8.4 Discrete Fourier Transform

From Fourier analysis in calculus we remember that any well-behaved continuous function can be described by an infinite Fourier series — namely, the sum of an infinite number of sine and cosine terms. In the case of a discrete time series with a finite number of points, we are required to have only a finite number of sine and cosine terms to fit our points exactly.

8.4.1 Definition

Using Euler's notation [$\exp(ix) = \cos(x) + i \sin(x)$, where i is the square root of -1] as a shorthand notation for the sines and cosines, we can write the discrete Fourier series representation of A(k) as:

$$\text{Inverse Transform:} \qquad A(k) = \sum_{n=0}^{N-1} F_A(n)\, e^{i\,2\pi nk/N} \qquad (8.4.1a)$$

where n is the frequency, and $F_A(n)$ is the ***discrete Fourier transform.*** We see that a time series with N data points (indexed from k=0 through N-1) needs no more than N different frequencies to describe it (actually, it needs less than N, as will be shown later).

There are a number of ways to describe frequency:

n = number of cycles (per time period $\mathbb{P}$),
$\tilde{n}$ = cycles per second = $n/\mathbb{P}$,
f = radians per second = $2\pi n/\mathbb{P} = 2\pi n/(N\Delta t)$.

A frequency of zero (n = 0) denotes a mean value. The ***fundamental frequency***, where n = 1, means that exactly one wave fills the whole time period, $\mathbb{P}$. Higher frequencies correspond to ***harmonics*** of the fundamental frequency. For example, n = 5 means that exactly 5 waves fill the period $\mathbb{P}$.

$F_A(n)$ is a complex number, where the real part represents the amplitude of the cosine waves and the imaginary part is the sine wave amplitude. It is a function of frequency because the waves of different frequencies must be multiplied by different amplitudes to reconstruct the original time series. If the original time series A(k) is known, then these coefficients can be found from:

$$\text{Forward Transform:} \qquad F_A(n) = \sum_{k=0}^{N-1} \left[\frac{A(k)}{N}\right] e^{-i\,2\pi nk/N} \qquad (8.4.1b)$$

Notice the similarity between (8.4.1a) and (8.4.1b). These two equations are called ***Fourier transform pairs.*** The second equation performs the ***forward transform***, creating a representation of the signal in ***phase space*** (another name for the frequency

or spectral domain). This process is also known as ***Fourier decomposition.*** The first equation performs the ***inverse transform***, converting from frequencies back into ***physical space.***

8.4.2 Example

Problem: Given the following 8 data points of specific humidity, q, as a function of time:

Index (k):	0	1	2	3	4	5	6	7
Time (UTC):	1200	1215	1230	1245	1300	1315	1330	1345
q (g/kg):	8	9	9	6	10	3	5	6

Perform a forward Fourier transform to find the 8 coefficients, $F_q(n)$. To check your results, perform an inverse transform to confirm that the original time series is recreated. Remember that the $F_q(n)$ coefficients are complex, each having a real and an imaginary part: $F_q(n) = F_{real}(n) + i\,F_{imag}(n)$.

Solution: $N = 8$ and $\Delta t = 15$ min. Thus, the total period is $\mathbb{P} = N\Delta t = 2$ h. Equation (8.4.1b) must be used to find $F_q(n)$. For those computer languages that accept complex numbers, (8.4.1b) can be programmed directly, where each of the A(k) data points has a real part equal to the value listed in the table, and an imaginary part of zero.

For hand calculation, we can use Euler's formula to translate (8.4.1b) back into sines and cosines:

$$F_A(n) = \frac{1}{N}\sum_{k=0}^{N-1} A(k)\cos(2\pi nk/N) \; - \; \frac{i}{N}\sum_{k=0}^{N-1} A(k)\sin(2\pi nk/N)$$

As an example, for n = 0, all of the cosines of zero are unity and all of the sines are zero. This leaves:

$$F_A(0) = \frac{1}{N}\sum_{k=0}^{N-1} A(k)$$

which is just the mean of A. For our case: $F_q(0) = 7.0 - 0.0\,i$. For n = 1 we can't make such a simplification, so we are forced to sum over all k for both the real and imaginary parts. This gives us $F_q(1) = 0.28 - 1.03\,i$. Continuing this procedure for all other n yields:

n	$F_q(n)$	n	$F_q(n)$
0	7.0	4	1.0
1	0.28 - 1.03 i	5	-0.78 + 0.03 i
2	0.5	6	0.5
3	-0.78 - 0.03 i	7	0.28 + 1.03 i

This is the answer to the first part of the problem. Note that for frequencies greater than 4, the Fourier transform is just the complex conjugate of the frequencies less than 4.

As a check of our transform, we can perform the inverse transform using (8.4.1a) directly in a computer program. Otherwise, we can use Euler's formula to write it as:

$$A(k) = \sum_{n=0}^{N-1} F_{(\text{real part})}(n) \cdot \cos(2\pi nk/N) - \sum_{n=0}^{N-1} F_{(\text{imag.part})}(n) \cdot \sin(2\pi nk/N)$$

In actuality, there are four sums, not just the two listed above. The remaining sums consist of the real part of F times the imaginary factor $i \cdot \sin(...)$, and the imaginary part of F times the real factor $\cos(...)$. Because the last half of the Fourier transforms are the complex conjugates of the first half (not counting the mean), these two sums identically cancel, leaving the two listed above. Upon performing the calculations for A(k), we do indeed reproduce the original time series.

