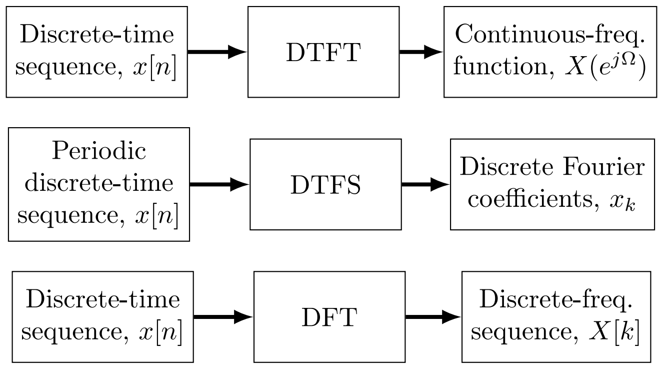
Chapter 7

Discrete Fourier Transform

We finally reach to a Fourier analysis technique that has no CT counterpart and is exclusive to DT signal processing: the discrete Fourier transform (DFT). While there are similarities to other DT Fourier analysis techniques like the DTFT and the DTFS, the DFT is different in the sense that that the transform maps any discrete-time sequence to a discrete-frequency sequence.

Figure 7.1: DT Fourier analysis techniques.


Discrete-time sequence, $x[n]$
DTFT
Continuous-freq. function, $X(e^{j\Omega})$
Periodic discrete-time sequence, $x[n]$
DTFS
Discrete Fourier coefficients, x_k
Discrete-time sequence, $x[n]$
DFT
Discrete-freq. sequence, $X[k]$

Just as the DTFT and the DTFS are periodic, the DFT is also periodic. This makes the DFT practical for real-time use as it can be computed over a single period with finite resources due to its discrete-frequency nature. Additionally, relationships between different transforms make it easy for offline analyis to convert from the DFT spectrum to another Fourier analysis spectrum.

As such, the DFT is one of the most eessential tools in DSP and provides the first foray into what practicing DSP looks like.

7.1 Discrete Fourier Transform

The *discrete Fourier transform* (DFT) of a DT signal $x[n]$ is an N-periodic discrete function of k defined as

$$X[k] = \text{DFT}[x[n]] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \tag{7.1}$$

$$= \sum_{n=0}^{N-1} x[n]W_N^{kn}, \text{ for } k = 0, 1, ..., N-1, \tag{7.2}$$

where $W_N = \exp(-j2\pi/N)$ is the *twiddle factor*, defined for ease of notation.

The DFT is essentially an operator that maps a signal defined in the *continuous-time domain* to another signal defined in the *discrete-frequency domain.* $x[n]$ and $X[k]$ constitute a unique *DFT pair.* This relationship can be written as

$$x[n] \iff X[k]. \tag{7.3}$$

Because of this relationship, there exists an *inverse DFT* defined by

$$x[n] = \text{DFT}^{-1}[X[k]] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{+j2\pi kn/N} \tag{7.4}$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-kn}, \text{ for } n = 0, 1, ..., N-1. \tag{7.5}$$

Because the DFT requires both input $x[n]$ and output $X[k]$ to be discrete sequences, the DFT over a single period N can be expressed as a matrix equation:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix} \tag{7.6}$$

The *orthogonality principle* may be of particular use when computing the DFT by hand and is defined as

$$\sum_{n=0}^{N-1} X[k]e^{j2\pi(m-k)n/N} = \begin{Bmatrix} N, & k = m \\ 0, & k \neq m \end{Bmatrix} = N \cdot \delta[k-m]. \tag{7.7}$$

Note that both $x[n]$ and $X[k]$ are defined over a single period N. It makes sense that the transform is periodic, but this also means there are some implications about $x[n]$.

1. If the DT signal is N_0-periodic, then by letting $N = N_0$, the DFT is simply just a scaled version of the DTFS with $X[k] = N_0 x_k$.
2. Else if the signal is nonperiodic, then the DFT extracts the windowed signal of interest (essentially clipping a signal to finite duration of length N), assumes that windowed signal is N-periodic, and essentially converts the DTFT into a DTFS with $X[k] = Nx_k$.

For now, let $x[n]$ be the length-N extracted DT signal. Later, we will look more into how to window a DT signal, for which $x_W[n]$ may be a more appropriate notation for the extracted signal.

Lastly, since the DFT is N-periodic, the modulo operation will be applied to the indices to represent its periodic nature. Angle brackets will be used for shorthand such that

$$\langle k \rangle_N = (k \mod N). \tag{7.8}$$

When isolated to just one period for $k \in [0, N-1]$, two index equations arise:

$$\langle k \rangle_N = k, \tag{7.9}$$
$$\langle -k \rangle_N = N - k. \tag{7.10}$$

Of course, this suggests that operations related to the DFT will be cyclic in a sense. Table 7.1 lists the properties of the DFT.

Table 7.1: Properties of the length–N DFT.

Property	$x[n]$	$X[k] = \mathrm{DFT}[x[n]]$
Superposition	$A_1x_1[n] + A_2x_2[n]$	$A_1X_1[k] + A_2X_2[k]$
(Circular) time shift	$x[\langle n-n_0 \rangle_N]$	$e^{-j2\pi kn_0/N}X[k] = W_N^{kn_0}X[k]$
Time reversal	$x[\langle -n \rangle_N]$	$X[\langle -k \rangle_N]$
Conjugation	$x^*[n]$	$X^*[\langle -k \rangle_N]$
Frequency shift	$e^{+j2\pi k_0n/N}x[n] = W_N^{-k_0n}x[n]$	$X[\langle k-k_0 \rangle_N]$
Circular convolution	$x_1[n] \circledast x_2[n]$	$X_1[k]X_2[k]$
Multiplication	$x_1[n]x_2[n]$	$\frac{1}{2}\left[X_1[k] \circledast X_2[k]\right]$
Duality	$X[n]$	$N\,x[\langle -k \rangle_N]$
Conjugate symmetry	$x[n]$ real	$\begin{cases} X^*[k] = X^*[\langle -k \rangle_N] \\ \mathrm{Re}(X[k]) = \mathrm{Re}(X[\langle -k \rangle_N]) \\ \mathrm{Im}(X[k]) = -\mathrm{Im}(X[\langle -k \rangle_N]) \\ \lvert X[k] \rvert = \lvert X[\langle -k \rangle_N] \rvert \\ \arg[X[k]] = -\arg[X[\langle -k \rangle_N]] \end{cases}$
Even-odd decomposition of real signals	$\begin{cases} x_e[n] = \frac{1}{2}[x[n] + x[\langle -n \rangle_N]] \\ x_o[n] = \frac{1}{2}[x[n] - x[\langle -n \rangle_N]] \end{cases}$	$\begin{cases} \mathrm{Re}(X[k]) \\ j\,\mathrm{Im}(X[k]) \end{cases}$

From the conjugate symmetry property, there are important implications about the properties of $x[n]$ and its DFT $X[k]$:

$x[n]$	$X[k]$
Real and even	Real and even
Real and odd	Imaginary and odd
Imaginary and even	Imaginary and even
Imaginary and odd	Real and odd

7.2 Parseval's Theorem for DFT

Parseval's theorem for the DFT is essentially a "conservation of (average) power" theorem, since $x[n]$ is operated on as if periodic. When mapped from the discrete-time to the discrete-frequency domain, the total signal average power is conserved. It follows that the total average power of a length-N DT signal $x[n]$ can be evaluated as:

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{k=0}^{N-1} |X[k]|^2. \tag{7.11}$$

Additionally, we can define the *one-sided power spectral density* (1-sided PSD) to be

$$PSD_1 = \begin{cases} |X[0]|^2, & k = 0 \\ 2|X[k]|^2, & k > 0 \end{cases} \tag{7.12}$$

and the *two-sided power spectral density* (2-sided PSD) to be

$$PSD_2 = |X[k]|^2. \tag{7.13}$$

Note that the 2-sided PSD is defined for all frequencies, whereas the 1-sided PSD is defined for only nonnegative frequencies. Because of this, the AC component of the 2-sided PSD are half the values of the 1-sided PSD. In signal processing, we tend to be more interested in the 1-sided PSD.

7.3 Circular Convolution

Here, we finally introduce the concept of *circular convolution* (also called *cyclic convolution* or *periodic convolution*), which is different from linear convolution. In its simplest form, circular convolution is an operation ($\circledast$) defined by the *circular convolution sum*

$$x[n] \circledast h[n] = \sum_{m=0}^{N-1} x[m]h[\langle n-m\rangle_N], \tag{7.14}$$

where $x[n]$ and $h[n]$ are both length-N. However, circular convolution also allows for convolving sequences of different lengths N_x and N_h. For the generalized case, the symbol $\textcircled{N}$ is used such that

$$x[n] \textcircled{N} h[n] = \sum_{m=0}^{N-1} x[m]h[\langle n-m\rangle_N], \text{ for } N = \max\{N_x, N_h\}. \tag{7.15}$$

Of course, this also implies that the selection of N is very flexible and does not have to be the maximum length of the two sequences. Those cases will not be covered here.

In general, circular convolution is a temporally aliased version of the linear convolution in the time domain. As a result, we can leverage that fact by aliasing the linear convolution to compute the circular convolution.

Recall that the linear convolution of two finite sequences is given by

$$y_L[n] = x[n] * h[n] = \sum_{m=-\infty}^{+\infty} x[m]h[n-m], \tag{7.16}$$

with (domain $x[n]$) $= [0, N_x - 1]$ and (domain $h[n]$) $= [0, N_h - 1]$. Then by wrapping the linear convolution sequence, the circular convolution over one period $n \in [0, N-1]$ for $N = \max\{N_x, N_h\}$ can be defined as

$$x[n] \textcircled{N} h[n] = \begin{cases} y_L[n] + y_L[n+N], & 0 \le n < N_L - N \\ y_L[n], & \text{otherwise.} \end{cases} \tag{7.17}$$

Example 7.3.1. Use temporal aliasing to find $y[n] = x[n] \circledast h[n]$, given that

$$x[n] = \{\underline{2}, 1, 4, 3\},$$
$$h[n] = \{\underline{5}, 3, 2, 1\}.$$

SOLUTION

Note that the period for both sequences is $N = 4$. Setting up the condensed tabular method for linear convolution:

$x[n]$ \ $h[n]$	$\underline{5}$	3	2	1
$\underline{2}$	10	6	4	2
1	5	3	2	1
4	20	12	8	4
3	15	9	6	3

Each right-to-left diagonal is highlighted a different color to show which entries should be summed. Then the linear convolution is given by:

$$\begin{aligned} y_L[n] &= \{\underline{10}, 6+5, 4+3+20, 2+2+12+15, 1+8+9, 4+6, 3\} \\ &= \{\underline{10}, 11, 27, 31, 18, 10, 3\}. \end{aligned}$$

Then by aliasing the output, the circular convolution is given by:

$$
\begin{aligned}
x[n] \circledast h[n] &= \begin{cases} y_L[n] + y_L[n+4], & 0 \le n < (7-4) \\ y_L[n], & \text{otherwise} \end{cases} \\
&= \{\underline{10+18}, 11+10, 27+3, 31\} \\
&= \{\underline{28}, 21, 30, 31\}\,.
\end{aligned}
$$

