CHAPTER 12

Methods of Data Analysis: Fourier Analysis of Time Series

Numerical Methods in the Hydrological Sciences
Special Publication Series 57
Copyright 2005 by the American Geophysical Union
doi:10.1029/057SP17

12. Methods of Data Analysis: Fourier Analysis of Time Series

12.1. Periodic functions

In previous chapters we have discussed numerical methods for solving algebraic and differential equations. Another area in which numerical methods play an important role in hydrological investigations is in the analysis of data. We touched on one aspect of this in Chapter 3.3 – fitting a polynomial to a set of points for the purpose of interpolation. Another aspect is one familiar to experimental hydrologists – fitting a function (line, curve) to a set of data points in order to describe the observed relationship between dependent and independent variables in a simple mathematical form. The most common example would be using a linear regression (Chapter 1.4).

Many relationships cannot be described adequately in terms of a linear or low-order polynomial function. Among these are periodic variations typical of many hydrological phenomena, such as monthly average river discharge, hourly evapotranspiration, and tidal amplitudes. An example of a time series with obvious periodicity is the widely reported record of CO_2 at Mauna Loa (Figure 12.1). In addition, there are many other time-varying processes, such as turbulence, floods, and precipitation that, while not strictly periodic, can be analyzed using the techniques discussed in this chapter.

The most common way to analyze periodic or quasi-periodic data is through Fourier analysis. In Fourier analysis, a function is represented by the sum of an infinite series of sinusoidal curves of varying amplitude and frequency. While it seems reasonable to expect that such a series would provide a good representation of truly periodic functions, we'll see that Fourier analysis can provide important information about almost any time series.

12.2. Fourier series

The underpinning of any Fourier analysis is the Fourier series

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left[a_n \cos\left(\frac{2p\,nx}{L}\right) + b_n \sin\left(\frac{2p\,nx}{L}\right)\right] \tag{12.1}$$

in which a periodic function $f(x)$, with period L, is represented as an infinite sum of sin and cos terms. The fundamental task of Fourier analysis is to determine the coefficients a_n and b_n. The sin and cos terms in the series are orthogonal, essentially representing parts of a function that are odd (antisymmetic) and even (symmetric), respectively, with respect to the origin. The orthogonality of the sin and cos terms leads to some identities that simplify the process of evaluating the coefficients a_n and b_n in (12.1).

$$\int_{-L/2}^{L/2} \sin\left(\frac{2p\,nx}{L}\right)dx = 0 \tag{12.2a}$$

$$\int_{-L/2}^{L/2} \cos\left(\frac{2p\,nx}{L}\right)dx = \begin{cases} 0, n \neq 0 \\ L, n = 0 \end{cases} \tag{12.2b}$$

$$\int_{-L/2}^{L/2} \sin\left(\frac{2p\,nx}{L}\right)\cos\left(\frac{2p\,mx}{L}\right)dx = 0 \tag{12.2c}$$

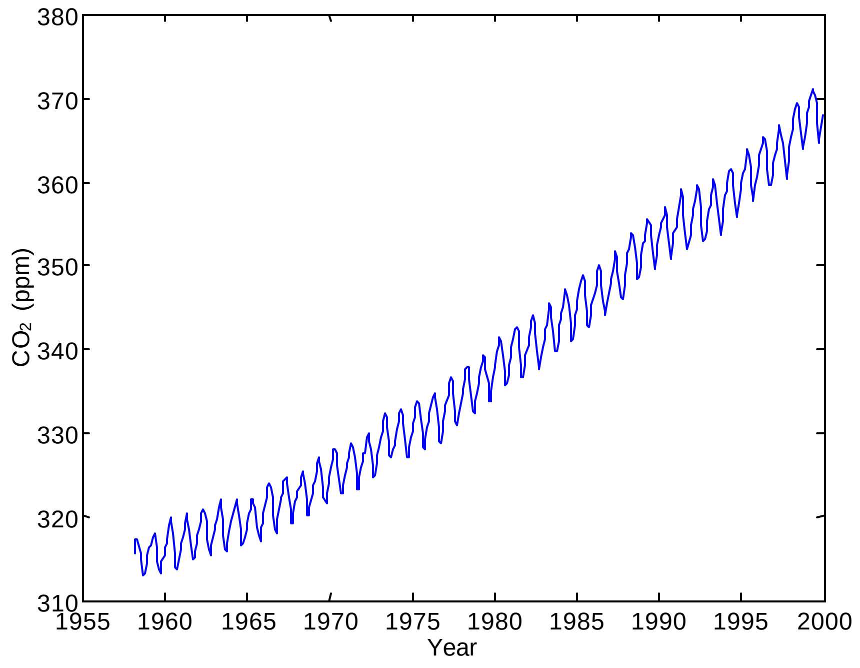


Figure 12.1. Atmospheric concentration of CO_2 measured at Mauna Loa, HI

$$\int_{-L/2}^{L/2} \sin\left(\frac{2p\,nx}{L}\right)\sin\left(\frac{2p\,mx}{L}\right)dx = \begin{cases} 0, n \neq m \\ L/2, n = m \end{cases} \tag{12.2d}$$

$$\int_{-L/2}^{L/2} \cos\left(\frac{2p\,nx}{L}\right)\cos\left(\frac{2p\,mx}{L}\right)dx = \begin{cases} 0, n \neq m \\ L/2, n = m \end{cases} \tag{12.2e}$$

To find the coefficient a_0 of the Fourier series, we integrate $f(x)$ over the interval $-L/2$ to $L/2$ and use the relationships in equations (12.2) to simplify the result.

$$\int_{-L/2}^{L/2} f(x)dx = \int_{-L/2}^{L/2} \frac{a_0}{2}dx + \sum_{n=1}^{\infty}\int_{-L/2}^{L/2} a_n \cos\left(\frac{2p\,nx}{L}\right)dx + \sum_{n=1}^{\infty}\int_{-L/2}^{L/2} b_n \sin\left(\frac{2p\,nx}{L}\right)dx$$

$$= \frac{a_0 L}{2}$$

or

$$a_0 = \frac{2}{L}\int_{-L/2}^{L/2} f(x)dx \tag{12.3}$$

Note that the 2nd and 3rd terms on the right-hand-side of the first expression disappear because of (12.2a) and (12.2b). The resulting equation for a_0 is just twice the average of $f(x)$ over the interval. Thus, the leading coefficient in the Fourier series, $a_0/2$, represents the average of the function.

