Lecture 10:

Reading: Bendat and Piersol, Ch. 5.2.1

Recap

Last time we looked at the sinc function, windowing, and detrending with an eye to reducing edge effects in our spectra. We've got a full recipe, but in this case, there's more than one way to bake a cake. Is there another way to do this? First a quick diversion to Monte Carlo simulation....

Monte Carlo simulation: How to avoid the traps imposed by standard statistical assumptions (and how to fake your way as a statistician through computational inefficiency rather than clever mathematics)

Most of the time, we estimate spectral error bars using basic statistical assumptions—that data are normally distributed, that we have enough samples for the central limit theorem to apply, that statistics are stationary. These assumptions make our statistical models tractable—we end up with equations we can manipulate, allowing us (or clever statisticians 100 years ago) to derive simple equations that give us rules for how systems should behave. But what happens when those assumptions break down? Or what happens when we have little doubts about the validity of the statistical model. We can always resort to a Monte Carlo process. In Monte Carlo methods, we throw theory on its head and use an empirical approach to generate many realizations of our data set, with noise appropriate to our problem.

As an example, consider the problem of determining the standard error of the mean. When we discussed it in class, we did a quick derivation to show that the standard error of the mean is $\sigma/\sqrt{N}$, where σ is the standard deviation and N is the effective degrees of freedom. But what if I didn't trust this realization? I could generate a large number of realizations of my data with noise typical of the real data, compute means for each realization, and look at the statistics of those values.

So let's put this to work. Suppose I'm computing the mean of $N = 500$ data points. With one sample, I can compute the mean μ and standard deviation σ, and standard error $\sigma/\sqrt{500}$. But I might wonder if μ is really representative. So I can generate an ensemble of fake data, perhaps 100 data sets based on adding Gaussian white noise (or non-Gaussian white noise) to the real data. Each of these data sets will have a mean μ_i and a standard deviation σ_i. And I can look at the standard deviation of all of the μ_i values. I can also look at the pdf of my μ_i's and other higher order statistics. For example:

```
A=randn(500,100);
mu=mean(A);
sigma_A=std(A);
std_A=sigma_A/sqrt(500);
[std(mu) mean(std_A)] % compare standard deviation of means
         %  vs standard error
```

Now we could expand on our example and ask, what if our noise were non-Gaussian or gappy or had other problems, and we could adjust our Monte Carlo process appropriately.

Filtering in the frequency domain

Last time, when we talked about windowing, we noted that windowing in the time domain is equivalent to convolution in the frequency domain (and filtering in the time domain is equivalent to multiplication in the frequency domain.) This could lead you to an interesting conclusion. What if you skipped all the windowing and just did convolutions (i.e. filtering) in the frequency domain? In the limit in which you choose the same filter, these options should be the same.

This approach was originally developed by Daniell and is nicely discussed by von Storch and Zwiers (see their section 12.3.11). Daniell's original idea was to run a moving average over the Fourier transform of the full record. In this case the confidence intervals are determined by:

$$P\left(\chi^2_{\nu,1-\alpha/2} < \nu\frac{\hat{E}(\sigma)}{E(\sigma)} < \chi^2_{\nu,\alpha/2}\right) \tag{1}$$

where ν in this case is 2 $\times$ the number of frequencies averaged together.

The advantages of this approach are that it provides an unbiased estimate of the true spectrum. The width of our averaging forces us to tradeoff bias (minimized if we do less averaging) vs variance (minimized with more averaging). One virtue of averaging in the frequency domain is that we can apply different levels of averaging (with different error bars) depending on the frequency.

Using the auto-covariance to think about spectra.

Now let's look at spectra from a different perspective. When we talked about Parseval's theorem, we took a look at autocovariance. That was the convolution of $x(t)$ with its time reversal, $x(-t)$.

$$y(\tau) = \int_{-\infty}^{\infty} x(t)x(\tau+t)dt. \tag{2}$$

More formally, we might write this autocovariance as $R_{xx}(\tau)$.

$$R_{xx}(\tau) = \int_{-\infty}^{\infty} x(t)x(\tau+t)dt. \tag{3}$$

Now, what if we Fourier transform R?

$$S_{xx}(\sigma) = \int_{-\infty}^{\infty} R_{xx}(\tau)e^{-i\sigma\tau}\,d\tau. \tag{4}$$

Now let's think about starting with two functions, $f(t)$ and $g(t)$. We can write their Fourier transforms:

$$F(\sigma) = \int_{-\infty}^{\infty} f(t)e^{-i\sigma t}\,dt \tag{5}$$

$$G(\sigma) = \int_{-\infty}^{\infty} g(t)e^{-i\sigma t}\,dt. \tag{6}$$

So now let's define F times the complex conjugate of G. (Why do we consider the complex conjugate? Because it's how we always multiply vectors.) So

$$F(\sigma)G^*(\sigma) = \int_{-\infty}^{\infty} k(t)e^{-i\sigma t}\,dt. \tag{7}$$

For the moment, we have no idea what $k(t)$ should be, but we should be able to figure it out. If $F(\sigma)G^*(\sigma)$ is a product in the frequency domain, then $k(t)$ should be a convolution in the time domain:

$$k(t) = \int_{-\infty}^{\infty} f(u)g(u-t)\,du. \tag{8}$$

We can plug $k(t)$ into our equation to check this.