■

Example 7.3.2. Use the DFT to find $y[n] = x[n] \circledast h[n]$, given that

$$
\begin{aligned}
x[n] &= \{\underline{2}, 1, 4, 3\}\,, \\
h[n] &= \{\underline{5}, 3, 2, 1\}\,.
\end{aligned}
$$

SOLUTION

From the table of DFT properties, we can multiply the DFTs of the two sequences and then compute the inverse DFT of the product. Then for $N = 4$:

$$
\begin{aligned}
X[0] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi(0)n/N} = 2 + 1 + 4 + 3 = 10 \\
X[1] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi(1)n/N} = 2 - j1 - 4 + j3 = -2 + j2 \\
X[2] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi(2)n/N} = 2 - 1 + 4 - 3 = 2 \\
X[3] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi(3)n/N} = 2 + j1 - 4 - j3 = -2 - j2 \\
&\ldots \\
H[0] &= \sum_{n=0}^{N-1} h[n] e^{-j2\pi(0)n/N} = 5 + 3 + 2 + 1 = 11 \\
H[1] &= \sum_{n=0}^{N-1} h[n] e^{-j2\pi(1)n/N} = 5 - j3 - 2 + j1 = 3 - j2 \\
H[2] &= \sum_{n=0}^{N-1} h[n] e^{-j2\pi(2)n/N} = 5 - 3 + 2 - 1 = 3 \\
H[3] &= \sum_{n=0}^{N-1} h[n] e^{-j2\pi(3)n/N} = 5 + j3 - 2 - 1 = 3 + j2
\end{aligned}
$$

Performing pointwise multiplication, we get

$$\begin{aligned}
Y[0] &= X[0]H[0] = (10)(11) = 110 \\
Y[1] &= X[1]H[1] = (-2+j2)(3-j2) = -2+j10 \\
Y[2] &= X[2]H[2] = (2)(3) = 6 \\
Y[3] &= X[3]H[3] = (-2-j2)(3+j2) = -2-j10
\end{aligned}$$

Lastly, compute the inverse DFT.

$$\begin{aligned}
y[0] &= \frac{1}{N}\sum_{k=0}^{N-1} Y[k]e^{+j2\pi k(0)/N} = \frac{1}{4}[110 + (-2+j10) + 6 + (-2-j10)] = 28 \\
y[1] &= \frac{1}{N}\sum_{k=0}^{N-1} Y[k]e^{+j2\pi k(1)/N} = \frac{1}{4}[110 + j(-2+j10) - 6 - j(-2-j10)] = 21 \\
y[2] &= \frac{1}{N}\sum_{k=0}^{N-1} Y[k]e^{+j2\pi k(2)/N} = \frac{1}{4}[110 - (-2+j10) + 6 - (-2-j10)] = 30 \\
y[3] &= \frac{1}{N}\sum_{k=0}^{N-1} Y[k]e^{+j2\pi k(3)/N} = \frac{1}{4}[110 - j(-2+j10) - 6 + j(-2-j10)] = 31
\end{aligned}$$

Though tedious, we get the same solution as the aliasing method. ■

While circular convolution is not the same as linear convolution, *zero-padding* can be used with circular convolution to obtain the same output as linear convolution. Let

$$z_x[n] = \{\underline{x[0]}, x[1], ..., x[N_x-1], \underbrace{0, 0, ..., 0}_{N-N_x}\}, \tag{7.18}$$

$$z_h[n] = \{\underline{h[0]}, h[1], ..., h[N_h-1], \underbrace{0, 0, ..., 0}_{N-N_h}\}, \text{ for } N \geq N_x + N_h - 1. \tag{7.19}$$

Then the unaliased linear convolution can be retrieved from the circular convolution of two zero-padded sequences:

$$y_L[n] = z_x[n] \circledN z_h[n], \text{ for } N \geq N_x + N_h - 1. \tag{7.20}$$

7.4 Relationships Between Transforms

It can be said that the DFT is a sampled version of the DTFT, with

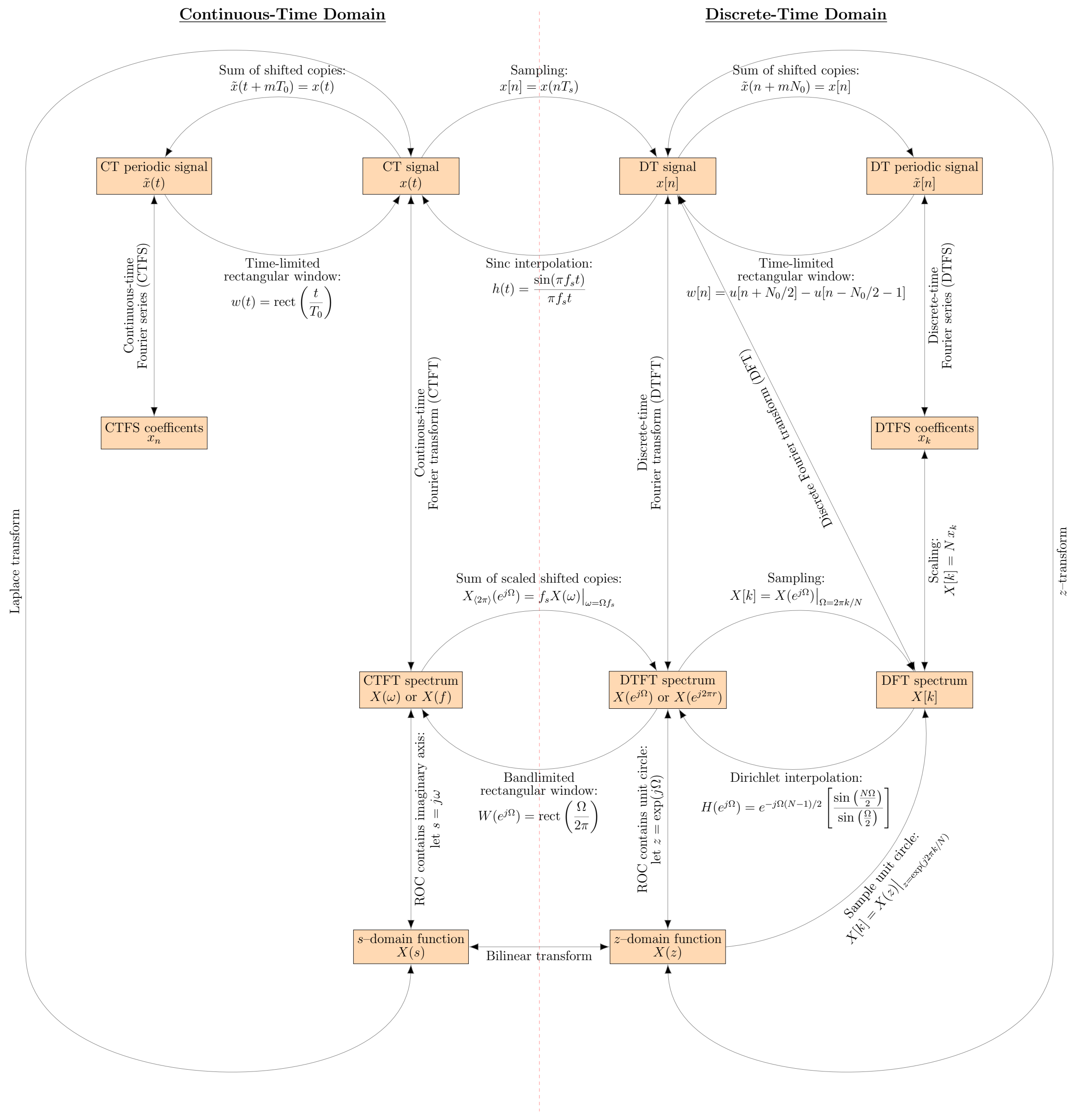
$$X[k] = X(e^{j\Omega})\big|_{\Omega=2\pi k/N}. \tag{7.21}$$

By virtue of the z–transform being related to the DTFT, it can also be said that the DFT samples the unit circle of the z–transform such that

$$X[k] = X(z)\big|_{z=\exp(j2\pi k/N)}. \tag{7.22}$$

Figure 7.2 shows the relationships between all CT and DT transforms that we have learned so far. As such, this marks the end of theory-based signals and systems as we begin moving towards DSP in practice.

Figure 7.2: Relationships between transforms.



Continuous-Time Domain
Discrete-Time Domain
Sum of shifted copies: $\tilde{x}(t+mT_0)=x(t)$
Sampling: $x[n]=x(nT_s)$
Sum of shifted copies: $\tilde{x}(n+mN_0)=x[n]$
CT periodic signal $\tilde{x}(t)$
CT signal $x(t)$
DT signal $x[n]$
DT periodic signal $\tilde{x}[n]$
Time-limited rectangular window: $w(t)=\mathrm{rect}\left(\frac{t}{T_0}\right)$
Sinc interpolation: $h(t)=\frac{\sin(\pi f_s t)}{\pi f_s t}$
Time-limited rectangular window: $w[n]=u[n+N_0/2]-u[n-N_0/2-1]$
Continuous-time Fourier series (CTFS)
Discrete-time Fourier series (DTFS)
Continous-time Fourier transform (CTFT)
Discrete-time Fourier transform (DTFT)
Discrete Fourier transform (DFT)
CTFS coefficents x_n
DTFS coefficents x_k
Laplace transform
z–transform
Scaling: $X[k]=N\,x_k$
Sum of scaled shifted copies: $X_{\langle 2\pi\rangle}(e^{j\Omega})=f_sX(\omega)\big|_{\omega=\Omega f_s}$
Sampling: $X[k]=X(e^{j\Omega})\big|_{\Omega=2\pi k/N}$
CTFT spectrum $X(\omega)$ or $X(f)$
DTFT spectrum $X(e^{j\Omega})$ or $X(e^{j2\pi r})$
DFT spectrum $X[k]$
ROC contains imaginary axis: let $s=j\omega$
Bandlimited rectangular window: $W(e^{j\Omega})=\mathrm{rect}\left(\frac{\Omega}{2\pi}\right)$
ROC contains unit circle: let $z=\exp(j\Omega)$
Dirichlet interpolation: $H(e^{j\Omega})=e^{-j\Omega(N-1)/2}\left[\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)}\right]$
Sample unit circle: $X[k]=X(z)\big|_{z=\exp(j2\pi k/N)}$
s–domain function $X(s)$
Bilinear transform
z–domain function $X(z)$

7.5 Recontextualizing the DFT

From what we already know, when extracting a length-N data sequence $x[n]$ and taking the DFT, we get some length-N sampled frequency spectrum $X[k]$. Now let us introduce some new terminology and variables that are more common in practice.

7.5.1 Time-Domain Terminology

From before, the *sampling rate* is the number of samples taken per second from a CT signal to create a DT signal and is defined as

$$f_s = \frac{1}{T_s} \text{ [samp/sec]}, \tag{7.23}$$

where T_s is the sampling period (or sampling interval), or the amount of time between consecutive samples. This extracted sequence $x[n]$ is typically causal and is referred to as a *frame* of data.

