Intuitive Guide to Fourier Analysis

Charan Langton
Victor Levin

Much of this book relies on math developed by important persons in the field over the last 200 years. When known or possible, the authors have given the credit due. We relied on many books and articles and consulted many articles on the internet and often many of these provided no name for credits. In this case, we are grateful to all who make the knowledge available free for all on the internet.

mntcastle@comcast.net

Website for this book: complextoreal.com/fftbook

Copyright 2016 Charan Langton and Victor Levin
ISBN- 13: 978-0-913063-26-2

6 | Discrete Fourier Transform (DFT)

Leopold Kronecker
7 December 1823 – 29 December 1891

Leopold Kronecker was a German mathematician who worked on number theory and algebra. He was quoted as having said, "God made the integers, all else is the work of man." An important part of Kronecker's research focused on number theory and algebra. In an 1853 paper on the theory of equations and Galois theory he formulated the Kronecker–Weber theorem, without however offering a definitive proof (the theorem was proved completely much later by David Hilbert). Also named for Kronecker are the Kronecker limit formula, Kronecker's congruence, Kronecker delta, Kronecker comb functions. –From Wikipedia

In Chapter 5, we discussed the Fourier transform of discrete signals, called the DTFT, for both aperiodic and periodic signals. The DTFT is a theoretical concept. It is not user-friendly, and it requires integration, something which neither we nor our computers do well. In this chapter we move from theory to reality. We discuss a modification of the DTFT which allows us to do Fourier transforms on finite length signals. The result, we note happily is a spectrum that is discrete. In fact the DFT is the most used of all signal processing tools. It is a useful and efficient tool for spectral analysis of real signals.

The four cases of Fourier transform

Let's review the two Fourier transforms, CTFT and DTFT for both the periodic and aperiodic signals.

Case 1- CTFT of continuous-time, periodic signal

When the input signal is continuous-time and periodic with period T_0, the spectrum (CTFT) is aperiodic (meaning it does not repeat.) and discrete. Each component has a frequency resolution of $2\pi/T_0$ as shown in 6.1.

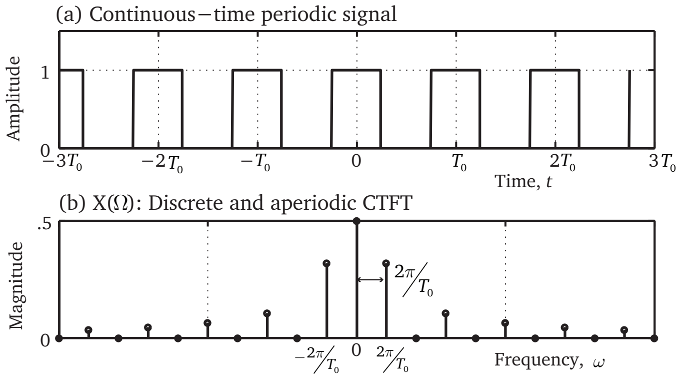

Figure 6.1: Case 1- CTFT of continuous-time, periodic signal

Case 2 - CTFT of continuous-time, aperiodic signal

When the input signal is continuous-time and non-periodic, the spectrum(CTFT) is aperiodic as in Case 1 but unlike case 1 is continuous in frequency. The x-axis of both case 1 and 2 is in terms of frequency, measured in Hz or radians per second.

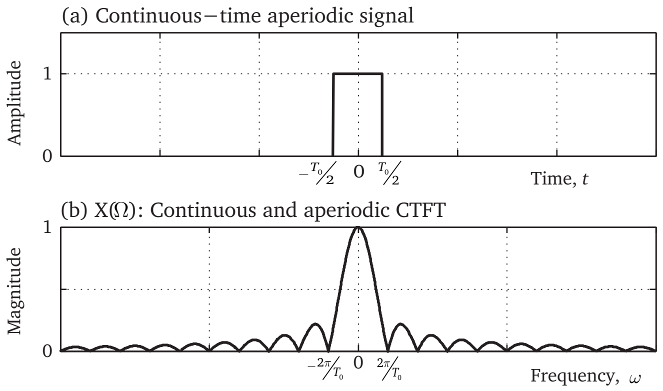

Figure 6.2: Case 2 - CTFT of continuous-time, aperiodic signal

Case 3 - DTFT of discrete-time, periodic signal

When the input signal is discrete-time and periodic with period N_0 samples, the spectrum (DTFT) is periodic (meaning it repeats) and is discrete in frequency. Each component has a frequency resolution of $2\pi/N_0$ as shown in 6.3.

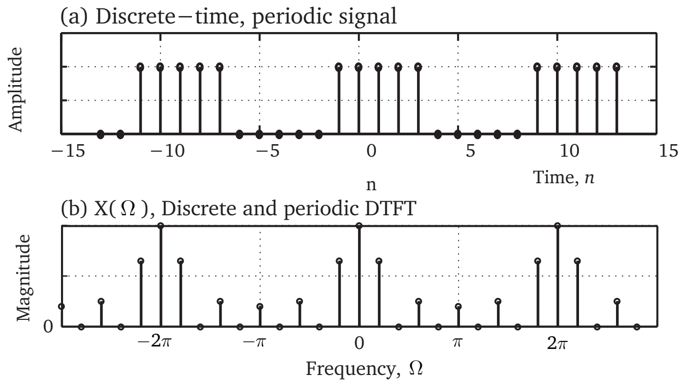

Figure 6.3: Case 3 - DTFT of discrete-time, periodic signal

Case 4 - DTFT of discrete-time, aperiodic signal

When the input signal is discrete-time but aperiodic, the spectrum (DTFT) is periodic or repeating, but is continuous in frequency. The x-axis of both case 3 and 4 is in terms of the digital frequency of range $-\pi$ to $+\pi$ radians.

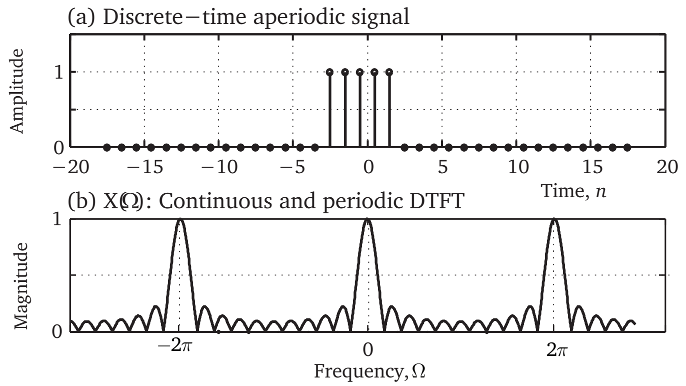

Figure 6.4: Case 4 - DTFT of discrete-time, aperiodic signal

Of these cases, Case 4 input signal fits closest to our needs. In real life we often have just a part of a signal, i.e. a finite number of samples, hence a signal we consider aperiodic. But we do not want a continuous spectrum. The DTFT would require a lot of math and we don't want to do that. This is where the DFT comes in. Since the DTFT of such a signal is continuous, can't we just sample it and get a discrete version of this spectrum? Yes, that is possible. However, the catch is that we would have to actually compute the DTFT first, so that's no good. We like the spectrum of Case 3, which is indeed discrete. Maybe there is away to combine these two. We find a compromise. We take the signal, and call all its samples, its period. Hence the length of the signal is presumed to be its period. Now the aperiodic signal becomes a periodic signal of period N. The DTFT as we see in Case 3, then is discrete.

The Discrete Fourier transform (DFT)

The Discrete Fourier Transform (DFT) borrows elements from both the discrete Fourier series and the Fourier transform. It might have had a better name such as Finite Length Fourier Transform (FLFT), but even that is confusing. So we are stuck with DFT, where it is not clear why the T from DTFT has been dropped.

The DFT however has some really great properties.

1. We can do a DFT on any arbitrary but finite-length, uniformly sampled signal.
2. We don't need to know the period of the signal, only the sampling frequency.
3. The DFT is easy to do in software.
4. The DFT is *discrete*.

The DFT as we shall see through examples, is not some second-rate estimate of the real thing but is an exact sampling of the DTFT at uniformly spaced frequencies. The signal itself can be either periodic or aperiodic, the DFT will give valid results for both. You may think of DFT as a DTFT with a coarser resolution. Conversely, doing a DFT with fine resolution allows computation of the DTFT without actually resorting to integration.

DFT from DTFT

To develop the DFT, we start with the definition of the DTFT.

$$X(\Omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n} \tag{6.1}$$

The DTFT is specified by the notation $X(\Omega)$. While the input signal $x[n]$ is discrete, the DTFT is a continuous function of frequency. The time index n, represents a fixed amount of time between each sample, called the sampling time, T_s. The inverse of this time is the sampling frequency, F_s of the signal. Note that a discrete signal has a sampling frequency even if the signal is not actually sampled.

Let's assume that we have a finite length signal $x[n]$ that exists only within $0 \leq n \leq N-1$ points. We don't know anything about this signal outside of the N samples. We also see no particular periodicity in the samples collected. If $X(\Omega)$ is the DTFT of this signal, then we can obtain N samples of the DTFT (which has an infinite number of points) by sampling the DTFT at N uniformly spaced frequencies. Note that Ω ranges from $0 \leq \Omega \leq 2\pi$. Hence if we are to sample it at k points, the frequency at these points is given by

$$\Omega_k = \frac{2\pi k}{N} \tag{6.2}$$

If we substitute this definition of Ω into Eq. (6.1), we get the definition of a sampled DTFT, or what we call the DFT.

$$\boxed{X[k] = X(k\Omega_0) = \sum_{n=-\infty}^{\infty} x[n]e^{j\frac{2\pi}{N}kn}} \tag{6.3}$$

The DFT is denoted by $X[k]$, where k is the k-th harmonic. The index k ranges from 0 to $(N-1)$ for a count of N. There are a total of N of these harmonic frequencies in this summation, quite unlike the DTFT where the harmonic index k is continuous. In fact it is not even present in its formulation in Eq. (6.1). In a DFT, the the index k steps through these N frequencies over the full 2π range of the DTFT. The index k is also referred to as the **bin number**, a name by which we might imagine a bucket in which we are gathering a quantity of the signal. Note that the term bin is used only in frequency domain. Time samples are never called bins.

DFT frequency resolution

What is N in Eq. (6.2)? This number is meant to be the period of the signal in samples. But what if we do not know the period of the signal? If we know it, thats great. Else, we take all the samples at hand and call that the period. Hence lacking knowledge of the period N, the total number of samples collected, N is set equal to the period of the signal by the DFT. The DFT presumes that samples used for the analysis form a true basic cycle. However if the number of samples used for the analysis is not equal to the true period, we have problems. (More about this later in the chapter.) However, for now let's assume, N is indeed equal to the period of the signal. We divide 2π, that being the range of the digital frequency Ω by this N and that becomes our fundamental frequency. This is the frequency resolution of the DFT. The integer multiples of this frequency become the N harmonics of the fundamental. These N frequencies are the finite basis set for the DFT, exactly the same concept we discussed for discrete Fourier series in Chapter 3. We evaluate the DTFT **only** at these N frequencies. No more and no less.

We can write the fundamental and the harmonic frequencies as

$$\text{Fundamental frequency: } \Omega_0 = \frac{2\pi k}{N},\ k = 1 \tag{6.4}$$

$$\text{Each harmonic frequency: } \Omega_k = \frac{2\pi k}{N},\ k = 2,3,\ldots,N-1 \tag{6.5}$$

N, the DFT size can be the number of samples collected or it can be even larger than the number collected, through a process called zero-padding. Clearly larger the N, the smaller(better) the frequency resolution per Eq. (6.5), which is a good thing.

The term ***frequency resolution*** can also be written as in Eq. (6.6) in terms of the sampling frequency. Here F_s is the sampling frequency in samples per second, or Hz. These two terms, F_s and N are independent. We increase the sampling frequency to improve the resolution and N to increase plotting density. Increasing both by the same factor is usually not helpful. The frequency resolution in Hz is defined by

$$R_s = \frac{F_s}{N} \tag{6.6}$$

The inverse DFT (iDFT) is given by

$$\boxed{x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi}{N}kn}} \tag{6.7}$$

The indexes k and n in the DFT equation are definitely confusing. Some books use n for the harmonic index and k for the time index. The thing to remember is that they are both equal in length to N. If we have N time samples, then they are referred to by n as the time index. Out of the forward DFT $X[k]$, we get N harmonics, even though we refer to the individual harmonics with index, k. Since the range of both indexes is N, the confusion often may still get you the correct answer!

The difference between the DFT and its inverse, iDFT is just a scaling term in front of the iDFT and a change of sign to the exponent. Both of these formulations, if not easy to understand, are at least easy to compute. The normalization by N takes the N additions and then normalizes it, so the DFT is a measure of relative magnitudes only, just like the DTFT. The two expressions are very similar and this makes the hardware/software implementation of the DFT algorithm easy.

Understanding the x-axis of a DFT

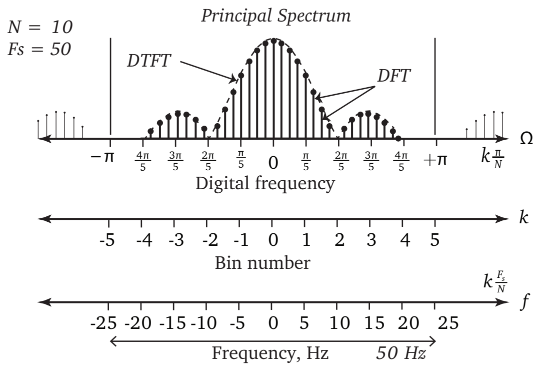

Figure 6.5: What the x-axis of a DFT means. (a) Measured from $-\pi$ to $+\pi$, (b) Specified by bin number from $-\frac{N-1}{2}$ to $+\frac{N-1}{2}$, with $N =$ DFT size, (c) Frequency, $k2F_s/N$, F_s, the sampling frequency in samples per second.

You will see three different ways of numbering/naming the x-axis of a DFT. In Fig. 6.5, we see a typical DFT plot. We note that the DFT components shown as stems are discrete. The DTFT is shown also as a dashed line for reference. Although the y-axis is always given either as amplitude or magnitude (linear or in dBs), the DFT x-axis can be specified three different ways.

