Review / Orientation to Matlab
GEO 384r
To be completed by February 1.

1. Complex numbers, matrices, vectors, m-files, and plots.

A. Write an m-file that generates and plots in 3D (Real, imaginary, and time axes) a complex sinusoidal function of time, (a phasor that looks like a corkscrew) of the form $A(t)\ \exp(i2\pi f(t)t)$ where A(t) is a specified time series of amplitudes, f(t) is a specified time series of frequencies, and t is time. Interesting examples might be A(t) and f(t) growing linearly with time. Use PLOT3

C. Plot a Matrix as a surface in color – Generate a Toeplitz matrix (see TOEPLITZ) as an example. Use SURF

D. Generate and plot, using SURF a 100 x 100 matrix as in 1C that is the product of a 100 x 1 vector and a 1 x 100 vector (The two vectors multiplied to form a full matrix constitute a dyad)

2. Sampling and aliasing, elementary statistics, random time series

A. Generate a plot of 2 single frequency cosine functions whose samples have exactly the same values (so the frequencies are aliases of each other). In general aliases are separated by the sampling frequency. Thus, when the sampling frequency is, say, 100 Hz, the alias of 25 Hz will be 125 Hz, 225 Hz, and so on. Plot the two sinusoids, and use a star symbol to indicate the sample values. We will confirm this result when we study the sampling theorem using the Fourier transform.

Aliasing is often explained by referring to the Nyquist frequency (1 cycle in 2 sample intervals or f=0.5 in standard notation) as the folding frequency. The idea is that a sinusoid above the Nyquist by some increment, will alias into a frequency below the Nyquist. For example, the frequency f=0.9, above the Nyquist (=.5) would be expected to be an alias of f=0.1, which lies below the Nyquist. Confirm that this is true when the cosine function is used, but when the sine function is sampled, the sign must be reversed to duplicate the samples of the other frequency. For example, integer spaced samples of $\sin(2\pi(0.1)t)$ are identical to integer spaced samples of $-\sin(2\pi(0.9)t)$. The reason why cosine and sine behave differently will be apparent when we prove the sampling theorem using Fourier transform theory.

3. Get a geophysical or other time series from the internet, your own research, or the 325K website. Also generate 1000 point time series of random numbers that are uniformly distributed (RAND), Gaussian distributed (RANDN), and Chi -squared with 5 degrees of freedom (sum of 5 independent Gaussian random variables N(0,1)) Then use Matlab functions MEAN, VAR, STD, MEDIAN, to determine the time series elementary statistics. Write a brief statement which defines what the meaning of each elementary statistic. Finally generate a histogram (use HIST) for each time series and discuss what it tells you about the time series. In the case of random numbers, the histogram will have a form similar to the probability density function.

4. Linear Digital filters, transfer functions, and implementation in matlab

A. A general linear digital filtered version of a time series x is implemented in Matlab using FILTER(b,a,x), with the Moving Average coefficients in the vector b, and the autoregressive coefficients in the vector a. The transfer function of the filter can be obtained with the command FREQZ(b,a). Zero-phase implementation of the filter can be done with FILTFILT(b,a,x).

Generate a time series of 1000 Gaussian random numbers RANDN(1,1000). Filter it with a=[1] and values of b that are HANNING weights that sum to 1, and vary in lengths from 3 to 13 in steps of 2. Plot the filtered series on top of each other using both FILTER and FILTFILT. Explain the meaning of 'zerophase' for FILTFILT, and compare the transfer functions of the various filters.

4. Least squares and linear algebra problems in Matlab

A. Use Matlab to find the intersection of the lines y = 3x + 4 and y=-3x -4. Plot the two lines and their intersection.

B. Use Matlab to find the least squares best fit line of the form y=a1 x + a2 to the x,y points (1,1), (1,3),(2,3) and plot the points and lines. Use POLYFIT to confirm that you get the same solution as from the normal equations.

Chapter 6 Exercises GEO 384r
6A – Due February 8, 2008

1. The integral Fourier transform can be defined in terms of angular frequency variable $\omega = 2\pi f$. Write down the Fourier transform relationship between x(t) and X(ω).

The Fourier transform can also be defined such that the transform from time to frequency domains has a positive sign in the exponent, instead of a negative sign. What is the derivative theorem when this is the sign convention?

2. Find even and odd parts of the following functions of time t. Which have Hermitian symmetry?
$\exp(2\pi i f t)$
H(t), the Heaviside step function (0 before t=0; 1 after t=0)
$\cos(2\pi f t - \pi/4)$
$2\pi t$

3. Find the even part of the product of two functions in terms of the even and odd parts of each function
Find the even and odd part of the sum of two functions in terms of the even and odd parts of each function

4. A. Show that convolution for continuous functions, via integration, is commutative. That is show from the integral definition and rearranging variables of integration, that g(t)*h(t)=h(t)*g(t).

B. Like discrete convolution, there are other properties (commutative, associative, and others) for convolution of continuous functions. These can be verified using rearrangement of integration variables as in the previous example in (4A), a process that is straightforward, but tedious. On the other hand it is quite easy to show by simple examples that these various properties hold for discrete convolution.

Show using short time series (length 3) that the following properties of discrete transient convolution are true: If (a,b,c) are discrete time series then discrete transient convolution is: associative (a*b)*c = a*(b*c), commutative, a*b=b*a, and distributive over addition, a*(b+c)=a*b + a*c.

