10/2-20-92 JS ①

SANDIA REPORT

SAND91 — 1533 • UC — 706
Unlimited Release
Printed December 1991

PSD Computations Using Welch's Method

Otis M. Solomon, Jr.

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

SAND--91-1533
DE92 007419

SAND91-1533
Unlimited Release
Printed December 1991

PSD Computations Using Welch's Method

Otis M. Solomon, Jr.
Division 2722
Test Data Analysis

Abstract

This report describes Welch's method for computing PSDs. We first describe the bandpass filter method which uses filtering, squaring, and averaging operations to estimate a PSD. Second, we delineate the relationship of Welch's method to the bandpass filter method. Third, the frequency domain signal-to-noise ratio for a sine wave in white noise is derived. This derivation includes the computation of the noise floor due to quantization noise. The signal-to-noise ratio and noise floor depend on the FFT length and window. Fourth, the variance of Welch's PSD is discussed via chi-square random variables and degrees of freedom. This report contains many examples, figures and tables to illustrate the concepts.

ACKNOWLEDGMENTS

The author thanks Paul Kuenstler for his support of this work. Jerry Biedscheid, Jeanne Bando, and Mike Rogers reviewed early drafts of the report. Vesta Bateman, Jerry Cap, Dan Gregory and Paul Kuenstler reviewed later drafts of the report. Jonathan Rogers via Jerry Cap told the author about the paper by Gade and Herlufsen. Bob Peet helped the author obtain copies of the reports by Gade and Herlufsen. Sam Stearns read over the report. Toni Garcia and Julie Solomon proofread the report. Toni Garcia helped to get this report published. My facilitator Fred Gutierrez provided computers and software for analyzing the data and writing this report.

Contents

List of Figures

Section 1 Introduction

Widespread use of Welch's method for computing power spectral densities (PSDs) continues even though it is well over 20 years old. Welch's short 4–page paper [26], written in 1967, discusses his estimation method. His earlier and longer paper [25], written in 1961, is referenced in [26]. This earlier paper covers much of the same material but without any mention of the fast Fourier transform (FFT). The FFT was popularized by Cooley and Tukey in their 1965 paper [3]. Much of the widespread acceptance of the Welch method is because it uses the FFT, which makes it computationally efficient. Many programs and scientific software libraries contain a version of Welch's method: MATLAB, Matfor, *Numerical Recipes* [21], *Digital Spectral Analysis* [14], *Signal Processing Algorithms* [24], and *C Language Algorithms for Digital Signal Processing* [4]. This report describes the bandpass filter method and its relation to Welch's method. References [1], [2], [8], [9], [7], and [15] describe the bandpass filter method. Blackman and Tukey call the bandpass filter method the direct analog computation method. Gardner calls it the wave analysis method. We illustrate the relationship through numerous examples, figures and tables. To assess the information in a PSD, one should know the maximum and minimum values that the algorithm can produce. Our discussion on signal-to-noise ratios and noise floors is meant to aid the analyst in this area. Reference [11], IEEE Std 1057, contains some of the formulas for signal-to-noise ratios. However, its derivations are very short and hard to understand. A new draft of IEEE Std 1057 should be published in 1992. Schoukens and Renneboog in [23] discuss how noise, both white and colored, influences discrete Fourier transform (DFT) coefficients. The paper does not describe the effect of windowing or how noise influences PSDs. Proper interpretation of PSDs requires an understanding of which wiggles are significant. Welch's variance expression helps us in this area. Blackman and Tukey [2, Section 9] provides motivation for the use of chi-square distributions and equivalent degrees of freedom for describing the stability of a PSD. Report [17] compares an approximate chi-square distribution of a PSD estimate to its exact distribution. It concludes that the approximate chi-square distribution provides a valid probabilistic description of Welch's PSD estimate.

Section 2 General Description of PSD Estimation

To perform spectral analysis on a computer, one begins with a sequence of data values or samples:

$$x[0], x[1], \ldots, x[N-1]$$

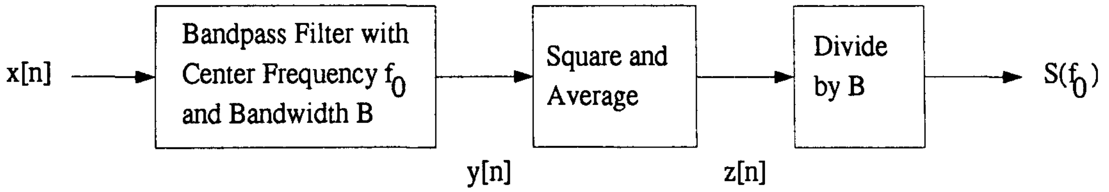
The independent variable of the data sequence ranges from 0 to $N-1$. The data values $x[n]$ are indexed by their sample number n. This is the sample value's position relative to the start of the sequence. The data samples are acquired at a constant rate. The time between two successive data samples $x[n]$ and $x[n+1]$ is T seconds. The sample rate is $1/T$ samples per second. The length of the data sequence in seconds is $T_{seq} = N * T$. The time of acquisition of a data value is related to its sample number by $t = t_0 + nT$ where t_0 is time when the first data sample was acquired.

The goal of spectral analysis is to decompose the data into a sum of weighted sinusoids. This decomposition allows one to assess the frequency content of the phenomenon under study. The phenomenon under study may be concentrated in some narrow frequency band. On the other hand, it might be spread across a broad range of frequencies. Spectral analysis is divided into two major areas. One can compute a Fourier transform or a power spectral density (PSD). When the data contains no random effects or noise, it is called deterministic. In this case, one computes a Fourier transform. One computes a PSD when random effects obscure the desired underlying phenomenon. To see the desired phenomenon, some sort of averaging or smoothing is employed.

The desired underlying phenomenon can itself be random. The desired underlying phenomenon is normally not white noise. For example, suppose that we want to test a new electronic lowpass filter. We excite the filter with random white noise. The output of the filter looks like noise. However, a PSD analysis will show (assuming that the filter works properly) that low frequencies are unattenuated and high frequencies are attenuated. The desired underlying phenomenon here is the electronic circuit. The purpose of the averaging and smoothing is to expose the underlying persistent behavior of the lowpass filter. Averaging and smoothing must be used in an intelligent manner to discover the true nature of the underlying phenomenon.

Figure 1 shows how to estimate a PSD at a single fixed frequency, f_0. This method is called direct analog computation by Blackman and Tukey [2] and wave analysis by Gardner [9]. Conceptually, the design of a PSD estimation procedure is equivalent to choosing: (1) the type of bandpass filter, (2) the bandwidth of the

Figure 1 Wave Analysis Method of PSD Estimation



bandpass filter, (3) a method of squaring and averaging the output of the bandpass filter, and (4) how long to average the squared output of the bandpass filter. The proper choices are dependent on the phenomenon under study. Appropriate choices for one phenomenon may be bad for a different phenomenon.

The purpose of computing a PSD is to see how the frequency content of $x[n]$ varies with frequency. To do this, one must choose many different center frequencies for the bandpass filter in Figure 1. One then plots $S(f)$ versus f, the different center frequencies of the bandpass filter.

Let us now focus our attention on estimating a single PSD value at a single fixed frequency f_0. We use the subscript 0 to remind us that f_0 is a single fixed frequency. The bandpass filter is considered to be ideal: the gain is 1 in its passband and 0 elsewhere. The bandpass filter limits the data sequence to a band of frequencies from $f_0 - B/2$ to $f_0 + B/2$ Hz. The bandpass filter removes the DC component of $x[n]$. The output of the bandpass filter $y[n]$ oscillates about zero. The average value of $y[n]$ is approximately zero. With reference to Figure 1, $z[n]$ is the average value of the square of the bandpass filtered output $y[n]$. $z[n]$ is called the mean square value of $y[n]$. The sequence $z[n]$ is not always zero because it is squared before it is averaged. It depends on the frequency content of the input sequence $x[n]$. In fact, it is zero only when the bandpass filtered sequence $y[n]$ is identically zero. For example, if $x[n]$ is a sine wave whose frequency is less than $f_0 - B/2$, then the average value of the square of the bandpass filtered output, $z[n]$, is zero. If $x[n]$ is a sine wave with frequency f_0, then $z[n]$ is a nonzero sequence. The value of $z[n]$ is proportional to the strength of the sine waves in $x[n]$ that lie between $f_0 - B/2$ and $f_0 + B/2$ Hz.

Let U denote the unit of measurement for the sequence $x[n]$. U could be volts, amperes, or pounds per square inch. What are the units for a PSD of $x[n]$? Look at Figure 1. The units for $y[n]$ are U because the bandpass filter does not change the units of $x[n]$. The units for $z[n]$ are U-squared because of the squaring operation. The averaging operation does not alter the units. Dividing

$z[n]$ by the bandwidth in Hz of the filter creates the units of U-squared per Hz for the PSD, $S(f)$.

The choice of the bandpass filter and the averager are interrelated to each other and the length of the data sequence. Some constraints on the choice of the bandpass filter and the averager are:

$$T_{seq} \geq Tavg \gg \frac{1}{B}$$

where

$T_{seq} =$ length of data sequence in seconds

$T_{avg} =$ averaging time in seconds

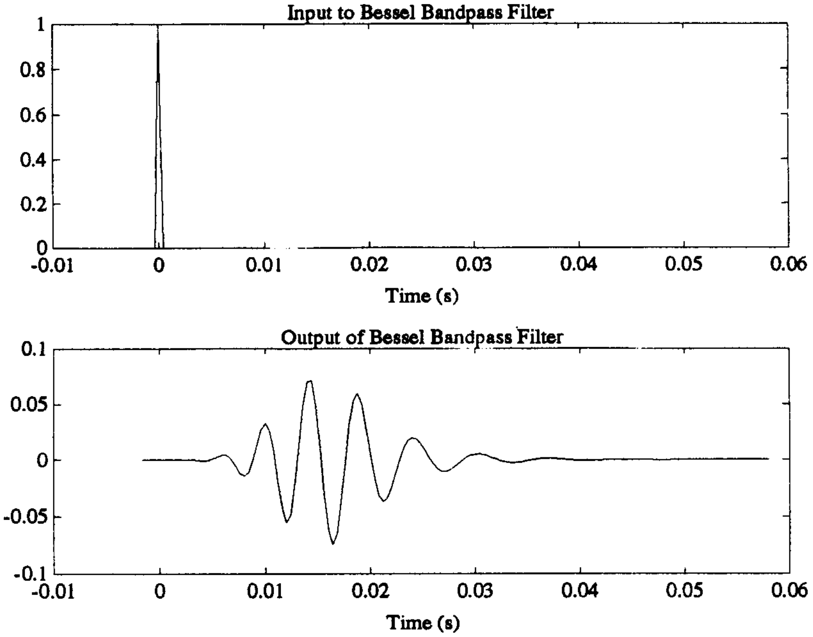
$B =$ bandwidth of bandpass filter in hertz

The constraint that $T_{seq} \geq T_{avg}$ insures that the length of the data sequence must be greater than or equal to the averaging time. Data must be gathered before it can be averaged. The more subtle constraint that $T_{avg} \gg 1/B$ is discussed below.

An important observation is that to reduce random variations, one must average independent data. Consider the problem of assessing an individual's knowledge of some subject by asking yes-and-no questions. If the subject has any breadth at all, one cannot hope to discover what this person knows by asking a single question. Asking the same question over and over, and computing the average number of correct responses is silly. It is much better to ask a variety of different questions. Then the average number of correct responses stands a chance of indicating the individual's knowledge. Asking the same question over and over does not provide any new information after the first question. Averaging the same old information over and over will not smooth out random noise. The answer to each new question adds more information about an individual's knowledge. Averaging independent data values reduces random variations.

When a very short pulse is bandpass filtered, the result is a sine wave whose amplitude varies with time. This result is called the impulse response of the filter. Figure 2 shows the input and output of a bandpass filter. The filter has a center frequency of $f = 210$ Hz and a 3-dB bandwidth of $B = 20$ Hz. The length of the amplitude-modulated sine wave is approximately $1/B = 0.05$ seconds. The averager begins to smooth out random effects when $T_{avg} = 1/B$. For effective smoothing, the averaging time must be much longer than the length of this signal, i.e., $T_{avg} \gg 1/B$. All of the nonzero data values in the lower plot are derived from the single nonzero value in the upper plot. Blackman and Tukey in [2, Section B.10] derive the above constraints on averaging time and analog filtering bandwidth. See [8, Section 11.4] for a short description of the wave analysis method. Chapter 4 of [9] is a more detailed discussion of the wave analysis method and its variations.

Figure 2 Bandpass Filter Response



At approximately $1/B$ seconds from the first nonzero data value, the bandpass filter finishes responding to the first nonzero data value. At approximately $1/B+T$ seconds from the first nonzero data value, the bandpass filter finishes responding to the second data value. In general, at approximately $1/B+(n-1)T$ seconds from the first nonzero data value, the bandpass filter finishes responding to the nth data value. Two adjacent values of the bandpass filter output, $y[n]$ and $y[n+1]$, contain almost the same information. They are the response to almost the same set of input values. Values of $y^2[n]$ closer together than $1/B$ seconds are correlated to one another. Values of $y^2[n]$ separated by more than $1/B$ seconds are independent of one another. To smooth out random effects requires the averaging of independent values of $y^2[n]$. To obtain a good PSD requires the averaging of many independent values of $y^2[n]$, i.e., $T_{avg} \gg 1/B$.

Section 3 Welch's Method

The tasks in Figure 1 can be accomplished in many ways. I will describe Welch's method and then explain how it relates to Figure 1. The steps are:

1. Partition the data sequence:

$$x[0], x[1], \ldots, x[N-1]$$

into K segments or batches:

Segment 1: $x[0], x[1], \ldots, x[M-1]$

Segment 2: $x[S], x[S+1], \ldots, x[M+S-1]$

$\vdots$

Segment K: $x[N-M], x[N-M+1], \ldots, x[N-1]$

where

$M =$ Number of points in each segment or batch size

$S =$ Number of points to shift between segments

$K =$ Number of segments or batches

2. For each segment ($k = 1$ to K), compute a windowed discrete Fourier transform (DFT) at some frequency $\nu = i/M$ with $-(M/2-1) \leq i \leq M/2$:

$$X_k(\nu) = \sum_m x[m]w[m] \exp(-j2\pi\nu m)$$

where

$m = (k-1)S, \ldots, M+(k-1)S-1$

$w[m] =$ the window function

3. For each segment ($k = 1$ to K), form the modified periodogram value, $P_k(f)$, from the discrete Fourier transform:

$$P_k(\nu) = \frac{1}{W}|X_k(\nu)|^2$$

where

$$W = \sum_{m=0}^{M} w^2[m]$$

4. Average the periodogram values to obtain Welch's estimate of the PSD:

$$S_x(\nu) = \frac{1}{K}\sum_{k=1}^{K} P_k(\nu)$$

Welch's method is also called the weighted overlapped segment averaging (WOSA) method and periodogram averaging method. The parameter M is the length of each segment. Some people refer to segments as batches. These people call M the batch size. Note that M is the length of the DFT. The parameter S is the number of points to shift between segments. It is the number of new points in each segment or batch. The number of points in common to two adjacent segments is $M - S$. Two adjacent segments are said to overlap by $M - S$ points or $100[(M-S)/M]\%$. When $S = M$, the segments do not overlap. When $S = 0.5M$, the segments contain 50% overlap. The M-point sequence $w[m]$ is the window function. Some common windows are the rectangular, Hann or Hanning, Hamming, Blackman, Blackman-Harris, and Kaiser-Bessel. References [10] and [18] contain good discussions of windows. The parameter K is the number of segments or batches. It is the number of periodograms that are averaged together to form the PSD estimate $S_x(f)$.

There are four ways to designate the frequencies of a DFT [24, Table 3.1, page 24]. The units of the four methods are hertz times seconds (Hz-s), radians (rad), hertz (Hz) and radians per second (rad/s). In step 2 above, the units for ν are Hz-s. Adjacent values of ν are separated by $1/M$ Hz-s. The frequency variable ν ranges from $-0.5 + 1/M$ to 0.5 Hz-s. PSDs are often plotted versus frequency in Hz. To convert the units in step 2 above to Hz, divide each ν by the sample interval T. The resulting frequency variable f ranges from $-1/(2T) + 1/(MT)$ to $1/(2T)$ Hz. The units of Hz-s and rad are common. PSD procedures seldom specify the sample interval T as a processing parameter.

The value of i in $\nu = i/M$ for Step 2 of Welch's method is not restricted to integers. However, it normally is an integer because DFT values are usually computed with a fast Fourier transform algorithm. A fast Fourier transform (FFT) is a fast algorithm for computing the DFT in Step 2 of Welch's method. See [24, Chapter 3] for a discussion of DFT and FFT algorithms. For PSD computations, the input to the FFT routine is normally M real-valued data points. The most common outputs are the M complex values. The M complex values are the DFT values at the frequencies $\nu = i/M$ Hz-s for the integers $i = 0$ to $M - 1$. The Nyquist frequency component occurs at $\nu = (M/2)/M = 0.5$. The frequencies $\nu = (M/2 + 1)/M = 0.5 + 1/M$ to $(M - 1)/M = 1.0 - 1/M$ are the negative frequencies. The most negative frequency is $\nu = (M/2 + 1)/M = 0.5 + 1/M$. When the input sequence to the FFT routine is real-valued, the negative frequencies are redundant and are often deleted. Exactly how the complex values are stored in a data array depends on the FFT software. Computer programmers are free to store the values in whatever order is reasonable given the constraints of their programming language, operating system, and computer. Read the description of your software to avoid mistakes.

The filtering method of Figure 1 and Welch's windowed overlapped segmented averaging method are equivalent. However, there are major differences in the implementation. Principally, the method of Figure 1 performs only time-domain operations whereas Welch's method performs mostly frequency-domain operations. An explanation of exactly why these two methods are equivalent is too lengthy for this report. My goal is to explain which parameters in Welch's method control the parameters of Figure 1. Recall that the main design parameters of Figure 1 are the bandpass filter bandwidth and the averaging time.

Steps 1 and 2 of Welch's method correspond to the leftmost block of Figure 1 which is the bandpass filter. The window $w[n] = 1$ for all n is called a rectangular window. Most windows other than the rectangular window are similar to the Gaussian bell shape of probability theory. The bandwidth of a window or filter is approximately equal to the reciprocal of its length. This is a very rough approximation which is accurate only to a factor of 2 to 4. How one defines the length of a window that decays smoothly to zero is not obvious. For the rectangular window, the bandwidth of the bandpass filter is approximately $1/M$ Hz-s. The window $w[n]$ is used to change the characteristics of the bandpass filter. Window choice is normally based on sidelobe behavior. Two definitions of bandwidth in wide use are the equivalent noise bandwidth (ENBW) and the 3 dB bandwidth. The equivalent noise bandwidth is always bigger than the 3 dB bandwidth. The ratio of these bandwidths is normally in the range of 1.032 to 1.229 [10, page 82]. Table II in [10] lists the ENBW and 3 dB bandwidths for many windows. We describe the relationship of the equivalent noise bandwidth to the amplitude of the windowed FFT and the PSD in a later section.

The parameter S in Step 1 controls how much the segments overlap. When $S = M$, adjacent segments do not overlap. When $S = 1$, adjacent segments differ at one value. In this case, one point was deleted and one point was added to form the next segment from the previous segment. When $S = 1$, the discrete Fourier transform values $X_k(\nu)$ and $X_{k+1}(\nu)$ are highly correlated with one another. Ideally S should be the smallest value such that $X_k(\nu)$ and $X_{k+1}(\nu)$ are nearly uncorrelated. This value of S would save arithmetic operations but provide the maximum amount of smoothing. The shift between segments S is usually in the range $0.4M \leq S \leq M$.

Steps 3 and 4 correspond to the rightmost 2 blocks of Figure 1. The squaring and averaging are performed in the frequency domain in Steps 3 and 4. Step 3 forms the periodogram or sample spectrum. The units for the $P_k(\nu)$ are the same as those for $S_x(\nu)$. The $P_k(\nu)$ are not good estimates of PSDs. They contain too much statistical oscillation. Step 4 averages the periodograms $P_k(\nu)$ to form a stable PSD estimate that does not oscillate wildly. When S is chosen properly, the K periodogram values, $P_1(\nu), P_2(\nu), \ldots, P_K(\nu)$, are approximately independent of one another.

Welch's method normalizes a PSD to satisfy Parseval's relation:

$$\begin{aligned}
&\mathrm{var}(x[n]) = \frac{1}{M}\sum_{\nu} S_x(\nu) \\
&\text{where} \\
&\nu = -0.5 + \frac{1}{M}, -0.5 + \frac{2}{M}, \ldots, 0.5 \\
&S_x(\nu) = \text{ Welch's PSD of } x[n] \text{ at } \nu \text{ Hz–s}
\end{aligned}$$

Many engineers learn that the integral of a continuous PSD equals the variance of the continuous signal. The above sum is an approximation to an integral. For the above sum to approximate the integral of a continuous PSD, the factor $1/M$ must equal the bandwidth. What are the units for f_m and $S_x(f_m)$? The units for ν are Hz-s as mentioned above. Engineers normally measure frequency in Hz or rad/s. The sequence ν equals a frequency sequence in Hz multiplied by the sample interval T. For this reason, it is sometimes called a normalized frequency scale [24, page 46]. Let U be the units for x_n. The units for $S_x(\nu)$ are U^2/Hz-s rather than U^2/Hz. PSDs computed via Welch's method are often plotted versus frequency f in Hz without modifying the PSD values. This practice can lead to confusion. The units for f are Hz, while the units for $S_x(\nu)$ are U^2/Hz-s. One seldom sees plots of PSDs with the ordinate labelled as U^2/Hz-s. Normally the ordinate is labelled as U^2/Hz even though it is really U^2/Hz-s. To convert the units for $S_x(\nu)$ to U^2/Hz, multiply $S_x(\nu)$ by the sample interval T, and divide ν

by T. Parseval's relation in the new units U^2/Hz and Hz is

$$\mathrm{var}(x[n]) = \frac{1}{MT}\sum_f S_x(f)$$

where

$$f = -\frac{1}{2T} + \frac{1}{MT}, -\frac{1}{2T} + \frac{2}{MT}, \ldots, \frac{1}{2T}$$

$$S_x(f) = T \times \text{ Welch's PSD } S_x(\nu)$$

We use $S_x(\nu)$ to denote Welch's PSD with unit of U^2/Hz-s, and $S_x(f)$ to denote a PSD with units of U^2/Hz. The power in a frequency range $[\nu_1, \nu_2]$ Hz-s is

$$\frac{1}{M}\sum_{\nu=\nu_1}^{\nu_2} S_x(\nu)$$

where

frequency range $= \nu_1$ to ν_2 Hz–s

and the power in a frequency range $[f_1, f_2]$ Hz is

$$\frac{1}{MT}\sum_{f=f_1}^{f_2} S_x(f)$$

where

frequency range $= f_1$ to f_2 Hz

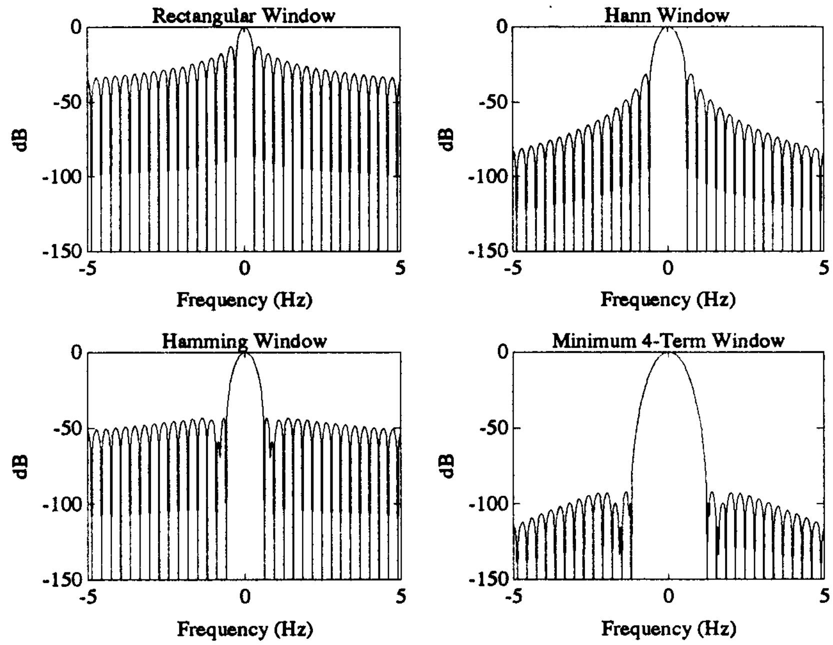
$$S_x(f) = T \times \text{ Welch's PSD } S_x(\nu)$$

PSDs of real-valued data are symmetric about $f = 0$. Analysts usually only view the positive frequencies. PSDs with both positive and negative frequencies are called two-sided PSDs. PSDs with only positive frequencies are called one-sided PSDs. A two-sided PSD can be converted to a one-sided PSD and vice versa. If you sum only the positive frequencies of a two-sided PSD, you will not get the total power. You get about half the total power. How is a two-sided PSD converted to a one-sided PSD? Multiply all nonnegative frequencies except 0 Hz-s and 0.5 Hz-s (0 Hz and $1/(2T)$ Hz) by 2. Delete all negative frequencies. A one-sided PSD contains $M/2 + 1$ values. This conversion does not change the bandwidth. The bandwidth of each measurement is about $1/M$ Hz-s or $1/(MT)$ Hz.

Window Examples

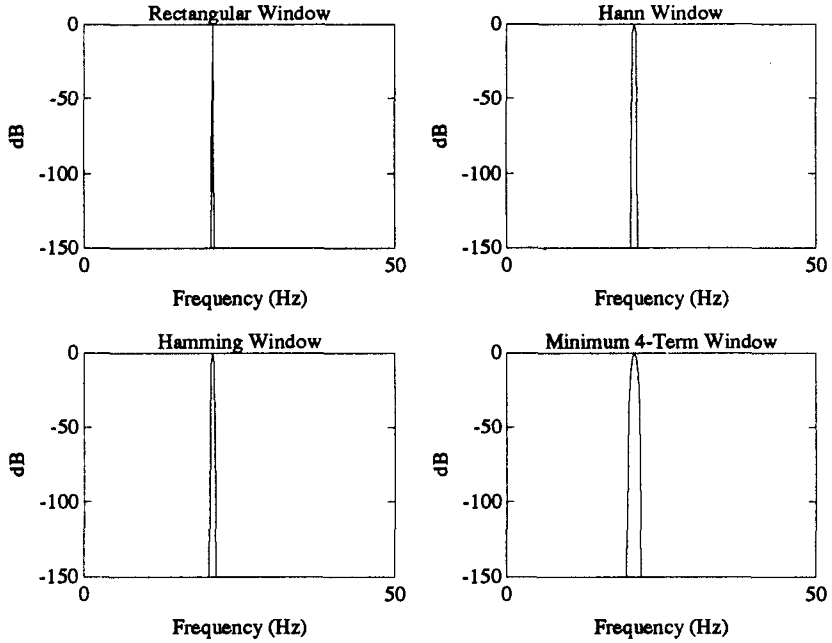
Figure 3 shows some windows in the frequency domain. The center lobes are centered on 0 Hz. For nonzero center frequencies, the window shapes in Figure 3 are translated to the new nonzero center frequency. The center lobe of the rectangular window is smaller than those of the other windows. However, the sidelobes of the rectangular window are higher than those of the other windows. Most people view the width of the main lobe as the resolution of the discrete Fourier transform. The sidelobes allow a sine wave with a frequency outside of the main lobe to contaminate the measurement of frequencies within the main lobe. Higher sidelobes increase the contamination. Windows can be designed with small sidelobes. Such windows will have a main lobe wider than the main lobe of the rectangular window. One cannot simultaneously make the main lobe narrow and the sidelobes small. References [10] and [18] discuss optimization strategies.

Figure 3 Bandpass Filter Shapes and Windows



The sidelobes form nice patterns which resemble a picket fence. Recall that T is the sample interval and M is the DFT length. Away from the center lobe, the deep nulls are spaced $1/(MT)$ Hz apart. Sine waves at multiples of $1/(MT)$ Hz cannot interfere with other frequencies. Multiples of $1/(MT)$ Hz are the basic or basis frequencies of the discrete Fourier transform. For a DFT of length 8192 points with a sample rate of 2500 samples per second, the frequency $67/(MT) = 20.45$ Hz will not interfere with other frequencies for a rectangular window. For windows other than the rectangular window, it interferes only when it and the other frequency both lie within the main lobe. It will never interfere with a frequency located far away from it. Figure 4 shows windowed DFTs of this sine wave. The only difference in the DFTs is the width of the spike at 20.45 Hz.

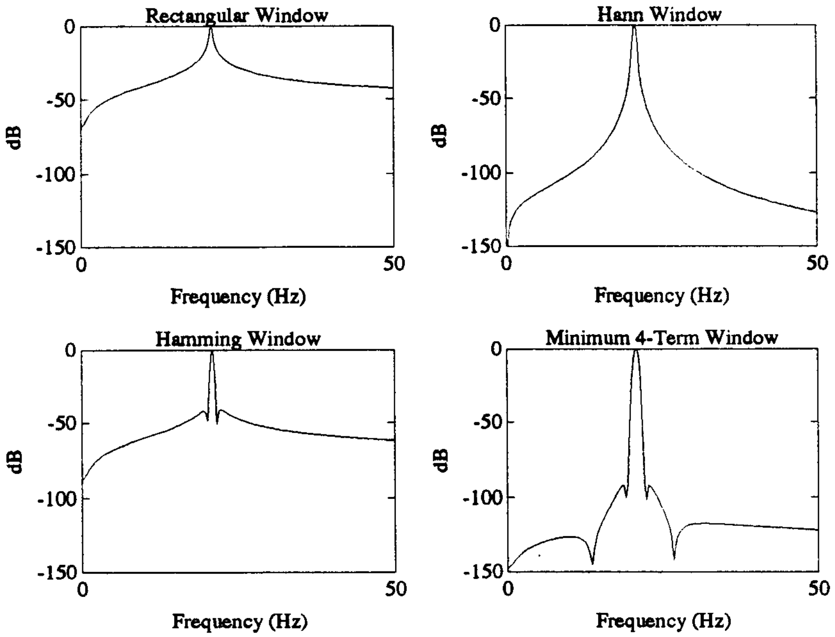
Figure 4 DFTs of a Basis Sine Wave



The frequency $67.5/(MT) = 20.60$ Hz interferes with all other frequencies. It lies halfway between two digital basis frequencies $67/(MT) = 20.45$ and $68/(MT) = 20.75$. Figure 5 shows windowed DFTs of this sine wave. All of the DFTs are nonzero away from 20.60 Hz. The rectangular window is particularly bad. The contamination for DFT with the minimum 4-term Blackman-Harris window is down 90 dB away from the main lobe. The main purpose of windowing is to alleviate this type of interference. The DFT of real data will almost certainly contain components at frequencies other than multiples of $1/(MT)$ Hz.

The length of the segments M and the window type are used to control the characteristics of the bandpass filter. M is the dominant parameter. In fact,

Figure 5 DFTs of a Non-Basis Sine Wave



for large amounts of data, a rectangular window with the proper segment length is adequate for any spectral analysis job. Since most applications have limited amounts of data, window choice is important.

Figure 6 shows the magnitude of four discrete Fourier transforms with different windows. The time-domain signal is the sum of a large and a small sine wave. The amplitudes of the sine waves are 128 and 0.128. The little sine wave is -60 dB smaller than the big sine wave. The frequencies of the big and little sine waves are $67.5/(MT) = 20.60$ Hz and $77.5/(MT) = 23.65$ Hz, respectively. The smaller sine wave is clearly observable with both the Hann and minimum 4–term windows. The rectangular and Hamming windows cannot see the little sine wave because it is obscured by the response of the sidelobes to the big sine wave.

How much of the response at the bandpass filter output is caused by the little sine wave and how much is caused by the big sine wave? The answer depends both on the gain of the filter at the frequencies of the sine waves and on the amplitudes of the sine waves. None of the frequencies in Step 2 of Welch's method coincide with $77.5/(MT) = 23.65$ Hz. The frequency of the little sine wave lies between 2 basis frequencies. The closest basis frequencies are: $77/(MT) = 23.50$ and $78/(MT) = 23.80$ Hz.

Figure 6 How Windows Influence Discrete Fourier Transforms

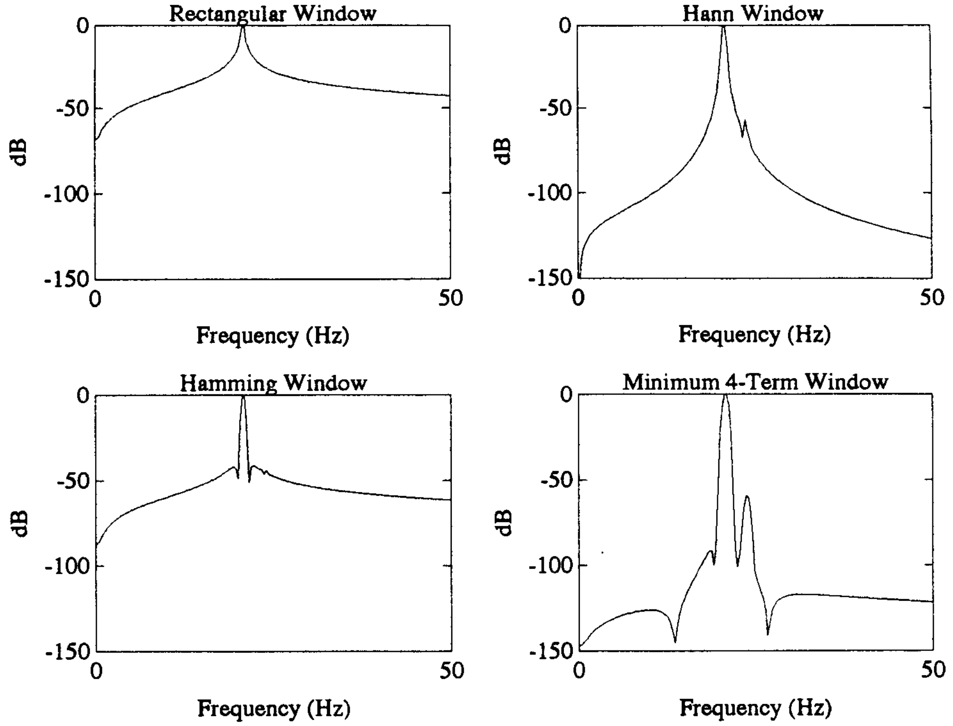


Table 1 Output of Bandpass Filter at 23.80 Hz

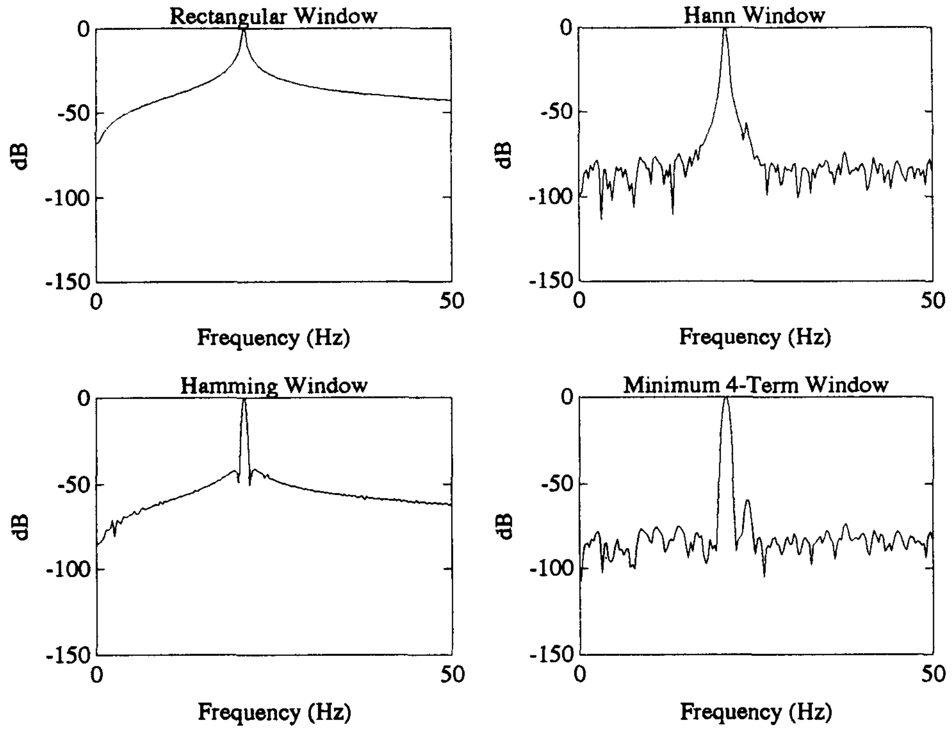
Window	**Gain at 20.60 Hz**	**Gain at 23.80 Hz**	**Amplitude of Big Sine Wave**	**Amplitude of Little Sine Wave**
Rectangular	-30 dB	0 dB	4.048	≈0.128
Hann	-71 dB	0 dB	0.036	≈0.128
Hamming	-48 dB	0 dB	0.510	≈0.128
Minumum 4-Term	-99 dB	0 dB	0.001	≈0.128

Center the bandpass filter for the rectangular window at the basis frequency above the frequency of the little sine wave: 23.80 Hz. The gain of the bandpass filter at 23.80 Hz is 0 dB or 1. The amplitude of the little sine wave at the output of the bandpass filter is slightly less than 0.128. The center frequency of the bandpass filter, 23.80 Hz, is slightly different from the frequency of the little sine wave, 23.65 Hz. The range of attenuation for the little sine wave is 0.75 to 4.00 dB. See the discussion of scalloping loss in [10] for more details. We ignore this attenuation of the little sine wave. What is the gain of this bandpass filter at the frequency of the big sine wave? Look at the bandpass filter for a rectangular window in Figure 3. The gain of the sidelobe at $23.80-20.60 = 3.20$ Hz from the center frequency is about -30 dB less than that of the main lobe. So, the gain of the bandpass filter at 20.60 Hz is about -30 dB or 0.032. The amplitude of the big sine wave at the output of the bandpass filter is approximately 4. Even though the filter attenuates the big sine wave 30 dB, it contributes more to the output than the little sine wave. The little sine wave is masked by the response of the sidelobes to the big sine wave. Table 1 lists the amplitudes of the big and little sine waves at the output of bandpass filters centered at 23.80 Hz. Each window changes the gain of the bandpass filter at 20.60 Hz. The gains at the frequency of the little sine wave also vary, but much less than at 20.60 Hz. Similar computations can be made for a bandpass filter centered at 23.50 Hz, which is the basis frequency below the frequency of the little sine wave.

Figure 7 shows the magnitude of four discrete Fourier transforms with different windows. The time-domain signal is the sum of a large, a small sine wave and a white noise sequence. The amplitudes and frequencies of the sine waves are as in the previous example. The amplitudes of the sine waves are 128 and 0.128. The frequencies of the big and little sine waves are $67.5/(MT) = 20.60$ Hz and $77.5/(MT) = 23.65$ Hz, respectively. The white noise added to the sine waves simulates quantization noise [4, Section 1.3]. The noise was uniformly distributed on ± 0.5. The smaller sine wave is observable with both the Hann and minimum 4–term Blackman-Harris windows. The rectangular and Hamming windows cannot see the little sine wave. The little sine wave is obscured by the response of the sidelobes to the big sine wave.

All measurement systems introduce noise. In our example, the uniformly distributed noise added to the sine waves represents system or measurement noise. It is not what we are interested in measuring. We want to know the amplitudes and frequencies of the sine wave. The random sequence models the unwanted, undesirable quantization noise. Our job would be easier if the noise would just go away. The average value of the DFT of the system or measurement noise is frequently called the noise floor. The DFTs for the rectangular and Hamming windows in Figure 7 are smooth and regular. Their shapes look very similar to

Figure 7 How Windows and White Noise Influence DFT



those in Figure 5. The DFT of a random sequence is not smooth. How can this be? The sidelobes of the rectangular and Hamming windows are larger than the noise floor. The smoothness in the rectangular and Hamming DFTs in Figure 7 is their response through their sidelobes to the big sine wave. The smoothness is not their response to the random sequence.

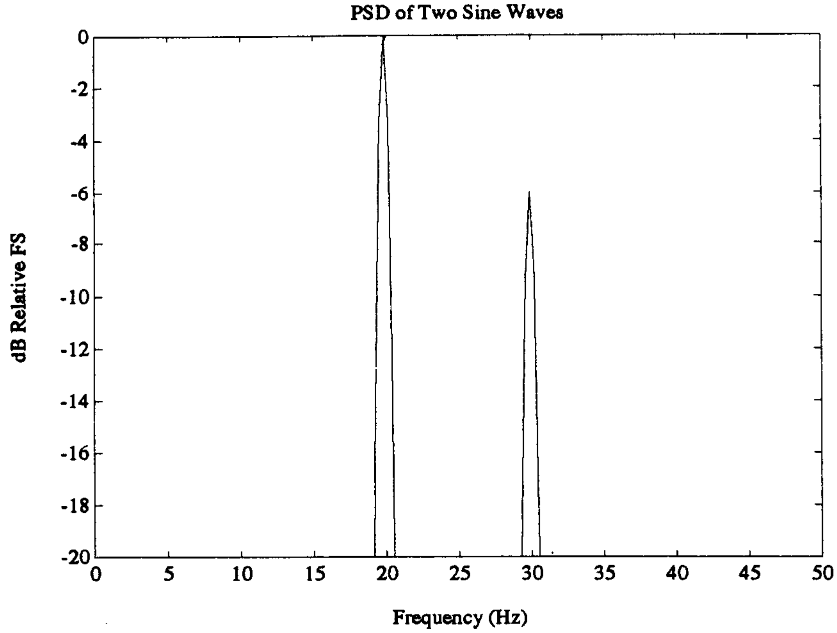
Section 4 Some Examples

PSDs computed on digital computers are usually normalized so that

$$\mathrm{var}(x[n]) = \frac{1}{M}\sum_{m=0}^{M} S_x(f_m)$$

Due to the many different gain ranges of our data, we elected to normalize our PSDs to a full scale sine wave. On our plots of PSDs, 0 dB corresponds to a full

Figure 8 PSD of Two Sine Waves



scale sine wave and -6 dB corresponds to a half scale sine wave. To convert a peak value on one of our PSDs to an amplitude, use the formula:

$$\text{Amplitude in \% of full scale} = 10^{(S_x(f)/20)} \times 100$$

Note that this formula assumes that the peak corresponds to a sine wave. Figure 8 shows the PSD of a sum of two sine waves. The amplitudes are full and half scale.

Figure 9 shows a very noisy sine wave. The sequence is the sum of a sine wave and a random sequence. The amplitude and frequency of the sine wave are 1.0 and 20.5 Hz. The random sequence has a Gaussian distribution with a mean of zero and a standard deviation of 20.0. The sample rate is 2500 samples per second. The sequence is 26.0 seconds long. The peak-to-peak amplitude of the sine wave is 2. The peak-to-peak amplitude of the noise is about 160. The rms value of the sine wave is 0.707. The rms value of the noise is 20. The signal-to-noise ratio is -29 dB. The sine wave is completely overwhelmed by the noise.

Figure 10 shows the PSDs of the noisy sine wave in Figure 9. The upper plot shows a PSD from the first 6.5 seconds of data and the lower plot shows a PSD of the entire 26.0 seconds of data. The sine wave is unobservable in the PSD of

Figure 9 Very Noisy Sine Wave

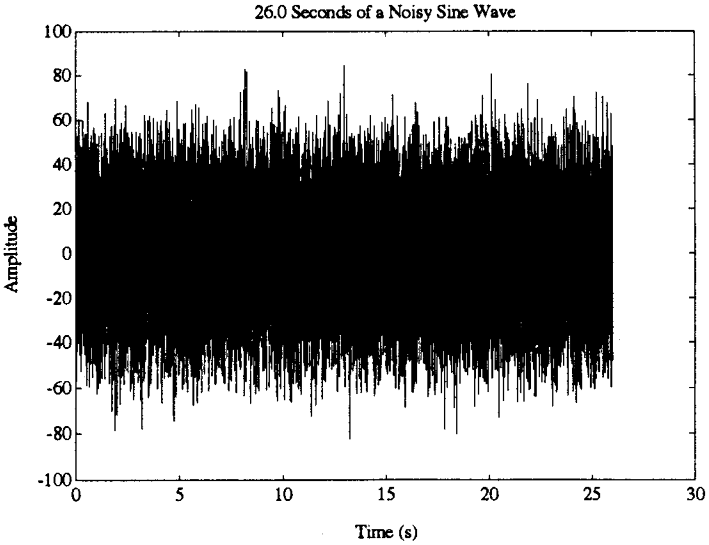
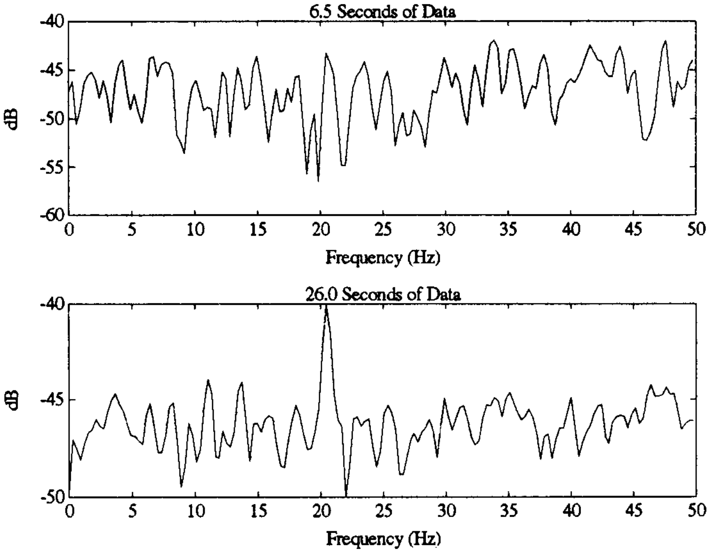
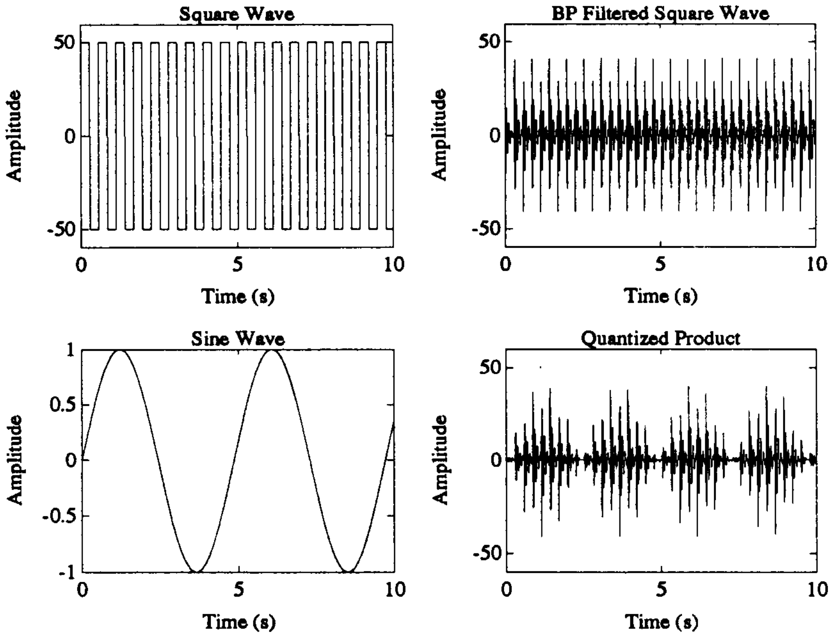


Figure 10 PSDs of a Sine Wave in Noise



the short sequence. The lower PSD is computed from 4 times as many segments or batches as the upper PSD. The K in Step 1 of Welch's method is 4 times larger for the lower PSD than it is for upper PSD. The lower PSD is the average of 4 times more periodograms than the upper PSD. This extra averaging reveals the sine wave at about 20 Hz in the lower PSD. This demonstrates how averaging smooths out the random effects of noise. A second observation is that sine waves are easy to find in broadband noise. Broadband noise spreads its energy across all frequencies. A sine wave concentrates all its energy at one frequency.

Figure 11 Product of a BP Filtered Square Wave and a Sine Wave



Let us now consider a more complex signal to show how a few simple operations can create a signal with a complicated spectrum. The upper plots of Figure 11 show the input and output of a filter. The input is a square wave with a period of 0.56 seconds and an amplitude of 50. The filter is a cascade of lowpass and highpass filter. The lowpass filter is a Butterworth with 4 poles and a cutoff frequency of 50 Hz. The highpass filter is a Butterworth with 8 poles and cutoff frequency of 11 Hz. The lower left plot of Figure 11 shows a sine wave with a frequency of 0.21 Hz and an amplitude of 1. The lower right plot of Figure 11 shows the product of the highpass filtered square wave and the sine wave. The product was rounded to the nearest integer to simulate quantization effects.

The top plot in Figure 12 shows a PSD of the product in the upper right of Figure 11. The PSD values near 0 Hz are about 150 dB below full scale. The PSD values near 120 Hz are also about 150 dB below full scale. The spacing between

Figure 12 PSD of the BP Filtered Square Wave

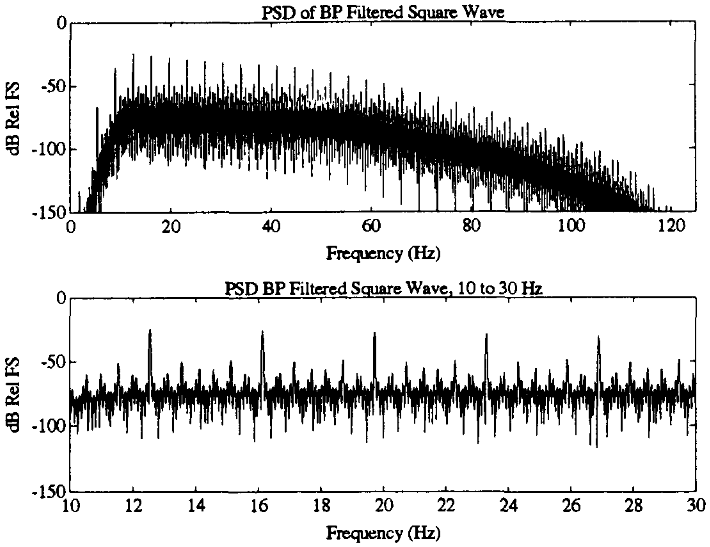
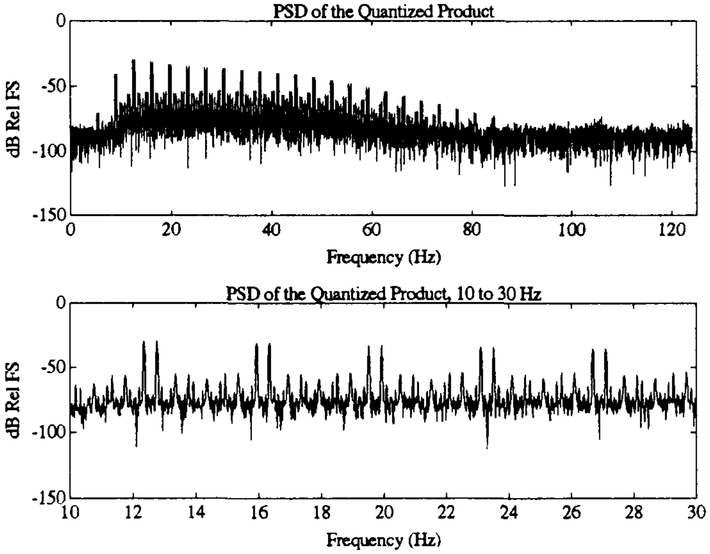


Figure 13 PSD of the Product



the major peaks is about 3.6 Hz. The period of the square wave train is 0.56 seconds. At each transition of the square wave (positive to negative, or negative to positive), the highpass filter output contains a transient. The repetition rate of these transitions is one divided by half the period of the wave train: $1/(0.56/2) = 3.6$

Hz. The bottom plot in Figure 12 shows the same PSD restricted to the frequency range 10 to 30 Hz. Observe the structure of the peaks.

The top plot in Figure 13 shows a PSD of the quantized product in the lower right of Figure 11. The rounding to the nearest integer to simulate quantization raised the noise floor. The PSD values in the frequency range from 0 to 10 Hz are about -86 dB. The PSD values in the frequency range from 100 to 120 Hz are about -63 dB. The spacing between the major peaks is about 3.6 Hz. The major peaks look a little different. The bottom plot in Figure 13 shows the same PSD restricted to the frequency range 10 to 30 Hz. Multiplication by the sine wave cause the single peaks in Figure 12 to split into twin peaks in Figure 13. The spacing between the adjacent large peaks is 0.42 Hz, which is double the frequency of the sine wave. The blobs of energy in the product of highpass filtered square wave and the sine wave are spaced about $1/0.42 = 2.38$ seconds apart. See the quantized product in the lower right hand corner of Figure 11.

Section 5 Noise Floors and the SNR

How does one calculate the amplitude of the noise floor? Assume a data record with M samples of the form:

$$\begin{aligned} &y_n = s_n + q_n \\ &\text{where} \\ &s_n = A\cos(\omega t_n) = \text{ digitizer input} \\ &q_n = \text{ quantization noise} \end{aligned}$$

Note that the peak amplitude of the sine wave is A. Assume also that the sine wave record contains an integer number of cycles. The signal-to-noise ratio in a single DFT frequency bin is the maximum value of the magnitude squared of the DFT of s_n divided by the average value of the magnitude squared of the DFT of q_n . Perform a DFT (with a rectangular window) on the record. This is step 2 of Welch's method with $w[m] = 1$ for all m. The magnitude of the DFT in the spectral bin corresponding to the frequency of the sine wave, $|\mathrm{DFT}(s_n) \text{ at } \omega|$, is [24, page 46]:

$$\begin{aligned} &|\mathrm{DFT}(s_n) \text{ at } \omega| = \frac{A \cdot M}{2} \\ &\text{or} \\ &|\mathrm{DFT}(s_n) \text{ at } \omega|^2 = \frac{A^2 M^2}{4} \end{aligned}$$

The rms value of q_n is $Q/\sqrt{12}$ where Q = code bin width (value of a least significant bit). Since the mean of q_n is zero, the standard deviation of q_n, σ_q, is also $Q/\sqrt{12}$. Perform a DFT (with a rectangular window) on q_n. If $a + jb$ is a complex value of the DFT of q_n at any frequency except DC, then a and b are approximately Gaussian with a mean of zero and variance given by [23, p. 282, Eq. (33)]:

$$\sigma^2 = \sigma_a^2 = \sigma_b^2 = \frac{M \cdot Q^2}{2 \cdot 12}$$

The many computations of the DFT turn an input with a uniform distribution into a DFT value with a Gaussian distribution. The magnitude squared of the normalized DFT value $(a + jb)/\sigma$ is $(a^2 + b^2)/\sigma^2$. Papoulis [20, p. 221] shows that this random variable is chi-square distributed with 2 degrees of freedom, χ_2^2. The expected or average value is:

$$E\left[\frac{a^2 + b^2}{\sigma^2}\right] = E\left[\chi_2^2\right] = 2$$

$$E\left[a^2 + b^2\right] = 2\sigma^2 = \frac{MQ^2}{12}$$

The average value of the magnitude squared of the DFT value $(a + jb)$ is $2\sigma^2$. So, the square of the average value of the magnitude of the DFT of q_n at any frequency (other than DC) is $(MQ^2)/12$. The ideal signal-to-noise ratio is:

$$\begin{aligned}
\text{SNR} &= \frac{\max\left(|\mathrm{DFT}(s_n)|^2\right)}{\mathrm{avg}\left(|\mathrm{DFT}(q_n)|^2\right)} \\
&= \frac{|\mathrm{DFT}(s_n) \text{ at } \omega|^2}{E[a^2 + b^2]} \\
&= \frac{A^2 M^2}{4} \frac{12}{MQ^2} \\
&= \frac{3A^2 M}{Q^2}
\end{aligned}$$

Windows modify the values of $|DFT(s_n)|^2$ and $|DFT(q_n)|^2$. The above equations are correct for the rectangular window. Let w_n be the coefficients of the window normalized so that their maximum value is 1. The peak signal power gain (PSPG) of the DFT window is the square of the sum of the weights.