To find the other a_n coefficients (for n=1,2 …), we multiply all terms in (12.1) by cos(2*pmx/L*) (*m* is any positive integer) and integrate

$$\int_{-L/2}^{L/2} f(x)\cos\left(\frac{2p\,mx}{L}\right)dx = \int_{-L/2}^{L/2} \frac{a_0}{2}\cos\left(\frac{2p\,mx}{L}\right)dx$$
$$+\sum_{n=1}^{\infty}\int_{-L/2}^{L/2} a_n\cos\left(\frac{2p\,nx}{L}\right)\cos\left(\frac{2p\,mx}{L}\right)dx+\sum_{n=1}^{\infty}\int_{-L/2}^{L/2} b_n\sin\left(\frac{2p\,nx}{L}\right)\cos\left(\frac{2p\,mx}{L}\right)dx$$

The 1st integral on the right-hand-side is zero (12.2b) and the 3rd integral is zero (12.2c). The 2nd integral is zero except when $m=n$, and then it equals $a_nL/2$ (12.2e). Therefore, all that is left is

$$a_n = \frac{2}{L}\int_{-L/2}^{L/2} f(x)\cos\left(\frac{2p\,nx}{L}\right)dx \qquad n=1,2,3,\ldots \tag{12.4}$$

We follow the same procedure to find b_n, except that now we multiply each term by sin(2*pmx*/*L*) before integrating

$$\int_{-L/2}^{L/2} f(x)\sin\left(\frac{2p\,mx}{L}\right)dx = \int_{-L/2}^{L/2} \frac{a_0}{2}\sin\left(\frac{2p\,mx}{L}\right)dx$$
$$+\sum_{n=1}^{\infty}\int_{-L/2}^{L/2} a_n\cos\left(\frac{2p\,nx}{L}\right)\sin\left(\frac{2p\,mx}{L}\right)dx+\sum_{n=1}^{\infty}\int_{-L/2}^{L/2} b_n\sin\left(\frac{2p\,nx}{L}\right)\sin\left(\frac{2p\,mx}{L}\right)dx$$

which, using (12.2), reduces to

$$b_n = \frac{2}{L}\int_{-L/2}^{L/2} f(x)\sin\left(\frac{2p\,nx}{L}\right)dx \qquad n=1,2,3\ldots \tag{12.5}$$

12.3. Example: Square wave

Consider a square wave (Figure 12.2),

$$f(x)=\begin{cases}-k & \text{when } -p<x<0\\ k & \text{when } 0<x<p\end{cases}$$

and

$$f(x+2p)=f(x)$$

This is clearly periodic (period=2*p*), but not sinusoidal. The average of the function over the interval -*p* to *p* is zero, so a_0=0. The function is odd (antisymmetric about the origin). As a result the Fourier coefficients a_n of the cosine terms (even terms) are also zero. To find the remaining Fourier coefficients, we use equation (12.5) to get

$$b_n=\frac{1}{p}\left[\int_{-p}^{0}-k\sin(nx)\,dx+\int_{0}^{p}k\sin(nx)\,dx\right]=\frac{k}{p}\left[\left.\frac{\cos(nx)}{n}\right|_{-p}^{0}-\left.\frac{\cos(nx)}{n}\right|_{0}^{p}\right]$$
$$=\frac{k}{p\,n}\left[\cos(0)-\cos(-np)-\cos(np)+\cos(0)\right]=\frac{2k}{p\,n}\left[1-\cos(np)\right]$$

The resulting b_n are zero for even n and $4k/(np)$ for odd n. So the Fourier series describing this square wave is

$$f(x) = \frac{4k}{p}\left(\sin(x) + \frac{1}{3}\sin(3x) + \frac{1}{5}\sin(5x) + \dots\right)$$

The original square wave and the Fourier series approximation to $f(x)$ when 1, 2, or 10 terms of the infinite series are retained are shown in Figure 12.2. The shape of the square wave is relatively well captured using 10 terms of the series, but noticeable oscillations in the approximation are still present, particularly just before and after the vertical jumps in the function.

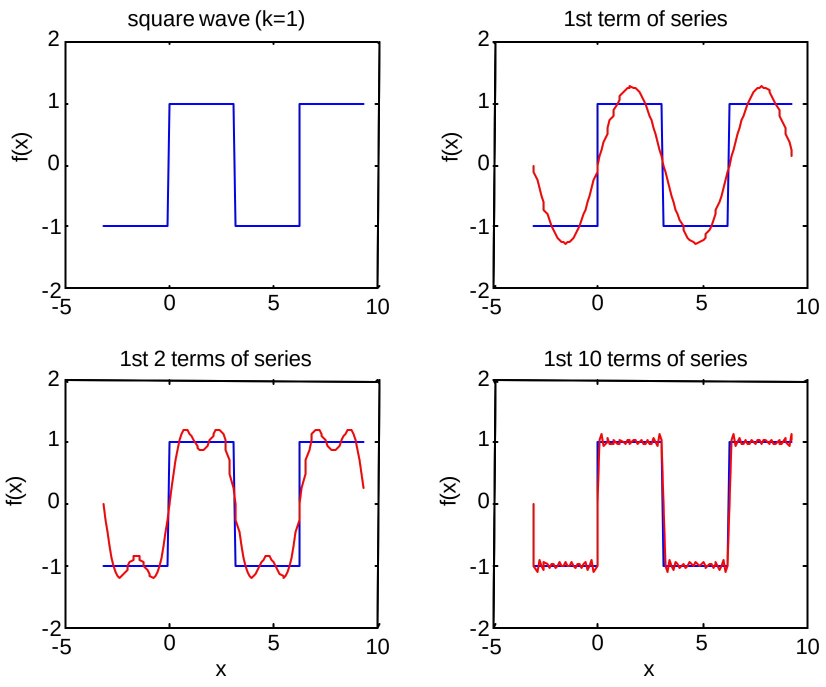


Figure 12.2. Fitting of Fourier coefficients to a square wave.

