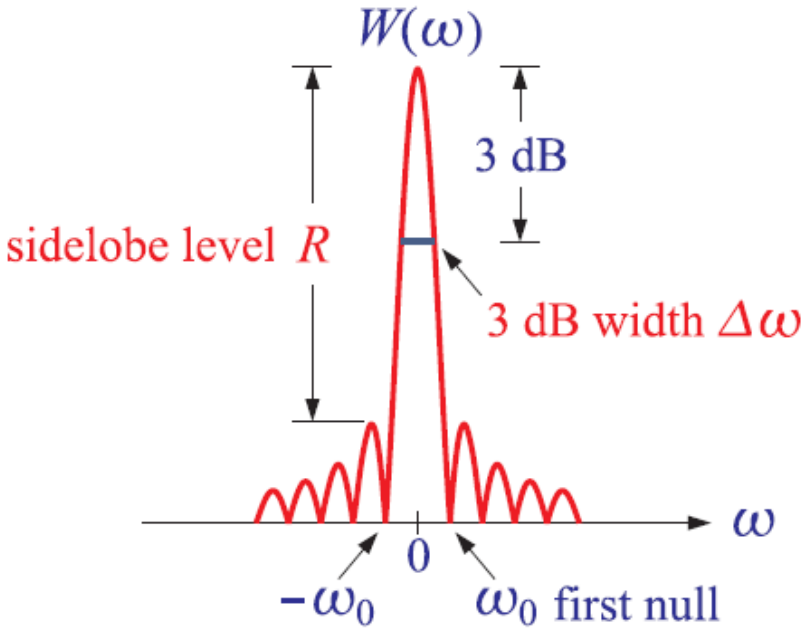
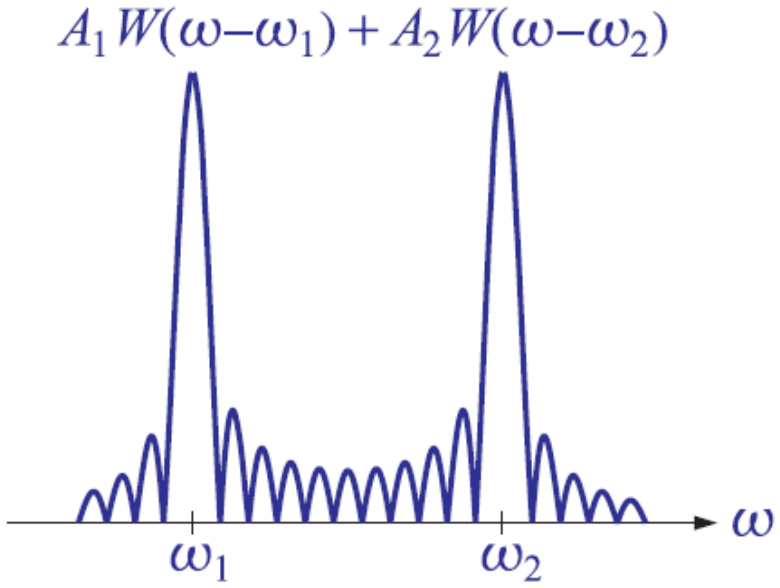
DSA – Feb. 22, 2021

Topics: DTFT, spectral analysis, windowing vs. resolution tradeoffs, power spectrum estimation, sample autocorrelation, periodogram, periodogram averaging and smoothing, window types: rectangular, Hamming, Kaiser, DPSS, Chebyshev.





DTFT
frequency resolution and windowing

I2SP – Ch.9
O&S – Ch.10

The discrete Fourier transform (DFT) and its fast implementation, the fast Fourier transform (FFT), have three major uses in DSP:

(a) the numerical *computation* of the frequency spectrum of a signal

(b) the efficient implementation of *convolution* by the FFT

(c) the *coding* of waveforms, such as speech or pictures, for efficient transmission and storage

To compute the spectrum of an analog signal digitally, a finite-duration record of the signal is sampled and the resulting samples are transformed to the frequency domain by a DFT or FFT algorithm. The sampling rate f_s must be fast enough to minimize aliasing effects. If necessary, an analog antialiasing prefilter may precede the sampling operation.

The spectrum of the sampled signal $\widehat{X}(f)$ is the replication of the desired analog spectrum $X(f)$ at multiples of the sampling rate f_s, as given by the Poisson summation formula. With the proper choice of sampling rate and prefilter, it can be guaranteed that $\widehat{X}(f)$ will agree with the desired $X(f)$ over the Nyquist interval,

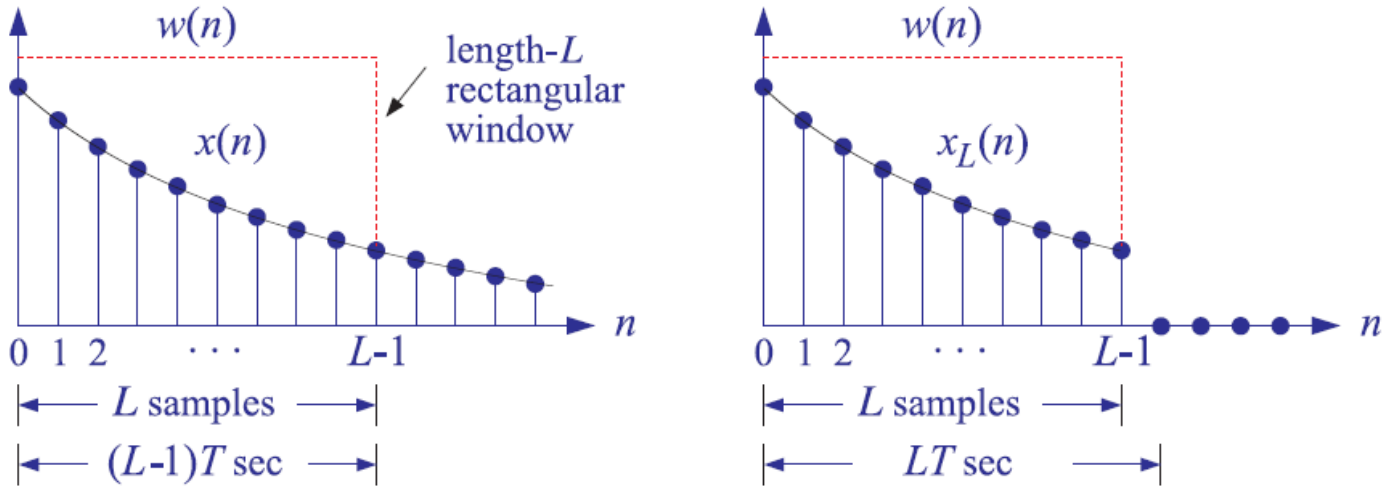
$$T\widehat{X}(f) = X(f), \qquad |f| \leq \frac{f_s}{2}$$

This property is a direct consequence of the sampling theorem, following from the non-overlapping of the spectral replicas in $\widehat{X}(f)$. However, if the replicas overlap, they will contribute to the right-hand side, making the sampled spectrum different from the desired one:

$$T\widehat{X}(f) = X(f) + \underbrace{X(f - f_s) + X(f + f_s) + \cdots}_{\text{replicas}}, \quad |f| \leq \frac{f_s}{2}$$

Because digitally we can only compute $\widehat{X}(f)$, it is essential that the replicated terms remain small over the Nyquist interval, which happens when $X(f)$ falls off sufficiently fast with f.

Even though $\widehat{X}(f)$ is the closest approximation to $X(f)$ that we can achieve by DSP, it is still not computable because generally it requires an infinite number of samples $x(nT)$, $-\infty < n < \infty$. To make it computable, we must make a second approximation to $X(f)$, keeping only a finite number of samples, say, $x(nT)$, $0 \leq n \leq L-1$. This *time-windowing* process is illustrated below.



In terms of the time samples $x(nT)$, the original sampled spectrum $\widehat{X}(f)$ and its time-windowed version $\widehat{X}_L(f)$ are given by:

$$\widehat{X}(f) = \sum_{n=-\infty}^{\infty} x(nT)e^{-2\pi jfnT}$$

(DTFT)

$$\widehat{X}_L(f) = \sum_{n=0}^{L-1} x(nT)e^{-2\pi jfnT}$$

We may take the duration of the data record to be, in seconds and in samples:

$$\boxed{T_L = LT \quad \Rightarrow \quad L = \frac{T_L}{T} = f_s T_L}$$

The windowed signal may be thought of as an infinite signal which is zero outside the range of the window and agrees with the original one within the window. Defining the *rectangular window* of length L:

$$w(n) = \begin{cases} 1, & \text{if } 0 \le n \le L-1 \\ 0, & \text{otherwise} \end{cases}$$

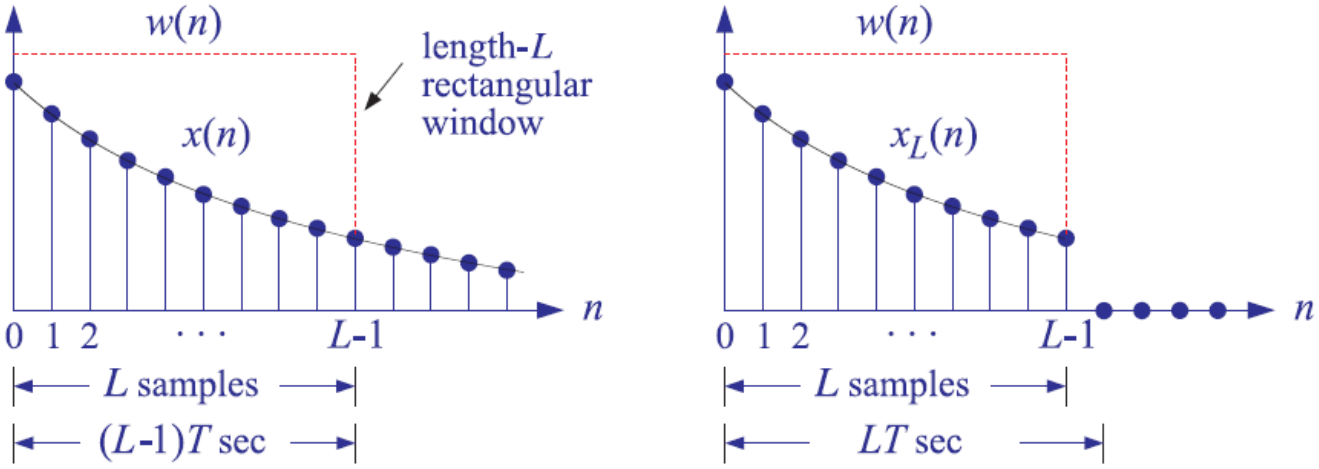
then, the windowed signal can be expressed as follows:

$$x_L(n) = x(n)w(n) = \begin{cases} x(n), & \text{if } 0 \le n \le L-1 \\ 0, & \text{otherwise} \end{cases}$$

In terms of digital frequency, $\omega = 2\pi f / f_s$, we may denote the DTFTs,

$$\boxed{\omega = \frac{2\pi f}{f_s}}$$

$$\boxed{\begin{aligned} X(\omega) &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \\ X_L(\omega) &= \sum_{n=0}^{L-1} x(n)e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x_L(n)e^{-j\omega n} \end{aligned}} \quad \text{(DTFT)}$$



As the length L of the data window increases, the windowed signal $x_L(n)$ becomes a better approximation of $x(n)$, and thus, $X_L(\omega)$, a better approximation of $X(\omega)$. In general, the windowing process has two major effects:

1. It reduces the *frequency resolution* of the computed spectrum, in the sense that the smallest resolvable frequency difference is limited by the length of the data record, that is, $\Delta f = 1/T_L$. This is the well-known "uncertainty principle."

2. It introduces *spurious* high-frequency components into the spectrum, which are caused by the sharp clipping of the signal $x(n)$ at the left and right ends of the rectangular window. This effect is referred to as "frequency leakage."

Both effects can be understood by deriving the precise connection of the windowed spectrum $X_L(\omega)$ to the unwindowed one $X(\omega)$. Using the property that the Fourier transform of the *product* of two time functions is the *convolution* of their Fourier transforms, we obtain the frequency-domain version of, $x_L(n) = x(n)w(n)$,

$$\boxed{X_L(\omega) = \int_{-\pi}^{\pi} X(\omega')W(\omega - \omega')\,\frac{d\omega'}{2\pi}}$$

where $W(\omega)$ is the DTFT of the rectangular window $w(n)$, that is,

$$W(\omega) = \sum_{n=0}^{L-1} w(n)e^{-j\omega n}$$

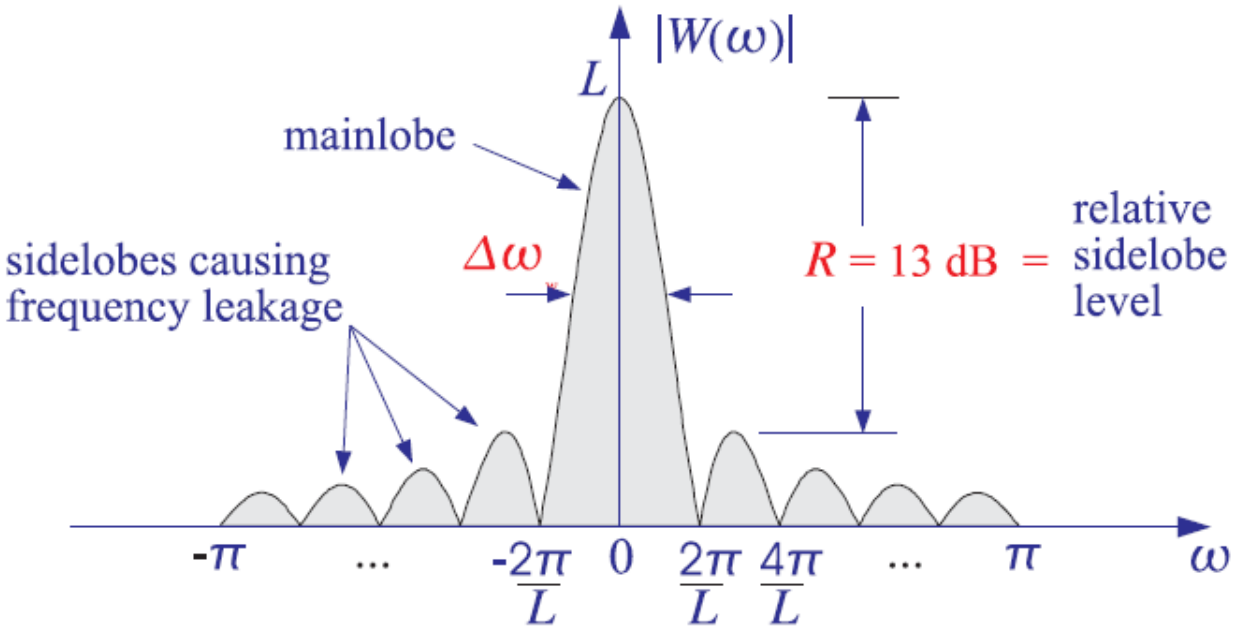
It can be thought of as the evaluation of the z-transform on the unit circle at $z = e^{j\omega}$. Setting $w(n) = 1$ in the sum, we find:

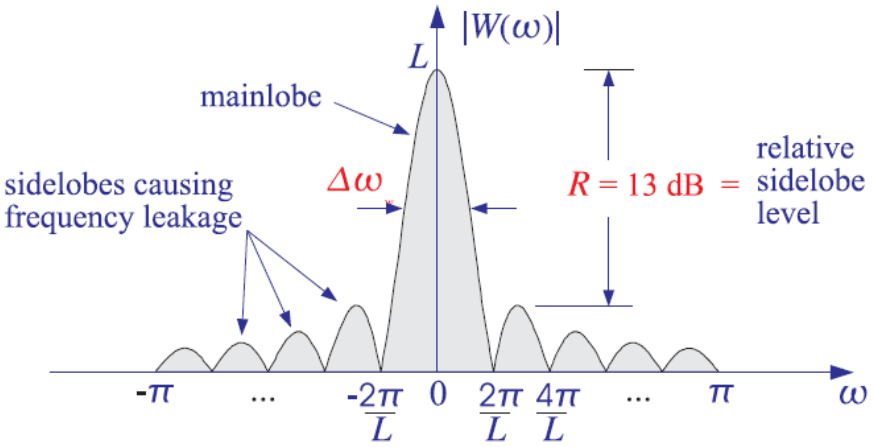
$$W(z) = \sum_{n=0}^{L-1} w(n)z^{-n} = \sum_{n=0}^{L-1} z^{-n} = \frac{1 - z^{-L}}{1 - z^{-1}}$$

Setting $z = e^{j\omega}$, we find for $W(\omega)$:

$$\boxed{W(\omega) = \frac{1 - e^{-jL\omega}}{1 - e^{-j\omega}} = \frac{\sin(\omega L/2)}{\sin(\omega/2)}\,e^{-j\omega(L-1)/2}}$$

$$W(\omega) = \frac{1 - e^{-jL\omega}}{1 - e^{-j\omega}} = \frac{\sin(\omega L/2)}{\sin(\omega/2)}\, e^{-j\omega(L-1)/2}$$





see project-4
for the more accurate
3-dB width

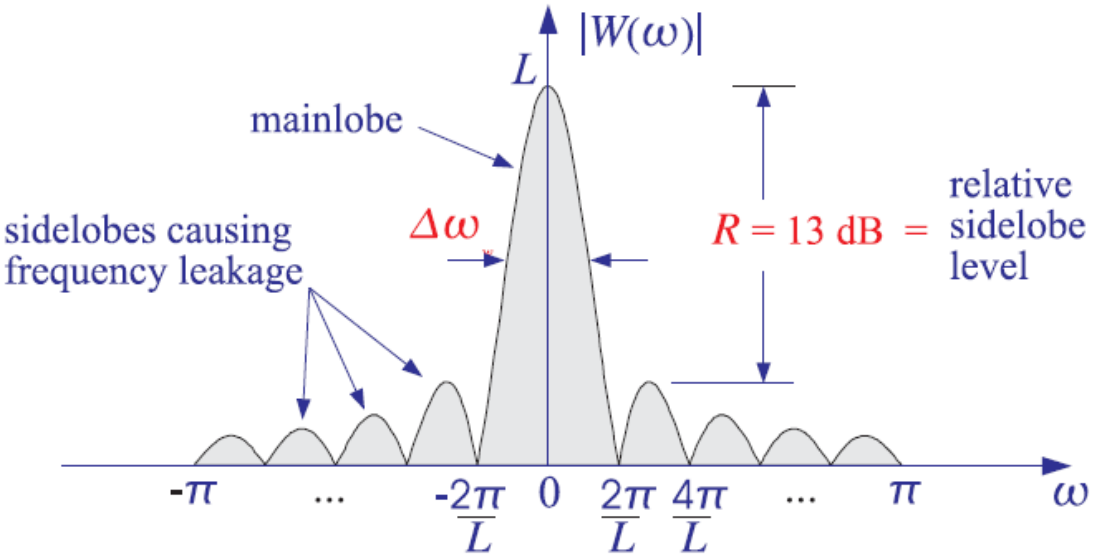
The mainlobe peak at DC dominates the spectrum, because $w(n)$ is essentially a DC signal, except when it cuts off at its endpoints. The higher frequency components that have "leaked" away from DC and lie under the sidelobes represent the sharp transitions of $w(n)$ at the endpoints.

The *width* of the mainlobe can be defined in different ways. For example, we may take it to be the width of the base, $4\pi/L$, or, take it to be the 3-dB width, that is, where $|W(\omega)|^2$ drops by $1/2$. For simplicity, we will define it to be *half* the base width, that is, in units of radians per sample:

$$\boxed{\Delta\omega_w = \frac{2\pi}{L}} \qquad \text{(rectangular window width)}$$

In units of Hz, it is defined through $\Delta\omega_w = 2\pi\Delta f_w/f_s$. Thus,

$$\boxed{\Delta f_w = \frac{f_s}{L} = \frac{1}{LT} = \frac{1}{T_L}}$$



The peak of the first sidelobe occurs approximately halfway between the two zeros $2\pi/L$ and $4\pi/L$, that is, at $\omega = 3\pi/L$. Using $W(0) = L$, we find that the *relative* heights are essentially independent of L:

$$\left|\frac{W(\omega)}{W(0)}\right|_{\omega=3\pi/L} = \left|\frac{\sin(\omega L/2)}{L\sin(\omega/2)}\right| = \left|\frac{\sin(3\pi/2)}{L\sin(3\pi/2L)}\right| \simeq \frac{1}{L\cdot(3\pi/2L)} = \frac{2}{3\pi}$$

We assumed that L was fairly large (typically, $L \geq 10$), and used the small-x approximation $\sin x \simeq x$ with $x = 3\pi/2L$. In decibels, the *relative sidelobe* level is

$$R = 20\log_{10}\left|\frac{W(\omega)}{W(0)}\right|_{\omega=3\pi/L} \simeq 20\log_{10}\left(\frac{2}{3\pi}\right) = -13.46 \text{ dB}$$

Next, consider the case of a single analog complex sinusoid of frequency f_1 and its sampled version:

$$x(t) = e^{2\pi j f_1 t}, \quad -\infty < t < \infty$$
$$x(nT) = e^{2\pi j f_1 nT} = e^{j\omega_1 n}, \quad -\infty < n < \infty$$

where $\omega_1 = 2\pi T f_1 = 2\pi f_1/f_s$. The spectrum of the analog signal $x(t)$ is the Fourier transform:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-2\pi j f t}\, dt = \int_{-\infty}^{\infty} e^{-2\pi j (f-f_1) t}\, dt = \delta(f - f_1)$$

Therefore, $X(f)$ consists of a single sharp *spectral line* at $f = f_1$. For a real sinusoid $x(t) = \cos(2\pi f_1 t)$, we would get *two* half-height lines at $f = \pm f_1$. Indeed, the Fourier transform of the cosine is:

$$\cos(2\pi f_1 t) = \frac{1}{2} e^{2\pi j f_1 t} + \frac{1}{2} e^{-2\pi j f_1 t} \longrightarrow \frac{1}{2}\delta(f - f_1) + \frac{1}{2}\delta(f + f_1)$$

Assuming that f_1 lies within the Nyquist interval, that is, $|f_1| \leq f_s/2$, we determine the spectrum of the signal $x(nT)$ for $|f| \leq f_s/2$:

$$X(\omega) = \widehat{X}(f) = \frac{1}{T}X(f) = \frac{1}{T}\delta(f - f_1)$$

Using the delta function property, $|a|\delta(ax) = \delta(x)$, we can express the spectrum in terms of the digital frequency $\omega = 2\pi f/f_s = 2\pi T f$, as follows:

$$2\pi\delta(\omega - \omega_1) = \frac{1}{T}\, 2\pi T\, \delta(2\pi T f - 2\pi T f_1) = \frac{1}{T}\delta(f - f_1)$$

Therefore, the spectrum of the sampled signal will be, over the Nyquist interval:

$$X(\omega) = 2\pi\delta(\omega - \omega_1), \qquad -\pi \leq \omega \leq \pi$$

Outside the Nyquist interval, the spectral line is replicated at multiples of 2π, that is, $2\pi\delta(\omega - \omega_1 - 2\pi m)$. The inverse DTFT generates the same sampled sinusoid:

$$x(nT) = \int_{-\pi}^{\pi} X(\omega)e^{j\omega n}\,\frac{d\omega}{2\pi} = \int_{-\pi}^{\pi} 2\pi\delta(\omega - \omega_1)e^{j\omega n}\,\frac{d\omega}{2\pi} = e^{j\omega_1 n}$$

The windowed sinusoid consists of the L samples:

$$x_L(n) = e^{j\omega_1 n}, \qquad n = 0, 1, \ldots, L-1$$

Its spectrum is obtained by the frequency convolution property:

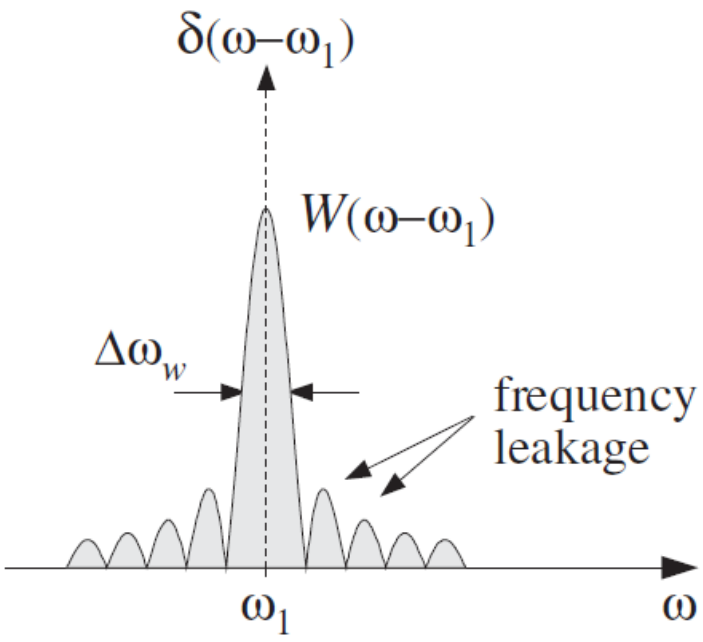
$$X_L(\omega) = \int_{-\pi}^{\pi} X(\omega')W(\omega - \omega')\,\frac{d\omega'}{2\pi} = \int_{-\pi}^{\pi} 2\pi\,\delta(\omega' - \omega_1)W(\omega - \omega')\,\frac{d\omega'}{2\pi}$$

Because of the delta function $\delta(\omega' - \omega_1)$ in the integrand, we obtain:

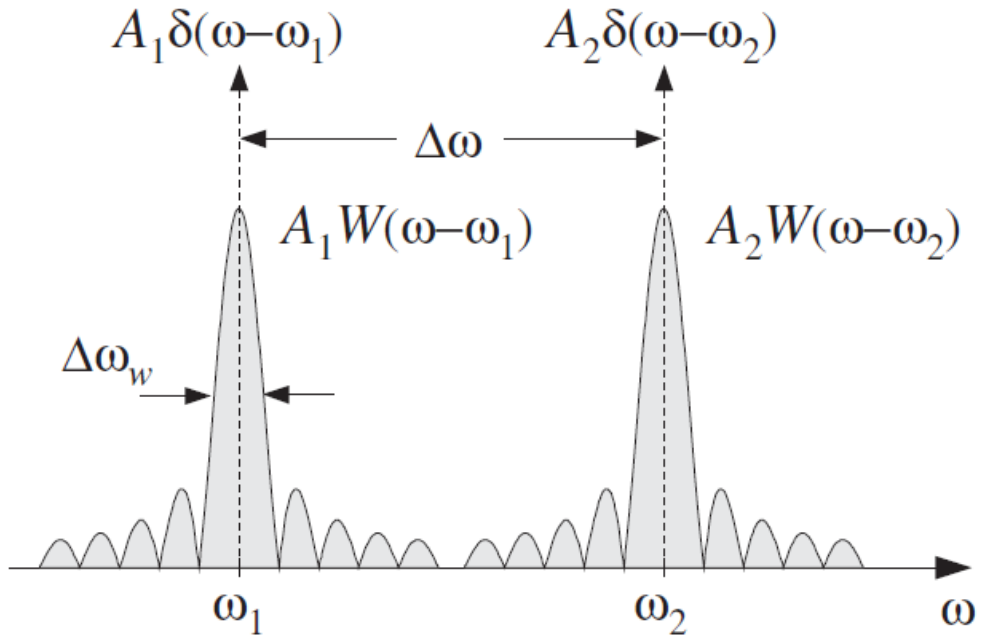
$$X_L(\omega) = W(\omega - \omega_1)$$

This is the translation of $W(\omega)$ centered about ω_1, as shown below. Thus, the windowing process has the effect of *smearing* the sharp spectral line $\delta(\omega - \omega_1)$ at ω_1 and replacing it by $W(\omega - \omega_1)$.

one sinusoid



two sinusoids



The *resolvability* condition that the two sinusoids appear as two distinct ones is that their frequency separation, $\Delta f = f_2 - f_1$, be *greater* than the mainlobe width:

$$\boxed{\Delta f \geq \Delta f_w = \frac{f_s}{L}} \qquad \text{(frequency resolution)}$$

or, in radians per sample:

$$\boxed{\Delta\omega \geq \Delta\omega_w = \frac{2\pi}{L}}$$

These equations can be rewritten to give the *minimum number* of samples required to achieve a desired frequency resolution Δf. The smaller the desired separation, the longer the data record:

$$\boxed{L \geq \frac{f_s}{\Delta f} = \frac{2\pi}{\Delta\omega}}$$

The mainlobe width of $W(\omega)$ determines the amount of achievable frequency resolution. The sidelobes, on the other hand, determine the amount of frequency leakage and are undesirable artifacts of the windowing process. They must be suppressed as much as possible because they may be confused with the mainlobes of *weaker* sinusoids that might be present.