We also now redefine the sampling period as *temporal resolution*, which is a measurement of how fine the spacing between the time samples are and is defined as

$$\Delta t = T_s = \frac{1}{f_s} \text{ [sec]}. \tag{7.24}$$

The point $x[n_0]$ is referred to as the *time sample* at time index n_0. The actual *elapsed time* at n_0 is given by $n_0 \Delta t$.

Instead of referring to $x[n]$ as length-N, we say that the *frame size* or (*frame length*) is N:

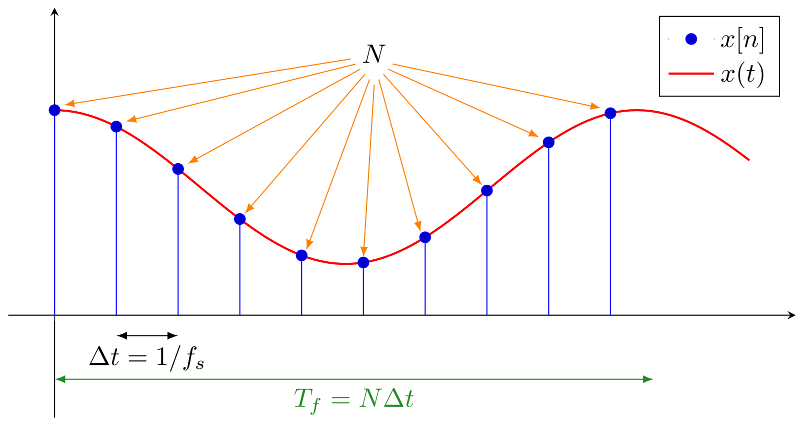
$$N = \text{count}(x[n]) \text{ [samp]}. \tag{7.25}$$

The *frame time* is the total amount of time corresponding to frame size N and is given by

$$T_f = N\Delta t = \frac{N}{f_s} \text{ [sec]}. \tag{7.26}$$

Its reciprocal is referred to as the *frame rate* of the signal.

Figure 7.3: Frame of data in the time domain.



7.5.2 Frequency-Domain Terminology

From before, the sampling rate f_s should follow the Nyquist–Shannon sampling theorem such that a CT signal with highest frequency content at f_{max} can be captured under the criterion

$$f_s \geq \frac{f_{max}}{2} \text{ [samp/sec]}, \tag{7.27}$$

with the Nyquist frequency defined at equality. This highest frequency f_{max} is referred to as the *bandwidth* of the signal being sampled.

As mentioned before, when performing the DFT, the transform essentially samples the DTFT spectrum. Because the frame size is N, we would take the length-N DFT. In other words, the *DFT size* is N. Just as the frame is defined by frame size N, the DFT spectrum is defined by DFT size N.

The *frequency resolution* then is a measurement of how fine the spacing between the spectral samples are and is defined as

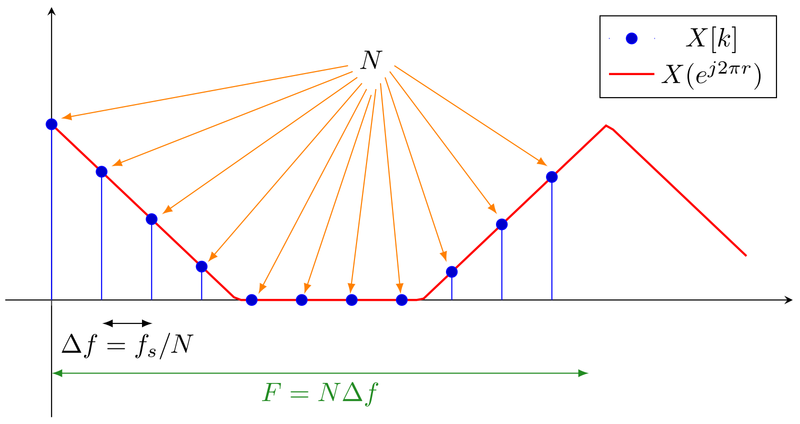
$$\Delta f = \frac{f_s}{N} \text{ [Hz]}. \tag{7.28}$$

The point $X[k_0]$ is referred to as the *spectral sample* at frequency index k_0. The spacing between frequency indices k_m and k_{m+1} is called the m^{th} *frequency bin.*

The *frequency range* of the (two-sided) spectrum corresponding to DFT size N is given by

$$F = N\Delta f \text{ [Hz]}. \tag{7.29}$$

Figure 7.4: DFT spectrum in the frequency domain.



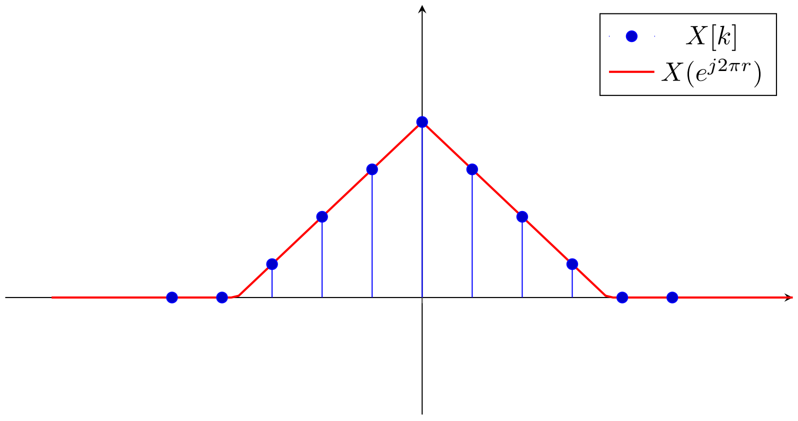
While not presented as two-sided, typically the latter half of the spectrum is shifted one period to the left to form a more meaningful sampled spectrum, as seen going from Fig. 7.4 to Fig. 7.5. Going forward, we will assume the circularly shifted form of the spectrum.

Lastly, while not as common, *angular frequency resolution* can be defined as

$$\Delta\omega = \frac{2\pi f_s}{N} = 2\pi\Delta f. \tag{7.30}$$

This relationship becomes important if one were to choose to plot the spectrum in terms of angular frequency.

Figure 7.5: Circularly shifted DFT spectrum in the frequency domain.

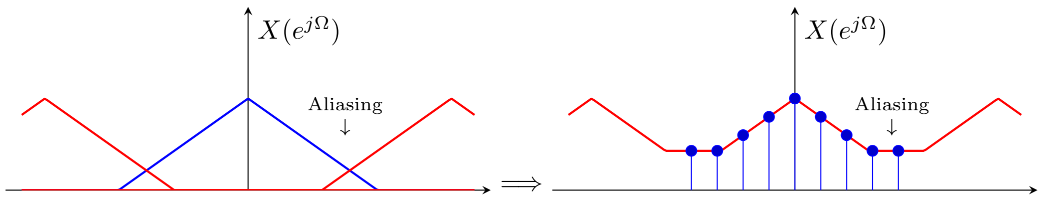


7.6 Error Sources in the DFT

Due to the discrete and implicitly periodic nature of the DFT, the DFT is prone to capturing some errors. This section will briefly cover the main sources and potential solutions.

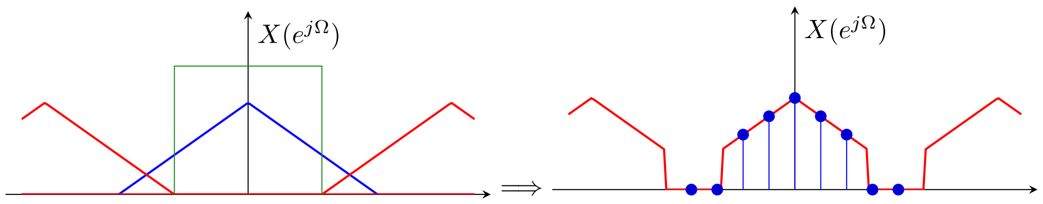
7.6.1 Aliasing

As seen before in undersampled signals as well as downsampled signals, *aliasing* is a type of signal distortion caused by spectral overlap between neighboring spectral copies in the frequency domain. Essentially, the higher frequency content unintentionally becomes amplified and more prominent in the captured signal, thus contaminating the original time signal. When performing the DFT on the contaminated signal, we would get a sampled version of this distorted frequency spectrum.



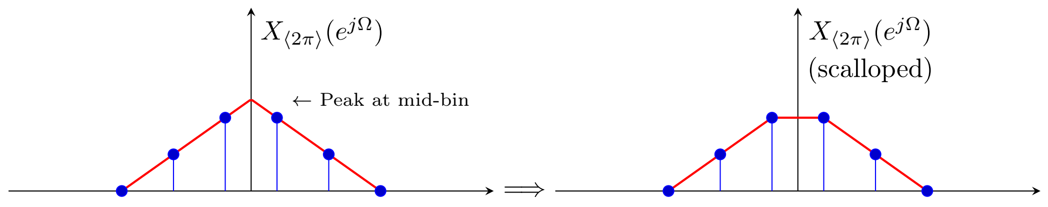
There are two possible methods to combat (spectral) aliasing. The first possible solution is to increase the sampling rate. This is typically done at the hardware-level, in which the the DSP engineer would upgrade or replace the ADC containing the sampler.

The second possible solution is to apply an anti-aliasing filter. As seen before with decimated signals, an anti-aliasing filter essentially restrict the bandwidth of the undersampled signal to remove all undesired higher frequency content that is causing the distortion. The cutoff frequency is typically at half the Nyquist frequency. This however also means that the recovered signal inevitably has some information loss.

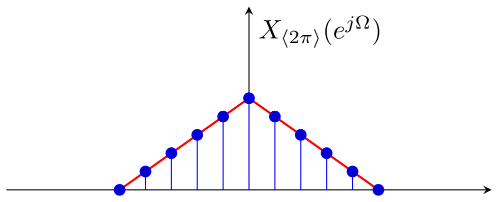


7.6.2 Picket-Fence Effect

The *picket-fence effect* (also called *scalloping loss*) is the phenomenon of missing pertinent frequency artifacts (such as a large spectral component or a spectral null) that occur mid-bin in the DFT spectrum due to insufficient frequency sampling. Recall that both the frame size and the DFT spectrum share the same value N; if not enough time samples are taken, that also gets reflected in the small number of spectral samples computed.


$X_{\langle 2\pi \rangle}(e^{j\Omega})$
← Peak at mid-bin
$\Longrightarrow$
$X_{\langle 2\pi \rangle}(e^{j\Omega})$
(scalloped)

There is a simple solution to this problem. Recall that the frequency resolution is given by $\Delta f = f_s/N$. Keeping the sampling rate the same, one can increase the frame size N by zero-padding the original extracted sequence such that the frequency resolution gets finer, and more spectral samples are collected.