12.4. Example: CO_2 time series

In the first example, we determined the Fourier coefficients for a function that could be expressed algebraically. When analyzing measured time series using Fourier analysis, we generally are not able to represent the data with a simple algebraic expression. However, we have already investigated methods of numerical integration and their application to experimental data. In this example, we'll use one of these methods to determine the Fourier coefficients for the Mauna Loa CO_2 time series (Figure 12.1). We'll also explore *MATLAB*'s `fft` (Fast Fourier Transform) function that does most of the work for us.

In the following m-file, the *MATLAB* function `trapz` (Chapter 3.8) is used to perform the necessary integrations. The coefficients a_n and b_n are determined in a loop after detrending the data by subtracting a linear trend (top panel of Figure 12.3). It is generally advisable to detrend a time series before performing a Fourier analysis for several reasons. From a theoretical standpoint, Fourier analysis assumes a time series is *stationary* (essentially, that is has no systematic trend in mean or variance). Detrending is also helpful from a practical point of view because retaining a trend can result in large coefficients at low frequencies (small values of *n*) that can dominate the results of the analysis.

When dealing with sampled data, such as the CO_2 time series, we can fit only a finite number of the terms in the infinite Fourier series. The maximum number of coefficients we can determine is constrained by the length of the time series and the sampling interval. We can never resolve frequencies higher than twice the sampling interval (monthly in this example) because it takes at least two points to define a sinusoidal function. The analysis in this example is done for a 35-year continuous record (there can't be data gaps when doing Fourier analysis). The total record length is 35×12=420, so we could determine the Fourier coefficients up to *n*=210. In practice, we can truncate the Fourier series well before the maximum number of coefficients that is theoretically possible. In this example, 35×4=140 coefficients are determined, enough to resolve a seasonal signal in the time series. We have taken the fundamental period *T* to be the length of the time series (in years), so that the coefficients for *n*=1 correspond to a signal with a 35-year period.

```
%co2anal.m   %finds Fourier coefficients using numerical integration
%
load maunaloa.co2   %matrix with 1 row for each year; the 1st column is
%                    the year and the next 12 columns are monthly values
y=maunaloa(:,1); [a,b]=size(maunaloa);
co2=maunaloa(:,2:b-1);
co2v=reshape(co2',(b-2)*a,1); %co2v is a row vector of all the data
m=[0:11]'./12;  M=repmat(m,length(y),1);
Y=repmat(y,1,12); Y=reshape(Y',(b-2)*a,1);
ym=Y+M;  %row vector of decimal year corresponding to each data value
i=find(co2v~=-99.99);  %-99.99 indicates missing data values

%use data section corresponding to an integer number of years without
%missing values
co2z=co2v(85:length(co2v)); ymz=ym(85:length(co2v));
%detrend data -- could also use MATLAB ‘detrend’ function
p=polyfit(ymz,co2z,1);
z=co2z-(p(1).*ymz+p(2));
ny=length(ymz)./12;  %length of the time series (in years)

%fit Fourier coefficients
a0=2.*mean(z);
for i=1:4*ny;
  a(i)=2./ny.*trapz(ymz,z.*cos(ymz*2*pi*i/ny));
  b(i)=2./ny.*trapz(ymz,z.*sin(ymz*2*pi*i/ny));
  s(:,i)=a(i).*cos(ymz*2*pi*i/ny)+b(i).*sin(ymz*2*pi*i/ny);
end;
fs=a0./2+sum(s,2);  %sum the terms to get full function

subplot(211), plot(ymz,z,'b',ymz,fs,'r')
xlabel('Year'); ylabel('Detrended CO_2 (ppm)');
x=1:4*ny;
subplot(425), stem(x,a,'bo')
hold on
```

```
plot(0,a0,'*',[0 2.5*ny],[0 0],'-k'); hold off
axis([-0.5 2.5*ny+0.5 -1 3])
ylabel('a_n')
subplot(427), stem(x,b,'ro')
hold on
plot([0 2.5*ny],[0 0],'-k'); hold off
axis([-0.5 2.5*ny+0.5 -1 3])
ylabel('b_n'); xlabel('n')
for i=1:4*ny,
   r(i)=sqrt(mean((z-sum(s(:,1:i),2)).^2));
end
subplot(224), plot(1:4*ny,r)
axis([-0.5 4*ny+0.5 0 2.5])
xlabel('n'); ylabel('RMS difference')
```

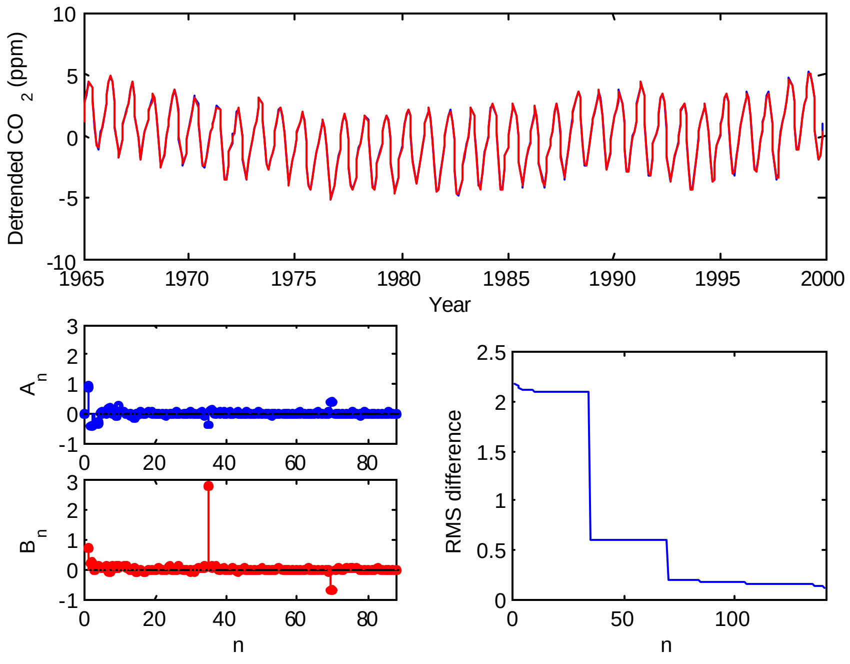


Figure 12.3. Fourier analysis of Mauna Loa CO_s time series.