Hamming window

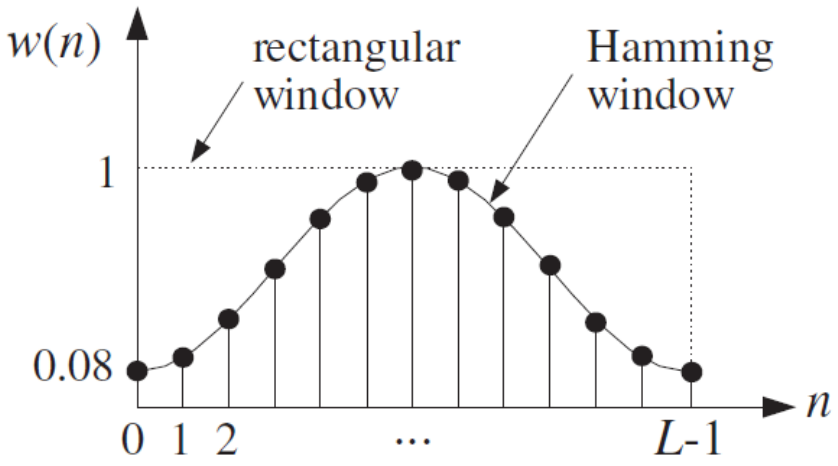
The standard technique for suppressing the sidelobes is to use a *non-rectangular window*—a window that cuts off to zero less sharply and more gradually than the rectangular one. There are literally dozens of possible shapes for such windows, such as trapezoidal, triangular, Gaussian, raised cosine, and many others.

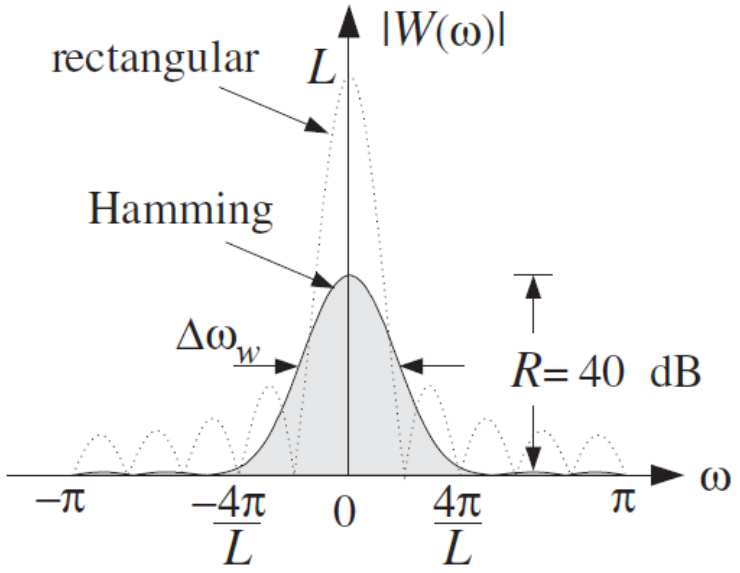
One of the simplest and most widely used window is the *Hamming window*. It provides a suppression of the sidelobes by at least 40 dB. Another one that allows the user to control the desired amount of sidelobe suppression is the *Kaiser window*. The Hamming window is a raised-cosine type of window defined as follows:

$$w(n) = \begin{cases} 0.54 - 0.46 \cos\left(\dfrac{2\pi n}{L-1}\right), & \text{if } 0 \le n \le L-1 \\ 0, & \text{otherwise} \end{cases}$$

Hamming window

At its center, $n = (L-1)/2$, the value of $w(n)$ is $0.54 + 0.46 = 1$, and at its endpoints, $n = 0$ and $n = L-1$, its value is $0.54 - 0.46 = 0.08$. Because of the gradual transition to zero, the high frequencies that are introduced by the windowing process are deemphasized. Its magnitude spectrum $|W(\omega)|$ is shown below. The sidelobes are still present, but are barely visible because they are suppressed relative to the mainlobe by $R = 40$ dB.





Hamming window

The main tradeoff in using any type of non-rectangular window is that its mainlobe becomes *wider* and shorter, thus, reducing the frequency resolution capability of the windowed spectrum. For any type of window, the effective width of the mainlobe is still *inversely proportional* to the window length:

$$\Delta f_w = c\,\frac{f_s}{L} = c\,\frac{1}{T_L}$$

or, in radians per sample:

$$\Delta \omega_w = c\,\frac{2\pi}{L}$$

where the constant c, known as the "broadening factor", depends on the window used and is always $c \geq 1$. The rectangular window has the *narrowest* width, corresponding to $c = 1$. The Hamming window has approximately $c = 2$, that is, its mainlobe is twice as wide as the rectangular one. The Kaiser window has variable c that depends on the prescribed amount of relative sidelobe level R.

Given a finite data record of L samples, $x(n)$, $n = 0, 1, \ldots, L-1$, the windowed signal is defined as follows, for the Hamming window, for $n = 0, 1, \ldots, L-1$,

$$x_L(n) = w(n)x(n) = \left[0.54 - 0.46\cos\left(\frac{2\pi n}{L-1}\right)\right] x(n)$$

If $x(n)$ consists of a linear combination of sinusoids, then each sharp spectral line $\delta(\omega - \omega_i)$ of $x(n)$ will be replaced by the Hamming window spectrum $W(\omega - \omega_i)$. The frequency resolution depends now on the width of the Hamming window Δf_w. It follows that the minimum resolvable frequency difference will be:

$$\boxed{\Delta f \geq \Delta f_w = c\,\frac{f_s}{L} = c\,\frac{1}{T_L}}$$

This implies that the minimum data record required to achieve a given value of Δf is c-times longer than that of a rectangular window:

$$\boxed{L \geq c\,\frac{f_s}{\Delta f} = c\,\frac{2\pi}{\Delta\omega}}$$

Kaiser window for spectral analysis

We saw that one of the main issues in spectral analysis was the *tradeoff* between frequency resolution and leakage. The more one tries to suppress the sidelobes, the wider the mainlobe of the window becomes, reducing the amount of achievable resolution.

The Hamming window provides about 40 dB sidelobe suppression at the expense of doubling the mainlobe width of the rectangular window. We recall that the mainlobe width Δf_w of a window depends inversely on the data record length L:

$$\Delta f_w = \frac{cf_s}{L} \quad \Leftrightarrow \quad L = \frac{cf_s}{\Delta f_w}$$

where the factor c depends on the window used. The more the sidelobe suppression, the larger the factor c. Thus, to maintain a certain required value for the resolution width Δf_w, one must increase the data length L commensurately with c.

Most windows have fixed values for the amount of sidelobe suppression and broadening factor c. Adjustable windows, like the Kaiser window, have a variable sidelobe level R that can be chosen as the application requires.

Kaiser window for spectral analysis

Kaiser and Schafer have developed simple design equations for the use of the Kaiser window in spectral analysis. Given a desired relative sidelobe level R in dB and a desired amount of resolution Δf_w, the design equations determine the length L and shape parameter α of the window. More details are to be found in project-4.

Once the window parameters $\{L, \alpha\}$ have been determined, the window may be calculated by:

$$w(n) = \frac{I_0\left(\alpha\sqrt{1-(n-M)^2/M^2}\right)}{I_0(\alpha)} \qquad n = 0, 1, \ldots, L-1$$

where $M = (L-1)/2$, and then applied to a length-L data record by

$$x_L(n) = w(n)x(n), \qquad n = 0, 1, \ldots, L-1$$

For filter design, a slightly different set of design equations are used to determine the parameters L, α, to be discussed in I2SP/Ch.10.

Kaiser window for spectral analysis

The modified Bessel function of first kind and zeroth order is defined as,

$$I_0(x) = \sum_{k=0}^{\infty} \left[\frac{x^k}{k! 2^k} \right]^2$$

It can be valuated at a vector of x's by the built-function **besseli**,

```
F = besseli(0,x);
```

The Kaiser window can be evaluated using **besseli**, or, using the built-in function **kaiser**, which requires as inputs the parameters L, α, and produces the L window samples, $w(n)$, $n = 0, 1, \ldots, L-1$,

```
w = kaiser(L, alpha);     % length-L column vector
```

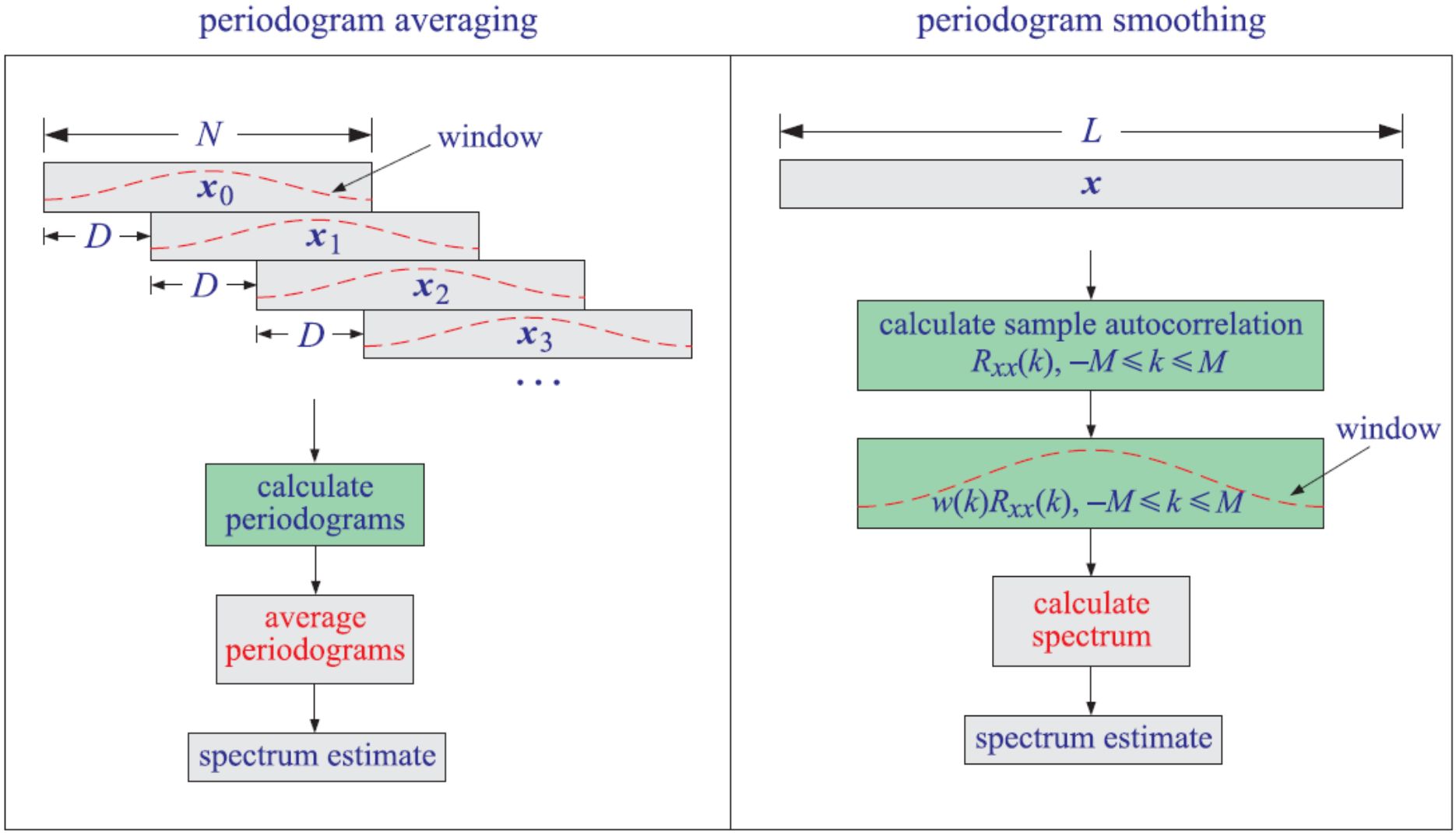
Power Spectrum Estimation

For random signals, such as sinusoids in noise, one must also deal with the *statistical reliability* of the computed spectra. In I2SP / Appendix and in project-4, we discuss the *periodogram averaging* method which may be used to reduce the statistical *variability* of the spectrum estimate. Another method of power spectrum estimation is *periodogram smoothing*, which is also explored in project-4.

Both periodogram methods, require the total length L to be large. In some applications, this may be impossible to achieve either because we cannot collect more data, or because beyond a certain length L, the signal will no longer remain stationary.

Parametric spectrum estimation methods, such as those based on linear prediction, maximum likelihood, and eigenvector techniques, offer the possibility of obtaining high-resolution spectrum estimates based on short data records.

periodogram improvement methods



Random Signal Concepts

The *autocorrelation function* of a zero-mean random signal is defined as the correlation between two samples $x(n)$ and $x(n+k)$ separated by a time lag k. It is a measure of the dependence of successive samples on the previous ones:

$$\boxed{R_{xx}(k) = E\big[x(n+k)x(n)\big]} \quad \text{(autocorrelation function)}$$

For stationary signals, $R_{xx}(k)$ depends only on the relative time-lag k, and not on the absolute time n. Note that $R_{xx}(k)$ is a double-sided sequence and, as a consequence of stationarity, it is symmetric in k, that is, $R_{xx}(-k) = R_{xx}(k)$.

Random Signal Concepts

The *power spectrum* of the random signal $x(n)$ is defined as the DTFT of its autocorrelation function $R_{xx}(k)$. It represents the frequency content of the random signal $x(n)$ in an average sense:

$$S_{xx}(\omega) = \sum_{k=-\infty}^{\infty} R_{xx}(k) e^{-j\omega k} \qquad \text{(power spectrum)}$$

where $\omega = 2\pi f / f_s$ is the digital frequency in radians per sample. The inverse DTFT relationship expresses $R_{xx}(k)$ in terms of $S_{xx}(\omega)$:

$$R_{xx}(k) = E\big[x(n+k)x(n)\big] = \int_{-\pi}^{\pi} S_{xx}(\omega) e^{j\omega k} \frac{d\omega}{2\pi}$$

In particular, setting $k = 0$, we obtain the *average power*, or variance, of the signal $x(n)$:

$$\sigma_x^2 = R_{xx}(0) = E\left[x(n)^2\right] = \int_{-\pi}^{\pi} S_{xx}(\omega)\,\frac{d\omega}{2\pi} = \int_{-f_s/2}^{f_s/2} S_{xx}(f)\,\frac{df}{f_s}$$

$$\boxed{S_{xx}(f) = \sum_{k=-\infty}^{\infty} R_{xx}(k)e^{-2\pi jfk/f_s}}$$

The quantity $S_{xx}(f)/f_s$ represents the *power per unit frequency* interval. Hence, the name "power spectrum" or "power spectral density" (psd). It describes how the signal's power is *distributed* among different frequencies. Its integral over the Nyquist interval gives the *total* power in the signal.