The first way is by the digital frequency. Each component occurs at a specific harmonic frequency. The range of the DFT harmonics in terms of the digital frequency is always 2π. If the spectrum (DFT) is centered at 0, then the range of frequency is $-\pi \leq \Omega \leq \pi$. If there are N data points, then each tick mark (bin) differs from the next by $2\pi/N$ radians. This method of marking the x-axis is the most confusing. This is because digital frequency has no time dimension.

The second way is to plot the DFT as a direct function of the bin number, k. If there are N samples, then k is shown from 0 to (N-1), or more often from $-N/2$ to $N/2$.

The third way is in terms of the *sampling frequency*. The sampling frequency does have a time-dimension, so using it, we convert a DFT to a particular real frequency range. If the sampling frequency of the signal is F_s samples per second, or Hz, then we can presume that the signal does not contain any frequencies higher than that. Hence the range of the DFT is 0 to F_s or from $-F_s/2$ to $+F_s/2$ Hz. Each value of k is equivalent to $\frac{k}{N}F_s$ Hz frequency or bin size. Hence at extremities, k is equal to $N/2$, from which we get the edge frequency $= \frac{N}{2}\frac{F_s}{N}$ or $f = F_s/2$. In many software packages, the range is often given just as $-1/2$ to $+1/2$ by setting $F_s = 1$. This is to make the range independent of the sampling frequency.

Example 6.1. A signal is sampled at the rate of 100 Hz and is band-limited to 25 Hz. The number of samples is 50.

1. What is the bin spacing?
2. What frequency is represented by bin number 20 and 40?

The frequency resolution is F_s / N. The DFT size or N is equal to 50. Each bin hence represents a bandwidth of $(100/50) =$ of 2 Hz. Bin 20 represents 40 Hz and bin 40 represents 80 Hz.

What if this signal was sampled at the rate of 25 Hz instead of 100? What would be the frequency at these bins then? Sampling frequency of 25 Hz is less than the Nyquist rate (needs to be 50 Hz because the signal has a bandwidth of 25 Hz) and we get aliasing. The spectral resolution is 0.5 Hz so for bin 20, the frequency is 10 Hz, and is 20 Hz for bin 40. This is beyond the ± 12.5 Hz range of the DFT, so at bin 20 we will get aliases from the adjacent spectrum value and the answer would not be correct.

Periodic signal assumption

DFT, like the DTFT assumes that the samples belong to a periodic signal, even when we say that the Fourier transform allows an aperiodic signal. This is a very confusing point. This is because even for the aperiodic case, the signal is still assumed to be periodic by setting $N = \infty$. What we are calling aperiodic is still actually periodic for the Fourier transform.

Let's examine the samples shown in Fig. 6.6(a). We hand these samples to the DFT machine for analysis. The DFT thinks that the points are one cycle of a periodic signal. It thinks that even though we have given it only a limited number of samples, they are actually coming from a big signal with infinite cycles of data, as in Fig. 6.6(b) and (d). It does not know the truth. It has no idea that we are fooling it, and that we actually have no idea what is outside of what we have collected. Periodic? We don't know. Probably not. If there are zeros in the samples collected, it thinks that they also are part of the cycle. If the data has repeating patterns of pulses, it thinks they are also part of the period. The whole of the samples collected are presumed to be one period and hence the length of the signal becomes the period to the signal.

The value of N for a DFT is the length of the signal offered to it including non-zero data points. If we give it signal [1 1 0 0] then it accepts the last two zeros as part of the cycle. The length of the signal for the first set in Fig. 6.6(a) is 10 samples and includes the two zeros on each side. The DFT includes these as part of the signal, N. The DTFT however would have ignored those end zeros.The DTFT cares only about non-zero points. Note although we offer the DFT pretty much the same samples (set 1 or set 2), the DFT has different ideas. We need to be mindful of what the DFT is assuming. It thinks those zeros you gave it for set 1 are real data, hence the first signal when concatenated by the DFT as in (c) looks different than one in (d) although we may think this is a trivial point. How can zeros appended to a signal have any important. But they do, as we shall see.

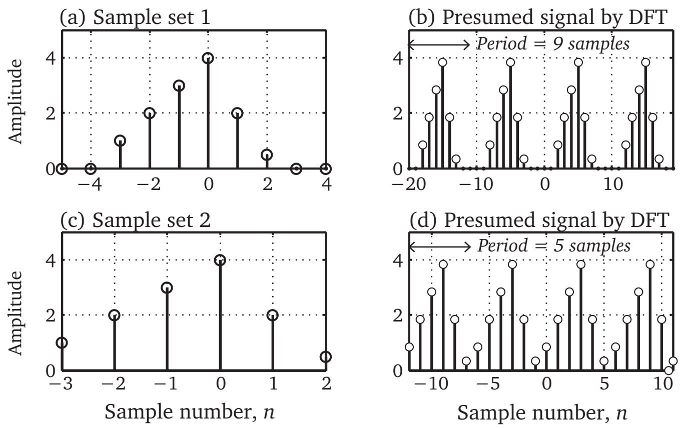

Figure 6.6: What the x-axis of a DFT means.
The signal of interest is any number of samples collected including the zeros as in (a). The DFT presumes the collected piece is one period of a periodic signal, as shown in (b) and (d).

Computing DFT

Computing DFT, the hard way with integration

Example 6.2. Compute the DFT of this signal given by the following four samples. x[n] = [2 1 -3 4]

We are going to use the DFT Eq. (6.3), although this is not a particularly hard calculation. N, the length of the signal is 4, hence the fundamental frequency is equal to $2\pi/4 = \pi/2$. We write the DFT as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi k}{N}n}$$
$$= 2 + e^{-j\frac{\pi}{2}k} - 3e^{-j2\frac{\pi}{2}k} + 4e^{-j3\frac{\pi}{2}k} labeleq:dftex2$$

From here we can compute the first two terms as

$$\begin{aligned}X[0] &= 2+1-3+4=4\\ X[1] &= 2+e^{-j\frac{\pi}{2}}-3e^{-j2\frac{\pi}{2}}+4e^{-j3\frac{\pi}{2}}\\ &= 2+j+3-j4\\ &= 5-j3\end{aligned}$$

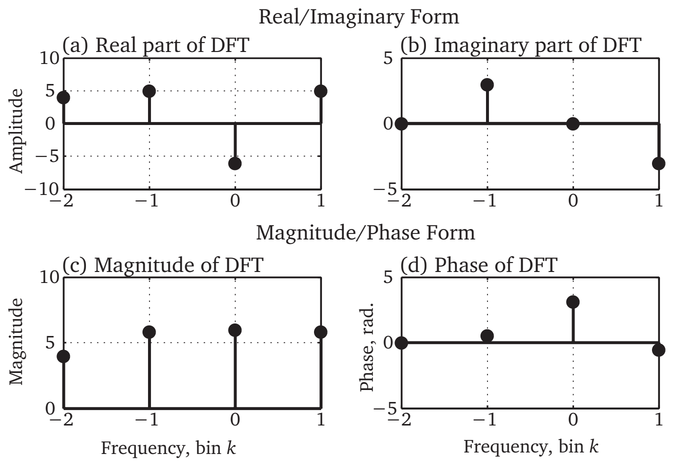

Figure 6.7: The two ways of looking at a DFT
The real and imaginary parts of the DFT vs. bin number in (a) and (b), and in (c) and (d) the magnitude and the phase for same bin numbers.

In Fig. 6.7 we see the computed DFT in both its common forms. In the top row, we see the Real and Imaginary plots and in the second row, we have the Magnitude and Phase plots. They are different but contain the same information. Both forms have their particular use. Since this is a 4 point sequence, we get just four values of the DFT, $X[k]$. It is hard to make sense of what is going on when there are so few values. We should plot them against the DTFT to see if there is a pattern. Where to get the DTFT? In this case it is pretty simple. We use Eq. (6.1). The signal is simple enough that we go ahead and solve the DTFT in closed form.

$$\begin{aligned}X(\Omega) &= \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n} = \sum_{n=0}^{3} x[n]e^{-j\Omega n}\\ &= (2+e^{-j1\Omega}+3e^{-j2\Omega}+4e^{-j3\Omega}) label eq:dtftex1\end{aligned}$$

Note the DTFT equation we used to compute this DTFT. Now we can plot the continuous DTFT from Eq. (**??**) along with the four computed points of the DFT from Eq. (**??**) and we see where these points are coming from. First thing to note, the DFT points are exactly the correct samples of the DTFT.

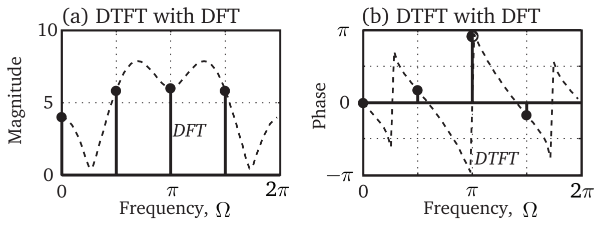

Figure 6.8: The phase and the magnitude of the signal when seen along with the DTFT as its sample points.

We can see from this picture that the four discrete points the DFT although accurate, missed the highs of the magnitude as well as the lows of the phase. Since discrete points is all we get from the DFT, we need more points to see the underlying spectrum better. But there is a problem, if we are given the sequence $x[n]$ which consists of just 4 points and no more data can be had, what can we do?

There *is* something magical we can do at this point and it is called **zero-padding**. This is about the most "fun" you can have in DSP for nothing. It's a very valuable tool. We add a bunch of zeros to the $\underline{\text{end of the signal}}$, $x[n]$ to lengthen it artificially. Now we redo the DFT for this zero-padded and hence lengthened signal (N + added zeros). Remember we said that the DFT counts the zeros at the ends of a signal as real data. But what is truly amazing is what we can extract from these zeros. In Fig. 6.9, we see the result of this zero-padding. The signal $x[n]$ is zero-padded with 12 zeros at the end to make it a 16 point signal. Because we count these new points as part of the period, N is now 16 samples. Its DFT in Fig. 6.9 now has a frequency resolution of $2\pi/16$ instead of $2\pi/4$ it had for the 4-point DFT. And of course we get 16 sampled data points of the DTFT instead of 4. The new plot looks quite good!

Computing DFT the easier way with matrix math

In previous example we calculated the DFT and the DTFT in closed form. The DTFT of arbitrary signals using this method is a difficult task, not one we want to do on paper. The DFT on the other hand, has a reasonably easy to understand architecture amenable to digital calculations. It can be implemented as a linear device as shown in Fig. 6.10 with N inputs

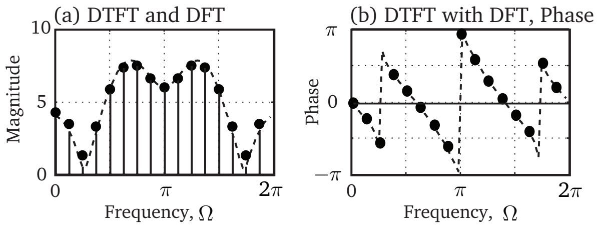

Figure 6.9: Zero-padding increases N and hence provides better sampling of the DTFT.

and N outputs. It is completely lossless and bidirectional. N numbers go in and N numbers come out, using only addition and multiplication.

We can view the DFT as a linear input/output processor consisting of N equations. We have collected L data symbols. If N is less than L, then we select N samples out of the L and perform the DFT calculations on these N points. If N is larger than the L collected samples, then we have to do something to increase the number of samples from L to N. Some algorithms of the DFT will automatically zero-pad a time domain signal to make the length equal to N.

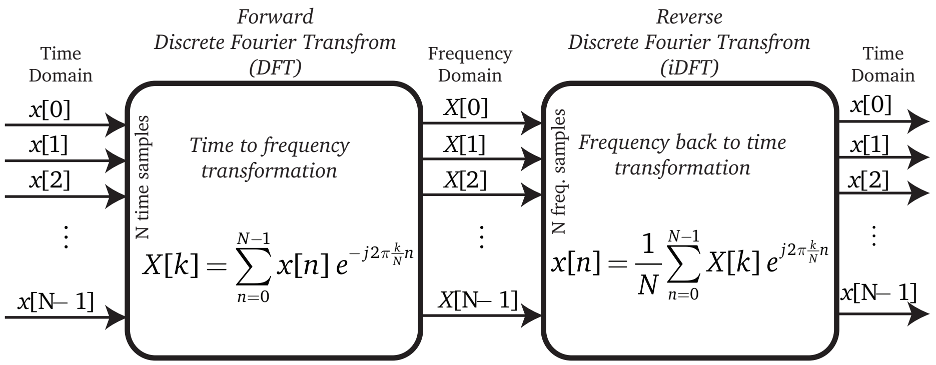
Time Domain
Forward Discrete Fourier Transfrom (DFT)
Frequency Domain
Reverse Discrete Fourier Transfrom (iDFT)
Time Domain
x[0]
x[1]
x[2]
x[N– 1]
N time samples
Time to frequency transformation
$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi \frac{k}{N} n}$
X[0]
X[1]
X[2]
X[N– 1]
N freq. samples
Frequency back to time transformation
$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k]\, e^{j2\pi \frac{k}{N} n}$
x[0]
x[1]
x[2]
x[N– 1]

Figure 6.10: Architecture of a DFT Processor

We can build the DFT calculating machine in two parts, one for the forward or the DFT and the other for the inverse, or the iDFT. We feed N time-domain samples into the DFT machine in Fig. 6.10. The device needs all N points simultaneously before anything can happen. This is because each sample makes a contribution to the each of the outputs, the harmonic components $X[k]$. N samples go in and N samples of frequency response, $X[k]$ come out. Each $X[k]$ gets a little contribution from each $x[n]$.

