Lecture 14: Frequency-wavenumber spectra, cross-coherence, and cross-spectra

Reading: Bendat and Piersol, Ch. 5.1-5.2, with attention to cross-covariance and cross-spectrum

Recap

Last time we looked at computing the spectrum from the autocovariance, at variance-preserving spectra, and at frequency-wavenumber spectra. Today we continue on frequency-wavenumber spectra and then move on to covariance and coherence.

Frequency-Wavenumber spectra: an example

Let's suppose that we have a propagating signal. We can generate an artificial signal of the form:

```
H=1000;

dt=0.05; %time interval about 20 times a day
dz=10; %10-m
t=dt:dt:10; %time in days
z=(dz:dz:H)'; %depth in m

%indices for the calculation
iz=1:length(z);
it=1:length(t);

%Generate a propagating signal with no noise.
[tt,zz]=meshgrid(t,z);
sig1=sin(3*pi*zz/H - 2*pi*1/3*tt);
%Change the sign to make propagation go the other way.

data=sig1;

% plot
figure(2)
imagesc(t,z,data);colorbar; axis xy
```

This will show a propagating signal.

To compute a frequency-wavenumber spectrum for this, we just need to Fourier transform in two dimensions. Since this is a noise free case, we can be fairly cavalier about how we approach it.

```
[m,n]=size(data);

fn=1/2/dt;
kn=1/2/dz;

%fundamental frequency and wavenumber
df=1./n./dt;
dk=1./m./dz;
```

```
% make frequencies and wavenumbers that run from -Nyquist to + Nyquist
f=[-fliplr(1:(n/2)) 0 (1:(n/2-1))].*df;
k=[-fliplr(1:(m/2)) 0 (1:(m/2-1))].'.*dk;

% Fourier transform in two dimensions
%   here we use fft2 for the 2-d Fourier transform
%   and fftshift to reorder the Fourier transform
st=fftshift(fft2(data))/m/n;

% alternatively you could do this as
st= fftshift(fft(fft(data')'))/m/n;

% turn this into a spectrum
spec=st.*conj(st)./df./dk; %UNITS: (m/s)^2/cpd/cpm

% and plot
figure(3)
imagesc(f,k,log10(spec)); axis xy
colormap(jet)
shg
xlabel('\omega / cpd')
ylabel('m / cpm')

% this plots the full 2-d plane

% But you might want a half plane.  In that case, you should
% scale by a factor of 2:
spec=spec(:,101:200); spec(:,102:200)=2*spec(:,102:200);
imagesc(f(101:200),k,log10(spec)); axis xy
colormap(jet)
shg
xlabel('\omega / cpd')
ylabel('m / cpm')
```

Now what happens when you have noise and need to segment. Let's make a larger data set and chop it up into pieces:

```
t=dt:dt:30; %time in days
z=(dz:dz:H)'; %depth in m

%indices for the calculation
iz=1:length(z);
it=1:length(t);

%Generate a propagating signal with no noise.
[tt,zz]=meshgrid(t,z);
sig1=sin(3*pi*zz/H - 2*pi*1/3*tt) + .5*randn(length(z),length(t));
```

```
data=sig1;

% plot
figure(2)
imagesc(t,z,data);colorbar; axis xy
```

To chop this into segments, we have choices to chop in time or space or both.

```
icount=0; clear st;
for i=1:50:501
 icount=icount+1;
 data_use=data(:,i:i+99);
 n_use=size(data_use,2);
 st(:,:,icount)=fftshift(fft2(data_use))/m/n_use;
end

spec=sum(abs(st).^2/df/dk,3);
f=[-fliplr(1:(n_use/2)) 0 (1:(n_use/2-1))].*df;
figure(3)
imagesc(f,k,log10(spec)); axis xy
colormap(jet)
shg
xlabel('\omega / cpd')
ylabel('m / cpm')
```

Interestingly, we can achieve the averaging that we need, either by segmenting in time or in space or in both directions. We do have to be careful about detrending, and we need to pay attention to our windowing strategy.

