Lecture 13: Autocovariance and Spectra, Variance-Preserving Spectra, and Frequency/Wavenumber Spectra

Recap

We've looked at multiple strategies for computing spectra in the frequency domain or by extension in the wavenumber domain. We've considered multiple methods and have discussed uncertainties, resolution, and aliasing. Now let's take a practical look at the autocovariance approach.

A practical look at the autocovariance

Let's think about the autocovariance of white noise:

$$R_{xx}(\tau) = \int_{-\infty}^{\infty} x(t)x(\tau + t)dt = \begin{cases} 0 & \text{for } \tau \neq 0 \\ 1 & \text{for } \tau = 0 \end{cases} \tag{1}$$

This is true, because white noise is uncorrelated except at zero lag.

Alternatively, if I consider red noise, then the noise will be correlated from point to point, and the autocovariance will have a bit of spread. We can test this out:

```
% define red data with autoregressive process
a=randn(10000,1);
b(1)=a(1);
for i=2:length(a)
  b(i)=b(i-1)+a(i);
end

BB=xcorr(b,b)/max(xcorr(b,b));
BB_unbiased=xcorr(b,b,'unbiased')/max(xcorr(b,b,'unbiased'));
plot(-999:999,[BB' BB_unbiased'],'LineWidth',3)
```

Biased vs unbiased estimators.

Notice that we could consider both the 'biased' and the "unbiased" estimator. There are arguments for either choice. The difference depends on how we normalize our discrete autocovariance. In the unbiased case, we're computing

$$R(\tau)_{unbiased} = \frac{1}{N-m}\sum_{n=1}^{N-m} x(t_n)x(t_{n+m}). \tag{2}$$

In the biased case, we change how we normalize:

$$R(\tau)_{biased} = \frac{1}{N}\sum_{n=1}^{N} x(t_n)x(t_{n+m}), \tag{3}$$

which means that as the number of values we consider becomes smaller, we constrain the magnitude of the autocovariance by continuing to divide by N. Emery and Thomson note that the biased estimator acts like a triangle window. For spectra, you will want to stick to the unbiased estimator, and then choose your own window carefully.

Practical implementation: Spectra from auto-covariance

Using the auto-covariance to compute spectra requires averaging, just as we did by segmenting our data and using the fft, but there's one tidy little trick. Let's use some white noise again, and take a look at our options:

1. Suppose we start with a big matrix of white noise, and we compute the autocovariance for each column of our matrix, then Fourier transform, and use these to compute a spectrum. We'll end up doing something along these lines:

```
A=randn(1000,100);
for i=1:100
 AcA(:,i)=xcov(A(:,i),A(:,i),'unbiased');  % autocovariance for
end
fAcA=fft(AcA(500:1500,:)); % Fourier transform of autocovariance
frequency=(0:500)/1000;
loglog(frequency,2*abs(mean(fAcA(1:501,:),2)),'LineWidth',3)
set(gca,'FontSize',16)
xlabel('Frequency (cycles per data point)','FontSize',16)
ylabel('Spectral energy','FontSize',16)
```

   As with the standard Welch method for spectral estimation (which we can properly refer to as a "periodogram"), we're only plotting half the spectrum, so we need double the energy.

2. Alternatively, we could average all of the autocovariances, and then Fourier transform:

```
mean_AcA=mean(AcA,2);
fmean_AcA=fft(mean_AcA(500:1500));
hold on
loglog(frequency,2*abs(fmean_AcA(1:501,:))*1.1,'r','LineWidth',3)
legend('average of FFTs of many autocovariances',...
     'FFT of averaged autocovariance (scaled by 1.1)')
```

3. For comparison, the peridogram-based determined from the fft of the data:

```
fA=fft(A);
ampA=abs(fA(1:501,:)).^2/1000; ampA(2:500,:)=2*ampA(2:500,:);
loglog(frequency,mean(ampA,2),'LineWidth',3)
```

4. Finally, you might skip all segmenting and just compute the autocovariance of the full record:

```
N=length(a);
aca=xcov(a,a,'unbiased');
faca=fft(aca(N-500:N+500));
loglog(frequency,2*abs(faca(1:501)),'LineWidth',3)
```

In the results, shown in Figure 1, the curves (for options 1 and 2) are identical, though the red line has been scaled up by 10% to make both visible. There are some normalizations here that we haven't properly confronted. (Notably, if our frequency had units, we'd need to scale by T/N, or include a Δt.) Details can be sorted out later, and Thomson and Emery provide a bit of guidance on this.

You shouldn't be surprised that averaging before or after the FFT leads to the same results, since averaging has no impact on the FFT. This might give you ideas of how you can take advantage of the autocovariance to compute spectra from gappy data. Importantly, we can compute spectra without needing to chunk our data and compute lots of ffts, provided that we had a good estimate of the autocovariance.