Here is how each output point will be calculated. There are N of these equations that the machine has to compute. Here we write out a few of these equations, for $k = 0$, 1 and

N.

$$\begin{aligned}
X[0] &= x[0]+x[1]+x[2]+\ldots x[N-1] \\
X[1] &= x[0]+x[1]e^{-j2\pi/N}+x[2]e^{-j4\pi/N}+\ldots+x[N-1]e^{-j2(N-1)\pi/N} \\
&\vdots \\
X[N-1] &= x[0]+x[1]e^{-j2(N-1)\pi/N}+x[2]e^{-j4(N-1)\pi/N}+\ldots+x[N-1]e^{-j2(N-1)^2\pi/N}
\end{aligned} \tag{6.8}$$

Computers were designed to do matrix math, so these equations can be easily solved. The only problem comes as N increases. The number of calculations gets very large. The matrix computation done this way requires a number of calculations that are proportional to N^2. If we take advantage of the symmetry as well the phase relationship of the sine and cosine, the number of computations can come down to a number more like $N\log_2 N$. This and other algorithmic innovations were first proposed by J.W. Cooley and John Tukey in 1965. There are now many variations on their original invention and they are all referred to as the **Fast Fourier transform** or FFT. Their innovation was extremely important particularly for hardware implementation. We use this algorithm today for calculating the DFT, so much so that we often find ourselves saying “take a FFT” when we really mean the DFT.

We notice that all calculations require the manipulation of the complex exponential. The complex exponential is hard to type, so we simplify it by writing it like this.

$$e^{j2\pi/N} = W_N$$

We introduced this idea in Chapter 3, when discussing the discrete Fourier series calculations. It is easy to see that W_N is a function of N only, the rest of the terms are all constants. Now we also define the same constant in terms of other two variables, n and k. These are the time and harmonic indexes.

$$W_N^{nk} = \left(e^{j2\pi/N}\right)^{nk} = e^{j2n\pi k/N}$$

This form has the variables n and k in the exponent and so the constant W_N is written with all three indexes as W_N^{nk}. Here we are merely raising W_N to power nk, or $\left(e^{j2\pi/N}\right)^{nk}$. Hence W_N^{nk} is a matrix of size $n \times k$ with each index ranging from 0 to $N-1$. Now rewrite the DFT using the term, W_N^{nk}. This is somewhat easier to write than the exponential, as it has less moving parts.

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk} \tag{6.9}$$

Now we write Eq. (6.9) in matrix form as

$$\mathbf{X} = W_N^{nk}\mathbf{x} \tag{6.10}$$

X is the output Fourier transform vector of size N and **x** is the input vector of the same size.

$$\mathbf{X} = \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

The W_N^{nk} term in Eq. (6.15) is called the DFT matrix and is written for all n and k indices as

$$W_N^{nk} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & \dots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \dots & W_N^{N(N-1)} \end{bmatrix}_{n\times k} \tag{6.11}$$

Note that the matrix size is $n \times k$, but since both n and k are equal to N, the DFT matrix size is equal to $N \times N$, hence this matrix is always square. The columns represent the index k of the harmonics and rows the index n, time. The first row has index $n = 0$, so all the first row terms are equal to 1, because $W_N^{(n=0)k} = e^{j\Omega(n=0)} = 1$.

The first column has index $k = 0$, hence all W_N^{nk} terms in the first column have a zero exponent and so all are also equal to 1. The third row, second term is equal to W_N^2, for $k = 1$ and $n = 2$. The rest can be understood the same way. The matrix is symmetrical about the right diagonal such that

$$W_N^{nk} = W_N^{kn}$$

All of the terms in Eq. (6.11) are constant for given n and k integers. If you stare at the equation long enough, I am sure just as Cooley and Tukey, you will be able to see the advantage of matrix calculations. This matrix can be pre-computed and stored for various value of N. That saves computation time. We don't need to compute the whole matrix as well, because it is symmetrical. There are actually a lot of tricks that can be used to make computations efficient. We can also show that the inverse of this matrix is easy to compute because it is just the inverse of the individual terms. This also makes reverse computation easier and quicker.

$$\left[W_N^{nk}\right]^{-1} = \frac{1}{N}\left[W_N^{nk}\right]^* \tag{6.12}$$

Let's write out the **DFT matrix** for $N = 4$. The fundamental frequency for $N = 4$ sample signal is equal to $\Omega_0 = \dfrac{2\pi}{4} = \dfrac{\pi}{2}$. Hence we write $W_4^{nk} = e^{-j\frac{2\pi k}{4}n}$.

This is equal in Euler form to

$$\begin{aligned} e^{-j\frac{\pi}{2}kn} &= \cos\left(\frac{\pi}{2}nk\right) - j\sin\left(\frac{\pi}{2}nk\right) \\ &= -j^{nk} \end{aligned} \tag{6.13}$$

Product nk is always an integer, hence the cosine term for this case is always zero. (Except when $n = 0$, $k = 1$, $\cos(0) = 1$. Same goes for the sine). The sine however is non-zero and depends on the specific value of product, nk. So depending on the values of the product, we get these values for the matrix entries.

$$\begin{aligned} n = 0, k = 1,\ W_4^0 &= +1 \\ n = 1, k = 1,\ W_4^1 &= -j \\ n = 2, k = 1,\ W_4^2 &= -1 \\ n = 3, k = 1,\ W_4^3 &= +j \end{aligned}$$

From this we can develop the W_4^{nk} matrix as

$$W_4 = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & j \end{bmatrix} \tag{6.14}$$

The inverse comes easily from Eq. (6.12)

$$W_4^{-nk} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \tag{6.15}$$

Notice from Eq. (6.14) and Eq. (6.15) that only the imaginary components changed sign. All the real terms kept their sign. Some values of the matrix are complex conjugate pairs so we don't even need to calculate them saving time. This is of course related to that fact that for the forward transform we need $e^{-jn\Omega_0 k}$ and for the reverse we need $e^{jn\Omega_0 k}$. The difference between these is that the imaginary part changes sign. $(\cos\theta + j\sin\theta)$ vs. the

$(\cos\theta - j\sin\theta)$. Hence in (6.12) for the inverse you are seeing the imaginary part changing sign but not the real part.

We can now use this *DFT matrix* to calculate the DFT of any arbitrary signal of 4 input points using the equation (6.9). Larger DFT matrices would be needed to compute larger size DFT. What is nice is that these matrices (of any given size) are always the same and can be pre-computed and kept in memory to speedup calculation.

Example 6.3. Now we will compute the DFT of the signal from Example 6.2 using the matrix method.

$$x[n] = [2 \quad 1 \quad -3 \quad 4]$$

We already computed the 4 point DFT of this signal in Example 6.2. The values computed were

$$|X[k]| = [4 \quad 5.83 \quad 6 \quad 5.83]$$

Now we will use the matrix method to verify the closed form solution. We write the matrix equation using the DFT matrix for $N = 4$.

$$\begin{aligned}
\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ -3 \\ 4 \end{bmatrix} \\
&= \begin{bmatrix} 4 \\ 5 - j3 \\ -6 \\ 5 + j3 \end{bmatrix}
\end{aligned}$$

Okay, we got the same thing!

Now let's compute the inverse DFT.

$$\mathbf{x} = \left[W_N\right]^{-1}\mathbf{X}$$

$$\begin{aligned}
\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}^{-1} \begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} \\
&= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} 4 \\ 5-j3 \\ -6 \\ 5+j3 \end{bmatrix} \\
&= \begin{bmatrix} 2 \\ 1 \\ -3 \\ 4 \end{bmatrix}
\end{aligned}$$

We get the original values back. The DFT matrix of size (4x4) can be used to determine the DFT and the iDFT of any discrete signal of length 4 samples. The generalization of this idea can be used to compute the DFT and the iDFT of any length signal using the matrix method. This is exactly what happens inside Matlab when asked to execute the FFT/iFFT function.

Computing DFT the easy way with Matlab

Now that we understand the matrix formulation of the DFT equation, we hand it over to Matlab to do the actual calculations, which it does with aplomb. Given any N samples, we can easily compute the DFT in Matlab with just a few lines of code like this.

```
x1 = [ 2 1 -3 4];
N = length(x1);
nf = -N/2: N/2-1
mag = fftshift(abs(fft(x1, N)));
phase = atan(imag(fft(x1, N))/real(fft(x1, N)))
Ichan = fftshift(real(fft(x1, N)));
Qchan = fftshift(imag(fft(x1, N)));
```

For the computation of the DFTs we use the algorithm built-in to Matlab. It allows us to compute an N-point DFT for any N, as long as it is an integer, although this function is called FFT in Matlab. The true FFT algorithms use signal lengths that must be powers of 2, but not in Matlab form of the FFT. You can use any integer, 3, 5, 13 etc.. Note that we can plot the DFT as pair of plots of either, Magnitude and Phase or as Real and Imaginary parts. In Matlab, the DFT command is implemented by a command called “fft”. The term “abs” in front of the fft command gives us the magnitude of the DFT, not its power. The *fftshift* command centers the DFT on zero, a preferred form of showing the DFT. The DFT is usually drawn as a stem plot. However, if N is large, a stem plot becomes a solid blob. The DFT in such cases is drawn with the “plot” function showing the envelope instead. But that is just convention of drawing and does not change the underlying nature of the DFT. It is defined only at N *discrete* points.

Example 6.4. Find the DFT of $x[n] = 2\delta[n-1] - 3\delta[n-2] + 5\delta[n-4]$.

This signal can be converted to 5 samples as follows: $x = [0 \quad 2 \quad -3 \quad 0 \quad 5]$. For $N = 5$, the fundamental frequency is equal to $\Omega_0 = 2\pi/5$. From Eq. (6.11), we compute the DFT by

$$\begin{aligned} X[k] &= \sum_{n=0}^{4} x[n] e^{-j\frac{2\pi k}{5}n} \\ &= 2e^{-j\frac{2\pi k}{5}} - 3e^{-j\frac{6\pi k}{5}} + 5e^{-j\frac{10\pi k}{5}} \end{aligned}$$

The DTFT similarly can be written as

$$\begin{aligned} X(\Omega) &= \sum_{n=0}^{4} x[n] e^{-j\Omega n} \\ &= 2e^{-j2\Omega} - 3e^{-j3\Omega} + 5e^{-j5\Omega} \end{aligned}$$

We can calculate the values of $X[k]$ easily for each k, by summing for n from 0 to 4. In Fig. 6.11(a) we plot the signal and in (b) its five DFT values calculated using Matlab. Since we have such few data points, the plot in (b) does not make sense, until we plot the data along with the DTFT. Plotted against the DTFT, we see that the DFT values are indeed the true values of the DTFT at the selected points. The sampling is not that great, it misses some of the high points. This is the same problem we saw in Example 6.2.

Zero-padding

To improve this DFT plot, we utilize zero-padding by adding zeros to the end of the signal. When we do the DFT (using Matlab) of this longer signal, we see in Fig 6.12(b), a denser sampling of the DTFT. In Matlab, zero-padding can be done by simply changing the size of the DFT. Matlab automatically adds zeros to the signal to lengthen it.

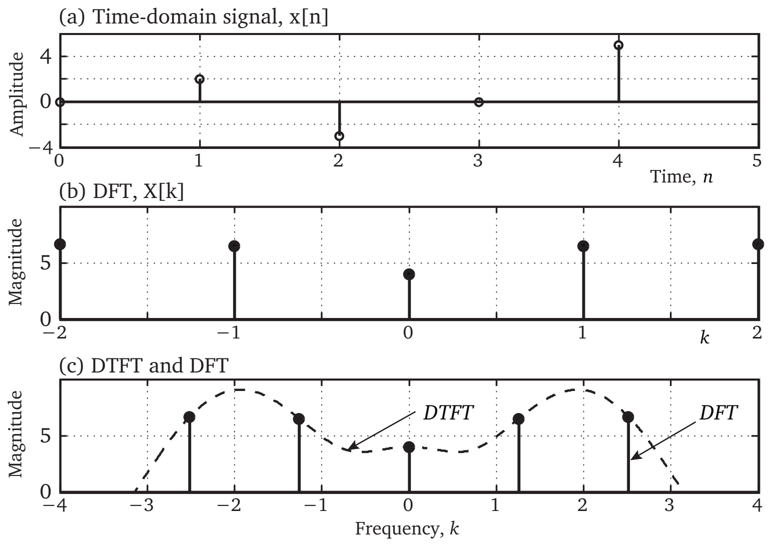

Figure 6.11: The DTFT and the DFT of signal $x = [0\ 2\ -3\ 0\ 5]$.

```
x = [ 1 2 -3 -4 2]; % A signal of length 5 samples
DFT7 = fft(x, 5)  : DFT of length 5
DFT17 = fft(x, 17)  : DFT of zero-padded signal of length 17
```

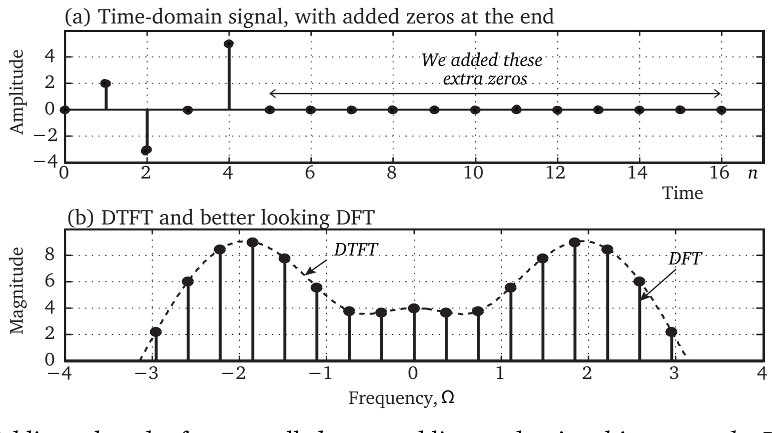

Figure 6.12: Adding a bunch of zeros, called zero-padding to the signal improves the DFT resolution.

We note that the DFT in the first case had only 5 samples, whereas the second zero-padded case has 17 points. In both cases, the number of samples in the frequency-domain are equal to the length of the signal. In the first case, the frequency resolution is $2\pi/5$ and in the second case it is $2\pi/17$, a much smaller number. (To convert these numbers to actual frequency, multiply them by F_s/N where sampling frequency F_s is 5, and N is 5 and 17 respectively for the two cases.)

