PSD-per-unit-time converges to finite values at all frequencies except those where 
$h(t)$ has a discrete sine-wave (or cosine-wave) component of finite amplitude. At 
those frequencies, it becomes a delta-function, i.e., a sharp spike, whose width gets 
narrower and narrower, but whose area converges to be the mean square amplitude 
of the discrete sine or cosine component at that frequency.

We have by now stated all of the analytical formalism that we will need in this 
chapter with one exception: In computational work, especially with experimental 
data, we are almost never given a continuous function $h(t)$ to work with, but are 
given, rather, a list of measurements of $h(t_i)$ for a discrete set of $t_i$’s. The profound 
implications of this seemingly unimportant fact are the subject of the next section.

CITED REFERENCES AND FURTHER READING:
Champeney, D.C. 1973, Fourier Transforms and Their Physical Applications (New York: Academic 
Press).
Academic Press).

12.1 Fourier Transform of Discretely Sampled 
Data

In the most common situations, function $h(t)$ is sampled (i.e., its value is 
recorded) at evenly spaced intervals in time. Let $\Delta$ denote the time interval between 
consecutive samples, so that the sequence of sampled values is

$$h_n = h(n\Delta) \quad n = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \quad (12.1.1)$$

The reciprocal of the time interval $\Delta$ is called the sampling rate; if $\Delta$ is measured 
in seconds, for example, then the sampling rate is the number of samples recorded 
per second.

Sampling Theorem and Aliasing

For any sampling interval $\Delta$, there is also a special frequency $f_c$, called the Nyquist critical frequency, given by

$$f_c \equiv \frac{1}{2\Delta} \quad (12.1.2)$$

If a sine wave of the Nyquist critical frequency is sampled at its positive peak value, 
then the next sample will be at its negative trough value, the sample after that at 
the positive peak again, and so on. Expressed otherwise: Critical sampling of a 
sine wave is two sample points per cycle. One frequently chooses to measure time 
in units of the sampling interval $\Delta$. In this case the Nyquist critical frequency is 
just the constant $1/2$.

The Nyquist critical frequency is important for two related, but distinct, reasons. 
One is good news, and the other bad news. First the good news. It is the remarkable
fact known as the sampling theorem: If a continuous function \( h(t) \), sampled at an interval \( \Delta \), happens to be bandwidth limited to frequencies smaller in magnitude than \( f_c \), i.e., if \( H(f) = 0 \) for all \( |f| \geq f_c \), then the function \( h(t) \) is completely determined by its samples \( h_n \). In fact, \( h(t) \) is given explicitly by the formula

\[
h(t) = \Delta \sum_{n=-\infty}^{+\infty} h_n \frac{\sin[2\pi f_c(t - n\Delta)]}{\pi(t - n\Delta)} \tag{12.1.3}
\]

This is a remarkable theorem for many reasons, among them that it shows that the “information content” of a bandwidth limited function is, in some sense, infinitely smaller than that of a general continuous function. Fairly often, one is dealing with a signal that is known on physical grounds to be bandwidth limited (or at least approximately bandwidth limited). For example, the signal may have passed through an amplifier with a known, finite frequency response. In this case, the sampling theorem tells us that the entire information content of the signal can be recorded by sampling it at a rate \( \Delta^{-1} \) equal to twice the maximum frequency passed by the amplifier (cf. 12.1.2).

Now the bad news. The bad news concerns the effect of sampling a continuous function that is not bandwidth limited to less than the Nyquist critical frequency. In that case, it turns out that all of the power spectral density that lies outside of the frequency range \(-f_c < f < f_c\) is spuriously moved into that range. This phenomenon is called aliasing. Any frequency component outside of the frequency range \((-f_c, f_c)\) is aliased (falsely translated) into that range by the very act of discrete sampling. You can readily convince yourself that two waves \( \exp(2\pi if_1t) \) and \( \exp(2\pi if_2t) \) give the same samples at an interval \( \Delta \) if and only if \( f_1 \) and \( f_2 \) differ by a multiple of \( 1/\Delta \), which is just the width in frequency of the range \((-f_c, f_c)\). There is little that you can do to remove aliased power once you have discretely sampled a signal. The way to overcome aliasing is to (i) know the natural bandwidth limit of the signal — or else enforce a known limit by analog filtering of the continuous signal, and then (ii) sample at a rate sufficiently rapid to give at least two points per cycle of the highest frequency present. Figure 12.1.1 illustrates these considerations.

