CSE 4631

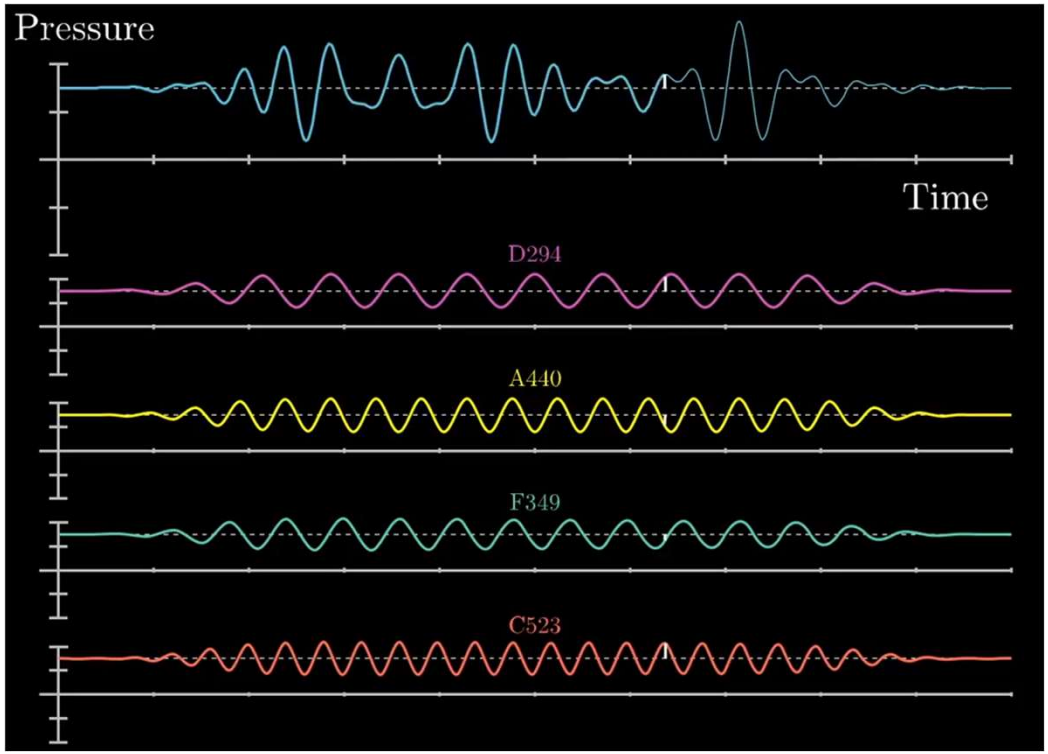
Chapter 8 - Smith

Md. Zahidul Islam

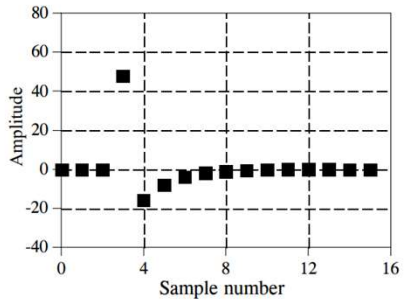
Lecturer, CSE, IUT

The Discrete Fourier Transform

- Fourier analysis is a family of mathematical techniques, all based on decomposing signals into sinusoids.