Covariance

Early in the quarter we discussed the variance, and we left for later the concept of correlation or covariance. If we want to compare two time series, we can compute the variance of one record relative to the other. Formally we can write:

$$\text{cov}(x, y) = \langle x(t)y(t) \rangle. \tag{1}$$

or in discrete terms

$$\text{cov}(x, y) = \frac{1}{N} \sum_{i=1}^{N} x_i y_i. \tag{2}$$

For comparison purposes, we often normalize this to produce a correlation coefficient, which is normalized by the variance:

$$r = \frac{\frac{1}{N} \sum_{i=1}^{N} x_i y_i}{\sqrt{\frac{1}{N} \sum_{i=1}^{N} x_i^2 \frac{1}{N} \sum_{i=1}^{N} y_i^2}}. \tag{3}$$

(You might wonder how to judge whether a correlation coefficient is statistically significant. Correlation coefficients should have a Gaussian distribution, which means that cumulative distribution

function will be an error function. We can use this to determine the correlation coefficient that we might expect from an equivalent number of random white noise variables:

$$\delta r = \text{erf}^{-1}(p)\sqrt{\frac{2}{N}} \tag{4}$$

where p is the significance level we want to consider, typically 0.95, and N is the effective number of degrees of freedom.)

Coherence

Coherence provides information that is analagous to a correlation coefficient for Fourier transforms. It tells us whether two series are statistically linked at any specific frequency. This can be important if we think that the records are noisy or otherwise uncorrelated at some frequencies, but that they also contain statistically correlated signals.

To compute coherence, first we need a cross-spectrum. (We looked at this in passing when we considered Parseval's theorem, but at that stage, I quickly set my different variables equal to each other.) Consider two time series $x(t)$ and $y(t)$:

$$x(t) = \sum_{n=-\infty}^{\infty} X_n e^{i2\pi f_n t} \tag{5}$$

$$y(t) = \sum_{n=-\infty}^{\infty} Y_n e^{i2\pi f_n t} \tag{6}$$

The the cross spectrum is computed in analogy with the spectrum:

$$\hat{S}_{XY}(f_m) = \frac{\langle X_m^* Y_m \rangle}{T} \tag{7}$$

The relationship between the cross-spectrum and the covariance is analogous to the relationship between the spectrum and the variance. There are some important details to notice.

1. The cross spectrum is complex, while the spectrum was real.
2. The cross spectrum is computed as an average of multiple spectral segments.
3. In our discrete Fourier transform, we should be normalizing by N, as always, but we're mostly concerned with relative values.

The cross-spectrum is complex, and when we use it we distinguish between the real and imaginary parts. The real part is called the "co-spectrum":

$$C(f_k) = \frac{1}{N}\Re\sum_{n=1}^{N}(X_k Y_k^*) \tag{8}$$

and the imaginary part is called the "quadrature spectrum"

$$Q(f_k) = \frac{1}{N}\Im\sum_{n=1}^{N}(X_k Y_k^*). \tag{9}$$

To determine the frequency-space relationship between two data sets x_n and y_n, we first divide them into segments and Fourier transform them, so that we have a set of X_k's and a set of Y_k's. When we computed spectra, we found the amplitude of each X_k and then summed over all our segments. Now we're going to do something slightly different. For each segment pair, we'll compute the product of X times the complex conjugate of Y: $X_k Y_k^*$. Then we'll sum over all the segments. In Matlab this becomes

```
sum(X.*conj(Y),2);
```

The corresponding amplitude is $\sqrt{C^2(f_k) + Q^2(f_k)}$. For comparison the spectra for X was:

$$S_{xx}(f_k) = \frac{1}{N} \sum_{n=1}^{N} X_k X_k^*, \tag{10}$$

and it was always real.

