Lecture 8:

Reading: Bendat and Piersol, Ch. 4.2.2, 5.2.2, 8.5.4

Recap

Last time we looked at Parseval's theorem as a starting point for computing spectra, and we started computing a test spectrum. So far, our basic recipe for computing spectra is simple:

1. Compute fft(data)
2. Compute squared amplitude for half of fft. Multiply by factor of 2
3. Normalize by N = length of record
4. Plot with a log-log axis

But we only considered some of the issues. Now we need to consider some real world issues. How do we cut down on all the wiggles, so that our spectra look like the ones in papers? What are uncertainties? What happens when there's a background trend? What about all of those frequencies that we don't resolve?

Computing spectra (continued.)

4. We can plot what we have. Typically we use a loglog axis (or at least semilogy). We'll need to compute our frequencies thoughtfully. So for example if we have measurements every 361 seconds, then we might want to convert to cycles per hour by scaling the frequency appropriately.

```
scale = 1/361 * 3600; % to convert from 1 cycle/2 pts to 0.5 cycles
                      % per 361 s to 0.5 cycles * 3600/361 /hr
frequency=(0:N/2)/N * scale; % for N even
frequency=(0:N/2-1)/N * scale; % for N odd
loglog(frequency,amp)
```

5. Now, we have a problem in that our results are way too noisy. We'll have a hard time distinguishing signal from peak. So clearly we're going to need more realizations. To do this, one common practice is chop our data into multiple segments. As a first step, we can just cut the data into M segments of N/M points each. For example, here's a brute strenght approach:

```
N=length(data); M=6; p=N/M;
data=reshape(data,N/M,M); % this gives us an array with N/M points
          % per column and M columns
b=fft(data);  % this computes the fft for each column
amp_b=abs(b(1:p/2+1,:)).^2; amp_b(2:p/2,:)=2*amp_b(2:p/2,:)/p;
loglog(frequency,mean(amp_b,2))
```

Here you can debate whether you should be plotting the sum (to conserve energy) or the average (to represent a mean spectrum for a data set of length p. In most cases you'll want the average.

Now the critical question? How many degrees of freedom does this record have? Is this M to represent M segments? Maybe you can think of it that way, but by convention, we get one degree of freedom for the real part and one for the imaginary part, so 2 per segment. We'll need this to compute error bars, but let's start by noting that our error bars are not the same as the standard error of the mean. We're computing the sum of M squared quantities, and that's going to depend on something that looks like a χ^2 distribution.

When we compute spectra from segments, clearly there are tradeoffs: if I have N data points total, I can have lots of segments with few points in each segment, or few segments with more points per segment. The Nyquist frequency will be the same whatever I choose, since that's determined by the interval between observations. But the low-frequency limit will differ, as will the increment between frequencies (which is determined by the lowest resolved frequency). There's no rule for how to handle this, and your decisions will depend whether you want small uncertainties or high resolution in frequency space.

Detrending

The Fourier transform of a linear trend puts energy into every possible frequency, meaning that if you don't detrend your data, any residual trend will give you a red spectrum. You can spend a lot of energy interpreting red spectra that result from linear trends, but they can really be viewed as an artifact of the data. Consider the linear trend in SST that we examined earlier in class. That linear trend produces a strongly red spectrum. To detrend in Matlab, you can least-squares fit, or simply use Matlab's detrend function:

```
      % import weekly CO2 records from Mauna Loa
a=importdata('weekly_in_situ_co2_mlo.csv',',',44);
      % segment data as 10 segments of length 303;
data=reshape(a.data(1:3030,303,10);
      % detrend each column, and then Fourier transform each column
fdata=fft(detrend(data));
```

We can estimate the spectral slope by eye by looking at the drop in y per order of magnitude in x. If the spectrum drops one order of magnitue in y for a half an order of magnitude increase in x, then it has a slope of f^{-2}. More formally, you could determine this slope through a least-squares fitting procedure.

Windowing: A quick introduction

Often we also want to smooth out the edges. The discrete Fourier transform implicitly assumes that our record repeats again and again, so any discontinuity between the beginning and end of the record can create a step function. To minimize that, we can multiply each segment by a function that minimizes the impact of the edges. (This sounds like it might grossly destroy our spectrum, but we use something that's a bit Gaussian, so its Fourier transform is also fairly Gaussian, and things come out reasonably.) We'll talk more about this, but for now, let's just try a Hanning window:

$$w(t) = \cos^2\left(\frac{\pi t}{2T}\right) = \frac{1+\cos(\pi t/T)}{2} = 0.5 + 0.5\cos(\pi t/T). \tag{1}$$

This is defined to be centered around 0, for $|t| \leq T$. So we can multiply our full record by the Hanning window. (Be sure to detrend first, so that you aren't zeroing out the mean.)

```
% detrend each column, Hanning window,
% and then Fourier transform each column
fdata=fft(detrend(data).*(hanning(303)*ones(1,10));
```

Exercises

What are the Fourier transforms of the following functions: $x(t) = 1, x(t) = \cos(2\pi ft), x(t) = \sin(2\pi ft), x(t) = \exp(-t^2/2\sigma^2), x(t) = t$?

In our examples, note that the Fourier transforms of single frequency sine and cosine give a single peak. (This uses the Kronecker delta, δ_{nm}.) The Gaussian ($x_f(t)$), has a transform of a Gaussian, though it's a bit distorted in this finite-length log-log domain. Formally if $x_f(t) = e^{-t^2/(2\sigma^2)}$, then $X_f(f) = \sqrt{2\pi}\sigma e^{-2\pi^2 f^2 \sigma^2}$ or with coefficients $x_f(t) = \sqrt{\alpha/\pi}e^{-\alpha t^2}$ corresponds to $X_f(f) = e^{-\pi^2 f^2/(a)}$, which says in essence that the Fourier transform of a Gaussian is still a Gaussian. The normalization for this is dependent on our exact notation for the Fourier transform. And the linear pattern should really be thought of as a repeating sawtooth. It's Fourier transform will is a dramatically red spectrum. What does this mean for the Fourier transform of any long-term trend?

Spectral Uncertainties

The problem with all of the spectra that we've computed so far (the amplitude of the Fourier Transform) is that we have no way to evaluate the uncertainty. By eye we can see that it's fairly noisy. We know from computing means that the error of the mean decreases as we average more quantities together.

How do we incorporate more data into our spectra? You might imagine that you could improve your spectrum by extending the input time series from N to $2N$ data points for example. Unfortunately, although adding data points will change your spectrum, it won't reduce your noise or make the spectrum more precise at any individual frequency. Instead it will increase the number of frequencies for which you obtain results from $N/2$ to N.

Error bars for spectra rely on a principle similar to what we used when we estimated the standard error of the mean. Our uncertainties in our spectra decrease as we average more spectra together. The challenge is to figure out how to obtain more spectra that can be averaged together. Typically what we do is to break our time series into segments, compute spectra for each of the segments, and average these to get a mean spectrum. Since we're averaging squared quantities (the spectral amplitudes) the distribution of the sum will follow a χ^2 distribution. And we'll use this to estimate the uncertainties. When we compute our estimate of the spectrum of our data, we'll compute coefficients $a(f)$, and for M segments, at frequency f, the estimated spectrum will be

$$\hat{E}(f) = \langle |a(f)|^2 \rangle = \frac{1}{n}\sum_{i=1}^{n} |a(f)|^2 \tag{2}$$

(Since we actually had two degrees of freedom for each segment, this will give us $\nu = 2n$ degrees of freedom.)

Now to estimate errors the whole argument stems from a notion that there is a true spectrum $E(f)$, which we don't know, and our best estimate $\hat{E}(f)$, that we've computed, but which might be noisy. We could consider the difference between E and $\hat{E}$, but this gets hairy quickly. Besides we're going to look at our results in log/log space anyway. So let's look at the ratio of the two. Recall that $\ln(\hat{E}/E) = \ln(\hat{E}) - \ln(E)$. We can define a variable that represents this sum of squares

normalized by the true spectrum

$$y = \nu \frac{\hat{E}_\nu(f)}{E(f)} \tag{3}$$

where $\hat{E}_\nu$ is our best estimate of the spectrum from real observations, when we have ν degrees of freedom. This variable should have a χ^2 distribution with ν degrees of freedom. Thus we'll always consider the ratio $\hat{E}(f)/E(f)$, where we use $\nu/2$ data segments.

So even though we don't know E, we know that the ratio of $\nu\hat{E}$ to E ($\nu\hat{E}/E$) is a χ^2 quantity. The expectation value of this ratio is ν, and the standard deviation is $\langle y^2 \rangle - \langle y \rangle^2 = 2\nu$ (see Bendat and Piersol, section 4.2.2). From that we can infer that with two degrees of freedom, the standard deviation $\langle \hat{E}^2 \rangle - \langle \hat{E} \rangle^2 = 2/\nu E_0^2 = E_0^2$, which means that the standard deviation is equal to the original value. Clearly we need more samples.