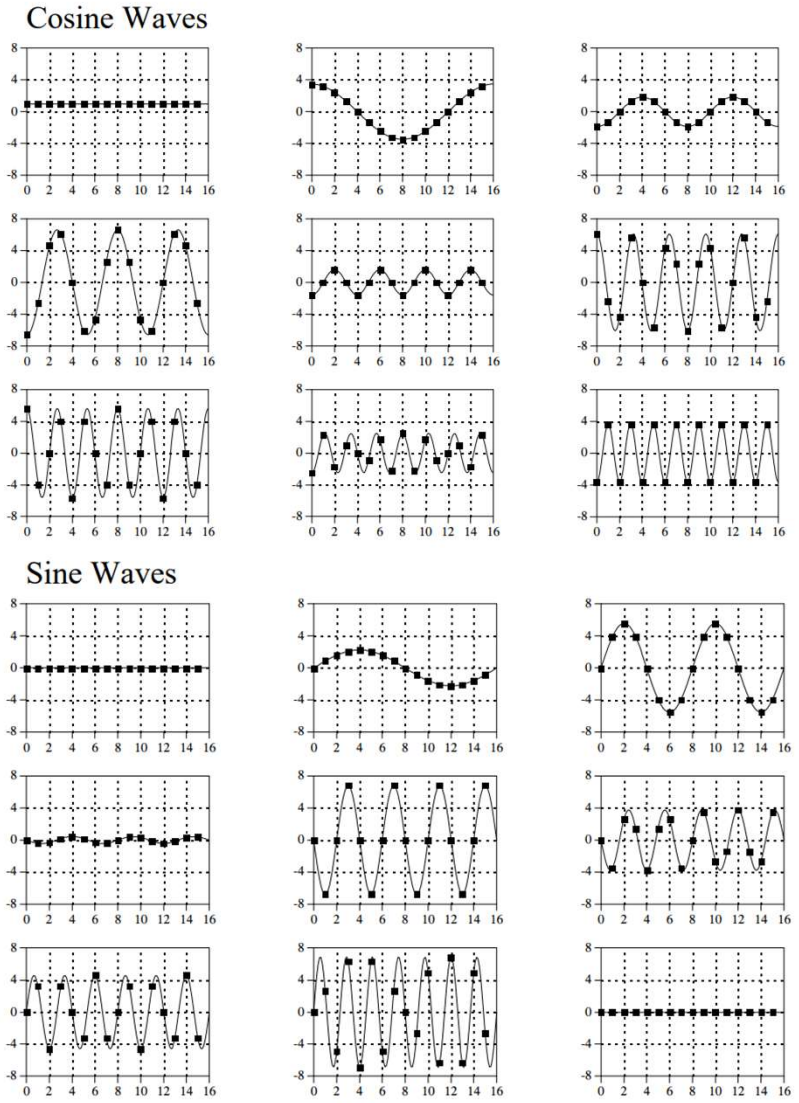


Fourier Decomposition and Synthesis



DECOMPOSE

SYNTHESIZE



Why Sinusoids?

Why are sinusoids used instead of, for instance, square or triangular waves? Remember, there are an infinite number of ways that a signal can be decomposed. The goal of decomposition is to end up with something *easier* to deal with than the original signal. For example, impulse decomposition allows signals to be examined one point at a time, leading to the powerful technique of convolution. The component sine and cosine waves are simpler than the original signal because they have a property that the original signal does not have: *sinusoidal fidelity*. As discussed in Chapter 5, a sinusoidal input to a system is guaranteed to produce a sinusoidal output. Only the amplitude and phase of the signal can change; the frequency and wave shape must remain the same. Sinusoids are the only waveform that have this useful property. While square and triangular decompositions are *possible*, there is no general reason for them to be *useful*.

Four Categories of Fourier Transform

A signal can be either *continuous* or *discrete*, and it can be either *periodic* or *aperiodic*. The combination of these two features generates the four categories, described below and illustrated in Fig. 8-2.

Aperiodic-Continuous
This includes, for example, decaying exponentials and the Gaussian curve. These signals extend to both positive and negative infinity *without* repeating in a periodic pattern. The Fourier Transform for this type of signal is simply called the **Fourier Transform**.

Fourier transform => Aperiodic - continuous
Fourier Series=> periodic - continuous
Discrete time fourier transform => aperiodic-discrete
Discrete fourier transform => periodic-discrete

Periodic-Continuous
Here the examples include: sine waves, square waves, and any waveform that repeats itself in a regular pattern from negative to positive infinity. This version of the Fourier transform is called the **Fourier Series**.

Aperiodic-Discrete
These signals are only defined at discrete points between positive and negative infinity, and do not repeat themselves in a periodic fashion. This type of Fourier transform is called the **Discrete Time Fourier Transform.**

Periodic-Discrete
These are discrete signals that repeat themselves in a periodic fashion from negative to positive infinity. This class of Fourier Transform is sometimes called the Discrete Fourier Series, but is most often called the **Discrete Fourier Transform.**

Four Categories of Fourier Transform

Type of Transform	Example Signal
Fourier Transform *signals that are continious and aperiodic*	
Fourier Series *signals that are continious and periodic*	
Discrete Time Fourier Transform *signals that are discrete and aperiodic*	
Discrete Fourier Transform *signals that are discrete and periodic*	

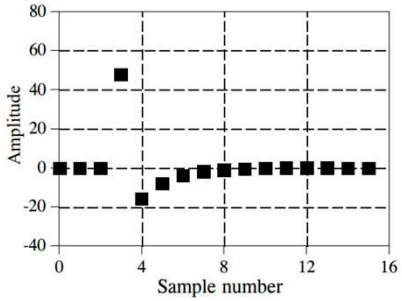
FIGURE 8-2
Illustration of the four Fourier transforms. A signal may be continuous or discrete, and it may be periodic or aperiodic. Together these define four possible combinations, each having its own version of the Fourier transform. The names are not well organized; simply memorize them.

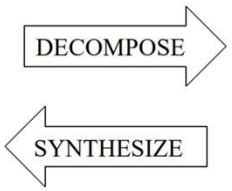
Why are we only interested with Discrete Fourier Transform?