5. By direct integration, show that the inverse Fourier transform of the Boxcar function, $F^{-1}\{ \Pi(f)\}$ is sinc(t). This must mean the sinc(t) is a 'band-limited' function because its Fourier transform goes to zero outside the frequency range (-½ to ½).

Fourier transform theorems (especially the convolution theorem) can be used to justify the following statement: "Band limited functions cannot be time-limited, that is, they have non-zero values that extend to infinite times" Therefore any transient time function (one that goes to zero outside some definite interval) must be composed of frequencies that are infinitely large, no matter how smooth it may appear to be.. That is, a time-limited function cannot be band-limited. Similarly a band-limited function cannot be time-limited. Write a few sentences that justify these statements, making reference to the convolution theorem, and using the sinc function-boxcar function transform pair as part of the argument.

6. The Dirac delta function is zero everywhere except at the origin, where it is infinite. Its properties can be deduced by considering a sequence of taller and narrower box-car functions, all of which have unit area. Use the Fourier transform theorems to construct a sequence of narrower and taller sinc functions {1/a)sinc(t/a), as the parameter a is allowed to get smaller and smaller, approaching zero. Sketch a few of this sequence of sinc functions to show their behavior near t=0. Show that in the limit, they become delta functions and their transform is 1. Repeat the argument, but this time use a sequence of functions involving first the triangle function Λ(t), then $\text{sinc}^2(t)$. That is, these sequences of quite different functions, some of which alternate in sign, will all converge, in the limit, to the same generalized function δ(t).

7. The Dirac delta function has properties related to the Fourier transform of continuous functions, analogous to the discrete time series impulse δ_t = (1, 0, 0, …) and its relationship with the DFT. Describe or show these relationships (for the Fourier transform and DFT)

The delta function (or impulse sequence) is: The identity operator for convolution (or discrete convolution)

The transform of the delta function (or impulse sequence) is a constant function of frequency (or for the DFT)

A delta function of frequency corresponds to a constant function in time.

8. Find the Fourier transforms of the following functions, using appropriate theorems:
   A. $\cos(2\pi ft) = [\exp(i2\pi ft) + \exp(-i2\pi ft)]/2$ (hint: shift theorem)
   B. $II(t)*\cos(2\pi Mt)$ for M an integer (hint: convolution theorem)
   C. $\text{sinc}(t)*\text{sinc}(2t)*\text{sinc}(3t)*\text{sinc}(4t)*\text{sinc}(5t)$ (hint: convolution theorem)
   D. $\text{sinc}(t)*\cos(10\pi t)$ (hint: convolution theorem)
   E. $\sin(t)/t$ (hint: similarity theorem)
   F. $\delta(t-1/4)*\sin(2\pi t)$

9. The Fourier transform provides a convenient way to understand the response of a simple harmonic oscillator to forcing. An example is a mass on a spring. When shaken, the mass will respond by moving up and down, but if shaken at a frequency near the resonant frequency, the resulting amplitude will be much larger. To describe this quantitatively, start with the differential equation governing a mass on a spring

$$d^2y/dt^2 + (2\pi f_0/Q)dy/dt + (2\pi f_0)^2y = x$$

where x=x(t) and y=y(t) are both functions of time t, with x being the forcing (acceleration applied to the spring base) and y the displacement of the mass from rest. Here $2\pi f_0$ is the resonant angular frequency (the square root of the spring constant over the mass of a mass-spring, for example), and Q is the quality factor, a dimensionless parameter which is big when there is little damping.

A. Find an expression for the transfer function, the ratio of Y(f)/X(f) for values of Q=1 and 10.

B. Write down an integral expression for the impulse response, that is, the output when $x(t)=\delta(t)$

C. Plot the transfer function fpr the two values of Q using Matlab, over a range from $-2f_0$ to $+2f_0$ For calculation purposes, you can pick f_0 to have some convenient numerical value, such as 1. Since the transfer function will be complex, make separate plots for its amplitude and phase.

10. Complete the following sentences

X(f) has an even real part and an even imaginary part so x(t) is _____

X(f) has an odd real part and an even imaginary part then x(t) is

x(t) is purely imaginary so X(f) has the property that its even part is _____ and its odd part is _____.

11. The autocorrelation function associated with x(t) is defined as the convolution of x(t) with its time-reversed complex conjugate. Show that the Fourier transform of the autocorrelation of x(t) is real, even and positive at all frequencies.

12. There are many similarities between the DFT and the Fourier transform. State the following properties of the Fourier transform and then use Matlab's FFT function to verify the analogous properties of the DFT.

The convolution theorem (for the DFT give examples of both circular convolution and zero padding convolution theorems).

The shift theorem (for the DFT, the shift is applied cyclically, so shifting a time series to the right by 1 term moves the last element of the series to the front of the series)

The Rayleigh Theorem (Parseval Theorem for the DFT)

Autocorrelation theorem (compare the DFT of the autocorrelation, rearranged to make the zero-lag term first, and negative lags appearing in reverse order after positive lags. Compare with the squared modulus of the DFT after it has been padded with sufficient zeroes. For example if x=[1,1,1], the autocorrelation is [1,2,3,2,1], and when rearranged to have the correct symmetry it is [3,2,1,1,2]. The DFT of this is compared with the squared modulus of the DFT of [1,1,1,0,0], which is padded to have the same length.

Chapter 6 Exercises GEO 384r
6B due February 29, 2008
These are mostly Matlab exercises. The simplest way to complete them is to open this file as a Word document, then copy and paste in figures and results from the Matlab command or figure windows. You can insert paper and pencil work as needed.