To put the best face on this, we can take the alternative point of view: If a continuous function has been competently sampled, then, when we come to estimate its Fourier transform from the discrete samples, we can assume (or rather we might as well assume) that its Fourier transform is equal to zero outside of the frequency range in between \(-f_c\) and \( f_c \). Then we look to the Fourier transform to tell whether the continuous function has been competently sampled (aliasing effects minimized). We do this by looking to see whether the Fourier transform is already approaching zero as the frequency approaches \( f_c \) from below, or \(-f_c\) from above. If, on the contrary, the transform is going towards some finite value, then chances are that components outside of the range have been folded back over onto the critical range.

**Discrete Fourier Transform**

We now estimate the Fourier transform of a function from a finite number of its sampled points. Suppose that we have \( N \) consecutive sampled values

\[
h_k \equiv h(t_k), \quad t_k \equiv k\Delta, \quad k = 0, 1, 2, \ldots, N - 1 \tag{12.1.4}
\]
Figure 12.1.1. The continuous function shown in (a) is nonzero only for a finite interval of time \( T \). It follows that its Fourier transform, whose modulus is shown schematically in (b), is not bandwidth limited but has finite amplitude for all frequencies. If the original function is sampled with a sampling interval \( \Delta \), as in (a), then the Fourier transform (c) is defined only between plus and minus the Nyquist critical frequency. Power outside that range is folded over or “aliquotted” into the range. The effect can be eliminated only by low-pass filtering the original function before sampling.

so that the sampling interval is \( \Delta \). To make things simpler, let us also suppose that \( N \) is even. If the function \( h(t) \) is nonzero only in a finite interval of time, then that whole interval of time is supposed to be contained in the range of the \( N \) points given. Alternatively, if the function \( h(t) \) goes on forever, then the sampled points are supposed to be at least “typical” of what \( h(t) \) looks like at all other times.

With \( N \) numbers of input, we will evidently be able to produce no more than \( N \) independent numbers of output. So, instead of trying to estimate the Fourier transform \( H(f) \) at all values of \( f \) in the range \(-f_c\) to \( f_c\), let us seek estimates only at the discrete values

\[
f_n = \frac{n}{N\Delta}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2}
\]  

(12.1.5)

The extreme values of \( n \) in (12.1.5) correspond exactly to the lower and upper limits of the Nyquist critical frequency range. If you are really on the ball, you will have noticed that there are \( N + 1 \), not \( N \), values of \( n \) in (12.1.5); it will turn out that the two extreme values of \( n \) are not independent (in fact they are equal), but all the others are. This reduces the count to \( N \).
The remaining step is to approximate the integral in (12.0.1) by a discrete sum:

\[ H(f_n) = \int_{-\infty}^{\infty} h(t)e^{2\pi ifn\Delta t} dt \approx \sum_{k=0}^{N-1} h_k e^{2\pi ifn\Delta} = \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i kn/N} \] (12.1.6)

Here equations (12.1.4) and (12.1.5) have been used in the final equality. The final summation in equation (12.1.6) is called the discrete Fourier transform of the \( N \) points \( h_k \). Let us denote it by \( H_n \),

\[ H_n = \sum_{k=0}^{N-1} h_k e^{2\pi i kn/N} \] (12.1.7)

The discrete Fourier transform maps \( N \) complex numbers (the \( h_k \)'s) into \( N \) complex numbers (the \( H_n \)'s). It does not depend on any dimensional parameter, such as the time scale \( \Delta \). The relation (12.1.6) between the discrete Fourier transform of a set of numbers and their continuous Fourier transform when they are viewed as samples of a continuous function sampled at an interval \( \Delta \) can be rewritten as

\[ H(f_n) \approx \Delta H_n \] (12.1.8)

where \( f_n \) is given by (12.1.5).