The zero-padded result in Fig. 6.12 is truly amazing and you should pause to appreciate this moment. We can say that the effect of N, the number of samples is like looking at a DTFT through a fence, with each fence board of a width inversely proportional to N. A small N is like a wide board and hides more of the signal. A large N, results in a narrow fence board and we can see more of the hidden signal. When N is infinite such as in DTFT, there is no fence at all, in which case we see the whole signal un-obsecured. Price we pay for this is increased complexity. Note that changing N by zero-padding cannot change the signal behind the fence, it only allows you to see more of it. The only way to change the DTFT behind the fence is to collect more sample points and do a better DTFT, or find ways improve the signal behind the fence itself. The DFT has no effect on the DTFT, it is a snapshot of the real thing through an observation "window". You might ask why bother with the DFT? The DFT gives us discrete values from a simple numerical calculation while the DTFT requires heavy duty integral math. So DFT wins, always, unless this is a homework problem.

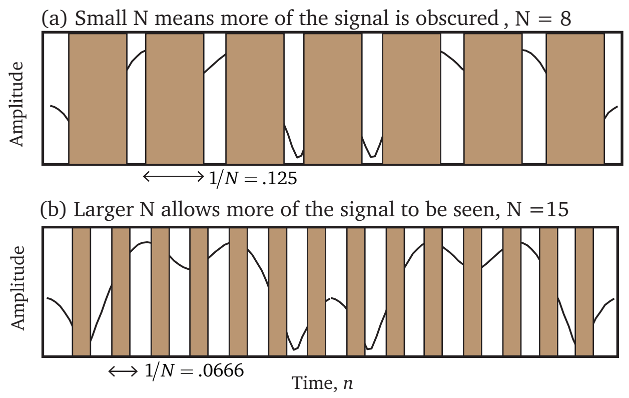
(a) Small N means more of the signal is obscured , N = 8
Amplitude
1/N = .125
(b) Larger N allows more of the signal to be seen, N =15
Amplitude
1/N = .0666
Time, n

Figure 6.13: DFT is a fence in front of the DTFT. Narrower boards, i.e. larger N allow us to see more of the signal.

Zero padding gives more points of interpolation into the DTFT. However, caution is warranted if the signal contains a higher frequency signal components beyond twice the sampling frequency, zero-padding would do nothing to help discover them. So although we say that zero-padding improves resolution, the strict meaning of that word implies the ability to resolve two closely spaced components in the spectrum. That we do not get that with zero-padding, and the only way to get better *resolvability* is to use more data. The correct interpretation for what zero-padding does is: <u>improved interpolation density</u>, just like using more points to plot a curve given its equation. It does not change the equation, only the number of points plotted.

Better interpolation density vs. better resolution

Zero-padding results in a better looking plot with a higher density spectrum but it does not effect or change the resolution of what we are seeing. To see this effect, lets take the following signal with two closely spaced frequency components of frequency .02 and .028 Hz.

Example 6.5.

$$x[n] = \sin(2\pi \cdot 0.02n) + \cos(2\pi \cdot 0.028n)$$

In Fig. 6.14 we see a 13 point DFT. We don't see the two sinusoids even though the signal has been properly sampled at $F_s = .1$ Hz. We zero-pad the 13 samples by 100 zeros. The DFT of the zero-padded signal although has better density of points, also does not show the two separate frequencies. Now instead of 13 samples and 100 zeros, we collect 130 actual data samples of the signal. The 130 point DFT on this signal indeed highlights the two components at $f_1 = .02$ and $f_2 = .028$ Hz. The same 130 point zero-padded to 1024 points does even better. The amplitudes for the zero-padded DFT are more accurate than the DFT on the true number of samples. Hence we see how better interpolation density is not the same thing as better resolution. Better resolution can only come from using more data points in the DFT. But once we have a *proper resolution,* increasing the DFT size generally makes things better.

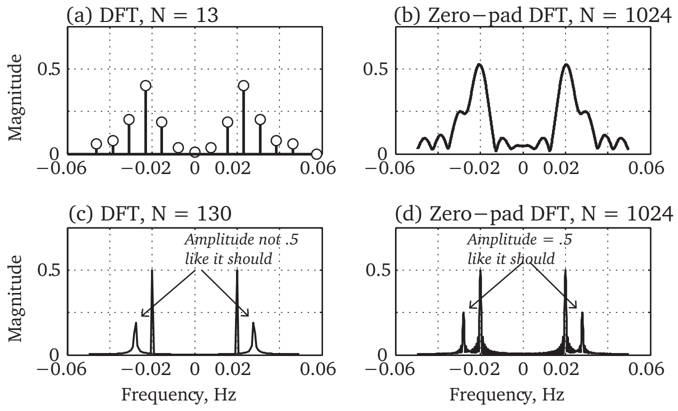

Figure 6.14: Zero-padding increases point density as we see in (b) but does not help with resolution. To increase resolution we need to use more samples in the DFT which improves resolution, as in (d).

Fundamental functions and their DFT

Most discrete signals can be represented in time domain by linear summation of these fundamental signals: the delayed impulse function, the CE, the sinusoid and the sinc/square pulse function. Hence let's examine their DFT.

DFT of an impulse function

The discrete signals are all about impulses. Here we have a sequence of just one impulse at the origin, with 11 zeros following it. We already know that the DTFT of an impulse is a constant 1 in the continuous frequency domain. The DFT however is a train of impulses of amplitude 1 because it is a sampled version of the DTFT, hence we see the impulse train.

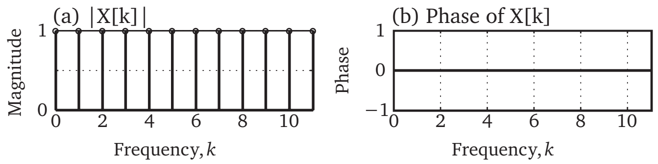

Figure 6.15: (a) The magnitude response of a single impulse function at origin, its Discrete Fourier Transform is an impulse train with frequency separation equal to $2\pi/N$, (b) the phase is zero.

The fundamental frequency for this 12 sample case, is equal to $2\pi/N$, with N = 12. Each impulse is spaced $\pi/6$ radians apart, reaching 2π after 12 samples. Then the whole thing starts over again. The pulses repeat endlessly in frequency domain. But of course, we only calculate one main set called the principal alias, as shown here. This idea that the spectrum repeats every 2π is a key point which you will see repeated often in this chapter. The iDFT of an impulse train will of course then be a single impulse. Why is the phase zero? Phase is zero because the imaginary part of the DFT is zero.

$$\begin{aligned} x[n] &= [100000000000] \\ X[k] &= \sum_{n=0}^{11} x[n]e^{-j(\pi/6)nk} \end{aligned} \tag{6.16}$$

The result of this summation is all 1's in the real domain from Eq. (6.16) and all zero's in the imaginary domain. We have plotted only the magnitude and the phase of this signal in Fig. 6.15

DFT of a delayed impulse

What if the impulse was delayed by n_0 samples? The time domain signal $x[n] = [100000]$ becomes: $x[010000]$ with a 1 sample delay (also called a shift). The shift represents the frequency of the CE. If we have six samples in the signal, then each shift is a 1 Hz change in frequency. In Fig. 6.17, the signal has been shifted by two samples and we see that this is a signal of 2 Hz as compared to Fig. 6.16 which is a signal of 1 Hz. The time-shift property says that the DFT of a delayed function is equal to the DFT of an un-delayed function multiplied by the CE of frequency n_0, or $e^{-j(2\pi/N)n_0k}$. (See Table of Properties, 4.1).

$$\begin{aligned} x[n-n_0] &\iff e^{-j(2\pi/N)n_0k}X[k] \\ X(k)e^{-j(2\pi/N)k_0n} &\iff X[k-k_0] \end{aligned} \tag{6.17}$$

The result of this multiplication is such that the magnitude stays the same while the phase changes. This is exactly what we see in Fig. 6.17, where the phase has changed.

Conversely if we take a time domain signal and multiply it by a CE of frequency $(2\pi k_0)/N$, then the spectrum just shifts to this frequency. This is the principle behind modulation. We take a baseband signal and multiply it by a carrier frequency and lo and behold, we have a modulated signal ready to be transmitted over the chosen medium.

DFT of a sinusoid

In Fig. 6.18 we plot the DFT of a sine and a cosine.The cosine is said to exist in the real domain and hence its real domain DFT is same as its magnitude. The sine is said to exist only in the imaginary plane. Its DFT results in two frequency components of opposite signs. The absolute value or the magnitude however is exactly the same as the cosine.

DFT of a complex exponential

Now we look at the DFT of a complex exponential, shown as a complex signal of a sine and a cosine of a particular frequency. The DFT of this elementary signal is a single impulse located at the frequency of the CE. It is actually the complex summation of a sine and cosine. Note that the real part of this signal is a cosine and hence we get two half-amplitude impulses from that, and the imaginary part is from the sine as shown. The CE is given by $\cos(\omega_0 t) + j\sin(\omega_0 t)$. The rotation of the sine DFT by $\pi/2$ causes one set of the components to cancel and the other to coincide and add. The net effect is a DFT consisting of a single impulse of amplitude of the CE. We discussed this in Chapter 2 for continuous signals.

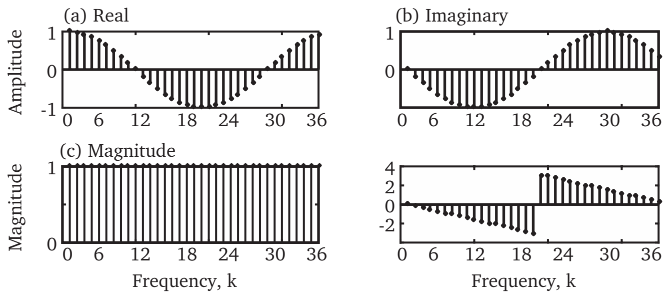

Figure 6.16: (a) The signal [1 0 0 0 0 0] *is delayed by one sample to become* [0 1 0 0 0 0]. *(b) Note that the frequency of this signal is* $2\pi/6$. *If N is 6 and* F_s *is also six, then the frequency represented by this shift is 1 Hz.*

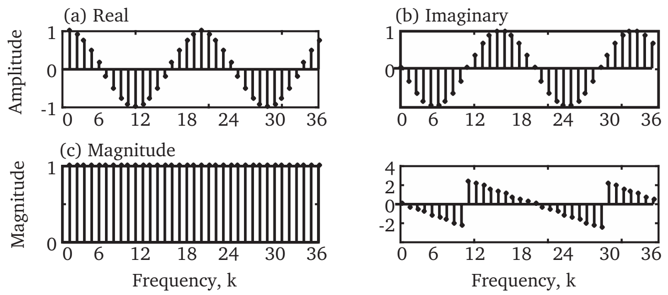

Figure 6.17: (a) The signal [1 0 0 0 0 0] is signal delayed by two samples to become [0 0 1 0 0 0]. (b) Note that the frequency of this signal is $2\pi/6$. *If N is 6 and* F_s *is also six, then the frequency represented by this shift of 2 samples is 2 Hz.*

We use this property of the CE DFT to up-convert signals. Any signal when multiplied by a CE will shift in spectrum to the frequency of the CE.

Example 6.6. Compute the DFT of this signal: $x[n] = \delta[n] - 0.5\delta[n-6]$, for $N = 7$.

Here we have a signal that is composed of an un-delayed and a delayed impulse. We can in principal compute its DFT by summing up the individual DFTs. But we choose to do it the hard way here.

Let's compute the DFT of the signal as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk\frac{2\pi}{N}n} = \sum_{n=0}^{7} x[n] e^{-jk\frac{2\pi}{7}n}$$
$$= \sum_{n=0}^{N-1} \big(\delta[n] - 0.5\delta[n-6]\big) e^{-jk\frac{2\pi}{7}n}$$

In the last step, the multiplication of the complex exponential by a delta function just means that we grab the value of the exponential at the location of the delta function which we do

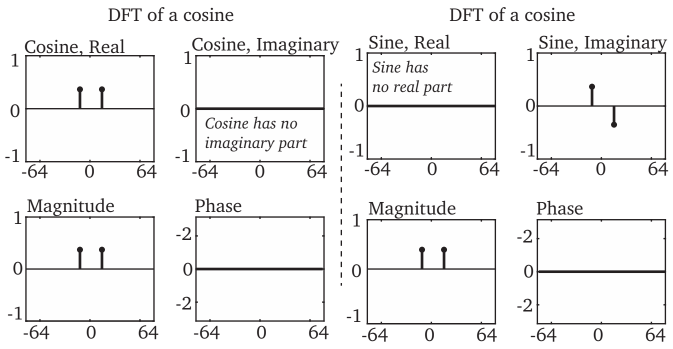

Figure 6.18: We show the DFT of a cosine and a sine. The cosine exists only in the real part. Its magnitude is exactly the same as it real part. The sine however has an odd imaginary part but has the same magnitude as the cosine. The phase for both is zero, or implies no phase change over the sampling frequency.

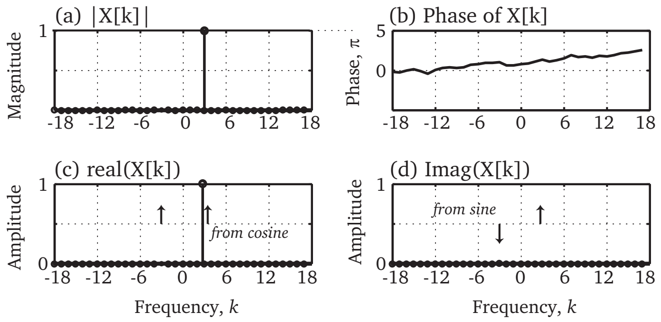

Figure 6.19: (a) Magnitude of a complex exponential consists of one impulse of magnitude 1.0, (b) Its phase goes from 0 to 6π over the three periods covered by 36 samples, (c) The real part of the DFT is zero since this is a sine wave and has in (d) only the imaginary components.

here for $n = 0$ and $n = 6$.

$$\sum_{n=0}^{N-1} \big(\delta[n] + 0.5\delta[n-6]\big) e^{-jk\frac{2\pi}{7}n} = e^{-jk\frac{2\pi}{7}n}\Big|_{n=0} + 0.5\, e^{-jk\frac{2\pi}{7}n}\Big|_{n=6} = 1 + 0.5e^{-jk\frac{3\pi}{2}}$$

$$X[k] = 1 + 0.5e^{-jk\frac{7\pi}{4}}$$

The first impulse at time = 0, gives us a constant value for the magnitude. The second impulse transforms to a complex exponential in the frequency domain. The result is a sinusoid of frequency 6 Hz as we see in Fig. 6.19 that has a DC value. We did the analysis for 7 points in (a) which sadly misses the essential features of the response. Now we do exactly the same thing for 21 and then for 71 points by padding the end of the signal with a lot of zeros, getting a far better signal spectrum.