Now in reality, we don't care about the details of χ^2 but rather the probability that our estimate of the spectrum $\hat{E}$ is close to or far from the unknown true spectrum E, so we're going to look at the probability that $\nu\hat{E}(f)/E(f)$ falls within a fixed range. So we want to use the χ^2 distribution as a probability distribution and ask about the probability that $\hat{E}(f)/E(f)$ should be within $\pm$ some range of 1. Formally, the probability that the estimated spectrum should be close in value to the true spectrum is:

$$P\left(\chi^2_{\nu,1-\alpha/2} < \nu\frac{\hat{E}(f)}{E(f)} < \chi^2_{\nu,\alpha/2}\right) = 1-\alpha \tag{4}$$

so if we want to find a 95% significance level, we set α to 0.05. We'll want to know the value of χ^2 that corresponds to a given point on the cdf, so for this we use the inverse χ^2 function ("chi2inv" in Matlab). So we can invert this relationship to provide the probability that the true value is within a particular range of the observed value:

$$P\left(\frac{\chi^2_{\nu,1-\alpha/2}}{\nu} < \frac{\hat{E}(f)}{E(f)} < \frac{\chi^2_{\nu,\alpha/2}}{\nu}\right) = 1-\alpha \tag{5}$$

$$P\left(\frac{\nu}{\chi^2_{\nu,1-\alpha/2}} > \frac{E(f)}{\hat{E}(f)} > \frac{\nu}{\chi^2_{\nu,\alpha/2}}\right) = 1-\alpha \tag{6}$$

$$P\left(\frac{\nu\hat{E}(f)}{\chi^2_{\nu,1-\alpha/2}} > E(f) > \frac{\nu\hat{E}(f)}{\chi^2_{\nu,\alpha/2}}\right) = 1-\alpha \tag{7}$$

$$P\left(\frac{\nu}{\chi^2_{\nu,\alpha/2}} < \frac{E(f)}{\hat{E}(f)} < \frac{\nu}{\chi^2_{\nu,1-\alpha/2}}\right) = 1-\alpha \tag{8}$$

Whether you use χ^2 or its reciprocal, the ratio between the high and low error bars should be the same, and the ratio will be what matters.

This error formulation differs from the usual error bars that we're used to seeing where we say for example that the true temperature should be the measured temperature plus or minus an uncertainty: $T = \hat{T} \pm \delta_T$. We can develop a similar expression for the true spectrum: $E(f)$ is in the range between $\nu\hat{E}(f)/\chi^2_{\nu,\alpha/2}$ and $\nu\hat{E}(f)/\chi^2_{\nu,1-\alpha/2}$, where ν is twice the number of segments. This expression isn't very easy to interpret, since it varies as a function of frequency, and the estimated value $\hat{E}(f)$ is not at the mid-point of the range.

Instead we'll keep in mind that we've computed the uncertainty for the ratio $\hat{E}(f)/E(f)$, and the probabilities for this ratio does not depend on frequency. On a log plot, error bars defined by

the range between $\nu/\chi^2_{\nu,\alpha/2}$ and $\nu/\chi^2_{\nu,1-\alpha/2}$ are the same size at all frequencies, so we can easily compare spectral peaks at different frequencies.

Some statistics books include look-up tables for χ^2, but we can compute it directly in Matlab. For $\nu = N/2$ data segments, the error limits are:

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

We can plot these values as:

```
semilogy([f f],[err_low err_high]*A);
```

where we set the frequency f and the amplitude A, so that the error bar ends up positioned in a convenient spot on the plot.

Now to have $\nu = N/2$ data segments, we have to split our long data record into shorter segments. We can do this by taking M data points at a time:

```
N=length(data);
M=segment_length; % define this value
for n=1:floor(N/M)
 d=data((n-1)*M+1:n*M); %select data for the nth segment
 fd(:,n)=fft(d);        % compute fft
end
sd=sum(abs(fd(1:M/2+1,:)).^2,2)/N; % sum over all spectra
                                    %(sum over 2nd index)
sd(2:end)=sd(2:end)*2;

nu=2*floor(N/M);
err_high = nu/chi2inv(.05/2,nu);
err_low =  nu/chi2inv(1-.05/2,nu);

semilogy(0:M/2,sd,[M/4 M/4],[err_low err_high]*sd(M/4))
```