Up to now we have taken the view that the index \( n \) in (12.1.7) varies from \(-N/2\) to \( N/2 \) (cf. 12.1.5). You can easily see, however, that (12.1.7) is periodic in \( n \), with period \( N \). Therefore, \( H_{n-n} = H_{N-n} \). With this conversion in mind, one generally lets the \( n \) in \( H_n \) vary from 0 to \( N-1 \) (one complete period). Then \( n \) and \( k \) (in \( h_k \)) vary exactly over the same range, so the mapping of \( N \) numbers into \( N \) numbers is manifest. When this convention is followed, you must remember that zero frequency corresponds to \( n = 0 \), positive frequencies \( 0 < f < f_c \) correspond to values \( 1 \leq n \leq N/2 - 1 \), while negative frequencies \( -f_c < f < 0 \) correspond to \( N/2 + 1 \leq n \leq N-1 \). The value \( n = N/2 \) corresponds to both \( f = f_c \) and \( f = -f_c \).

The discrete Fourier transform has symmetry properties almost exactly the same as the continuous Fourier transform. For example, all the symmetries in the table following equation (12.0.3) hold if we read \( h_k \) for \( h(t) \), \( H_n \) for \( H(f) \), and \( H_{N-n} \) for \( H(-f) \). (Likewise, “even” and “odd” in time refer to whether the values \( h_k \) at \( k = \) 0 and \( |N - k| \) are identical or the negative of each other.)

The formula for the discrete inverse Fourier transform, which recovers the set of \( h_k \)'s exactly from the \( H_n \)'s is:

\[ h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i kn/N} \] (12.1.9)

Notice that the only differences between (12.1.9) and (12.1.7) are (i) changing the sign in the exponential, and (ii) dividing the answer by \( N \). This means that a routine for calculating discrete Fourier transforms can also, with slight modification, calculate the inverse transforms.
The discrete form of Parseval’s theorem is
\[ \sum_{k=0}^{N-1} |h_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |H_n|^2 \] (12.1.10)

There are also discrete analogs to the convolution and correlation theorems (equations 12.0.9 and 12.0.11), but we shall defer them to §13.1 and §13.2, respectively.

CITED REFERENCES AND FURTHER READING:

12.2 Fast Fourier Transform (FFT)

How much computation is involved in computing the discrete Fourier transform (12.1.7) of \( N \) points? For many years, until the mid-1960s, the standard answer was this: Define \( W \) as the complex number
\[ W \equiv e^{2\pi i/N} \] (12.2.1)

Then (12.1.7) can be written as
\[ H_n = \sum_{k=0}^{N-1} W^{nk} h_k \] (12.2.2)

In other words, the vector of \( h_k \)'s is multiplied by a matrix whose \((n,k)\)th element is the constant \( W \) to the power \( n \times k \). The matrix multiplication produces a vector result whose components are the \( H_n \)'s. This matrix multiplication evidently requires \( N^2 \) complex multiplications, plus a smaller number of operations to generate the required powers of \( W \). So, the discrete Fourier transform appears to be an \( O(N^2) \) process. These appearances are deceiving! The discrete Fourier transform can, in fact, be computed in \( O(N \log_2 N) \) operations with an algorithm called the fast Fourier transform, or FFT. The difference between \( N \log_2 N \) and \( N^2 \) is immense. With \( N = 10^6 \), for example, it is the difference between, roughly, 30 seconds of CPU time and 2 weeks of CPU time on a microsecond cycle time computer. The existence of an FFT algorithm became generally known only in the mid-1960s, from the work of J.W. Cooley and J.W. Tukey. Retrospectively, we now know (see [1]) that efficient methods for computing the DFT had been independently discovered, and in some cases implemented, by as many as a dozen individuals, starting with Gauss in 1805!

One “rediscovery” of the FFT, that of Danielson and Lanczos in 1942, provides one of the clearest derivations of the algorithm. Danielson and Lanczos showed that a discrete Fourier transform of length \( N \) can be rewritten as the sum of two discrete Fourier transforms, each of length \( N/2 \). One of the two is formed from the