Lecture 17:

Recap

We've now spent some time looking closely at coherence and how to assign uncertainties to coherence. Can we think about coherence in a different way?

There are a couple more things we want to do. First, we finished up last time with a quick look at a variance-preserving spectrum, and we want to understand how to make sense of that. And then, let's

Variance preserving spectra

One of the virtues of the properly normalized spectrum is that the area under the curve should represent the signal variance within a specific frequency band:

$$\text{variance in a band} = \int_{f-\Delta f/2}^{f+\Delta f/2} |X(f)|^2 \, df \tag{1}$$

This sounds good as a concept, but in log-log space, it's challenging to figure out what the area under the curve really represents. Variance-preserving spectra provide a visual representation for this. Consider a spectrum $S_{xx}(f)$ derived from our Fourier amplitudes $X(f)$. In log space, the area under our curves is a pseudo-variance s_*^2:

$$s_*^2 = \int_{f-\Delta f/2}^{f+\Delta f/2} \log(S_{xx}(f)) \, df \tag{2}$$

If we want to plot something that is more directly representative of variance, we can try this:

$$s^2 = \int_{f-\Delta f/2}^{f+\Delta f/2} S_{xx}(f) \, df = \int_{f-\Delta f/2}^{f+\Delta f/2} f S_{xx}(f) d[\log(f)] \tag{3}$$

This means that instead of plotting log$S$ vs log(f), we could plot $fS_{xx}(f)$ in linear space vs f in log space. This is especially useful for features that have strong peaks not exactly at the lowest frequency.

Here's an example, using the red noise that we started with earlier:

```
bb=reshape(b,500,10000/500);
fbb=fft(detrend(bb));
amp=abs(fbb(1:251,:)).^2 / 500;
sbb=mean(amp,2);
sbb(2:250)=sbb(2:250)*2;

subplot(2,1,1)
loglog(0:250,sbb,'LineWidth',3)
xlabel('Frequency','FontSize',14)
ylabel('(units^2)/frequency','FontSize',14)

subplot(2,1,2)
semilogx(0:250,(0:250)'.*sbb,'LineWidth',3)
xlabel('Frequency','FontSize',14)
ylabel('(units^2)','FontSize',14)
```

Multi-tapers and spectral peaks

Spectra can come in two flavors. Some have distinct single peaks, associated with tides. Some have large-scale structure associated with the general red structure of the ocean. If we want to find exactly the right peaks, then we can try different strategies to what we use when we want to find the general structure.

When we have narrow peaks, they aren't always easy to differentiate, particularly if our sampling is a bit coarse compared with the signals we'd like to detect. Consider the following case of a sinusoidal cycle that might or might not be well sampled, depending how long our instruments survive:

```
time=1:.5:120;
A=2*cos(2*pi*time/30)+cos(2*pi*time/60);
B=A(1:200);
C=[A(1:200) zeros(1,40)];

plot(time,A,time(1:200),B,'LineWidth',3)
set(gca,'FontSize',16)
xlabel('time','FontSize',16)
ylabel('amplitude','FontSize',16)

fA=fft(A);
fB=fft(B);
fC=fft(C);

frequency1=(0:120)/120;
frequency2=(0:100)/100;

loglog(frequency1,abs(fA(1:121)).^2,frequency2,abs(fB(1:101)).^2,...
  frequency1,abs(fC(1:121)).^2,'LineWidth',3)
set(gca,'FontSize',16)
xlabel('frequency','FontSize',16)
ylabel('spectral density','FontSize',16)
legend('full record','truncated record','zero padding')
```

When you look at this example, you might conclude that without perfect sampling of full sinusoidal cycles, we'll never find the correct spectral peaks. In essence this is a windowing problem. When we have narrow peaks, they aren't always easy to differentiate, particularly if our sampling is a bit coarse compared with the signals we'd like to detect. See the slides for an example (from Rob Pinkel.)

If we don't have adequate resolution what are our options?

1. Possibility 1. Pad the short record with zeros to make it as long as we want. Since resolution is $f = 1/(N\Delta t)$. In this case, we'll see the impact of a sinc function bleeding into the frequencies that we'd like to resolve. Clearly this doesn't fully solve our problem.

2. Possibility 2. Obtain a longer record. This will be critical if we really want to resolve our signal.

Even if our record is norminally long enough, we also need to figure out how to optimize our detection of spectral peaks. Earlier we looked at the impact of windows, and examples from the Harris (1978) study showed how much impact a good windowing strategy can have in identifying spectral peaks. (For continuous spectra, windowing approaches work well.)

Formally, you'll recall that we can represent our record length problems using a convolution of our data with a finite width filter:

$$\hat{X}(f_n) = \int_{-\infty}^{\infty} X(f_m) W(f_n - f_m)\, df_m, \tag{4}$$

where

$$W(f) = \frac{\sin(2\pi f T)}{2\pi f T} = \text{sinc}\,(2\pi f T) \tag{5}$$

This means that the spectrum is essentially convolved with $W(2\pi f T)^2$. But as we noted earlier, we can switch from a boxcar window to a triangle window or something a bit more Gaussian than that and cut down on the sidelobes in our window to obtain a cleaner spectrum, although we have to widen the central peak of the window in the frequency domain, which means de-emphaizing the beginning and end of the data series.

What if we want to improve our resolution. Consider Rob Pinkel's example of a record equivalent to

$$x = 100\cos(2\pi 20.5/10001) + 80\cos(2\pi 30.4/100001) + 100\cos(2\pi 40.8/100001) + 10\cos(2\pi 50.3/100001) \tag{6}$$

The quality of our spectral estimate will depend on the length of our record. (Why is that? The resolution is the lowest resolved frequency.) So what can we do to improve resolution. One strategy would be to pad our record with zeros to make it as long we want. That buys us something, but it gives us plenty of spectral ringing.

If we want to optimize resolution, we can try a multitaper approach. (See for example Ghil et al, Reviews of Geophysics, 2001). In a multitaper approach, we replace our single window with a set of tapers. The tapers are designed to minimize spectral leakage, and they are referred to as "discrete prolate spheroidal sequences" or "Slepian" tapers (after Slepian, who studied them). Tapers are what we've been calling windows—they pre-multiply the data, Fourier transforms are computed, and then the spectrum is computed as a weighted sum of all of the squared Fourier transforms. This effectively averages over an ensemble of windows to minimize variance. This is very effective for extracting narrow peaks that would otherwise be undetectable. Matlab has a multi-taper method package ('pmtm'), but if you really want this to work, you probably want to dig into the guts of the algorithm a bit further.

Here's the Matlab example, modified slightly to make a longer record:

```
fs = 1000;                  % Sampling frequency
t = (0:3*fs)/fs;              % One second worth of samples
A = [1 2];                  % Sinusoid amplitudes
f = [150;140];              % Sinusoid frequencies

xn = A*sin(2*pi*f*t) + 0.1*randn(size(t));
pmtm(xn,4,[],fs)
```

This produces an impressive two spectral peaks. Of course this example isn't too tricky. Here's what we get if we take the same data and split them into 6 non-overlapping segments, even with no windowing or detrending:

```
fa=fft(reshape(xn(1:3000),500,6));
semilogy(mean(abs(fa(1:250,:)').^2))
```

These are reassuringly similar results.

Decibels and Powers of 10

Decibels ("db", not to be confused with decibars) quantify power or variance. We often focus on orders of magnitude. But power in decibels is on a $\log_{10}$ scale, with a factor of ten normalization.

$$P_{db} = 10\log_{10}(P/P_0), \tag{7}$$

where P_0 is a reference level of power: Variance is a squared quantity so

P/P_0	P relative to P_0 (db)
1000	30
10	10
2	3
0.5	-3
0.1	-10

$$P_{db} = 20\log_{10}(V/V_0). \tag{8}$$

By convention dB is used for sound pressure and db for everything else.

Transfer function:

We discussed the fact that coherence is analogous to a correlation coefficient. It tells us if two things vary in tandem in a consistent way, but it doesn't tell us how big they are or how to use one variable to approximate a second variable. If we want to look at relative sizes, or if we want to approximate a variable y based on its relationship with x, in the time domain we think about finding a regression coefficient. We're used to solving a least-squares fitting equation of the form

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{9}$$

with a solution of the form

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x} \tag{10}$$

For illustration we can simplify this to a case where the matrix $\mathbf{A}$ has only column, that is where we regress $\mathbf{y}'$ (with the prime telling us that the mean has been removed, since we don't want to complicate our least-squares fit) against just one variable, making $\mathbf{A}$ a column vector (e.g. $\mathbf{z}'$). In this case x becomes a scalar and the matrix inverse $(\mathbf{A}^T\mathbf{A})^{-1}$ is just the reciprocal of the variance of $\mathbf{z}'$.

$$x = \frac{\langle y'z' \rangle}{\langle z'^2 \rangle}. \tag{11}$$

Compare this with the correlation coefficient of the demeaned variables

$$r = \frac{\langle y'z' \rangle}{\sqrt{\langle y'^2 \rangle \langle z'^2 \rangle}}. \tag{12}$$

Viewed in this way, the correlation coefficient r and the regression coefficient x (another term for the least-squares fit with only one variable) look nearly the same, aside from the normalization. Both the correlation coefficient and the regression coefficient convey useful information. And that might make you think that in the Fourier-transform domain there should be a form analogous to regression.