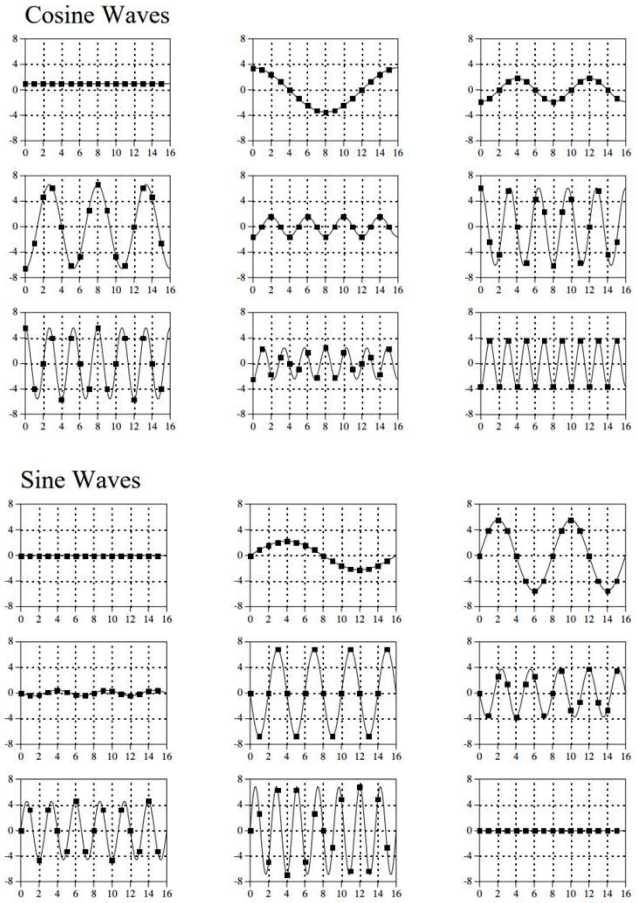
- An infinite number of sinusoids are required to synthesize a signal that is aperiodic. So can't store in a computer memory!

- Computer's can only handle information that are discrete and finite.

An Example of DFT







Look back at the example DFT decomposition in Fig. 8-1. On the face of it, it appears to be a 16 point signal being decomposed into 18 sinusoids, each consisting of 16 points. In more formal terms, the 16 point signal, shown in (a), must be viewed as a single period of an infinitely long periodic signal.

Time Domain and Frequency Domain

As shown in Fig. 8-3, the discrete Fourier transform changes an N point input signal into two $N/2+1$ point output signals. The input signal contains the signal being decomposed, while the two output signals contain the *amplitudes* of the component sine and cosine waves (scaled in a way we will discuss shortly). The input signal is said to be in the **time domain**. This is because the most common type of signal entering the DFT is composed of samples taken at regular intervals of *time*. Of course, any kind of sampled data can be fed into the DFT, regardless of how it was acquired. When you see the term "time domain" in Fourier analysis, it may actually refer to samples taken over time, or it might be a general reference to any discrete signal that is being decomposed. The term **frequency domain** is used to describe the amplitudes of the sine and cosine waves (including the special scaling we promised to explain).

Forward and Inverse DFT

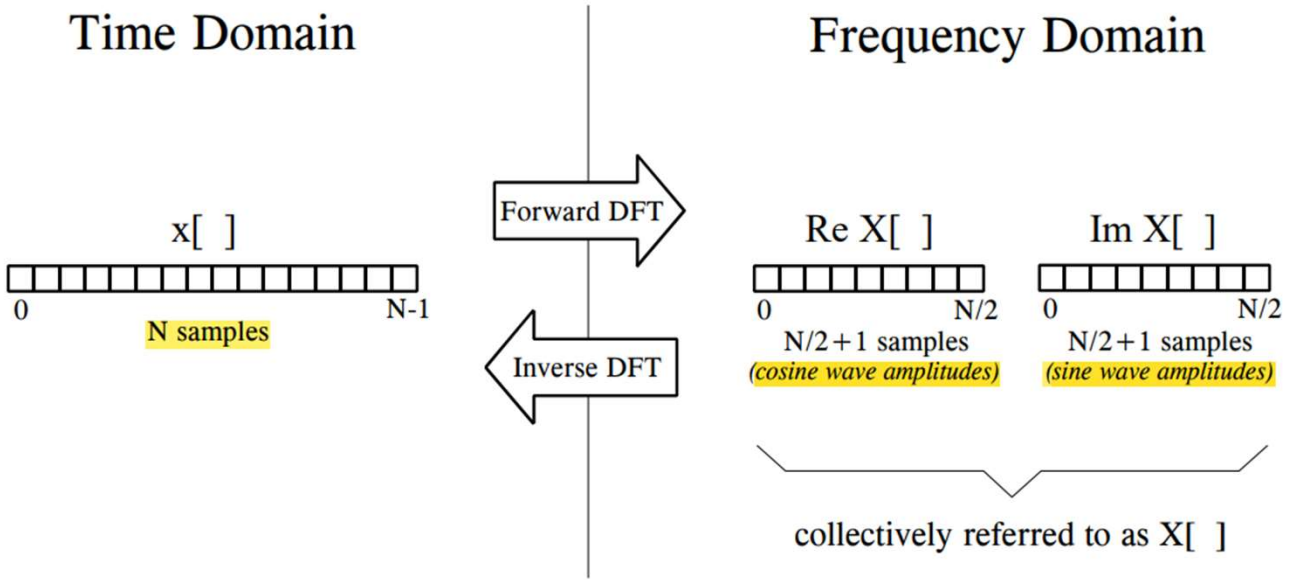


FIGURE 8-3
DFT terminology. In the time domain, $x[\ ]$ consists of N points running from 0 to $N-1$. In the frequency domain, the DFT produces two signals, the real part, written: $ReX[\ ]$, and the imaginary part, written: $ImX[\ ]$. Each of these frequency domain signals are $N/2+1$ points long, and run from 0 to $N/2$. The Forward DFT transforms from the time domain to the frequency domain, while the Inverse DFT transforms from the frequency domain to the time domain. (Take note: this figure describes the **real DFT**. The **complex DFT**, discussed in Chapter 31, changes N complex points into another set of N complex points).

