Lecture 6:

Reading: Bendat and Piersol, Ch. 2.1-2.2

Recap

Last time we looked at the Fourier transform. We considered cosine and sine transforms, derived coefficients (a_q and b_q) for cosine and sine, and then showed that we could recombine these to make complex coefficients for $e^{i2\pi q f_1 t}$ and $e^{-i2\pi q f_1 t}$. We found these coefficients to be complex conjugates of each other. Since cosine/sine transformations and Fourier transforms using $e^{\pm i2\pi q f_1 t}$ are closely related, we can express results of one in terms of the other. In other words, instead of computing $\sum_{j=1}^{N} x_j \cos(2\pi f_j t)$ and $\sum_{j=1}^{N} x_j \sin(2\pi f_j t)$, we can instead find $\sum_{j=1}^{N} x_j \exp(i2\pi f_j t)$ and then use the real and imaginary parts to represent the cosine and sine components.

Fourier transform in discrete form

We finished up last time by looking at the notation for a Fourier transform in continuous form. Bendat and Piersol use the following:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi f t}\, dt \tag{1}$$

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{i2\pi f t}\, df \tag{2}$$

Data come in discrete form. If they are uniformly separated (in time or space), then they are easy to Fourier transform. The same questions about choices of notation apply in the discrete form that we consider when we analyze data. And we can get ourselves really confused. So we have to keep in mind one rule: we don't get to create energy. That means that we need to have the same total variance in our data set in the time domain as we have in the frequency domain. This is Parseval's theorem, and we'll return to it.

One of the glories of the Fourier transform is that we can take all of these projections and make them extremely efficient through the Fast Fourier Transform (FFT). In principle, FFT's are most efficient if you compute them for records that are a power of 2 in length, so 64 or 128 or 256 points for example. But modern FFTs are fast even if your data set doesn't have 2^n elements. Moreover, a year doesn't have 2^n days, so trying to force a data record to conform to a length of 2^n can suppress some of the natural periodicity.

Mathematically the Matlab definitions look like this:

$$X_k = \sum_{n=1}^{N} x_n \exp(-i2\pi(k-1)(n-1)/N), \tag{3}$$

where frequency labels k and data labels n go from 1 to N. Here capital letters are used to denote Fourier transformed variables. Matlab computes this using the command "fft".

The inverse of the Fourier transform is computed using "ifft" and is defined to be:

$$x_n = \frac{1}{N} \sum_{k=1}^{N} X_k \exp(i2\pi(k-1)(n-1)/N) \tag{4}$$

In Matlab the Fourier transform and inverse Fourier transform become:

```
f=fft(x)
x_new=ifft(f)
```

To make Parseval's theorem work, the variance of our data has to equal the variance of the Fourier transform. Thus we'll want to compare:

```
sum(x.^2)
sum(abs(f).^2)
f'*f
sum(f.*conj(f))
```

They don't quite agree, so we'll see that we should divide the Fourier transform by N, the number of data points.

What do we gain by Fourier transforming our data?

We live life in the time domain, so it's sometimes hard to think about the world as seen in the frequency domain. While linear trends aren't well represented by the Fourier transform, the Fourier tranform is particularly effective for representing sinusoidal oscillations. Solar radiation that warms the Earth varies on a 365.25 day cycle with the seasons, and on a 24 hour cycle, with the rising and setting of the sun. Ocean tides vary at semidiurnal (12.4 hour) and diurnal frequencies (as well as being modulated on fortnightly and monthly intervals.) Thus if you look at data from a tide gauge, you see oscillatory fluctuations at a variety of different frequencies, as shown in the slides. If we solve for the tidal amplitudes, we find for example:

Symbol	Frequency (cpd)	Amplitude (cm)	Greenwich Epoch
O1	0.92953571	8.91	217
P1	0.99726209	5.32	224
K1	1.00273791	16.12	225
M2	1.93227361	9.97	354
S2	2.00000000	6.45	357

The complex Fourier coefficients that emerge from the fft might seem confusing, but they give us a lot of information about our data, allowing us, for example to tell whether there is more energy at frequency σ_j compared with frequency σ_l. The Fourier coefficients are complex so this comparison might seem confusing, but we'll just examine the squared magnitudes of the coefficients: $|a_j|^2$.