1. Demonstrate these properties of the Discrete Fourier Transform

A. Matlab functions FFT and IFFT are inverses for a simple time series.

B. The DFT of a real time series has Hermitian symmetry.

C. When a time series of length N consists of samples of a single Fourier frequency (f= m/N where m is an integer), then its DFT is zero except at that (positive and negative) frequency, Furthermore, show that the real part of the DFT is N/2 times the amplitude of the cosine component of the sinusoid, and the imaginary part is N/2 times the amplitude of the sine part (note the sign of the sine component is reversed at positive and negative frequencies.

D. The Convolution Theorem (2 forms) for the DFT (Zero padding and circular convolution)

E. When N is an odd number, the DFT is not computed at the Nyquist frequency, so generally all values of the DFT of a real time series will be complex, except at f=0.

F. When N is an even number, the DFT is computed at the Nyquist frequency with the result that the imaginary part at f=0 and f=1/2 will be zero for real time series.

G. Construct a non-trivial time series of length 16 which is odd, so that its DFT is purely imaginary

H. Construct a non-trivial time series of length 16 which is even, so that its DFT is purely real.

2. For the case of the continuous function Fourier Transforms show that F[x(t)] and F[x(-t)], where x(-t) is the time reversed version of the function, are complex conjugates of each other. Show that when x(t) is a complex valued function, the corresponding result is that the Fourier transform of the complex conjugate of the time reversed function, F[x*(-t)], is the complex conjugate of F[x(t)].

A similar relationship holds for the DFT, with the proper interpretation of how to write down a time-reversed series. Since a time series is 'understood' by the DFT to be periodic, time reversal corresponds to leaving the t=0 (first element) where it is, and swapping positions of the other time series elements.

Construct two simple time series, one with 4 elements, one with 5. Then find the time-reversed version of it. Verify that you have done this correctly by showing that the DFT of your time reversed version is the complex conjugate of the original series.

3. The Sinc and Boxcar functions are Fourier transform pairs. Similar (but periodic) discrete time function pairs exist for the DFT. That is, the transform of a discrete finite length boxcar time series is not precisely a Sinc function, though it is similar. Construct a 100 term discrete boxcar function that is zero for times 10 and larger. To make it symmetric, you will need to put 'negative times' of the boxcar function on the right hand side of the time series. That is, the boxcar time series is [1, ones(1,10),zeros(1,79),ones(1,10)]. Plot the real part of its DFT (a discrete sinc-like function) along with the exact sinc function (properly scaled in amplitude so the value at f=0 is unity, and scaled in time so that its zero crossing match the computed one) to see how they differ. The discrete sinc-like function is periodic, whereas the exact sinc function is not. Matlab has SINC as one of its built-in functions. The

4. The DFT is a handy tool to perform interpolation between discrete time series samples. The operation is analogous to Whittaker's interpolation formula, but is done in the frequency domain, rather than as time domain convolution. The interpolated values are samples of a band-limited function that passes through all the original samples. The method works because the IDFT is really an underlying continuous time function, a sum of continuous functions (the complex sinusoidal functions of time) multiplied by Xm, the DFT values of x. Recognizing this, the IDFT can be evaluated either at the original discrete integer times, or any time between them, or even outside the original interval of the time series. To evaluate the IDFT at other than integer times, one 'tricks' the algorithm by padding zeros. Evaluating the IDFT outside the original time interval [0,N-1] one will find periodic replication with period N (the time series length), as with an ordinary Fourier series.

Using the 4 point time series x=[4,3,2,1] perform interpolation at midpoints (2-fold interpolation) using the DFT via the following zero-padding procedure: Note that this recipe gives array index values starting at 0, but when working in Matlab, arrays all start at 1. Interpret the following recipe accordingly.

Find X, the DFT of x
Find one-half the Nyquist value X2.
Form a new frequency series Y = [X0, X1, X2/2, 0, 0, 0,,X2/2, X3] (explain why this has Hermitian symmetry, and thus is the DFT of a real time function)
Find IFFT of Y, and multiply the result by 2 (this corrects for the normalization (1/8) on the IFFT, and changes it to (1/4) as for the original series. The times corresponding to the 8 point series are [0, .5, 1, 1.5…], the original times and midpoints.

Plot the result, along with the original 4 points, with star symbols at their times (0,1,2,3).
What does the last value in the interpolated series correspond to, and what difficulties would this present in a practical situation?

Repeat this process for 16-fold interpolation (the interpolated series will have 64 points instead of 8) and plot the underlying continuous function along with the original time series values, to show that it is very much like you would anticipate from sinc-function interpolation of the original 4 points.

Explain how interpolated values are a result of 'wrap-around' at times after the last time of the original series.

Repeat, for the 16 fold interpolation case where the time function is [1,0,0,0]. That is, interpolate the delta sequence. Explain the result.

Finally, explain what the analogous result if you pad zeros in the time domain. What frequencies do you obtain, for example if you pad N zeros, and how do the values relate to the original DFT values? Show this with an example of zero padding of 2, 4, and 8 fold.