ReX[] and ImX[]

- The output of DFT is this two arrays/signals, ReX[] and ImX[] each containing N/2+1 samples.

- The elements of these two array contain the **scaling factor** of each sinusoid.

- So, each sinusoid gets scaled/multiplied with a corresponding number in the array.

- The Cosines get scaled with numbers in ReX and the Sines get scaled with numbers in ImX. So, these numbers are basically **Amplitudes** of those sine and cosine waves.

We prefer choose signals with N samples where N is power of 2. Why?

The number of samples in the time domain is usually represented by the **variable *N***. While N can be any positive integer, a power of two is usually chosen, i.e., 128, 256, 512, 1024, etc. There are two reasons for this. First, digital data storage uses binary addressing, making powers of two a natural signal length. Second, the most efficient algorithm for calculating the DFT, the Fast Fourier Transform (FFT), usually operates with N that is a power of two. Typically, N is selected between 32 and 4096. In most cases, the samples run from 0 to $N-1$, rather than 1 to N.

Time domain and Freq domain signal notations

Standard DSP notation uses **lower case letters** to represent time domain signals, such as $x[\ ]$, $y[\ ]$, and $z[\ ]$. The corresponding **upper case letters** are used to represent their frequency domains, that is, $X[\ ]$, $Y[\ ]$, and $Z[\ ]$. For illustration, assume an N point time domain signal is contained in $x[\ ]$. The frequency domain of this signal is called $X[\ ]$, and consists of two parts, each an array of $N/2 + 1$ samples. These are called the **Real part of $X[\ ]$**, written as: ***Re* $X[\ ]$**, and the **Imaginary part of $X[\ ]$**, written as: ***Im* $X[\ ]$**. The values in $Re\,X[\ ]$ are the amplitudes of the cosine waves, while the values in $Im\,X[\ ]$ are the amplitudes of the sine waves (not worrying about the scaling factors for

- We call them real and imaginary just for convention. Actually, both of these arrays will contain **real** numbers.
- Just Know that,
  - **Real Part = Cosine Wave Amplitudes**
  - **Imaginary Part = Sine Wave Amplitudes**

Horizontal Axis of Freq Domain written in Four ways

- 0 to N/2 - k
- 0 to 0.5 - k/N
- 0 to 2*pi*0.5 - 2*pi*k/N
- 0 to SR*0.5 - SR*k/N

