which we have been quietly ignoring.

If we form the autocorrelation of the data \( D \), the result will be the same as the autocorrelation of the series of reverberations \( M \) convolved with the wavelet \( W \). The reflectivity series \( R \) is considered to be random because later values of reflectivity may not, in general, be predicted from earlier values. (This ignores the possibility of transgressive and regressive sequences in geology, which may be repetetive.) We may also consider the noise \( N \) to be composed of the sum of a random and non-random part.

The autocorrelation of the reflectivity series will be assumed to not contribute to the autocorrelation of the data. We call the matrix of shifted autocorrelations \( \Phi \) which will be approximately

\[
\Phi = (MW R)^T MW R \\
\Phi = R^T W^T M^T MW R \\
\Phi \approx W^T M^T MW \quad (11.8.5)
\]

where

\[
\Phi = \begin{bmatrix}
\phi_0 & \phi_1 & \phi_2 & \ldots & \phi_{n-1} \\
\phi_{-1} & \phi_0 & \phi_1 & \ldots & \phi_{n-2} \\
\phi_{-2} & \phi_{-1} & \phi_0 & \ldots & \phi_{n-3} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\phi_{-(n-1)} & \ldots & \phi_{-1} & \phi_0
\end{bmatrix} \quad (11.8.6)
\]

Here each row is a shifted version of the autocorrelation of the full data \( D \) which is the symmetric waveform \( \phi_{-(n-1)}, \phi_{-(n-2)}, \phi_{-(n-3)}, \ldots, \phi_{-3}, \phi_{-2}, \phi_{-1}, \phi_0, \phi_1, \phi_2, \phi_3, \ldots, \phi_{(n-3)}, \phi_{(n-2)}, \phi_{(n-1)} \).

The character of this autocorrelation will be as follows. If the wavelet \( W \) is minimum phase (aka minimum delay), the portion in the middle about the value \( \phi_0 \) will be a sinc-like waveform consisting of a main lobe centered at \( \phi_0 \), symmetric with just a couple
of side lobes, allowing us to estimate the autocorrelation of the wavelet $W$—hence our use of the width of the autocorrelation waveform for $\text{maxlag}=\text{in supef}$ for spiking deconvolution.

The autocorrelation will contain repetitions owing to the autocorrelation of the multiples $M$ with the reflectivity series. That repetition time will be related to the two-way traveltime in the water column. If we apply spiking decon to our data, the autocorrelation waveform will be a spike, and what remains will be repeating spikes.

Our autocorrelation will start repeating at some sample $k$, so we are going to predict our reverberations, and then subtract them off. We are solving the problem of finding the filter that will predict and then eliminate the reverberations, which is to find $F$ given

$$
\begin{bmatrix}
\phi_0 & \phi_1 & \phi_2 & \ldots & \phi_{n-1} \\
\phi_{-1} & \phi_0 & \phi_1 & \ldots & \phi_{n-2} \\
\phi_{-2} & \phi_{-1} & \phi_0 & \ldots & \phi_{n-3} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\phi_{-(n-1)} & \ldots & \phi_{-1} & \phi_0
\end{bmatrix}
\begin{bmatrix}
f_0 \\
f_1 \\
f_3 \\
\vdots \\
f_{n-1}
\end{bmatrix}
= 
\begin{bmatrix}
\phi_k \\
\phi_{k+1} \\
\phi_{m+2} \\
\phi_{m+3} \\
\phi_{(k+n-1)}
\end{bmatrix}.
\tag{11.8.7}
$$

Thus, we are finding a Wiener filter $F$ that will predict the repetitions in the autocorrelation using the earlier values of the autocorrelation. Hence the notion of “predictive” decon. The repetitions start at sample $k$, so this is the delay of the filter. We don’t want the repetitions, so the filter we apply to the data is formed by subtracting our filter (delayed by $k$ samples).

In our notation, the prediction error filter is given by $1-F = \{1, 0, 0, \ldots, 0, -f_0, -f_1, \ldots, -f_{\text{max}}\}$. Here there are $k-1$ zeros, which is the “gap” in the gapped decon. The “prediction error” is the difference between the data and the predicted value from the Weiner filter.