Discussion: To graphically demonstrate that the sum of these sines and cosines does indeed equal our original series, Fig 8.5 shows each individual wave multiplied by its appropriate amplitude. As can be seen, the reconstructed time series fits perfectly the eight original data points. In between these points, however, the sum oscillates in a manner that is not necessarily realistic, but which is irrelevant because it occurs below the discretization resolution specified by the original data points.

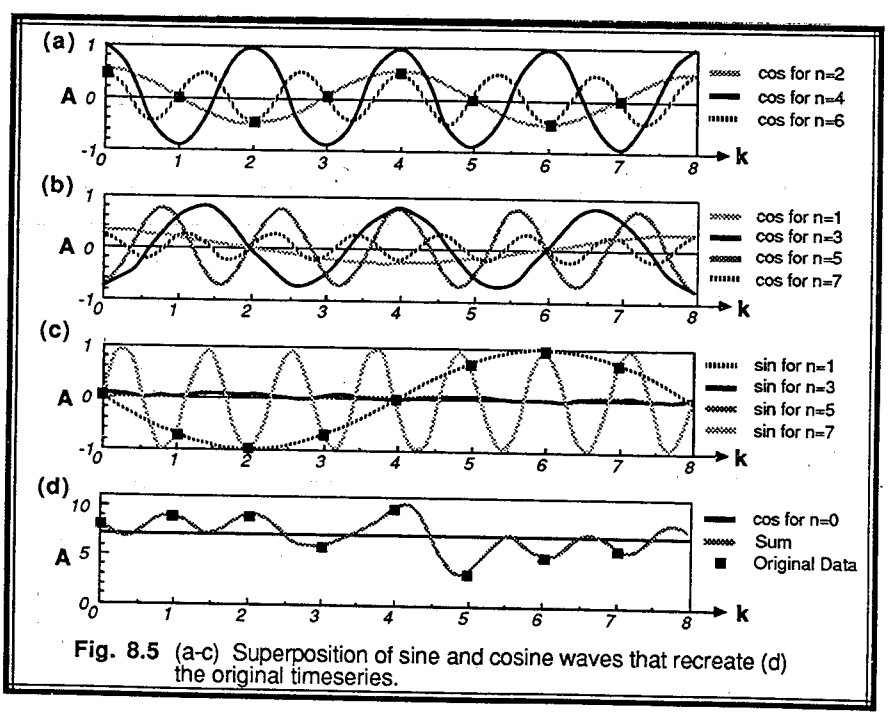

Fig. 8.5 (a-c) Superposition of sine and cosine waves that recreate (d) the original timeseries.

8.4.3 Aliasing and Other Hazards.

Measurements: A basic rule of discrete data analysis is that at least two data points are required per period or wavelength in order to resolve a wave. Since Fourier analysis involves splitting arbitrary signals into waves, the two data point requirement also holds for our arbitrary signals. For example, if we have a total of N data points, then the highest frequency that can be resolved in our Fourier transform is $n_f = N/2$, which is called the ***Nyquist frequency***. These requirements apply to measurements; namely, if a wave period as small as 0.1 s must be measured while flying in an aircraft, then the physical signal must be digitized at least once every 0.05 s. Similarly, if a wavelength as small as 1 m must be measured, then the physical signal must be digitized at least once every 0.5 m.

What happens when there is a physical signal of high frequency that is not sampled or digitized frequently enough to resolve the signal? The answer is that the true high-frequency signal is ***folded*** or ***aliased*** into a lower frequency, creating an erroneous and deceiving Fourier transform. This is illustrated with aid of the example in the previous subsection. Look at the first graph in Fig 8.5, where the cosine waves for n = 2, 4 and 6 are plotted. Since we started with N = 8 data points, we can anticipate a Nyquist frequency of $n_f = 4$. Namely, the shortest period wave that can be resolved is one that has 4 cycles per period $\mathbb{P}$. Thus, the curve corresponding to n = 6 is greater than the Nyquist frequency, and is likely to cause problems.

Look closely at the curves for n = 2 and n = 6. They coincide exactly at the points k = 0, 1, 2, 3, 4, 5, 6 and 7. In other words, if there were a true signal of n = 6 that was sampled only at the integer k values listed above, then anyone connecting the resulting plotted points with a line or curve would find that they have drawn a wave with n = 2 cycles per period. In other words, the n = 6 signal was folded into the n = 2 frequency. Similarly, looking at the third graph in Fig 8.5, the n = 7 sine waves are folded into an n = 1 sine wave. In general, if n_h represents a frequency higher than the Nyquist frequency, then the signal or amplitude of that wave will be folded down to a frequency of $n = N - n_h$, where it will be added to any true amplitude that already exists at n.

Since this folding or reflection occurs around the Nyquist frequency, it is also known as the ***folding frequency***. Such folding is readily apparent when wave amplitudes are plotted as spectral energies (to be discussed in Section 8.6). As illustrated in Fig 8.6, any nonzero wave amplitudes and spectral energies in the "true" signal at frequencies higher than the Nyquist frequency are folded back and added to the energies of the "true" signal at the lower frequencies, yielding an aliased (and erroneous) spectrum.