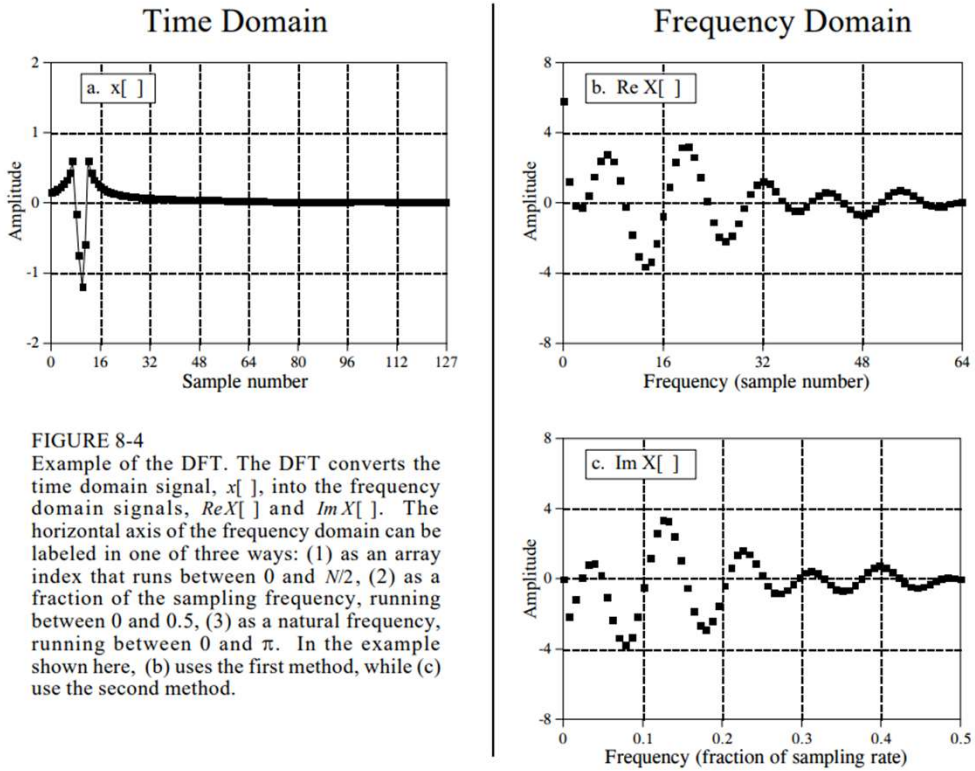


FIGURE 8-4
Example of the DFT. The DFT converts the time domain signal, *x*[ ], into the frequency domain signals, *Re X*[ ] and *Im X*[ ]. The horizontal axis of the frequency domain can be labeled in one of three ways: (1) as an array index that runs between 0 and *N*/2, (2) as a fraction of the sampling frequency, running between 0 and 0.5, (3) as a natural frequency, running between 0 and π. In the example shown here, (b) uses the first method, while (c) use the second method.

Where does those sine and cosine functions come from?

They are called **DFT Basis Functions.** **They are bunch of sine and cosine waves with unity amplitude and different frequencies.**

The DFT basis functions are generated from the equations:

EQUATION 8-1
Equations for the DFT basis functions. In these equations, $c_k[i]$ and $s_k[i]$ are the cosine and sine waves, each N points in length, running from $i = 0$ to $N-1$. The parameter, k, determines the frequency of the wave. In an N point DFT, k takes on values between 0 and $N/2$.

$$c_k[i] = \cos(2\pi ki/N)$$

$$s_k[i] = \sin(2\pi ki/N)$$

Each value of k produces one cosine and one sine wave. And k ranges from 0 to N/2.

So, N/2 + 1 Cosine waves.

And, N/2 + 1 Sine waves.

DFT Basis Functions

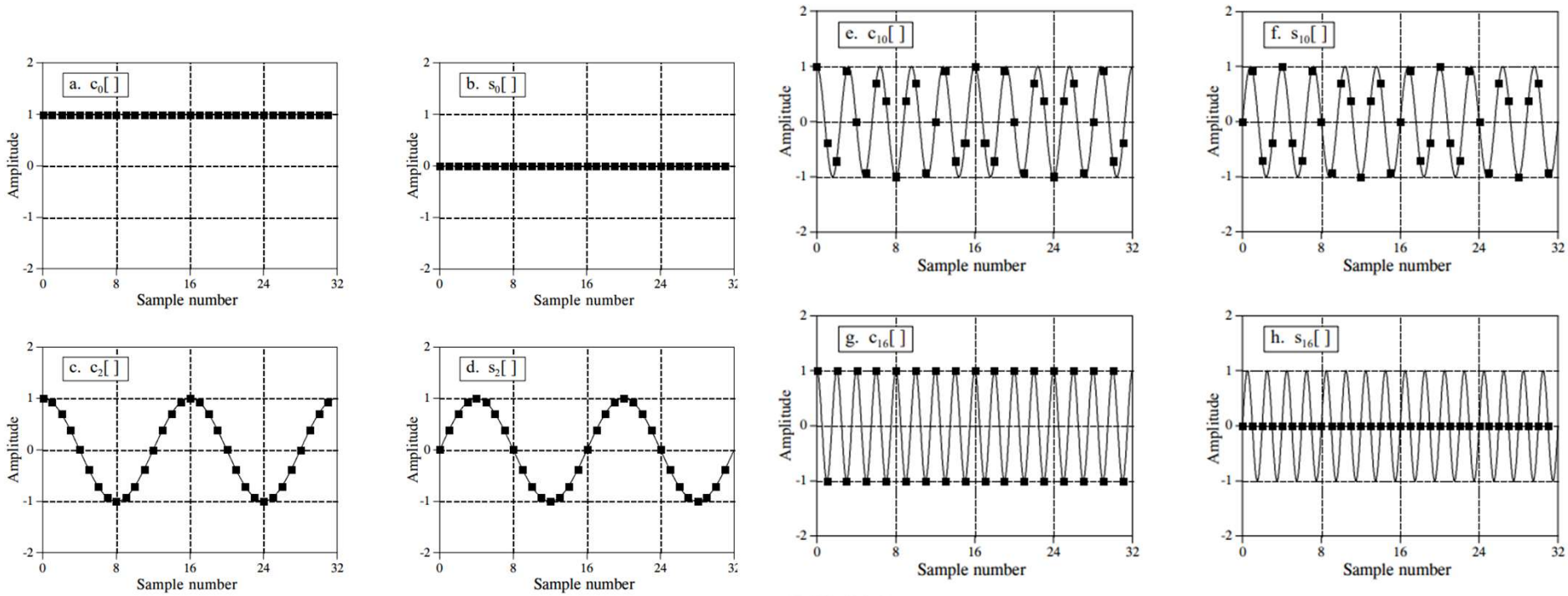
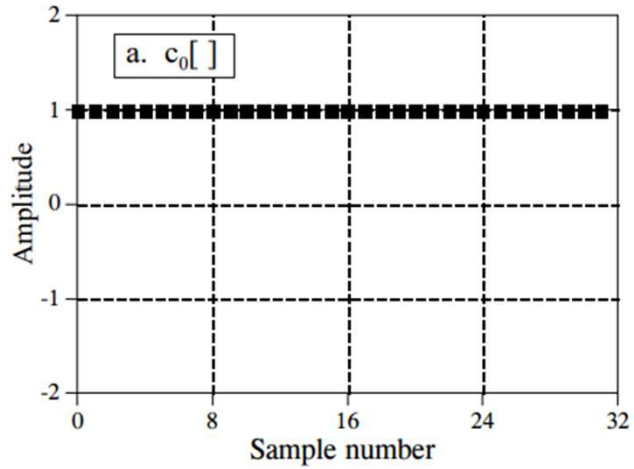