$$\text{Peak signal power gain} = \left[\sum_{i=1}^{M} w(i)\right]^2$$

The peak signal power gain is also called the coherent power gain. The peak signal power gain of the rectangular window is M. The above equations for the maximum value of the DFT of the signal s_n were for a rectangular window. To convert the result from the rectangular window to other windows, we divide by the peak signal gain of the rectangular window and multiply by that of the new window. The normalized peak signal power gain (NPSPG) of the DFT window is the peak signal power gain divided by the DFT length squared M^2.

$$\text{Normalized peak signal power gain} = \left[\frac{\sum_{i=1}^{M} w(i)}{M}\right]^2$$

The NPSPG is always less than 1. The normalized peak signal power gain of the DFT window attenuates the value of $|\mathrm{DFT}(s_n) \text{ at } \omega|^2$ calculated with a rectangular window. The noise power gain (NPG) of the DFT window is the sum of the squared weights.

$$\text{Noise power gain} = \sum_{i=1}^{M} w^2(i)$$

The noise power gain is also called the incoherent power gain. The noise power gain of the rectangular window is M. The equation for the average value of the DFT of the noise q_n is for a rectangular window. To convert the result from a rectangular window to other windows, we divide by the noise power gain of the rectangular window and multiply by that of the new window. The normalized noise power gain (NNPG) of the DFT window is the noise power gain divided by the DFT length M.

$$\text{Normalized noise power gain} = \frac{\sum_{i=1}^{M} w^2(i)}{M}$$

The normalized noise power gain of the DFT window attenuates the average value of the DFT of the noise, $E[a^2 + b^2]$, calculated for the rectangular window. The ratio of normalized noise power gain to the normalized signal power gain is a measure of the equivalent noise bandwidth (ENBW) of the window in question [12]:

$$\text{ENBW} = \frac{M \sum_{n=1}^{M} w_n^2}{\left(\sum_{n=1}^{M} w_n\right)^2}$$

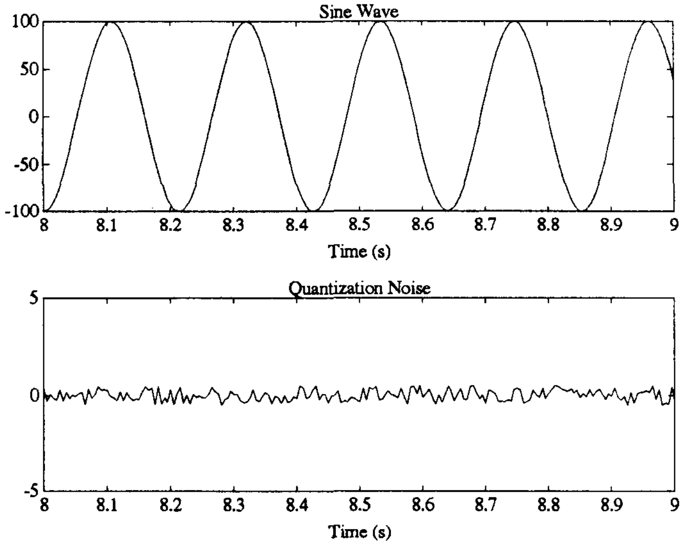
(Equation 11 for ENBW in [10] should be multiplied by the FFT length M. The ENBWs listed in Table I of [10] include this factor.) For any DFT window, the ideal signal-to-noise ratio is:

$$\begin{aligned}
\text{SNR} &= \frac{\text{NPSPG} \times |\text{DFT}(s_n) \text{ at } \omega|^2}{\text{NNPG} \times E[a^2 + b^2]} \\
&= \frac{\text{NPSPG} \times \frac{A^2M^2}{4}}{\text{NNPG} \times \frac{MQ^2}{12}} \\
&= \frac{1}{\text{ENBW}} \frac{3A^2M}{Q^2} \\
&= 10 \log_{10} \left(\frac{1}{\text{ENBW}} \frac{3A^2M}{Q^2} \right) \text{ dB}
\end{aligned}$$

What is the SNR in a modified periodogram as defined by Step 3 of Welch's method? A modified periodogram equals the magnitude squared of a DFT divided by the noise power gain of the window. Since the NPG divides both the signal and the noise components of the DFT, the SNR for the magnitude squared of a DFT and a PSD is the same. However, the peak value of the PSD of s_n does not equal the peak value of the magnitude squared of the DFT of s_n. The peak value of the PSD of s_n is

$$\begin{aligned}
\text{PSD}(s_n) \text{ at } \omega &= \frac{|\text{Windowed DFT}(s_n) \text{ at } \omega|^2}{\text{NPG}} \\
&= \frac{\text{NPSPG}}{\text{NPG}} \frac{A^2M^2}{4} \\
&= \frac{1}{\text{ENBW}} \frac{A^2M}{4}
\end{aligned}$$

Figure 14 Sine Wave and Quantization Noise

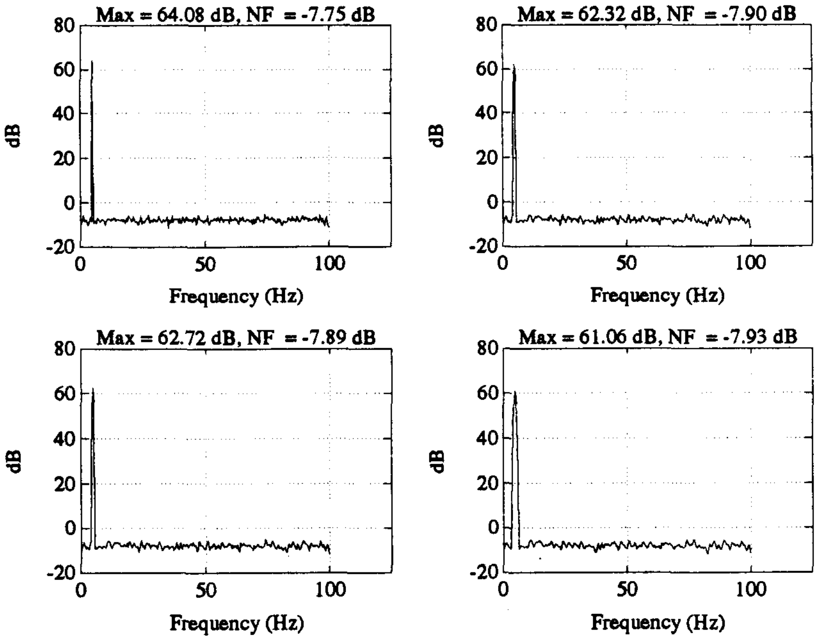


The average value of the PSD of q_n is

$$\begin{aligned} \mathrm{PSD}(q_n) &= \frac{\mathrm{avg}\left(|\mathrm{Windowed\ DFT}(\mathrm{q_n})|^2\right)}{\mathrm{NPG}} \\ &= \frac{\mathrm{NNPG}}{\mathrm{NPG}} \frac{MQ^2}{12} \\ &= \frac{Q^2}{12} = \sigma_q^2 \end{aligned}$$

To make 0 dB correspond to a full scale sine wave, a two-sided PSD computed by Welch's method is divided by $(A^2M)/(4\ \mathrm{ENBW})$, which is the maximum value of a two-sided PSD of a sine wave with amplitude A. The above equations are for two-sided PSDs that contain negative as well as positive frequencies. For a one-sided PSD that contains frequencies from 0 Hz to the Nyquist frequency $1/(2T)$ Hz, multiply the above values by 2. To make 0 dB correspond to a full scale sine wave, a one-sided PSD computed by Welch's method is divided by $(A^2M)/(2\ \mathrm{ENBW})$, which is the maximum value of a one-sided PSD of a sine wave with amplitude A.

Figure 15 PSD of Sine Wave + Quantization Noise

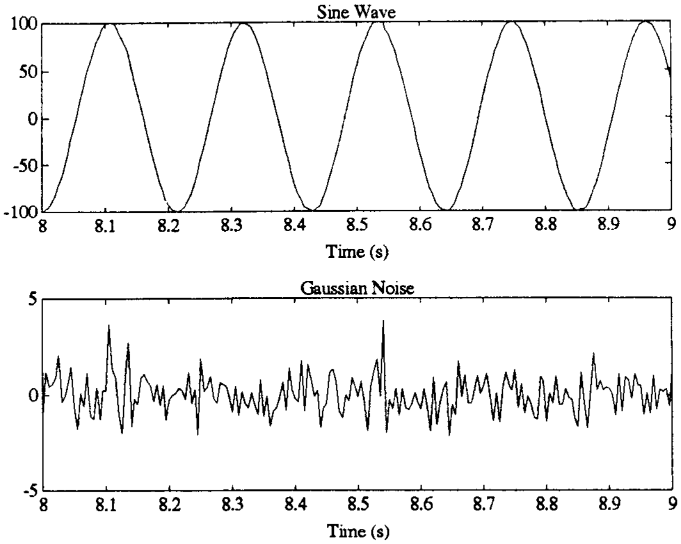


We now consider an example:

$$
\begin{aligned}
y_n &= s_n + q_n \\
&\text{where} \\
s_n &= 100\cos(2\pi 4.69 t_n) = \text{ digitizer input} \\
q_n &= \text{ quantization noise}
\end{aligned}
$$

See Figure 14. The digitizer has 8 bits. The quantization is uniformly distributed on ± 0.5. The sample rate is 200 samples per second. The total number of data samples is 8192. Figure 15 shows PSDs with 4 different windows. The segment or batch size is 512 points for all PSDs. The frequency 4.69 Hz is a multiple of $1/(MT)$ Hz. So, leakage and the picket fence effect are irrelevant for this example. The PSDs are not normalized relative to a full scale sine wave. They are one-sided PSDs with frequencies ranging from 0 to 100 Hz, which is the Nyquist frequency. The noise floor for all of the PSDs is approximately $10\log_{10}(1/6) = 2\sigma_q^2$. The noise floors (NF) are printed on the plots. The noise floor is $2\sigma_q^2$ rather than σ_q^2 because the PSDs are one-sided. The peak values of the PSDs vary with the window from a minimum of 61.06 to a maximum 64.08 dB.

Figure 16 Sine Wave and Gaussian Noise



We consider another example:

$$y_n = s_n + e_n$$

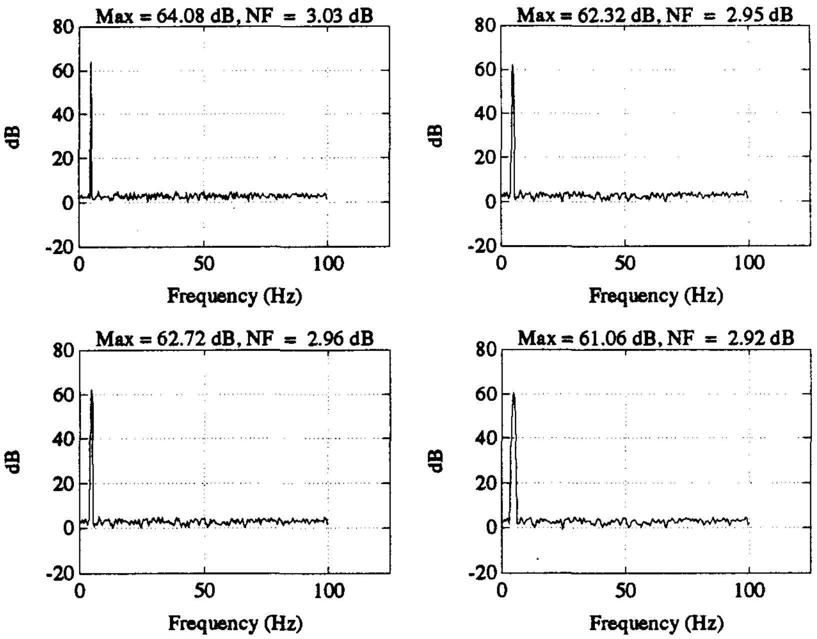
where

$$s_n = 100\cos(2\pi 4.69 t_n) = \text{ digitizer input}$$

$$e_n = \text{ Gaussian noise with unit variance}$$

See Figure 16. The additive noise is Gaussian. Its variance is 12 times larger than the uniformly distributed noise of the previous example. The sample rate is 200 samples per second. The total number of data samples is 8192. Figure 17 shows PSDs with 4 different windows. The segment or batch size is 512 points for all PSDs. The PSDs are not normalized relative to a full scale sine wave. They are one-sided PSDs with frequencies ranging from 0 to 100 Hz, which is the Nyquist frequency. The noise floor for all of the PSDs is approximately equal to $10\log_{10}(2) = 2\sigma_e^2$. The noise floors (NF) are printed on the plots. The noise floor is 2 rather than 1 because the PSDs are one-sided. The peak values of the PSDs vary with the window from a minimum of 61.06 to a maximum 64.08 dB. This is the same range as in the previous example. These examples clearly illustrate that the noise floor of a Welch PSD equals the variance of the additive noise. The distribution does not matter. However, the noise must be uncorrelated from sample to sample.

Figure 17 PSD of Sine Wave + Gaussian Noise



Section 6 Variance of Welch's PSD

Computations on two different data sequences from the same experiment should result in similar PSD estimates. Often, this is not the case. Sometimes, the experiment is different even though it is believed to be the same. Other times, the computations are not identical for the two sequences. When the computations and the experiment are the same, variations will occur. Averaging reduces the variations caused by random noise; it does not eliminate them. The variance of the estimator tells the analyst what size variations are likely to occur.

Welch derived an expression for the variance of his estimator. His derivation is based on the concept of equivalent degrees of freedom developed by Blackman and Tukey in [2, Sect. 6–9]. Blackman and Tukey approximate the PSD estimate, $S_x(f)$, with a multiple of the chi-square random variable, $a\chi^2_\nu$ with ν degrees of freedom. The coefficient of variation of a random variable is the standard deviation divided by the mean. This value is also called the normalized rms error [1]. When the coefficient of variation of an estimator is large, all estimates are close to the average value of the estimator. The mean and variance of a chi-square random variable are ν and 2ν, respectively. So, its coefficient of variation is $\sqrt{2/\nu}$. Approximate $S_x(f)$ with the chi-square random variable which has the same mean and variance:

$$\nu = \frac{2 \cdot \mathrm{mean}^2(\chi^2_\nu)}{\mathrm{var}(\chi^2_\nu)} = \frac{2 \cdot \mathrm{mean}^2(S_x(f))}{\mathrm{var}(S_x(f))}$$

The value of ν that satisfies the above equation is called the equivalent degrees of freedom of the estimator $S_x(f)$. The above equation equates the inverse of the coefficient of variation of the estimator $S_x(f)$ with that of a chi-square random variable. To find the value of ν that satisfies the above equation, we need expressions for the mean and variance of the estimator $S_x(f)$.