The best estimate of the autocovariance should have the most averaging, and you can maximize your averages by using the longest possible records (option 4 above). Suppose you have 1000 data points, and two possible ways to analyze the data.

- If you split the record into 10 segments of 100 points, the zero-lag autocovariance will be an average of 1000 points, which is great, but the lag 100 autocovariance will represent an average of only 10 points, which is not so great.
- In contrast, if you don't split the record, the zero-lag autocovariance will still be an average of 1000 points, while the lag 100 auto-covariance will be an average of 900 points, on the whole a lot more averaging.

In the days before the development of the FFT, the autocovariance was a natural pathway to determining the spectrum, since it was clean and easy to compute. And now, with modern computing, you might not feel like there's any need to take advantage of the FFT anymore. If you can obtain the best possible estimate of the autocovariance, by whatever means necessary, then you should be able to compute one FFT and obtain reasonable estimate of the spectrum, without concern for data gaps or computational speed.

Formally if you compute spectra from the autocovariance, you need to think about averaging just as thoroughly as you do for segmented windowed data (through the Welch method) or for frequency-averaged spectra (the Daniell method). In this case, the challenge comes in deciding what fraction of the autocovariance to actually Fourier transform. If you use the autocovariance over the entire data range, your autocovariance estimator has too much uncertainty at the large lags, and the resulting spectrum will have large uncertainties.

When we use the autocovariance to compute spectra, we'll want to omit the poorly sampled edges of the spectrum. There are a number of ways to do this. We have to decide what fraction of the autocovariance to use. For example I could take half, or a quarter. What difference does it make? In essence, the fraction that I use determines the amount of averaging that I do, and therefore defines the number of degrees of freedom. In the simplest form, we use a boxcar window to extract values:

```
N=length(b);  % number of data points
M=N/4;  % half width of points to use
fBB=fft(BB(N-M+1:N+M));  % since the fft assumes the record to
              % be circular, remove one point at the end

loglog(0:M-1,abs(fBB(1:M)))
```

The degrees of freedom are determined by the width of the window. If I have N data points, and I use M of them in the Fourier transform, then I'm implicitly averaging N/M independent samples, which gives me N/M degrees of freedom. I can compute an error bar based on this in exactly the way that we did earlier for the periodogram approach:

```
hold on
nu=N/M;
err_high=nu/chi2inv(.05/2,nu);
```

```
err_low=nu/chi2inv(1-.05/2,nu);

loglog([10 10],[err_high err_low],'LineWidth',3)
```

Now you might decide that a boxcar windowed view of the autocovariance is likely to have unfortunate characteristics in the Fourier domain, in which case you could use a different window instead: a triangle window (also called a Bartlett window), or a Hanning window, or a Hamming window. For example:

```
fBB_t=fft(BB(N-M:N+M).*triang(2*M+1)');
fBB_h=fft(BB(N-M:N+M).*hanning(2*M+1)');

loglog(0:M-1,abs(fBB_t(1:M)))
loglog(0:M-1,abs(fBB_h(1:M)))
```

In this case we adjust the error bars to account for the windowing of the autocovariance. As it turns out, the table (5.5 in Thomson and Emery, 2014) that was so misleading for overlapping segmented data is exactly what we need here:

Window type	Equivalent degrees of freedom (ν)
Truncated peridogram (boxcar)	N/M
Bartlett (triangle)	$3N/M$
Daniell (sinc)	$2N/M$
Parzen	$3.708614N/M$
Hanning	$8/3N/M$
Hamming	$2.5164N/M$

This works well when M is small compared with N and when the autocovariance is comparatively narrow. In other words, for white noise, this converges nicely; for a broad red noise peak, the autocovariance tends to have negative lobes, making the choice of M difficult and the results not amenable to interpretation. The challenge in dealing with the truncation of the autocovariance is perhaps the reason that although this approach is often presented in equations (in idealized cases with infinite data), it is less frequently implemented for real applications.

Side bar: Autocovariance in discrete form. Last time we went through the representation of the autocovariance using an integral form. Let's rewrite this in terms of the discrete Fourier transform. In this case, the mean of our data is:

$$\langle x \rangle = \frac{1}{2T}\int_{-T}^{T} x(t)e^{i0}\,dt = a_0. \tag{4}$$

and the variance is

$$\langle x * x \rangle = \frac{1}{2T}\int_{-T}^{T} x^*(t)x(t)\,dt - |a_0|^2. \tag{5}$$

We use the complex conjugate here, just in case $x(t)$ is represented as a complex number, since this will give us the sum of the squares. Notice that we've remembered to subtract out the mean (our frequency zero Fourier coefficient).