Aliasing is a problem whenever two conditions both occur: (1) the sensor can respond to frequencies higher than the rate that the sensor is sampled; and (2) the true signal has frequencies higher than the sampling rate. As we already know, there is a spectrum of wavenumbers and frequencies of turbulence in the atmosphere, some of which are very high. All measurement systems have limitations on the rate at which they can sample.

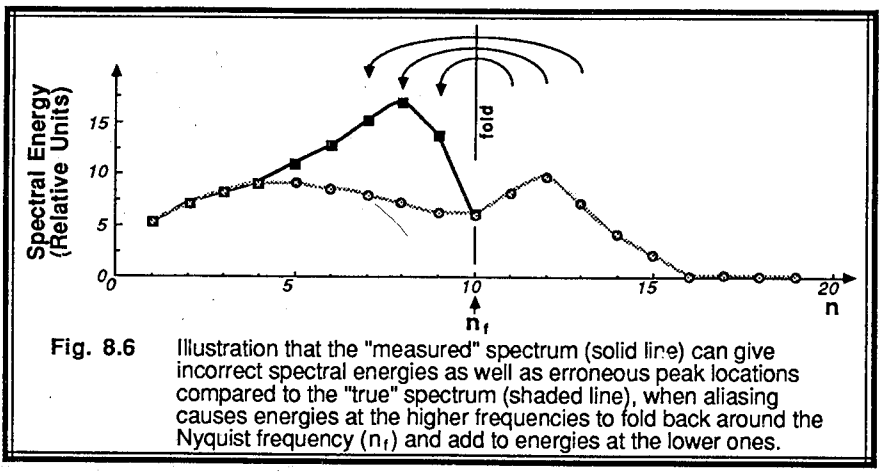

Fig. 8.6 Illustration that the "measured" spectrum (solid line) can give incorrect spectral energies as well as erroneous peak locations compared to the "true" spectrum (shaded line), when aliasing causes energies at the higher frequencies to fold back around the Nyquist frequency (n_f) and add to energies at the lower ones.

Sometimes this rate is given by limitations of the data logger or computer digitizer. If this is the case, then the raw electronic analog signal from the sensor (thermistor, thermocouple, gust probe accelerometer, etc) should be filtered by an analog electronic filter prior to the digitizing or sampling to remove frequencies higher than the Nyquist frequency. Sometimes the sensor itself has such a slow response that it performs the analog filtering automatically.

If the analog filtering is not performed, then there is no way to remove the erroneous aliased component from the resulting time series. Postprocessing of the discrete time series with digital filters will NOT work, because it is impossible to know what portion of the wave amplitude at the resolvable frequencies is real, and what is folded into it.

Digital averaging is sometimes successfully used for other reasons, however. Suppose that a sensor is designed with appropriate analog filters to yield unaliased data when sampled at a very high frequency. Next, suppose that the amount of this unbiased discrete data is too large to record in a convenient manner, or is coming in too fast to be processed. The stream of incoming discrete data values can be block averaged (e.g., average every 10 data points), or filtered with a variety of filters (e.g., Butterworth filters) before being recorded or processed further. This yields a lower-frequency sample without aliasing errors. If, however, one records only every fifth or tenth (or any interval) value from the sampled stream, then aliasing is again a problem.

Fourier analysis. Now that we are convinced that we can't resolve frequencies greater than the Nyquist frequency, why does the Fourier transform operation given by (8.4.1b) give amplitudes $F_A(n)$ up to the frequency $n = N - 1$? The answer is that it doesn't really. Looking at example 8.4.2 again, we again note that $F_q(n)$ for $n > n_f$ is just the complex conjugate of the $F_q(n)$ values for $n < n_f$. This is always the case and can be proved mathematically, assuming that the initial time series consists of only real numbers. Hence, the half of the $F_q(n)$ values for which $n > n_f$ give no new information.

Thus, the N different F_A values having both real and imaginary parts superficially gives 2N pieces of information, but since half of that is the complex conjugate of the other half, we are left with only N pieces of spectral information. It is reassuring that given only N data points in our original time series in physical space, we require only N pieces of information in phase space to precisely describe the data.

For an original time series consisting of complex numbers (2N pieces of data), the Fourier transform does not have the complex conjugate property described above, resulting in 2N pieces of information is phase space too. For meteorological data where the time series is usually real, we still need to utilize the whole Fourier transform with the complex conjugate information, because without it the inverse transform will produce complex numbers instead of our desired real number physical field.

Data Window. Fourier series apply to infinite-duration periodic data sets. Stated in other words, if we examine only a finite size record of data, the Fourier analysis implicitly assumes that the data is periodic and thus repeats itself both before and after our limited period of measurement.

In boundary layer meteorology, nothing is periodic for infinite time, or for infinite distance. Given a true signal (for example temperature) that varies as in Fig 8.7a, if we measure it over period $\mathbb{P}$ (our ***data window***) as in Fig 8.7b, then we are left with the segment shown in Fig 8.7c. Given this segment, the Fourier analysis assumes that it is dealing with a periodic (repeating) signal as shown in Fig 8.7d.

