Lecture 4:

Reading: Bendat and Piersol, Ch. 4 (esp. section 4.6)

Recap

Key concepts from last time focused on the standard error of the mean and its links to error propagation, the chi-squared and Rayleigh distributions, and methods for telling whether one pdf is statistically like another (the Komogorov-Smirnov test and the chi-squared test).

You'll recall that the Rayleigh distribution represents the square root of two independent Gaussian components, $y = \sqrt{x_1^2 + x_2^2}$.

$$p(y) = \frac{y}{\sigma^2} \exp\left[-\frac{y^2}{2\sigma^2}\right]. \tag{1}$$

And the χ^2 distribution represents the sum of n squared variables:

$$\chi_n^2 = z_1^2 + z_2^2 + z_3^2 + ... + z_n^2. \tag{2}$$

The slides showed examples of these, and we noted that the χ^2 distribution has a mathematical formulation in terms of the Γ function. I glossed over the definition of the Γ function, because it's not very mathematically tractable, and is normally handled via a look-up table.

We also looked at two possible tests for telling if two pdfs differ. I didn't show you how to apply these, so here's a quick guide. One test was the Kolmogorov-Smirnov test. (If you have one data set, and want to know whether it is plausibly Gaussian, put all the data in a vector $\mathbf{x}$, and in Matlab use:

```
kstest(x)
```

If the output is zero, you stick to the null hypothesis that the data are Gaussian; if the output is 1, you reject that hypothesis at the 5% significance level and thus surmise that perhaps the data weren't Gaussian. So for example, if we want to test if a vector of Gaussian noise has a Gaussian distribution, we could type:

```
kstest(randn(10000,1))
```

which produces 0. For a non-Gaussian (uniform distribution) of random numbers

```
kstest(rand(10000,1))
```

produces 1, implying that the input vector is not Gaussian.

If we're comparing two data sets, we can make two vectors (e.g. x and y), and use

```
kstest2(x,y)
```

. Again, an output of 0 says plausible the two records come from the same "true" distribution; output of 1 says that at the 5% significance level, you can reject the null hypothesis that x and y come from the same general source. Compare these two cases:

```
kstest(randn(10000,1),randn(10000,1))
kstest(randn(10000,1),rand(10000,1))
```

The second was a χ^2 test, in which you bin your data into histograms and ask if the number of data you find in each bin is consistent.

A second strategy is to bin the data and ask whether the number of data in the bin is consistent with what we'd expect, using a χ^2 statistics. In this case for comparisons with a theoretical pdf,

$$\chi^2 = \sum_i \frac{(N_i - n_i)^2}{n_i}, \tag{3}$$

where N_i is the observed number of events in bin i, and n_i is the theoretical or expected number of events in bin i. For comparisons between two distributions,

$$\chi^2 = \sum_i \frac{(N_i - M_i)^2}{N_i + M_i}, \tag{4}$$

where N_i and M_i are each observed numbers of events for bin i. The values of χ^2 are evaluated using the χ^2 probability function $Q(\chi^2|\nu)$, which is an incomplete gamma function, where ν is the number of bins (or the number of bins minus one, depending on normalization). In Matlab this is

```
gammainc(chi_squared/2,nu/2)
```

or equivalently

```
chi2cdf(chi_squared,nu)
```

To test this out, you can produce a few sets of binned random data. (Be sure to use the same bins for each record.)

```
[a1,a2]=hist(randn(100000,1),-6:.5:6);
[b1,b2]=hist(randn(100000,1),-6:.5:6);
[c1,c2]=hist(rand(1000000,1)-.5,-6:.5:6);
```

The tests produce the following:

```
chi2=sum((b1-a1).^2 ./(b1+a1))
chi2cdf(chi2,25)
gammainc(chi2/2,25/2)

chi2=sum((c1-a1).^2 ./(c1+a1))
chi2cdf(chi2,25)
gammainc(chi2/2,25/2)
```

If the data sets are similar at least at the 95% level, then we expect gammainc to return a value less than or equal to 0.95.

Fitting a function to data: least-squares fitting

Now, let's return to our time series. You might remember we were looking for a linear trend for:

$$\mathbf{T} = T_o + b\mathbf{t} + \mathbf{n}, \tag{5}$$

where $\mathbf{T}$ represents our measured temperature data (as a vector), T_o is a constant (unknown), $\mathbf{t}$ is time, and b is the time rate of change (also unknown), and since this is the real world, $\mathbf{n}$ is noise (representing the part of the signal that isn't a linear trend. Formally, provided that we have more than two measurements, aside from the unknown noise vector, this is an over-determined system. Since the noise is unknown, and there are lots of independent values, the system is formally underdetermined. But we won't lose hope. We just move forward under the assumption that the noise is small.

Last time we started writing this as a matrix equation:

$$\mathbf{Ax} + \mathbf{n} = \mathbf{y}, \tag{6}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ \vdots & \vdots \\ 1 & t_N \end{bmatrix}, \tag{7}$$

making $\mathbf{A}$ an $N \times 2$ matrix. And $\mathbf{y}$ is an N-element column vector containing, for example, our temperature data:

$$\mathbf{y} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_N \end{bmatrix}. \tag{8}$$

Then $\mathbf{x}$ is the vector of unknown coefficients, in this case with 2 elements (e.g. $x_1 = T_o$ and $x_2 = b$).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{9}$$

How can we find the best solution to this equation to minimize the misfit between the data $\mathbf{y}$ and the model $\mathbf{Ax}$? The misfit could be positive or negative, and absolute values aren't mathematically tractable, so let's start by squaring the misfit.