FIGURE 8-5
DFT basis functions. A 32 point DFT has 17 discrete cosine waves and 17 discrete sine waves for its basis functions. Eight of these are shown in this figure. These are discrete signals; the continuous lines are shown in these graphs only to help the reader's eye follow the waveforms.

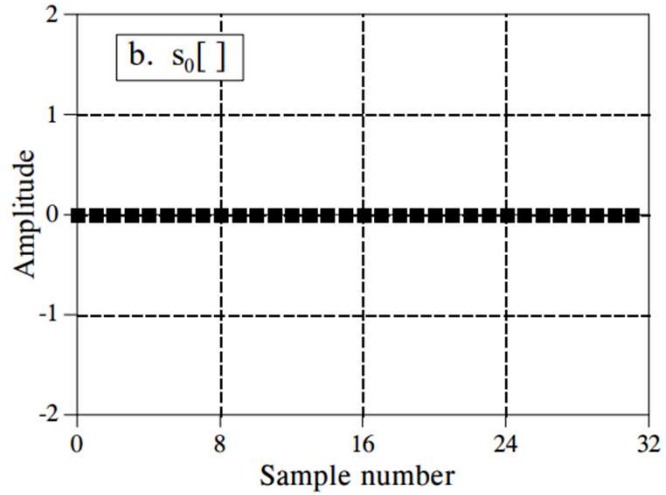
$C_0[\ ]$

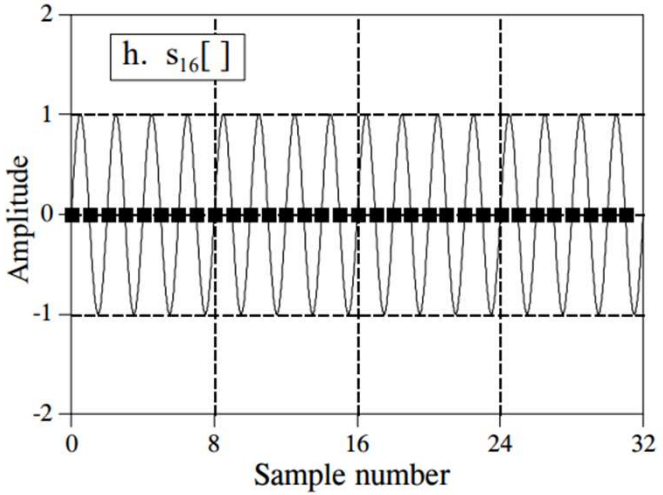
- The first cosine basis function.
- Constant value of one. So, zero frequency.
- This means, ReX[0] will hold the average value of the all the samples in the signal. This is called **DC offset.**



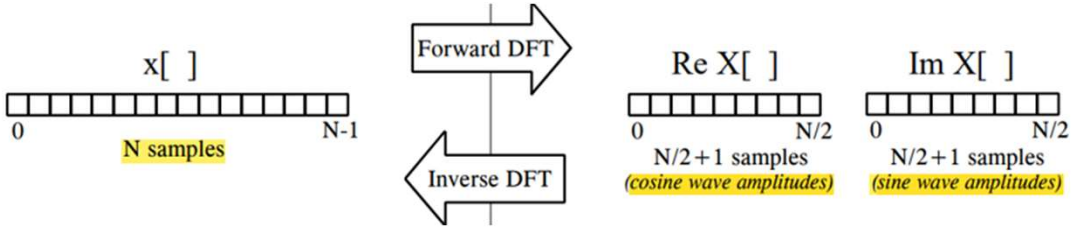
$S_0[\,]$ and $S_{16}[\,]$

- The first sine basis function.
- Constant value of zero. So, zero frequency.
- ImX[0] is irrelevant. Because, no matter what the value of ImX[0] is the contribution of $S_0[\,]$ is still the same to the original signal (no contribution).
- Same for $S_{16}[\,]$ and ImX[16]





N to N+2 | where does the extra 2 samples come from?



Here's a puzzle: If there are N samples entering the DFT, and $N+2$ samples exiting, where did the extra information come from? The answer: two of the output samples contain *no* information, allowing the other N samples to be fully independent. As you might have guessed, the points that carry no information are $Im\,X[0]$ and $Im\,X[N/2]$, the samples that always have a value of zero.