In this example, a smoothly varying meteorological signal appears as a saw-tooth pattern to the Fourier analysis. From basic calculus, recall that a Fourier analysis can indeed describe series such as sawtooth or square wave patterns, but a wide range of frequencies are required to get the sums of all the sines and cosines to make the sharp bends at the points of the teeth. These spurious frequencies are called ***red noise*** by analogy to visible light because they appear at the low frequency end of the spectrum. To avoid red noise, we must at the very least ***detrend*** the data series by subtracting the straight line best-fit from the data segment (Fig 8.7c), leaving a modified time series as exemplified in Fig 8.7e.

In general, any very low frequency that has a period longer than our whole sampling period will also generate the noise. If we know a-priori the period of this frequency, such as diurnal or annual, then we can perform a least-squares fit of this frequency to the time series and subtract the result from the series. Otherwise, we might try to fit a simple polynomial curve to the data and subtract it to both detrend it and remove these low frequencies.

Even after detrending, the sharp edges of the data window cause what is known as ***leakage***, where spectral estimates from any one frequency are contaminated with some spectral amplitude leaking in from neighboring frequencies. To reduce leakage, a modified data window with smoother edges is recommended, such as is shown in Fig 8.7f. Although a variety of smoothers can be used, a common one utilizes sine or cosine squared terms near the beginning and ending 10% of the period of record, and is known as a ***bell taper***:

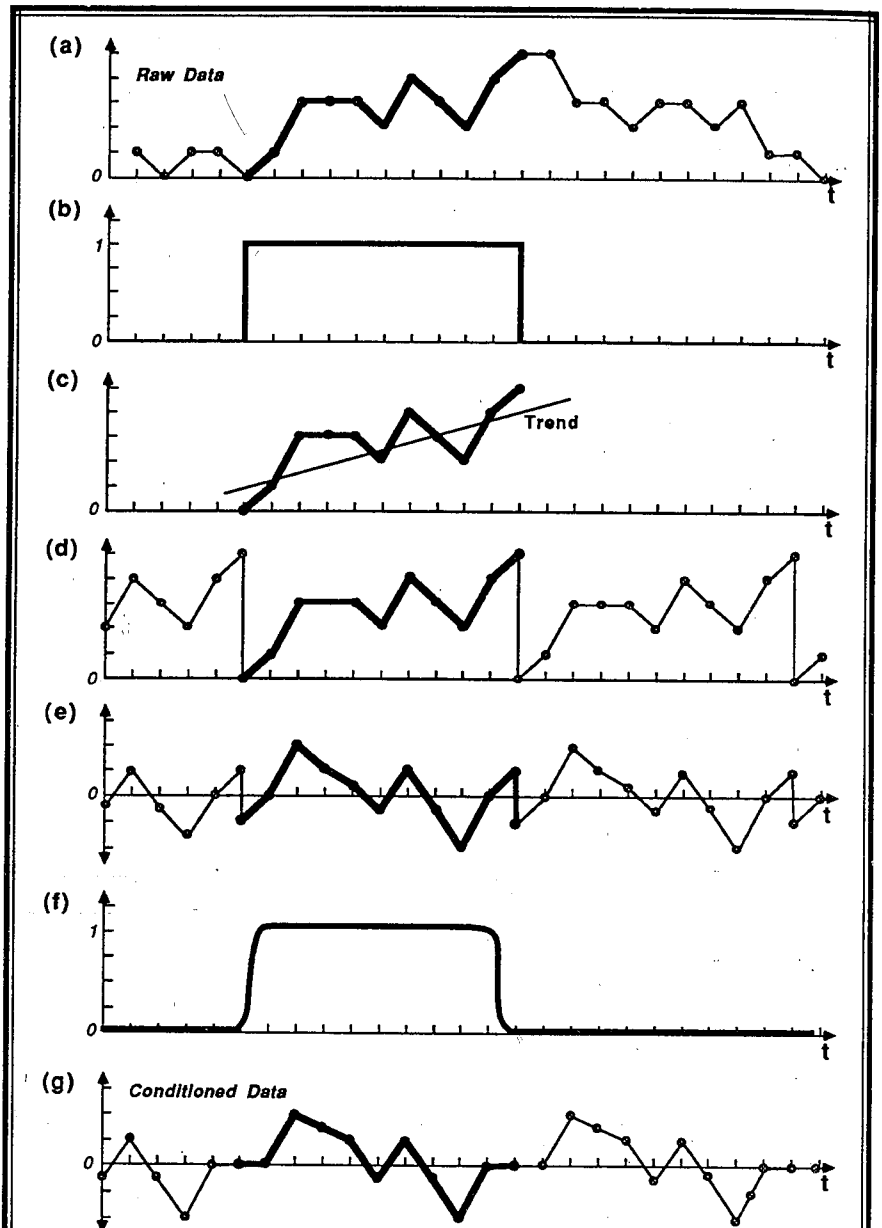

Fig. 8.7 A true signal (a) sampled over a finite time (b) gives the segment (c) that is assumed to be periodic (d) by the Fourier transform. When the detrended signal (e) is sine or cosine tapered (f) the result is called "conditioned" data (g). This "conditioned" data (g) can then be used by FFT routines.

$$W(k) = \begin{cases} \sin^2(5\pi k/N) & \text{for } 0 \le k \le 0.1N \\ 1 & \text{elsewhere} \\ \sin^2(5\pi k/N) & \text{for } 0.9\,N \le k \le N \end{cases} \qquad (8.4.3)$$

When this window weight, W(k), is multiplied by the time series, A(k), the result yields a modified time series with fluctuations that decrease in amplitude at the beginning and end of the series (see Fig 8.7g). The Fourier transform can then be performed in this modified time series.