In similar notation, we can write the covariance (for finite record length $2T$) as:

$$R(\tau) = \frac{1}{2T}\int_{-T}^{T} x^*(t)x(t+\tau)\,dt - |a_0|^2. \tag{6}$$

This lets us write out an expression for the variance R in terms of the discrete Fourier coefficients:

$$\begin{aligned}
R(\tau) &= \frac{1}{2T}\int_{-T}^{T}\left[\sum_{n=-\infty}^{\infty} a_n^* e^{-i2\pi f_n t}\sum_{m=-\infty}^{\infty} a_m e^{i2\pi f_m(t+\tau)}\right] dt - |a_0|^2 && (7)\\
&= \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} a_n^* a_m e^{+i2\pi f_m\tau}\frac{1}{2T}\int_{-T}^{T} e^{i(-2\pi f_n + 2\pi f_m)t}\,dt - |a_0|^2 && (8)\\
&= \sum_{m=-\infty}^{\infty} |a_m|^2 e^{+i2\pi f_m \tau} - |a_0|^2 && (9)
\end{aligned}$$

where we used a Kronecker delta (δ_{nm}) to eliminate the integral with $e^{2\pi i(-f_n+f_m)t}$ except when $n = m$, and we subtracted a_0^2 at the end to match our original definition. In setting this up, recall (from lecture 5) that the Fourier transform uses $e^{-i2\pi ft}$, and the inverse transform uses $e^{+i2\pi ft}$. We're using the inverse transform here (though the signs reversed when we had the complex conjugate. This tells us that the Fourier transform of the autocovariance can be expressed by the squared Fourier coefficients. (So we could avoid the Fourier transform completely and just work with the auto-covariance.)

In this form, Parseval's theorem simply says that

$$\begin{aligned}
R(0) &= \frac{1}{2T}\int_{-T}^{T} x^*(t)x(t+0)\,dt - |a_0|^2 && (10)\\
&= \sum_{m=-\infty}^{\infty} |a_m|^2 - |a_0|^2 && (11)
\end{aligned}$$

meaning that the variance of x is the sum of magnitudes of the Fourier coefficients.

Variance preserving spectra

Last time we said, that 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:

$$\text{variance in a band} = \int_{f-\Delta f/2}^{f+\Delta f/2} |X(f)|^2\,df \tag{12}$$

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.

That might not bother you, but sometimes that visual representation might seem helpful. 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:

$$s_*^2 = \int_{f-\Delta f/2}^{f+\Delta f/2} \log(S_{xx}(f))\,df \tag{13}$$

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

$$s^2 = \int_{f-\Delta f/2}^{f+\Delta f/2} S_{xx}(f)\, df = \int_{f-\Delta f/2}^{f+\Delta f/2} f S_{xx}(f) d[\log(f)] \tag{14}$$

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 some red noise with a sinusoidal signal:

```
lambda=10; % 10 m wavelength
V=0.3; % 0.3 m/s propagation
n2s=0.2; % noise-to-signal ratio
time=(1:10000)';
b=n2s*cumsum(randn(10000,1))+cos(2*pi/lambda*V*time);

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)
```

Frequency-Wavenumber examples.

Now, we'll consider what happens when we want to consider sinusoidal patterns of variability in both time and space. First consider some example frequency-wavenumber spectra. What frequencies and wavenumbers are resolved? What is plotted? What is the Nyquist frequency and Nyquist wavenumber? Is the full frequency-wavenumber space represented?

As a reminder, frequency represents cycles per unit time, and wavenumber represents cycles per unit distance.

We use frequency-wavenumber spectra as a means to track propagating sinusoidal patterns. If it has a characteristic wavelength and frequency, we might suppress that variability if we didn't think about the full structure of the propagating wave. Westward propagating Rossby waves and eastward propagating Kelvin waves have characteristic frequencies and characteristic wavenumbers.

Basics.

Consider a data set $y(x,t)$ where $-x_f < x < x_f$ and $-T < t < T$ (where here we're using $\pm x_f$ for the end points in space, and following our previous examples $\pm T$ for the end points in time.) We know from our definition of the Fourier transform that we can represent y as

$$y(x,t) = \sum_{n=-\infty}^{\infty} a_n e^{i2\pi f_n t}, \tag{15}$$

or

$$y(x,t) = \sum_{m=-\infty}^{\infty} a_m e^{i2\pi k_m x}, \tag{16}$$

and by extension we can do this process in two dimensions:

$$y(x,t) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} a_{nm} e^{i2\pi(f_n t + k_m x)}, \tag{17}$$

where

$$2\pi k_m = \frac{2\pi m}{2x_f} \tag{18}$$

$$2\pi f_n = \frac{2\pi n}{2T} \tag{19}$$

The corresponding spectral density estimate can be calculated from the squared coefficients:

$$\hat{E}(k_m, f_n) = \frac{|a_{nm}|^2}{\Delta k \Delta f} \tag{20}$$

Practicalities.