The original time series and its Fourier series approximation are shown in the top panel of Figure 12.3 – only one curve is visible because they lie right on top of each other. Most of the calculated Fourier coefficients (lower left of Figure 12.3) are close to zero. Non-zero values for small values of *n* represent the low frequency variation apparent in the signal. The magnitude of these coefficients is relatively small, however, because the amplitude of the low-frequency variation is much smaller than the amplitude of the yearly variation. The largest coefficient is associated with *n*=35, the number of years in the time series, indicating a frequency of one cycle per year (period of 1 year). The b_{35} coefficient is much larger then the a_{35} coefficient, indicating that the phase of the annual signal is closer to a sine than a cosine wave. There is also a semiannual (seasonal) signal represented by the non-zero coefficients at *n*=70 (frequency of twice per year). The lower right panel shows the root-mean-square (rms) difference between the measured time series and the estimated time series as a function of the number of Fourier coefficients used to approximate the function. The figure clearly shows that the annual and semiannual signals account for most of the observed variation in the time series. The top panel shows the original time series (in blue) and the Fourier series approximation (in red). They plot essentially on top of each other, indicating that the approximation provides a good representation of the original series.

12.5. *MATLAB* methods

The approach to determining the Fourier coefficients in the m-file above is straightforward, but not very efficient. Applied mathematicians have developed a more efficient method for calculating these coefficients called the "fast Fourier transform" or fft. *MATLAB* has a built-in fft function that calculates the Fourier coefficients as complex coefficients, $c_n = \sum_{k=0}^{N-1} f(k)\, e^{-i2pnk/L}$. c_n [the discrete Fourier transform (DFT) of $f(t)$] is related to the Fourier coefficients a_n and b_n as: $a_0 = c_0,\ a_n = c_n + c_{-n},\ b_n = i\left(c_n - c_{-n}\right)$ [Box 12.1]. We can also write $f(k) = \sum_{n=-\infty}^{\infty} c_n\, e^{i2pnk/L}$ (analogous to equation 12.1), which is termed the *inverse* Fourier transform. *MATLAB* provides this information about their fft function[1]:

```
FFT Discrete Fourier transform.
    FFT(X) is the discrete Fourier transform (DFT) of vector X.  For
    matrices, the FFT operation is applied to each column. For N-D
    arrays, the FFT operation operates on the first non-singleton
    dimension.

    FFT(X,N) is the N-point FFT, padded with zeros if X has less
    than N points and truncated if it has more.

    FFT(X,[],DIM) or FFT(X,N,DIM) applies the FFT operation across the
    dimension DIM.

    For length N input vector x, the DFT is a length N vector X,
    with elements
                     N
       X(k) =       sum  x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
                    n=1
    The inverse DFT (computed by IFFT) is given by
                     N
       x(n) = (1/N) sum  X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
                    k=1

    See also IFFT, FFT2, IFFT2, FFTSHIFT.
```

We can see the correspondence between the results of *MATLAB*'s `fft` function and the direct calculation of the Fourier coefficients (as in the earlier m-file) by returning to our last example. Calculating

```
X=fft(z)
```

in *MATLAB*, we obtain a complex vector of length *N*, the length of the time series *z*. (The last (*N*-1)/2 values of an fft are a mirror image of the (*N*-1)/2 values `X(2:N/2)`; the first value

[1] Reprinted with permission from MathWorks, Inc.

of *X* is a_0, the mean of *f*(*t*). For the purposes of calculations and plotting, it is common to use twice the first half of the fft rather than the full vector.) The magnitude of *X* divided by *N* (or `2*abs(X(2:N/2))/N`) is equal to the to the magnitude of the Fourier coefficients calculated in the earlier m-file (Figure 12.4).

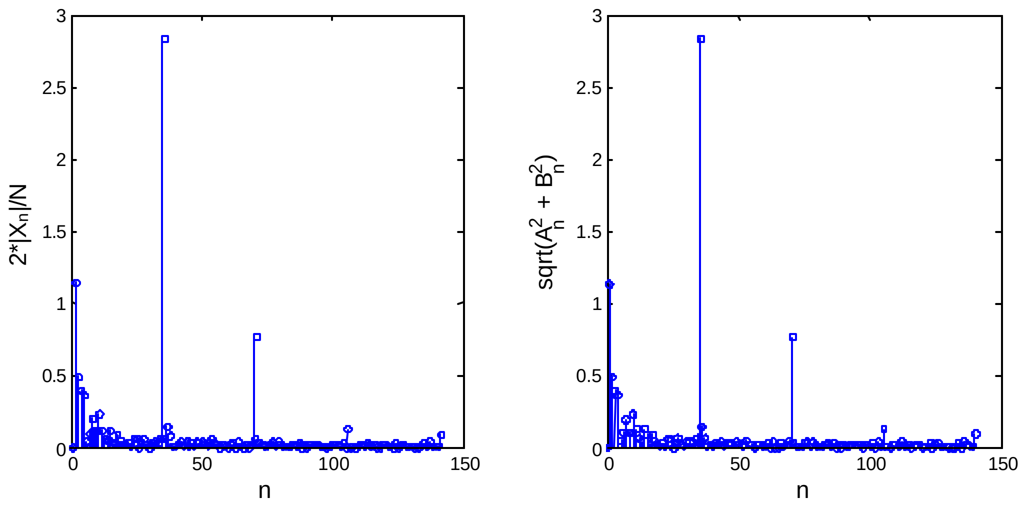


Figure 12.4. Comparison of Fourier coefficients calculated by `fft` and `co2anal.m`.