In his derivation, Welch assumes that the sequence $x[n]$ has a Gaussian distribution, a zero mean, a local variance that does not vary with time, and uncorrelated adjacent values. His variance expression for f not close to 0 or $1/(2T)$ Hz is

$$\mathrm{var(S_x(f))} = \frac{\text{true } \mathrm{S_x^2(f)}}{\mathrm{K}} \left(1 + 2\sum_{\mathrm{k=1}}^{\mathrm{K-1}} \frac{\mathrm{K-k}}{\mathrm{K}} \rho(\mathrm{k,S})\right)$$

where

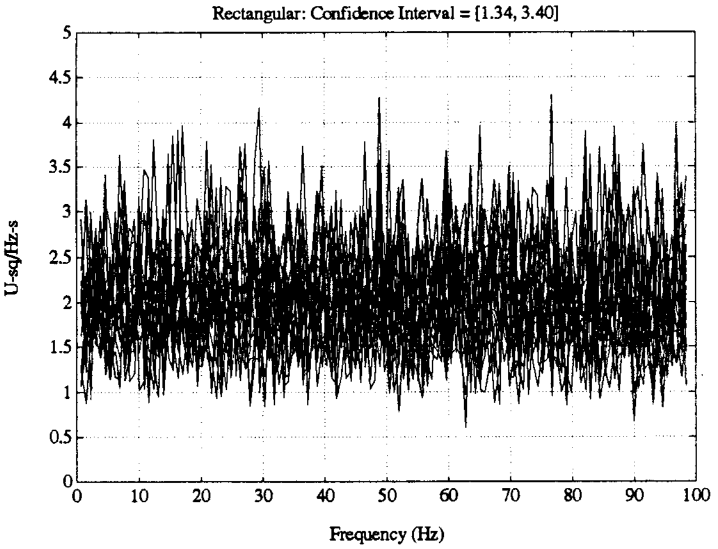
$$\rho(k,S) = \begin{cases} \left(\frac{1}{\mathrm{NPG}} \sum\limits_{m=1}^{M-kS} w(m)w(m+kS)\right)^2, & 1 \leq k \leq \mathrm{int}(M/S) \\ 0, & S \geq M \end{cases}$$

The function, $\rho(k,S)$, is called the normalized window correlation function because of the division by the noise power gain. This forces $\rho(k,S)$ to always be less than or equal to one. The mean value of Welch's estimate equals the true value of the PSD. With this justification, replace the true $S_x(f)$ with its estimate $S_x(f)$ in the above equation. Substitute Welch's variance expression into the equation for the equivalent degrees of freedom. After rearranging, we obtain an expression for the equivalent degrees of freedom:

$$\nu = \frac{2 \cdot \mathrm{mean^2(S_x(f))}}{\mathrm{var(S_x(f))}} = \frac{2 \cdot \mathrm{K}}{1 + \sum\limits_{\mathrm{k=1}}^{\mathrm{K-1}} \frac{\mathrm{K-k}}{\mathrm{K}} \rho(\mathrm{k,S})}$$

From the equivalent degrees of freedom and the percentage points of the chi-square distribution, we get the confidence interval for $S_x(f)$ (see [13, p. 274,

Figure 18 PSDs with Rectangular Window, 50% Overlap



Eq. (8–41)], [22, p. 468])

$$\left(\frac{2\nu S_x(f)}{\chi_\nu^2(1-\alpha/2)}, \frac{2\nu S_x(f)}{\chi_\nu^2(\alpha/2)}\right)$$

where

$\chi_\nu^2(1-\alpha/2) =$ lower $100(1-\alpha)\%$ point of the χ^2 distribution

$\chi_\nu^2(\alpha/2) =$ upper $100(1-\alpha)\%$ point of the χ^2 distribution

The probability that the true $S_x(f)$ is in this interval is $100(1-\alpha)\%$. Tables are normally used to find the percentage point of the chi-square distribution. However, tables are not amenable to automatic computation on a computer. A percentage point of the chi-square distribution is the solution to an equation involving the complement to the incomplete gamma function $Q(a,x)$. Specifically, the value of x that satisfies [21, Section 14.5, page 556]

$$Q(\nu/2, x/2) = 1-\alpha$$

is equal to $\chi_\nu^2(x/2)$. In the above equation, ν is the equivalent degrees of freedom and α is the confidence level. Tables are obviated by solving for x with a root finding program. See [21, Section 6.5, page 171] for a discussion of the incomplete gamma function and its complement. Appendix A contains a C function for computing confidence intervals.

Figure 19 PSDs with Minimum 4-Term Window, 50% Overlap

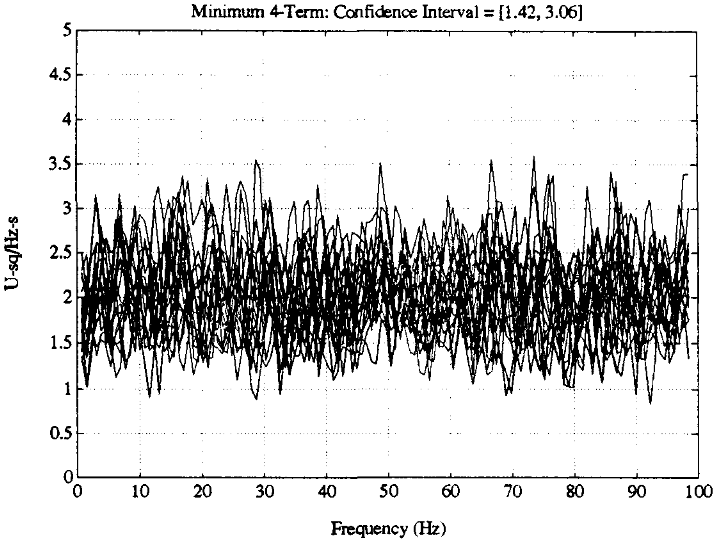


Table 2 Confidence Intervals and Statistics for 50% Overlap

Window	LCL	UCL	Avg	Std	Min	Max
Rectangular	1.24	3.77	2.01	0.48	0.60	4.31
Minumum 4-Term	1.34	3.32	2.01	0.37	0.84	3.59

To illustrate the statiscal properties of Welch's PSD estimator, we performed some computer simulations. Figure 18 shows 20 Welch PSDs with a rectangular window and $\nu = 25.8$. Figure 19 shows 20 Welch PSDs with a minimum 4-term window and $\nu = 37.9$. The confidence intervals are at the 95% confidence level. The parameters for both PSDs are $N = 2560$, $M = 256$, $S = 128$ and $K = 19$. Table 2 compares the confidence intervals and statistics of the two PSD estimators. The only difference in the estimators is the window: rectangular versus minimum 4-term. The minimum 4-term window has less correlation between data segments than the rectangular window. The values of $\rho(k, S)$ are smaller for the minimum 4-term window than they are for the rectangular. The denominator of the expression for the equivalent degrees of freedom is smaller for the minimum 4-term window than it is for the rectangular window. This translates into a smaller confidence interval for the PSD estimator with minimum

4–term window. The lower and upper chi-square confidence limits are listed under "LCL" and "UCL" in the table. The "Avg", "Std", "Min", and "Max" values are the average, standard deviation, minimum, and maximum values over all 20 PSDs. The PSD values at 0 Hz and 100 Hz were not included in the statistics. The average values minus twice the standard deviations, 1.05 and 1.27, are smaller than the lower chi-square confidence limits for both PSD estimators. The upper chi-square confidence limits are larger than the average values plus twice the standard deviations, 2.97 and 2.75, respectively. If one assumes a Gaussian rather than a chi-square distribution, the 95% confidence intervals for the two PSD estimators are [1.05, 2.97] and [1.27, 2.75]. These confidence limits are nearly symmetrical about the average value, which is 2. Look at Figure 18. The bottom envelope of the PSDs is much flatter than the top envelope. The distribution at each frequency is chi-square with the same number of degrees of freedom. The asymmetry of the envelopes is caused by the asymmetry of the chi-square distribution about its mean. Note that the top and bottom envelopes of Figure 19 are much more similar than those of Figure 18. The equivalent degrees of freedom is 37.9 as opposed to 25.8 for the previous figure. As the degrees of freedom increases, the statistics for both estimators becomes Gaussian because of the central limit theorem. The statistics for the estimator of Figure 19 are more nearly Gaussian because of the increased number of equivalent degrees of freedom.

Let us consider a slightly different example. The parameters are as before except that the shift between segments is 256. This results in no overlap between segments. The parameters for both PSDs are $N = 2560$, $M = 256$, $S = 256$ and $K = 10$. Figures 20 and 21 show the PSDs from 20 simulations. Note that the confidence intervals on the plots are the same. The equivalent degrees of freedom equals twice the number of segments for this example, i.e., $\nu = 2K = 20$. The difference in windows does not affect the equivalent degrees of freedom for non-overlapping data segments. Table 3 compares the confidence intervals and statistics of the two PSD estimators. The average values and standard deviations are almost equal. The difference in the bandwidths of the windows is not great enough to force a perceptible difference in the standard deviations. I have no explanation for the difference in minimum and maximum values.

The above plots of 20 PSDs on the same grid give one a feel for the variation in a Welch PSD. More quantitative information is available in a histogram. A histogram of PSD cannot be directly compared to the probability density function

Figure 20 PSDs with Rectangular Window, No Overlap

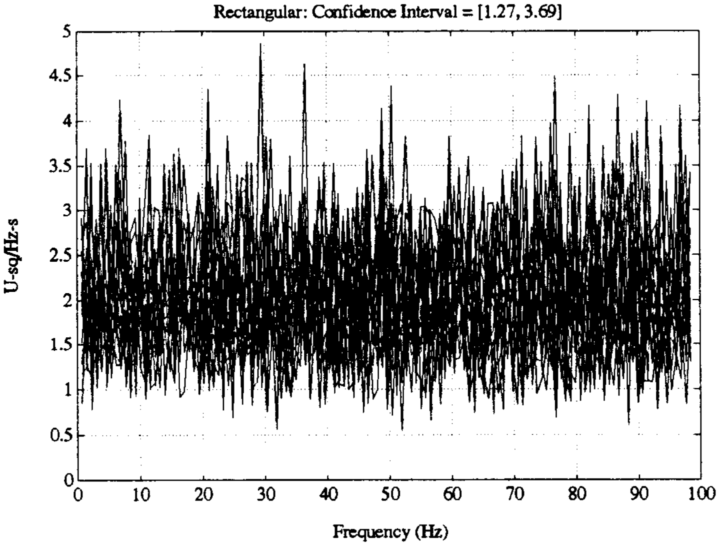
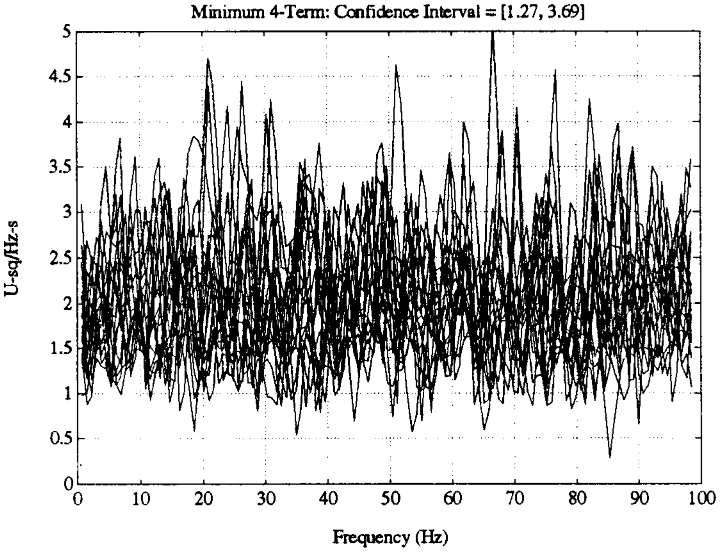


Figure 21 PSDs with Minimum 4–Term Window, No Overlap



(PDF) of a chi-square random variable. To make a comparison requires some special scaling factors. Earlier, we noted that Blackman and Tukey approximate the PSD estimate, $S_x(f)$, with a multiple of the chi-square random variable, $a\chi_\nu^2$. However, we neglected to specify the value of the multiplier a. References [13,

Table 3 Confidence Intervals and Statistics for No Overlap

Window	LCL	UCL	Avg	Std	Min	Max
Rectangular	1.17	4.17	2.01	0.54	0.55	4.86
Minumum 4-Term	1.17	4.17	2.02	0.53	0.28	5.06

pp. 272–274] and [22, pp. 466–468] contain good discussions of how to derive confidence intervals for PSDs using chi-square random variables. Koopmans [13, p. 273, Eq. (8.37)] derives the above equation by solving a system of 2 equations in 2 unknowns. Both authors show that $S_x(f)$ is approximately distributed as $a\chi_\nu^2$ with a equal to the true value of $S_x(f)$ divided by ν. So, if $S_x(f)$ is multiplied by the equivalent degrees of freedom, ν, and divided by the true value of $S_x(f)$, the result is a chi-square random variable:

$$\frac{\nu S_x(f)}{\text{true } \mathrm{S_x(f)}} \approx \chi_\nu^2.$$

We computed 100 PSDs with the rectangular window with 50% overlap. All values, but the first and last, are approximately distributed as a chi-square random variable. We deleted the first and last value of each PSDs because they have a larger variance. Each PSD was scaled as above. All of the scaled PSD values for the rectangular window were pooled together to form a data set with 12700 points. Figure 22 shows a histogram of the pooled PSD values along with true histogram. The true histogram value for a bin was calculated by integrating the chi-square PDF over the bin and multiplying by the sample size, 12700. The bin width for the histogram is one. The histogram is not symmetric. Its slope is steeper below the peak value than above the peak value. This agrees with our observation that the bottom envelope of the PSDs in Figures 18, 19, 20, and 21 is much flatter than the top envelope. We computed 100 PSDs with the minimum 4–term window with 50% overlap. The number of degrees of freedom is 37.9. Figure 23 shows a histogram of the pooled PSD values along with true histogram for PSDs computed with the minimum 4–term window. For both the rectangular and minimum 4–term windows, the measured histograms match the true histograms.

In this report, the error model for the PSD estimates assumes that the errors are distributed as a chi-square random variable. Another common assumption is that the errors are distributed as a normal random variable. For the previous example (the minimum 4-term window with 50% overlap), the true histogram was calculated for a PSD estimator under the assumption that the errors are distributed as a normal random variable. Figure 24 compares the true histograms for the 2 different error models. The true histogram value for a bin was calculated by integrating the PDF over the bin and multiplying by the sample size, 12700. The bin widths for the histograms are one. The chi-square histogram is slightly taller than the normal histogram. Compare Figure 23 with Figure 24. The measured histogram in Figure 23 does not match the true histogram for normally distributed errors in Figure 24. The errors for this PSD estimator are distributed as a chi-square random variable rather than as a normal random variable. The normal histogram is symmetric about its mean value of 37.9. The chi-square histogram is not symmetric about its mean value of 37.9. The chi-square and normal histograms differ significantly in four intervals: (1) bins 10 to 23, (2) bins 23 to 38, (3) bins 38 to 54, and (4) bins 54 to 70. The lower tail of the normal histogram is too large. For bins 10 to 24, the normal histogram lies above the chi-square histogram. For bins 24 to 38, the normal histogram lies below the chi-square histogram. For bins 38 to 54, the normal histogram lies above the chi-square histogram. The upper tail of the normal histogram is too small. For bins 54 to 70, the normal histogram lies below the chi-square histogram. The area of the absolute value of the difference between the two curves is about 1485 PSD values out the total of 12700 PSD values, or about 12%.

In above examples, the data values form a white noise sequence. Data values from an experiment will seldom be a white noise sequence. Recall that Blackman and Tukey proposed the equivalent degrees of freedom argument as an approximation to the true statistics. Nuttall in [17] gives further justification for the use of equivalent degrees of freedom for predicting the stability of Welch's method. Nuttall derives exact probability distribution function for a signal tone in Gaussian noise. He concludes that the probabilistic description based on the concept of equivalent degrees of freedom is satisfactory over a wide range of windows, overlaps and time-bandwidth products. However, one should keep in mind that these confidence intervals are approximations, not absolute laws.

Variance in Terms of the BT Product

Another popular way to describe the variation in a PSD uses the concept of a bandwidth-time (BT) product [1, 2, 15, 16]. In the 1968 and 1971 editions of [1], Bendat and Piersol define three bandwidths: (1) the 3-dB bandwidth, (2)

Figure 22 Measured and True Histograms: Rectangular

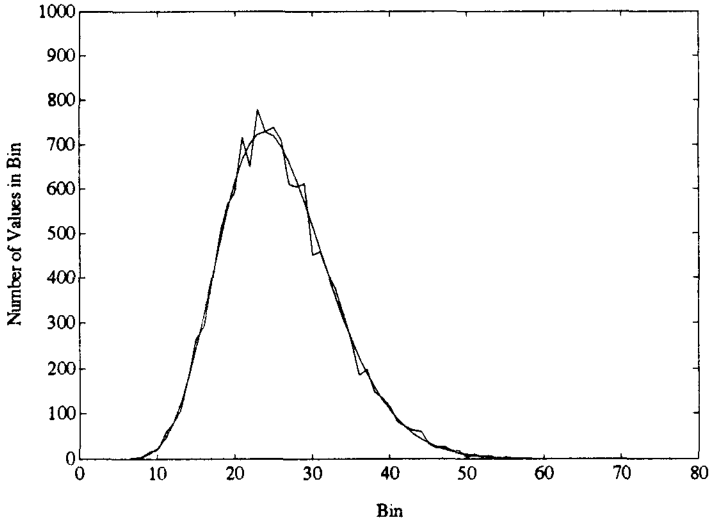
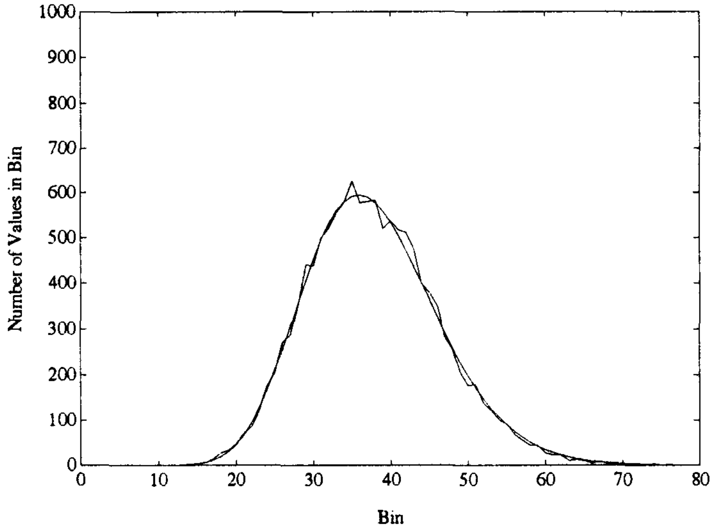
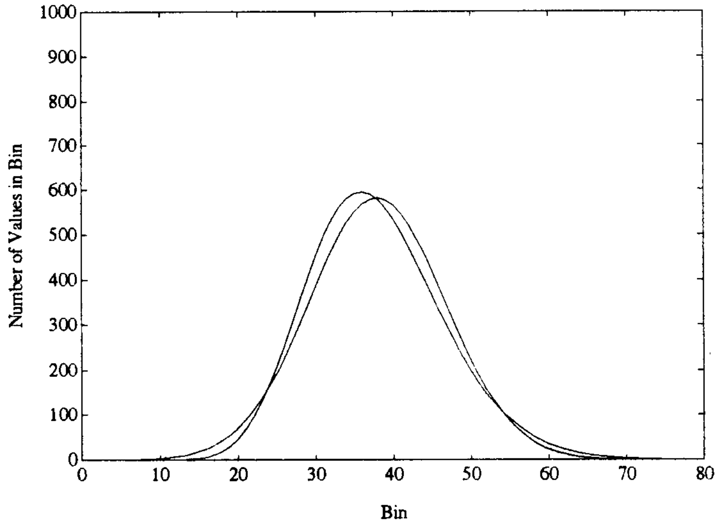


Figure 23 Measured and True Histograms: Minimum 4-Term



the equivalent noise bandwidth, and (3) the equivalent statistical bandwidth. The 3-dB bandwidth is easy to measure. The equivalent noise bandwidth is used as the normalization factor B in Figure 1. The equivalent statistical bandwidth is used for calculating the standard deviation of PSD estimates. For a Hann window, the 3-dB

Figure 24 True Histograms for Chi-Square and Normal RVs



bandwidth is about $1.44/M$ Hz-s, the equivalent noise bandwidth is about $1.50/M$ Hz-s, and the equivalent statistical bandwidth is about $2.08/M$ Hz-s, where M is the FFT length. The ratio of the 3-dB bandwidth to the equivalent statistical bandwidth for common windows [18] varies from about 0.83 to 1.45. For most windows and filters, the 3-dB bandwidth and the equivalent noise bandwidth agree to within 5%. A notable exception is the rectangular window for which they differ by 12%. Table I in Harris' paper [10] lists both the 3-dB and equivalent noise bandwidths for many windows. For the PSD estimate of Figure 1, the statistical bandwidth is given by

$$B = \frac{\left[\int |H(f)|^2 df\right]^2}{\int |H(f)|^4 df}$$

where

$H(f) =$ frequency response of the bandpass filter

The numerator is the square of the noise power gain of the bandpass filter. Note that the magnitude of the bandpass filter is raised to the 4th power in the above denominator. Nuttall in [16] lists 2.079 as the statistical bandwidth of the Hann window.

The equivalent degrees of freedom ν for Welch's PSD estimate depends on the overlap of adjacent segments. If we shift S points between segments, the overlap is $100(M - S)/M\%$. For a Welch PSD estimate with a Hann window, Nuttall in [16] shows that 61% overlap yields the minimum variance and maximum equivalent degrees of freedom. For 2560 total points and a FFT length of 256, the equivalent degrees of freedom ranges from 20 for no overlap to 38.9 for 61% overlap. No overlap corresponds to a shift of 256 points between segments and 61% overlap corresponds to a shift of 100 points between segments. A shift of 1 point between segments or 99.6% overlap yields 38.1 equivalent degrees of freedom.

Nuttall in [18] defines the quality ratio for a PSD estimator $S(f)$ as

$$Q = \frac{\text{var}(S(f))}{\text{avg}^2(S(f))}$$

The minimum value of the quality ratio for a nonparametric PSD estimator is

$$Q = \frac{1}{BT}$$

where B is the equivalent statistical bandwidth and T is the length of the data record. For any amount of overlap between segments, the quality ratio for a Welch PSD estimate is larger than $1/(BT)$ where B is the equivalent statistical bandwidth. The preceding statement is not true for the 3-dB bandwidth or equivalent noise bandwidth. The equivalent degrees of freedom equals $2/Q$. For the Hann window with 2560 total points, a FFT length of 256 and $T = 1$, the equivalent statistical bandwidth is 2.079/256 Hz. For this statistical bandwidth and 2560 seconds of data, the maximum number of degrees of freedom is 41.6. The maximum equivalent degrees of freedom for Welch's method is 38.9 with an overlap of 61%.

Parametric PSD estimators with quality ratios better than $1/(BT)$ can be designed for specific applications. One approach is to fit a ratio of polynomials to the power spectral density function. See Marple's book [14] for some examples.

Nuttall in [16] derives the following approximate rule of thumb for finding the maximum equivalent degrees of freedom:

$$\nu_{max} \approx 3(BT - 1)$$

where

$B =$ 3 dB bandwidth of the window in Hz

$T =$ length of data record in seconds

This empirical formula was derived from tables of equivalent degrees of freedom for 4 different windows. Nuttall derives a more exact expression specifically for the Hann window

$$\nu_{max} \approx 2.88BT - 2.43$$

where

$B =$ 3 dB bandwidth of the window in Hz

$T =$ length of data record in seconds

Gade and Herlufsen in [7] define an effective BT-product per segment as

$$\mathrm{BT}_{\mathrm{eff}} = \frac{1}{K\mathrm{var}(S_x(f))}$$

where

$\mathrm{var}(S_x(f)) =$ variance of Welch's PSD

$K =$ number of segments

References [5] and [7] contain a short table of BT_{eff} versus overlap. From a table of BT_{eff} values, one can compute the variance of Welch's PSD estimate :

$$\mathrm{var}(S_x(f)) = \frac{1}{\mathrm{BT}_{\mathrm{eff}}K}$$

Nuttall's approximate formulas for the maximum degrees of freedom and the effective BT-product are useful for quickly computing the variance of Welch's PSD estimate. However, Nuttall's formulas are approximations. They contain some error. Calculations with the effective BT-product are limited to the tabulated values. For example, Gade and Herlufsen's table does not contain a BT_{eff} for 61% overlap. Neither method allows one to easily find a confidence interval for the PSD estimate. Appendix A contains a C program for computing confidence intervals. It calculates confidence intervals for 5 window types: (1) rectangular, (2) Hann, (3) Hamming, (4) minimum 4–term Blackman-Harris, and (5) Kaiser-Bessel with $\alpha = 3$. The C program prompts the user for the parameters of the PSD estimate: the total number of data points, the FFT length, the number of data points to shift between segments, the window type and the confidence level.

Section 7 Conclusion

Welch's method for computing PSDs and its relation to direct analog computation has been described. A sine wave data sequence results in the largest

PSD amplitudes for a given RMS value. White noise data results in the smallest PSD amplitude for a given RMS value. Signal-to-noise ratios and noise floors depend on the FFT length and the window. Confidence intervals for Welch's PSD estimate are based on his expression for the estimate's variance. A program for calculating confidence intervals is listed in Appendix A.

Bibliography

1. J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures*, New York: John Wiley & Sons, Inc., 1986.
2. R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra*, New York: Dover Publications, Inc., 1959.
3. J. W. Cooley and J. W. Tukey, "An Algorithm for Machine Computation of Complex Fourier Series," *Mathematics of Computation*, Vol. 19, April 1965, pp. 297-301.
4. P. M. Embree and B. Kimble, *C Language Algorithms for Digital Signal Processing*, Englewood Cliffs: Prentice-Hall, Inc., 1991.
5. Svend Gade and Henrik Herlufsen, "Windows to FFT Analysis," *Sound and Vibration*, March 1988, pp. 14–22
6. Svend Gade and Henrik Herlufsen, "Windows to FFT Analysis (Part I)," *Bruel & Kjaer Technical Review*, No. 3, 1987, pp. 1–28.
7. Svend Gade and Henrik Herlufsen, "Windows to FFT Analysis (Part II)," *Bruel & Kjaer Technical Review*, No. 3, 1987, pp. 1–35.
8. W. A. Gardner, *Introduction to Random Processes*, New York: Macmillan Publishing Company, 1986.
9. W. A. Gardner, *Statistical Spectral Analysis*, Englewood Cliffs: Prentice-Hall, Inc., 1988.
10. F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE*, Vol. 66, No. 1, January 1978, pp. 51–83.
11. *IEEE Trial-Use Standard for Digitizing Waveform Recorders*, IEEE Std 1057, IEEE, New York, NY, July 1989.
12. C. S. Lindquist, *Adaptive & Digital Signal Processing*, Miami: Steward & Sons, 1989.
13. L. H. Koopmans, *The Spectral Analysis of Time Series*, New York: Academic Press, 1974.
14. S. L. Marple, Jr., *Digital Spectral Analysis*, Englewood Cliffs: Prentice-Hall, Inc., 1987.
15. D. E. Newland, *Random Vibrations and Spectral Analysis*, New York: Longman Group Limited, 1975.

16. A. H. Nuttall, "Spectral Estimation by Means of Overlapped Fast Fourier Transform Processing of Windowed Data," NUSC Technical Report 4169, Naval Underwater Systems Center, Newport, Rhode Island, October 1971.
17. A. H. Nuttall, "Probability Distribution of Spectral Estimates Obtained via Overlapped FFT Processing of Windowed Data," NUSC Technical Report 5529, Naval Underwater Systems Center, Newport, Rhode Island, December 1976.
18. A. H. Nuttall, "Some Windows with Very Good Sidelobe Behavior," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-29, No. 1, February 1981, pp. 84–91.
19. A. Papoulis, *Probability, Random Variables and Stochastic Processes*, New York: McGraw-Hill Book Company, 1965.
20. A. Papoulis, *Probability & Statistics*, Englewood Cliffs: Prentice-Hall, Inc., 1990.
21. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes in C*, New York: Cambridge University Press, 1988.
22. M. B. Priestley, *Spectral Analysis and Time Series, Volume 1*, New York: Academic Press, 1981.
23. J. Schoukens and J. Renneboog, "Modeling the Noise Influence on the Fourier Coefficients After a Discrete Fourier Transform," *IEEE Transactions on Instrumentation and Measurement*, Vol. IM-33, No. 3, September, 1986.
24. S. D. Stearns and R. A. David, *Signal Processing Algorithms*, Englewood Cliffs: Prentice-Hall, Inc., 1988.
25. P. D. Welch, "A Direct Digital Method of Power Spectrum Estimation," *IBM Journal*, April, 1961.
26. P. D. Welch, "The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms," *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-15, No. 2, June, 1967.

Appendix A Program to Compute Confidence Intervals

This appendix contains some C functions to compute confidence intervals for Welch's PSD estimate. The function uses several routines from the book *Numerical Recipes in C* [21] to calculate the percentage points of a chi-square distribution. Some functions for computing the Hann window, Hamming window, minimum 4–term Blackman-Harris window and Kaiser-Bessel window for $\alpha = 3.0$ are also included. References [10] by Harris and [18] by Nuttall are good general references on window functions. Gade and Herlufsen in [7, Appendix B, Table B.1] list coefficients for Kaiser-Bessel window for $\alpha = 3.0$. These are the Kaiser-Bessel weights used in the B & K Analyzers Types 2032 and 2034. Gade and Herlufsen list slightly different weights for the Kaiser-Bessel window in [5] and [6]. In their papers, Harris and Nuttall define the Kaiser-Bessel window in terms of the modified zero-order Bessel function of the first kind. Gade and Herlufsen do not explain how their coefficients are related to the modified zero-order Bessel function of the first kind. The C function below for the Kaiser-Bessel window uses Gade and Herlufsen's weights in [7, Appendix B, Table B.1]. References [16] by Nuttall and [26] by Welch contain good discussions of how to calculate the equivalent degrees of freedom for Welch's PSD estimate. Reference [13, pp. 272–277] by Koopmans clearly describes how to compute confidence intervals from the equivalent degrees of freedom.

The C function prompts the user for the parameters of the PSD estimate: the total number of data points, the FFT length, the number of data points to shift between segments, the window type and the confidence level. The program is designed to handle large values for the total number of data points, the FFT length, and the number of data points to shift between segments. The FFT length and shift between segments must be less than the total number of points. Try a few billion for the total number of points. For such large numbers, the program computes for a about a minute on a 25 MHz 80386 with a math coprocessor. The confidence level can be any number greater than 0 and less than 100.

The C functions from [21] were modified to handle larger numbers. All of the `float` declarations in the C functions `gammln()`, `gammq()`, `gcf()`, `gser()`, and `rtbis()` were changed to `double` declarations. The named constant `ITMAX` was changed to 1000 from 100 in the C function `gcf()`. The named constant ITMAX was changed to 10000 from 100 in the C function gser(). The named constant EPS was changed to 3.0e−16 from 3.0e−7 in the C functions gcf() and gser(). The named constant JMAX was changed to 100 from 40 in C function rtbis().

```
/***********************************************
confiden.c  Otis Solomon  12-11-91
***********************************************/
#include <stdio.h>
#include <math.h>
/***********************************************
These files are in the book "Numerical Recipes in C"
by Press, Flannery, Teukolsky & Vetterling.
***********************************************/
#include "gammln.c"
#include "gammq.c"
#include "gcf.c"
#include "gser.c"
#include "nrerror.c"
#include "rtbis.c"

#define SQR(a) ((a)*(a))

double hannwin(N,k)
unsigned long int N;
unsigned long int k;
/***********************************************
hannwin(N,k) computes the kth weight of the
Hanning window of length N.
Window Parameters:
alfa      highest side-lobe (dB)   3 dB BW (bins)
 2.0            -31.47                  1.44
Example: hannwin(N,k)
Input: N = length of window
       k = index of weight
Return value: value of the kth window weight
```

```
************************************************/
{
 double pi, w, x;
 pi=acos(-1.0);
 x=(double)k;
 w=0.5*(1.0-cos(2.0*pi*x/N));
 return w;
}

double hammwin(N,k)
unsigned long int N;
unsigned long int k;
/************************************************
hammwin(N,w) computes the kth weight of the Hamming
window of length N in a column.
Window Parameters:
   highest side-lobe (dB)   3 dB BW (bins)
         -43.19                  1.3
Example: hammwin(N,k)
Input: N = length of window
       k = index of weight
Return value: value of the kth window weight
************************************************/
{
 double pi, w, x;
 pi=acos(-1.0);
 x=(double)k;
 w=0.54-0.46*cos(2.0*pi*x/N);
 return w;
}

double kbeswin(N,k)
unsigned long int N;
unsigned long int k;
/************************************************
kbeswin(N,w) computes the kth weight of the Kaiser-
```

```
Bessel window of length N.
References:
  1. Svend Gade and Henrik Herfulsen, "Windows
to FFT Analysis (Part II)," Bruel & Kjaer Technical
Review, No. 3, 1987, Table B.1, p. 14.
Example: kbeswin(N,k)
Input: N = length of window
       k = index of weight
Return value: value of the kth window weight
************************************************/
{
 double pi, w, x, a0, a1, a2, a3;
 double sum;
 pi=acos(-1.0);
 a0=1.0;
 a1=(-1.298);
 a2=0.244;
 a3=(-0.003);
 /* Normalize st max value = 1.0. */
 /* sum(ai) = 1.0. */
 sum=a0+(-a1)+a2+(-a3);
 a0=a0/sum;
 a1=a1/sum;
 a2=a2/sum;
 a3=a3/sum;
 x=(double)k;
 w=a0+a1*cos(2.0*pi*x/N)+
      a2*cos(4.0*pi*x/N)+a3*cos(6.0*pi*x/N);
 return w;
}

double rectwin(N,k)
unsigned long int N;
unsigned long int k;
/************************************************
rectwin(N,w) computes the kth weight of the
```

```
rectangular window of length N.
Example: rectwin(N,k)
Input: N = length of window
       k = index of weight
Return value: value of the kth window weight
************************************************/
{
 return 1.0;
}

double trm4win(N,k)
unsigned long int N;
unsigned long int k;
/************************************************
trm4win(N,w) computes the kth weight of the minimum
4 term window of length N.
Window Parameters:
   highest side-lobe (dB)   3 dB BW (bins)
         -92.01                 1.9
Example: trm4win(N,w)
Input: N = length of window
       k = index of weight
Return value: value of the kth window weight
************************************************/
{
 double pi, w, x, a0, a1, a2, a3;
 pi=acos(-1.0);
 a0=0.35875;
 a1=0.48829;
 a2=0.14128;
 a3=0.01168;
 x=(double)k;
 w=a0-a1*cos(2.0*pi*x/N)+
      a2*cos(4.0*pi*x/N)-a3*cos(6.0*pi*x/N);
 return w;
}
```

```
double rho(w,M,k,S)
double (*w)();
unsigned long int M,k,S;
/************************************************
 rho computes the window autocorrelation function
Inputs:
 w = window
 M = FFT length
 k = lag of window correlation
 S = number of new points in each FFT
Return Value:
 r = autocorrelation of window
*************************************************/
{
 double r = 0.0, Pw = 0.0;
 unsigned long int i;
 Pw=0.0;
 r=0.0;
 for(i=0; i<=M-1; i++)
  Pw=Pw+w(M,i)*w(M,i)/M;
 for(i=0;i<=(M-k*S-1); i++)
  r=r+w(M,i)*w(M,i+k*S);
 r=SQR(r/(M*Pw));
 return r;
}

double edf(w,M,N,S)
double (*w)();
unsigned long int M,N,S;
/************************************************
 edf computes the equivalent degrees of freedom
Inputs:
 w = window
 M = FFT length
 N = total number of data points
 S = number of new points in each FFT
```

```
Return value:
 v = equivalent degrees of freedom in a PSD estimate
*************************************************/

{
 double s = 0.0, v = 0.0;
 unsigned long int i, k;
 k=1+(N-M)/S;
 printf("Number of segments = %ld\n",k);
 s=0.0;
 for(i=1; i<=k-1; i++)
 {
  if( i*S < (M-1) )
  {
   s=s+((double)k-i)/((double)k)*rho(w,M,i,S);
  }
 }
 /* Inverse of Welch’s variance expression */
 v = k/(1.0+2.0*s);
 printf("P(f) = estimate of PSD value ");
 printf("at frequency f\n");
 printf("Welch’s variance estimate = ");
 printf("%#8.6f * (P(f) squared)\n", 1.0/v);
 printf("Welch’s standard deviation estimate = ");
 printf("%#8.6f * P(f)\n", sqrt(1.0/v));
 /* Degrees of freedom */
 v = 2.0*v;
 return v;
}

/* Global variables */
double v_dof,p_cl;

double gammq2(x)
double x;
{
 return gammq(v_dof/2,x/2)-p_cl;
}

void confiden(w,M,N,S,cl)
double (*w)(),*cl;
unsigned long int M,N,S;
/************************************************
confiden finds confidence limits of a PSD estimate
Inputs:
 w = window
 M = FFT length
 N = total number of data points
 S = number of new points in each FFT
Output:
 cl = confidence limits
*************************************************/
{
 double xacc, chi_upper, chi_lower, alpha, conf;
 v_dof = edf(w,M,N,S);
 printf("Equivalent degrees of freedom = %#4.1f\n",
  v_dof);
 xacc=1.0e-12;
 /* Set alpha to 0.975 for 95% confidence interval. */
 do {
  printf("Enter confidence level as a percentage: ");
  scanf("%lf",&conf);
 } while ((conf<=0.0)||(conf>=100.0));
 fflush(stdin);
 conf=conf/100.0;
 alpha=1.0-(1.0-conf)/2.0;
 p_cl=1.0-alpha;
 chi_upper=rtbis(gammq2,0.0,5*v_dof,xacc);
 cl[0]=v_dof/chi_upper;
 p_cl=alpha;
 chi_lower=rtbis(gammq2,0.0,5*v_dof,xacc);
 cl[1]=v_dof/chi_lower;
 printf("Confidence interval for chi-square RV = ");
 printf("[%g, %g]\n",chi_lower,chi_upper);
 return;
}

void main()
{
 double (*w)(), cl[2], y;
 unsigned long int N=0, M=0, S=0, i;
 double Nd=0.0, Md=0.0, Sd=0.0, pick;
 printf("\n\n");
 printf("Compute confidence interval ");
 printf("for Welch’s PSD estimate.");
 printf("\n");
 printf("Enter total number of points: ");
 scanf("%lf",&Nd);
 N=Nd;
 do {
  printf("Enter FFT length: ");
  scanf("%lf",&Md);
 } while ((Md<=0.0)||(Md>Nd));
 M=Md;
 do {
  printf("Enter number of points ");
  printf("to shift between segments: ");
  scanf("%lf",&Sd);
 } while ((Sd<=0.0)||(Sd>Nd));
 S=Sd;
 printf("Enter: 1 for Rectangular\n");
 printf("       2 for Hann\n");
 printf("       3 for Hamming\n");
 printf("       4 for Minimum 4-Term ");\
 printf("Blackman-Harris\n");
 printf("       5 for Kaiser-Bessel ");
 printf("with alpha = 3.0\n");
 do {
  printf("Choose a window: ");
  scanf("%lf",&pick);
 } while ((pick<=0.0)||(pick>=6));
 fflush(stdin);
 if(pick==1.0) w=rectwin;
 if(pick==2.0) w=hannwin;
 if(pick==3.0) w=hammwin;
 if(pick==4.0) w=trm4win;
 if(pick==5.0) w=kbeswin;
 y=0.0;
 for(i=0; i<M; i++)
  y=y+((*w)(M,i))*((*w)(M,i));
 printf("Normalized noise power gain of window = ");
 printf("%f\n", y/M);
 confiden(w,M,N,S,cl);
 printf("Confidence interval for PSD = ");
 printf("[%g * P(f), %g * P(f)]\n", cl[0],cl[1]);
 cl[0]=10.0*log10(cl[0]);
 cl[1]=10.0*log10(cl[1]);
 printf("Confidence interval for PSD = ");
 printf("[P(f) - %g, P(f) + %g] dB\n",
  -cl[0],cl[1]);
}
```

Distribution:

1544	Charlie Adams
1545	Dave Martinez
1545	John Red-Horse
1551	Bill Millard
1554	Ed Clark
2334	Stu Kohler
2344	Bill Hensley
2700	Ruth David
2713	Wayne Lathrop
2715	Bob Peet
2715-2	Carl Dreyer
2722	Jeanne Bando
2722	Jerry Biedscheid
2722	Bob Isidoro
2722	Mike Rogers
2722	Otis Solomon (20)
2725	Doug Dederman
2732	Don Thalhammer
2737	Jim Nakos
2741	Tom Baca
2741	Jerry Cap
2741	Bill Dunn
2741	Bill Sieger
2741	Ervin Smith
2741	John Snethen
2741	Roger Zimmerman
2743	Ron Rodeman
2743	Vesta Bateman
2743	Tom Carne
2743	Neil Davie
2743	Jim Lauffer
2743	Randy Mayes
2743	Dennis Roach
2744	Dave Smallwood
2744	Dan Gregory

2744	Tom Paez
2744	Jonathan Rogers
2745	Vern Gabbard
2757	Pat Walter
5143	Cliff Harris
5144	Dave Ryerson
5144	Ron Franco
5144	Cass Gowins
5144	Jeff Kalb
5144	Vince Salazar
5144	Ray Wood
5147	Garth Maxam
5167	Glenn Bell
6258	Thurlow Caffey
6258	Gerry Sleefe
9141	Paul Kuenstler (3)
9144	Jerry McDowell
9145	Keith Miller
9145	Dave Overmier
9145	Dan Talbert
9222	Harold Eyer
9222	Dave Smith
9223	Dennis Reynolds
9223	Steve Gentry
9223	Rex Kay
9223	Jeff Kern
9223	Kurt Lanes
9244	Pres Herrington
9244	Bobby Corbel
9244	Dick Kromer
9244	Tim MacDonald
9311	Sam Stearns
8523-2	Central Technical Files
3141	S. A. Landenberger (5)
3145	Document Processing for DOE/OSTI (8)
3151	G. C. Claycomb (3)