Synthesis / Inverse DFT (Freq domain to Time Domain)

- Suppose, we have ReX[] and ImX[] arrays, Now how do we get our original time domain signal back? Answer: **Synthesis**

$$x[i] = \sum_{k=0}^{N/2} \overline{ReX}[k] \cos(2\pi ki/N) + \sum_{k=0}^{N/2} \overline{ImX}[k] \sin(2\pi ki/N)$$

$$\overline{ReX}[k] = \frac{ReX[k]}{N/2}$$

$$\overline{ImX}[k] = -\frac{ImX[k]}{N/2}$$

except for two special cases:

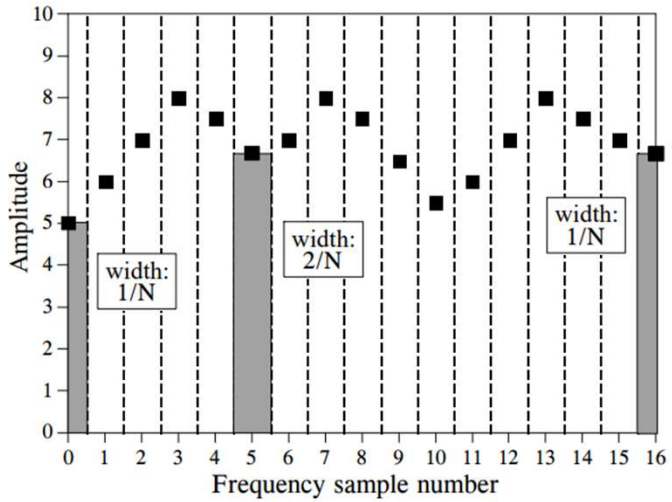
$$\overline{ReX}[0] = \frac{ReX[0]}{N}$$

$$\overline{ReX}[N/2] = \frac{ReX[N/2]}{N}$$

Why different scales for ReX[0] and ReX[N/2]

Spectral density describes how much signal (amplitude) is present *per unit of bandwidth*. To convert the sinusoidal amplitudes into a spectral density, divide each amplitude by the bandwidth represented by each amplitude. This brings up the next issue: how do we determine the bandwidth of each of the discrete frequencies in the frequency domain?

sample number 6 occurs in the band between 5.5 and 6.5, etc. Expressed as a fraction of the total bandwidth (i.e., $N/2$), the bandwidth of each sample is $2/N$. An exception to this is the samples on each end, which have one-half of this bandwidth, $1/N$.



So, why did not do it for ImX[0] and ImX[N/2]? Because, they don't matter!

Analysis / Decomposition / Calculating the DFT

Given a time domain signal, how to find out the ReX[] and ImX[]?

DFT by Simultaneous Equations

Think about the DFT calculation in the following way. You are given N values from the time domain, and asked to calculate the N values of the frequency domain (ignoring the two frequency domain values that you know must be zero). Basic algebra provides the answer: to solve for N unknowns, you must be able to write N linearly independent equations. To do this, take the first sample from each sinusoid and add them together. The sum must be equal to the first sample in the time domain signal, thus providing the first equation.

- Computationally Expensive

Analysis / Decomposition / DFT

DFT By Correlation

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

$$ImX[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi k i / N)$$

Why does it work?

Because correlation measures the similarity between two signals. By measuring similarity between a basis function and the original signal we are finding out the basis function's contribution in the original signal.

In words, each sample in the frequency domain is found by multiplying the time domain signal by the sine or cosine wave being looked for, and adding the resulting points. If someone asks you what you are doing, say with confidence: "I am correlating the input signal with each basis function." Table 8-2 shows a computer program for calculating the DFT in this way.

A Property that is required for basis functions

In order for this correlation algorithm to work, the basis functions must have an interesting property: each of them must be completely *uncorrelated* with all of the others. This means that if you multiply any two of the basis functions, the sum of the resulting points will be equal to zero. Basis functions that have this property are called **orthognal**. Many other orthognal basis functions exist, including: square waves, triangle waves, impulses, etc. Signals can be decomposed into these other orthognal basis functions using correlation, just as done here with sinusoids. This is not to suggest that this is *useful*, only that it is *possible*.

Duality of DFT

- Synthesis and analysis equations are strikingly similar.

$$x[i] = \sum_{k=0}^{N/2} Re\bar{X}[k] \cos(2\pi k i / N) + \sum_{k=0}^{N/2} Im\bar{X}[k] \sin(2\pi k i / N)$$

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

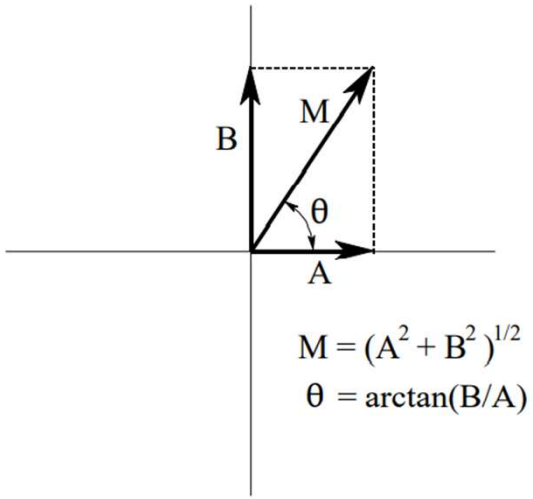
$$ImX[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi k i / N)$$

we arrived at the two procedures. In fact, the only significant difference between the two equations is a result of the time domain being *one* signal of N points, while the frequency domain is *two* signals of $N/2 + 1$ points. As discussed in later chapters, the *complex DFT* expresses both the time and the frequency domains as complex signals of N points each. This makes the two domains completely symmetrical, and the equations for moving between them virtually *identical*.