### 11.8.3 Applying gapped decon in SU – supef

By selecting the appropriate combination of $\text{minlag}=\text{and maxlag}=\text{defining the Wiener filter, we can eliminate repetitions in the data, such as those caused by multiples. This is known as gapped predictive decon in the parlance of the geophysical community.}$

We begin by spiking our $\text{fake+water+pegleg.su}$ which are our data with water-bottom and pegleg multiples

```
$ \text{supef < fake+water+pegleg.su maxlag=.0673 | suxwigb perc=99 xcur=2}$
```

We then view the autocorrelation of the data in a broader window choosing $\text{ntout}=1024$ samples in the output. The idea is to look for repetitions in the autocorrelation

```
$ \text{supef < fake+water+pegleg.su maxlag=.0673 | suacor ntout=1024 | suxwigb perc=90}$
```
What we are looking for are repetitions in the autocorrelation. We know that the two-way traveltime for the water speed is about .5 s, and we see stripes that are at about .51 s above and below the autocorrelation waveform spike. Also we notice that there is an offset effect. Thus, we apply a moveout correction to flatten the data

```
$ sunmo vnmo=1500 smute=20 < fake+water+pegleg.su | supef maxlag=.0673 |
  suacor ntout=1024 | suxwigb perc=90
```

where we have used \textit{smute} = 20 in \textit{sunmo} to turn off the stretch mute. Notice that the result is sensitive to the value of \textit{vnmo}. It might be that making \textit{vnmo} = slightly bigger gives a slightly flatter collection of spikes

```
$ sunmo vnmo=1800 smute=20 < fake+water+pegleg.su | supef maxlag=.0673 |
  suacor ntout=1024 | suxwigb perc=90
```

The repetition time of the signal is the value that is needed to define the “gap” in the gapped decon. In this case the gap is .51 seconds. This value is our choice for \textit{minlag}. The \textit{maxlag} will be the value of \textit{maxlag} we used for spiking the data added to the value of the gap, \textit{maxlag} = \textit{maxlag} for spiking + gap.

Finally, we finish by doing inverse NMO

```
$ sunmo vnmo=1800 smute=20 < fake+water+pegleg.su |
  supef maxlag=.0673 |
  supef minlag=.51 maxlag=.5773 |
  sunmo invert=1 vnmo=1800 smute=20 | suxwigb perc=99 xcur=2
```

Here we have performed \textit{spiking decon} and have followed this with a \textit{gapped decon}. It may be better to do the gapped decon only, which would be done via

```
$ sunmo vnmo=1800 smute=20 < fake+water+pegleg.su |
  supef minlag=.51 maxlag=.5773 |
  sunmo invert=1 vnmo=1800 smute=20 | suxwigb perc=99 xcur=2
```

where we note that the \textit{maxlag} = .51 + .0673 is chosen as if we had performed spiking decon. Again we have used \textit{smute} = 20 to turn off the stretch mute in \textit{sunmo}. The value of \textit{minlag} must not exceed the value of the actual reverberation time, but it may be less. The value of \textit{maxlag} again is not so sensitive.

Again the value of \textit{pnoise} may be adjusted to improve the result.

We may view effect on the multiples by comparing semblance panels

```
$ suvelan < fake+water+pegleg.su nv=150 dv=15 fv=1450 |
  suximage d2=15 f2=1450 cmap=hsv2 bclip=.5 title="cdp 265" &
$ suvelan < pef.fake+water+pegleg.su nv=150 dv=15 fv=1450 |
  suximage d2=15 f2=1450 cmap=hsv2 bclip=.3 title="PEF" &
```
The multiples with speeds near the water speed have been suppressed, as have some of the multiples from the strong reflector near 2 sec and 2000 m/s. However, this is not as clean as the radon transform filtered data.

We may repeat the process to eliminate other repetitions in the data, such as those from pegleg multiples. As with radon domain filtering, we choose appropriate `nmo` velocities to flatten the arrivals we choose to remove. You may want to try repeating the last several steps using the data `fake+water+pegleg.su`.