5. The analytic signal associated with a continuous real function is a complex function whose real part is the function and whose imaginary part is its Hilbert transform. Conventions for the definition of analytic signal may vary a bit, in particular the sign of the imaginary part – some books use the negative of the Hilbert transform for the imaginary part, but Matlab sets the imaginary part to be the positive Hilbert transform. The Hilbert transform of a real valued time function is a 90 degree phase-shifted 'twin' called the quadrature signal. The modulus of the analytic signal is the instantaneous amplitude, and the time rate of change of phase (divided by 2π) is the instantaneous frequency. When applied to discrete time series in geophysics these concepts have many important uses. Two examples are: as an aid in seismic reflection both instantaneous amplitude and phase are examples of seismic 'attributes' ; the variable phase shift that results when a wave is reflected beyond the critical angle can be implemented using a linear combination of the original wavelet and its Hilbert transform.

Read the Matlab help files for the function HILBERT. Then pick a simple example time series x and compare the DFT of x with that of its analytic signal. Explain how the HILBERT algorithm works, and how it relates to the properties of the continuous Fourier transform, as described in class. An example of a simple time series would be a sampled cosine function.

For a 32 point cosine time series of frequency 1/16 (which is a Fourier frequency) show that its analytic signal reproduces the cosine for the real part, and that the Hilbert transform appearing in the imaginary part (also called the quadrature function) is the sine. This result is exact when the sinusoid is a Fourier frequency. If you did not do so previously, show that it is only approximate for other frequencies.

Make up a 'wavelet' time series w_t of length approximately 30, with a nice wiggle or two in it near the middle. (example – a sine function that makes about 4 oscillations in 30 samples and exponentially or linearly decays to near zero by the last sample) Show that you can create a general phase shifted version of it of the form $w_t\cos(q) + w_t^{hi}\sin(q)$ and that as q varies from 0 to 90 degrees, the phase shift does as well. Find the instantaneous amplitude A and instantaneous phase P (use the function PHASE to get an unwrapped phase). Plot these along with the wavelet w_t and its hilbert transform w_t^{hi} Find instantaneous frequency F(t) by convolution with [1,-1] and division by 2π as an approximation to the derivative, and compare with the frequency of your wavelet. Repeat this, but use a linear variation in frequency chirp (Matlab CHIRP function).

Homework 7A Digital filters and transfer functions
Due March 5

1. Use Matlab to study properties of the following linear digital filters - for each perform the analysis and obtain plots of their transfer functions. Write

A. First Difference $y_t = .5x_{t+1} - .5x_{t-1}$
B. Crank-Nicholson $y_t = 2x_t - 2x_{t-1} - y_{t-1}$
C. Second Difference: $y_t = x_{t+1} - 2x_t + x_{t-1}$
D. 11 term Boxcar moving average
E. 11 term Triangular moving average
F. 11 term hanning taper moving aaverage

G. Damped sinusoid of frequency f_0, with quality factor Q (pick your own values here – say a frequency of 0.1, and a Q of 2) so that the impulse response is a damped sinusoid, well damped in about 30 time units. The Matlab command generates the correct autoregressive array values:

a=[1,-2*cos(2*pi*f)*exp(-pi*f/Q),exp(-2*pi*f/Q)].

Corresponding to the filter equation

$y_t = x_t + 2\cos(2\pi f_0)\exp(-\pi f/Q)y_{t-1} - \exp(-2\pi f/Q)y_{t-2}$

H. Damped sinusoid as in H, but written as an MA filter

For each of the filters above do the following for groups (A,B,C), (D,E,F), (G,H)

. Write down the array of filter coefficients, a and b arrays used by Matlab

Find and plot the impulse response time series.

Find and plot the impulse response of the inverse filter. Explain why some are unstable.

Verify that the convolution of the inverse filter with the impulse response of each filter is the delta sequence.

Find the transfer function as a polynomial in Z

Find the poles and zeroes and plot in the Z plane (use ZPLANE and ROOTS)

Use Matlab (freqz) to plot amplitude (modulus) vs frequency. Also plot the power transfer function (modulus squared on a db scale vs frequency). For the filters that approximate derivatives (A,B,C) also plot on the same graph the transfer function of the exact first or second derivative

. Find the peak of the power transfer function in db, and the filter pass band and stop band. The pass band is within 3db of the peak, the stop band is anything outside of this frequency band (-3db is .707 of the peak value on a linear scale)

2. Use Matlab to calculate and plot the signal to noise ratio in decibels of the inverse Crank-Nicolson filter as an integrator, as a function of frequency, zero to the Nyquist. Plot it with the signal to noise ratio of the other integrator $y_t = x_t + y_{t-1}$

Plot the phase error in radians as a function of frequency for each.

3. Factoring Polynomials

The inverse function to ROOTS is CONV. Demonstrate this by factoring a quartic polynomial that you have created using CONV.

A. How do the roots of a polynomial compare with the roots of the polynomial whose coefficients are reversed in order? Do an example and plot the results in the complex plane to confirm.

B. Write down an example of an ARMA filter of order (4,4), and express it as a cascade of 8 or more different single or multiple pole and zero filters. Verify that the convolution of all their impulse responses in the same as the original impulse response.

Verify that the sum of all their power transfer functions (on a db scale) equals the original.

4. Write your own m-file that finds the transfer function of a digital filter, giving it at least some of the same functionality as freqz. Use the DFT, zero padding, and unlike freqz, allow the filter to be a-causal (so that the moving average part can use future and past values). Test your m-file by finding power and phase transfer functions for a symmetric MA filter, like $y_t=[x_{t+2} + x_{t+2} + x_t + x_{t-1} + x_{t-2}]/5$, as well as by comparing with freqz.