Clearly, the `fft` function provides the same information for a lot less effort than the approach used in the code presented earlier. As the *MATLAB* help file indicates, the inverse Fourier transform, `ifft(X)`, returns the original time series.

12.6. Spectral analysis and periodograms

In the figures above, the Fourier coefficients are plotted against the index *n*. It typically is more informative to plot the coefficients against the frequency of the corresponding sinusoidal function. The lowest frequency that can be resolved (corresponding to *n*=1) is dictated by the length of the time series, i.e., one cycle over the whole record. The highest frequency that can be resolved (corresponding to *n*=*N*/2, where *N* is the length of the time series) is dictated by the sampling interval *dt*, and is called the Nyquist frequency, 1/(2**dt*).

We can calculate the frequency associated with each Fourier coefficient as

```
f=(0:N/2)./(N*dt);
```

where *f*=0 corresponds to a_0. So, for example, if we plot half the normalized magnitude squared of the 1st *N*/2+1 terms of the fft of the Mauna Loa CO_2 time series (`2*abs(X(2:N/2)^2)/N^2`) against frequency, we obtain the result shown in Figure 12.5a. The peaks are located at the frequencies that dominate the time series (account for most of its variance). In the example, the annual signal (frequency of 1/yr) is the largest peak; the half-yearly (2/yr) component is the second largest peak. There are two other smaller peaks at higher frequencies (3/yr and 4/yr) and some smaller peaks at lower frequencies.

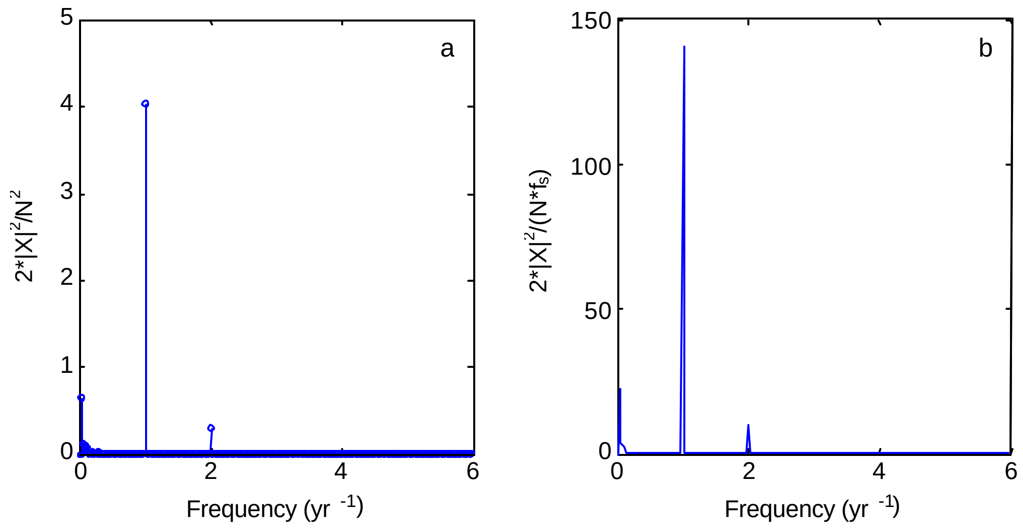


Figure 12.5. Discrete (a) and continuous (b) periodograms of Mauna Loa CO_2 data.

If we take the sum $\frac{2}{N}\sum_{n=0}^{N/2}\left|X_n^2\right|$ or $\frac{1}{N}\sum_{n=-N/2}^{N/2}\left|X_n^2\right|$, we obtain the total sum of squared deviations from the mean ($\sum_{i=1}^{N}\left(z_i-\overline{z}\right)^2$), which, when divided by N (the length of the time series), is the variance of the original time series apart from a factor of $N/(N-1)$ (Parseval's theorem; e.g., Chatfield, 2004). Therefore we can think of each value plotted in Figure 12.5a as representing the contribution of that frequency to the total variance of the time series. The sum of the values plotted in Figure 12.5a is 5.43 ppm^2; the total variance of the original time series is 5.44 ppm (N=420). We can also find the amplitude of the signal at a given frequency as (2*variance at that frequency)^{½}. So, for example, the annual signal has a variance of 4.0 ppm^2 (Figure 12.5a), giving an amplitude of 2.8 ppm (Figure 12.4) or an annual variation (high value – low value) of 5.7 ppm.

Most phenomena in the physical sciences operate over a continuous range of frequencies rather than a discrete set as plotted in Figure 12.5a. To represent the results as a continuous spectrum, we adjust the magnitudes of the coefficients so that the integral of the spectrum (total area under the curve) is equal to the variance of the original time series by multiplying the values plotted in Figure 12.5a by N/f_s, where f_s is the sampling frequency (= 12/year for our example) (Chatfield, 2004). The resulting continuous spectrum (Figure 12.5b), called a periodogram, provides an estimate of the power spectral density (psd) and shows how variance is distributed among the component frequencies in a time series; the integral over a specific range of frequencies is the variance associated with those frequencies. If we use `trapz` to calculate the integral of the full spectrum in Figure 12.5b, we again get 5.43 ppm^2.

Periodograms are informative and relatively easy to calculate, but it turns out that they have some bad statistical properties. In particular, the variance associated with a given frequency does not decrease as N increases. As a result, it is not a "consistent" estimator of the true spectral density function (e.g., Chatfield, 2004). There are two things that can be done to obtain a better estimate. One is to break the time series up into smaller segments, calculate the spectrum for each segment, and then average the results. The segments are often overlapping. This is referred to as Welch's method. The other thing that can be done is to "window" the periodogram. We can think of dividing a time series up into segments as being equivalent to multiplying the time series by a box-car function (Figure 12.6) that has values of one over the desired segment and zeros everywhere else. For reasons that are too involved to go into here, a better spectral estimate can be obtained by replacing the simple box-car function used in the periodogram with one of a number of "windows" that are still zero outside the desired range, but have different shapes inside the range. One example is the Tukey-Hanning window (Figure 12.6), which is used in the m-file below.