$$\epsilon = (\mathbf{Ax} - \mathbf{y})^T(\mathbf{Ax} - \mathbf{y}) = \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}. \tag{10}$$

Then we can minimize the squared misfit. The natural route to minimization comes by taking the derivative, and then setting the results equal to zero. Our unknown is $\mathbf{x}$, so we'll minimize in terms of that:

$$\frac{\partial \epsilon}{\partial \mathbf{x}} = 2\mathbf{A}^T\mathbf{Ax} - 2\mathbf{A}^T\mathbf{y} = 0, \tag{11}$$

and this implies that

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{y}, \tag{12}$$

so

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}. \tag{13}$$

In Matlab, you'd code this as follows:

```
% assuming time matches the SST standard with days counting from
% January 1, 1800
% and data is the monthly sea surface temperature
plot(time+datenum(1800,1,1),data,'LineWidth',2); hold on
A=[ones(length(time),1) time(:)];
x=inv(A'*A)*A'*data;
plot(time+datenum(1800,1,1),A*x,'r','LineWidth',2)
```

As we've noted, if we want to find a trend, we define $\mathbf{A}$ to have a column of ones (to identify the mean) and a column containing the time, to identify the rate of change. We can make our model $\mathbf{A}$ progressively more complicated by adding additional columns. What do we do if we want to find the annual cycle? Before I give you any answers, take a moment to think about this.

We could use:

$$\mathbf{A} = \begin{bmatrix} 1 & \cos(t_r) & \sin(t_r) \\ \vdots & \vdots & \vdots \end{bmatrix}, \tag{14}$$

where time t is measured in days, and $t_r = 2\pi t/365.25$, is the time in radians. We need the sine and cosine because we don't know the phase of our annual cycle exactly. You might imagine that we could fit for the phase (e.g. $\sin(t_r + \phi)$), and we could, but that would be a non-linear fitting process, and the power of least-squares fitting won't work if we try that—we'd quickly be plunged into the murky world of non-linear fitting procedures, which is messy, unreliable, and not necessary in this case.

Here's an example

```
% continuing from previous example....
A2=[ones(length(time),1) time(:) sin(2*pi*time/365.25) ...
    cos(2*pi*time/365.25)];
x2=inv(A2'*A2)*A2'*data;
plot(time+datenum(1800,1,1),A2*x2,'m','LineWidth',2)
```

Orthogonality and Least-Squares Fits

Let's think about one important detail of our fitting procedure. What would happen if we wanted to fit $T = x_1 + x_2 t_r + x_3 t_r$, that is to fit two constants to the same variable? In this case, clearly x_2 and x_3 are completely indistinguishable. What happens when we try to use our redundant functions in our matrix $\mathbf{A}$?

$$\mathbf{A} = \begin{bmatrix} 1 & t_1 & t_1 \\ 1 & t_2 & t_2 \\ \vdots & \vdots & \vdots \\ 1 & t_N & t_N \end{bmatrix}. \tag{15}$$

Then we find:

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} N & \sum t_i & \sum t_i \\ \sum t_i & \sum t_i^2 & \sum t_i^2 \\ \sum t_i & \sum t_i^2 & \sum t_i^2 \end{bmatrix}. \tag{16}$$

The second and third rows of $\mathbf{A}^T\mathbf{A}$ are identical, which tells us that the third row is adding no additional information to the system. As a result, the matrix $\mathbf{A}^T\mathbf{A}$ is singular, and we won't be able to find an inverse for it. (In Matlab, when you try to do the inversion, you'll see a message, "Warning: Matrix is singular to working precision.")

You probably wouldn't try to fit coefficients to two identical functions, but you might do something that was fairly similar. For example, $T = x_2 t_r + x_3 \sin(t_r)$ poses a similar problem when t_r is near zero. In this case, the rows of $\mathbf{A}^T\mathbf{A}$ might not be identical, but they might be nearly the same so that Matlab would give you an error message.

Similarly, you'll have trouble if you try: $T = x_1 + x_2 t_r + x_3(1 + t_r)$.

None of this was an issue when we used sine and cosine, because they are orthogonal, so they contain no redundant information.

Building complexity: multiple oscillatory signals

Now let's think about the pressure record from the Scripps pier, because we know that had a lot of sinusoidal variability. What might we include in that fit? We might hypothesize that data collected on the pier could be influenced by an annual cycle (1 cycle/365.25 days), a diurnal cycle (1 cycle/24 hours), and a tidal cycle (1 cycle/0.5175 days = 1 cycle/(12 hours + 25.2 minutes)). How would we fit for all of these components?

What is the maximum number of sinusoidal cycles that we can resolve in N points? One possibility would be that the maximum is N cycles per N points. That would require a full sinusoidal oscillation squeezed between data element 1 and data element 2. But if you think about it, we wouldn't expect to have enough information to determine the amplitude of a sine wave that had to squeeze itself between consecutive observations. Moreover if N cycles per N points were the maximum, this would mean that we'd be solving for $2N$ coefficients with only N data points. Clearly that would require more information than we have available.

So the maximum must be less than N. There are two ways to think about this. One way is to think about the information content of our data. At the most basic level, if I have N elements in my data matrix, then at best, I can hope to have N independent equations, so I can fit N coefficients as an exactly determined problem. So I can solve for frequencies from 0 cycles per N points to $N/2$ cycles per N points, with 2 coefficients for all but the end members of my record. The other way to think of this is to observe that you'll need at least 2 data points per cycle to determine even a minimum amount of information about the sine or cosine amplitudes of your data. So the highest frequency that you can possibly detect of $N/2$ cycles per N data points. This is the *Nyquist frequency* and it is one of the most central concepts in time series analysis.