Of course, if we knew the frequency exactly, we could just do a least-squares fit, but often we aren't exactly sure of the frequencies in question—there might be energy spread over a broad range of frequencies, and the Fourier tranform provides us with a way to examine our data in terms of oscillatory signals.

Three great traits of the Fourier transform

We've talked about the effectiveness of the Fourier transform for identifying frequencies that are particularly energetic without having to know a priori what frequencies might have resonant peaks, and we've noted that the Fourier transform is useful for evaluating the size of one peak relative to another.

1. Derivatives in time become multiplication in the frequency domain. Fourier coefficients have some additional mathematical power. For example, suppose I want to take the time derivative of my data. If I start with

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

then

$$\frac{\partial A(t)}{\partial t} = \sum_{n=-\infty}^{\infty} a_n \frac{\partial e^{-i2\pi f_n t}}{\partial t} = \sum_{n=-\infty}^{\infty} -i2\pi f_n a_n e^{-i2\pi f_n t} \tag{6}$$

So the first derivative become a multiplication by frequency. Higher derivatives are similarly simple

$$\frac{\partial^q A(t)}{\partial t^q} = \sum_{n=-\infty}^{\infty} (-i2\pi f_n)^q a_n e^{-i2\pi f_n t}. \tag{7}$$

Integration can be represented as a division operation:

$$\int A(t)\, dt = \sum_{n=-\infty}^{\infty} (i2\pi f_n)^{-1} a_n e^{i2\pi f_n t} \tag{8}$$

though we'll run into a bit of trouble if $f_0 \neq 0$, that is if the record has a non-zero mean. That can mean that we might want to remove the mean before we start doing anything more complicated.

In class we illustrated this by looking at the time series of the Southern Annular Mode from http://www.nerc-bas.ac.uk/icd/gjma/sam.html. I had done a bit of pre-editing of the ASCII data file to remove the header and make it a full matrix. Then we did the following

```
% read the data
data=load('sam_nohead.txt');
data=data(:,2:13);  % remove the first column with the years
data=data'; % rotate the data so that months run down.  By doing
          % this, we can obtain a time series by using data(:);
%
% compute FFTs
fft_data = fft(data(1:717)); % eliminate the last 3 points which
                             % were NaN
fft_ddata = fft(diff(data(1:717))); % compute the fft of the first
                                    % derivative
%
% plot comparisons
semilogy(0:716,abs(fft_data).^2);  % plot the squared amplitude of
            % the fft note the symmetry, since we haven't truncated
            % at the Nyquist frequency
hold on
semilogy(0:715,abs(fft_ddata).^2,'r'); % plot the squared amplitude
                                       % of the first derivative
semilogy(0:357,abs(fft_data(1:358)).^2 .*((0:357)).^2 /715/25,'m')
    % plot the amplitude scaled by frequency, with an arbitrary
    % multiplicative factor to help us get the amplitudes to match
legend('squared fft of data','squared fft of data derivative',...
      'frequency squared times squared fft of data')
```

The results are shown in Figure 1. Here we've done this is the sloppiest way possible, but it still gives us a demonstration that the fft of the first derivative has the same spectral structure as the fft multiplied by frequency

2. Fourier transforms simplify convolution.

Suppose you plot some noisy data—the data features crazy amplitude swings, and no one can make any sense of it, but you think that hiding behind all this noise, there might be a slowly varying signal. You might be told, just do a running mean to smooth it out. That running mean is a convolution.

Convolution plays an important role in thinking about the Fourier transform, so we need to spend a little time on the concept. Here's the basic convolution integral:

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau. \tag{9}$$

You can think of x as the data, and h as a filtering operator (such as a "boxcar" filter, or a triangle filter, or a roughly Gaussian-shaped window, or anything else that suits you.