Exercises 7B GEO 384r and GEO 365n
Digital Filter Design - due March 23

1. Suppose that the output of a seismometer y_t is related to ground motion x_t by the following filter.

$$y_t = 2x_t - 5x_{t-1} + 2x_{t-2} + .5y_{t-1}$$

a. Find the exact inverse to this filter that recovers x_t given y_t (now regarding x as the output and y as the input) Show the filter is unstable (that is, find its poles and plot them to show they are in the region of the Z plane associated with instability – this region is outside the unit circle in the Matlab convention, inside the unit circle in the one used in the notes), and compute enough of the impulse response to verify this property.

b. Show that transfer function factors (1-aZ) and (a*-Z), whether they are poles or zeros, have exactly the same power transfer function. (a* is the complex conjugate of the number a, in the general case when it is complex). Use an example for a particular value of parameter a to show that the location of the pole or zero of (1-aZ) is at the polar reciprocal of a* (lies along the same radial line from the origina at reciprocal distance.

c. Using the result from (b), find the approximate inverse filter, instead of that in (a) to recover x from y that has the same power transfer function as the exact inverse, but is stable. That is, factor the transfer function, replace the unstable pole with the one that corresponds to stability, and then recalculate the filter coefficients of the stable inverse.

d. Find and plot the seismometer impulse response (seismometer output y for a delta function input x); seismometer impulse response after filtering with the unstable inverse filter, which should be exactly a delta function, except for end effects; seismometer impulse response after filtering with the stable inverse filter from c, which will be the best approximation to a delta function that can be achieved.

2. Given an example of two 5 term time series (or equivalently MA filter coefficients), one of which is minimum delay (or minimum phase), and the other is not, but which have the same amplitude spectrum. Plot the modulus and phase graphs (for both) for the frequency interval [-.5 to .5] (First find the Z transform of the series evaluated at $\exp(-i2\pi f)$ after adjusting for the linear phase associated with not being centered at time zero by multiplying by Z^{-2})

3. The bilinear transform digital filter is, $y_t = 2\,x_t - 2x_{t-1} - y_{t-1}$. Find the poles and zeros of this filter, showing that there is a pole on the unit circle, near the frequency f=0, making it meta-stable, thus potentially dangerous in a numerical

calculation. By moving the pole just off the unit circle, you can make it stable. Find the new filter with the pole at radius (1+a), where a is a small positive number. Plot its transfer function relative to the exact derivative result for a couple of values of a (say 0.01 and 0.1).

4. If the digital sample interval is 2 milliseconds, use trial placement of poles and zeros to design an MA filter that rejects zero and the Nyquist frequencies, and has a power transfer function of 1 (unity) at 125 Hz.

6. A. Use two of the Matlab filter design algorithms to develop 2 different low pass filters with fewer than 9 coefficients total. The filter design objective is to reject frequencies above 1/3 the Nyquist. That is, you are designing a low pass filter.

B. As a second possibility you can develop a time-symmetric filter using a truncated sinc function. The starting point should be the exact filter function (1/3)sinc(t/3). Sample this at 9 points, symmetric about t=0. Then apply a taper (from the matlab collection, perhaps a hanning taper). Plot the power transfer function of the truncated sinc function, the exact transfer function that is desired (1 in the pass band, 0 in the stop band), and that of the tapered sinc function, to show that the taper effectively reduces ripples in the stop band, at the expense of a more gradual transition from pass to stop band. Compare with the performance of the filters in A.

7. Find a time series for the impulse response of constant Q attenuation, assuming it is minimum phase, using polynomial factorization and Hilbert transform methods, as explained in the notes. As shown in the notes, the zero-phase time series is $1/[1 + (2Qt/\tau)^2]$ where τ is wave travel time and Q is the quality factor. You need to pick a value for Q/τ that gives a time series that decays in about 50 samples, and plot it for both positive and negative times. Plot the results from the two methods. This filter would be one way to simulate attenuation in synthetic seismograms, as a convolution filter. We will look at the same problem later as a least squares filter design problem.

Chapter 7C Exercises
Least Square filter design and applications - Due April 11

1. The deconvolution filter in the seismic reflection problem

A. Generate a time series y by the following Matlab commands. As you can see, it is a filtered version of the series x plus white noise. This is a convolutional model, in which y, the observed signal (for example a seismogram), is a linearly filtered version of x, plus added noise (n). Assume that the digital sample interval is 2 milliseconds to get the frequency values in Hertz.

```
noise=randn(1,1000);
x=[zeros(1,10),20,zeros(1,100),-20,40,-20, zeros(1,886)]+randn(1,1000);
b=[1,1,1,1,1];
a=[1.2,-1.4,1];
signal=filter(b,a,x);
y=signal + noise;
```

The time series y (a 'seismogram') consists of signal (linearly filter version of x) plus noise (Gaussian, white). In reflection seismic terminology, the signal x is a combination of random variations (small reflectors) and some isolated 'large reflectors' The linear filter used to turn x into y is not a pure AR filter (it is ARMA). Find the transfer function of this filter (via freqz(b,a), and comment about the frequencies that will be missing in y that were in x, and therefore could never be recovered from y.

B. Verify that the shape of the power spectrum of the signal (y) is very similar to the spectrum of the impulse response of the filter (the spectrum is the squared modulus of the transfer function) defined by the MA and AR coefficients (b's and a's). The simplest way to do this is to compare autocorrelation functions, since this is the Fourier transform of the spectrum. Compute the autocorrelation of the signal y, and compare it with the autocorrelation of the impulse response of the filter (b,a). To do this, normalize by the zero-lag value, so the maximum value of the autocorrelation for each is (1). You can then compute the squared modulus of the DFT of each to get a spectrum. Plot the two autocorrelation functions, and the two power spectra. Explain the results

C. Next you will show that the prediction error filter is an approximate inverse to the filter (wavelet) in A and has the effect of whitening the seismogram (balancing the spectrum) of the seismogram time series.