Often it is more convenient to work with the z-transform of the auto-correlation and replace $z = e^{j\omega} = e^{2\pi jf/f_s}$ to obtain the power spectrum $S_{xx}(\omega)$ or $S_{xx}(f)$:

$$\boxed{S_{xx}(z) = \sum_{k=-\infty}^{\infty} R_{xx}(k)z^{-k}}$$

Random Signal Concepts

White noise has a delta-function autocorrelation and a *flat* spectrum, as shown below.

Because (by definition) successive signal samples are independent of each other, the autocorrelation function will factor for $k \neq 0$ into the product of the means which are zero:

$$R_{xx}(k) = E\big[x(n+k)x(n\big] = E[x(n+k)] \cdot E[x(n)] = 0$$

whereas for $k = 0$, we get the variance

$$R_{xx}(0) = E\big[x(n)^2\big] = \sigma_x^2$$

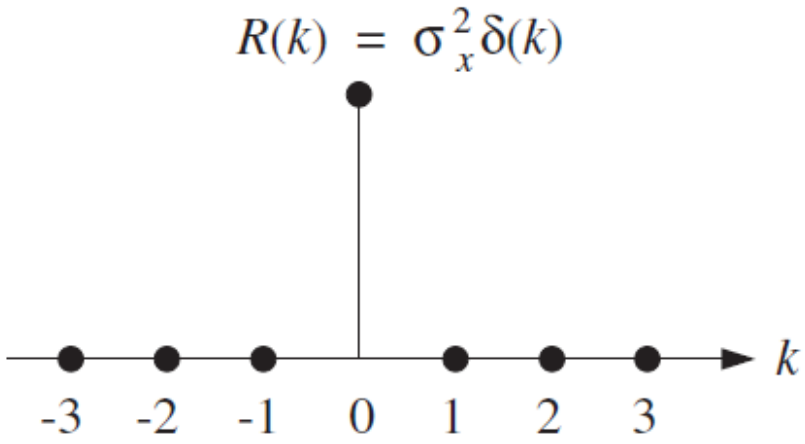
Combining them into a single equation, we have:

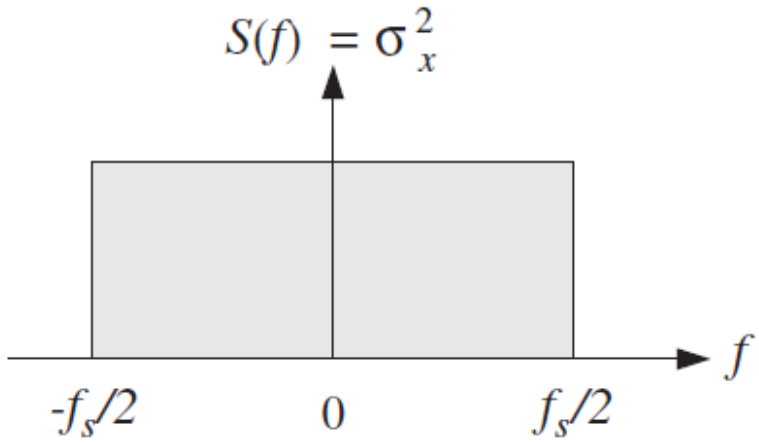
$$R_{xx}(k) = \sigma_x^2 \delta(k) \qquad \text{(white noise autocorrelation)}$$

Only the $k = 0$ term survives the sum giving the flat spectral density (over the Nyquist interval):

$$S_{xx}(f) = \sigma_x^2, \qquad \text{for } -\frac{f_s}{2} \leq f \leq \frac{f_s}{2} \qquad \text{(white noise spectrum)}$$

White noise has a delta-function autocorrelation and a *flat* spectrum





Sample Autocorrelation

Given a length-N block of signal samples $x(n)$, $n = 0, 1, \ldots, N-1$, one can compute an *estimate* of the statistical quantity $R_{xx}(k)$ by the so-called *sample autocorrelation* obtained by replacing the statistical average by the time average:

$$\widehat{R}_{xx}(k) = \frac{1}{N} \sum_{n=0}^{N-1-k} x(n+k)x(n) \qquad \text{(sample autocorrelation)}$$

for $k = 0, 1, \ldots, N-1$. The negative tail can be defined using the symmetry property $\widehat{R}_{xx}(-k) = \widehat{R}_{xx}(k)$, so that we can write, for $|k| \leq N-1$,

$$\widehat{R}_{xx}(k) = \frac{1}{N} \sum_{n=0}^{N-1-|k|} x(n+|k|)x(n) \qquad \text{(sample autocorrelation)}$$

The rule of thumb is that only about the first 5–10% of the lags are statistically reliable, that is, $0 \leq k \leq N/10$.

Sample Autocorrelation

The built-in function **xcorr** computes $\hat{R}_{xx}(k)$, for $-M \leq k \leq M$, with any $M \leq N-1$, with usage,

```
Rxx = xcorr(x,M);    % with M <= length(x)-1

% examples

  x = 0:5;
  Rxx = xcorr(x,3);
% Rxx = [14   26   40   55   40   26   14]

  Rxx = xcorr(x,5);
% Rxx = [0   5   14   26   40   55   40   26   14    5   0]
```

with the outputs listed in the order:

$$\left[\hat{R}_{xx}(-M)\,,\ \cdots\,,\ \hat{R}_{xx}(-1)\,,\ \hat{R}_{xx}(0)\,,\ \hat{R}_{xx}(1)\,,\ \cdots\,,\ \hat{R}_{xx}(M)\right]$$

Periodogram and Its Improvements

It can be shown that for wide-sense stationary signals, $\widehat{R}_{xx}(k)$ is a good estimate of $R_{xx}(k)$, converging to the latter for large N (in the mean-square sense):

$$\widehat{R}_{xx}(k) \to R_{xx}(k) \quad \text{as} \quad N \to \infty$$

The DTFT of $\widehat{R}_{xx}(k)$ is called the *periodogram spectrum* and can be thought of as an *estimate* of the true power spectrum $S_{xx}(\omega)$:

$$\widehat{S}_{xx}(\omega) = \sum_{k=-(N-1)}^{N-1} \widehat{R}_{xx}(k) e^{-j\omega k}$$

Using the definition of $\widehat{R}_{xx}(k)$, and rearranging summations, we can express the periodogram in the alternative way:

$$\boxed{\widehat{S}_{xx}(\omega) = \frac{1}{N} |X_N(\omega)|^2} \qquad \text{(periodogram spectrum)}$$

where $X_N(\omega)$ is the DTFT of the length-N data block $x(n)$, which can be computed efficiently using FFTs, or, **freqz**,

$$X_N(\omega) = \sum_{n=0}^{N-1} x(n) e^{-j\omega n}$$

Periodogram and Its Improvements

It can be shown that for wide-sense stationary random signals the mean of the periodogram converges to the true power spectrum $S_{xx}(\omega)$ in the limit of large N, that is,

$$\boxed{S_{xx}(\omega) = \lim_{N\to\infty} E[\widehat{S}_{xx}(\omega)] = \lim_{N\to\infty} E\big[\frac{1}{N}\,|X_N(\omega)|^2\big]}$$

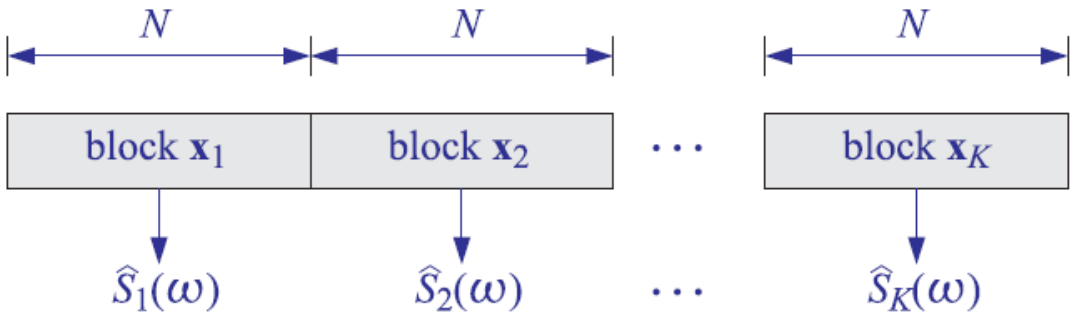
Unfortunately, the periodogram is not a good estimator of the power spectrum. It does not approximate $S_{xx}(\omega)$ well, even in the limit of large N. That is, even though the mean of the periodogram tends to $S_{xx}(\omega)$, the periodogram itself $\widehat{S}_{xx}(\omega)$ does not, in the sense that it is not a *consistent estimator* of $S_{xx}(\omega)$, with its variance not converging to zero for large data blocks N.

The subject of *classical spectral analysis* is essentially the subject of fixing the periodogram to provide a good estimate of the power spectrum.

There are two basic techniques that improve the periodogram: periodogram *averaging* and periodogram *smoothing*. The averaging method tries to emulate the ensemble averaging operation $E[\,]$. In its simplest form, it consists of dividing the signal into contiguous blocks, computing the ordinary periodogram of each block, and then averaging the computed periodograms.

The method is depicted below, where there are K blocks, each of length N, so that the total length of the data record is $L = KN$. The signal is required to remain stationary at least over the length L. The block size N must be chosen to provide sufficient *frequency resolution*. Denoting the ith block by $x_i(n)$, $n = 0, 1, \ldots, N-1$, we compute its ordinary periodogram:

$$\widehat{S}_i(\omega) = \frac{1}{N} |X_i(\omega)|^2, \qquad i = 1, 2, \ldots, K$$



where $X_i(\omega)$ is its DTFT:

$$X_i(\omega) = \sum_{n=0}^{N-1} x_i(n) e^{-j\omega n}$$

and then average the K periodograms:

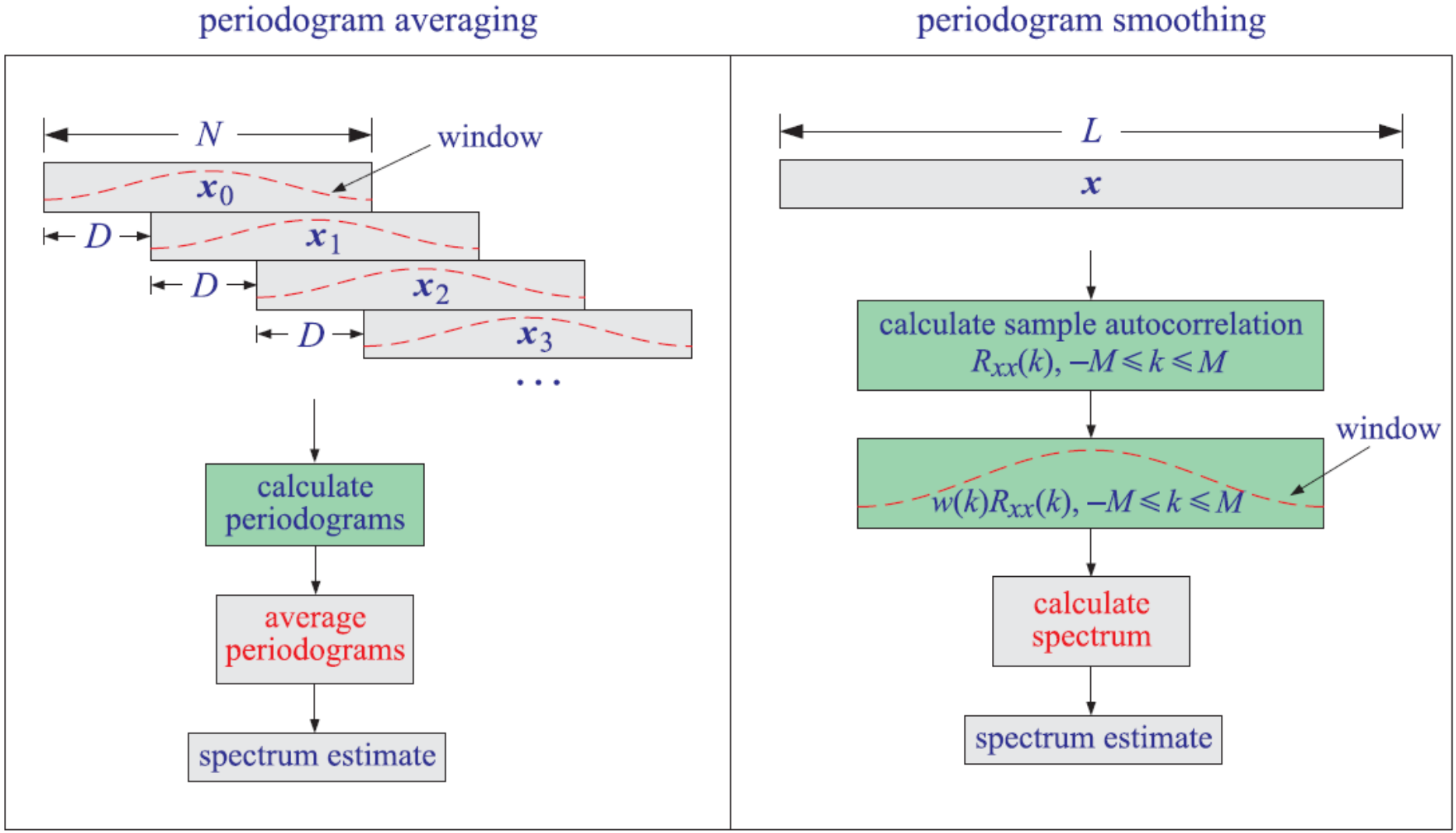
$$\begin{aligned} \widehat{S}(\omega) &= \frac{1}{K} \left[\widehat{S}_1(\omega) + \widehat{S}_2(\omega) + \cdots + \widehat{S}_K(\omega) \right] \\ &= \frac{1}{KN} \left[|X_1(\omega)|^2 + |X_2(\omega)|^2 + \cdots + |X_K(\omega)|^2 \right] \end{aligned}$$

Periodogram and Its Improvements

It can be shown that $\widehat{S}(\omega)$ is a good estimator of $S_{xx}(\omega)$, with the (mean-square) error between the two decreasing like $1/K$, for large K. The periodogram smoothing method has similar performance, and is described in more detail in project-4.

There are two basic shortcomings with such classical spectral analysis methods: One is that to achieve high statistical reliability, a large value of K must be used, which implies a long total signal length $L = KN$. Such long blocks may not be possible to obtain in certain applications. The second is that even if a long data record could be measured, it may not be usable because the signal may not remain stationary over such long periods of time, as for example, in speech.

Periodogram and Its Improvements



Filtering of Random Signals

In designing filters to remove noise, it is necessary to know the effect of filtering on the autocorrelation function and on the power spectrum of a random signal.

Suppose the input to a *strictly stable* filter $H(z)$ with impulse response $h(n)$ is a wide-sense stationary signal $x(n)$. Then, the corresponding output $y(n)$ will also be a wide-sense stationary random signal:

$$y(n) = \sum_m h(m)x(n-m) \qquad x(n) \longrightarrow \boxed{H(z)} \longrightarrow y(n)$$

It can be shown that the power spectrum of the output is related to that of the input by:

$$\boxed{S_{yy}(\omega) = |H(\omega)|^2 S_{xx}(\omega)}$$

Thus, the input spectrum is reshaped by the filter spectrum. A simple way to justify this result is in terms of periodograms. The filtering equation in the z-domain is $Y(z) = H(z)X(z)$, and in the frequency domain $Y(\omega) = H(\omega)X(\omega)$. It follows that the output periodogram will be related to the input periodogram by a similar equation as,

$$\frac{1}{N}|Y(\omega)|^2 = |H(\omega)|^2 \cdot \frac{1}{N}|X(\omega)|^2$$

Filtering of Random Signals

Applying this result to the special case of a *white noise input* with a flat spectral density $S_{xx}(\omega) = \sigma_x^2$ gives

$$\boxed{S_{yy}(\omega) = |H(\omega)|^2 \sigma_x^2}$$

Similarly, in z-transform notation,

$$\boxed{S_{yy}(z) = H(z)H(z^{-1})\sigma_x^2}$$

where we replaced $H(\omega) = H(z)$ and $H(\omega)^* = H(z^{-1})$, the latter following from the fact that $h(n)$ is real-valued. Indeed, with $z = e^{j\omega}$ and $z^{-1} = z^* = e^{-j\omega}$, we have:

$$H(\omega)^* = \Big(\sum_n h(n)e^{-j\omega n}\Big)^* = \sum_n h(n)e^{j\omega n} = H(z^{-1})$$

The filtered noise $y(n)$ is no longer white-noise. Its power spectrum acquires the shape of the filter's spectrum. Its autocorrelation function is no longer a delta function.

Filtering of Random Signals

A measure of whether the filter attenuates or magnifies the input noise is given by the variance of the output σ_y^2:

$$\sigma_y^2 = \int_{-\pi}^{\pi} S_{yy}(\omega)\,\frac{d\omega}{2\pi} = \sigma_x^2 \int_{-\pi}^{\pi} |H(\omega)|^2\,\frac{d\omega}{2\pi}$$

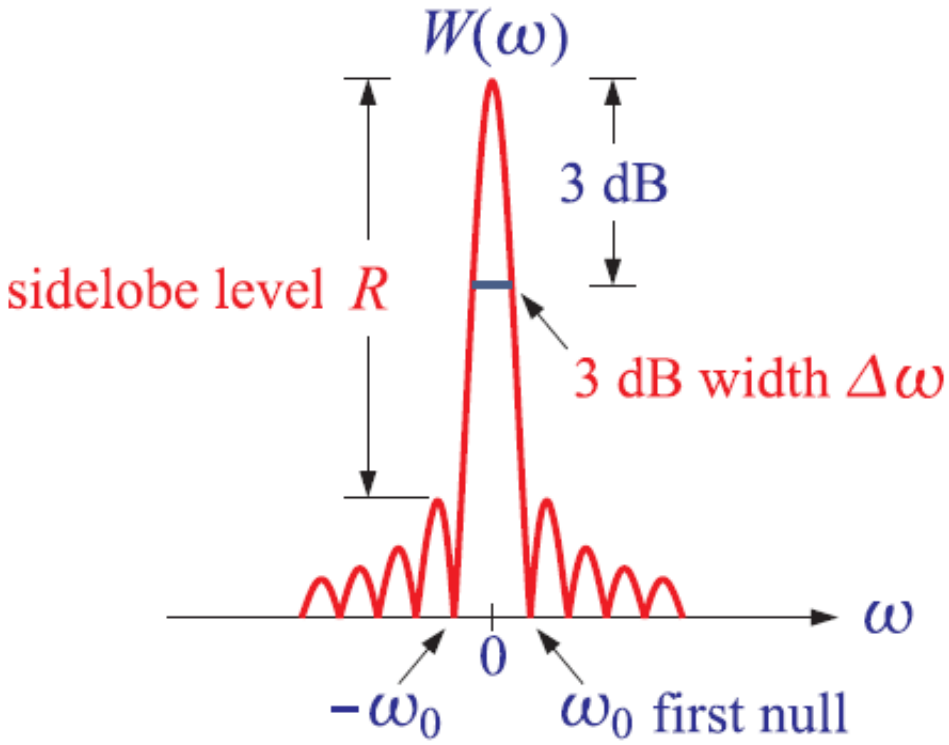
which can be written in the form:

$$\boxed{\text{NRR} = \frac{\sigma_y^2}{\sigma_x^2} = \int_{-\pi}^{\pi} |H(\omega)|^2\,\frac{d\omega}{2\pi} = \sum_n h^2(n)}$$

(noise reduction ratio)

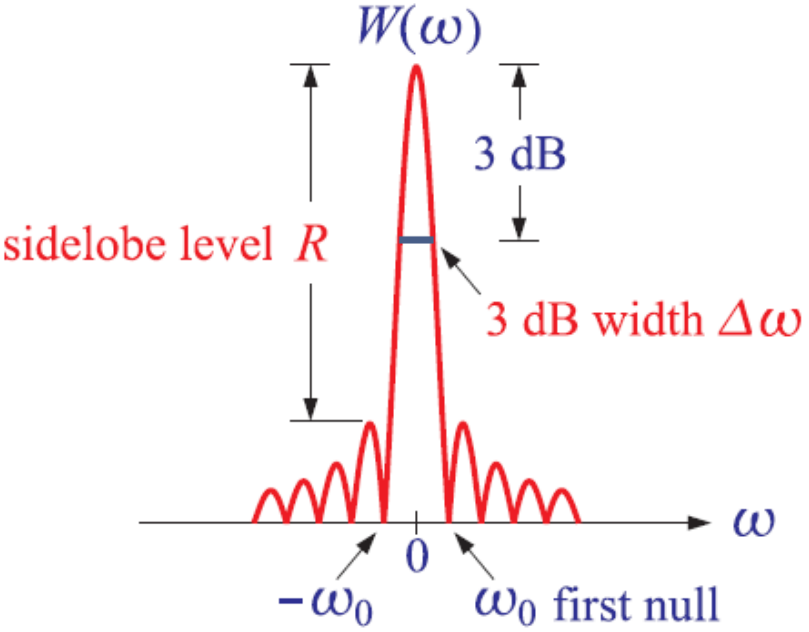
A necessary assumption for the derivation of these results is that the filter $h(n)$ be strictly stable. The stability of $h(n)$ is required to ensure that the stationary input signal $x(n)$ will generate, after the filter transients die out, a stationary output signal $y(n)$.

further notes on windows and project 4



1. Rectangular
2. Hamming
3. Kaiser
4. DPSS
5. Chebyshev

further notes on windows and project 4



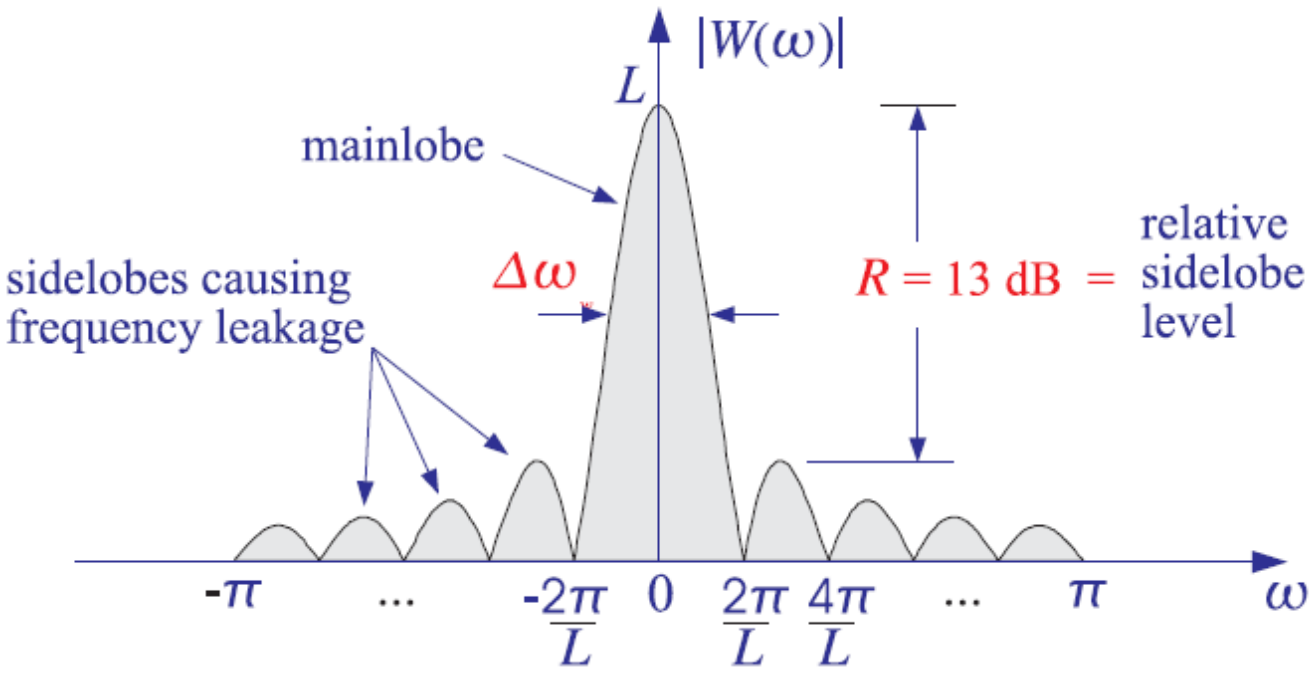
Rectangular window DTFT:

$$W(\omega) = \frac{1 - e^{-jL\omega}}{1 - e^{-j\omega}} = \frac{\sin(\omega L/2)}{\sin(\omega/2)}\, e^{-j\omega(L-1)/2}$$

$$\left|\frac{W(\omega)}{W(0)}\right| = \left|\frac{\sin(\omega L/2)}{L\sin(\omega/2)}\right| = \left|\frac{\sin(\pi x)}{L\sin(\pi x/L)}\right|, \quad \omega = \frac{2\pi x}{L}$$

for $L > 10$,

$$F(x) = \frac{\sin(\pi x)}{L\sin(\pi x/L)} \approx \frac{\sin(\pi x)}{L \cdot \pi x/L} = \frac{\sin(\pi x)}{\pi x} = \mathrm{sinc}(x)$$


$|W(\omega)|$
L
mainlobe
sidelobes causing
frequency leakage
$\Delta\omega$
$R = 13$ dB = relative sidelobe level
$-\pi$
...
$-\frac{2\pi}{L}$
0
$\frac{2\pi}{L}$
$\frac{4\pi}{L}$
...
π
ω

3-dB frequency is obtained by solving the condition,

$$\left|\frac{W(\omega)}{W(0)}\right|^2 = \frac{1}{2} \quad \Rightarrow \quad F^2(x) = \frac{1}{2} \quad \Rightarrow \quad F(x) = \frac{1}{\sqrt{2}}$$

```
% solved with MATLAB's fzero function
  x3 = fzero(@(x) sinc(x)-1/sqrt(2), 0.4);
% x3 = 0.443,  2*x3 = 0.886
```

3-dB frequency and 3-dB width,

$$\omega_3 = \frac{2\pi x_3}{L} = 0.443\,\frac{2\pi}{L} \quad \Rightarrow \quad \Delta\omega_{3\text{dB}} = 2\omega_3 = 0.886\,\frac{2\pi}{L}$$

sidelobe level is determined by finding local maximum next to mainlobe,

```
% solved with MATLAB's minimization function
  xside = fminbnd(@(x) -abs(sinc(x)), 1,2);
% xside = 1.4303
% then, evaluate F(xside) in dB
  R  = -20*log10(abs(sinc(xside)));
% R = 13.2615 dB
```

Hamming window DTFT with odd length, $L = 2M + 1$,

$$w(n) = 0.54 - 0.46 \cos\left(\frac{\pi n}{M}\right), \quad n = 0, 1, \ldots, 2M$$

$$w(n) = 0.54 - 0.23\, e^{j\pi n/M} - 0.23\, e^{-j\pi n/M}$$

$$W(\omega) = 0.54\, W_{\text{rect}}(\omega) - 0.23\, W_{\text{rect}}\left(\omega - \frac{\pi}{M}\right) - 0.23\, W_{\text{rect}}\left(\omega + \frac{\pi}{M}\right)$$

$$W_{\text{rect}}(\omega) = \frac{1 - e^{-jL\omega}}{1 - e^{-j\omega}} = \frac{\sin(\omega L/2)}{\sin(\omega/2)}\, e^{-j\omega M}, \quad M = \frac{L-1}{2}$$

with approximate broadening factor and sidelobe level,

$$\boxed{\begin{aligned} \Delta\omega_{3\text{dB}} &= 0.886\, \frac{2\pi b}{L} \\ b &= 1.47 + \frac{0.98}{L} \approx 1.5 \\ R &\approx 40 \text{ dB} \end{aligned}}$$

further notes on windows and project 4

The symmetric Kaiser window is based on the following Fourier transform pair in continuous time,

$$2\tau_0 \frac{\sinh\left[\sqrt{\tau_0^2\Omega_0^2 - \tau_0^2\Omega^2}\right]}{\sqrt{\tau_0^2\Omega_0^2 - \tau_0^2\Omega^2}} = \int_{-\tau_0}^{\tau_0} I_0\left[\tau_0\Omega_0\sqrt{1 - t^2/\tau_0^2}\right] e^{j\Omega t}\, dt$$

defining the time-bandwidth product, $\alpha = \tau_0\Omega_0$, and sampling at some rate $f_s = 1/T_s$, assuming an integer number of samples within τ_0,

$$\begin{aligned}
&\tau_0 = MT_s \\
&t = kT_s\,, \quad -M \le k \le M \\
&\omega = \Omega T_s = \text{digital frequency} \\
&\tau_0\Omega = MT_s\Omega = M\omega
\end{aligned}$$

then, the sampled and normalized symmetric window becomes,

$$w(k) = \frac{I_0\left[\alpha\sqrt{1 - k^2/M^2}\right]}{I_0(\alpha)}\,, \quad -M \le k \le M$$

further notes on windows and project 4

from the sampling theorem, the DTFT of $w(k)$ will be the above continuous-time transform scaled by T_s, with all its replicas at multiples of f_s added. For small enough T_s, the replicas can be ignored approximately, resulting in the scaled and normalized spectrum,

$$W(\omega) = 2M \, \frac{\sinh\left[\sqrt{\alpha^2 - M^2\omega^2}\right]}{I_0(\alpha)\sqrt{\alpha^2 - M^2\omega^2}}, \quad \omega = \frac{2\pi f}{f_s} = \Omega T_s$$

$$\frac{W(\omega)}{W(0)} = \frac{\alpha}{\sinh(\alpha)} \cdot \frac{\sinh\left[\sqrt{\alpha^2 - M^2\omega^2}\right]}{\sqrt{\alpha^2 - M^2\omega^2}}$$

the causal version of the window is obtained by delaying the symmetric one by M samples,

$$w(n) = \frac{I_0\left[\alpha\sqrt{1 - (n-M)^2/M^2}\right]}{I_0(\alpha)}, \quad n = 0, 1, \ldots, 2M$$

further notes on windows and project 4

the 3-dB width and sidelobe levels can be obtained from,

$$\frac{W(\omega)}{W(0)} = \frac{\alpha}{\sinh(\alpha)} \cdot \frac{\sinh\left[\sqrt{\alpha^2 - M^2\omega^2}\right]}{\sqrt{\alpha^2 - M^2\omega^2}}$$

see EWA/Ch.20 for computational details with MATLAB. The resulting broadening factor and the exact relationship between α and R are:

$$\boxed{\begin{aligned} b &= 0.0124\,R + 1.0221 \\ R &= 13.26 + 20\,\log_{10}\left[\frac{\sinh(\alpha)}{\alpha}\right] \end{aligned}}$$

with R in dB – the exact relationship can be solved for α by the Kaiser-Schafer approximation:

$$\alpha = \begin{cases} 0, & R \le 13.26 \\ 0.76609\,(R - 13.26)^{0.4} + 0.09834\,(R - 13.26), & 13.26 < R \le 60 \\ 0.12438\,(R + 6.3), & 60 < R \le 120 \end{cases}$$

The prolate window maximizes the power that resides within the mainlobe of the window. Assuming the mainlobe is within the range of digital frequencies $[-\omega_c, \omega_c]$, with ω_c to be determined from α and L, the minimization criterion is,

$$J = \frac{\displaystyle\int_{-\omega_c}^{\omega_c} |W(\omega)|^2 d\omega/2\pi}{\displaystyle\int_{-\pi}^{\pi} |W(\omega)|^2 d\omega/2\pi} = \max$$

$$W(\omega) = \sum_{n=0}^{L-1} w(n) e^{-j\omega n}$$

$$\mathbf{w} = \big[w(0), w(1), \ldots, w(L-1)\big]^T$$

$$W_c = \frac{\omega_c}{2\pi}$$

$$J = \frac{\mathbf{w}^T A \mathbf{w}}{\mathbf{w}^T \mathbf{w}} = \max \quad \text{(Rayleigh quotient)}$$

$$A_{nm} = \frac{\sin\big(2\pi W_c(n-m)\big)}{\pi(n-m)}, \quad n, m = 0, 1, \ldots, L-1$$

prolate matrix

The maximization of the Rayleigh quotient is realized by the maximum eigenvector of the prolate matrix A, that is, the eigenvector belonging to the maximum eigenvalue, say, λ_0:

$$A\mathbf{w} = \lambda_0 \mathbf{w}$$

The prolate matrix is notoriously ill-conditioned having approximately $2LW_c$ eigenvalues that are very near one, and the remaining eigenvalues decreasing rapidly to zero. The following table lists the eigenvalues in decreasing order for the case $L = 21$ and $W_c = 0.2$, so that $2LW_c = 8.4$, its condition number being, $\text{cond}(A) = 5.1063 \times 10^{16}$:

i	λ_i	i	λ_i
0	0.99999999998517786000	11	0.00131552671490021500
1	0.99999997955146270000	12	0.00007986915605618046
2	0.99999987170540139000	13	0.00000365494381482577
3	0.99999517388508363000	14	0.00000012731149204486
4	0.99987947149714795000	15	0.00000000336154097643
5	0.99792457099956200000	16	0.00000000006621668668
6	0.97588122145542644000	17	0.00000000000094327944
7	0.83446090480119717000	18	0.00000000000000920186
8	0.45591142240913063000	19	0.00000000000000004034
9	0.11887181858959120000	20	0.00000000000000001958
10	0.01567636516215985600		

These were generated by the following MATLAB code:

```
L = 21; Wc = 0.2;
n = 0:L-1;
f = 2*Wc*sinc(2*Wc*n);
A = toeplitz(f,f);
lambda = svd(A);
```

The eigenvectors of the prolate matrix are referred to as the *discrete prolate spheroidal sequences* (DPSS).

the approximate relationship between W_c and the Kaiser α works well over the range $14 \leq R \leq 120$ dB:

$$W_c = \frac{0.95\alpha/\pi + 0.14}{L}$$

However, using the predicted Kaiser 3-dB bandwith is an alternative that also works well for the DPSS window, and can be used in project-4. The window samples can be computed using the built-in function **dpss**.

```
w = dpss(L, alpha/pi, 1);    % window w(n), Lx1 vector
w = w/max(w);                % normalize to unity maximum
```

Most windows have largest sidelobes near the main lobe. If a window is designed to achieve a minimum sidelobe attenuation of R dB, then typically R will be the attenuation of the sidelobes nearest to the mainlobe; the sidelobes further away will have attenuations higher than R.

Because of the tradeoff between mainlobe width and sidelobe attenuation, the extra attenuation of the furthest sidelobes will come at the expense of increased mainlobe width. If the attenuation of these sidelobes could be decreased (up to the level of the minimum R), then the mainlobe width would narrow.

It follows that for a given *minimum* desired sidelobe level R, the *narrowest* mainlobe width will be achieved by a window whose sidelobes are all *equal* to R. Conversely, for a given *maximum* desired mainlobe width, the *largest* sidelobe attenuation will be achieved by a window with equal sidelobe levels.

This "optimum" window is the Dolph-Chebyshev window, which is constructed with the help of Chebyshev polynomials. The mth Chebyshev polynomial $T_m(x)$ is:

$$T_m(x) = \cos\bigl(m \cos^{-1}(x)\bigr)$$

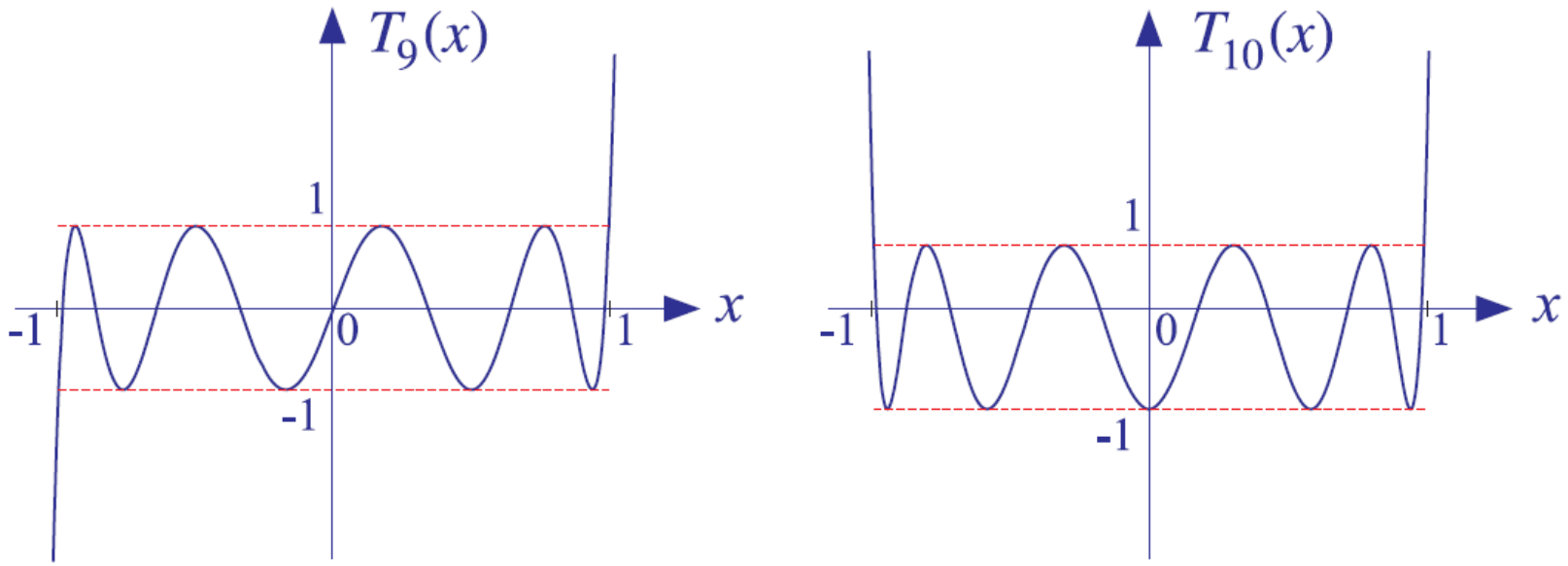
If $|x| > 1$, the inverse cosine $\cos^{-1}(x)$ becomes imaginary, and the expression can be rewritten in terms of hyperbolic cosines:

$$T_m(x) = \cosh\bigl(m \cosh^{-1}(x)\bigr)$$

Setting $x = \cos\theta$, or $\theta = \cos^{-1}(x)$, we see that $T_m(x) = \cos(m\theta)$. Using trigonometric identities, the quantity $\cos(m\theta)$ can always be expanded as a polynomial in powers of $\cos\theta$. The expansion coefficients are precisely the coefficients of the powers of x of the Chebyshev polynomial, e.g.,

$$
\begin{aligned}
\cos(0\theta) &= 1 && & T_0(x) &= 1 \\
\cos(1\theta) &= \cos\theta && & T_1(x) &= x \\
\cos(2\theta) &= 2\cos^2\theta - 1 && \Rightarrow \quad & T_2(x) &= 2x^2 - 1 \\
\cos(3\theta) &= 4\cos^3\theta - 3\cos\theta && & T_3(x) &= 4x^3 - 3x \\
\cos(4\theta) &= 8\cos^4\theta - 8\cos^2\theta + 1 && & T_4(x) &= 8x^4 - 8x^2 + 1
\end{aligned}
$$

For $|x| < 1$, the Chebyshev polynomial has equal ripples, whereas for $|x| > 1$, it increases like x^m. Moreover, $T_m(x)$ is even in x if m is even, and odd in x if m is odd. The figure below depicts the Chebyshev polynomials $T_9(x)$ and $T_{10}(x)$.



The Dolph-Chebyshev window is defined such that its sidelobes will correspond to a portion of the equi-ripple range $|x| \leq 1$ of the Chebyshev polynomial, whereas its mainlobe will correspond to a portion of the range $x > 1$.

For either even or odd L, the window spectrum $W(\omega)$ can be written in general as a polynomial of degree $L-1$ in the variable $u = \cos(\omega/2)$. Indeed, we have for the mth terms:

$$\cos(m\omega) = \cos\left(2m\frac{\omega}{2}\right) = T_{2m}(u)$$

$$\cos\big((m-1/2)\omega) = \cos\left((2m-1)\frac{\omega}{2}\right) = T_{2m-1}(u)$$

Thus in the odd case, the summation of such terms will result in a polynomial of maximal degree $2M = L-1$ in the variable u, and in the even case, it will result into a polynomial of degree $2M-1 = L-1$.

The DTFT of the Dolph-Chebyshev window is defined by the Chebyshev polynomial of degree $L-1$ in the scaled variable $x = x_0\cos(\omega/2)$, that is,

$$\boxed{W(\omega) = T_{L-1}(x), \quad x = x_0\cos\left(\frac{\omega}{2}\right)}$$

The relative sidelobe attenuation level in absolute units and in dB is defined in terms of the ratio of the mainlobe to the sidelobe heights:

$$R_a = \frac{W_{\text{main}}}{W_{\text{side}}}\,, \quad R = 20\log_{10}(R_a)\,, \quad R_a = 10^{R/20}$$

Because the mainlobe peak occurs at $\omega = 0$ or $x = x_0$, we will have $W_{\text{main}} = T_{L-1}(x_0)$, and because the sidelobe level is equal to the Chebyshev level within $|x| \leq 1$, we will have $W_{\text{side}} = 1$. Thus, we find:

$$\boxed{R_a = T_{L-1}(x_0) = \cosh\big((L-1)\cosh^{-1}(x_0)\big)}$$

which can be solved for x_0 in terms of R_a:

$$\boxed{x_0 = \cosh\left(\frac{\cosh^{-1}(R_a)}{L-1}\right)}$$

Once the scale factor x_0 is determined, the window samples $w(n)$ can be computed by constructing the z-transform of the DTFT from its zeros and then doing an inverse z-transform. The $L-1$ zeros of $T_{L-1}(x)$ are easily found to be:

$$T_{L-1}(x) = \cos\big((L-1)\cos^{-1}(x)\big) = 0 \quad \Rightarrow \quad x_i = \cos\left(\frac{(i-1/2)\pi}{L-1}\right)$$

for $i = 1, 2, \ldots, L-1$. Solving for the corresponding frequencies through $x_i = x_0 \cos(\omega_i/2)$, we find the DTFT zeros:

$$\omega_i = 2\cos^{-1}\left(\frac{x_i}{x_0}\right), \qquad z_i = e^{j\omega_i}, \qquad i = 1, 2, \ldots, L-1$$

The symmetric z-transform of the window is then constructed in terms of its zeros:

$$W(z) = \prod_{i=1}^{L-1}(1 - z_i z^{-1})$$

The inverse z-transform of $W(z)$ are the window coefficients $w(n)$. The typical MATLAB code is,

```
L1 = L-1;                    % number of zeros
Ra = 10^(R/20);              % sidelobe level in absolute units
x0 = cosh(acosh(Ra)/L1);     % scaling factor

i = 1:L1;
xi = cos(pi*(i-0.5)/L1);     % L1 zeros of Chebyshev polynomial
omi = 2 * acos(xi/x0);       % L1 zeros in omega-space
zi = exp(j*omi);             % L1 zeros of W(z) polynomial

w = real(poly(zi));          % zeros-to-polynomial-coefficients
                             % see also the more accurate poly2
                             % in the EWA toolbox
```

The window coefficients resulting from this construction can be normalized to unity maximum.

The 3-dB frequency ω_3 is defined by the half-power condition:

$$W(\omega_3) = T_{L-1}(x_3) = \frac{T_{L-1}(x_0)}{\sqrt{2}} = \frac{R_a}{\sqrt{2}} \quad \Rightarrow$$

$$\cosh\big((L-1)\cosh^{-1}(x_3)\big) = \frac{R_a}{\sqrt{2}}$$

Solving for x_3 and the corresponding 3-dB angle ω_3, $x_3 = x_0\cos(\omega_3/2)$,

$$x_3 = \cosh\left(\frac{\cosh^{-1}(R_a/\sqrt{2})}{L-1}\right), \qquad \omega_3 = 2\cos^{-1}\left(\frac{x_3}{x_0}\right)$$

From ω_3, one calculates the 3-dB width, $\Delta\omega_{3\text{dB}} = 2\omega_3$. However, the following approximate relationship works well for the broadening factor, over the range $13 \le R \le 180$ dB, and may be used in project-4,

$$b = 0.65 + 0.0195\,R - 0.00005\,R^2$$

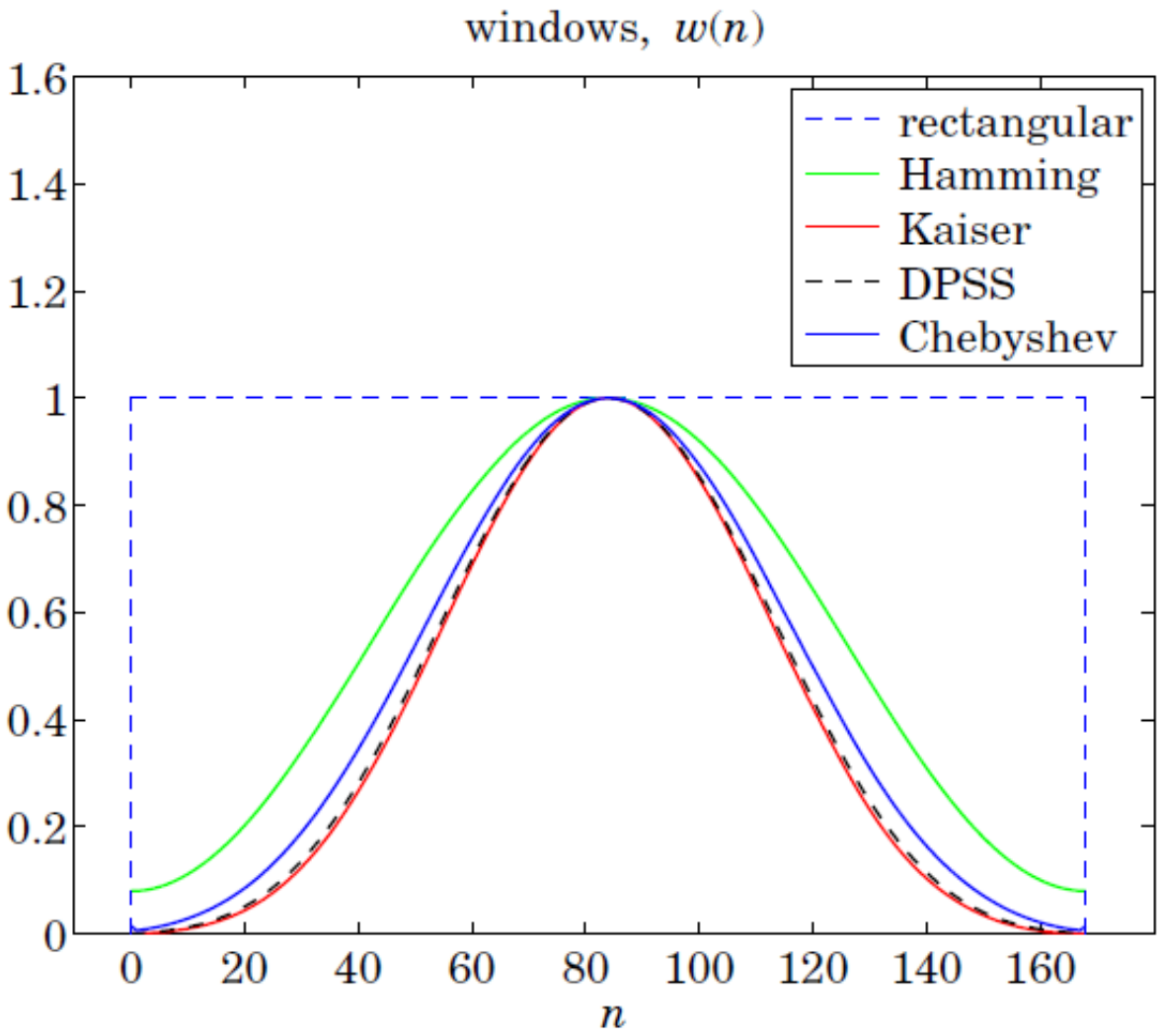
further notes on windows and project 4

In summary, the following broadening factors may be used in project-4 for the various windows:

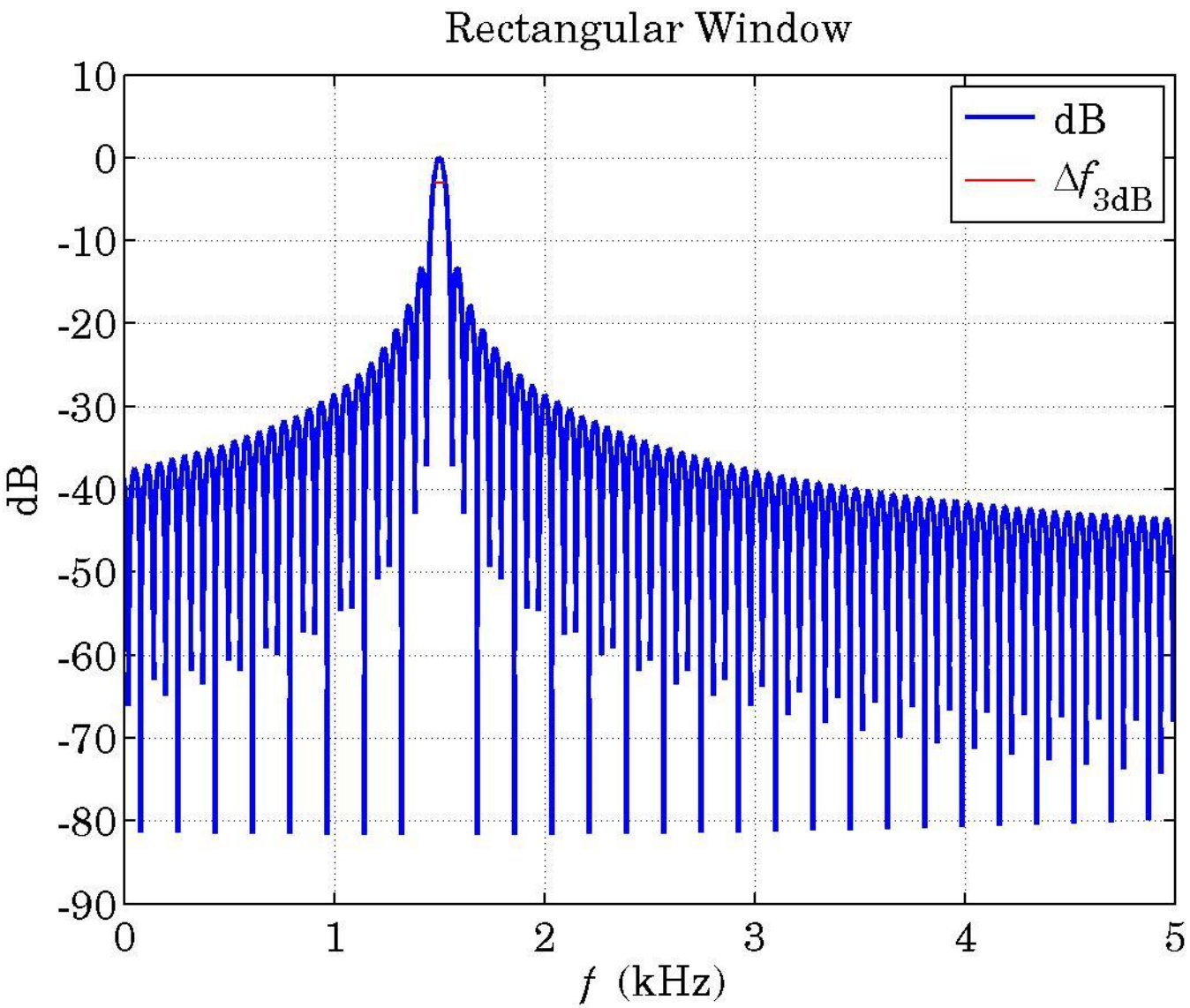
Rectangular	$b = 1$
Hamming	$b = 1.47 + 0.98/L$
Kaiser	$b = 0.0124\,R + 1.0221$
DPSS	$b = 0.0124\,R + 1.0221$
Chebyshev	$b = 0.65 + 0.0195\,R - 0.00005\,R^2$

$$\Delta\omega_{\text{3dB}} = 0.886\,\frac{2\pi\,b}{L} \quad \Rightarrow \quad \Delta f_{\text{3dB}} = 0.886\,\frac{f_s\,b}{L}$$

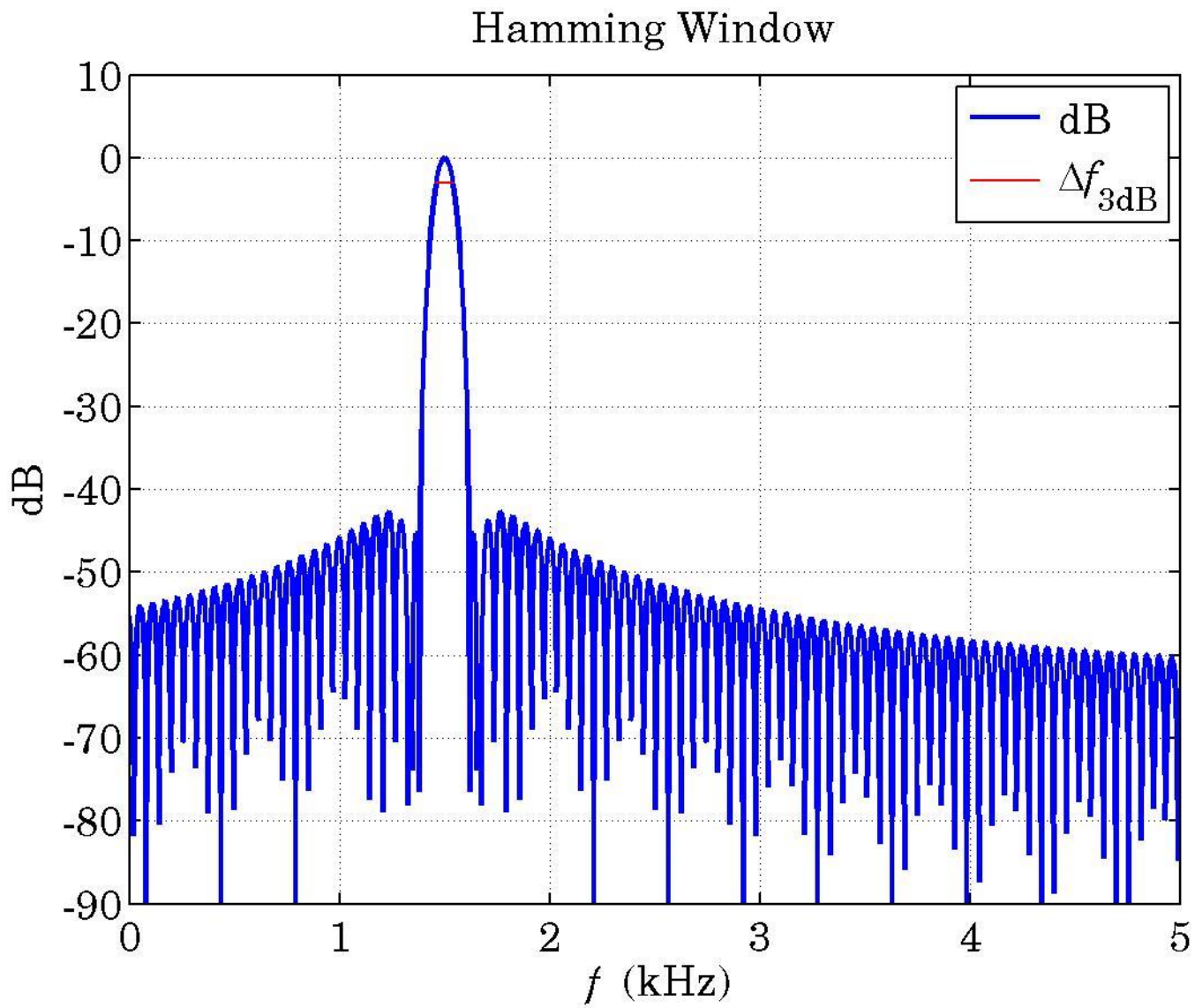
further notes on windows and project 4



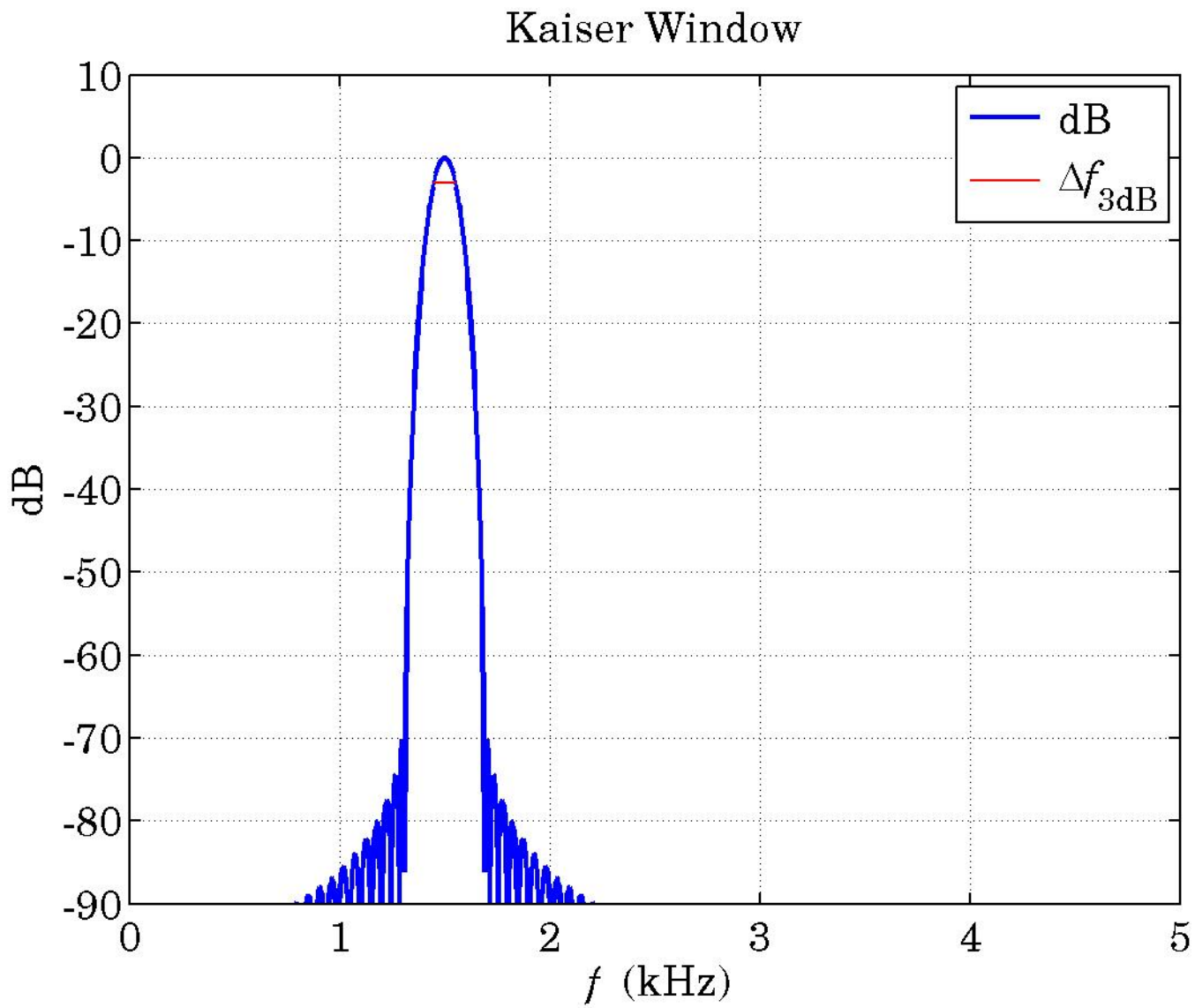
further notes on windows and project 4



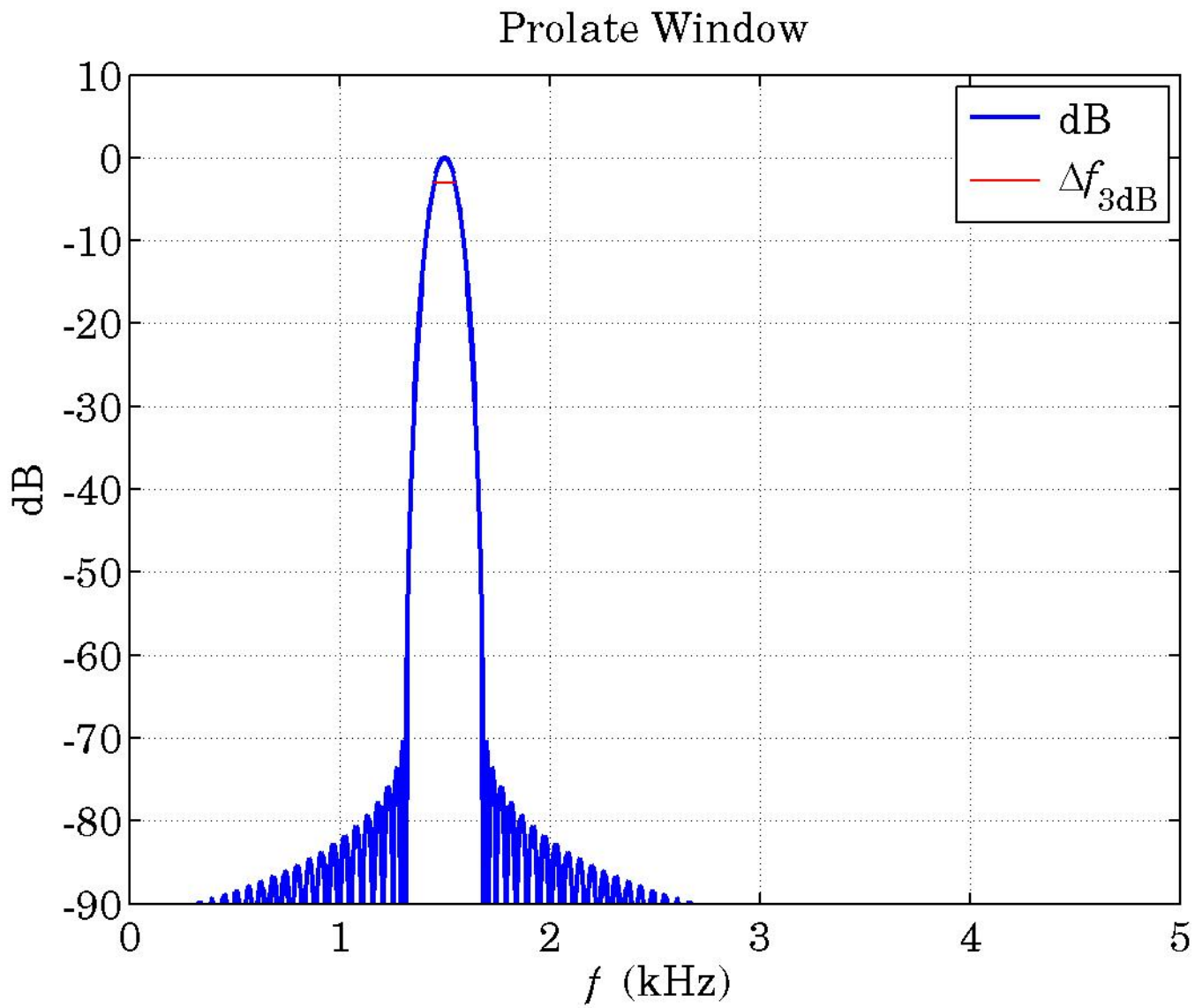
further notes on windows and project 4



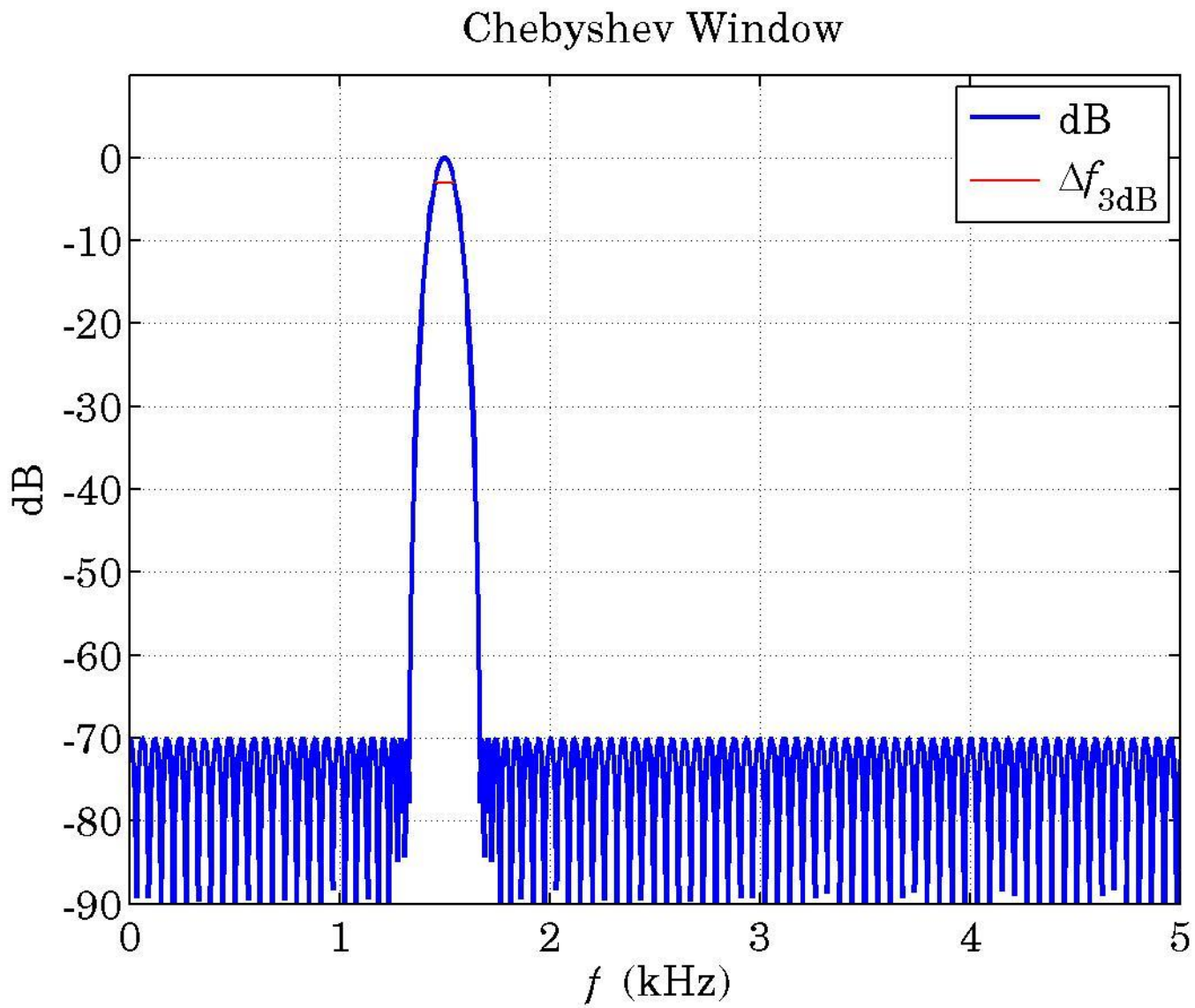
further notes on windows and project 4



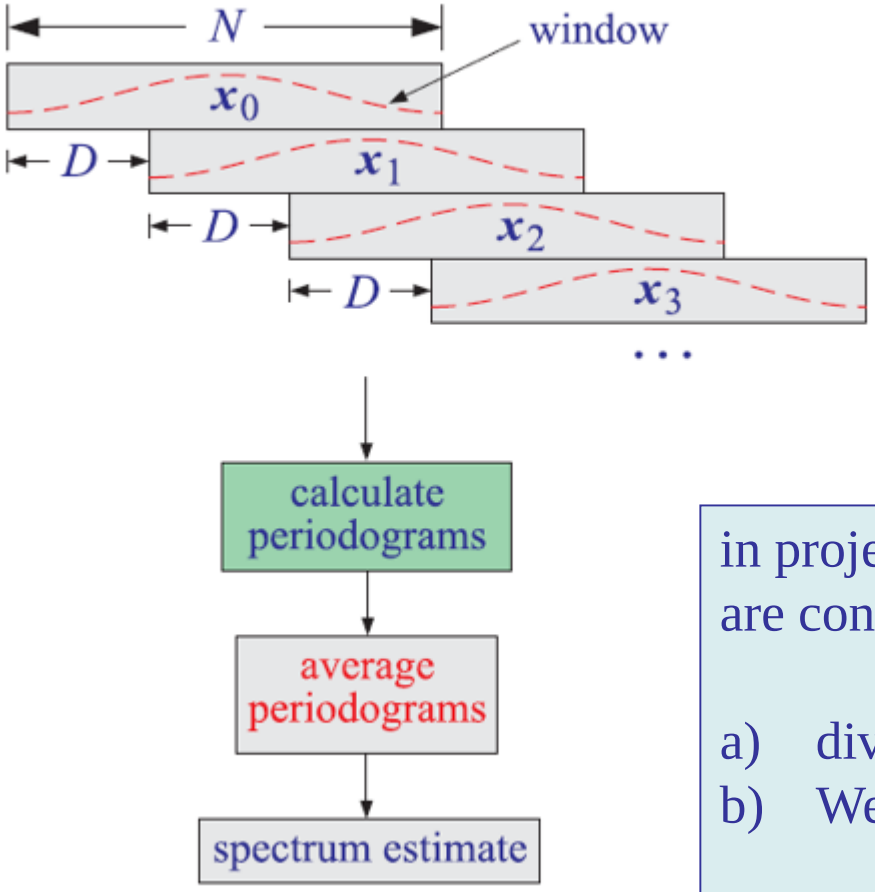
further notes on windows and project 4



further notes on windows and project 4



further notes on windows and project 4



in project-4, two periodogram averaging methods are considered:

a) dividing into non-overlapping blocks, $D = N$
b) Welch method, 50% overlapping, $D = N/2$

both cases can be implemented conveniently with the help of the built-in function **buffer**

its usage is explained with some examples below

further notes on windows and project 4

```
x = 10:2:46; N = 6;

% x =
%    10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46

X = buffer(x,N)          % non-overlapping, incomplete blocks

% X =
%      10     22     34     46
%      12     24     36      0
%      14     26     38      0
%      16     28     40      0
%      18     30     42      0
%      20     32     44      0
```

further notes on windows and project 4

```
x = 10:2:46; N = 6;

% x =
%    10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46

[X,~] = buffer(x,N)       % non-overlapping, complete blocks

% X =
%     10    22    34
%     12    24    36
%     14    26    38
%     16    28    40
%     18    30    42
%     20    32    44
```

further notes on windows and project 4

```
x = 10:2:46; N = 6;

% x =
%    10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46

X = buffer(x,N,N/2)       % 50% overlapping, delayed by N/2

% X =
%       0    10    16    22    28    34    40
%       0    12    18    24    30    36    42
%       0    14    20    26    32    38    44
%      10    16    22    28    34    40    46
%      12    18    24    30    36    42     0
%      14    20    26    32    38    44     0
```

further notes on windows and project 4

```
x = 10:2:46; N = 6;

% x =
%    10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46

X = buffer(x,N,N/2,'nodelay')    % 50% overlapping, no-delay

% X =
%      10     16     22     28     34     40
%      12     18     24     30     36     42
%      14     20     26     32     38     44
%      16     22     28     34     40     46
%      18     24     30     36     42      0
%      20     26     32     38     44      0
```

further notes on windows and project 4

```
x = 10:2:46; N = 6;

% x =
%    10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46

[X,~] = buffer(x,N,N/2,'nodelay')  % no-delay complete blocks

% X =
%     10    16    22    28    34
%     12    18    24    30    36
%     14    20    26    32    38
%     16    22    28    34    40
%     18    24    30    36    42
%     20    26    32    38    44
```