The bell taper data window is not without its problems. Although the tapered ends reduced the leakage, they also reduce our ability to resolve spectral amplitude differences between small changes in frequencies. Also, the tapered window reduces high-frequency noise at the expense of introducing low-frequency noise.

The process of detrending, despiking (removing erroneous data points), filtering, and bell tapering is known as ***conditioning*** the data. Conditioning should be used with caution, because anytime data is modified, errors or biases can be introduced. The best recommendation is to do as little conditioning as is necessary based on data quality.

8.5 Fast Fourier Transform

The fast Fourier transform, or ***FFT***, is nothing more than a discrete Fourier transform that has been factored and restructured to take advantage of the binary computation processes of the digital computer. As a result, it produces the same output, and has the same limitations and requirements as the discrete transform. It can also be used for forward as well as inverse transforms. The description that follows is not meant to be a comprehensive review of FFT methods, but is designed to give an overview of the process.

In general, both the forward and the inverse discrete transform can be written as

$$X = \sum_{\substack{k=0 \text{ (forward)} \\ n=0 \text{ (inverse)}}}^{N-1} Y\,Z^{nk} \qquad (8.5a)$$

where

Forward Transform	Inverse Transform
$X(n) = F_A(n)$	$X(k) = A(k)$
$Y(k) = A(k)/N$	$Y(n) = F_A(n)$
$Z_N = \exp(-i2\pi/N)$	$Z_N = \exp(i2\pi/N)$

The decimal numbers n and k can be represented by their binary equivalents:

$$n = \sum_{j=0}^{\infty} 2^j n_j \qquad \text{and} \qquad k = \sum_{j=0}^{\infty} 2^j k_j \tag{8.5b}$$

where n_j and k_j represent the individual bits of the number. For example, if N = 8, then we need only three bits (j = 0 to 2) to represent n and k, since they can take on values of only 0 to 7. Thus $n = 4 \cdot n_2 + 2 \cdot n_1 + 1 \cdot n_0$. For example, 101 is the binary representation of the decimal 5, giving $n_2 = 1$, $n_1 = 0$, and $n_0 = 1$.

Using this binary representation, any function of n is now a function of n_2, n_1, and n_0, with similar forms for functions of k. Thus, X(n) becomes $X(n_2, n_1, n_0)$. Equation (8.5a) can now be rewritten, using the forward transform with N = 8 as the example, as:

$$X(n_2, n_1, n_0) = \sum_{k_2=0}^{1} \sum_{k_1=0}^{1} \sum_{k_o=0}^{1} Y(k_2, k_1, k_0)\, Z^{(4n_2 + 2n_1 + n_o)(4k_2 + 2k_1 + k_o)}$$

Performing the multiplications in the exponent of Z, rearranging terms, and remembering that Z to certain powers equals unity because of the nature of sines and cosines, we find:

$$X(n_2, n_1, n_0) = \sum_{k_o=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} Y(k_2, k_1, k_0)\, Z^{4n_0k_2} Z^{4n_1k_1} Z^{2n_ok_1} Z^{4n_2k_o} Z^{2n_1k_o} Z^{n_ok_o}$$

In this last equation the Z's are essentially weighting factors. To solve this equation, the inner sum is performed, using only the first weight because it is the only weight that is a function of k_2. When the next sum is performed, the additional two weights are included. Finally, the last sum uses the remaining three weights. This pattern of solving the sums, and gradually eliminating the k bits and replacing them with n bits can be programmed recursively, requires relatively little scratch storage, and is very efficient in computer time.

To a first approximation, the normal discrete Fourier transform requires N^2 operations, while the FFT requires only $(3N/2)\log_2 N$ operations. For small data sets ($N < 100$) the resulting computer time or cost difference is insignificant for all practical purposes, because of other overhead costs such as input and output. But for a data set of 1000 points, for example, the FFT computation takes 0.5% of the time that a traditional discrete transform computation would take. There is even some microprocessor hardware available that is specially configured to run FFTs. The bottom line is that the FFT is fast.

Most modern computer centers, and some statistical packages for microcomputers,

have "canned" FFT algorithms that users can access without having to write their own. Some of the early FFT packages were restricted to data sets with $N = 2^m$, where m was any integer. This meant that data sets slightly too long were truncated to the proper size, or data sets slightly too short we lengthened by adding bogus data (often zeros or the mean value). Both of these data mutilation tricks are not recommended. Modern FFTs factor the series into a variety of prime numbers in addition to the prime number 2, resulting in very little truncation of the time series.