MATLAB has a number of functions for spectral analysis in their signal-processing tookbox, including `periodogram`, `pwelch` and `pmtm`. `pwelch` makes use of averaging and windowing; `pmtm` is a multitaper method that employs a combination of windows. These functions calculate the power spectral density (psd) with units of power per unit frequency. (The power spectrum is given by the psd times the sampling frequency). *MATLAB* help provides more information on these and other spectral functions. The following simplified function file, `specclc.m`, calculates the psd of a time series using averaging and a Tukey-Hanning window. The function assumes a window length of 256 with a 50% overlap – a good choice for many problems – but the window length can be changed. The windows used in most windowing methods (non-rectangular windows such as the Tukey-Hanning window (Figure 12.6)) affect the power of the signal. This can be compensated for by normalizing the window so that its average power is 1. Following *MATLAB*[2], `specclc` uses a normalizing factor equal to the sum of squares of the window coefficients divided by window length – like the formulation used for the fft of the time series itself.

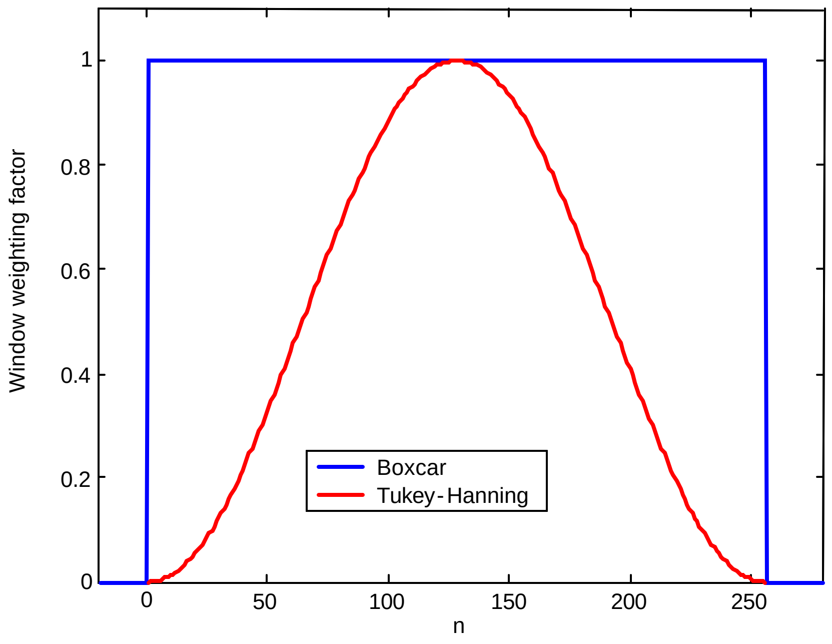


Figure 12.6. Boxcar and Tukey-Hanning windows.

```
function [cx,fx]=specclc(x,fs)
```

[2] *MATLAB* Signal Processesing Toolbox User's Guide, Version 6, MathWorks, Inc.; available through *MATLAB's* help section on Non-Parametric Spectral Estimation Methods.

```
%x = time series, fs = sampling frequency (1/dt)
%cx = vector of Fourier coefficients, fx = vector of frequencies

nw=256; %nw=window length (must be even)
no=nw/2;  %no=overlap length (50%);
nx=length(x); x=x(:);  %nx is length of record
if nx<nw, x(nx+1:nw)=0; n=nw; end; %add zeros if n<nw
nseg=floor(2*nx/nw-1);  %calculate number of segments
iw=[0:no-1]';  %window index
halfwin=0.5.*(1-cos(pi*iw/(no-1)));  %Tukey-Hanning window
win=[halfwin;flipud(halfwin)];
sswn=sum(abs(win.^2))/nw;  %sum of squares of window / nw
csum=zeros(nw,1);
for i=1:nseg,
  iseg=[1:nw]'+(i-1)*no;   %segment index
  xseg=win.*x(iseg);
  cseg=abs(fft(xseg,nw)).^2;    %fft of segment
  csum=csum+cseg;   %sum contributions from each segment
end;
cx=csum./nseg  %average summed fft
%cx is 2*normalized first half of average summed fft
cx=2*cx(1:no+1)./(nw*fs*sswn)
fx=(0:no)'*fs/nw;  %fx is frequency vector
```

We can view the psd by plotting cx vs. fx.

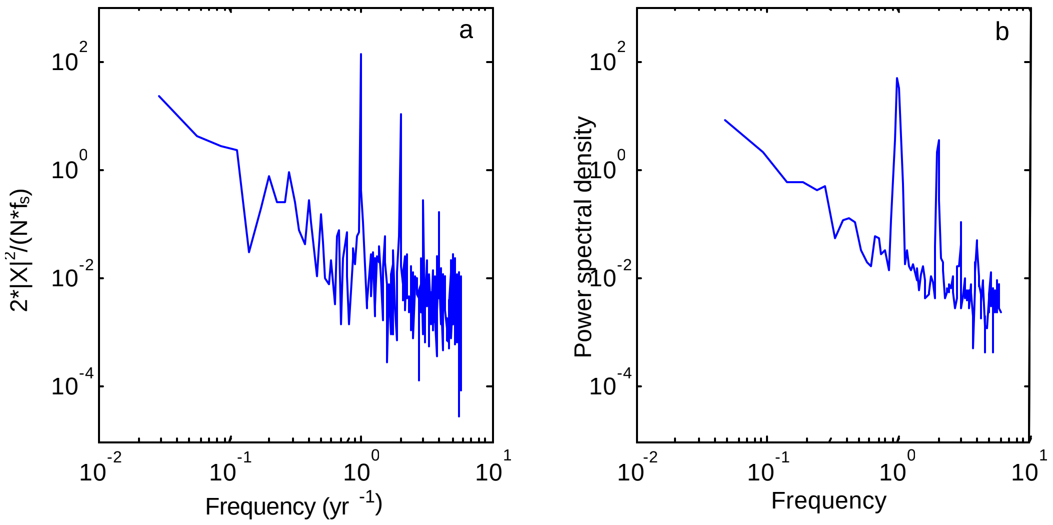


Figure 12.7. (a) Periodogram and (b) psd calculated using specclc for the Mauna Loa CO_2 data.