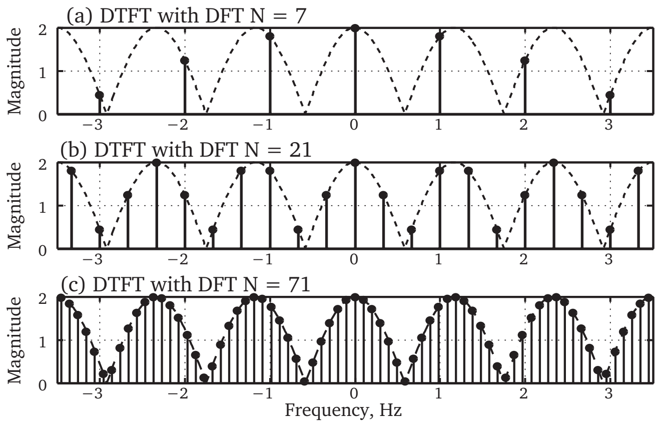

Figure 6.20: DFT of a signal with different amounts of zero padding, length $= 7$*, length* $= 21$*, length* $= 71$.

DFT of a square pulse

In Fig. 6.21, we calculate the DFT of three square pulse signals, all are of length 12 samples with pulse width equal to 3, 5 and 7 samples. Note that the DFT magnitude follows the DTFT equation, Eq. 6.23 of a square pulse. As we increase the width of the square pulse, we see more lobes. The number of lobes are a function of the width of the square pulse. Note that had the pulse size been one single sample, then we devolve to the case of the DFT of a delta function at the origin. And if the pulse width were exactly equal to the length of the signal, then we would get a single impulse. The DTFT would be sampled only at the origin.The DTFT as calculated in Chapter 4 is given as follows, with pulse $N = 2M + 1$ samples wide.

$$X(\Omega) = \frac{\sin\left(\frac{2M+1}{2}\Omega\right)}{\sin\left(\frac{1}{2}\Omega\right)} \tag{6.18}$$

DFT of a finite sinc function

Alternately when the input to the DFT in time domain is a sinc function as shown in Fig. 6.22, we get (hope to get) a square pulse. In theory, the sinc signal would give a perfectly flat-top like spectrum. But because we are unable to use an infinite length sinc pulse, we don't get a perfect flat-top response as we see in Fig. 6.15(b) and (d). The shape of the response is a function of the length of sinc pulse, or the number of data points used for the DFT.

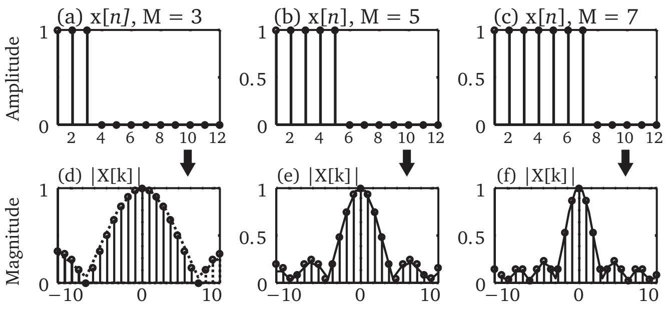

Figure 6.21: (a) We see three pulses of different widths, M = 3, 5, and 7 samples. The DFT magnitude of each is a sinc function with number of lobes equal to the width of the pulse in samples.

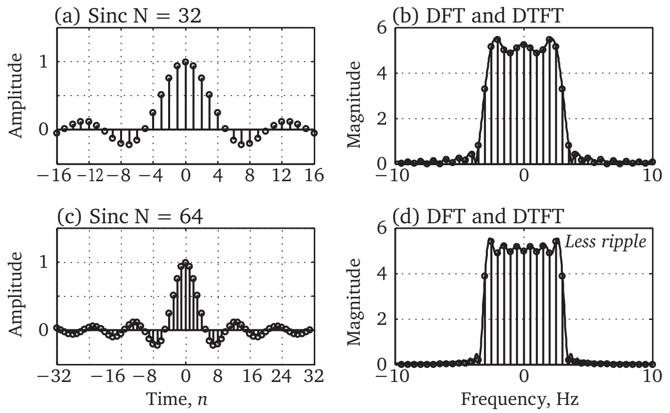

Figure 6.22: A sinc pulse of finite length has a less than perfect DFT. We were expecting a flat top square pulse but we don't get that due to the finite length of the sinc samples.

In case 1 in Fig. 6.22, we use 32 samples of the sinc. Notice that when this piece is periodically extended we would not get a true sinc. Hence the DTFT and DFT on the right show a ripple in passband. When the number of samples is increased to 64 in (c), the DTFT and the DFT both improve with a smaller ripple. An infinitely long sinc would and should result in a perfect flat-top frequency response but most sinc pulses used are finite length and hence will have a ripple as shown here.

DFT repeats with sampling frequency

Let's not forget the important fact that the DFT repeats with the sampling frequency, just as does the DTFT. If we were to sample a sinc function with $F_s = 1.5$, 2 and 3 Hz, take a DFT of each, then each "square" pulse DFT (shown in a continuous line) as we see in Fig. 6.23, is centered at integer multiple of the sampling frequency. This is what is meant by repeating spectrum or the DFTs.

DFT is often plotted by connecting the values instead of with stem function. Stem function is more instructive only when N is small. Hence all practical spectrum are plotted as an envelope despite being discrete.

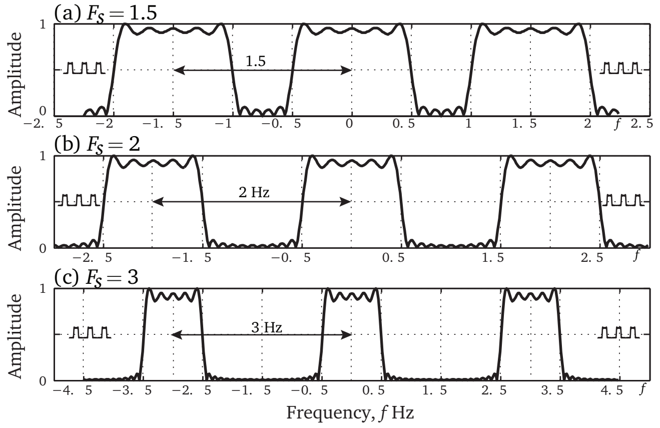

Figure 6.23: When sampled with F_s = 1.5, 2, and 3 Hz, the DFT repeats with same frequency.

DFT Periodicity and Leakage

Let's examine the DFT of an elementary signal, a discrete periodic sinusoid of frequency $2\pi/N$, amplitude of 1 and a fundamental period of N_0 samples. We know the DTFT of the sinusoid. It consists of two impulses of half-amplitude located at $+2\pi/N$ and $-2\pi/N$. Because the DFT is a sampled version of the DTFT, we note that this should make the DFT computation trivial. The DFT should also have two impulses and nothing more. In Fig. 6.24, we take a sinusoid of period $N_0 = 7$. We will do five different DFT's of this signal for N = 7, 9, and 11, 12 and 14 samples.

However, the DFT is ideal (i.e. only two impulses) only for the two cases where the DFT length N is equal to an integer times the period of this sinusoid, $N_0 = 7$ samples. The three others DFTs, for N = 9, 11 and 12, show more than just two impulses, and can be considered "inaccurate" or noisy. This "noisiness" is a function of the selection of the number of samples for the DFT and is called leakage. Sort as if the ideal impulse values were melting and leaking into adjacent bins.

These DFT results, supposedly done on the same sinusoid, vary because we really aren't doing the DFT of the same sinusoid, per the thinking of the DFT. We are forgetting that DFT is picturing the signal differently than what we have in mind. These are in fact five different signals as far as the DFT is concerned and hence we see five different frequency representations. So what appears inaccurate to us is actually the correct answer from the

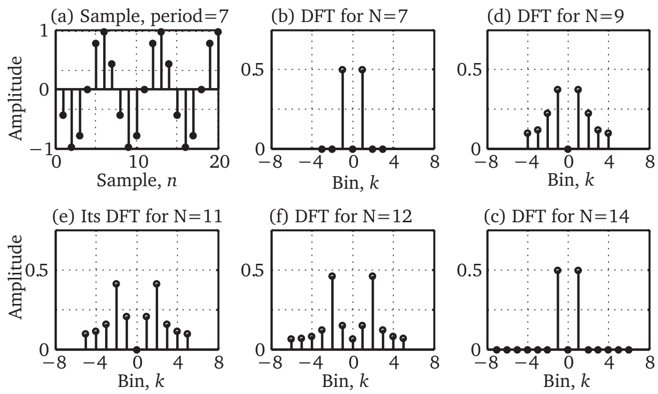

Figure 6.24: The DFT of samples from a sinusoid of frequency 1 Hz and $N_0 = 7$ samples. The DFT of the signal for $N = 7$ results in an ideal response. When N is 9, 11 and 12, the result is noisy and not ideal. The DFT is again ideal when N is equal to 14.

point of view of the DFT. But this is not what we had hope to see. In each case, the input signal is the exactly the same. All that differs is the number of samples used to do the DFT. How did the DFT come up with these different results and why do they not match the case of 7 and 14 samples?

This behavior is caused by what is formally called the truncation effect. When we select a certain number of samples on which to do the DFT, we have in effect truncated the signal. For DFT length of $N = 7$, or 14, the number of points used for the DFT are equal to an integer multiple of the period $N_0 = 7$. Hence the DFT looks exactly as we would expect, just two impulses. However when we selected 9, 11 and 12 samples, we violated the periodicity requirements. In Fig. 6.20 we see what the DFT presumes it has. It is different from what you and I are thinking. We see the signal representation by the DFT and in case (b), (c) and (d), the periodicity falls apart after the given number of samples. In each case, DFT thinks that that the number of samples given to it are all part of one period, hence its representation of the signal and what we want (as shown by the dashed line) differ greatly depending on how far we are from the true period. The leakage is worst when the DFT size is $N_0/2$ samples off the main period.

Truncation effect

When we collect a signal, any signal, we are “truncating” it. No matter how many samples we choose, by choosing a certain number we are in essence multiplying the signal with a truncation function. In mathematical sense, we were multiplying the signal we were measuring with a truncation function consisting of a set of all 1’s, sort of like an on and off

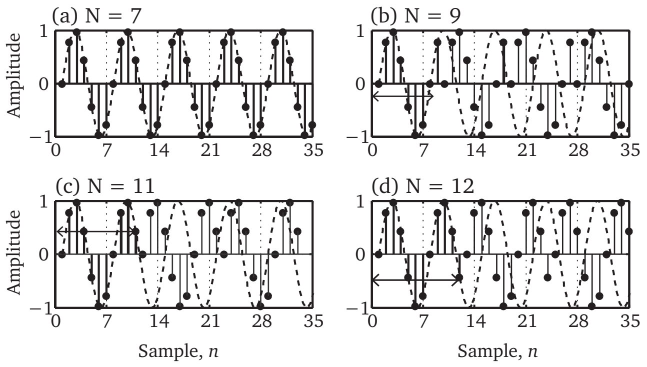

Figure 6.25: Seven samples of the signal when concatenated in (a), result in the desired signal. When 9, 11 or 12 samples points are used, the resulting signal per the DFT, is not the one we want.

switch. This seems like a benign and even ridiculous thing to talk about. Of course, we have to stop collecting the data somewhere! And yet, any time we stop, we truncate the signal, we have multiplied it with an N point all 1's rectangular function.

From the convolution property of the Fourier transform, we know that the DFT of the truncated signal can be written as the convolution of the DFTs of the original signal x[n] and the truncation signal w[n].

Our signal is a sinusoid and we know that its transform looks like two impulses, centered at the sinusoid frequency around zero.

$$X(\omega) = \pi\delta(\omega - \omega_0) + \pi\delta(\omega + \omega_0)$$

The DTFT of the window function, which is a square pulse function, is a Diric function. Its DTFT is given by

$$W(\omega) = e^{-j\omega\frac{N-1}{2}} \frac{\sin\left(N\frac{\omega}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}$$

The multiplication of the two sequences and their DFT can be plotted in Matlab using

```
N = 16;
om = 0: .01: 2*pi;
x = exp(-1i*om*(N-1)/2).*diric(om, N)
plot(om-pi, fftshift(20*log10(abs(x))))
axis([-pi pi -40 0])
grid on
```

When plotted, we get a N lobed sinc function over a 2π range. In Fig. 6.26 we see these lobes over the sampling frequency range of -$F_s/2$ to $F_s/2$. The sampling frequency is 7 samples per period. Hence for each case, we get more lobes as N increases. In the second column we see the DTFT of the sinusoid. No matter, how many samples, the DTFT is always the same for a sinusoid, always just two impulses. In the third column, we see the result of the convolution. In effect, the DTFT of the truncation function, a diric function, has been lifted and moved to the location of the impulses, one to the right, the other to the left. Since DFT is the sampled version of the DTFT, we see that it is indeed picking up the true values of the DTFT. The sampled values are the same DFT values shown in Fig. 6.26

When a signal frequency does not fall on a DFT bin, its peak amplitude is not measured. What is measured are the intermediate values of the Diric function and these values trail off as the Diric function decays. The lesson to be learned is that one needs to always do the DFT on the number of samples that are an integer multiple of the fundamental period of the discrete signal. This is easier said than done and is a major challenge.

Truncation causes leakage

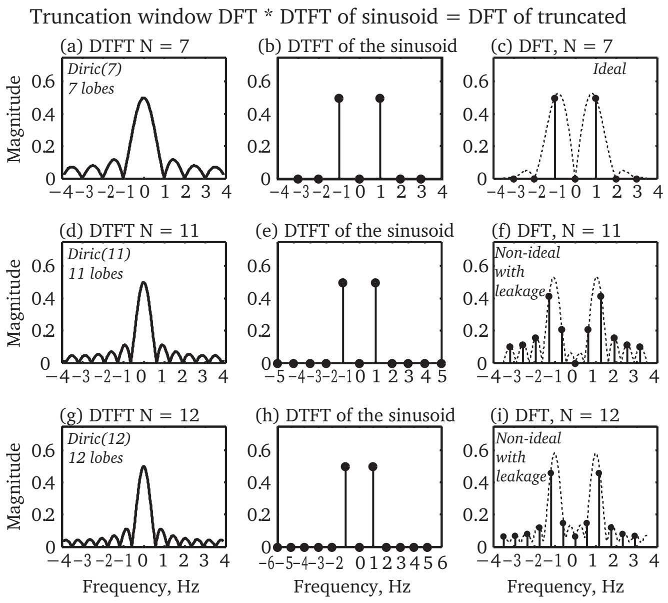

Figure 6.26: When the frequency of the signal does not coincide with a bin, the energy is spread across several bins. (a) the DTFT of N = 7. (b) DFT for N = 7, we get an ideal response. (c) DFT falls at peaks of the DTFT.

Of course in real life, given a bunch of samples, we never know if we have an integer number of periods of all the frequency components in the signal. From the given samples, we do not know if every single frequency component will align perfectly with our chosen sample size. This effect can be alternately visualized by looking at the edge discontinuities of the samples collected. If there is a discontinuity between the right and the left side of the samples, i.e, the right side is at -1 and left at + 1, then we will get a truncation effect. This effect is also called **Leakage**, and is nearly always present in the DFT/FFT of real signals. We are always going to get this sort of distortion particularly when the number of samples used for the DFT/FFT must be a power of 2. We can either tolerate this error (acceptable if small) or do something to eliminate or reduce it.

We mentioned a type of distortion called, aliasing in Chapter 3. How is aliasing related to leakage? Aliasing and Leakage are not related at all. These are two different effects. Aliasing results when sampling frequency is less than two times the highest frequency in the signal. It occurs at the point when a decision is being made to sample the signal. Leakage occurs, when we pick the length of the DFT. More specifically, leakage (its yet another name is *smearing*) results when the number of samples selected for doing a DFT are not equal to an integer multiple of the period of *each and every frequency* in the signal. We can easily avoid aliasing but leakage is very difficult to eliminate.

We notice that the deleterious effect of leakage is coming from the frequency response of the truncation function, which is most commonly the rectangular function or what we call the all 1's signal. We notice in Fig. 6.27, that as N is increased, the main lobe of the Diric gets narrower and hence approaches an impulse. This tells us that a larger N, even when it is not an integer multiple of the period will tend to reduce leakage.

Reducing leakage by doing a longer DFT

In Fig. 6.28(a) we look at a signal which is a sum of two sinusoids. In first case, we set the frequencies of the two sinusoids to 2 and 4 Hz. The sampling frequency is 16 samples per second. Note that if we do a 16 point DFT, then the frequency resolution is 16/16 or 1 Hz. Hence the signal frequencies, both 2 and 4 Hz fall on a bin. Hence the DFT in (b) shows the exact frequency with just four impulses, one for each frequency. This is a text book scenario however, most signals don't have such integer frequencies. In the next case, we change the two frequencies to 2.3 and 4.667 Hz and do a 128 point DFT. We only use 16 collected points, one second worth of data, but we zero-pad these to 128 points. The DFT of this signal does not look like the first case. The energy at both actual frequencies is spread across some number of bins. If we increase the DFT size to 1024 points, the frequency resolution becomes quite small, 16/1024 Hz. The spectrum becomes more like the ideal, with impulses at the two frequencies. The low frequency signal amplitude are nearly equal to (b). The higher signal energy is spread in two bins. We see the leakage as measured by the height of

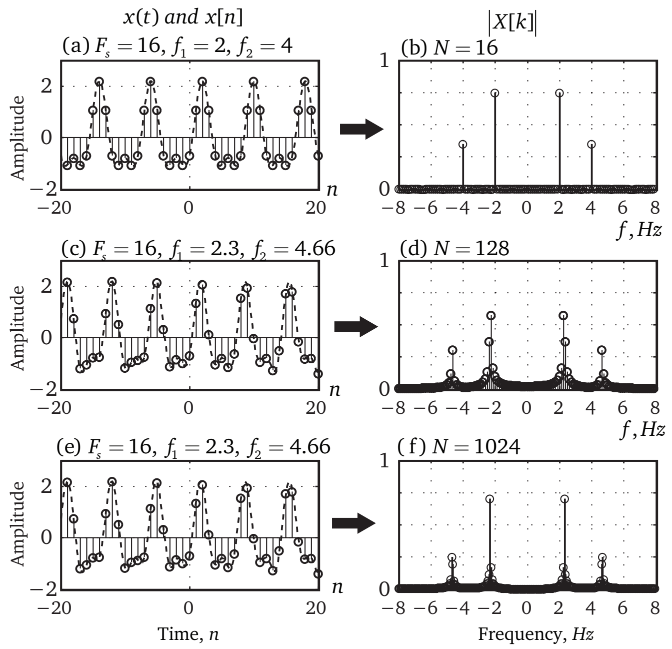

Figure 6.27: The DFT length vs. leakage effect. As the length increases, leakage decreases.

the smaller lobes compared to the main lobe, is smaller when the DFT size is 1024 points, vs. when it is 128 points.

Important observation: DFT size should be as large as practically possible.

Reducing leakage by windowing

How can we reduce this leakage/smearing due to truncation other than by increasing the DFT size? We note that periodic signals match at their end points. We can fake periodicity by just forcing the end points of any signal to zero. If we do that, the piece will have no discontinuities, the right side matches the left since they are both zeros. This we can accomplish by multiplying the time domain signal with a shape that forces the end points to zero. This is artificially distorting the signal, literally bending the signal down to zero.

After forcing the end points to zero, the signal appears periodic. Zero ends mean that this is one period and the next period, also with zero ends will slide right up against it hence completing the illusion of periodicity in the "mind" of the DFT. A rectangular window cuts the signal where it is without distorting any points, so to force points to zero requires a non-rectangular window, one that tapers to zero. No matter what the amplitude of the last few points, such tapering windows push the points down to zero. Windowing changes/distorts the signal and in that it is quite different from zero-padding.

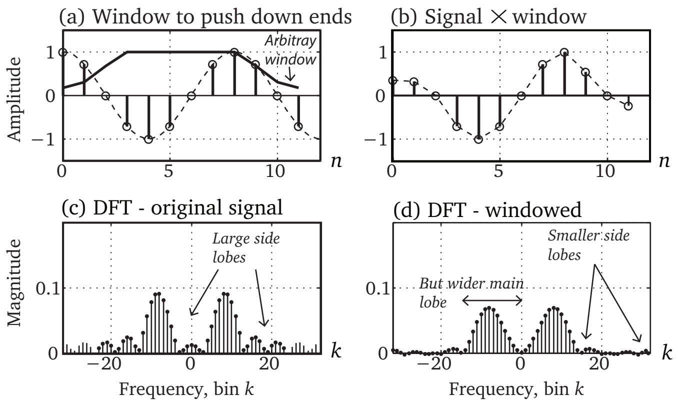

Figure 6.28: (a) sinusoid samples and a "window", (b) the windowed samples, note the end points (b) DFT of the un-windowed samples, (d) the DFT of the windowed samples. The DFT in (d) has the least amount of noise, although the lobes are wider.

In Fig. 6.28(a), we have created an arbitrary window function, shown in solid line in order to force the end points to near-zero. In (c), we see the samples multiplied by this window function. The end points are now near zero. Its DFT in (d) shows some improvements over the one for the actual samples in (b). Comparing the DFTs, in (b) and (d), the noise is less in (d) but not much else. The main lobe is a bit wider as well. Can we do better?

We say to ourselves: perhaps there are other window functions that work better, better at reducing the noise as well as enhancing the main signal components. Yes, there are. We will discuss this topic, called Windowing in Chapter 7 as it is important in dealing with leakage introduced into the spectrum from injudicious (but all too common) truncation effect.

DFT and Multi-rate processing

Up-sampling and seeing images

let's take 20 samples of an arbitrary signal and zero-pad it to 128 samples. But wait; do we add the zeros at the end of the samples, as in Fig. 6.29(a) or do we do this as in Fig. 6.29(b), half on each side? Which method is better?

Previously we said that zero padding means appending zeros at the end of the signal. But how about if we zero-pad on each side? Would that make a difference in the DFT. Well, yes and no. Zero padding on one side is basically a time shift, phase will change but the magnitude will stay the same per the time-shift property. So if we are not interested in phase (and yes, phase can often be ignored) we can add the zeros on either side of the signal.

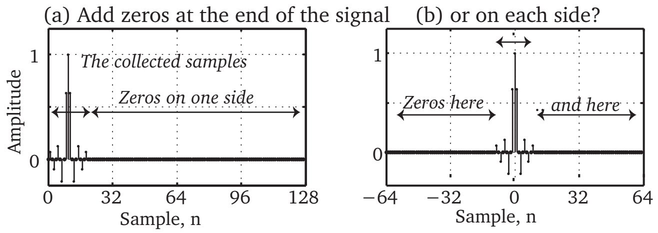

Figure 6.29: Where to add the zeros to the samples? At the end or on each-side? Does it make a difference?

Now we ask, what happens, if instead of adding zeros at the end the signal, we stick them in between the samples. That is, we spread out the collected samples.

Imagine two impulses around the origin in time domain. This corresponds to a cosine in frequency domain. Now imagine inserting extra zeros outside of the two impulses, what happens in frequency domain? Nothing. The signal retains its frequency domain behavior. Now add zeros between the origin and the location of the impulses. This changes things, and now the result is a different signal. The two impulses when moved further apart represent a different frequency. Hence when we are inserting zeros between the samples, we are changing the frequency response of the signal. This case of inserting zeros in between the samples is not benign as is adding zeros at the ends.

In Fig. 6.30(a) we see an arbitrary signal with its DFT in (b). We expand the signal by inserting three zeros in between each sample and then do its DFT again. The effect is surprising in (d). The interpolation by a factor of 4 in time domain, results in the spectrum compressing and replicating 4 times. What is happening here? Is this same as alias replication with sampling frequency? The answer is no, it is not. These three additional copies are called **images** of the spectrum. They are a result of a whole new complication.

In Table 4.1, we give a property of the CTFT called scaling. The property says that if a time domain signal is scaled by a factor a, then its Fourier transform scales by the inverse of the scale factor.

$$\begin{aligned} x(t) &\longleftrightarrow X(\omega) \\ x(at) &\longleftrightarrow \frac{1}{|a|}X\left(\frac{\omega}{a}\right) \end{aligned} \tag{6.19}$$

When we insert zeros in between the samples, we are actually changing the essential nature of the signal and its frequency distribution. We are changing the sampling rate of

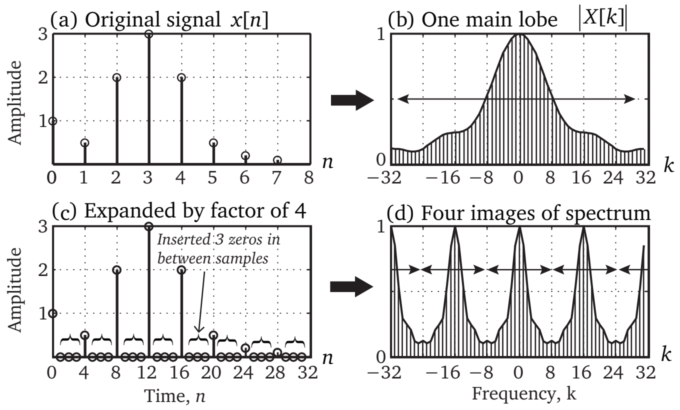

Figure 6.30: Interpolating a discrete signal with zeros causes the DFT to develop equal number of images of the spectrum for each period.

the signal. This brings us into the realm of multi-rate signal processing. ***Multi-rate signal processing*** is often necessary when different parts of a system have different bandwidths and require different sampling frequencies in order to keep their sampling rate at the Nyquist rate. Why do you need this? We often have disparate systems working at their own speed, so to speak. For example when a singer goes to a studio to record her music, it is usually done at a rate of 192 kHz. Now assume the studio needs to press a CD for her. The CD audio format is standardized to be sampled at 44.1 kHz. To put her song on a CD, requires down sampling at a ratio 44/192. This down-sampling is part of a process generally referred to as multi-rate signal processing. This section is only intended to give you the basic concepts involved in multi-rate processing.

To make multi-rate sampling work, filtering the images is necessary and hence the term ***upsampling*** becomes ***interpolation*** when combined with filtering. Similarly when a signal is down sampled, it too has to be followed by filtering, a process called ***decimation***.

In Fig. 6.31(c), we see signal (a) that has been up-sampled by a factor of 5. We note that its spectrum in (d) is indeed images and not the replicated spectrum. The images are reversed in pairs because the DFT of the un-upsampled signal is not symmetrical.

Let's see if we can examine the mathematical basis for this replication. Assume that we are going to insert M-1 zeros between each sample of an N-sample signal. The new signal generated from the old signal is given by

$$y[Mn] = \begin{cases} x[n] & \text{for } M = \pm 1, \pm 2, \ldots \\ 0 & \text{else} \end{cases} \tag{6.20}$$

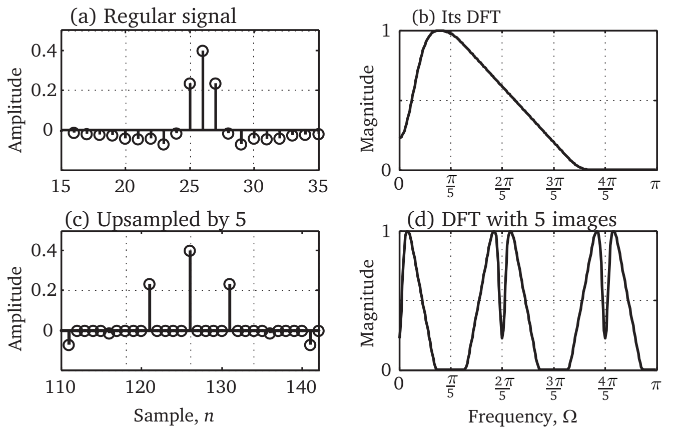

Figure 6.31: Example of a band limited up-sampled spectrum.