This symmetry between the time and frequency domains is called **duality**, and gives rise to many interesting properties. For example, a single point in the frequency domain corresponds to a sinusoid in the time domain. By duality, the inverse is also true, a single point in the time domain corresponds to a sinusoid in the frequency domain. As another example, convolution in the time domain corresponds to multiplication in the frequency domain. By duality, the reverse is also true: convolution in the frequency domain corresponds to multiplication in the time domain. These

Polar Notation

As it has been described so far, the frequency domain is a group of amplitudes of cosine and sine waves (with slight scaling modifications). This is called **rectangular** notation. Alternatively, the frequency domain can be expressed in **polar** form. In this notation, $ReX[\ ]$ & $ImX[\ ]$ are replaced with two other arrays, called the **Magnitude of $X[\ ]$**, written in equations as: ***Mag X*[ ]**, and the **Phase of $X[\ ]$**, written as: ***Phase X*[ ]**.



$$M = (A^2 + B^2)^{1/2}$$

$$\theta = \arctan(B/A)$$

$$MagX[k] = (ReX[k]^2 + ImX[k]^2)^{1/2}$$

$$PhaseX[k] = \arctan\left(\frac{ImX[k]}{ReX[k]}\right)$$

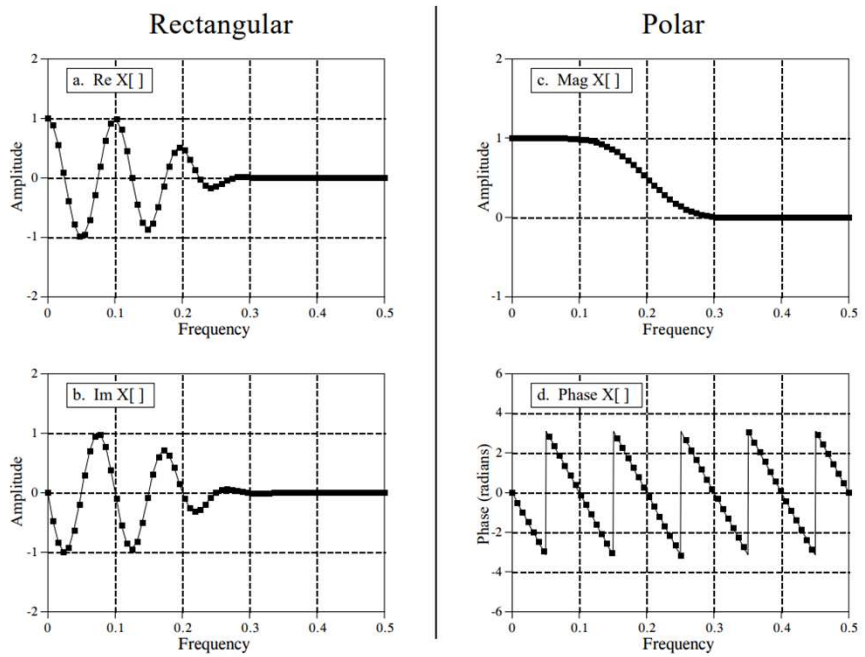
$$ReX[k] = MagX[k]\cos(PhaseX[k])$$

$$ImX[k] = MagX[k]\sin(PhaseX[k])$$

When to use rectangle and when to use polar?

When should you use rectangular notation and when should you use polar? Rectangular notation is usually the best choice for calculations, such as in equations and computer programs. In comparison, graphs are almost always in polar form. As shown by the previous example, it is nearly impossible for *humans* to understand the characteristics of a frequency domain signal by looking at the real and imaginary parts. In a typical program, the frequency domain signals are kept in rectangular notation until an observer needs to look at them, at which time a rectangular-to-polar conversion is done.

Even though the polar and rectangular representations contain exactly the same information, there are many instances where one is easier to use that the other. For example, Fig. 8-10 shows a frequency domain signal in both rectangular and polar form. Warning: Don't try to understand the shape of the real and imaginary parts; your head will explode! In comparison, the polar curves are straightforward: only frequencies below about 0.25 are present, and the phase shift is approximately proportional to the frequency. This is the frequency response of a low-pass filter.



Polar Nuisances

- There are some difficulties while working with Polar forms of signals.
- More details in the book.

Reading Assignment: Chapter 8

Thank You.