Use ARYULE or LPC (they should give the same result or nearly so). To find the prediction error filter and associated prediction error for filter orders 2,4,6,8,10. (filter lengths 3-11), using the time series y in A. There are several ways you could do this. (One possibility is to use the series y as in conventional seismic processsing, or since you have the exact impulse response of the filter defined by (b,a) you could use that also). Explain why the size (variance) of the prediction error diminishes with successively higher filter order.

Plot the power spectra (power transfer function) of each of the 5 prediction error filters, and the spectrum of the original time series (y or or the impulse respone) on the same graph. You should find that as the order of the prediction error filter increases, more peaks and troughs appear in the power spectra, but generally they have a shape that is reciprocal to that of the spectrum of y (or the exact wavelet)

Apply the prediction error filter for a filter length of 4 (order 3) and 11 (order 10) to your data. This is a pure MA filter. Show that after filtering y with the prediction error filter, the result is a time series with an autocorrelation function more like white noise (more like a delta function).

2. Here you will illustrate issues that arise when you use a prediction error filter as a deconvolution or inverse filter in the presence of noise.

A. If there were no noise present, (that is, if y=signal) and you happened to know (b,a), the exact filter, then you could recover x from y via the Matlab command

x=filter(a,b,signal).

verify that this works. But y actually includes noise, because it was generated above by y=signal+noise

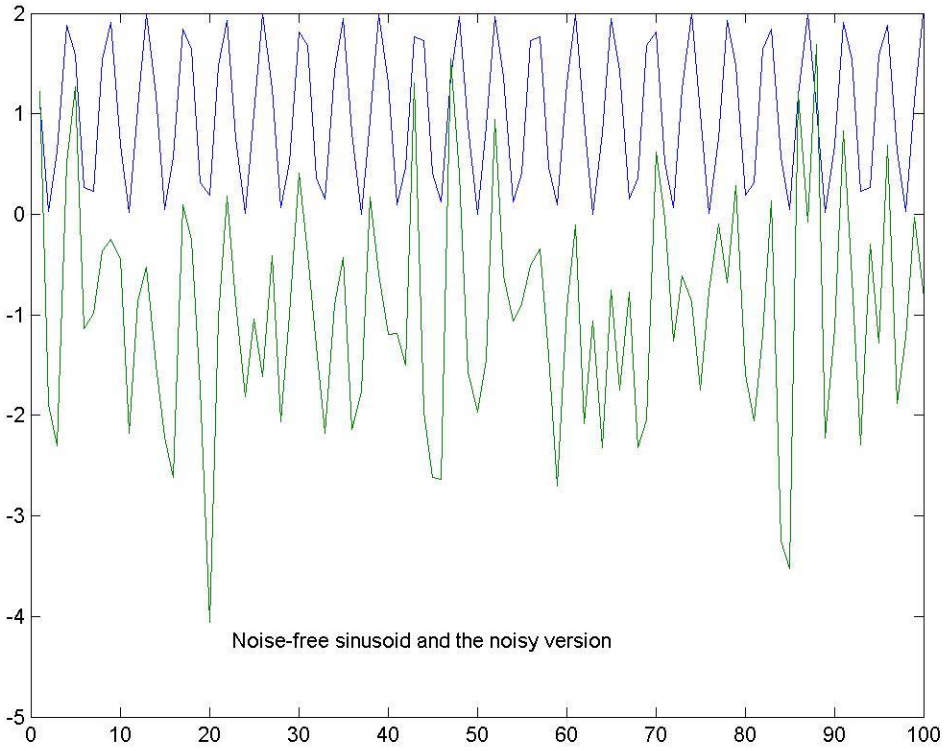
Try executing the command
xest=filter(a,b,y)
for the noisy y. Explain why xest looks so different from x.

To deal with the noise, construct 2 or 3 prediction error filters of length 5 using 'false white noise' values of 0, 10, 50 and 100% (that is, inflating the zero lag autocorrelation by these percentages). Compare the result with the true x (which you know from above). Plot the result of the 4 trials, together with y and comment on their differences. It might help to make all of them zero mean and unit variance by subtracting the mean value and dividing each series by its own standard deviation.

3. Since the Prediction error filter has a power transfer function or spectrum that is inverse to that of the series y, it can be used to form a power spectrum estimate. This is the Maximum Entropy Method. The following illustrates that it gives reasonable results in a simple case of a single sinusoid.

Generate a sinusoidal time series of length 100, and amplitude 1 at some frequency, not a Fourier frequency. Call this series x

Add white noise to x with a variance of 1, and call this series y.