Suppose we take a time-space data set and Fourier transform it in both directions. Here are some issues that might concern us:

1. *Does it matter whether we Fourier transform time or distance first?* All other things being equal, no. Time and space are orthogonal, and one will not influence the other. To see this consider the following:

$$\int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} y(x,t) e^{i2\pi kx}\, dx \right] e^{i2\pi ft}\, dt = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} y(x,t) e^{i2\pi ft}\, dt \right] e^{i2\pi kx}\, dx = \tag{21}$$

   (How you demean or detrend might matter, and you can ponder these issues.)

2. *Can we compute a frequency-wavenumber spectrum by Fourier tranforming a frequency spectrum in space?* No. If you look at the above equation, you'll see that you need the full complex structure of the Fourier transformed variables in order to determine the spectrum. You have to retain the phase information. If you computed spectra first, you would suppress some of the signal that you wanted.

3. *When we compute frequency spectra, we usually only plot positive frequencies? Does the same strategy apply for frequency-wavenumber spectra?* No, at a given frequency we can have forward and backward propagating signals, so we'll often want to plot a half plane for frequency with positive and negative wavenumbers, or vice versa.

4. *In practical terms, how do we implement this?* If you Fourier transform in time and space, you'll end up with a domain that should be structured as in Figure **??**. But Matlab will of course line up the frequencies and wavenumbers starting with positive and then negative. You can use the Matlab command "fftshift" to swap the presentation around.

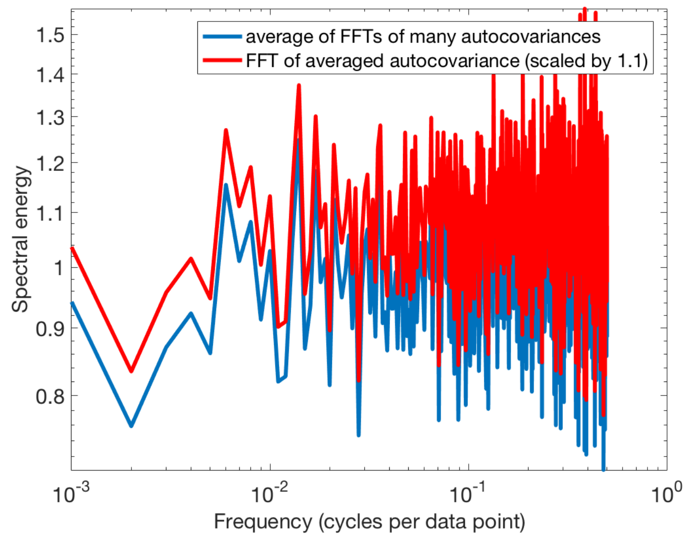

Figure 1: Spectra for white noise, computed by Fourier transforming 100 realizations of the autocovariance function (blue), or by Fourier transforming a smoothed autocovariance function computed from 100 realizations of the data (red). The red line is scaled upward by a factor of 1.1.

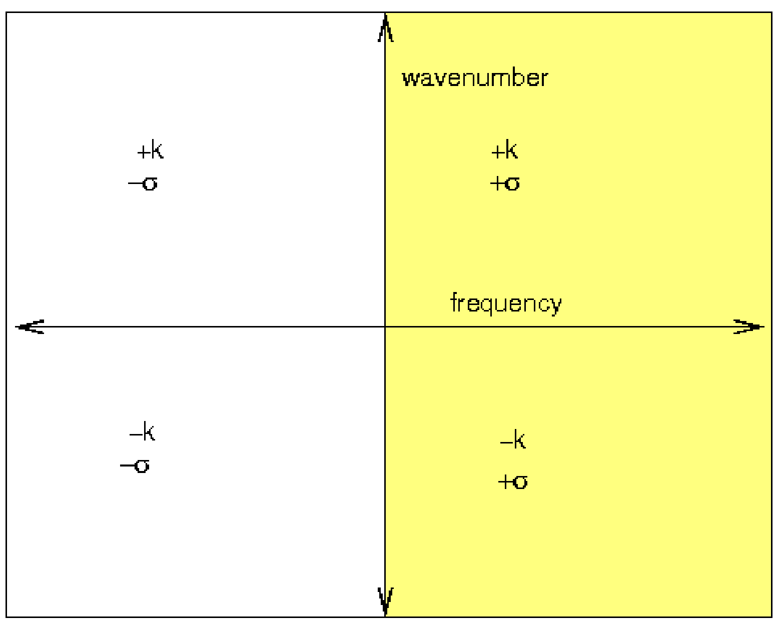

Figure 2: Fourier transform of time/space domain to form frequency/wavenumber domain. Note that $+k, +f$ is the complex conjugate of $-k, -f$ and similarly $+k, -f$ is the complex conjugate of $-k, +f$ so we normally plot only half the domain.