In this case if $M = 4$, then $y[4] = x[1]$, $y[8] = x[2]$, $y[12] = x[3]$ and in between values are all zeros. We calculate the DFT of this stretched, up-sampled or interpolated signal by

$$Y_s[k] = \sum_{n=0}^{MN-1} y[Mn] W_{MN}^{nk} = \sum_{n=0}^{N-1} x[n] W_{MN}^{nk} \tag{6.21}$$

The index MN is changed to N, as it is not a factor of M, and we change k to $0, 1, \dots, NM-1$, we rewrite this as

$$Y_s[k] = \sum_{m=0}^{N-1} x[m] W_N^{mk} = Y[k] \tag{6.22}$$

This says that we get a spectrum at every N samples instead of NM samples. The spectrum of the up-sampled sequence is same as that of non up-sampled sequence but repeats over M periods. The stretching via the addition of the zeros in between the samples, does a shrinkage of the resolution of the spectrum. However it is still an exact sampling of the DTFT and can be filtered and the original signal can be recreated from a single image.

Down-sampling

When a signal is down-sampled, we are doing the inverse. The signal is being sampled at slower speed by a factor of M, as we can see in Fig. 6.32. The DFT of the signal is limited to $\pi/5$ in (b) but when the same signal is down-sampled by a factor of 3, its DFT spreads by a factor of 3 as in (d). At this rate of down-sampling, the time-domain signal still retains some similarity to the original signal and hence its spectrum is nearly identical in shape to the original. However when the same signal is down sampled by a factor of 6, it both loses its similarity in time domain to the original and its frequency response. This is intuitive. You cannot keep removing information and still expect to be retain the essential nature of the signal.

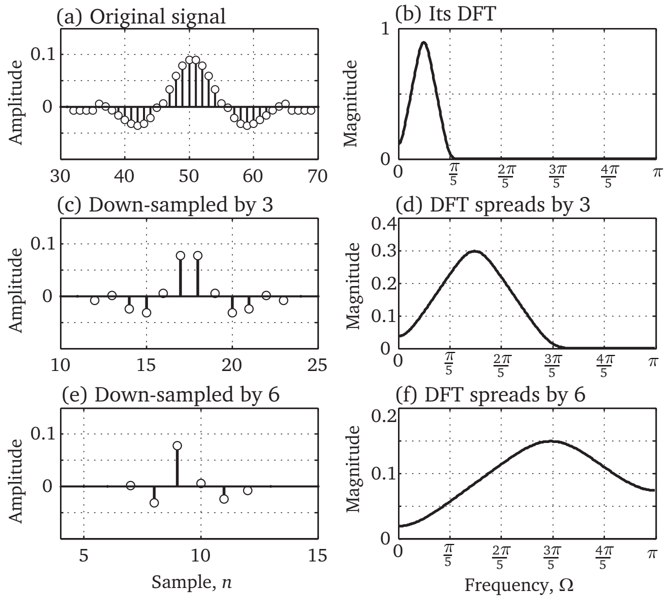

Figure 6.32: Down-sampling of the signal by a factor of 3 *spreads the spectrum by a factor of* 3. *Larger down-sampling as in (e) may distort the spectrum depending on the spectrum bandwidth.*

Interpolation

Interpolation is a form of guessing. Given two points, we find it easy to guess the middle value. The up-sampled signal, when it has zeros in between the samples, is a signal that has not been interpolated. At these zero points we have no information. But what if we do some sort of interpolation on these intermediate samples. The process of interpolation consists of guessing these intermediate values based on available data. The simplest thing we can do is sample-and-hold operation on the signal. Or perhaps a different interpolation method. How would that effect the DFT? In Fig. 6.32, we see three different types of interpolations of the signal (a) with zero-insertion which is basically up-sampled, (b) with zero-order hold, in (c) first order hold or linear interpolation and (d) sinc interpolation. In Fig. 6.33, we see the DFT of each version. Only the DFT of zero-insertion, gives us significant images. The reason is that the other methods by having interpolation performed on them have in effect filled in the zeros, whereas no such interpolation has occurred for the zero-insertion case and hence all the images are present for to us to see. Perhaps it is a way of the FT telling us that it has no information so it guesses from what it knows. Interpolation process *adds information* and in the process attenuates the images which represent ambiguity. The samples could belong to sinusoids of higher frequencies in the images. The interpolation removes this ambiguity with the extra information by changing the zeros to data.

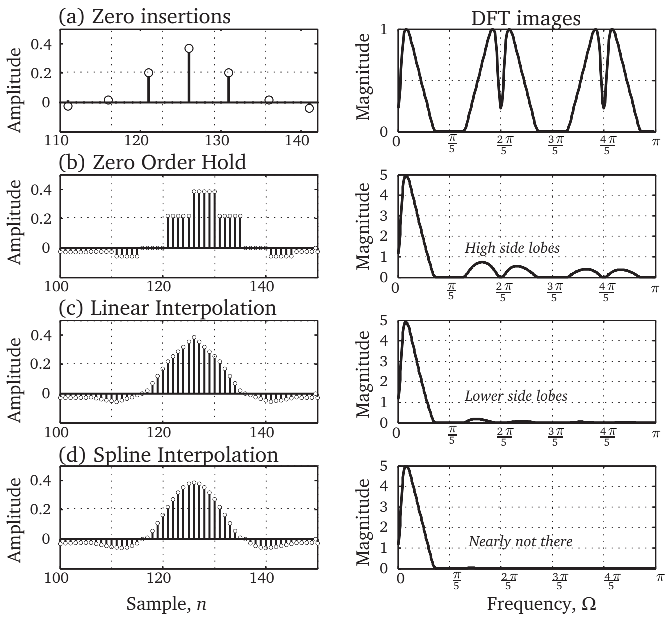

Figure 6.33: The DFT of the signal with various types of interpolation
DFT of the zero-added signal shows M images, (b) DFT of the zero-order hold signal still has some level of images (c) DFT of the first-order hold signal and (d) the DFT of the spline interpolation are nearly perfect with no images.

Convolution by DFT

The convolution property is one of the most utilized properties of Fourier transform. This property states that the multiplication of two functions in one domain is convolution in the other domain. Convolution is an important mathematical function in signal processing. We can accomplish both the linear and circular convolution using the DFT far quickly than using the straight multiply-add method.

What is the difference between a linear and a circular convolution? First of all, linear convolution convolves two independent signals and comes up with a result, the length of which is different from the two input signals. Intuitively speaking it is somewhat like correlation and is most commonly used to compute the response of a system to an impulse. We hit the system with an impulse, watch how it responds, by looking at the output and call that the impulse response of the system. The response is mathematically defined as the convolution of the impulse with the characteristic behavior of the system. The signals or data being convolved do not have to be of equal length. Linear convolution makes no assumption about the periodicity of the two signals. Circular convolution on the other hand is only defined for equal length periodic signals. Because the input functions are periodic, the convolution is also periodic and so the result must be fully specified by one of its periods. Circular convolution comes into discussion only when we want to do linear convolution using the DFT.

To see an example of the two types of convolution, we select two arbitrary signals, $x[n] = [1 \quad 2 \quad 3 \quad 4]$, and $h[n] = [1 \quad -1 \quad 1 \quad -1]$.

Let's do their linear convolution, which means to hold one signal fixed, reversing the other and then doing a point by multiplication of all overlaps and adding these to get a value at that shift. We use Matlab to do the linear convolution using the command, `conv`, to get the following sequence.

$$y = [1 \quad 1 \quad 0 \quad 0 \quad -5 \quad -1 \quad 4]$$

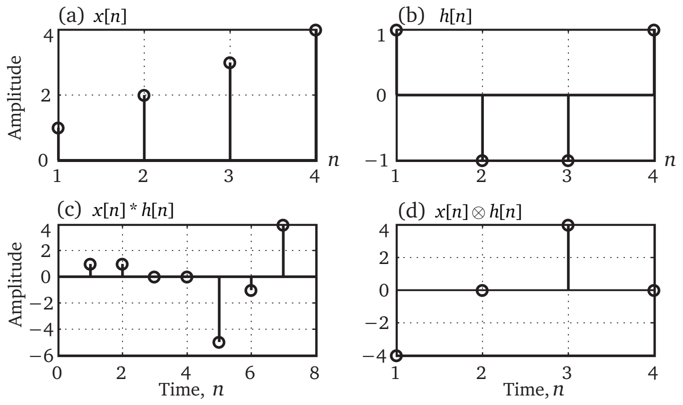

Figure 6.34: Linear vs. circular convolution, (a) and (b) are two sequences, (c) linear convolution and (d) circular convolution.

Now we will use the DFT to compute the convolution. Since the signal length is 4 points, we will just do a 4 point DFT. We get the following two DFT's.

$$X[k] = [10.000 \quad 2.8224 \quad 2.000 \quad 2.8284]$$
$$H[k] = [0.000 \quad 2.8284 \quad 0.0000 \quad 2.8284]$$

We multiply these and do an inverse DFT to see if we get the same y sequence. No need to do the DFT, as we know that we will not get a 7 point sequence from the multiplication of two 4 point DFTs. However let's take a look at what we get.

$$\tilde{y} = [0 \quad 4 \quad 0 \quad 4]$$

In Matlab, we invoked `cconv(x, h, 4)` command to get this result. This is the ***circular convolution*** of the two sequences, x and h, and not linear convolution. Hence the DFT convolution property works only with circular convolution and not linear convolution. The process of circular convolution is different from linear convolution. So what if we wanted linear convolution and not circular? We can get the linear convolution, if we pad each signal to be equal in length to $\text{length}(x) + \text{length}(h) - 1$ or greater.

Why do we even bother with the DFT to compute the convolution if can do it directly anyway? The reason is we find that DFT is in fact a computationally efficient method of computing linear convolution. Machines don't like to compute convolution either, it's that tedious! The actual reason is a practical one. If a device already has a built-in facility to compute a FFT, then why design a new algorithm to compute the convolution, why not use the DFT for convolution too. But if not done right, the answer DFT gives is the circular convolution. Algorithms have been developed to take this into account and to automatically extend and zero-pad the signals to produce the correct answer. Using the DFT to compute the convolution is often called **Fast convolution**, since it requires less computational effort than the direct method.

There are some well-known methods that make these calculations efficient. The two most popular methods are called Overlap-and-Add and the Overlap-and-Save method. In both of these methods, the input signal is subdivided in smaller blocks which are convolved with the impulse response of the filtering sequence h[n]. This subdivision reduces the number of multiplications and makes both of these algorithms very efficient for computing convolution.

Short-Time Fourier transform

To have a valid Fourier transform a signal must have stable power with time, and its frequency range does not change. So what about a signal like the following? This is a signal where the frequency is linearly increasing with time.

$$x[n] = A\sin(2\pi f_0 n^2).$$

This type of signal is called a **chirp signal**. Here we plot one with $f_0 = 100$ Hz and $A = 0.5$. In Matlab, we frame such signals with three parameters shown in 6.35(a). Here the starting frequency of 100 Hz is expected to become 300 Hz in one second time.

We could run this through a DFT but we know this is wrong. Can we even use the DFT on such a signal? Like the engineer who invented the STFT, we go ahead and give it try recognizing that if we make the DFT size small, then we can assume that the frequency is not changing over this short interval. Maybe we can still use the FFT to extract useful information from this signal. The way to do that is to subdivide the signal into several very short time intervals. This allows us to treat the signal as stationary and time-invariant over a small time period. We can do a DFT on each small segment and then look at the changes in the DFTs of these segments in time. People have learned to read and use this time-varying DFT, called the **Short-time Fourier Transform**. STFT has three dimensions, time, frequency and the magnitude. Plotting all three will generate a 3-D plot. A 2-D version with just 2 of these dimensions is called **Spectrograph** as implemented in Matlab. The spectrograph uses color to denote magnitude. In Fig. 6.35, we show the STFT of this chirp signal computed in Matlab by using two different set of parameters. We note in each of the two STFTs, the y-axis is time and x-axis is frequency. We see that the signal frequency starts at 100 Hz and is about 300 Hz one second later. We see in both Fig. 6.35(b) and (c) that the frequency varies linearly with time, and that is true since we chose a linear chirp. The fuzzy-ness of the line indicates the range of frequencies over each time range. STFT is often used for audio signals, where variety of qualitative assessments can be made on a signal using the STFT. One information jumps out at us, and that is that the frequency of this signal is varying linearly.

There is a trade off between the time interval and the frequency resolution. To get better knowledge of the frequencies in an interval, the interval must be short enough that so the frequency content doesn't change significantly. If the period is too long, there is some confusion about the frequencies in the interval. However, shortening the interval will increase the effect of rectangular truncation effect.

DFT in real-life

When we have a large amount of data and we select a subset of it to do a DFT, how do we know which part to choose, where to start?

Fig. 6.36(a) shows both I and Q channels of a signal. The signal has gone through a non-linear amplifier which distorts the input signal in both amplitude and phase. We note that the signal has some transient behavior at the beginning. This would not be a good place to start the DFT since Fourier transform requires a signal that is unchanging in average power.