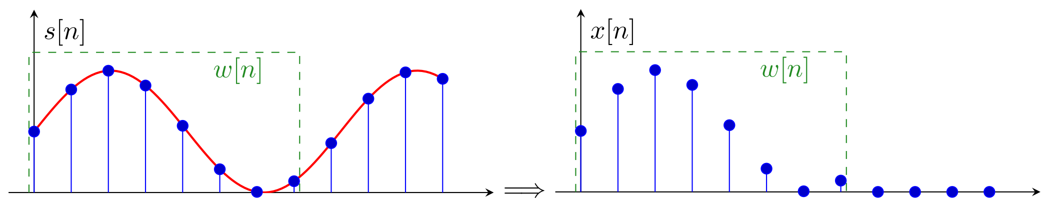
7.6.3 Windowing and Spectral Leakage

Windowing is the process of extracting a frame of data by multiplying the original source signal $s[n]$ by a finite-duration *window function* $w[n]$. Mathematically, this is represented by

$$x[n] = s[n]w[n], \tag{7.31}$$

where $x[n]$ is the extracted frame of data. Since the window $w[n]$ is typically a rectangular sequence, the windowed sequence is given by

$$x[n] = \begin{cases} s[n], & 0 \leq n < N-1 \\ 0, & \text{otherwise.} \end{cases} \tag{7.32}$$

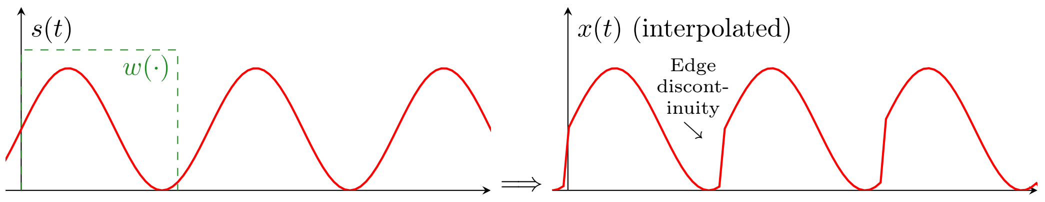


Of course, this is implies that two different source signals can have the same windowed signal and thus the same DFT due to the DFT's assumption of periodicity.

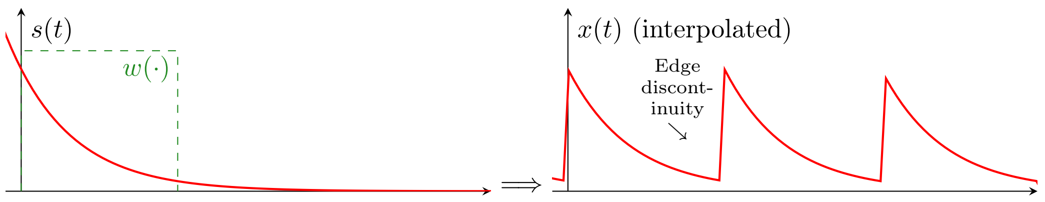
Additionally, the rectangular window is an ideal window for the DFT as its DTFT is the Dirichlet kernel (also called the periodic sinc), which acts as a convolution filter in Ω that interpolates between frequency bins. The higher the frame size N, the more accurate the DFT is as the sampled DTFT.

However, because of this assumed periodicity, *spectral leakage* can occur, in which the presence of endpoint discontinuities (called *edge discontinuities*) can introduce higher-frequency distortion, forcing the DFT spectrum to redistribute its power as power undesirably "leaks" from large spectral components to neighboring frequencies (essentially a smearing effect).

There are two ways endpoint discontinuities can occur. First, if the source signal is already periodic itself, then edge discontinuities can occur if the windowed sequence does not contain an integer number of cycles. Below shows what happens when such sampled signals are interpolated and periodically "stitched" to form the distorted periodic signal.



The second and more common way assumes source signals are nonperiodic (such as speech signals). Regardless of where the windowing ends, unintentional sharp transitions between the endpoints of a windowed sequence introduce edge discontinuities that can cause spectral leakage.



For periodic signals, one way to mitigate spectral leakage is to increase the frame size by extending the duration of the window, ideally until there is minimal separation between the two endpoints. However, there is another solution that can be generalized for all signals.

Until now, the window function is assumed to be rectangular. This does not have to be strictly followed. Table 7.2 shows a list of some commonly used windows and their features.

7.6.4 Rectangular Window

The *rectangular window* can be defined as

$$w[n] = \begin{cases} 1, & n \in [0, N-1] \\ 0, & \text{otherwise} \end{cases} \tag{7.33}$$

$$\Longrightarrow W(e^{j\Omega}) = e^{-j\Omega(N-1)/2} \left[\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)} \right]. \tag{7.34}$$

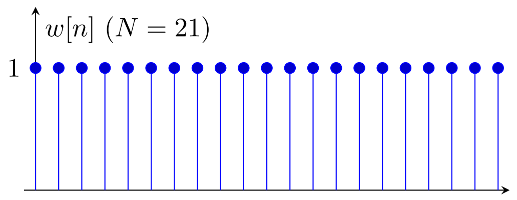


Table 7.2: Window functions.

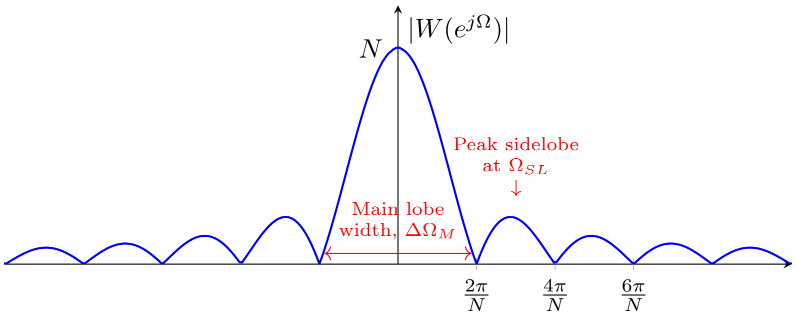
Window function $w[n]$, $n \in [0, N-1]$	Main lobe width (in Ω) $\Delta\Omega_M$ [rad/samp]	Sidelobe attenuation Γ [dB]	Sidelobe roll-off rate ΔL [dB/oct]	Normalization factor $1/W(e^{j0})$
Rectangular	$4\pi/N$	-13	-6	$1/N$
Bartlett	$8\pi/(N-1)$	-27	-12	$2/(N-1)$
Hann	$8\pi/(N-1)$	-32	-18	$2/(N-1)$
Hamming	$8\pi/(N-1)$	-43	-6	$1/(0.54N-0.46)$
Blackman	$12\pi/(N-1)$	-58	-18	$1/(0.42(N-1))$

The *main lobe width* $\Delta\Omega_M$ is the total bandwidth that occupies the central lobe of a window's magnitude spectrum. This is calculated by finding the positive frequency where the first *null* occurs (essentially where the magnitude spectrum is zero), then multiplying the frequency by 2 to get the main lobe width. That is,

$$\Delta\Omega_M = 2 \times \underbrace{\min\left\{\Omega > 0 \mid |W(e^{j\Omega})| = 0\right\}}_{\Omega_{null_1}}, \tag{7.35}$$

For the rectangular window,

$$\begin{aligned}\Omega_{null_1} = \arg[|W(e^{j\Omega})| = 0] = \arg\left[\left|\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)}\right| = 0\right] &= \arg\left[\sin\left(\frac{N\Omega}{2}\right) = 0\right] \\ &= \arg\left[\frac{N\Omega}{2} = \pi\right] = \frac{2\pi}{N} \\ \implies \Delta\Omega_M = 2 \times \frac{2\pi}{N} = \frac{4\pi}{N}.\end{aligned}$$



The *sidelobe attenuation* Γ, sometimes called the *sidelobe level* relative to the main lobe, describes the peak sidelobe relative to the main lobe and is measured in negative decibels. This is measured by calculating

$$\Gamma = 20\log_{10}\frac{|W(e^{j\Omega_{SL}})|}{|W(e^{j0})|}\ \text{[dB]},\ \text{for} \tag{7.36}$$

$$\Omega_{SL} = \min\left\{\Omega > 0 \;\middle|\; \frac{d}{d\Omega}|W(e^{j\Omega})| = 0\right\} \approx \frac{\Omega_{null_1}+\Omega_{null_2}}{2}, \tag{7.37}$$

where the estimate is arrived by assuming the first sidelobe peak to be around halfway between the first two nulls. For the rectangular window, using small angle approximation ($\sin\theta \approx \theta$ as $\theta \to 0$),

$$\begin{aligned}
\Omega_{SL} &\approx \frac{\frac{2\pi}{N}+\frac{4\pi}{N}}{2} = \frac{3\pi}{N} \\
\implies \Gamma = 20\log_{10}\frac{|W(e^{j3\pi/N})|}{|W(e^{j0})|} &= 20\log_{10}\frac{\frac{1}{|\sin(3\pi/(2N))|}}{N} \\
&\approx 20\log_{10}\frac{\frac{1}{|3\pi/(2N)|}}{N} = 20\log_{10}\frac{(2N)/(3\pi)}{N} \\
&= 20\log_{10}\frac{2}{3\pi} \approx -13\ \text{[dB]}.
\end{aligned}$$

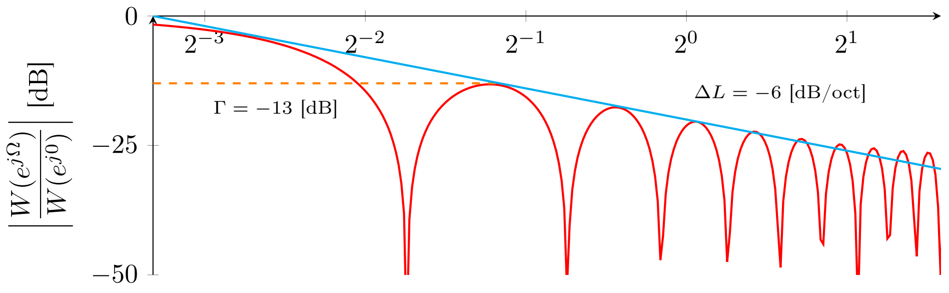
The *sidelobe roll-off rate* ΔL describes how the envelope of the magnitude spectrum changes over a period, i.e., how the sidelobe peaks change as $|\Omega| \to \pi$. While *decades* (dec., for short) are used in continuous-time applications such as Bode plots to describe a frequency change by a factor of 10, i.e.,

$$\text{number of decades} = \log_{10}\left(\frac{f_2}{f_1}\right), \tag{7.38}$$

discrete-time applications such as spectral analysis tend to use *octaves* (oct., for short) to describe a frequency change by a factor of 2, i.e.,

$$\text{number of octaves} = \log_2\left(\frac{f_2}{f_1}\right). \tag{7.39}$$

It follows that an asymptotic approximation of -20 [dB/dec] is roughly the same as an asymptotic approximation of -6 [dB/oct].

