Fitting Data to Sines and Cosines

The set of $N$ sines and cosines span all space, which means that if $A$ has $N$ columns and $N$ rows, we can find an exact fit, $b = b$. This is the basis for a Fourier transform—we can represent any $N$-element data vector $b$, as a set of $N$ amplitudes of sines and cosines.

Let’s carry out a least squares fit, $Ax = b$, where $A$ encompasses all possible sines and cosines resolved in our data $b$:

$$A = \begin{bmatrix} 1 & \cos(\omega t) & \sin(\omega t) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \quad (1)$$

Since the columns of $A$ are all orthogonal, $A^T A$ will have zeros except on its diagonal:

$$A^T A = \begin{bmatrix} N & 0 & 0 & \cdots \\ 0 & \sum_{i=1}^{N} \cos^2(\omega t_i) & 0 & \cdots \\ 0 & 0 & \sum_{i=1}^{N} \sin^2(\omega t_i) & \cdots \\ 0 & 0 & 0 & \ddots \end{bmatrix}. \quad (2)$$

This means that we can compute $(A^T A)^{-1}$ easily by taking the reciprocal of each diagonal element of $A^T A$.

$$\begin{array}{l}
(A^T A)^{-1} = \\
\quad = \begin{bmatrix} 1/N & 0 & 0 & \cdots \\ 0 & 1/\sum_{i=1}^{N} \cos^2(\omega t_i) & 0 & \cdots \\ 0 & 0 & 1/\sum_{i=1}^{N} \sin^2(\omega t_i) & \cdots \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \quad (3)
\end{array}$$

where we’re using the fact that $\cos^2 x = (1 + \cos(2x))/2$ and $\sin^2 x = (1 - \cos(2x))/2$. This means that each element of $x$ is independent, and we can find them fairly efficiently from $x = (A^T A)^{-1} A^T b$.

$$x_i = 2 \sum_{j=1}^{N} a_{ij} b_j \frac{\cos(\omega_i t)}{N} = 2 \cos(\omega_i t) b \quad \text{or} \quad 2 \sin(\omega_i t) b, \quad (5)$$

where $\omega_1 = \omega_2 = 2\pi/N$, $\omega_3 = \omega_4 = 4\pi/N$, and so forth. Thus the solution vector $x$ consists of covariances of $b$ with sine or cosine at each possible frequency. We often think of $x$ as the projection of our data $b$ onto each of our orthogonal functions, and we can use this concept of a projection or covariance to make the calculation efficient.

We won’t normally find a Fourier transform by least-squares fitting since that involves keeping track of elements in a large matrix, so it’s inefficient. However, before we talk about how to find the Fourier transform, let’s look at some examples of the world as seen in Fourier transformed space.

Viewing the World Spectrally

We live life in the time domain, so it’s sometimes hard to think about the world as seen in the frequency domain. Still there are some things you probably do know about. 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 Figure 1. If we solve for the tidal amplitudes, we find for example:

We’re not limited to looking at amplitudes at only a few frequencies. Figure 2 shows the total energy (combined cosine and sine components) as a function of frequency for all resolved frequencies. This is called a spectrum. We see big energy peaks at one cycle per day (cpd) and two cycles per day, corresponding
Figure 1: (left) Predictions of tidal sea level fluctuations at Haleiwa, Oahu, Hawaii from numerical simulation by D. Luther. (right) Sea level fluctuations from Scripps Pier, January 2000.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency (cpd)</th>
<th>Amplitude (cm)</th>
<th>Greenwich Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>0.92953571</td>
<td>8.91</td>
<td>217</td>
</tr>
<tr>
<td>P1</td>
<td>0.99726209</td>
<td>5.32</td>
<td>224</td>
</tr>
<tr>
<td>K1</td>
<td>1.00273791</td>
<td>16.12</td>
<td>225</td>
</tr>
<tr>
<td>M2</td>
<td>1.93227361</td>
<td>9.97</td>
<td>354</td>
</tr>
<tr>
<td>S2</td>
<td>2.00000000</td>
<td>6.45</td>
<td>357</td>
</tr>
</tbody>
</table>

Table 1: Leading tidal constituents in Haleiwa, Hawaii. See http://www.soest.hawaii.edu/oceanography/dluther/HOME/Tables/Hal.htm for more tidal constituents.

to the dominant tidal forcing frequency. This is really useful, because we can now represent continuous fluctuations in the time domain as a single preferred frequency or two in the frequency domain. The spectrum gives us a compact way to characterize sea level variability.

We also see plenty of energy at nontidal frequencies. One distinct feature of the spectrum is that it is “red”, meaning that there is more energy at low frequencies than at high frequencies. Notice that it has been plotted on a semilog axis. The terminology “red” comes from optics—low frequency (long wavelength) light is red; high frequency (short wavelength) light is blue. In the same way, white noise, is white because the energy is the same at all frequencies, with no preferred periodicity.

Our next step is to figure out how to represent our measurements in frequency space and how to compute energy spectra.

**Representing Sines and Cosines in Exponential Notation**

Before we do anything further, we need a quick review of complex numbers and exponential notation. Here are the important rules:

\[
i = \sqrt{-1} \quad (6)
\]

\[
\exp(it) = \cos(t) + i\sin(t) \quad (7)
\]

\[
\cos(t) = \frac{\exp(it) + \exp(-it)}{2} \quad (8)
\]

\[
\sin(t) = \frac{\exp(it) - \exp(-it)}{2i} \quad (9)
\]

This means that instead of computing \(\sum_{j=1}^{N} b_j \cos(\omega_j t)\) and \(\sum_{j=1}^{N} b_j \sin(\omega_j t)\), we can instead find \(\sum_{j=1}^{N} b_j \exp(i\omega_j t)\) and then use the real and imaginary parts to represent the cosine and sine components.
This gives us a quick shorthand for representing our results as sines and cosines.

Fourier transforms are normalized in a variety of different ways. We’ll use the Matlab definitions. So

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

(10)

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:

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

(11)