The coherence resembles a correlation coefficient. It's the amplitude squared divided by the power spectral amplitudes for each of the two components:

$$\gamma_{xy}^2(f_k) = \frac{C^2(f_k) + Q^2(f_k)}{S_{xx}(f_k) S_{yy}(f_k)}. \tag{11}$$

(Sometimes you'll see G_{xx}, G_{yy}, and G_{xy} in place of S_{xx}, S_{yy}, and S_{xy}. Bendat and Piersol define S to represent the two sided cross-spectra density and G to represent to represent one-sided spectra.)

In addition to the coherence amplitude, we can also infer a phase. The phase $\phi(f_k) = \tan^{-1}(-Q(f_k)/C(f_k))$ tells us the timing difference between the two time series. If $\phi = 0$, changes in x and y happen at the same time. If $\phi = \pi$, then x is at a peak when y is at a trough. And a value of $\phi = \pi/2$ or $\phi = -\pi/2$ tells us that the records are a quarter cycle different.

Digression: Extracting phase information from the Fourier coefficients

After all this effort to square Fourier coefficients, you might wonder what the real and imaginary parts are really good for. They are useful for sorting out the phasing of your sinusoidal oscillations. When is the amplitude at a maximum? To do this you can keep in mind that

$$A\cos(\sigma t + \phi) = a\cos(\sigma t) + b\sin(\sigma t). \tag{12}$$

This can be rewritten:

$$\cos(\sigma t)\cos(\phi) - \sin(\sigma t)\sin(\phi) = \frac{a}{A}\cos(\sigma t) + \frac{b}{A}\sin(\sigma t), \tag{13}$$

which means that

$$\frac{a}{A} = \cos(\phi) \tag{14}$$

$$\frac{b}{A} = -\sin(\phi) \tag{15}$$

so

$$\phi = \text{atan}\left(-\frac{b}{a}\right). \tag{16}$$

Actually there's more information in the Fourier coefficients than this conveys, since you know the signs of both a and b, and not just their relative magnitudes. The arctangent function doesn't distinguish +45°from -135°, but we can. In some numerical implementations, you can address this using a function called atan2.

```
phi = atan2(-b,a);
```

Coherence: Hypothetical Example for Mission Bay Channel

The power of coherence comes because it gives us a means to compare two different variables. With spectra we can ask, is there energy at a given frequency? With coherence we can ask whether wind energy at a given frequency drives an ocean response at a given frequency. Does the ocean respond to buoyancy forcing? Does momentum vary with wind? Does one geographic location vary with another location? Coherence is our window into the underlying physics of the system.

Let's put this to work, starting with an idealized case: Suppose we want to estimate currents entering and leaving the Mission Bay Channel. How do waves travel through the channel? We can represent this with a dispersion relationship describing the dominant propagation in frequency-wavenumber space: $k = K(f)$.

You could imagine measuring Mission Bay by installing one current meter (with a cost of $10-$20,000), but another approach is to install a couple of pressure recorders along the axis of the channel (at a cost of $1000 each). Let's assume all waves come from the ocean, and travel along the channel axis at speed $V = c + U_{current}$, where c is the wave speed and $U_{current}$ the background current speed. If the waves are surface gravity waves, $c = \sqrt{gD}$. The sensors measure time series of pressure only, so provide frequency information f. How does f relate to velocity? If we have a wavenumber $2\pi k = 2\pi/\lambda$, what does the pressure sensor see? It will detect frequencies $f = kV = k(c + U_{current})$. So we can compute the cross spectrum between our two records.