The application of `specclc` to the Mauna Loa time series is shown in Figure 12.7b; the spectrum shown in Figure 12.7a is the same as the one in Figure 12.5b, but plotted on log-log axes as is common for spectral data. Comparing Figure 12.7a with Figure 12.7b, we can see that the effect of averaging and windowing is to smooth the power spectral density function while preserving the main spectral peaks and their relative magnitudes. There are several large peaks in the psd as well as many smaller ones. To know which of these are significant, it is necessary to calculate confidence intervals for the psd. There is an option to calculate confidence intervals in the *MATLAB* function `pmtm`. The integral of the psd in Figure 12.7b (e.g., `trapz(fx,cx)`), 5.11 ppm^2, is still approximately equal to the variance of the input time series, although windowing and averaging result in small differences in values. To find the contribution of a particular range of frequencies, e.g., those spanning the width of a peak in the psd, to the total variance, we can calculate the ratio of the integral over the specified range of frequencies to the integral of the entire psd.

12.7. Filtering time series

When analyzing time series, we are often more interested in some frequencies than others. For example, we might be interested in determining the flood signal in water levels in a tidally influenced channel or the multi-year climatic signal in a discharge record. In these cases, we would be concerned with frequencies lower than tidal or annual frequencies, respectively. In other cases, we might be most interested in the high frequency portion of a time series. To select or eliminate a given range of frequencies from a time series, we *filter* the data. This can be done in the *time domain* (e.g., using a running average, which is very good at smoothing data but is poor at preserving information about phase and doesn't have as sharp a frequency cutoff) or in the *frequency domain*. We will focus on the latter. If we are interested in the low frequencies, we want to *low-pass* filter the data. To retain only high frequencies, we want a *high-pass* filter. To select some intermediate range of frequencies, we must use a *band-pass* filter.

The concept behind frequency-domain filtering of a time series is straightforward once we've determined its power spectral density. Essentially, we want to apply a box-car-like function to the psd, retaining the frequencies of interest and setting the coefficients of frequencies outside the range of interest to zero. The inverse Fourier transform of the modified psd should yield a time series containing only the desired frequencies of the original time series. The biggest issue we face in doing this is the choice of the 'box-car-like' function. We want a filter that introduces the least distortion into the time series in the chosen range of frequencies. The best filters use a smoothly varying function (taper) between the 1's assigned to the frequencies to be retained and the 0's assigned to the frequencies to be removed. Optimal filter design is a problem that has received much attention. There are many approaches and many considerations. In fact, *MATLAB* has a whole toolbox devoted to filter design. Here, we consider just a simple example to illustrate the basic ideas of a frequency-domain filter.

The m-file, `lpfilt.m`, is an example of a low-pass filter for time series. The syntax is

```
xf=lpfilt(x,dt,cutoff_frequency)
```

where `x` is the original time series, `xf` is the filtered time series, `dt` is the sampling interval (used to determine the frequencies), and `cutoff_frequency` is the highest frequency to be retained. This can be modified to obtain a high-pass filter by adjusting the frequencies assigned 0's and 1's and the points assigned the taper values (0.715, 0.24, 0.024). Optimal values for the taper depend on the sampling frequency and period of the signal and the width of the frequency range to be passed in the filter (bandwidth). The taper values used in `lpfilt.m` fall near the middle of the range of values calculated by Rabiner et al. (1970; generally ±0.05 or less for 0.715 and 0.24; ±0.01 for 0.024).

```
function fdata = lpfilt(data,delta_t,cutoff_f)
```

```
%  Peforms low-pass filtering by multiplication in frequency domain using
%  a 3-point taper (in frequency space) between pass-band and stop band.
%  Coefficients suggested by D. Coats, Battelle, Ventura.
%  Written by Chris Sherwood, 1989; revised by P. Wiberg, 2003.
n=length(data);
mn=mean(data);
data=data-mn;  %linearly detrend data series
P=fft(data); N=length(P);
filt=ones(N,1);
k=floor(cutoff_f*N*delta_t);
% filt is a tapered box car, symmetric over the 1st and 2nd half of P,
% with 1's for frequencies < cutoff_f and 0's for frequencies > cutoff_f
filt(k+1:k+3)=[0.715 0.24 0.024];
filt(k+4:N-(k+4))=0*filt(k+4:N-(k+4));
filt(N-(k+1:k+3))=[0.715 0.24 0.024];
P=P.*filt;
fdata=real(ifft(P));  %inverse fft of modified data series
fdata=fdata(1:n)+mn;   %add mean back into filtered series
```

To illustrate the use of `lpfilt.m`, we can low-pass filter the Mauna Loa CO_2 record to eliminate the most prominent frequencies and focus on the variations at periods longer than a year. We use `zf=lpfilt(z,1/12,1/2)` to obtain a time series that includes only periods of two-years or longer. The result, for the detrended time series, is plotted in Figure 12.8. The low-pass filtered time series (Figure 12.8) shows that there is interannual variability in the CO_2 time series, though these variations are small compared to the annual signal (with an annual range of almost 6 ppm) and the long-term trend (about 55 ppm).