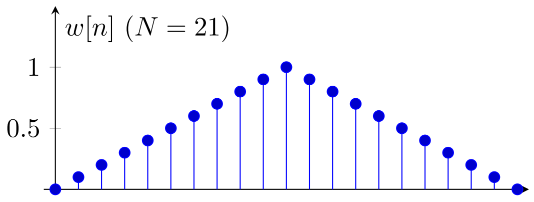


7.6.5 Bartlett Window

The *Bartlett window* is defined as

$$w[n] = \begin{cases} 1 - \dfrac{\left|n - \frac{N-1}{2}\right|}{\left(\frac{N-1}{2}\right)}, & n \in [0, N-1] \\ 0, & \text{otherwise} \end{cases} \tag{7.40}$$

$$\Longrightarrow W(e^{j\Omega}) = \frac{2}{N-1} \cdot e^{-j\Omega(N-1)/2} \left[\frac{\sin\left(\frac{(N-1)\Omega}{4}\right)}{\sin\left(\frac{\Omega}{2}\right)} \right]^2 . \tag{7.41}$$

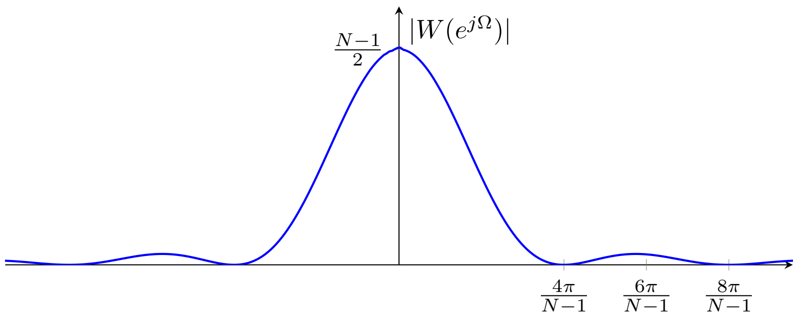


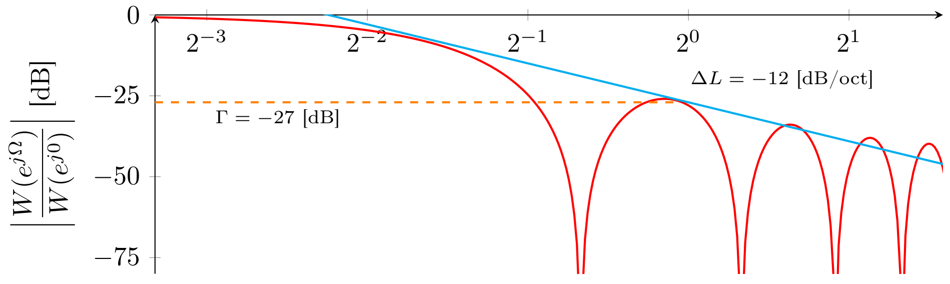
Then it follows that

$$\begin{aligned} \Omega_{null_1} &= \arg\left[\left| \frac{\sin^2\left(\frac{(N-1)\Omega}{4}\right)}{\sin^2\left(\frac{\Omega}{2}\right)} \right| = 0 \right] = \arg\left[\sin^2\left(\frac{N\Omega}{4}\right) = 0 \right] \\ &= \arg\left[\frac{(N-1)\Omega}{4} = \pi \right] = \frac{4\pi}{N-1}, \\ \Longrightarrow \Delta\Omega_M &= 2 \times \frac{4\pi}{N-1} = \frac{8\pi}{N-1}, \end{aligned}$$

with

$$\begin{aligned} \Omega_{SL} &\approx \frac{\frac{4\pi}{N-1} + \frac{8\pi}{N-1}}{2} = \frac{6\pi}{N-1}, \\ \Longrightarrow \Gamma = 20\log_{10} \frac{|W(e^{j6\pi/(N-1)})|}{|W(e^{j0})|} &= 20\log_{10} \frac{\frac{2/(N-1)}{|\sin^2(6\pi/(2(N-1)))|}}{(2/(N-1)) \cdot (N-1)^2/4} \\ &\approx 20\log_{10} \frac{\frac{1}{(6\pi/(2(N-1)))^2}}{(N-1)^2/4} = 20\log_{10} \frac{(4(N-1)^2)/(36\pi^2)}{(N-1)^2/4} \\ &= 20\log_{10} \frac{16}{36\pi^2} \approx -27 \text{ [dB]}. \end{aligned}$$



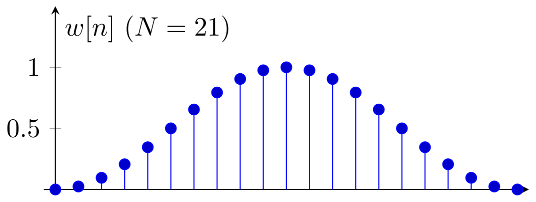


7.6.6 Hann Window

The *Hann window* (or *Hanning window* – a verbing of the name) is defined as

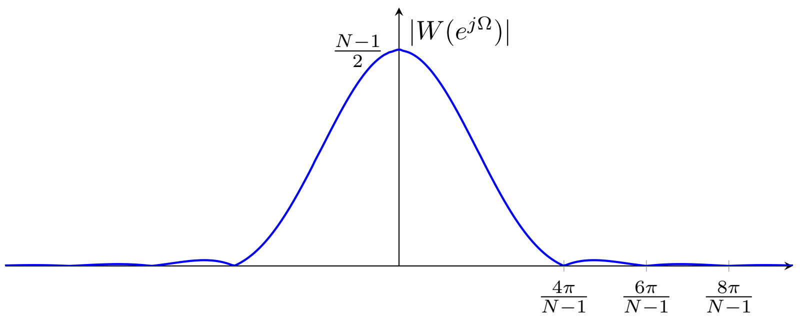
$$w[n] = \begin{cases} \frac{1}{2}\left(1-\cos\left(\frac{2\pi n}{N-1}\right)\right), & n \in [0, N-1] \\ 0, & \text{otherwise} \end{cases} \tag{7.42}$$

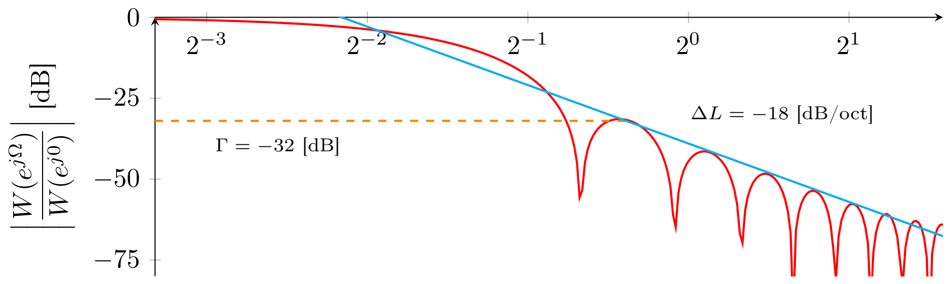
$$\begin{aligned} \Longrightarrow W(e^{j\Omega}) &= 0.5W_R(e^{j\Omega}) - 0.25W_R(e^{j(\Omega-\Omega_0)}) - 0.25W_R(e^{j(\Omega+\Omega_0)}) \\ &= e^{-j\Omega(N-1)/2}\left(0.5\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)} + 0.25\frac{\sin\left(\frac{N(\Omega-\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega-\Omega_0}{2}\right)} + 0.25\frac{\sin\left(\frac{N(\Omega+\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega+\Omega_0}{2}\right)}\right), \\ \text{for } \Omega_0 &= \frac{2\pi}{N-1} \text{ and } W_R(e^{j\Omega}) = e^{-j\Omega(N-1)/2}\left[\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)}\right]. \end{aligned} \tag{7.43}$$



By visual inspection, the first and second nulls happen at $4\pi/(N-1)$ and $6\pi/(N-1)$ such that

$$\begin{aligned} \Delta\Omega_M &= 2 \times \frac{4\pi}{N-1} = \frac{8\pi}{N-1}, \\ \Omega_{SL} &\approx \frac{\frac{4\pi}{N-1} + \frac{6\pi}{N-1}}{2} = \frac{5\pi}{N-1}, \\ \Longrightarrow \Gamma = 20\log_{10}\frac{|W(e^{j5\pi/(N-1)})|}{|W(e^{j0})|} &\approx 20\log_{10}\frac{\left|\frac{0.5}{\sin(5\pi/(2(N-1)))} - \frac{0.25}{\sin(3\pi/(2(N-1)))} - \frac{0.25}{\sin(7\pi/(2(N-1)))}\right|}{(N-1)/2} \\ &\approx 20\log_{10}\frac{\left|\frac{0.5}{5\pi/(2(N-1))} - \frac{0.25}{3\pi/(2(N-1))} - \frac{0.25}{7\pi/(2(N-1))}\right|}{(N-1)/2} \\ &= 20\log_{10}\left|\frac{2}{5\pi} - \frac{1}{3\pi} - \frac{1}{7\pi}\right| \approx -32 \text{ [dB]}. \end{aligned}$$



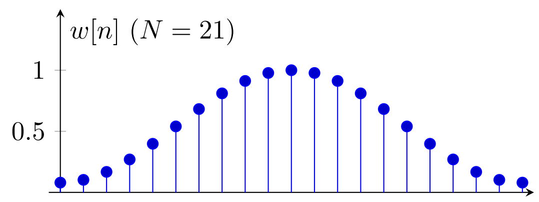


7.6.7 Hamming Window

The *Hamming window* is defined as

$$w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & n \in [0, N-1] \\ 0, & \text{otherwise} \end{cases} \tag{7.44}$$

$$\begin{aligned} \implies W(e^{j\Omega}) &= 0.54W_R(e^{j\Omega}) - 0.23W_R(e^{j(\Omega-\Omega_0)}) - 0.23W_R(e^{j(\Omega+\Omega_0)}) \\ &= e^{-j\Omega(N-1)/2}\left(0.54\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)} + 0.23\frac{\sin\left(\frac{N(\Omega-\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega-\Omega_0}{2}\right)} + 0.23\frac{\sin\left(\frac{N(\Omega+\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega+\Omega_0}{2}\right)}\right), \\ \text{for } \Omega_0 &= \frac{2\pi}{N-1} \text{ and } W_R(e^{j\Omega}) = e^{-j\Omega(N-1)/2}\left[\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)}\right]. \end{aligned} \tag{7.45}$$