Let's test this out. We'll define a hypothetical data set:

```
lambda=10; % 10 m wavelength
V=0.3; % 0.3 m/s propagation
n2s=0.2; % noise-to-signal ratio
time=(1:5000)';
x=n2s*randn(5000,1)+cos(2*pi/lambda*V*time);
y=n2s*randn(5000,1)+cos(2*pi/lambda*V*(time)+pi/2);
```

What happens if you Fourier transform without bothering to segment? Then the data end up being unrevealing. We can demonstrate this:

```
fx=fft(x);
fy=fft(y);
sx=abs(fx(1:end/2)).^2; sx(2:end)=2*sx(2:end);
sy=abs(fy(1:end/2)).^2; sy(2:end)=2*sy(2:end);
cxy=conj(fx(1:end/2)).*fy(1:end/2); cxy(2:end)=2*cxy(2:end);
C=abs(cxy)./sqrt(sx.*sy);
plot(C)
```

In this case, the coherence is 1 everywhere. Why is that? Because without averaging, we're merely computing:

$$\gamma_{xy}^2 = \frac{(X^*Y)^*(X^*Y)}{X^*XY^*Y} \tag{17}$$

$$= \frac{XY^*X^*Y}{X^*XY^*Y} \tag{18}$$

$$= \frac{X^*XY^*Y}{X^*XY^*Y} \tag{19}$$

$$= 1 \tag{20}$$

When it's done properly, coherence measures how well different segments of x and y show the same type of relationship at a given frequency. We need the averaging to find out if the phase relationship between x and y is repeatable. With only one segment, both x and y are guaranteed to have information at each frequency with a definable phase relationship between x and y. The multiple segments allow us to test whether this phase relationship is relatively stable in time: does x always lead y by about the same fraction of a cycle?

To do the coherence calculation more constructively, we determine the frequency-space relationship between two data sets x_n and y_n, by first dividing them into segments and then Fourier transforming them, so that we have a set of X_k's and a set of Y_k's. When we computed spectra, we found the amplitude of each X_k and then summed over all our segments. Now we're going to do something slightly different. For each segment pair, we'll compute the product of X times the complex conjugate of Y: $X_k Y_k^*$. Then we'll sum over all the segments. In Matlab this becomes:

```
segment_length=500;
N=length(x);
M=segment_length/2; % define this value
Nseg=N/segment_length;
x_use=[reshape(x,segment_length,Nseg) ...
   reshape(x(M+1:end-M),segment_length,Nseg-1)];
y_use=[reshape(y,segment_length,Nseg) ...
  reshape(y(M+1:end-M),segment_length,Nseg-1)];
Nuse=size(x_use,2); % segment count, should be 2*Nseg-1
fx=fft(x_use); % should window and detrend here, but we're
               % skipping that for now
fy=fft(y_use);
sx=sum(abs(fx(1:M+1,:)).^2,2)/Nuse; % average over all spectra
                                    % (sum over 2nd index)
sx(2:end)=sx(2:end)*2;
sy=sum(abs(fy(1:M+1,:)).^2,2)/Nuse; % average over all spectra
                                    % (sum over 2nd index)
sy(2:end)=sy(2:end)*2;
cxy=sum(fx(1:M+1,:).*conj(fy(1:M+1,:)),2)/Nuse;
cxy(2:end)=cxy(2:end)*2;  % since we multiplied the spectra by 2,
                      % we also need to multiply the cospectrum by 2

nd=size(x_use,2);
```

From this we can compute the coherence and phase:

```
C=abs(cxy)./sqrt(sx.*sy);
phase_C = atan2(-imag(cxy),real(cxy));
```

The phase difference that emerges from this is only relevant at the phase where there is coherence energy (15 cycles/1000 points in the example above), and in that case the phase is a quarter cycle different, with relatively small error bars. If we reverse the order of x and y, we'll find negative phase, so a lead will turn into a lag. ‘

The phase $\phi(f_k) = \tan^{-1}(-Q(f_k)/C(f_k))$ tells us the timing difference between the two time series. If $\phi = 0$, changes in x and y happen at the same time. If $\phi = \pi$, then x is at a peak when y is at a trough. And a value of $\phi = \pi/2$ or $\phi = -\pi/2$ tells us that the records are a quarter cycle different.