One problem with all discrete Fourier transforms including FFTs, is that the input must consist of equally-spaced data points. No missing data is allowed. If the data set has gaps caused by instrument failures or by spurious data spikes that were removed, then artificial data points must be inserted to fill the gap. One is not allowed simply to close the gap by bringing the remaining parts of the data set together, because this alters the periods or wavelengths present in the original signal. The artificial data points must be chosen with care, otherwise this "fudge" can destroy an otherwise unbiased data set. Data with significant gaps can be analyzed with periodogram methods instead (see Section 8.9).

8.6 Energy Spectrum

8.6.1 Discrete Energy Spectrum

In meteorology we are frequently curious about how much of the variance of a time series is associated with a particular frequency, without regard to the precise phase of the waves. Indeed for turbulence, we anticipate that the original signal is not physically like waves at all, but we still find it useful to break the signal into components of different frequencies that we like to associate with different eddy sizes.

The square of the norm of the complex Fourier transform for any frequency n is:

$$|F_A(n)|^2 = [F_{\text{real part}}(n)]^2 + [F_{\text{imag. part}}(n)]^2 \qquad (8.6.1a)$$

When $|F_A(n)|^2$ is summed over frequencies n = 1 to N-1, the result equals the total biased variance of the original time series:

$$\sigma_A^2 = \frac{1}{N}\sum_{k=0}^{N-1}(A_k - \overline{A})^2 = \sum_{n=1}^{N-1}|F_A(n)|^2 \qquad (8.6.1b)$$

Thus, we can interpret $|F_A(n)|^2$ as the portion of variance explained by waves of frequency n. Notice that the sum over frequencies does not include n=0, because $|F_A(0)|$ is the mean value and does not contribute any information about the variation of the signal about the mean. To simplify the notation for later use, define: $G_A(n) = |F_A(n)|^2$. The ratio $G_A(n) / \sigma_A^2$ represents the fraction of variance explained by component n, and is

very much like the correlation coefficient squared, r^2.

For frequencies greater than the Nyquist frequency the $|F_A(n)|^2$ values are identically equal to those at the corresponding folded lower frequencies, because the Fourier transforms of high frequencies are the same as those for the low frequencies, except for a sign change in front of the imaginary part. Also, since frequencies higher than the Nyquist cannot be resolved anyway, the $|F_A(n)|^2$ values at high frequencies should be folded back and added to those at the lower frequencies.

Thus, ***discrete spectral intensity (or energy)***, $E_A(n)$, is defined as $E_A(n) = 2 \cdot |F_A(n)|^2$, for n = 1 to n_f, with N = odd. For N = even, $E_A(n) = 2 \cdot |F_A(n)|^2$ is used for frequencies from n = 1 to $(n_f - 1)$, along with $E_A(n) = |F_A(n)|^2$ (not times 2) at the Nyquist frequency. This presentation is called the ***discrete variance (or energy) spectrum***. It can be used for any variable such as temperature, velocity, or humidity to separate the total variance into the components, $E_A(n)$, related to different frequencies. For variables such as temperature and humidity, however, we must not associate the resulting spectrum with concepts of eddy motions, because variations in these variables can persist in the atmosphere in nonturbulent flow as the "footprints" of formerly active turbulence.

The variance of velocity fluctuations, u', has the same units as turbulence kinetic energy per unit mass. Thus, the spectrum of velocity is called the ***discrete energy spectrum***. As defined above, the name "energy spectrum" is sometimes used for all variance spectra.

8.6.2 Spectral Density

Although this chapter has dealt with discrete spectra, a number of theoretical concepts such as the spectral similarity discussed in the next chapter use continuous spectral representations. Namely, instead of summing the discrete spectral energy over all n to yield the total variance, these theories assume that there is a ***spectral energy density***, $S_A(n)$ that can be integrated over n to yield the total variance.

$$\sigma_A^2 = \int_n S_A(n)\, dn \qquad (8.6.2a)$$

The spectral energy density has units of A squared per unit frequency.

We can approximate the spectral energy density by

$$S_A(n) = \frac{E_A(n)}{\Delta n} \qquad (8.6.2b)$$

where Δn is the difference between neighboring frequencies. When n is used to represent frequency, $\Delta n = 1$. For other representations of frequency such as f, we will find that Δf is not necessarily equal to unity.

The $S_A(n)$ points estimated from (8.6.2b) can then be connected with a smooth curve to represent the shape of the spectrum. An example of this was shown in Chapter 2, Fig. 2.2. Thus, even with discrete meteorological data, we can estimate spectral densities that can be compared to theories.

8.6.3 Example

Problem: Use the results from the N = 8 data point example of section 8.4.2 to calculate the discrete spectral energies for all frequencies. Plot the result in the usual presentation format for discrete spectra. Show an additional graph of the estimate of spectral density.

Solution:

n	$F_q(n)$	$\lvert F_q(n)\rvert^2$	$E_q(n)$	$S_q(n)$
0	7.0 (= mean)			
1	0.28 - 1.03 i	1.14	2.28	2.28
2	0.5	0.25	0.5	0.5
3	-0.78 - 0.03 i	0.61	1.22	1.22
$4 = n_f$	1.0	1.0	1.0	1.0
5	-0.78 + 0.03 i	0.61		
6	0.5	0.25		
7	0.28 + 1.03 i	1.14		
	Sum =	5.0	=	5.0

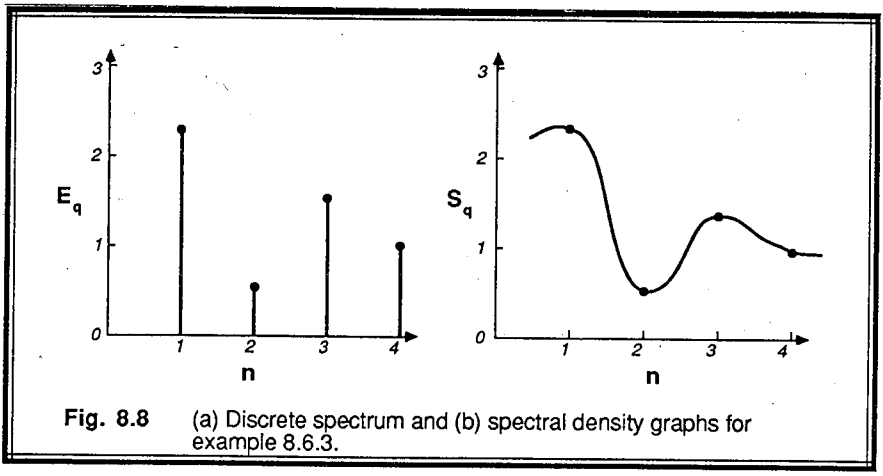

Fig. 8.8 (a) Discrete spectrum and (b) spectral density graphs for example 8.6.3.

where $E_q(n)$ has units of specific humidity squared, and $S_q(n)$ has units of specific humidity squared per unit frequency. Finally, the discrete spectrum is plotted in Fig 8.8a, and the spectral energy density is plotted in Fig 8.8b.

Discussion. The sum of the spectral energies equals the biased variance of the original signal, $\sigma_q^2 = 5.0$. This is always a good check of the FFT for you to perform.

8.6.4 Graphical Presentation of Atmospheric Spectra

A wide range of intensities are present in atmospheric turbulence spectra over an even larger range of frequencies. Atmospheric turbulence spectral energies characteristically peak at the lowest frequencies, namely at about 1 to 10 cycles per hour. At higher frequencies, the spectral energy decreases. For example, at frequencies of 10^4 cycles per hour the energy is one to two orders of magnitude smaller than at the peak.

We are often concerned about the full range of the spectrum: the peak is associated with the production of turbulence and usually the largest eddy sizes; the middle frequencies are associated with the inertial subrange, which is important for estimated dissipation rates; and the highest frequencies are associated with the dissipation of TKE into heat by viscous effects. Hence, we need a way to graphically present the spectral data in a form that not only highlights the important peaks and other characteristics, but which shows all portions of the wide range of data.

In the discussions that follow, a single idealized spectrum is presented in a variety of formats in Fig 8.9. The data for these plots is listed in Table 8-1. See Chapter 9 for examples of real atmospheric spectra.

Linear-linear presentation. When $S_A(f)$ is plotted vs. f on a linear-linear graph, the result has the desirable characteristic that the area under the curve between any pair of frequencies is proportional to the portion of variance explained by that range of frequencies. Unfortunately, the plot is useless to view because the wide range in values results in a compression of the data onto the coordinate axes (see Fig 8.9a). Alternatives include expanding the low frequency portion of the spectrum (Fig 8.9b) and plotting $f \cdot S(f)$ instead of just S(f) on the ordinate (Fig 8.9c). Both techniques focus on the spectral peak at the expense of losing information at the higher frequencies.

Note that the $f \cdot S(f)$ plot causes the apparent peak to shift from the low frequency end of the spectrum towards the middle of the spectrum. Since $f \cdot S(f)$ is also used in a number of the other formats listed below, we should not be deceived into thinking that the middle frequencies are the ones with the most spectral energy.

Semi-log presentation. By plotting $f \cdot S_A(f)$ vs. log f, the low frequency portions of the spectra are expanded along the abscissa. Also, the ordinate for the high frequency portions are enhanced because the spectral density is multiplied by frequency (see Fig 8.9d). Another excellent quality is that the area under any portion of the curve continues to be proportional to the variance.

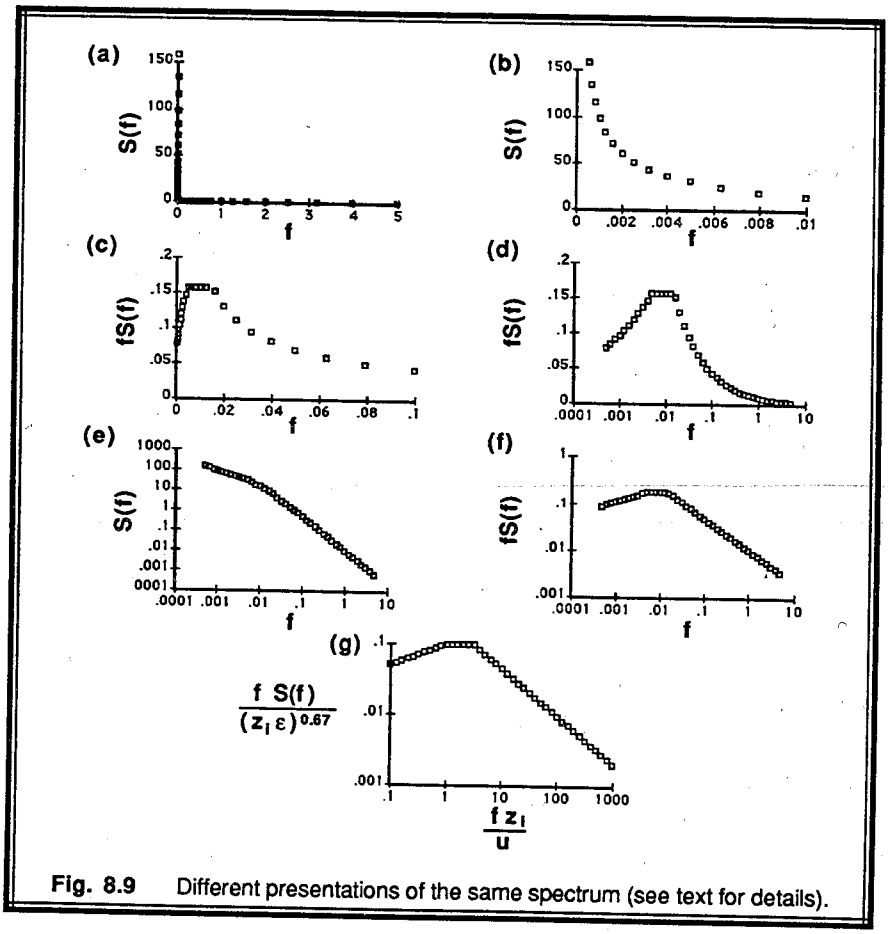

Fig. 8.9 Different presentations of the same spectrum (see text for details).

Log-log presentation. When $\log[S_A(f)]$ vs. $\log f$ is plotted, the result allows a wide range of frequencies and spectral densities to be displayed. Also, any power law relationships between $S_A(f)$ and f appear as straight lines on this graph. As will be discussed in more detail in the next chapter, $S_A(f)$ is proportional to $f^{-5/3}$ in the inertial subrange portion of the spectrum, which will appear as a straight line with -5/3 slope on a log-log graph (see Fig. 8.9e). Unfortunately, the area under the curve is no longer proportional to the variance.

Log f $S_A(f)$ vs. log f. A plot of $\log[f \cdot S_A(f)]$ vs. $\log f$, has all of the desirable characteristics of the log-log presentation described above. In addition, the quantity $f \cdot S_A(f)$ has the same units as the variance of A, making scaling or normalization easier. Unfortunately, the area under the curve is also not proportional to variance (see Fig. 8.9f). Regardless of this problem, this presentation is the most used in the literature.

Table 8-1. Artificial data and spreadsheet calculations used to demonstrate various ways to present spectra.

This is assumed to be the spectrum of a time series of velocity measurements.

Variable	Value
Zi (m)	1000
U (m/s)	5
Dissip.(m2 s-3)	0.002
Size	21

Logarithm of Normalized Frequency	Logarithm of Normalized Spectrum	Normalized Frequency	Normalized Spectrum	f (1/s)	S (m2/s3)	f S (m2/s2)
-1.0	-1.3010	0.1000	0.0500	0.0005	158.7401	0.0794
-0.8	-1.2412	0.1580	0.0574	0.0008	115.3005	0.0911
-0.6	-1.1807	0.2510	0.0660	0.0013	83.4309	0.1047
-0.4	-1.1204	0.3980	0.0758	0.0020	60.4486	0.1203
-0.2	-1.0602	0.6310	0.0871	0.0032	43.8016	0.1382
-0.0	-1.0000	1.0000	0.1000	0.0050	31.7480	0.1587
0.2	-1.0000	1.5850	0.1000	0.0079	20.0303	0.1587
0.4	-1.0000	2.5120	0.1000	0.0126	12.6385	0.1587
0.6	-1.0827	3.9810	0.0827	0.0199	6.5914	0.1312
0.8	-1.2175	6.3100	0.0606	0.0316	3.0495	0.0962
1.0	-1.3521	10.0000	0.0445	0.0500	1.4112	0.0706
1.2	-1.4868	15.8490	0.0326	0.0792	0.6530	0.0517
1.4	-1.6215	25.1190	0.0239	0.1256	0.3022	0.0379
1.6	-1.7562	39.8110	0.0175	0.1991	0.1398	0.0278
1.8	-1.8909	63.0960	0.0129	0.3155	0.0647	0.0204
2.0	-2.0255	100.0000	0.0094	0.5000	0.0299	0.0150
2.2	-2.1602	158.4890	0.0069	0.7924	0.0139	0.0110
2.4	-2.2949	251.1890	0.0051	1.2559	0.0064	0.0080
2.6	-2.4296	398.1070	0.0037	1.9905	0.0030	0.0059
2.8	-2.5643	630.9570	0.0027	3.1548	0.0014	0.0043
3.0	-2.6990	1000.0000	0.0020	5.0000	0.0006	0.0032