In Matlab you can do this as:

```
y=conv(data(:),boxcar(12)/12);
```

which produces the same results as:

```
y=filter(boxcar(12)/12,1,data(:));
```

In both cases these will be shifted by half the width of the filter, so we can plot:

```
plot(data(:))
hold on
t(-6:731-7,conv(data(:),boxcar(12)/12),'r','LineWidth',2)
xlabel('time (months)','FontSize',14)
ylabel('SAM','FontSize',14)
legend('monthly SAM','one-year running mean of SAM')
```

See Figure 2

Formally the notation for a convolution of two records h and x is written

$$h * x = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau. \tag{10}$$

What happens if we Fourier transform this?

$$\begin{aligned}
\mathcal{F}(h*x) &= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau\right]e^{-it2\pi f}\,dt &(11)\\
&= \int_{-\infty}^{\infty} h(\tau)\int_{-\infty}^{\infty}\left[x(t-\tau)e^{-it2\pi f}\,dt\right]d\tau &(12)\\
&= \int_{-\infty}^{\infty} h(\tau)e^{-i\tau 2\pi f}\mathcal{F}(x(f))\,d\tau &(13)\\
&= \mathcal{F}(h)\mathcal{F}(x) &(14)
\end{aligned}$$

where here I've represented the Fourier transform with a script $\mathcal{F}$.

This has profound consequences. It means that anything that required a convolution in the time domain I can handle trivially in the Fourier domain. Suppose I want to filter my data. If I don't like the hassle of convolving, I can just Fourier transform, multiply by the Fourier transform of my filter, and inverse Fourier transform. This will prove to be amazingly powerful.

3. Parseval's theorem: Total variance in the time domain equals total variance in the frequency domain

The third trait of the Fourier transform is that it conserves energy (or variance). Formally, we refer to this as Parseval's theorem, and we'll take a closer look later. Parseval's theorem is the form

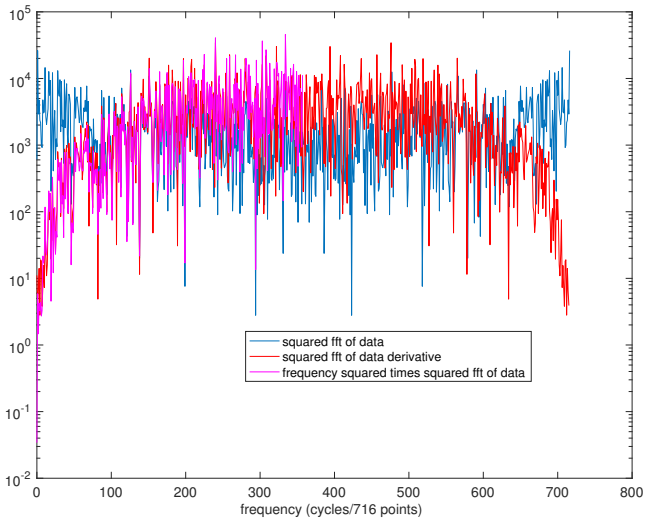

Figure 1: Squared Fourier amplitudes computed from the time series of the Southern Annular mode, as discussed in the text.

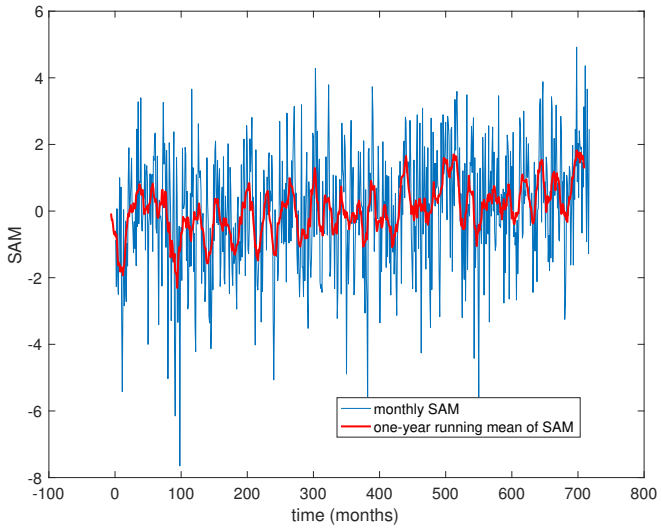

Figure 2: Time series of the Southern Annular Mode (SAM) and a one-year running mean.