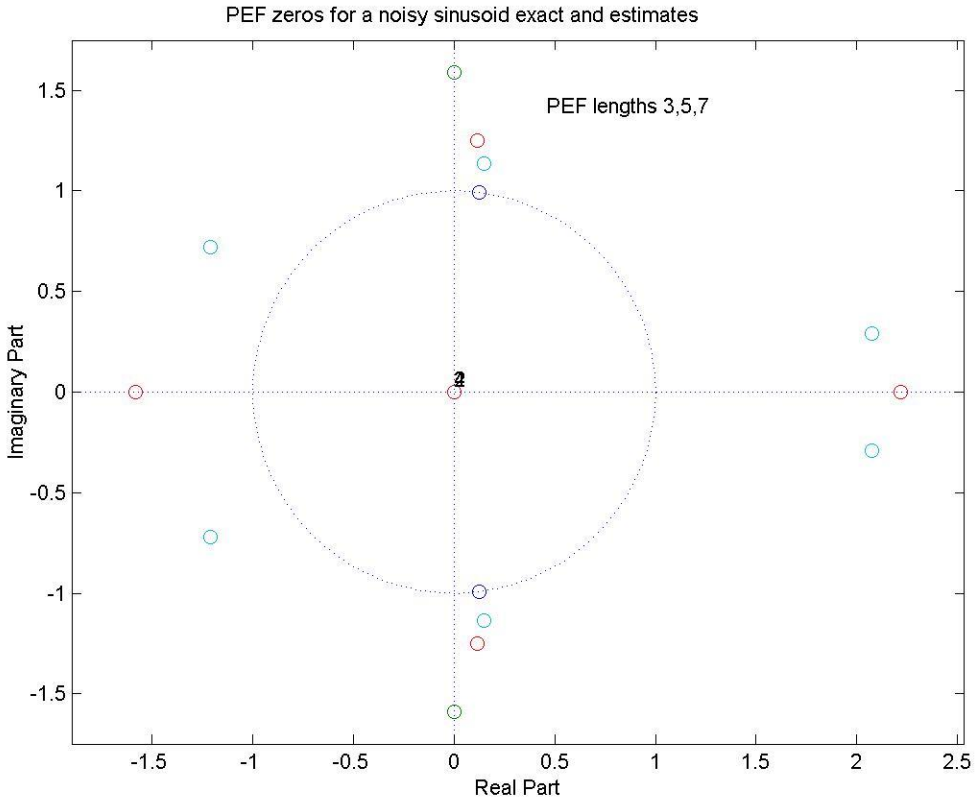
Form the prediction error filters for x and y. The simplest matlab function to use is lpc, since this works directly with the time series. See the help file HELP LPC for details. There is only one frequency present, so it is correct to pick the order N to be 2. Also try 4 and 6 to see how the result varies. Save the various filters for x and y. An example from the above generated the following results.

>> real(lpc(x,2)) 1.0000 -0.2431 0.9794
>> real(lpc(x,4)) 1.0000 -0.1242 0.9735 0.1154 0.0242
>> real(lpc(x,6) 1.0000 -0.1241 0.9743 0.1222 0.0162 0.0088 -0.0072

>> real(lpc(y,2)) 1.0000 -0.0024 0.3960
>> real(lpc(y,4)) 1.0000 0.0339 0.3241 0.1567 -0.1807
>> real(lpc(y,6)) 1.0000 0.0510 0.2791 0.1058 -0.1611 -0.1773 0.0873

The power spectrum shape is given by the reciprocal of the squared modulus of the prediction error filters. You can generate these by padding all with zeros to give them a length of say 200, taking the DFT, and then forming its reciprocal. Plotting on a decibel scale is usually more interesting and useful. Plot these for the 6 cases

Use roots or zplane to locate the zeros of the prediction error filter (there should be two in conjugate pairs. The exact peak in the frequency is given by the angle of one of the zeros multiplied by 0.5, and divided by pi. Compare with the exact result, corresponding to your chosen frequency.



4. More on seismic deconvolution: In the seismic deconvolution problem the seismogram is assumed to be the convolution of a wavelet (unknown), with a sequence of reflection coefficients that behave as random noise, and therefore have an autocorrelation function that is a delta function. Demonstrate that this idea works via the following steps.

Create your own wavelet. For example, place two poles in the Z plane at conjugate points in a mid frequency, such as 1.5i and –1.5i. Use the matlab function poly to find the z polynomial Find the first 20 or so values of the impulse response – this is your wavelet.

Generate a set of random numbers (say 200 values either gaussian or uniform distributed). Convolve with your wavelet. This is your seismogram.

Form the deconvolution operator. You could use lpc, as before. Compare with the exact result (a filter having zeros at 1.5i and –1.5i, in this example. A

Convolve with the true wavelet to see how well it works as an inverse. Also plot the zeros of the filter and the poles of your wavelet on the same graph, to compare.

Compare the estimated reflection coefficients with the true reflection coefficients.

Find the exact inverse to your wavelet, and find a least squares approximate inverse for a length of 3 and 5.

Matlab Least Square Filter Design, PEF and Modeling functions

ARYULE AR parameter estimation via Yule-Walker method.
    A = ARYULE(X,ORDER) returns the polynomial A corresponding to the AR
    parametric signal model estimate of vector X using the Yule-Walker
    (autocorrelation) method. ORDER is the model order of the AR system.
    This method solves the Yule-Walker equations by means of the Levinson-
    Durbin recursion.

    [A,E] = ARYULE(...) returns the final prediction error E (the variance
    estimate of the white noise input to the AR model).

    [A,E,K] = ARYULE(...) returns the vector K of reflection coefficients.

    See also PYULEAR, ARMCOV, ARBURG, ARCOV, LPC, PRONY.

ARBURG AR parameter estimation via Burg method.
    A = ARBURG(X,ORDER) returns the polynomial A corresponding to the AR
    parametric signal model estimate of vector X using Burg's method.
    ORDER is the model order of the AR system.

    [A,E] = ARBURG(...) returns the final prediction error E (the variance
    estimate of the white noise input to the AR model).

    [A,E,K] = ARBURG(...) returns the vector K of reflection
    coefficients (parcor coefficients).

    See also PBURG, ARMCOV, ARCOV, ARYULE, LPC, PRONY.

LPC Linear Predictor Coefficients.
    A = LPC(X,N) finds the coefficients, A=[1 A(2) ... A(N+1)],
    of an Nth order forward linear predictor

       Xp(n) = -A(2)*X(n-1) - A(3)*X(n-2) - ... - A(N+1)*X(n-N)

    such that the sum of the squares of the errors

       err(n) = X(n) - Xp(n)

    is minimized. X can be a vector or a matrix. If X is a matrix
    containing a separate signal in each column, LPC returns a model
    estimate for each column in the rows of A. N specifies the order
    of the polynomial A(z).

    If you do not specify a value for N, LPC uses a default N = length(X)-1.

    [A,E] = LPC(X,N) returns the variance (power) of the prediction error.

    LPC uses the Levinson-Durbin recursion to solve the normal equations
    that arise from the least-squares formulation. This computation
    of the linear prediction coefficients is often refered to as the
    autocorrelation method.

    See also LEVINSON, ARYULE, PRONY, STMCB.

STMCB Compute linear model via Steiglitz-McBride iteration
    [B,A] = stmcb(X,NB,NA) finds the coefficients of the system
    B(z)/A(z) with approximate impulse response X, NA poles and
    NB zeros.

    [B,A] = stmcb(X,NB,NA,N) uses N iterations. N defaults to 5.

    [B,A] = stmcb(X,NB,NA,N,Ai) uses the vector Ai as the initial
    guess at the denominator coefficients. If you don't specify Ai,
    STMCB uses [B,Ai] = PRONY(X,0,NA) as the initial conditions.

    [B,A] = STMCB(X,U,NB,NA,N,Ai) finds the system coefficients B and
    A of the system which, given U as input, has X as output. N and Ai
    are again optional with default values of N = 5, [B,Ai] = PRONY(X,0,NA).
    X and U must be the same length. See also PRONY, LEVINSON, LPC and ARYULE.

PRONY Prony's method for time-domain IIR filter design.
    [B,A] = PRONY(H, NB, NA) finds a filter with numerator order
    NB, denominator order NA, and having the impulse response in
    vector H.   The IIR filter coefficients are returned in
    length NB+1 and NA+1 row vectors B and A, ordered in
    descending powers of Z. H may be real or complex.
    See also STMCB, LPC, BUTTER, CHEBY1, CHEBY2, ELLIP, INVFREQZ.

LEVINSON Levinson-Durbin Recursion.
    A = LEVINSON(R,N) solves the Hermitian Toeplitz system of equations

        [R(1)   R(2)* ... R(N)*] [A(2)] = [-R(2)]
        [R(2)   R(1) ... R(N-1)*] [A(3)] = [-R(3)]
        [.   .   .   ] [.   ] = [.   ]
        [R(N-1) R(N-2) ... R(2)*] [A(N)] = [-R(N)]
        [R(N) R(N-1) ... R(1)] [A(N+1)] = [-R(N+1)]

    (also known as the Yule-Walker AR equations) using the Levinson-
    Durbin recursion. Input R is typically a vector of autocorrelation
    coefficients with lag 0 as the first element.

    N is the order of the recursion; if omitted, N = LENGTH(R)-1.
    A will be a row vector of length N+1, with A(1) = 1.0.

    [A,E] = LEVINSON(...) returns the prediction error, E, of order N.

    [A,E,K] = LEVINSON(...) returns the reflection coefficients K as a
    column vector of length N. Since K is computed internally while
    computing the A coefficients, then returning K simultaneously
    is more efficient than converting A to K afterwards via TF2LATC.

    If R is a matrix, LEVINSON finds coefficients for each column of R,
    and returns them in the rows of A
    See also LPC, PRONY, STMCB.

XCORR Cross-correlation function estimates.
    C = XCORR(A,B), where A and B are length M vectors (M>1), returns
    the length 2*M-1 cross-correlation sequence C. If A and B are of
    different length, the shortest one is zero-padded. C will be a
    row vector if A is a row vector, and a column vector if A is a
    column vector.

    XCORR produces an estimate of the correlation between two random
    (jointly stationary) sequences:
           C(m) = E[A(n+m)*conj(B(n))] = E[A(n)*conj(B(n-m))]
    It is also the deterministic correlation between two deterministic
    signals.

    XCORR(A), when A is a vector, is the auto-correlation sequence.
    XCORR(A), when A is an M-by-N matrix, is a large matrix with
    2*M-1 rows whose N^2 columns contain the cross-correlation
    sequences for all combinations of the columns of A.
    The zeroth lag of the output correlation is in the middle of the
    sequence, at element or row M.

    XCORR(...,MAXLAG) computes the (auto/cross) correlation over the
    range of lags: -MAXLAG to MAXLAG, i.e., 2*MAXLAG+1 lags.
    If missing, default is MAXLAG = M-1.

    [C,LAGS] = XCORR(...) returns a vector of lag indices (LAGS).

    XCORR(...,SCALEOPT), normalizes the correlation according to SCALEOPT:
      'biased'   - scales the raw cross-correlation by 1/M.
      'unbiased' - scales the raw correlation by 1/(M-abs(lags)).
      'coeff'   - normalizes the sequence so that the auto-correlations
                   at zero lag are identically 1.0.
      'none'   - no scaling (this is the default).

    See also XCOV, CORRCOEF, CONV, COV and XCORR2.