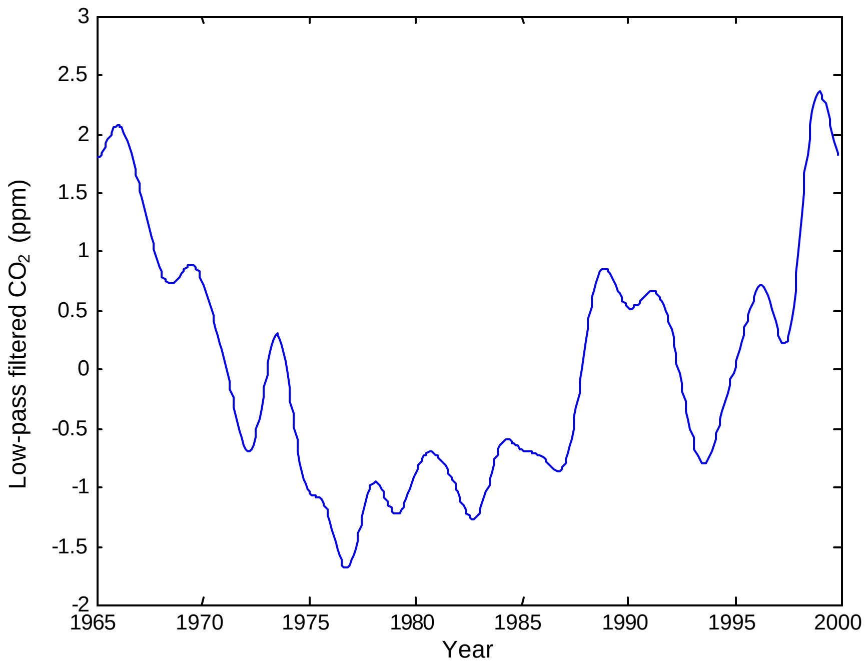
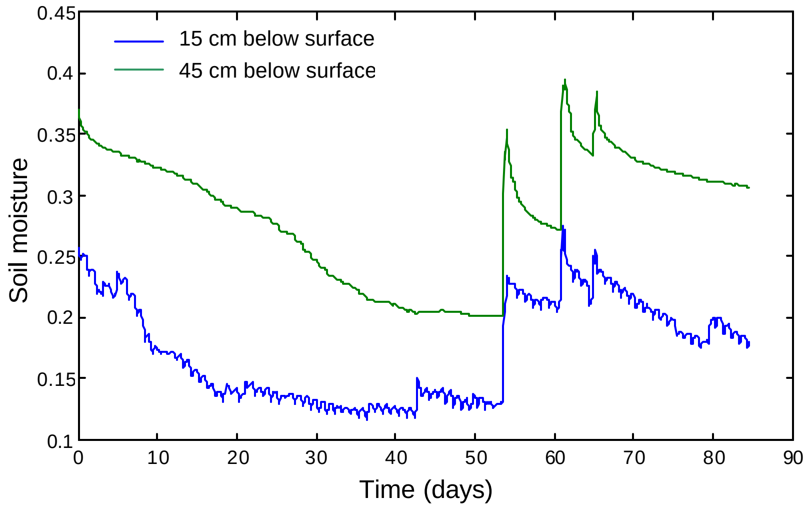


Figure 12.8. Low-pass filtered CO_2 record from Mauna Loa.

12.8. Problems

The time series of soil moisture plotted below was collected in an agricultural field at the Virginia Coast Reserve (VCR) Long-Term Ecological Research (LTER) site on the Eastern Shore of Virginia during the summer of 1998. The time series comprises hourly measurements at depths of 15 and 45 cm below the ground surface over a period of 85 days. The time series generally shows falling values of soil moisture due to evapotranspiration punctuated by sharp rises associated with precipitation. The data are in the file ltersm.dat.



1. Select a 20-day or longer segment (integer number of days) of the 15-cm record lacking any sharp rises in soil moisture. Calculate the Fourier coefficients (using `fft`) and the power spectral density (using `specclc` or one of the *MATLAB* signal processing functions such as `periodogram` or `pwelch`). What percentage of the total variance of this time series is represented by the diurnal signal? Estimate the amplitude of the diurnal variation in soil moisture.

2. a) Compare the power spectral density function for the segment used in 1) with that for the full time series. Suggest an interpretation for the difference between the two spectra.

   b) Compare the power spectral density function for the 15-cm and 45-cm records of soil moisture. Suggest an interpretation for the difference between the two spectra.

3. a) Filter the time series from the shallower depth to remove the diurnal signal. Compare the time series and spectra of the filtered and unfiltered time series to see how effective the filter was.

   b) Modify lpfilt.m to obtain a high-pass filter. Filter out everything with a lower-than-daily frequency. How does the amplitude of the signal compare with the value obtained in 1)?

12.9. References

Chatfield, C., The Analysis of Time Series: an Introductoin, 6th Ed, 333 pp, Chapman and Hall/CRC, Boca Raton, FL, 2004.

Rabiner, L.R., B. Gold, and C.A. McGonegal, An approach to the approximation problem for non-recursive digital filters. *IEEE Trans. Audio and Electroacoustics*, AU-18: 83-106, 1970.

Box 12.1. Fourier series in terms of complex numbers

Complex numbers have the form z=a+ib, where a and b are real numbers and i^2 =-1. We say that a=real(z) and b=imag(z). The real numbers that we most commonly use in mathematical calculations can be considered a subset of the system of complex numbers where b=0. All standard arithmetic operations can be done with complex numbers.

The complex variable z can also be defined in terms of its magnitude r= $\sqrt{a^2+b^2}$ and direction q as z=r(cos$q$ + $i$sin$q$); a=$r$cos$q$, b=$r$sin$q$. Defining $\cos q + i\sin q \equiv e^{iq}$ (Euler's identity), we can write z=re^{iq}; $e^{-iq} = \cos q - i\sin q$. Using Euler's identity, we can rewrite the Fourier series (equation (12.1)) as

$$f(k) = \sum_{n=0}^{\infty} \frac{a_n + ib_n}{2}(\sin\frac{2p\,nk}{L} + i\cos\frac{2p\,nk}{L}) + \frac{a_n - ib_n}{2}(\sin\frac{2p\,nk}{L} - i\cos\frac{2p\,nk}{L})$$

$$= \sum_{n=0}^{\infty} c_n e^{i2pnk/L} + \sum_{n=0}^{\infty} c_{-n} e^{-i2p\,nk/L} = \sum_{n=-\infty}^{\infty} c_n e^{i2pnk/L}$$

Expressing the Fourier series in terms of complex numbers permits a range of tools of complex analysis to be used in evaluating the Fourier coefficients. In fact, the development of complex analysis is linked to the study of Fourier analysis. Methods such as the fast Fourier transform (fft) are developed using the complex form of the Fourier transform.