$$\begin{aligned}
\int_{-\infty}^{\infty} k(t)e^{-\sigma t}\,dt &= \int_{-\infty}^{\infty}\left\{\int_{-\infty}^{\infty} f(u)g(u-t)\,du\right\}e^{-i\sigma t}\,dt && (9)\\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u)g(u-t)e^{i\sigma(u-t)}e^{-i\sigma u}\,dt\,du && (10)\\
&= \int_{-\infty}^{\infty} f(u)e^{-i\sigma u}\left\{\int_{-\infty}^{\infty} g(u-t)e^{i\sigma(u-t)}\,dt\right\}du && (11)\\
&= \int_{-\infty}^{\infty} f(u)e^{-i\sigma u}G^*(\sigma)\,du && (12)\\
&= G^*(\sigma)\int_{-\infty}^{\infty} f(u)e^{-i\sigma u}\,du && (13)\\
&= G^*(\sigma)F(\sigma). && (14)
\end{aligned}$$

So we can think about what happens when $g(t) = f(t)$, so that

$$k(t) = \int_{-\infty}^{\infty} f(u)f(u-t)\,du. \tag{15}$$

This means that $k(t)$ is the autocovariance of f. and

$$|F(\sigma)|^2 = \int_{-\infty}^{\infty} k(t)e^{-i\sigma t}\,dt. \tag{16}$$

This says that the Fourier transform coefficients squared (what we use when we compute spectra) ar equivalent to the Fourier transform of the autocovariance.

Now we can rewrite this in terms of the discrete Fourier transform. In this case, the mean of our data is:

$$\langle x\rangle = \frac{1}{2T}\int_{-T}^{T} x(t)e^{i0}\,dt = a_0. \tag{17}$$

and the variance is

$$\langle x * x\rangle = \frac{1}{2T}\int_{-T}^{T} x^*(t)x(t)\,dt - |a_0|^2. \tag{18}$$

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:

$$R(\tau) = \frac{1}{2T}\int_{-T}^{T} x^*(t)x(t+\tau)\,dt - |a_0|^2. \tag{19}$$

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

$$\begin{aligned}
R(\tau) &= \frac{1}{2T}\int_{-T}^{T}\left[\sum_n a_n^* e^{-i\sigma_n t}\sum_m a_m e^{i\sigma_m(t+\tau)}\right]dt && (20)\\
&= \sum_n\sum_m a_n^* a_m e^{-i\sigma_m\tau}\frac{1}{2T}\int_{-T}^{T} e^{i(-\sigma_n+\sigma_m)t}\,dt && (21)\\
&= \sum_m |a_m|^2 e^{-i\sigma_m t} - |a_0|^2 && (22)
\end{aligned}$$

where we used a Kronecker delta (δ_{nm}) to eliminate the integral with $e^{i(-\sigma_n+\sigma_m)t}$ except when $n = m$, and we subtracted a_0^2 at the end to match our original definition. 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

$$R(0) = \frac{1}{2T}\int_{-T}^{T} x^*(t)x(t0)\,dt - |a_0|^2 \tag{23}$$
$$= \sum_m |a_m|^2 - |a_0|^2 \tag{24}$$

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

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,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)
```

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,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)')
```

In the results, shown in Figure 1, the curves 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 a missing factor of 2 and a factor of N or Δt to properly normalize our fft.) Details can be sorted out later, and Thomson and Emery provide a bit of guidance on this.

Notice that we use the "unbiased" estimator. That's up for debate, and there's also an argument to use the "biased" estimator. The difference depends on how we normalize our discrete autocovariance. In the unbiased case, we're computing

$$R(\tau)_{unbiased} = \frac{1}{N-m}\sum_{n=1}^{N-m} x(t_n)x(t_{n+m}). \tag{25}$$

In the biased case, we change how we normalize:

$$R(\tau)_{biased} = \frac{1}{N}\sum_{n=1}^{N} x(t_n)x(t_{n+m}), \tag{26}$$

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.

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. But this might give you an idea of how you can take advantage of the autocovariance to compute spectra from gappy data.

All of this means that we could compute spectra without needing to chunk our data and compute lots of ffts, provided that we had a good estimate of the autocovariance. 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.

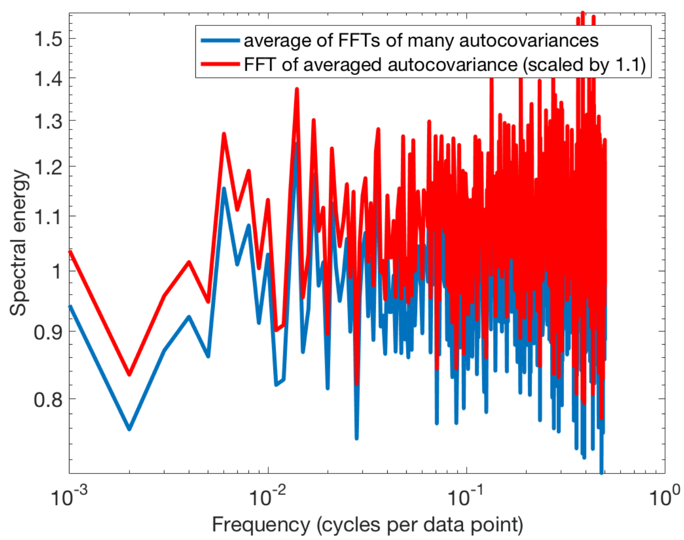

Figure 1: Spectra for white noise, computed by Fourier transforming 100 realizations of the autocovariance function (blue), or by Fourier transforming a smoothed autocovariance function computed from 100 realizations of the data (red). The red line is scaled upward by a factor of 1.1.