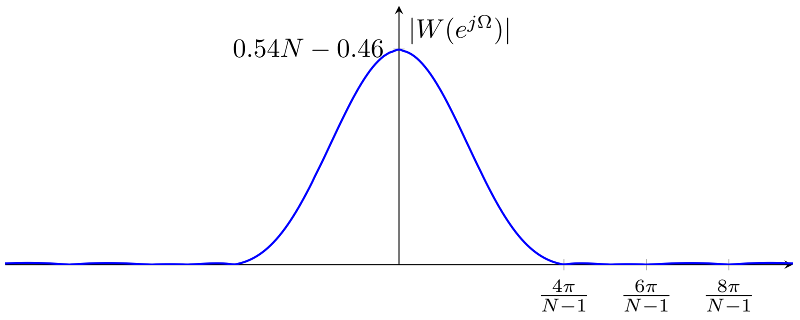


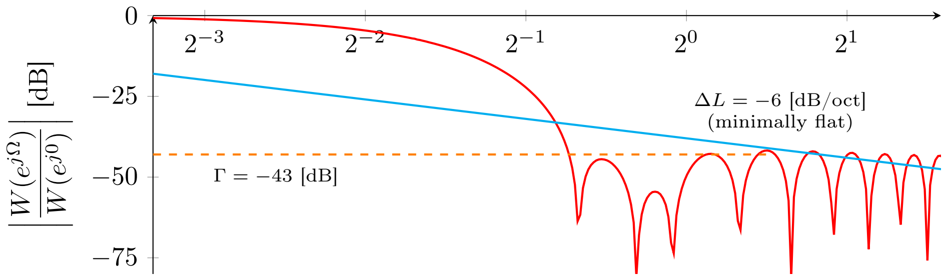
By visual inspection,

$$\Omega_{null_1} \approx \frac{4\pi}{N-1} \implies \Delta\Omega_M \approx 2 \times \frac{4\pi}{N-1} = \frac{8\pi}{N-1}.$$

However, the peak sidelobe actually does not occur at the first sidelobe for the Hamming window, hence the approximation between the first and second nulls would not work here for sidelobe attenuation. Instead, comparing to the nulls of the Hann window, we choose the point $9\pi/(N-1)$ as a neat ballpark estimate of the peak sidelobe location such that

$$\Gamma = 20\log_{10}\frac{|W(e^{j9\pi/(N-1)})|}{|W(e^{j0})|} = 20\log_{10}\frac{\left|\frac{0.54}{\sin(9\pi/(2(N-1)))} - \frac{0.23}{\sin(7\pi/(2(N-1)))} - \frac{0.23}{\sin(11\pi/(2(N-1)))}\right|}{0.54N-0.46}$$
$$\approx 20\log_{10}\frac{\left|\frac{0.54}{9\pi/(2(N-1))} - \frac{0.23}{7\pi/(2(N-1))} - \frac{0.23}{11\pi/(2(N-1))}\right|}{0.54(N-0.85)}$$
$$\approx 20\log_{10}\left|\frac{2}{9\pi} - \frac{0.85}{7\pi} - \frac{0.85}{11\pi}\right| \approx -43\ [\text{dB}].$$





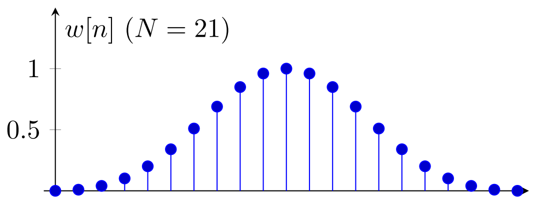
7.6.8 Blackman Window

The *Blackman window* is defined as

$$w[n] = \begin{cases} 0.42 - 0.50\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right), & n \in [0, N-1] \\ 0, & \text{otherwise} \end{cases} \tag{7.46}$$

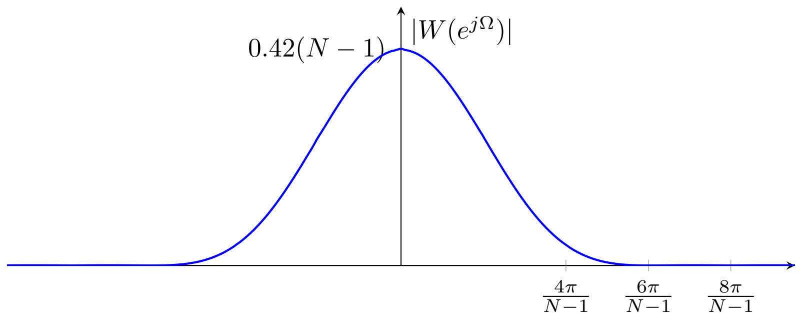
$$\implies W(e^{j\Omega}) = 0.42W_R(e^{j\Omega}) - 0.25[W_R(e^{j(\Omega-\Omega_0)}) + W_R(e^{j(\Omega+\Omega_0)})]$$
$$+ 0.04[W_R(e^{j(\Omega-2\Omega_0)}) + W_R(e^{j(\Omega+2\Omega_0)})]$$
$$= e^{-j\Omega(N-1)/2}\left(0.42\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)} + 0.25\frac{\sin\left(\frac{N(\Omega-\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega-\Omega_0}{2}\right)} + 0.25\frac{\sin\left(\frac{N(\Omega+\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega+\Omega_0}{2}\right)}\right.$$
$$\left. + 0.04\frac{\sin\left(\frac{N(\Omega-2\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega-2\Omega_0}{2}\right)} + 0.04\frac{\sin\left(\frac{N(\Omega+2\Omega_0)}{2}\right)}{\sin\left(\frac{\Omega+2\Omega_0}{2}\right)}\right),$$

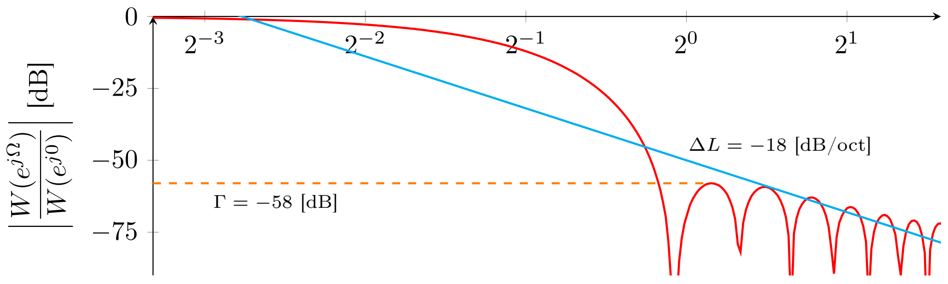
$$\text{for } \Omega_0 = \frac{2\pi}{N-1} \text{ and } W_R(e^{j\Omega}) = e^{-j\Omega(N-1)/2}\left[\frac{\sin\left(\frac{N\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)}\right]. \tag{7.47}$$



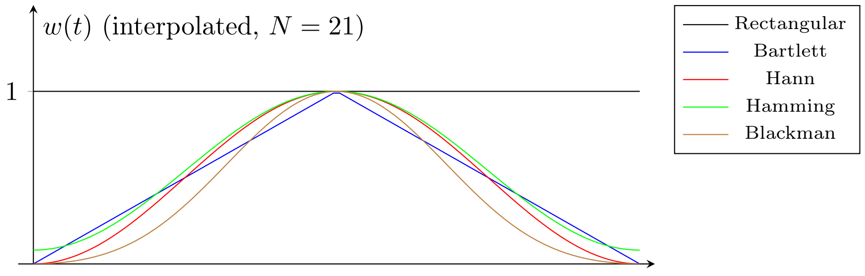
By visual inspection, the peak sidelobe happens between first and (technically) third nulls $6\pi/(N-1)$ and $8\pi/(N-1)$. Then

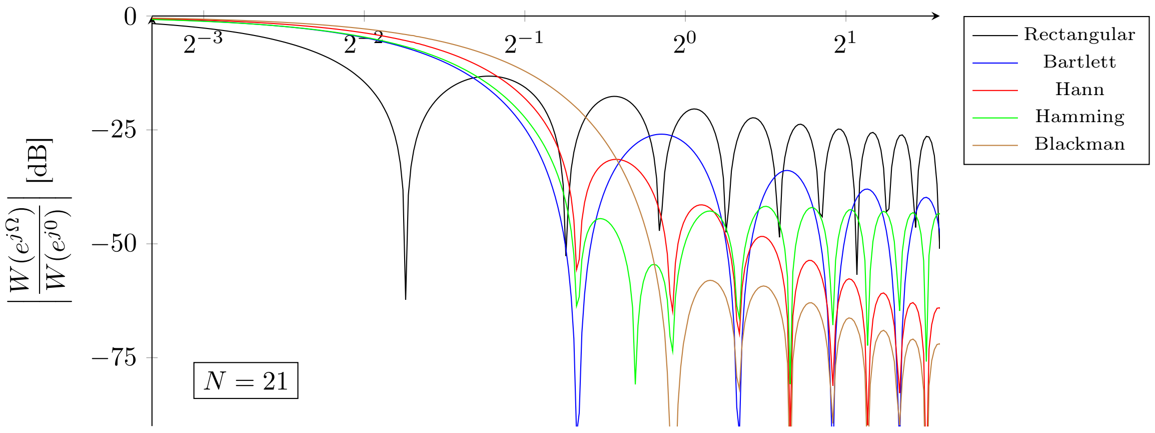
$$\begin{aligned}
\Delta\Omega_M &= 2 \times \frac{6\pi}{N-1} = \frac{12\pi}{N-1},\\
\Omega_{SL} &\approx \frac{\frac{6\pi}{N-1} + \frac{8\pi}{N-1}}{2} = \frac{7\pi}{N-1},\\
\implies \Gamma &= 20\log_{10}\frac{|W(e^{j7\pi/(N-1)})|}{|W(e^{j0})|} = 20\log_{10}\frac{|W(e^{j7\pi/(N-1)})|}{0.42N - 0.5 + 0.08}\\
&\approx 20\log_{10}\frac{\left|-\frac{0.42}{7\pi/(2(N-1))} + \frac{0.25}{5\pi/(2(N-1))} + \frac{0.25}{9\pi/(2(N-1))} - \frac{0.04}{3\pi/(2(N-1))} - \frac{0.04}{11\pi/(2(N-1))}\right|}{0.42(N-1)}\\
&\approx 20\log_{10}\left|-\frac{2}{7\pi} + \frac{1.19}{5\pi} + \frac{1.19}{9\pi} - \frac{0.19}{3\pi} - \frac{0.19}{11\pi}\right| \approx -58\ [\text{dB}].
\end{aligned}$$





Putting all five normalized windows on the same plot, we can now see how some windows are more advantageous than others.





7.7 Frequency Analysis using Python

Here, we provide an example of using Python to perform frequency analysis on a sampled sine wave and using spectral peak detection to find the frequency of the sine wave. This is all done in Jupyter notebook, as seen in the next few pages.