The term for the Fourier domain analog to regression is the transfer function:

$$\hat{H}_{xy}(f) = \frac{\hat{G}_{xy}(f)}{\hat{G}_{xx}(f)}, \tag{13}$$

which provides a (complex-numbered) recipe for mapping from x to y.

Formally, we talk about the transfer function when we think about constructing a linear system:

$$\mathcal{L}(y(t)) = x(t) \tag{14}$$

If $\mathcal{L}$ is a linear operator, then we could think of this relationship as a convolution:

$$y_t = \int_{-\infty}^{\infty} h(u)x(t-u)\,du \tag{15}$$

or if we Fourier transform, this would state:

$$Y(f) = H(f)X(f). \tag{16}$$

Salinity spiking examples

In class I showed some examples of 'salinity spiking', which results from the temperature and conductivity sensors having different response times. If we assume that simultaneous measurements represent the same water sample, when in reality they don't, we end up computing:

$$S = S(T(t), C(t-\Delta t)). \tag{17}$$

If temperature and salinity are constant, this doesn't pose a problem, but when T or C change abruptly, this can lead to very odd results. How can we use the transfer function (or coherence) approach to examine this (even if we don't know the actual response characteristics of the sensor)?

Appendix: More detail on variance of cross-spectra

If we have two degrees of freedom, the cross spectrum is

$$\begin{aligned} |\hat{G}_{XY}(f)|^2 &= \hat{G}^*_{XY}\hat{G}_{XY} && (18)\\ &= |X^*(f)Y(f)|^2 && (19)\\ &= X(f)Y^*(f)X^*(f)Y(f) && (20) \end{aligned}$$

So trying all combinations to get the sum of the 4-term product:

$$\begin{aligned} \langle|\hat{G}_{XY}(f)|^2\rangle &= \langle XX^*\rangle\langle YY^*\rangle + \langle XY^*\rangle\langle X^*Y\rangle + \langle XY\rangle\langle X^*Y^*\rangle && (21)\\ &= G_{XX}G_{YY} + |G_{XY}|^2 && (22) \end{aligned}$$

where G here is the total cross-spectrum. The variance is then

$$\begin{aligned} \text{var}[\hat{G}_{XY}] &= \langle|\hat{G}_{XY}(f)|^2\rangle - |\hat{G}_{XY}(f)|^2 && (23)\\ &= G_{XX}G_{YY} && (24)\\ &= \frac{|G_{XY}|^2}{\gamma^2_{xy}} && (25) \end{aligned}$$

where γ_{xy} is the coherence.

With more degrees of freedom, error scales with the square root of the number of samples, just like the standard error:

$$\text{var}G_{XX} = \frac{G_{xx}^2}{n_d} \tag{26}$$

$$\text{var}G_{YY} = \frac{G_{YY}^2}{n_d} \tag{27}$$

$$\text{var}G_{XY} = \frac{|G_{XY}|^2}{\gamma_{xy}^2 n_d} \tag{28}$$

This scaling gives us the uncertainty for the coherence and phase after some manipulation. (See Bendat and Piersol, Ch. 9 for details.)

This means that the normalized uncertainty of G_{XY} is

$$\epsilon[|\hat{G}_{XY}|] = \frac{\text{std}[G_{XY}]}{G_{XY}} = \frac{1}{|\gamma_{xy}|\sqrt{n_d}} \tag{29}$$

Using cross covariance to derive cross-spectra

Finally, you might be thinking that if we could use the auto-covariance to derive the spectrum, we should also be able to use the cross-covariance between two different variables to compute the cross-spectrum. Bendat and Piersol are helpful on this topic (see section 5.2).

Here's a test Matlab code to see how this works:

```
% define two random data sets with some shared information
a=randn(10000,1);
b=a+randn(10000,1);

% compute autocovariance and cross-covariance
cab=xcov(a,b);
caa=xcov(a,a);
cbb=xcov(b,b);

% Fourier transform
fcab=fft(cab(10000-500:10501));
fcaa=fft(caa(10000-500:10501));
fcbb=fft(cbb(10000-500:10501));

% compute coherence and plot
coher=abs(fcab(1:501)).^2 ./abs(fcaa(1:501)) ./abs(fcbb(1:501));
semilogx(0:500,coher)

% for comparison do same procedure with segments (here
%  non-overlapping, non-windowed, quick and dirty)
% Fourier transform
faa2=fft(reshape(a,1000,10));
fbb2=fft(reshape(b,1000,10));
```

```
% compute spectra and cross spectrum
saa=mean(abs(faa2(1:500,:)).^2,2);
sbb=mean(abs(fbb2(1:500,:)).^2,2);
sab=mean(faa2(1:500,:).*conj(fbb2(1:500,:)),2);

% plot on top of other
hold on
semilogy(0:499,abs(sab).^2 ./(saa.*sbb))
```