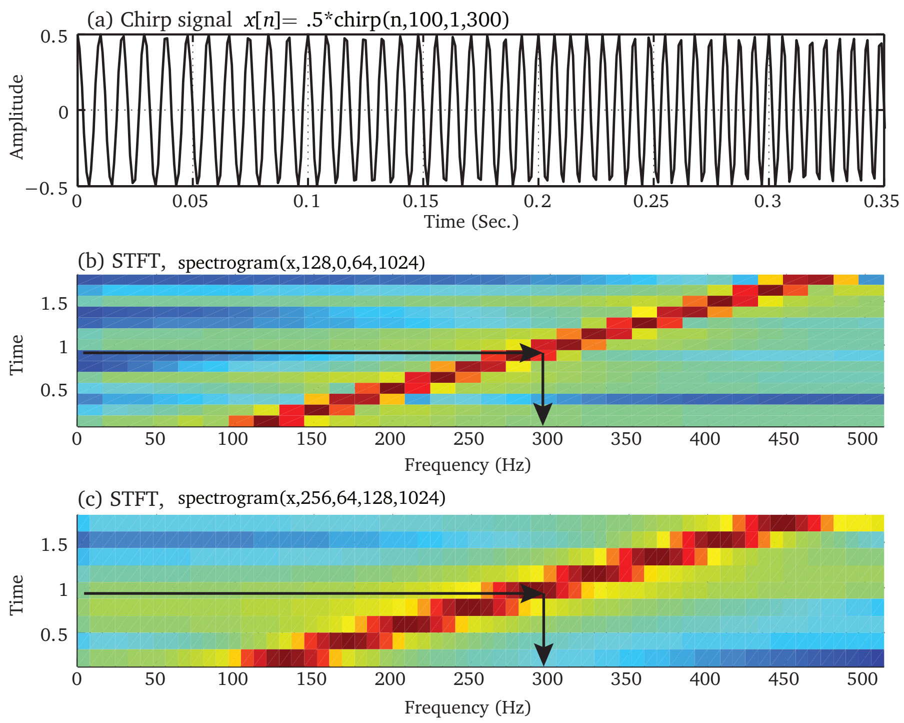

Figure 6.35: Short-time Fourier transform (a) a chirp signal with $f_0 = 100$ Hz, (b) Spectrogram with segment length $= 128$ and no overlap, (c) Spectrogram with segment length $= 256$, overlap $= 64$.

Hence we should skip this part. We ought to pick something in the middle or at the end, and select a number of samples that we have already determined we need.

Fig. 6.37 shows 4096 point DFT of the signal, one done at the start which has a transient nature and the second done on the steady-state part of the signal. The signal contains no noise and yet look how noisy it is, unlike any of the ones we have been seeing. The reason for this is that this is a *random signal*. We were not particularly careful where we started the DFT. Ideally we should start at the beginning of a period. We should also sample at a rate that is an integer multiple of the period. But this is often an impractical condition to meet. We work with what we have.

We see that the two DFTs are different, (note the arrows at shoulders and other key points in the figure) often by several dBs. The DFT of such signals has a lot of variance. Such DFT is hard to read and varies far too much from one bin to the next.

This variability of the DFT is quite bothersome but cannot be avoided. We are interested in judging the average level of the signal, and we find it varying as much as 5 to 10 dB from bin to bin. For this reason direct DFT is not often used to compute the power spectrum of

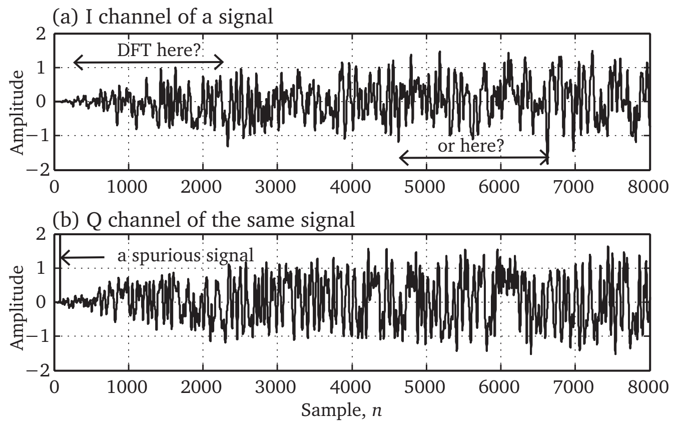

Figure 6.36: A large amount of collected data. Where to do the DFT?

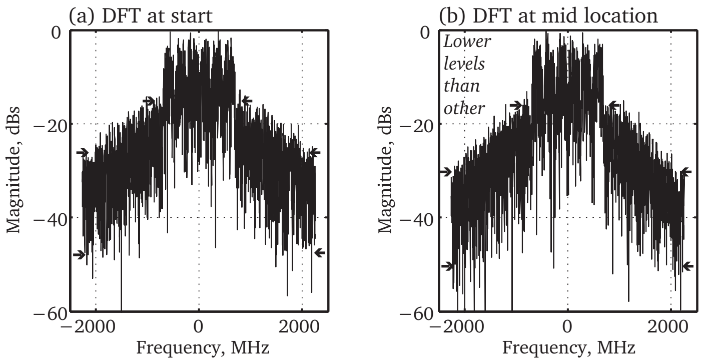

Figure 6.37: The (DFT) of a random digital signal done at two different points in the signal. The DFT is different because the signal for case (a) is in a transient or changing zone. Note the level at the shoulders.

random signals. In Fig. 6.38 we plot a smoothed version of the DFT called the **Periodogram**. This clearly looks much better than Fig. 6.37 but these looks are obtained at a cost. This smoothing sacrifices resolution for reduced variance and ease of qualitative understanding. This is a topic we discuss in chapter 8.

Fast Fourier Transform (FFT)

The acronym, FFT is so common that we use without thinking, often synonymously for DFT.

The DFT although clear and easy to compute, requires a good many calculations. For each harmonic, we have N multiplications and we do this N times, giving us approximately N^2 calculations, referred to as multiply-and-accumulate (MAC) operations in DSP. A 256

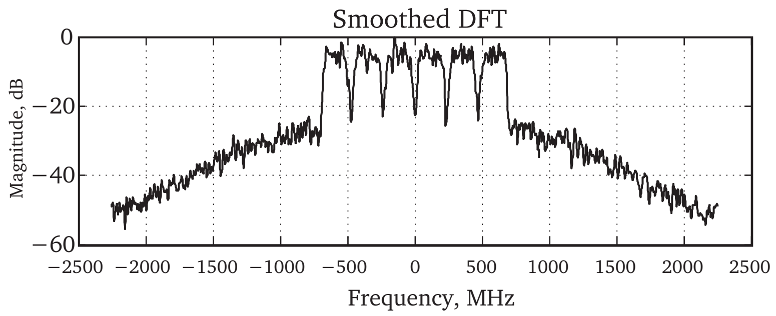

Figure 6.38: A real-life power spectrum of acceptable levels of amplitude variance. The method used to create this spectrum is called the Periodogram.

point DFT would require 65, 792 calculations. The algorithm for Fourier transform existed for more than 200 years before it came into widespread use mostly because before the advent of the digital computers we could not cope with such large number of computations. The algorithm waited for the development of the microprocessor.

In 1965, Cooley and Tukey came up with a computational breakthrough called the Fast Fourier transform algorithm. This was an ingenious manipulation of the inherent symmetry in the calculations. It allowed the computation of an N point DFT as a function only $N \log_2 N$ instead of the N^2. So a 256 point DFT would only require 2048 MAC calculations, a huge improvement from 65, 792 calculations doing it the long laborious way. The algorithm was quickly and widely adopted and is the basis of all modern signal processing.

Most DSP books spend a lot of time going through the mechanism of the FFT and how it is computed and implemented. All those butterfly figures in books help to confuse and confound us as to what is actually going on. Most of us just give up on it. Let me alleviate your guilt. Although extremely important in itself, the understanding of the mechanism of how the FFT is "calculated" is not all that important. So, we are not going to cover FFT calculation algorithm in this book. This statement maybe an anathema to some reading this book. We justify that omission by noting that this is a book about understanding the Fourier transform and how to apply it. Its efficient computation, we leave to others.

The FFT algorithm has been programmed in all sort of software and we can safely skip it without impacting our understanding of the ***application*** of the DFT and the FFT. In Matlab, you can do a FFT of any size. Of course in hardware, efficiency is important so the FFT algorithm (and there are several versions) is based on signal size N, such that N is a number generated by an integer power of 2 or is equal to $N = 2^k$, with k an integer. This is not a big issue. The only time we find ourselves wanting to do DFT/FFT of a number other than a power of 2 is in textbooks and for illustrative purposes. Even in Matlab if you use a signal that is not a power of 2, it uses a slower version of the FFT. We assume that the FFT implementation is a black-box, into which we feed the data. As long you understand *what* this box is doing, understanding how efficiently it is doing its job is not that important.

Summary of Chapter 6

In this chapter we developed the discrete Fourier transform (DFT). The DFT, a finite length transform can be considered a sampled version of the DTFT. How good the DFT is at representing the true DTFT depends on the chosen sampling frequency.

Terms we introduced in this chapter:

- **DFT** - Discrete Fourier Transform, a sampled version of the DTFT
- **Bin number** - Individual frequency component identified by a number from 0 to N-1, with N the size of the DFT.
- **DTFT estimation** - Estimating a DTFT by doing a very long DFT
- **Leakage and smearing** - When the DFT does not contain an integer number of periods, we get leakage, also called smearing.
- **Truncation** - Any extraction of data from another longer signal results in a default truncation effect.
- **Windows** - The data can be made to look periodic by multiplying it with an another time-domain function, called a Window.
- **Multi-rate processing** - Change in sampling rate of a system
- **Images** - When a sampling rate is increased by a factor a, the new spectrum contain a images of the original spectrum.
- **STFT** - Fast Fourier Transform

1. The DTFT spectrum is continuous with frequency and requires an infinite length signal, hence it is not practical for computers.
2. For finite length signals, we use the DFT. The DFT assumes that the data available is a one basic period of some arbitrary signal.
3. The DFT is a sampled version of the DTFT computed at selected frequencies. These selected frequencies are given by

$$\Omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, N-1$$

   where N is the size of the DFT.
4. The DFT as samples of DTFT is given by

$$X[k] = X(\Omega_k) = X(k\Omega_0) = X\left(\frac{2\pi k}{N}\right)$$

5. The DFT is formulated from the DTFT for a finite number of points as given by

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi k}{N}n}$$

6. The DFT takes N uniformly placed time domain samples $x[n]$ and creates a spectrum which we call the DFT.
7. The x-axis of the DFT can be given three ways; by k the bin number, by digital frequency over the range $-\pi$ to π or by the frequency, kF_s/N in Hz. The spectrum is discrete and repeats with the sampling frequency.
8. The spectrum is said to cover 2π radians or in frequency from $-F_s/2$ to $+F_s/2$ where F_s is the sampling frequency of signal.
9. Zero-padding can be used to improve the interpolation density. It is like using more points to plot a curve given its equation. It does not change the *equation*, only the number of points plotted.
10. when the DFT size of a single sinusoid is equal to the integer number of periods, the DFT falls on the zeros of the underlying sinc function and we get the ideal DTFT of a sinusoid.
11. Leakage occurs when the DFT size is not equal to an integer multiple of the period of each and every frequency component in the signal. The DFT of individual sinusoids do not fall on zeros of the sinc function and we see this as a form of smeared spectrum. Leakage is unavoidable in real signals.
12. The bin frequency resolution is given by sampling frequency divided by the size of the DFT, N.
13. The frequency of each bin is bin number times the resolution, kF_s/N.
14. We often do not know the period of real signals. However by looking at the ends of the signal, we can tell if we will get leakage. If the ends do not match, then leakage will be present due to the end mismatch.
15. Windows functions can be used to force the end points to zeros, so the signal (artificially) becomes periodic.
16. Leakage can be mitigated by doing a larger length DFT or by widowing.
17. DFT can have large spectral variance.
18. DFT can be used to implement fast convolution.
19. STFT can be used to assess signals which are not normally amenable to Fourier analysis, such as when the signal frequency is changing rapidly.
20. Fast Fourier Transform is an algorithm that uses signal lengths of 2^N. It speeds up the computation using symmetry properties of the DFT matrix.

Questions

1. What is the conceptual difference between a DTFT and DFT?
2. The DFT is a sampled version of both the DTFT and the CTFT. True or false?
3. Given N samples for a DFT, what can we say about the sampling frequency of this signal?
4. Given 9 samples, we zero-pad it to a length of 90 samples and reduce the sampling frequency by one-half. By what factor does the resolution change?
5. Zero-padding can improve the resolution of the DFT and of the DTFT. True or false?
6. What is the DFT of this signal: $x[n] = [0101]$?
7. The DTFT and the DFT repeats with what frequency?
8. DFT is a linear process. True or False?
9. The DFT size is N = 1024, the sampling frequency is 2048 Hz. What is the frequency of the components at $k = 10; -100; +200$?
10. What is the DFT of a constant function?
11. A signal has two frequency components 1200 Hz apart. The signal is sampled at a rate of 48 KHz What size DFT should we do to see both of these components clearly.
12. What is the CTFT of this complex exponential: $e^{(-j12t)}$.
13. What does the real part of the DFT of a complex exponential look like?
14. If the DFT of $x[n]$ is $X[k]$, then what is the DFT of a time shifted function $x[n-2]$?
15. What is the bandwidth of the main lobe of the DFT of a square pulse of period 1 second?
16. We do an iDFT on a function shaped like a sinc, with main lobe bandwidth W. What is the shape of the time-domain pulse and its time.
17. We want a signal of bandwidth 5 Hz. What sinc function will produce this result?
18. Find the DFT of $A\sin(\Omega t)$. How different is it from $A\cos(\Omega t)$.
19. Under what conditions will the DFT a sinusoid match the ideal?
20. An even signal has an even DFT. What does that mean?
21. The DFT is an estimated version of the DTFT, True or False?
22. Why does zero-padding improve the spectrum of the signal?
23. Can we add the zeros to the left or right of the data? What effect does this have?
24. Why is the DTFT of the square pulse with center at 0 not the same as one that starts at zero.
25. When doing the DFT, why is it important to have an integer number of cycles of data?
26. Write the DFT matrix for size 2.
27. If we have 2 cycles of data vs. 4 cycles, will the DFT be the same? What about 2.5 cycles vs. 5 cycles?
28. DFT size must be power of 2, True or False?
29. FFT and DFT are the same, True or false?

30. DFT is similar to discrete Fourier series in what sense?