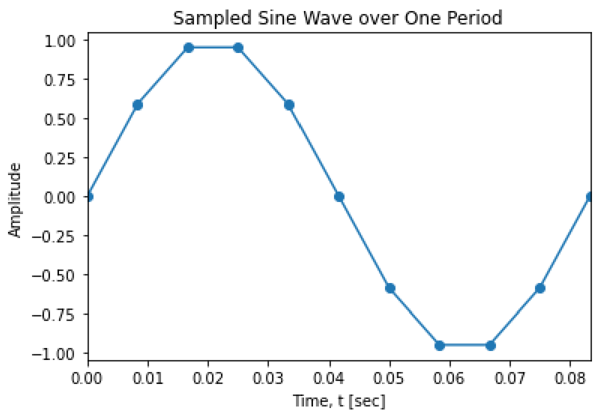
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import time
```

Single-Frequency Tone (sampled at integer multiple)

Suppose an analog sine wave with frequency $freq$ is being sampled at sampling rate fs. We know that the DTFT is two spectral lines at $\pm freq$.

If fs is a multiple of $freq$ such that letting frame size $N = \dfrac{fs}{freq}$ represents one cycle being sampled, then $freq$ will be detected.

In [2]:
```
# SIMULATE SAMPLING A SINE WAVE

freq = 12                       # analog signal frequency = 12 Hz
fs = 120                        # sampling rate = 120 samp/sec
Ts = 1/fs                       # sampling period
n = np.arange(0,1000)           # array of (DT) indices, n
t = n*Ts                        # array of (CT) time points corresponding to n
x_t = np.sin(2*np.pi*freq*t)    # analog signal sampled at fs

# PLOT SAMPLED WAVE OVER ONE PERIOD

plt.plot(t,x_t, marker='o')
plt.xlim([0,1/freq])
plt.title('Sampled Sine Wave over One Period')
plt.xlabel('Time, t [sec]')
plt.ylabel('Amplitude')
plt.show()
```



Window One Cycle of Sampled Wave

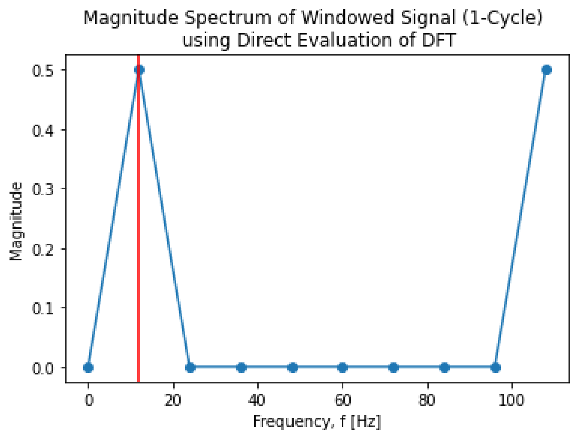
First, we compute the DFT by using the matrix form $X = Wx$. Directly evaluating the equation gives a time complexity of $O(N^2)$.

In the plot below, the blue curve depicts the magnitude spectrum of the windowed sine wave that was sampled, whereas the red vertical line represents the frequency $freq$ of the sine wave.

In [3]:
```
# WINDOW ONE PERIOD OF SAMPLED WAVE

t_cycle = t[t < 1/freq]                 # time points corresponding to one period of sampled wave
x_t_cycle = x_t[:len(t_cycle)]          # one period of sampled wave
w = np.ones_like(t_cycle)               # rectangular window, w
N = len(t_cycle)                        # frame size N

# COMPUTE N-POINT DFT

f = np.arange(0, N) * fs / N            # array of CT frequencies, f

t0 = time.time()
kn = np.array([[(k*n) \
        for k in np.arange(0,N)] \
            for n in np.arange(0,N)])
W_twiddle = np.exp(-1j * 2*np.pi * kn / N)  # matrix of twiddle factors
X_f = W_twiddle @ (x_t_cycle * w)           # N-point DFT of windowed signal
```

```
dft_elapsed = time.time()-t0
print(f"DFT elapsed time: {dft_elapsed:.6f} seconds")

rect_normz = 1 / len(t_cycle)                # normalization factor for spectrum of rect-windowed signal

plt.plot(f, abs(X_f) * rect_normz, marker='o')
plt.title('Magnitude Spectrum of Windowed Signal (1-Cycle) \n using Direct Evaluation of DFT')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f[np.argmax(abs(X_f))]):.3f} [Hz]")
```

```
DFT elapsed time: 0.000470 seconds
```



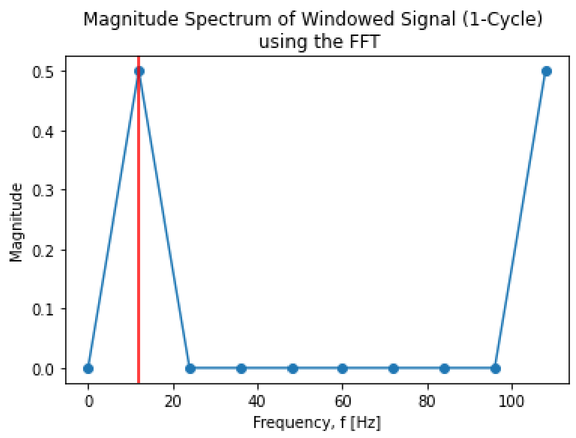
```
Peak frequency: 12.000 [Hz]
```

Notice that since fs is a multiple of $freq$ and one cycle is being sampled, we can pick up $freq$ precisely.

Now, we introduce the Fast Fourier Transform (FFT), which is designed to compute the DFT in less time. For most cases, the time complexity is $O(N \log_2 N)$.

In [4]:
```
# WINDOW ONE PERIOD OF SAMPLED WAVE

t_cycle = t[t < 1/freq]                 # time points corresponding to one period of sampled wave
x_t_cycle = x_t[:len(t_cycle)]          # one period of sampled wave
w = np.ones_like(t_cycle)               # rectangular window, w
N = len(t_cycle)                        # frame size N

# COMPUTE N-POINT DFT

f = np.arange(0, N) * fs / N            # array of CT frequencies, f
t0 = time.time()
X_f = np.fft.fft(x_t_cycle * w, N)      # N-point DFT of windowed signal using FFT

fft_elapsed = time.time()-t0
print(f"FFT elapsed time: {fft_elapsed:.6f} seconds")
rect_normz = 1 / len(t_cycle)           # normalization factor for spectrum of rect-windowed signal

# PLOT MAGNITUDE SPECTRUM

plt.plot(f, abs(X_f) * rect_normz, marker='o')
plt.title('Magnitude Spectrum of Windowed Signal (1-Cycle) \n using the FFT')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f[np.argmax(abs(X_f))]):.3f} [Hz]")
```

```
FFT elapsed time: 0.000221 seconds
```



```
Peak frequency: 12.000 [Hz]
```

Here, we get the same answer as the direct evaluation of the DFT, but in less time. One can see that $\frac{O(N^2)}{O(N\log_2 N)} \approx O\left(\frac{N}{\log_2 N}\right)$.

For $N = 10$, it follows that $\frac{N}{\log_2 N} = \frac{10}{\log_2 10} \approx 3.01$.

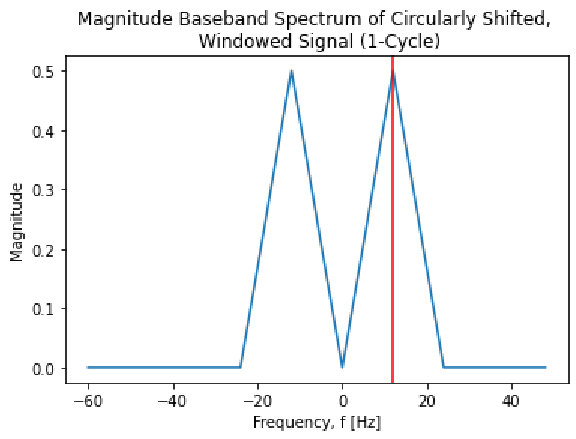
In [5]:
```
print(f"Ratio of DFT-to-FFT elapsed times: {dft_elapsed / fft_elapsed:.6f}")
```

```
Ratio of DFT-to-FFT elapsed times: 2.123922
```

One can see that as N increases, it becomes more ideal to use FFT rather than direct evaluation. For the rest of the text, we'll use the FFT to compute the DFT of a sequence.

Now we want to circularly shift half of the spectrum such that we obtain the baseband spectrum symmetric about the vertical axis.

In [6]:
```
# CIRCULARLY SHIFT TO GET SYMMETRIC MAGNITUDE SPECTRUM

if N % 2 == 0:
    f_shift = np.arange(-N/2, N/2) * fs / N
    X_f_shift = np.hstack((X_f[int(N/2):], X_f[:int(N/2)]))
else:
    f_shift = np.linspace(-(N-1)/2, (N-1)/2, N) * fs / N
    X_f_shift = np.hstack((X_f[int((N+1)/2):], X_f[:int((N+1)/2)]))

plt.plot(f_shift, abs(X_f_shift) * rect_normz)
plt.title('Magnitude Baseband Spectrum of Circularly Shifted, \n Windowed Signal (1-Cycle)')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f_shift[np.argmax(abs(X_f_shift))]):.3f} [Hz]")
```

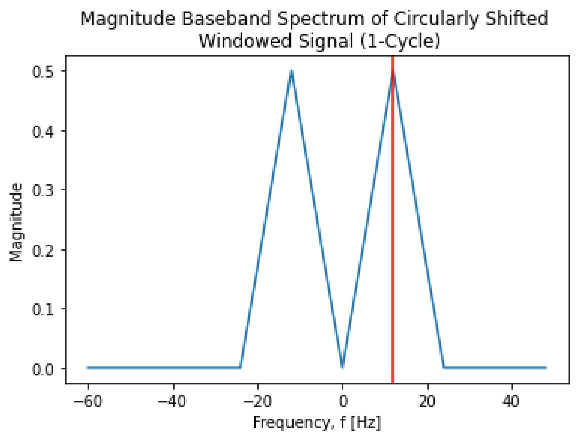


```
Peak frequency: 12.000 [Hz]
```

The numpy package in Python has built-in functions **fftshift** and **fftfreq** that will take care of the circular shifting.

In [7]:

```
f_shift = np.fft.fftshift(np.fft.fftfreq(N, d=Ts))
X_f_shift = np.fft.fftshift(np.fft.fft(x_t_cycle * w, N))

plt.plot(f_shift, abs(X_f_shift) * rect_normz)
plt.title('Magnitude Baseband Spectrum of Circularly Shifted \n Windowed Signal (1-Cycle)')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f_shift[np.argmax(abs(X_f_shift))]):.3f} [Hz]")
```



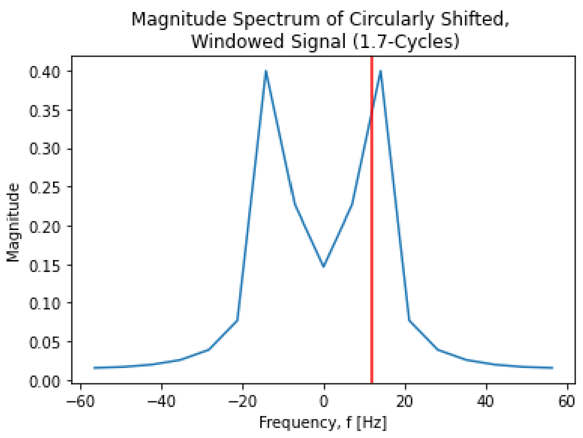
```
Peak frequency: 12.000 [Hz]
```

Window Non-Integer Cycles of Sampled Wave

If a non-integer number of cycles is windowed, then $freq$ cannot be precisely detected.

In [8]:

```
# WINDOW NON-INTEGER PERIOD OF SAMPLED WAVE

t_cycle = t[t < 1.7/freq]               # time points corresponding to 1.7 periods of sampled wave
x_t_cycle = x_t[:len(t_cycle)]          # one period of sampled wave
w = np.ones_like(t_cycle)               # rectangular window, w
N = len(t_cycle)                        # frame size N