11.9 What (else) did predictive decon do to our data?

The fact that we are applying an inverse filter to our data means that in some sense we are making the output look “more like a bunch of spikes” or “more like a bunch of Dirac delta functions”. Because we know that a spike contains all frequencies, the term `spectral whitening` is applied to describe the effect of such filters in the frequency domain. This bug/feature may be observed in your data by comparing the amplitude spectra

\[
s\text{spectcfx} < \text{fake+water.su} | \text{suximage title} = "\text{data before spiking decon}"
\]
\[
s\text{spectcfx} < \text{pef.fake+water.su} | \text{suximage title} = "\text{data after spiking decon}"
\]

On one hand, it may seem that the increased frequency content is a good thing. However, can we really trust that those frequencies have been correctly added to the data? These may be simply an artifact of the filter that causes more harm than good. Some spectral whitening is desirable, but most should probably be suppressed by filtering. For example we might consider simply applying a filter to the data as part of the processing

\[
\ldots | su\text{filter } f=0,2,60,70 | \ldots
\]

where the values of the corner frequencies of the filter are chosen to reflect a reasonable range of frequencies in the data that can be trusted. So finally, the processing sequence for our fake data with waterbottom multiples is

\[
\ldots | su\text{filter } f=0,2,60,70 | \ldots
\]

or, if we seek to do multiple-suppression only, without spiking decon

\[
\ldots | su\text{filter } f=0,2,60,70 | \ldots
\]

For the real data, some variation on this processing flow, in terms of the values of `minlag=` and `maxlag=` will exist. Indeed, these values are guaranteed to vary some across the survey.
### 11.9.1 Deconvolution in the Radon domain

Another possibility is to apply the prediction error filtering in the radon domain. For example, employing the linear $\tau - p$ transform we forward radon transform the data, apply the prediction error filtering:

```
summo vnmo=1500 smute=20 < gain.jon=1.cdp=265.su |
suradon choose=0 igopt=3 pmin=-1500 pmax=1000 interoff=-262 offref=-3237 |
sudef minlag=.15 maxlag=1.0 |
suradon choose=4 igopt=3 pmin=-1500 pmax=1000 interoff=-262 offref=-3237 |
summo vnmo=1500 invert=1 smute=20 > radonpef.su
```

The process will do a good job on simple water-bottom reverberations, but other multiples will not be as well suppressed, unless these can be made exactly periodic in the radon domain.

### 11.10 FX Decon

There is application of prediction error filtering in the frequency domain, called “fx decon" that was created in the 1984 by L.L. Canales. This technique uses predictive decon in the space-frequency domain to identify and eliminate random noise.

For example, consider the spectrally whitened version of `fake.su`:

```
$ suwfft w0=0 w1=1 w2=0 < fake.su | suiff > white.fake.su
```

Applying `sufxdecon` to these data:

```
$ sufxfdecon < white.fake.su | suxwigb perc=99
```

Try this operation on different versions of CDP 265.

It may be best to reserve ‘FX decon” for the later stages of processing, after the stack.

### 11.11 Lab Activity #20: Wavelet shaping

Many papers written in the 1970s dealt with the issue of wavelet estimation. That is, using statistical methods to determine the shape of the average wavelet throughout the dataset, or in regions in a dataset. The motivation for this is to use deconvolution to change the waveforms of the data to a new desired output waveform.

Currently in SU, there is no sophisticated wavelet estimation code as yet. The user can get a crude estimate of the wavelet by selecting the waveform from a horizontal portion of a reflector in the data. Knowing the trace number and the time window of the wavelet, we may use suwind to capture this “average wavelet” via:

```
suwind key=tracl min=TRACE max=TRACE tmin=TMIN tmax=TMAX > wavelet.su
```
where TRACE, TMIN, and TMAX are replaced with the actual values of the trace number, and minimum and maximum times that where the wavelet of choice is located.

Also, we can make a desired output waveform by using the program **suwaveform**

**SUWAVEFORM** - generate a seismic wavelet

```
suwaveform <stdin >stdout [optional parameters]
```

Required parameters:
- one of the optional parameters listed below

Optional parameters:
- `type=akb` wavelet type
  - `akb`: AKB wavelet defined by max frequency `fpeak`
  - `berlage`: Berlage wavelet
  - `gauss`: Gaussian wavelet defined by frequency `fpeak`
  - `gaussd`: Gaussian first derivative wavelet
  - `ricker1`: Ricker wavelet defined by frequency `fpeak`
  - `ricker2`: Ricker wavelet defined by half and period
  - `spike`: spike wavelet, shifted by time `tspike`
  - `unit`: unit wavelet, i.e. amplitude = 1 = const.

- `dt=0.004` time sampling interval in seconds
- `ns=` if set, number of samples in output trace
- `fpeak=20.0` peak frequency of a Berlage, Ricker, or Gaussian,

For example

```
$ suwaveform > dfile.su type=ricker1 fpeak=15
```

where **dfile.su** contains a Ricker wavelet with peak frequency `fpeak` of 15 Hz. The frequency content of the desired output waveform should approximately match the frequency content of the input wavelet.

Given the wavelet (**wavelet.su**) and the desired output waveform (**dfile.su**) we may use the wavelet shaping code, called **sushape**

**SUSHAPE** - Wiener shaping filter

```
sushape <stdin >stdout [optional parameters]
```

Required parameters:
- `w=` vector of input wavelet to be shaped or ...
- ... or ...

206
wfile= ... file containing input wavelet in SU (SEGY trace) format
d= vector of desired output wavelet or ...
...or ...
dfile= ... file containing desired output wavelet in SU format
dt=tr.dt if tr.dt is not set in header, then dt is mandatory

Optional parameters:
nshape=trace length of shaping filter
pnoise=0.001 relative additive noise level
showshaper=0 =1 to show shaping filter

For example, our waveforms may be shaped via:

```
$ sushape dfile=dfile.su wfile=wavelet.su < data.su > shaped_data.su
```

The shaping filter works by effectively by performing the operation of deconvolving the data to remove `wavelet.su` and the convolution of the resulting “spiked” data by the desired output waveform `dfile.su`. The `sushape` program makes use of Wiener-Levinson theory to perform this operation in the time domain.

### Finding the wavelet and making target waveforms

A great deal of work has been put into “wavelet estimation” techniques in the exploration seismic community. Ideally we should know the wavelet for each shot, and even the wavelet as a function of angle from the shot. Here, we assume for simplicity that waveforms chosen carefully off of the data, using `suwind` are sufficient for our purposes.

To construct the target waveform `dfile.su` we may use one of the wavelets generated from the program `suwaveform` with either `type=ricker1` or `type=ricker2` being the best choices, although the Berlage waveform is not a bad choice, either. When constructing a target waveform, make sure that the frequency content of the desired output waveform is roughly the same as that of the data, so that values not be “manufactured” by the program.

### 11.12 Filling in missing shots

We have noted that in the Viking Graben Dataset, there are a number of missing shots. These may be identified by viewing a ”shooting chart” via:

```
$ suchart < seismic.su key1=sx key2=gx | xgraph n=120120 linewidth=0
label1="sx" label2="gx" marksize=2 mark=8 &
```
By zooming in on the plot, we can see that there are gaps in the data between shots located between \( sx = 5187 \) and 5262 meters, between 5387 and 5487 meters, between 14412 and 14512 meters, and between 22162 and 22262 meters. Because there is a 25 meter spacing between successive shot positions this means that shots at the following locations are missing: 5212, 5237, 5412, 5437, 5462, 14437, 14462, 14487, 22187, 22212, and 22237 meters.

Missing data is big issue in exploration seismology because holes in our data are a problem for processing algorithms, such as most migration routines, that expect that data are uniformly sampled and are “complete.”

One method that can be employed is to simply replace missing shots with nearest neighbor shot gathers, or by the average of nearest neighbor shot gathers. For example we could capture and average two nearest neighbor shot gathers by windowing the data

```bash
# capture 5187 and 5262
suwind key=sx min=5187 max=5262 < seismic.su > junk1.su
```

and then by sorting into offsets, much as we did when we made a supershot gather in a previous section

```bash
# sort and stack into an average shot gather
susort dt offset < junk1.su > junk2.su
sustack key=offset < junk2.su > average5187_5262.su
```

We then make approximations to the missing shots by setting the trace headers to the values that are necessary so that these new shot gathers take the place of the missing shots

```bash
## make shot 5212
#
# set sx,ep,nhs header fields fields compute gx from sx and offset
sushw key=sx,ep,nhs a=5212,180,0 < average5187_5262.su
  | suchw key1=gx key2=sx key3=offset a=0 b=1 c=1 > shot5212.su

## make shot 5237
# set sx,ep,nhs header fields fields compute gx from sx and offset
sushw key=sx,ep,nhs a=5237,181,0 < average5187_5262.su
  | suchw key1=gx key2=sx key3=offset a=0 b=1 c=1 > shot5237.su
```

Similar commands would be applied for to create average shot gather approximations of the other missing shots.
Finally, the original data and the new shot gathers are concatenated together to produce a modified version of the shot gathers

...  
\# concatenate, sort to shot gathers, reset cdp field.
cat shot22237.su shot22212.su shot22187.su shot14487.su
  shot14462.su shot14437.su shot5462.su shot5437.su
  shot5412.su shot5237.su shot5212.su seismic.su > seismic1.su

susort < seismic1.su sx offset > seismic2.su

The file \texttt{seismic2.su} still has one additional change to be made. That is to set the \texttt{cdp} field of the headers.

We begin by setting the value of \texttt{cdp} to the midpoint value, multiplied by 10 as to not lose accuracy by round off error. We have to do this, because the \texttt{cdp} header field can be only integer valued

\begin{verbatim}
suchw key1=cdp key2=sx key3=gx b=10 c=10 d=2 < seismic2.su |
suchw key1=cdp key2=cdp key3=cdp a=-16060 b=1 c=0 |
suchw key1=cdp key2=cdp key3=cdp a=0 b=1 c=0 d=125 > seismic3.su
\end{verbatim}

The second line subtracts off 16060 so that the first \texttt{cdp} header field value is 125, which is 10 times the midpoint spacing in meters. In the third line, we divide the values of \texttt{cdp} by 125, which, upon inspecting the headers, causes the first \texttt{cdp/} to have a value of 1, and the last value to be 2142, which is what the original data had. New trace count is 121440 traces.

Finally, the data can be sorted into cdp's

\begin{verbatim}
susort cdp offset < seismic3.su > seis_repaired.cdp.su
\end{verbatim}

and all of the operations we have discussed so far can be applied to the new file \texttt{seis_repaired.cdp.su}.

### 11.13 Advanced gaineing operations

Before gaining our data, we would like to remove the effect of the differing source strengths, and receiver gains on our data. These effects tend to cause vertical striping in our data. Indeed, this section should probably appear in the section on gaining, but as this requires some additional sorting of the data, we discuss the operation here.

We must use some estimate for source strength, but we also know that there are likely variabilities due to the receiver gains, so a statistical approach is used. Here, we apply RMS power balancing. The approach we use here is fast and simple, but it is not the only approach that may be applied.

Alternatively, if we had an estimate of the waveform for each shot, and an estimate of the receiver response for each receiver, we could apply \textit{signature deconvolution} to create
a surface-consistent correction for these source and receiver characteristics, as well as the individual variations in the response.

While the great pains are taken to make sources, such as airguns or vibrators reproducible and to make receivers that all have the same response, failures in reproducibility happen either because of unexpected behaviors of the instruments, or because of local conditions in the source and receiver environments.

For the Viking Graben data, we can exploit the fact that the each hydrophone is at a fixed distance to make these differences quantifiable.

### 11.13.1 Differing source strengths

We may study the source strength by looking at the rms power of each shot. We do this by windowing the data to a single offset. For example we might consider looking at the first arrival at a fixed offset, such as `offset=-262` meters. Rather than look at the full trace, we might consider looking at the maximum of the RMS value of the first reflected arrival at -262 meters.

This would be done via:

```bash
$ suwind tmin=.48 tmax=.56 < seis_offset_m262.su |
   suxmax mode=rms label12="amplitude"
   label1="energy point number" x1beg=101
   title="RMS amplitude comparisons at offset=-262 m" &
```

We first remove source strength, by beginning with our data as shot gathers. Here this is the file `seismic.su` before any other processing. There is a program called `susplit` which will split the data out into separate files based on header field value. To make separate files of shots, we first move the data into a convenient location.

```
$ mkdir Temp
$ mv seismic.su Temp
$ cd Temp
$ susplit key=ep close=1 < seismic.su
```

There are 1001 shots, so there will be 1001 files that begin with the word `split`. We may loop over these, performing a gaining operation that balances the data by shot gather panel. The gaining of choice is to divide by the **RMS power** of the data, which is the square root of the sum of the squares of the seismic data values in all of the traces of a given shot gather. In a shell script we run:

```
rm pbal.shot.su
for i in `ls split_*`
  do
    sugain panel=1 pbal=1 < $i >> pbal.shot.su
  done

rm split*
```
The double redirect out >> says “append values”, so the file `pbal.shot.su` contains all of the power balanced shots. We are free remove the separate shot files after the process is complete via

```
rm split_*
```

### 11.13.2 Correcting for differing receiver gains

Similarly, we can take the resulting power balanced shot gathers and sort these into receiver gathers via

```
$ susort gx offset < pbal.shot.su |
susplit key=gx close=1
```

The result now is a collection of files whose names begin in the word `split` each containing a single receiver gather. As before, we run in a shell script the operations

```
rm pbal.rec.su
for i in 'ls split_*'
do
      sugain panel=1 pbal=1 < $i >> pbal.rec.su
done
rm split_*
```

to yield the power balanced receiver gathers.

The final file `pbal.rec.su` of receiver gathers can be re-sorted into CDP gathers and gained, and the other processing we have discussed already can begin

```
$ susort < pbal.rec.su cdp offset | sugain jon=1 > gain.jon=1.cdp.su
```

where here, we recognize that “gain” also includes power balancing for shot strength and receiver gain. Again, we are free remove the separate shot files after the process is complete via

```
$ rm split_*
```

Both of these operations are captured in the shell script `Pbal` located in `/data/cwpscratch/Data5/`

### 11.14 Advanced deconvolution— Homomorphic Wavelet Estimation and signature decon

One method of wavelet extraction that we can try is called *homomorphic wavelet estimation*. The principle is simple. If the only thing that does not change in a collection of seismic traces is the wavelet, then the average of the Fourier representations of the traces would, by the law of large numbers, tend to the spectrum of the wavelet.
We can see why this is so by the following. If we consider the signal \( s(t) \) to be the convolution of the wavelet \( w(t) \) with the reflectivity series \( r(t) \), further convolved with the multiple series \( m(t) \), with each having its own noise, we have

\[
s(t) = w(t) \ast r(t) \ast m(t).
\]  

(11.14.1)

In the frequency domain, the convolutions become multiplications

\[
S(\omega) = W(\omega)R(\omega)M(\omega).
\]  

(11.14.2)

To make matters simpler, we would represent the data by the natural logarithm of the spectrum

\[
\ln(S(\omega)) = \ln(\left| A_w(\omega)A_r(\omega)A_m(\omega)e^{i(\phi_w(\omega)+\phi_r(\omega)+\phi_m(\omega))}\right|)
\]

\[
= \left[ \ln |A_w(\omega)| + \ln |A_r(\omega)| + \ln |A_m(\omega)| \right] + i[\phi_w(\omega) + \phi_r(\omega) + \phi_m(\omega)]
\]  

(11.14.3)

Thus the natural log of the amplitude spectra can be averaged and the phases can be averaged over a collection of traces, the result exponentiated, and the inverse transform performed.

The amplitude spectra should all be similar to the source spectrum, with a loss of higher frequencies due to anelastic attenuative loss. If the wavelet is the only thing that does not change very much trace by trace, then sum over these should tend to the log-amplitude spectrum and phase of the wavelet.

The result can be exponentiated and inverse Fourier transformed giving an estimate of the wavelet for each shot.

**Phase unwrapping**

There is an added complication in that phase that is calculated by taking the arctangent of the ratio of the imaginary part over the real part of the Fourier transformed data must be “unwrapped.” The arctangent function only returns the principal branch, which means that the arctangent function only returns phase angles between \(-\pi\) and \(\pi\). There are several strategies for doing unwrapping the phase. The trend in the phase is also removed to aid in averaging the phase values.

The output wavelet is assumed to be minimum phase. As a last step we convert the output wavelet to its minimum phase equivalent. This is done in what is called the cepstral domain, which is the inverse Fourier transform of the resulting log frequency domain representation. In SU the program *suminphase* performs this function.

The resulting wavelet may then be deconvolved from the data using *sucddecon*.

The shell script **Signature_Decom** located in **Data5** sorts the data into shot gathers, splits the shot gathers into separate files, performs homomorphic wavelet extraction and deconvolution with the extracted wavelet, and concatenates the result on **$outfile**.

The wavelet estimation seems to work best on the raw data or on the the muted raw data that has been gained.
#! /bin/sh

# estimate source wavelet by shot and receiver response
# and deconvolve input data

set -x

infile=shot_gathers.su
outfile=shot_receiver_sigdecon_$infile

echo "Signature decon with homomorphic wavelet estimation"

## Assumptions:
## 1) wavelet is constant within a shot gather
## 2) reflectivity and multiple series are random
## 3) wavelet is minimum phase

## correcting for source wavelet
# remove output files
rm shot_$outfile
rm $outfile
rm all_shot_wavelets.su
rm all_receiver_responses.su

# split the original data into shot gathers
# split shot data
susort sx offset < $infile > shot_gathers.su
susplit < shot_gathers.su key=sx

# loop over shot gather files
for i in 'ls split_sx*' 'do
    # stack real part of the complex log transform (log|A|)
suclogfft < $i | suamp mode=real | sustack key=dt > real.su

    # stack imaginary part of the complex log transform (phase)
suclogfft < $i | suamp mode=imag | sustack key=dt > imag.su

    # combine real and imag and inverse clog transform
    # and output a minimum phase version of
suop2 real.su imag.su op=zipper | suclogfft |
suminphase | suwind itmax=199 > wavelet_est.su

cat wavelet_est.su >> all_wavelets.su

# deconvolve with sucddecon
sucddecon sufile=wavelet_est.su < $i >> shot_$outfile

done

rm split_sx*

...This shell script can be easily extended to the problem of removing the receiver response, by re-sorting the resulting output into common receiver gathers, and performing the same wavelet extraction and deconvolution, much as done in the Pbal script in the previous section.

The term *surface consistent deconvolution* is often used as a label for deconvolutional processes that correct for both source and receiver effects.

### 11.15 Muting NMO corrected data

The program *sumute* may be used to surgically remove undesirable noise on the CMP gathers that occurs for times early than the water-bottom reflection. Because our prospect has a roughly flat surface the time of the reflection of the water bottom is at approximately time .48 seconds. In addition to the noise before the water-bottom reflection, there are some unsuppressed multiples, or other arrivals on the far offsets that are undesirable.

We may use predictive deconvolution to clean up those near offset traces, or we may consider eliminating the near offset traces entirely with *suwind* before further processing. Puting these together, after NMO we may insert the commands

```bash
... | suwind key=offset min=-3237 max=-450 |
    sumute key=offset tmute=.45,.45 xmute=-3237,-450 | ...
```

prior to the stack to clean up the image. The choice of nearest offset to include is a matter of personal preference. The value of −450 is not necessarily the best value.

### 11.16 Ghost reflections

The waves that travel from the source to the water surface and then propagate down in the model, as well as the reflection that travel from the subsurface, to the water surface
and to the receiver array interfere with the more direct reflections to produce what are called \textit{ghosts}. The issue of ghosting, and \textit{deghosting} can be complicated.

Because the delay between the primary reflections and the ghost reflection is short, the phenomenon reveals itself as a notch in the spectrum at the frequency where the peak from the ghost reflection cancels the trough from the primary. Thus, from the source and receiver depth, and the speed of sound in water, we can estimate this ghost notch as

\[
 f_{\text{notch}} = \frac{1}{2} \frac{V_{\text{water}}}{h_{\text{source or receiver}}} \tag{11.16.1}
\]

where \( h \) is the depth to the source or receiver.

\section*{11.17 Surface related multiple elimination}

A modern approach to multiple elimination is the Surface Related Multiple Elimination (SRME) method invented in 1991 by Erich Verschuur, then a Ph.d. student at Delft University of Technology. The method is a data driven annihilation method that makes assumptions about the structure of multiples based on an autoconvolution model of multiples. Verschuur began with the observation that multiples could be made by the convolution of a seismic trace with itself (suitably shifted) and reasoned that it should be possible to use the data itself to model the multiples, and then use adaptive subtraction to remove the multiples from the data.

\subsection*{11.17.1 The auto-convolution model of multiples}

The SRME method operates on a very simple model of multiples. If we consider the auto-convolution of data with itself, then such resulting autoconvolved data are the same as multiples, assuming simple single layer reflection.

For example if we convolve the noise free version of the \texttt{fake.su} data with itself and add this to the noise free data

\begin{verbatim}
$ suconv < fake_no_noise.su sufile=fake.su panel=1 > autoconv.su
$ susum autoconv.su fake_no_noise.su > fake+autoconv.su
$ suxwigb < fake+autoconv.su title="autoconvolution multiples"
\end{verbatim}

we see that the result is similar to what we would expect for data with multiples.

In theory, we could use the earliest arrivals on a seismic reflection profile to build a model of the first multiples through autoconvolution, and then adaptively subtract these out of the data. The demultipled portion of the data would then be autoconvolved to generate a model of the second order bounces, which in turn would be subtracted. The process of modeling followed by adaptive subtraction can then be repeated until the data are completely cleaned of multiples, as best as the algorithm could handle this.

Both 2D and 3D versions of SRME have been implemented.

Unfortunately, we do not yet have an SRME code in the SU.
11.18 Homework Assignment #8, Due Thursday 5 Nov, before 9:00am and Tuesday 3 Nov 2015

This exercise is similar to problem #7 however you will be applying more processing techniques.

- Start with the ungained and unmuted data.
- Mute and gain dataset.
- Apply one of the spectral methods to sharpen the waveform of the data.
- Use suwind to break the full dataset into blocks that are about 500 CDPs in size. The number of blocks is up to you. For example, the last block was more than 500 CDPs, so you could split that one up. Or maybe you want to use bigger blocks.
- Perform the Radon-domain multiple suppression on each of these smaller blocks, to do a better job of multiple suppression. It is more important to preserve the real reflections than it is to try to suppress all multiples, at the expense of data. You may have done all of this already in the previous assignment. Redo parts only if feel that you need to improve the result.
- Concatenate the blocks together to form the full multiple-suppressed version of the dataset. You can call this dataset radon.gain.yourgainparameters.cdp.su
- Repeat exercise #5 on this multiple-suppressed dataset (except do not show near and far offset stacks. Stack over all traces, and use your best stacking velocities.) Save your NMO velocities in a file called, say radon_nmo_vel.par. For example, if you used the CDPs at 250,750,1250,1750 to get your NMO velocities, then the contents of the file would look something like:

```
cdp=250,750,1250,1750
tnmo=0,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
vmo=1500,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
tnmo=0,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
vmo=1500,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
tnmo=0,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
vmo=1500,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
tnmo=0,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
vmo=1500,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.
```

where the $tnmo= vmo=$ are those that you got for the respective CDPs. Note that the number of $tnmo= vmo=$ values per pair have to be the same, but each pair may have a different number of values from other pairs.
The idea is to concatenate and then NMO-correct the data. Use this `radon_nmo_vel.par` for performing the NMO correction before stacking.

```
$ sunmo par=radon_nmo_vel.par < radon.gain.yourgainparameters.cdp.su | sustack > stack.su
```

Obviously, you cannot complete this assignment on time if you do not start working on this immediately. Ideally, everyone will have a Radon multiple-suppressed version of the full dataset `radon.gain.yourgainparameters.cdp.su` ready for class on the due date so we can proceed with fancier velocity analysis.

Other tips:

- Feel free to use `at` to run the jobs at night.

- Furthermore, you might consider regaining the data after you have done multiple suppression.

- Who says that you need to stack all of your data? It may be that the far offsets and the nearest near offsets could be omitted from your data, and make the dataset a bit smaller. However, is it worth the loss of data? You decide.

11.18.1 How are we doing on multiple suppression and NMO Stack?

The subset approach that is pursued in the Homework 7 and 8 suffers from a serious flaw. While we have a set of stacking velocities for each block, we are not taking advantage of the ability of our programs to interpolate these values across the section. We may see a blocky appearance.

While performing this procedure in subsets makes it a bit quicker, this is for instructional purposes only. **From now on, we work with the full dataset. We do not break the data into blocks.**

11.19 Concluding Remarks

Much of exploration seismic research conducted prior to the mid 1980s was focused on the problem of seismic deconvolution and wavelet estimation. The CWP/SU:Seismic Unix package was largely developed during a time right after this, when exploration seismic research was focused on amplitude preserving depth migration, so consequently we have a lot of migration related tools and a comparatively few deconvolution-related programs, so deconvolutional methods are not yet well represented in the SU package.

Though we have not really done justice here to the broad topic of deconvolution and the other spectral methods, we can see that the application of these techniques is more
involved than merely applying the operation. Considerable preconditioning of the data is required to make the traces look more alike, so that the deconvolutional process may remove the parts (such as multiples) that we don’t want.

Predictive decon really means that we use the first part of the data to predict the repetitions in the latter part of the data, and to use those predictions to annihilate those repetitions (multiples). For this to work the repetitions must closely match the initial waveforms. Hence, making the amplitudes as uniform as possible is desirable for such techniques to be applied.

In the modern world there is an increasing demand for amplitude information for the extraction of amplitude versus angle (AVA) also known as amplitude versus offset (AVO) information. Balancing away all of the amplitude variability in the data is not desirable, so methods that preserve amplitudes and are data driven are preferred.