# CIRCULARLY SHIFT TO GET SYMMETRIC MAGNITUDE SPECTRUM

f_shift = np.fft.fftshift(np.fft.fftfreq(N, d=Ts))
X_f_shift = np.fft.fftshift(np.fft.fft(x_t_cycle * w, N))
rect_normz = 1 / len(t_cycle)

plt.plot(f_shift, abs(X_f_shift) * rect_normz)
plt.title('Magnitude Spectrum of Circularly Shifted, \n Windowed Signal (1.7-Cycles)')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f_shift[np.argmax(abs(X_f_shift))]):.3f} [Hz]")
```

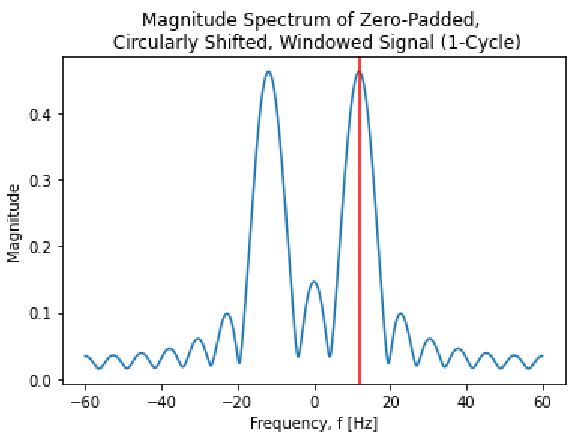


```
Peak frequency: 14.118 [Hz]
```

To remedy that as best as we can, increase the frame size N by zero-padding. In doing so, we also begin to smooth out the spectrum. Here, we can see the DTFT of a sine wave being convolved with the Dirichlet kernel, the effect of windowing a signal.

In [9]:

```
# APPLY ZERO PADDING

N = 2048

f_pad = np.fft.fftshift(np.fft.fftfreq(N, d=Ts))
X_f_pad = np.fft.fftshift(np.fft.fft(x_t_cycle * w, N))

plt.plot(f_pad, abs(X_f_pad) * rect_normz)
plt.title('Magnitude Spectrum of Zero-Padded, \n Circularly Shifted, Windowed Signal (1-Cycle)')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f_pad[np.argmax(abs(X_f_pad))]):.3f} [Hz]")
```



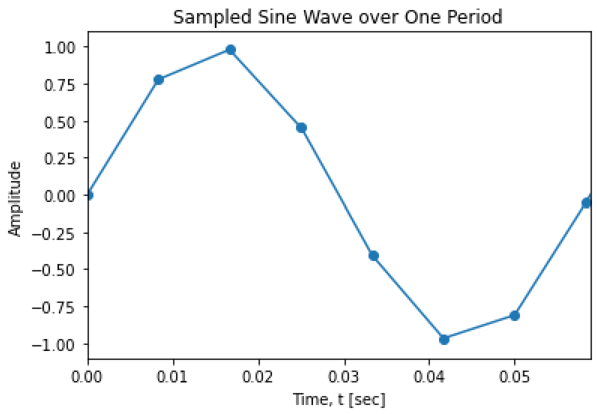
```
Peak frequency: 11.836 [Hz]
```

Single-Frequency Tone (sampled at non-integer multiples)

Suppose one cycle is collected, but fs is not a multiple of $freq$. Then $freq$ cannot be detected.

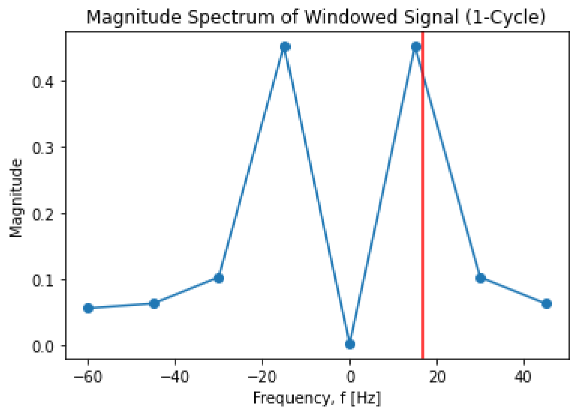
In [10]:

```
# SIMULATE SAMPLING A SINE WAVE

freq = 17                          # analog signal frequency = 17 Hz
fs = 120                           # sampling rate = 120 samp/sec
Ts = 1/fs                          # sampling period
n = np.arange(0,1000)              # array of (DT) indices, n
t = n*Ts                           # array of (CT) time points corresponding to n
x_t = np.sin(2*np.pi*freq*t)       # analog signal sampled at fs

# PLOT SAMPLED WAVE OVER ONE PERIOD

plt.plot(t,x_t, marker='o')
plt.xlim([0,1/freq])
plt.title('Sampled Sine Wave over One Period')
plt.xlabel('Time, t [sec]')
plt.ylabel('Amplitude')
plt.show()
```



Window One Cycle of Sampled Wave

In [11]:
```
# WINDOW ONE PERIOD OF SAMPLED WAVE

t_cycle = t[t < 1/freq]                # time points corresponding to one period of sampled wave
x_t_cycle = x_t[:len(t_cycle)]         # one period of sampled wave
w = np.ones_like(t_cycle)              # rectangular window, w
N = len(t_cycle)                       # frame size N

# COMPUTE N-POINT DFT

f = np.fft.fftshift(np.fft.fftfreq(N, d=Ts))
X_f = np.fft.fftshift(np.fft.fft(x_t_cycle * w, N))
rect_normz = 1 / len(t_cycle)

# PLOT MAGNITUDE SPECTRUM

plt.plot(f, abs(X_f) * rect_normz, marker='o')
plt.title('Magnitude Spectrum of Windowed Signal (1-Cycle)')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f[np.argmax(abs(X_f))]):.3f} [Hz]")
```

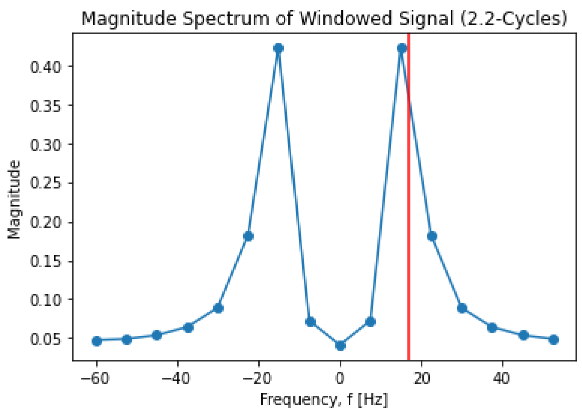


```
Peak frequency: 15.000 [Hz]
```

Window Non-Integer Cycles of Sampled Wave

In [12]:
```
# INCREASE NUMBER OF CYCLES

t_cycle = t[t < 2.2/freq]              # time points corresponding to 2.2 periods of sampled wave
x_t_cycle = x_t[:len(t_cycle)]         # one period of sampled wave
w = np.ones_like(t_cycle)              # rectangular window, w
N = len(t_cycle)                       # frame size N
```

```
# COMPUTE N-POINT DFT

f = np.fft.fftshift(np.fft.fftfreq(N, d=Ts))
X_f = np.fft.fftshift(np.fft.fft(x_t_cycle * w, N))
rect_normz = 1 / len(t_cycle)

# PLOT MAGNITUDE SPECTRUM

plt.plot(f, abs(X_f) * rect_normz, marker='o')
plt.title('Magnitude Spectrum of Windowed Signal (2.2-Cycles)')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f[np.argmax(abs(X_f))]):.3f} [Hz]")
```

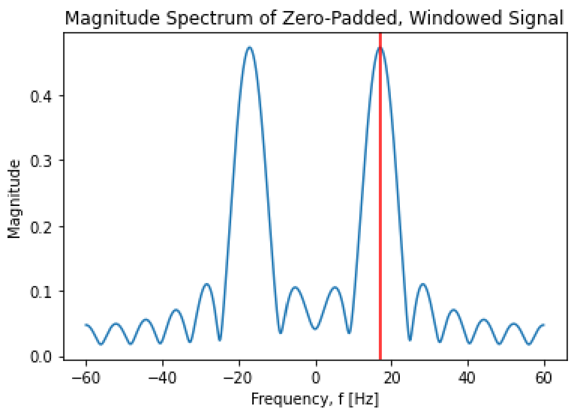


```
Peak frequency: 15.000 [Hz]
```

We can get a more accurate spectrum for a more accurate peak detection by zero-padding. In numpy, this is done by specifying a different N in **fft**, larger than the length of the windowed sequence.

In [13]:

```
# APPLY ZERO PADDING

N = 2048

f_pad = np.fft.fftshift(np.fft.fftfreq(N, d=Ts))
X_f_pad = np.fft.fftshift(np.fft.fft(x_t_cycle * w, N))
rect_normz = 1 / len(t_cycle)

plt.plot(f_pad, abs(X_f_pad) * rect_normz)
plt.title('Magnitude Spectrum of Zero-Padded, Windowed Signal')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f_pad[np.argmax(abs(X_f_pad))]):.3f} [Hz]")
```

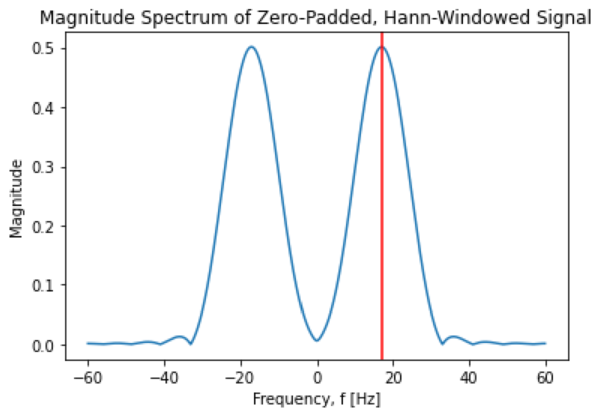


```
Peak frequency: 17.051 [Hz]
```

Using a Non-Rectangular Window

Different windows have different effects on mainlobe width and sidelobe roll-off. Here, we use one of the most common non-rectangular windows: the Hann window. The window is applied first before zero-padding.

In [14]:
```
# APPLY HANNING, then ZERO PAD

w = np.hanning(len(t_cycle))

N = 2048

f_pad = np.fft.fftshift(np.fft.fftfreq(N, d=Ts))
X_f_pad = np.fft.fftshift(np.fft.fft(x_t_cycle * w, N))
hann_normz = 1 / ((len(t_cycle) - 1) / 2) # normalization factor for spectrum of Hann-windowed signal

plt.plot(f_pad, abs(X_f_pad) * hann_normz)
plt.title('Magnitude Spectrum of Zero-Padded, Hann-Windowed Signal')
plt.xlabel('Frequency, f [Hz]')
plt.ylabel('Magnitude')
plt.axvline(x=freq, color='r')
plt.show()

# PRINT FREQ WHERE PEAK
print(f"Peak frequency: {abs(f_pad[np.argmax(abs(X_f_pad))]):.3f} [Hz]")
```



```
Peak frequency: 17.051 [Hz]
```

In [ ]: