EE434
Biomedical Signal Processing
Lecture # 2

Instructor: M. Zübeyir Ünlü, PhD
zubeyirunlu@iyte.edu.tr

Digital Signal Processing

A Review - Part 1

Outline

- Fundamentals of DSP
- Sinusoidal Signals
  - The complex exponentials
- Time domain representation of signals
  - Impulse response
  - CCLDE representation
- Frequency domain representation of signals
  - Fourier transforms
  - z-transform
- Sampling Theorem
- Filtering
  - FIR filters
  - IIR filters

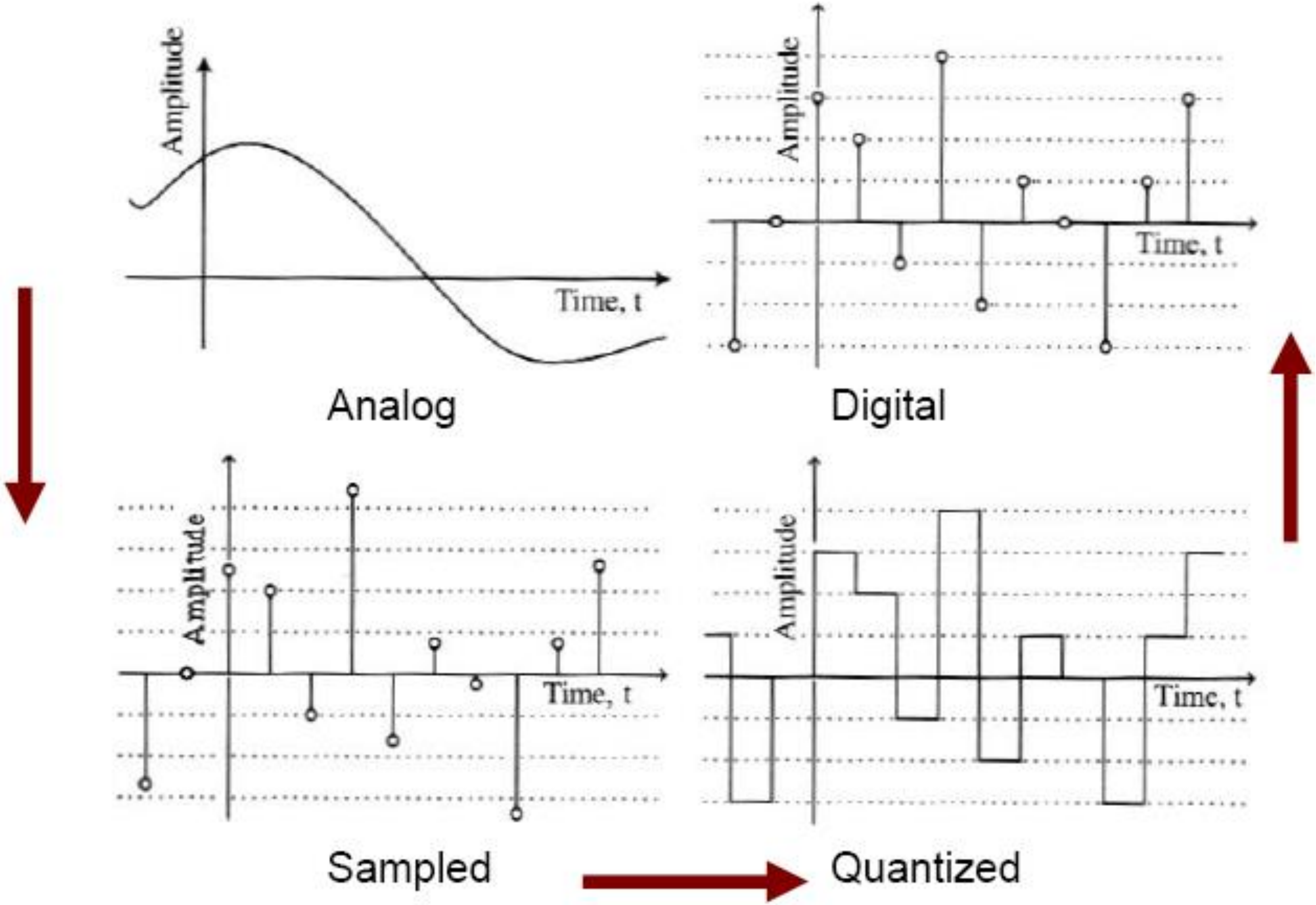
What is DSP?

Digital Signal Processing:

- Mathematical and algorithmic manipulation of **discretized** and **quantized** or **naturally digital signals** in order to extract the most relevant and pertinent information that is carried by the signal
- For the purposes of this class, the signals are of biological origin, such as ECG, EEG, respiratory signals, etc.

What is a signal? What is a system? What is processing?

Signals


Amplitude
Time, t
Analog
Amplitude
Time, t
Digital
Amplitude
Time, t
Sampled
Amplitude
Time, t
Quantized

Signals & Sinusoids

- Any physical quantity that is represented as a function of an independent variable is called a **signal**
  - Independent variable can be ***time***, ***frequency***, ***space***, etc.
- **Sinusoids** play a very important role in signal processing, because
  - They are easy to generate
  - They are easy to work with – their mathematical properties are well known
  - Most importantly: All signals can be represented as a sum of sinusoids
    - **Fourier transforms**

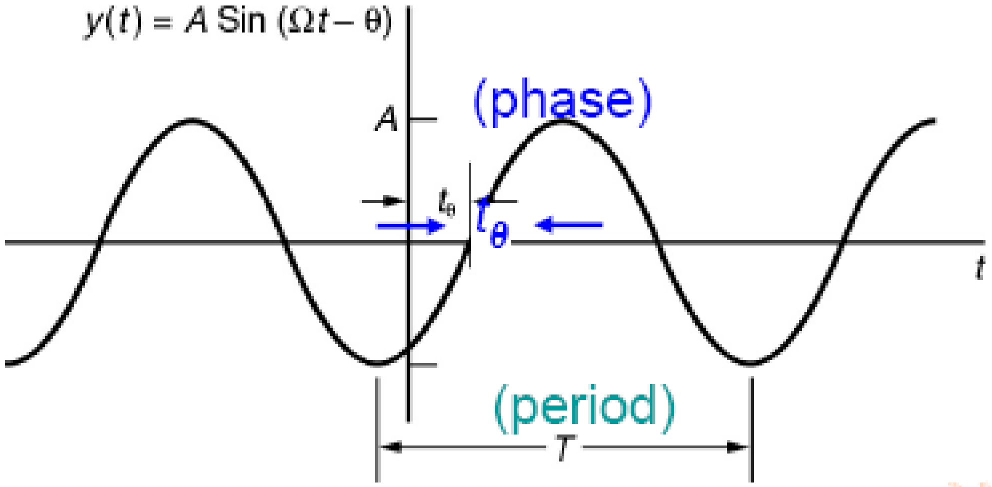
Sinusoids

- In continuous time a **sinusoid** is

$$y(t) = A \sin (\Omega t - \theta)$$

where A is amplitude, Ω is angular frequency (radians/sec.), and θ is phase (radians).

- A continuous time domain sinusoid is a periodic signal
- **Period**: The time after which the signal repeats itself: $y(t) = y(t+T)$



Sinusoids

- **Frequency**: Inverse of the period
- **Angular frequency**: A different measure of rate of change in the signal, easier to use with sinusoidal signals, represented in radians/second and shown by Ω.
  - **Analog frequency** (f – measured in Hertz, 1/sec), the period T (measured in seconds), and the angular frequency Ω are related to each other by

$$f = \frac{\Omega}{2\pi} \quad \text{or} \quad \Omega = 2\pi f \qquad T = \frac{1}{f} \quad \text{or} \quad \Omega = \frac{2\pi}{T}$$

- **Phase**: The number of degrees –in radians – the sinusoid is shifted from its origin
- If the sinusoid is shifted by t_θ seconds, then the phase is

$$\theta = 2\pi f t_\theta = 2\pi \frac{t_\theta}{T}$$

Discrete Time Signals

- A **discrete-time signal**, commonly referred to as a **sequence**, is only defined at discrete time instances, where t is defined to take integer values only

- **Discrete-time signals** may also be written as a sequence of numbers inside braces:

$$\{x[n]\} = \{\ldots, -0.2, \mathbf{2.2}, 1.1, 0.2, -3.7, 2.9, \ldots\}$$

  - n indicates **discrete time**, in integer intervals, the bold-face denotes time $n = 0$

Discrete Time Signals

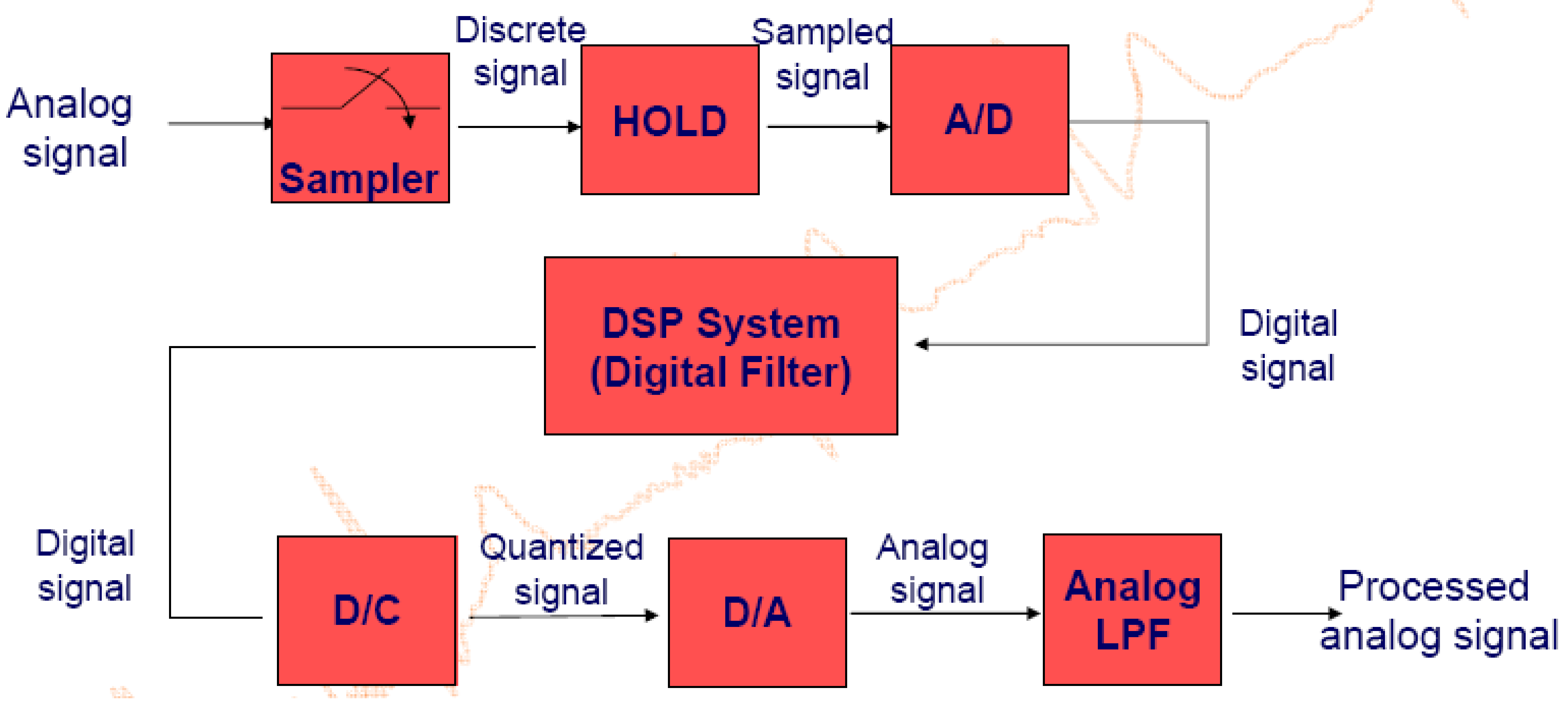
- **Discrete time signals** are often generated from **continuous time signals** by **sampling** which can roughly be interpreted as quantizing the independent variable (time)

$$\{x(n)\} = x(nT_s) = x(t)\big|_{t=nT_s} \qquad n = \ldots -2, -1, 0, 1, 2, \ldots$$

where T_s : Sampling interval or period

$f_s = 1/T_s$: Sampling frequency

Components of DSP System



Filters

By far the most commonly used DSP operation

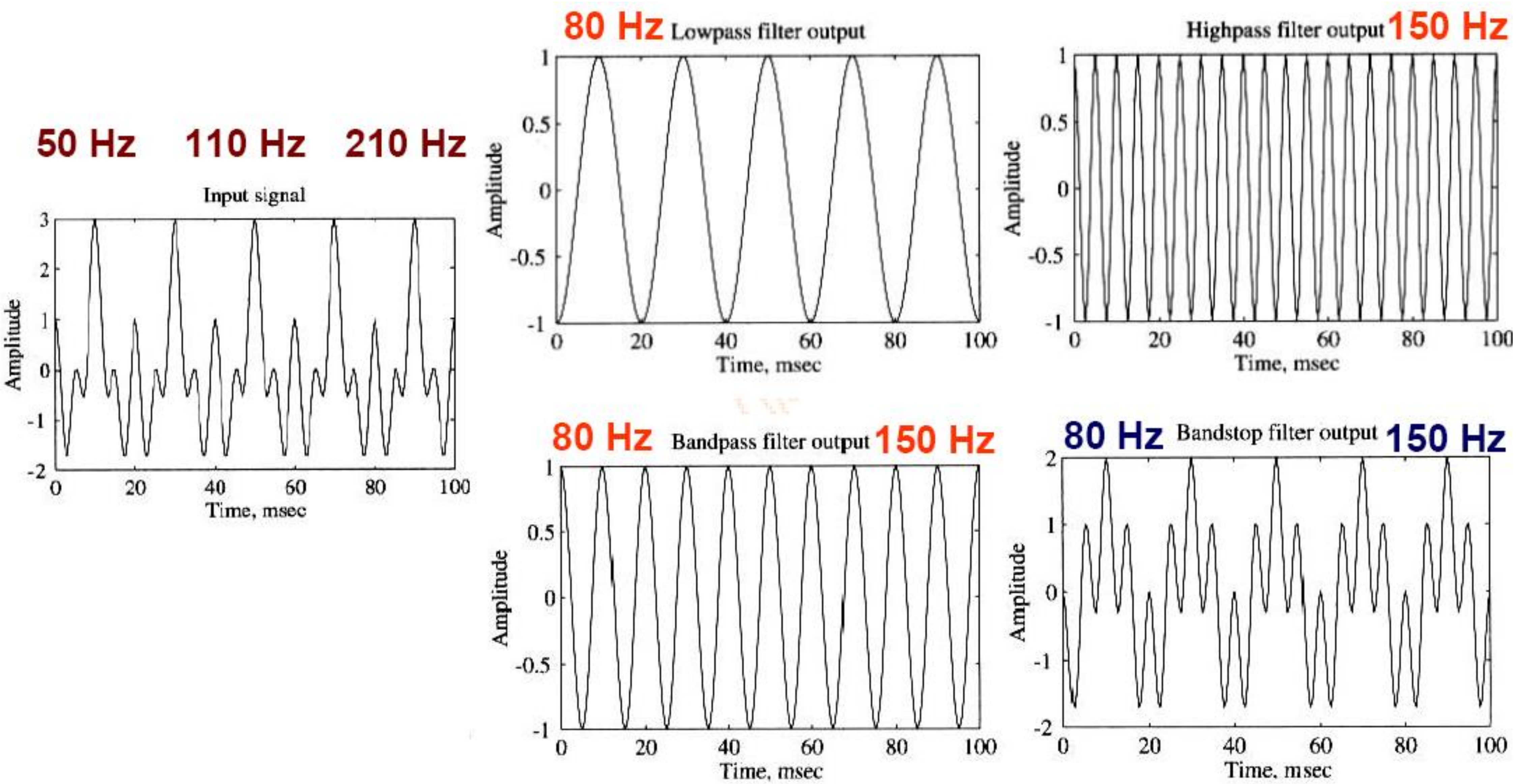
- Filtering refers to deliberately changing the frequency content of the signal, typically, by removing certain frequencies from the signals
- For denoising applications, the (frequency) filter removes those frequencies in the signal that correspond to noise
- In communications applications, filtering is used to focus to that part of the spectrum that is of interest, that is, the part that carries the information

Filters

Typically we have the following types of filters

- **Lowpass (LPF)** – removes high frequencies, and retains (passes) low frequencies
- **Highpass (HPF)** – removes low frequencies, and retains high frequencies
- **Bandpass (BPF)** – retains an interval of frequencies within a band, removes others
- **Bandstop(BSF)** – removes an interval of frequencies within a band, retains others
- **Notch filter** – removes a specific frequency

Filters


50 Hz
110 Hz
210 Hz
Input signal
Amplitude
Time, msec
80 Hz Lowpass filter output
Highpass filter output 150 Hz
80 Hz Bandpass filter output 150 Hz
80 Hz Bandstop filter output 150 Hz
Amplitude
Time, msec

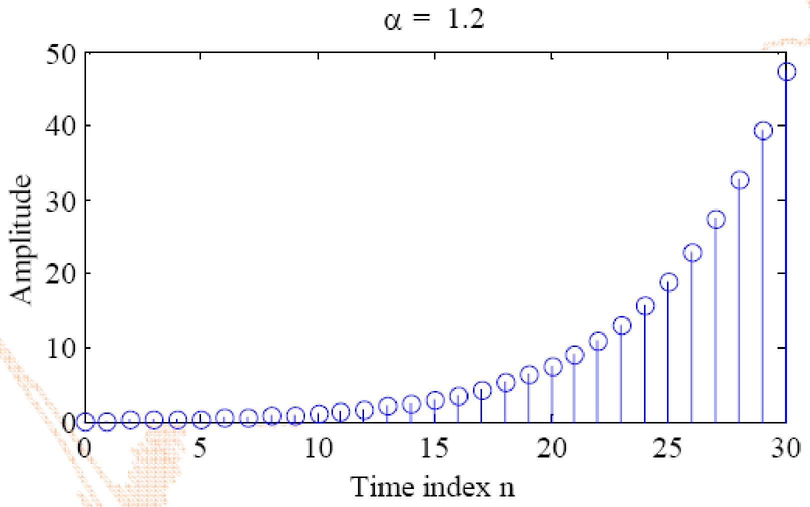
Exponential Sequence

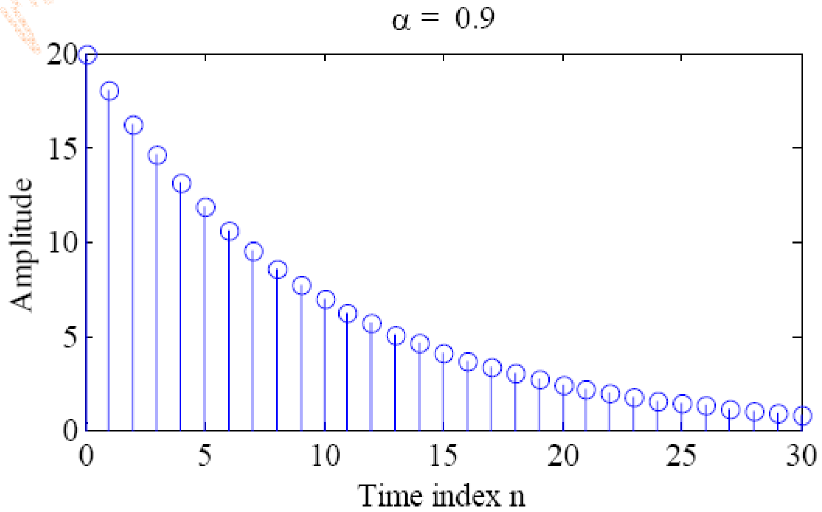
- A special case of the **exponential signal** is very commonly used in DSP. A: real constant, and $\alpha = e^{j\omega_o}$ is purely imaginary, i.e.,

$$x[n] = A\, e^{j\omega_o n} = A(\cos[\omega_o n] + j \sin[\omega_o n])$$

- If both A and α are purely real, then we have a **real exponential sequence**

$$x[n] = A\,\alpha^n, \quad -\infty < n < \infty,\ A,\ \alpha \in \Re$$





Bizarre Properties of Discrete Signals

- **Property 1**: Consider $x[n] = e^{j\omega_1 n}$ and $y[n] = e^{j\omega_2 n}$ with $0 < \omega_1 < \pi$ and $2\pi k < \omega_2 < 2\pi(k+1)$, where k is any positive integer

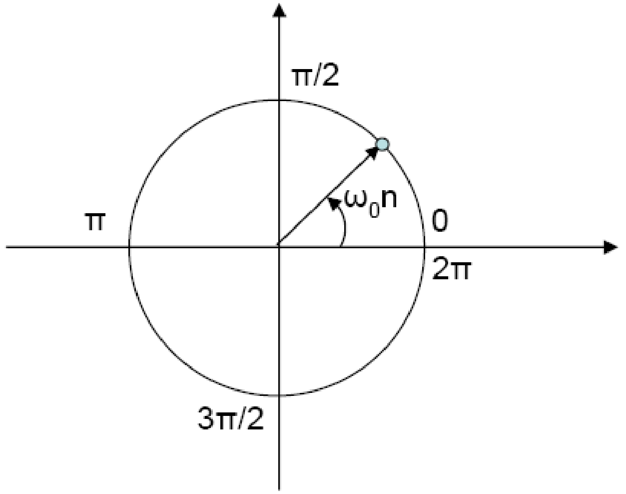
  If $\omega_2 = \omega_1 + 2\pi k$, then $x[n] = y[n]$

  Thus, $x[n]$ and $y[n]$ are **indistinguishable**

- What does this mean?

  Two periodic discrete **exponential sequences** are **indistinguishable**, if their angular frequencies are $2\pi k$ apart from each other !!!

Bizarre Properties of Discrete Signals



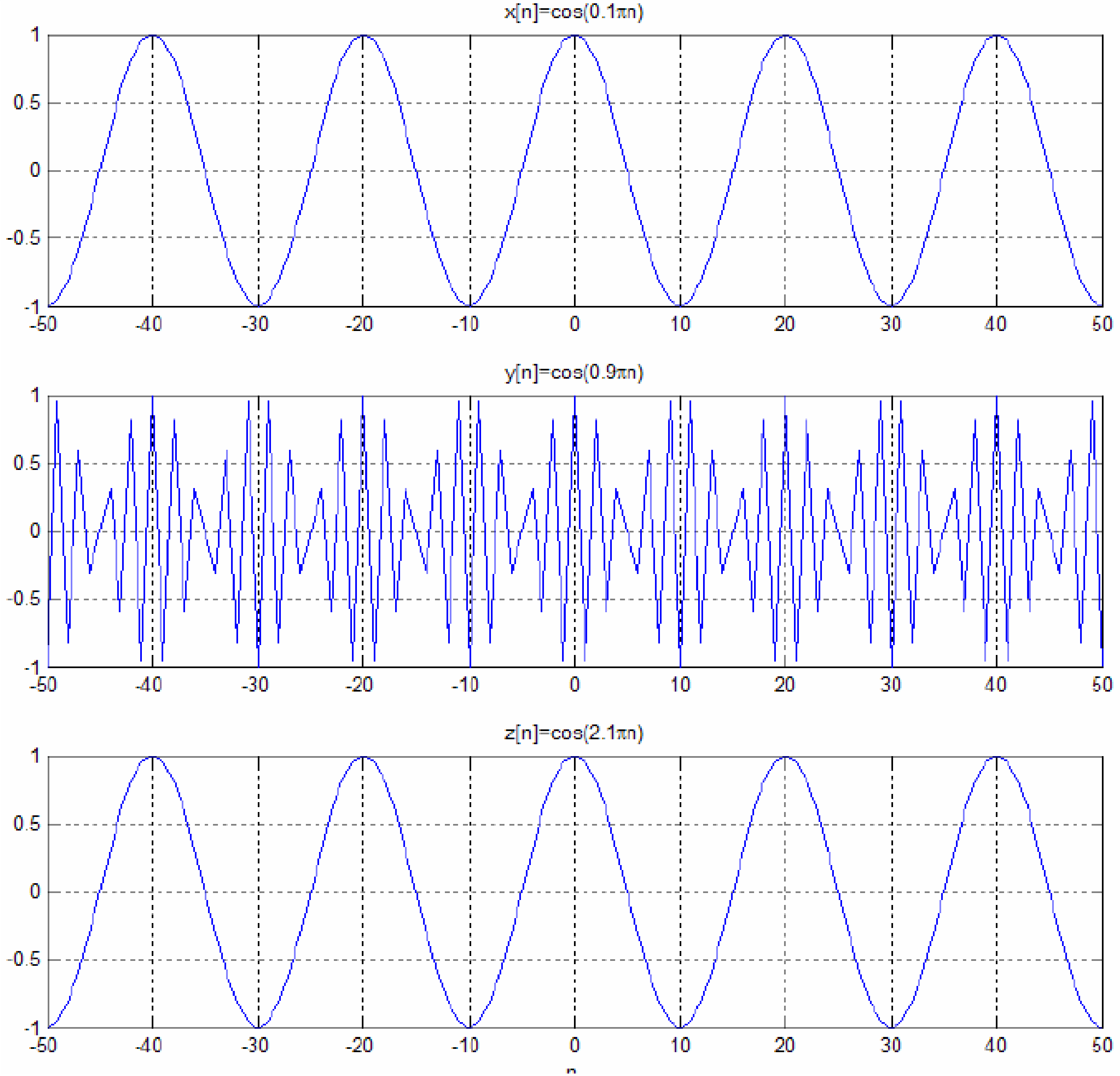
- **Property 2**: Digital angular frequency:
  - The frequency of oscillation of $A\cos(\omega_o n)$;

    increases as ω_o increases from 0 to π, and then

    decreases as ω_o increases from π to 2π
- Thus, frequencies in the neighborhood of ω = 0 or $2\pi k$ are called **low frequencies**, whereas, frequencies in the neighborhood of ω = π or $\pi(2k+1)$ are called **high frequencies**
  - Note that, the frequencies around ω = 0 and ω = 2π are both low frequencies. In fact, ω = 0 and ω = 2π are both identical frequencies
  - Due to these two properties a frequency in the neighborhood of ω = 2π is indistinguishable from a frequency in the neighborhood of ω = $2\pi \pm 2\pi k$

Try This!

```
n=-50:50;
x=cos(pi*0.1*n);
y=cos(pi*0.9*n);
z=cos(pi*2.1*n);
subplot(311)
plot(n,x)
title('x[n]=cos(0.1\pin')
title('x[n]=cos(0.1\pin)')
grid
subplot(312)
plot(n,y)
title('y[n]=cos(0.9\pin)')
grid
subplot(313)
plot(n,z)
grid
title('z[n]=cos(2.1\pin)')
xlabel('n')
```



Discrete Convolution

- The operation by far the most commonly used
- At the heart of any DSP system:

$$x[n] \longrightarrow \boxed{\begin{array}{c}\text{Discrete-Time System}\\ h[n]\end{array}} \longrightarrow y[n]$$

Input sequence — Output sequence

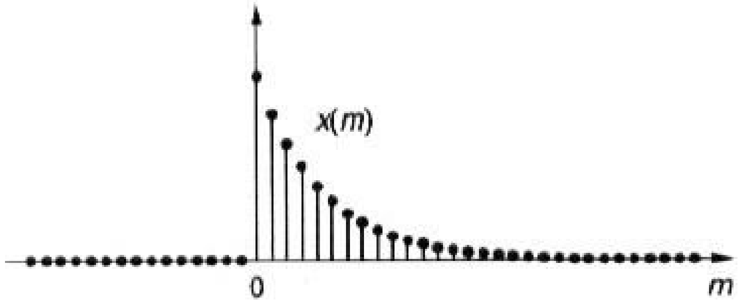
$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m] = \sum_{m=-\infty}^{\infty} h[m]x[n-m]$$

where $h[n]$: **Impulse response of the system**

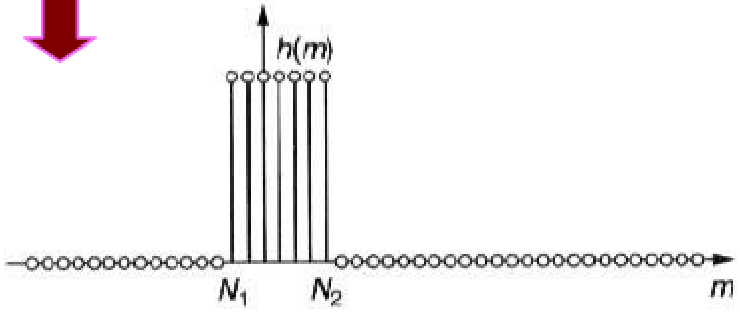
Convolution Example

$$x[n] = a^n u[n]$$

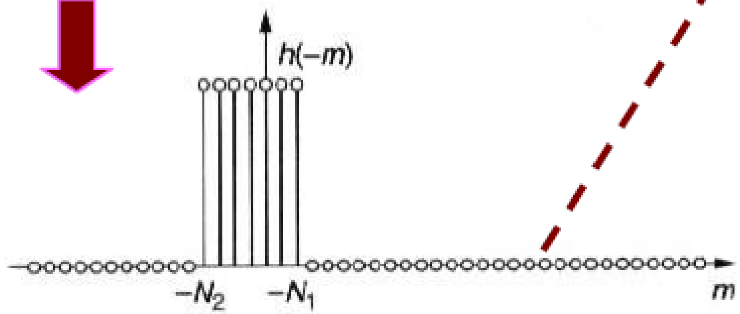
$$h[n] = \begin{cases} 1 & N_1 \le n \le N_2 \\ 0 & otherwise \end{cases}$$



(a)

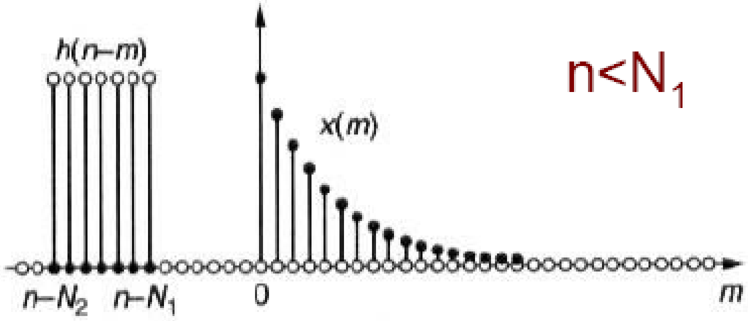


(b)



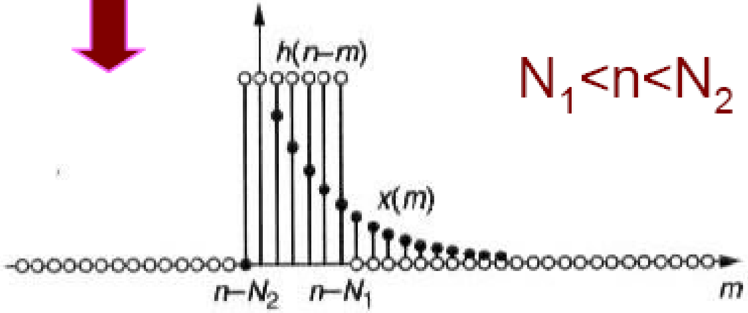
(c)

$$y[n] = \begin{cases} 0 & n < N_1 \\ \dfrac{1-a^{n-N_1+1}}{1-a} & N_1 \le n < N_2 \\ a^{n-N_2}\dfrac{1-a^{N_2-N_1+1}}{1-a} & n \ge N_2 \end{cases}$$



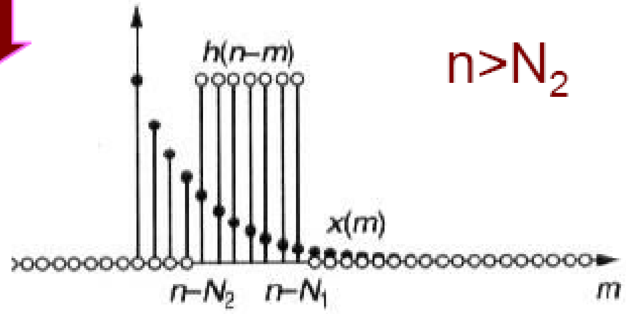
(d)

$n<N_1$



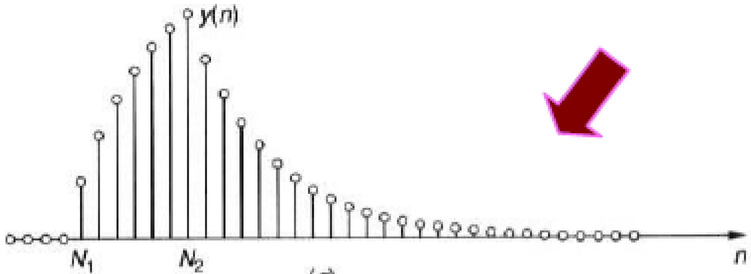
(e)

$N_1<n<N_2$



(f)

$n>N_2$

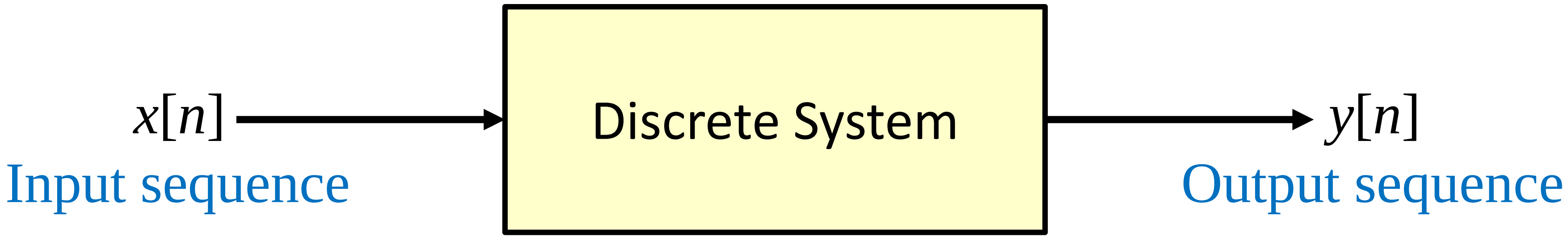


(g)

Characterization of Discrete Systems

Discrete systems can be characterized in several ways:

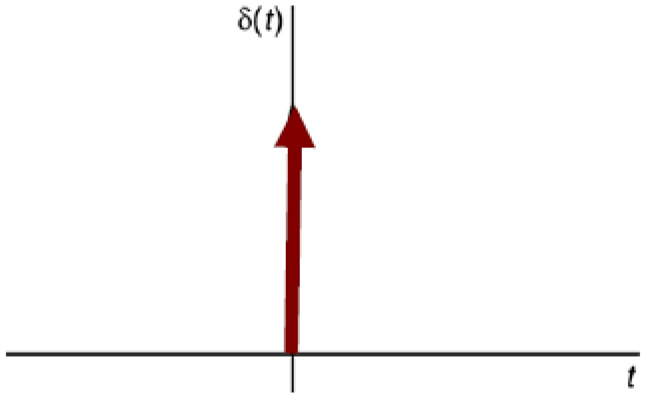
- **Impulse response** (in time)
- **Linear Constant Coefficient Difference Equations** (in time)
- **Frequency response** (in frequency)
- **Transfer function** (in frequency)

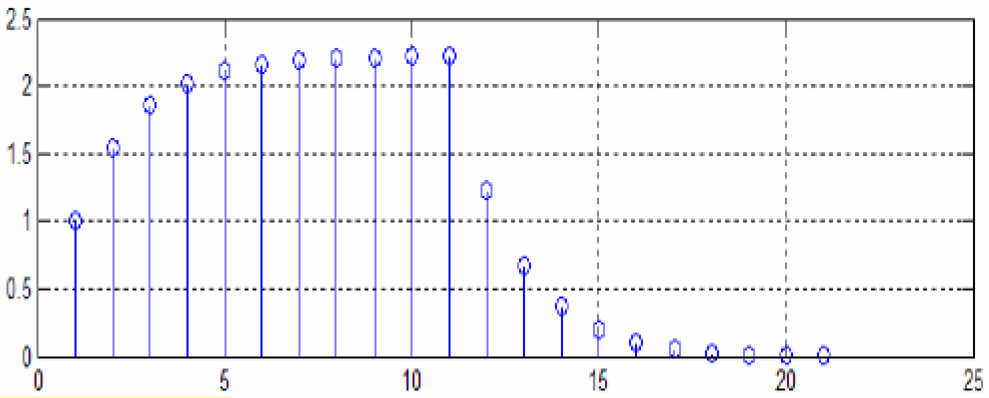


Impulse Response, $h[n]$

The **response** of a discrete system to a **unit impulse sequence** $\delta[n]$ is called the **impulse response of the system**, and it is typically denoted by $h[n]$

$x[n] = \delta[n]$ → Discrete System → $y[n] = h[n]$

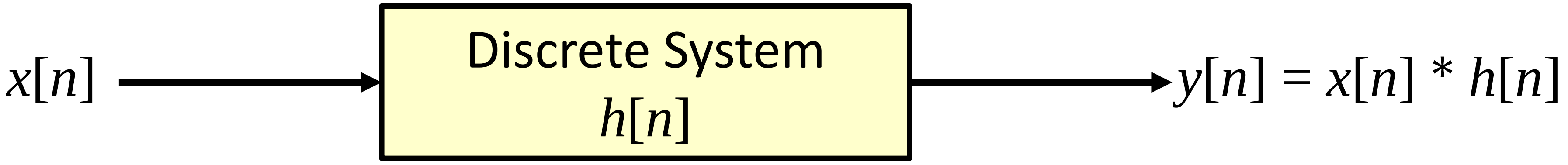




$$h[n] = x[n] * \delta[n] = \sum_{m=-\infty}^{\infty} x[m]\delta[n-m]$$

Impulse Response, $h[n]$

- So, what is the big deal?
- The **impulse response** plays a monumental role in characterization of LTI systems
  - In fact, if you know the **impulse response** of a discrete LTI system, then you know the **response of the system** to **any arbitrary input**!
  - You tell me $h[n]$, I will tell you the response to **any** $x[n]$



$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m]$$

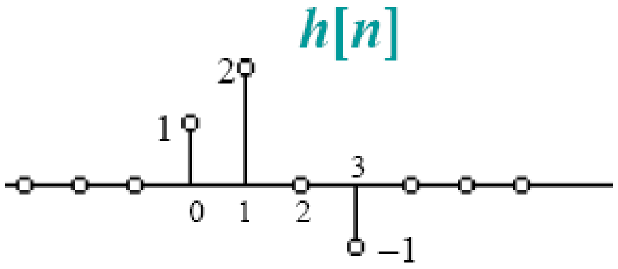
Finite Impulse Response Systems

- If the impulse response $h[n]$ of a system is of **finite length**, that system is referred to as a **finite impulse response (FIR) system**

$$h[n] = 0 \text{ for } n < N_1 \text{ and } n > N_2,\ N_1 < N_2$$

- The output of such a system can then be computed as a finite convolution sum

$$y[n] = \sum_{k=N_1}^{N_2} h[k]x[n-k]$$



- E.g., $h[n]$ =[**1** 2 0 -1] is a **FIR system** (**filter**)

- **FIR systems** are also called **nonrecursive systems** (for reasons that will later become obvious), where the output can be computed from the current and past input values only – without requiring the values of previous outputs

Infinite Impulse Response Systems

- If the impulse response is of infinite length, then the system is referred to as an **infinite impulse response (IIR) system**. These systems cannot be characterized by the convolution sum due to infinite sum

  - Instead, they are typically characterized by **Linear Constant Coefficient Difference Equations**, as we will see later

Infinite Impulse Response Systems

- Recall accumulator and note that it can have an alternate – and more compact - representation that makes the current output a function of previous inputs and outputs

$$y[n]=\sum_{l=-\infty}^{\infty} x[l] \quad \Rightarrow \quad y[n]=y[n-1]+x[n]$$

- The **impulse response of this system** (which is of infinite length), cannot be represented with a finite convolution sum. Note that, since the current output depends on the previous outputs, this is also called a **recursive system**

- All discrete systems can also be represented using **Linear Constant Coefficient Difference Equations** of the form

$$y[n]+a_1y[n-1]+a_2y[n-2]+\ldots+a_Ny[n-N] = b_0x[n]+b_1x[n-1]+\ldots+b_Mx[n-M]$$

Outputs $y[n]$ Inputs $x[n]$

$$\sum_{i=0}^{N} a_i y[n-i] = \sum_{j=0}^{M} b_j x[n-j]$$

Constant coefficients

- Constant coefficients a_i and b_i are called filter coefficients
- Integers M and N represent the maximum delay in the input and output, respectively. The larger of the two numbers is known as the order of the filter
- Any LTI system can be represented as two finite sum of products!

FIR Filter

Note that the expression indicates the most general form of an LTI system:

$$\sum_{i=0}^{N} a_i y[n-i] = \sum_{j=0}^{M} b_j x[n-j], \quad a_0 = 1$$

- If the current output $y[n]$ does not depend on previous outputs $y[n\text{-}i]$, that is if all a_i=0 (except a_0=1), then we have **no recursion** – such systems are **FIR** (**non-recursive**) **systems**

$$y[n] = \sum_{j=0}^{M} b_j x[n-j]$$

FIR Filter

- Note that the **impulse response** of an **FIR system** can easily be obtained from its **LCCDE** representation:

$$y[n]=\sum_{j=0}^{M} b_j x[n-j] \Rightarrow h[n]=\sum_{j=0}^{M} b_j \delta[n-j]=b_0\delta[n]+b_1\delta[n-1]+\ldots+b_M\delta[n-M]$$

- The sum of finite numbers will always be finite, therefore, the impulse response of this system will be **finite**, hence, **finite impulse response (FIR)**
- Finite Impulse Response $\leftrightarrow$ Nonrecursive

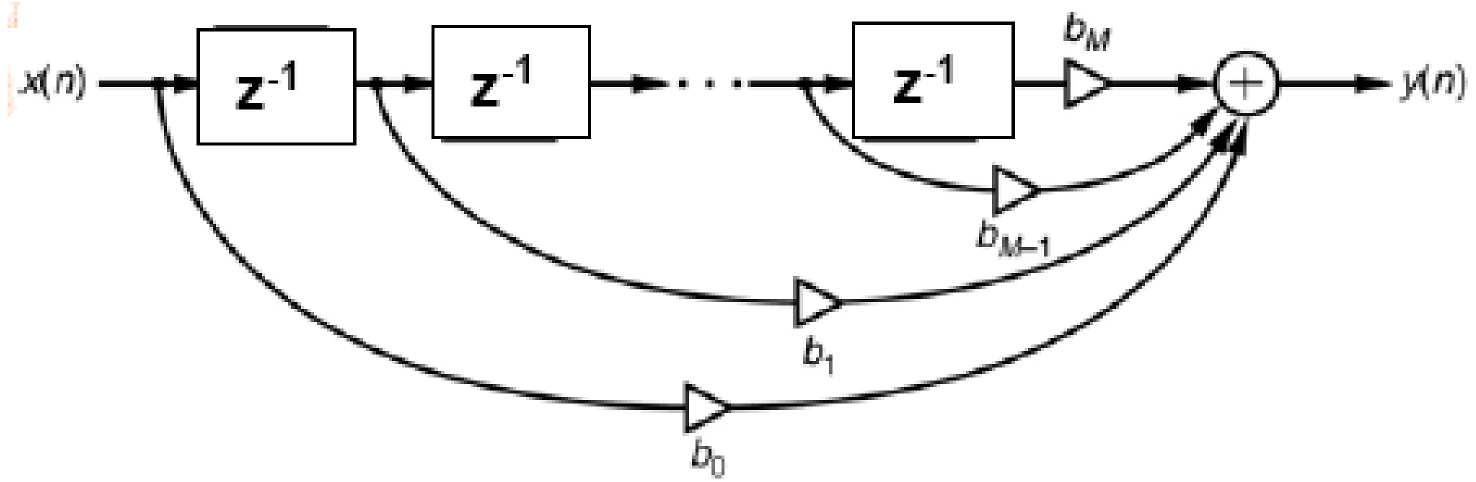
FIR Systems

$$y[n]=\sum_{j=0}^{M} b_j x[n-j] = b_0 x[n] + b_1 x[n-1] + \ldots + b_M x[n-M]$$

- Note that this representation looks similar to the definition of convolution
- In fact, $y[n] = b_n * x[n]$, that is the system output of an FIR filter is simply the convolution of input $x[n]$ with the filter coefficients b_n
- Since we already know that the output of a system is the convolution of its input with the system impulse response, it follows that filter coefficients b_n is the impulse response of an FIR filter!

FIR Systems

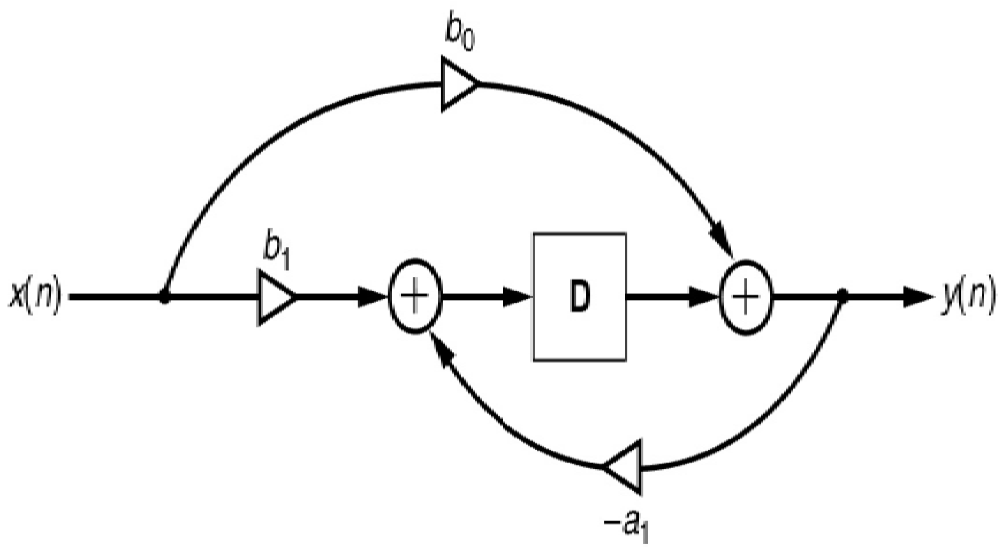
- The LCCDE representation of an FIR system can schematically be represented using the following diagram, known as the "filter structure"



- The hardware implementation follows this structure exactly, using delay elements, adders and multipliers

IIR Systems

- If in the general expression, a_i are not zero, then the output depends on former outputs, and hence this is a recursive system
- The impulse response of an IIR system cannot be represented as a closed finite convolution sum precisely due to recursion
- The filter structure of IIR systems – which has a distinct feedback (recursion) loop, has the following form:



$$y[n]=\ldots$$

LCCDEs

- Note that, assuming that system is **causal**, $y[n]$ can be pulled out of the CCLDE equation to obtain:

$$y[n] = -\sum_{i=1}^{N} \frac{a_i}{a_0} y[n-i] + \sum_{j=1}^{M} \frac{b_j}{a_0} x[n-j]$$

- Since the impulse response of an FIR system consists of finite terms, it is always **stable** – a significant advantage of FIR systems

- IIR systems are not guaranteed to be stable, since their $h[n]$ consists of infinite number of terms. Their design requires **stability checks**!

The Frequency Domain

- Time domain operation are often not very informative and/or efficient in signal processing
- An alternative representation and characterization of signals and systems can be made in transform / frequency domain
  - Much more can be said, much more information can be extracted from a signal in the transform / frequency domain
  - Many operations that are complicated in time domain become rather simple algebraic expressions in transform domain
  - Most signal processing algorithms and operations become more intuitive in frequency domain, once the basic concepts of the frequency domain are understood

Fourier Transforms

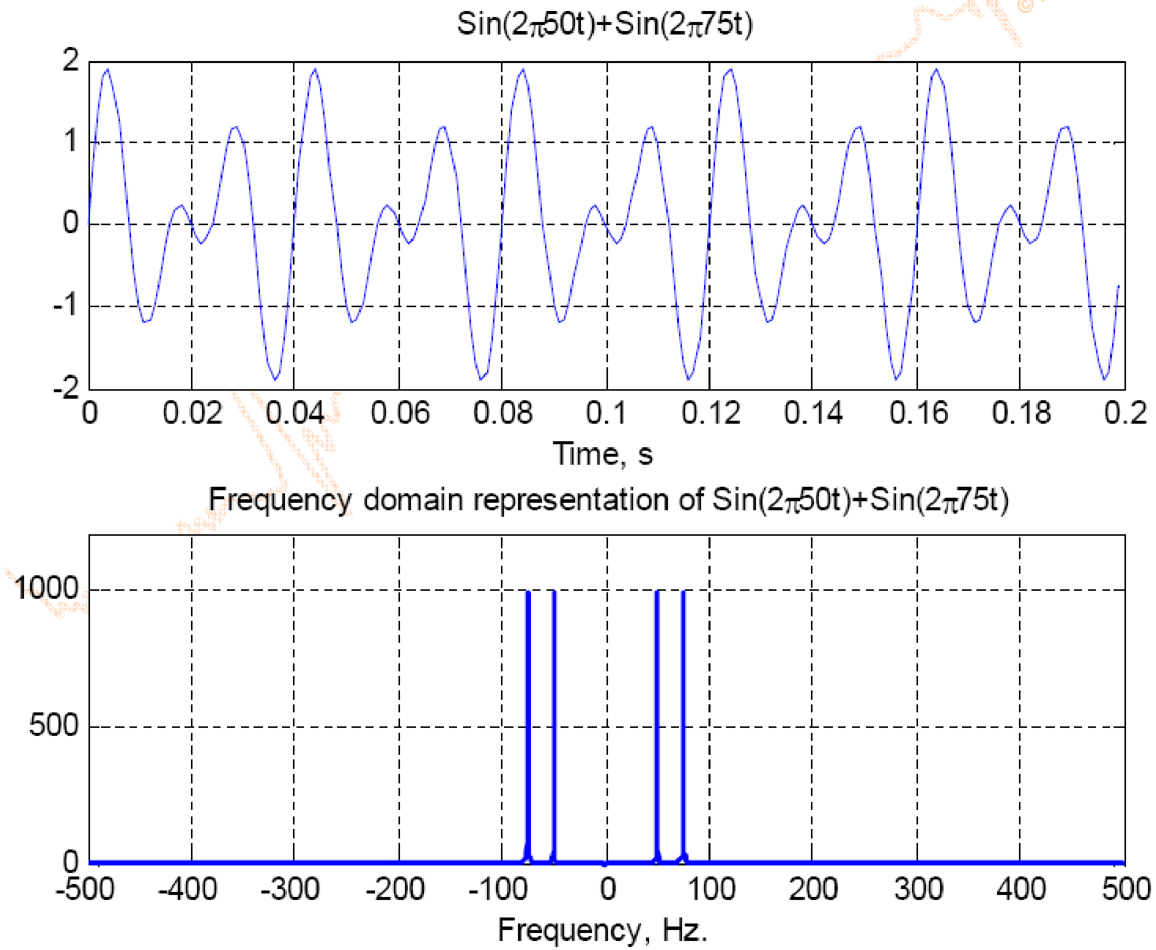
- Frequency representation of a signal is typically obtained in one of **Fourier transforms** or ***z*-transform**:
- **Fourier Transforms**:
  - **Fourier series** – for periodic continuous time signals
  - **Continuous Time Fourier Transform (CTFT)** – for aperiodic continuous time signals $x(t) \overset{\Im}{\leftrightarrow} X(\Omega)$
  - **Discrete Time Fourier Transform (DTFT)** – for aperiodic discrete time signals (frequency domain is still continuous however) $x[n] \overset{\Im}{\leftrightarrow} X(\omega)$
  - **Discrete Fourier Transform (DFT)** – DTFT sampled in the frequency domain $x[n] \overset{\Im}{\leftrightarrow} X[k]$
  - **Fast Fourier Transform (FFT)** – Same as DFT, except calculated very efficiently

- ***z*-transform**:
    - A generalized version of the DTFT. The de-facto transform used in representing discrete signals and systems in frequency domain. Also used in designing filters

$$x[n] \overset{\mathfrak{I}}{\leftrightarrow} X(z)$$

Frequency Domain (Example)

```
t=-1:0.001:1;
x=sin(2*pi*50*t)+sin(2*pi*75*t);
subplot(211)
plot(t(1001:1200),x(1:200))
grid
title('Sin(2\pi50t)+Sin(2\pi75t)')
xlabel('Time, s')
subplot(212)
X=abs(fft(x));
X2=fftshift(X);
f=-499.9:1000/2001:500;
plot(f,X2);
grid
title(' Frequency domain representation of Sin(2\pi50t)+Sin(2\pi75t)') xlabel('Frequency, Hz.')
```



Key Facts to Remember

- All FT pairs provide a transformation between time and frequency domains: The frequency domain representation provides how much of which frequencies exist in the signal → More specifically, how much $e^{j\Omega t}$ exists in the signal for each Ω
- In general, the frequency representation is complex (except when the signal is even)
  - $|X(\Omega)|$: **The magnitude spectrum** → the power of each Ω component
  - $\text{Ang}\ X(\Omega)$: **The phase spectrum** → the amount of phase delay for each Ω component

Key Facts to Remember

- The **FS is discrete in frequency domain**, since it is the discrete set of exponentials – integer multiples of Ω_0 – that make up the signal. This is because only a finite number of frequencies are required to construct a periodic signal

- The **FT is continuous in frequency domain**, since exponentials of a continuum of frequencies are required to reconstruct a non-periodic signal

- Both transforms are **non-periodic in frequency domain**

$$X(\Omega) = \Im\left\{x(t)\right\} = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t}dt$$

$$x(t) \overset{\Im}{\Leftrightarrow} X(\Omega)$$

$$x(t) = \Im^{-1}\left\{X(\Omega)\right\} = \int_{-\infty}^{\infty} X(\Omega)e^{j\Omega t}d\Omega$$

Discrete-Time Fourier Transform (DTFT)

- Similar to continuous time signals, discrete time sequences can also be periodic or non-periodic, resulting in **discrete-time Fourier series** or **discrete–time Fourier transform**, respectively
- Most signals in engineering applications are non-periodic, so we will concentrate on **DTFT**
- We will represent the discrete frequency as ω, measured in radians/sample

$$X(\omega) = \Im\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

$$x[n] \overset{\Im}{\Leftrightarrow} X(\omega)$$

$$x[n] = \Im^{-1}\{X(\omega)\} = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\omega) e^{j\omega n} d\omega$$

Quick facts:

- Since $x[n]$ is discrete, we can only add them, hence summation
- The sum of $x[n]$, weighted with continuous exponentials, is continuous → The DTFT $X(\omega)$ is continuous (non-discrete)
- Since $X(\omega)$ is continuous, $x[n]$ is obtained as a continuous integral of $X(\omega)$, weighted by the same complex exponentials
- $x[n]$ is obtained as an integral of $X(\omega)$, where the integral is over an interval of 2π. → This is our first clue that **DTFT is periodic with 2π in frequency domain**
- $X(\omega)$ is sometimes denoted as $X(e^{j\omega})$ in some books. While $X(e^{j\omega})$ is more accurate, we will use $X(\omega)$ for brevity

Frequency Response

- This theorem constitutes the fundamental cornerstone for the concept of **frequency response**. $H(\omega)$, the DTFT of $h[n]$, is called the **frequency response of the system**

- Why is it important?

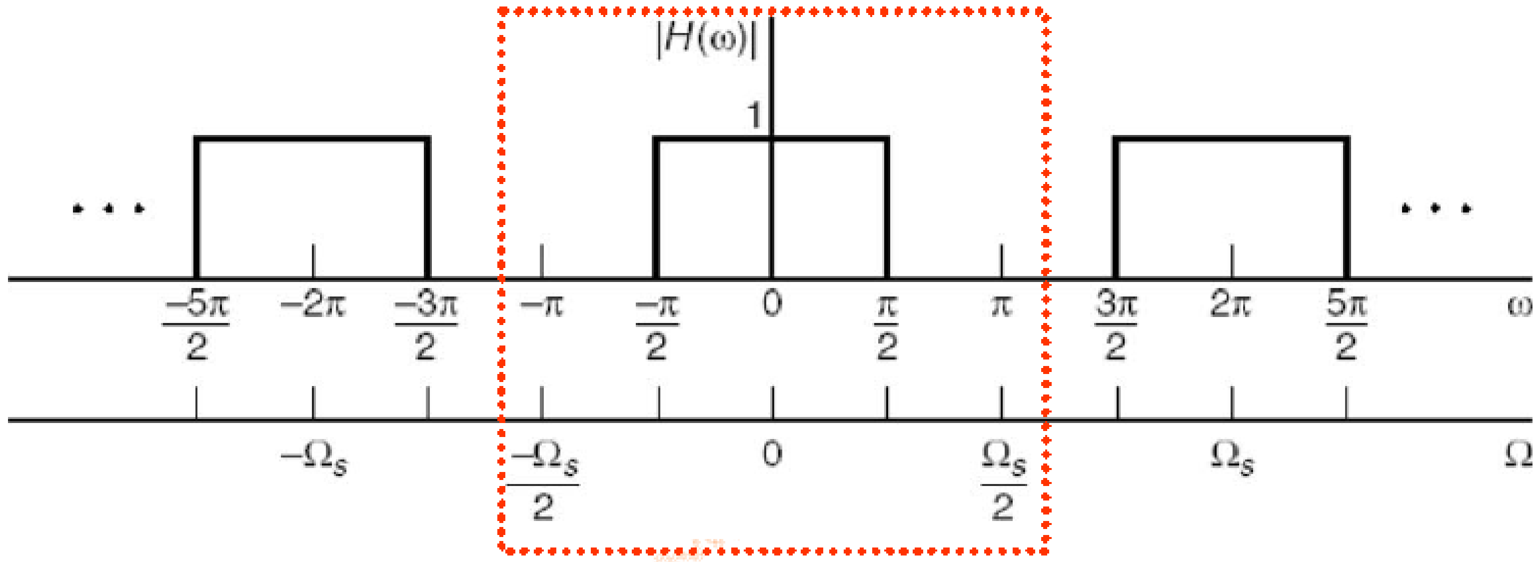
- If a sinusoidal sequence with frequency ω_0 is applied to a system whose **frequency response** is $H(\omega)$, then the output can be obtained simply by evaluating $H(\omega)$ at $\omega = \omega_0$

- Since all signals can be written as a superposition of sinusoids at different frequencies, then the output to an arbitrary input can be obtained as the superposition of $H(\omega_0)$ for each component that makes up the input signal!

Implications of The Periodicity Property

- **IMPORTANT**: The discrete frequency 2π rad of the discrete-time sequence $x[n]$, corresponds to the sampling frequency Ω_S used to sample the original continuous signal $x(t)$ to obtain $x[n]$
- **Proof**: $\omega = \Omega T_S \rightarrow$ For $\Omega = \Omega_S$, we have $\omega = \Omega_S T_S = 2\pi f_S T_S = 2\pi$

$$\boxed{H(\omega) = H(\omega + 2\pi)}$$



Convolution

- **Convolution** in time domain is equivalent to **multiplication** in frequency domain

$$x[n] * h[n] \overset{\Im}{\Leftrightarrow} X(\omega) \cdot H(\omega)$$

  - This is one of the fundamental theorems in filtering. It allows us to compute the **filter response** in frequency domain using the **frequency response** of the filter

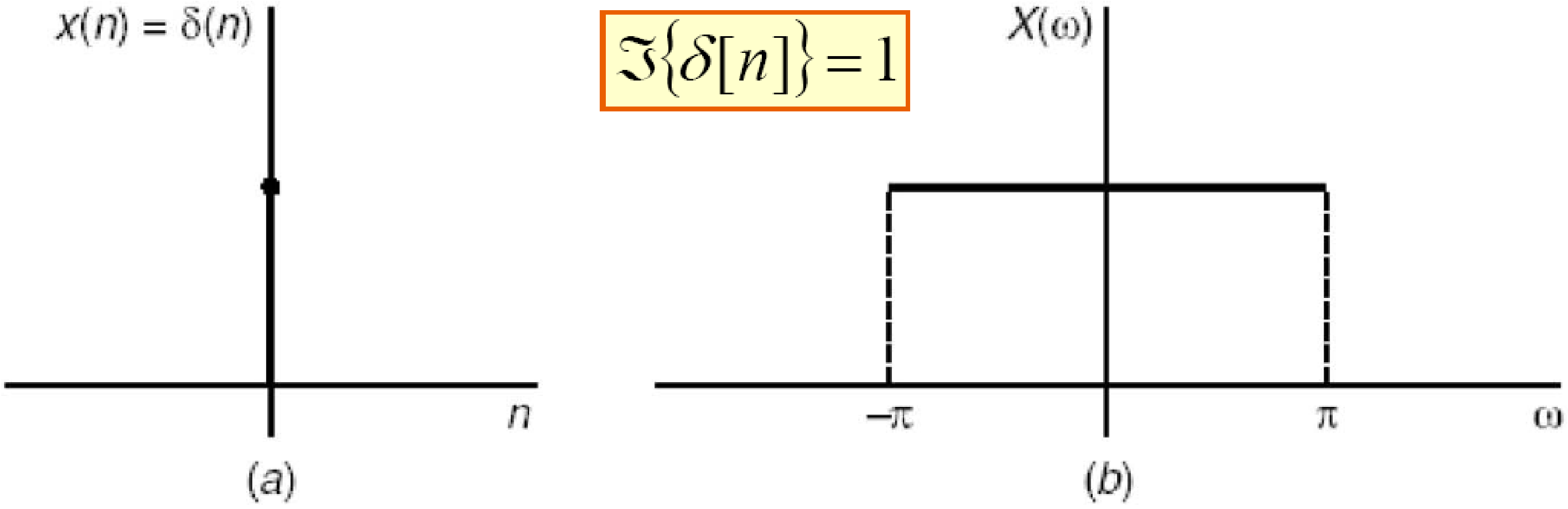
- **Multiplication** in time domain is equivalent to **convolution** in frequency domain

$$x[n] \cdot h[n] \overset{\Im}{\Leftrightarrow} \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\gamma) H(\omega - \gamma) d\gamma$$

Important DTFT Pairs: Impulse Function

- The DTFT of the impulse function is "1" over the entire frequency band

$$\boxed{\Im\{\delta[n]\} = 1}$$

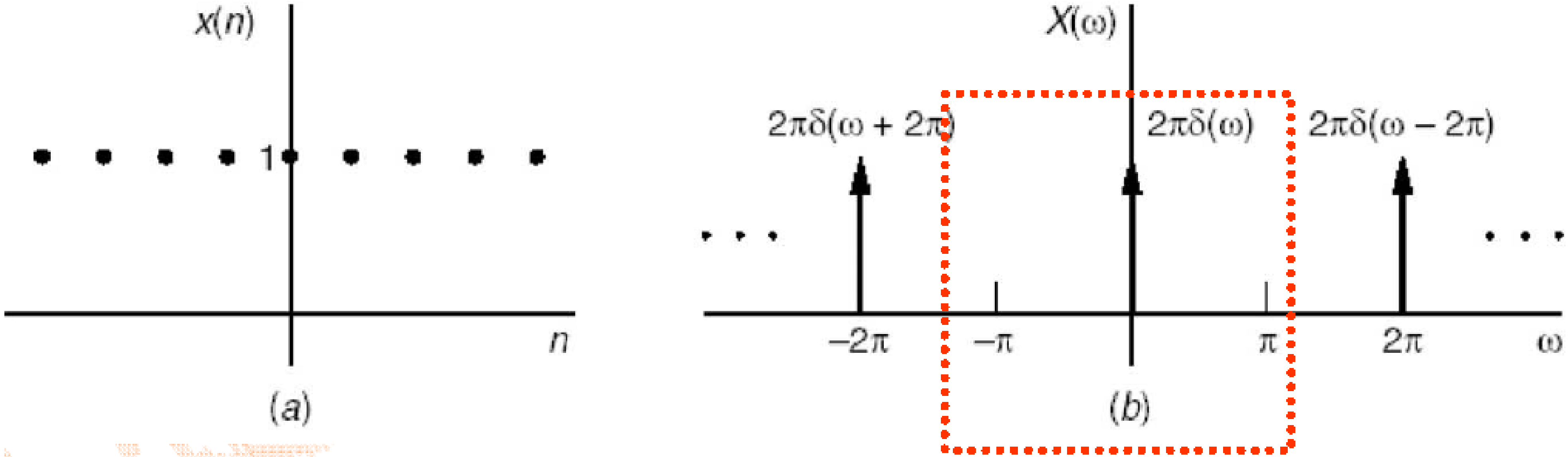


Extend of the frequency band
in discrete frequency domain

Important DTFT Pairs: Constant Function

- Note that $x[n] = 1$ (or any other constant) does not satisfy absolute summability. However, we can show that the DTFT of the constant function is an impulse at $\omega = 0$. (this should make sense!!!)

$$\boxed{\Im\{1\} = 2\pi \sum_{m=-\infty}^{\infty} \delta(\omega - 2\pi m)}$$

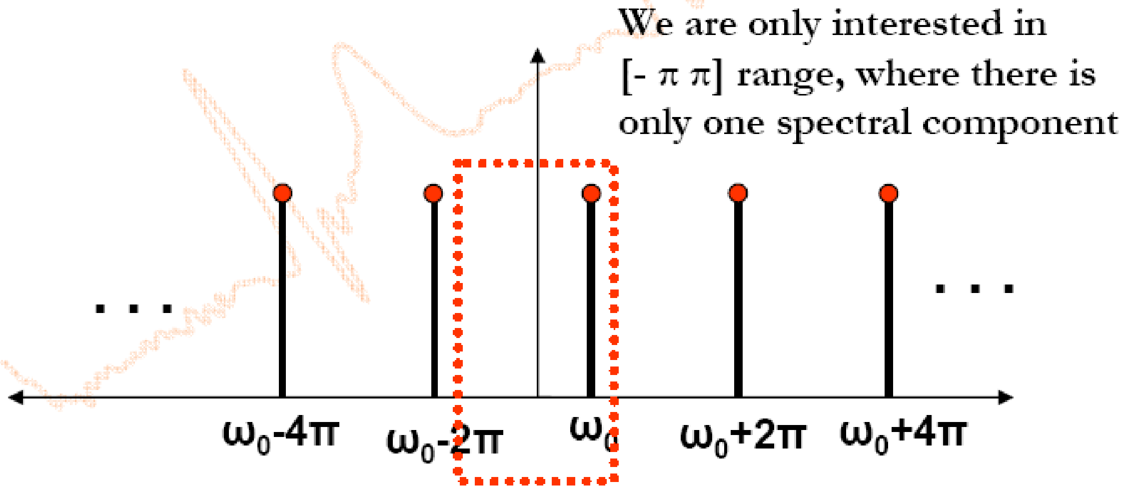


Important DTFT Pairs: The Complex Exponential

- The DTFT of the complex exponential:

$$x[n] = e^{j\omega_0 n} \Leftrightarrow X(\omega) = 2\pi \sum_{k=-\infty}^{\infty} \delta(\omega - \omega_0 + 2\pi k)$$

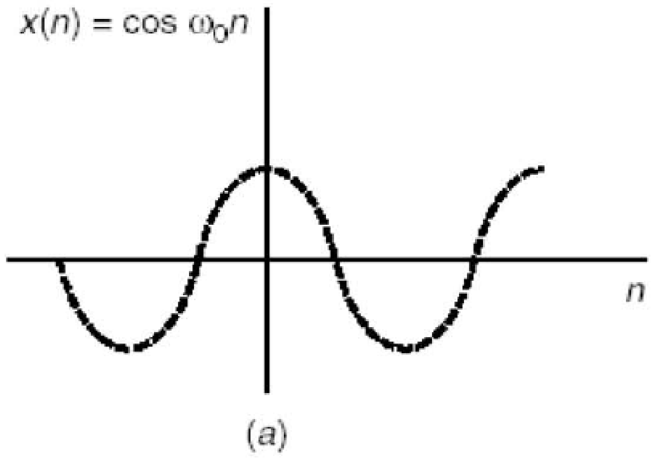
- Hence, the spectrum of a single complex exponential at a specific frequency is an impulse at that frequency
- This can be verified by computing the inverse DTFT of $X(\omega)$ given above

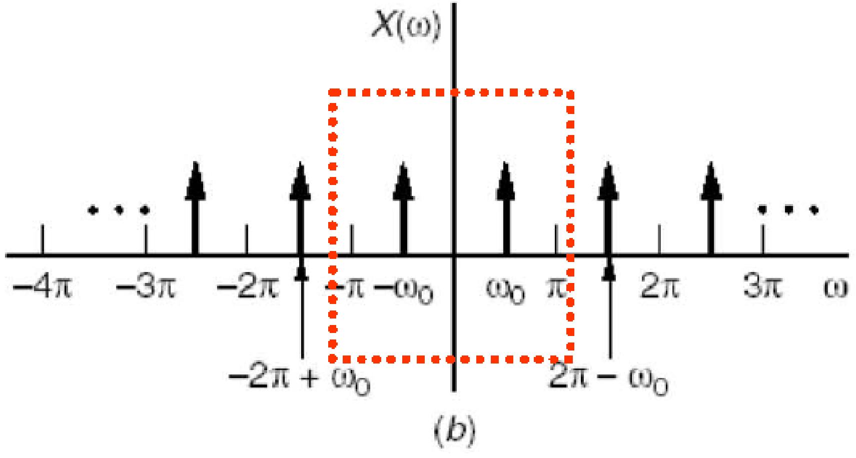


Important DTFT Pairs: The Sinosoid at ω_0

- By far the most often used DTFT pair (it is less complicated then it looks):

$$x[n] = \cos(\omega_0 n) \overset{\Im}{\Leftrightarrow} \pi \sum_{m=-\infty}^{\infty} \delta(\omega - 2\pi m - \omega_0) + \pi \sum_{m=-\infty}^{\infty} \delta(\omega - 2\pi m + \omega_0)$$





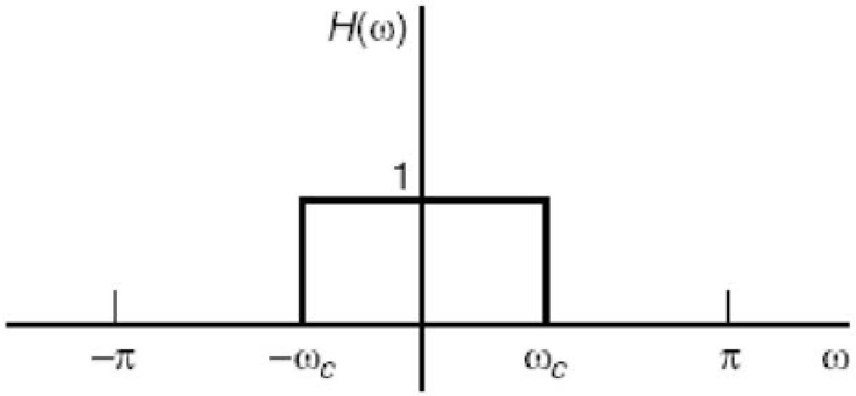
$$x[n] = e^{j\omega_0 n} \overset{\Im}{\Leftrightarrow} 2\pi \sum_{m=-\infty}^{\infty} \delta(\omega - \omega_0 \pm 2\pi m)$$

The expression can also be obtained from the DTFT of the complex exponential through the Euler's formula

Ideal Lowpass Filter

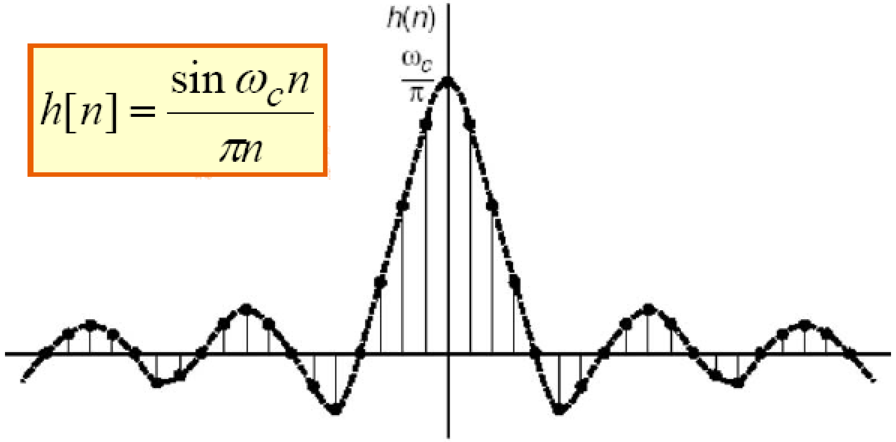
- The **ideal lowpass filter** is defined as

$$H(\omega)=\begin{cases}1, & |\omega|\le\omega_c\\ 0, & \omega_c\le\omega\le\pi\end{cases}$$



- Taking its inverse DTFT, we can obtain the corresponding impulse function $h[n]$:

$$h[n]=\frac{\sin\omega_c n}{\pi n}$$

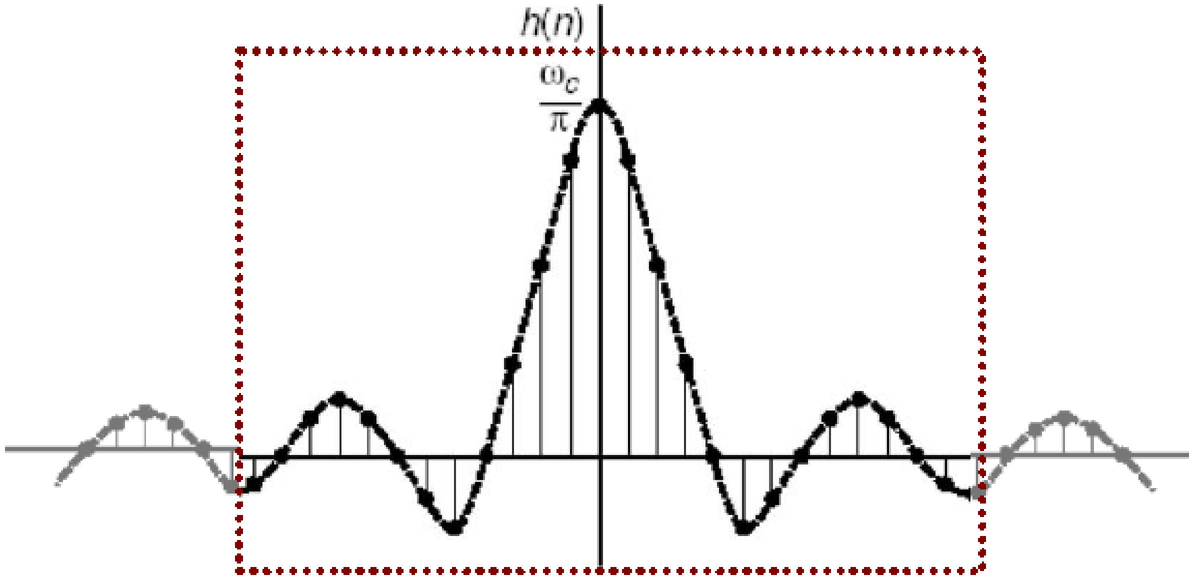


Ideal Lowpass Filter

Note that:

- The impulse response of an ideal LPF is infinitely long → This is an IIR filter. In fact $h[n]$ is not absolutely summable → its DTFT cannot be computed → an ideal $h[n]$ cannot be realized!

- One possible solution is to truncate $h[n]$, say with a window function, and then take its DTFT to obtain the frequency response of a realizable FIR filter



Some Useful Matlab Functions

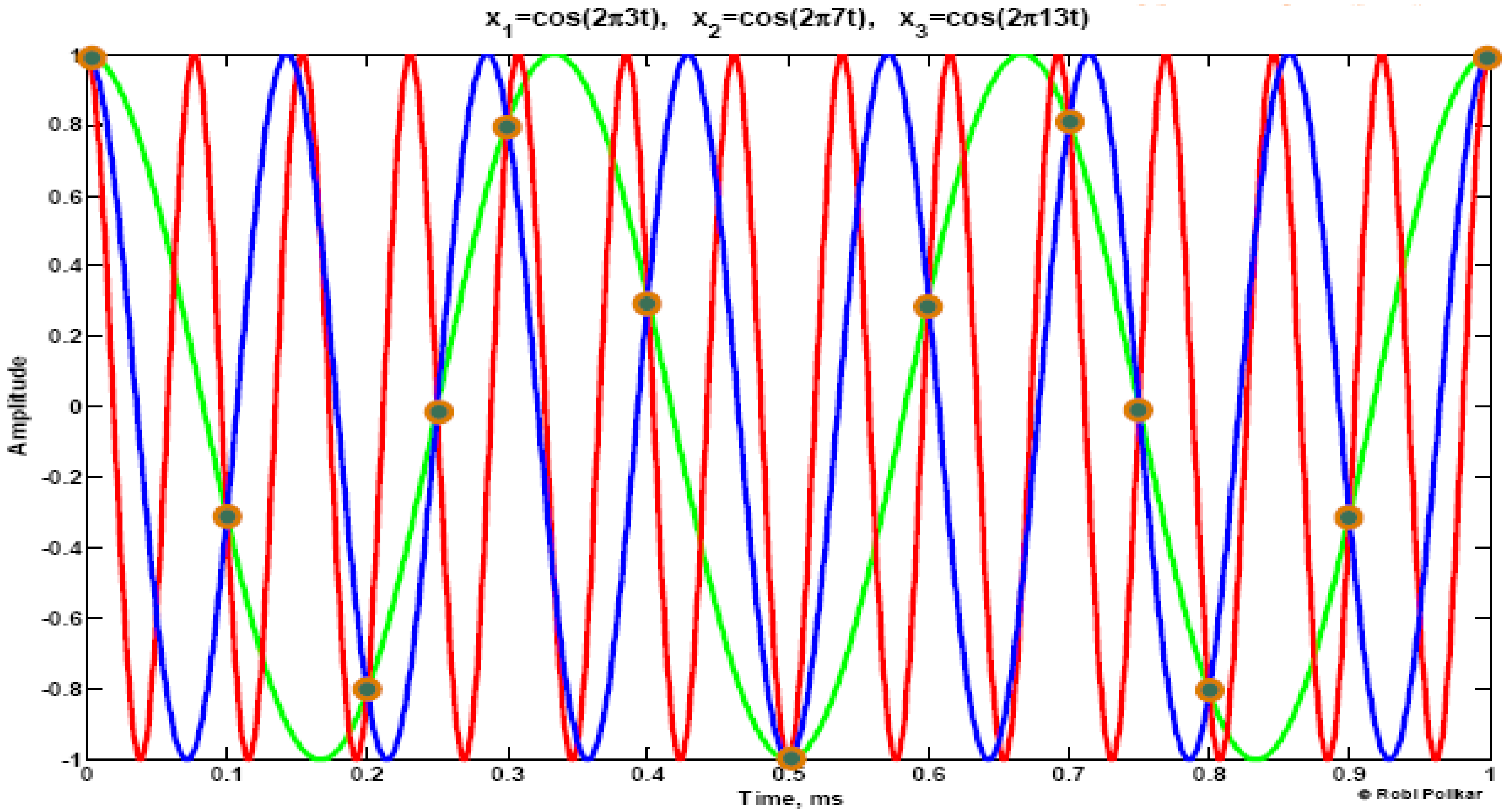
- Matlab cannot explicitly calculate the DTFT, since the frequency axis is continuous
- However, it can calculate an approximation of the DTFT using a given number of points
- y=fft(x, N) – Calculates the Discrete Fourier Transform (DFT) of the signal x at N points. If N is not provided, length of y is the same as x. DFT is a sampled version of the DTFT, where the samples are taken at N equidistant points around the unit circle from 0 to π
- [h,w]=freqz(b,a,N,'whole') – Calculates the frequency response of a filter whose CCLDE coefficients are given as b and a, using N number of points around the unit circle. If ‘whole’ is included, it returns a frequency base of ω from 0 to 2π, otherwise, from 0 to π

Some Useful Matlab Functions

- y=abs(x)- Calculates the absolute value of signal x. For complex values signals, the output is the magnitude (spectrum) of the complex argument
- y=angle(x) – Calculates the phase (spectrum) of the signal x
- q=unwrap(p) - corrects the radian phase angles in a vector p by adding multiples of 2π when absolute jumps between consecutive elements of p are greater than the default jump tolerance of π radians
- y=fftshift(x) - rearranges the outputs of fft by moving the zero-frequency component to the center of the array. It is useful for visualizing a Fourier transform with the zero-frequency component in the middle of the spectrum

Sampling & Aliasing

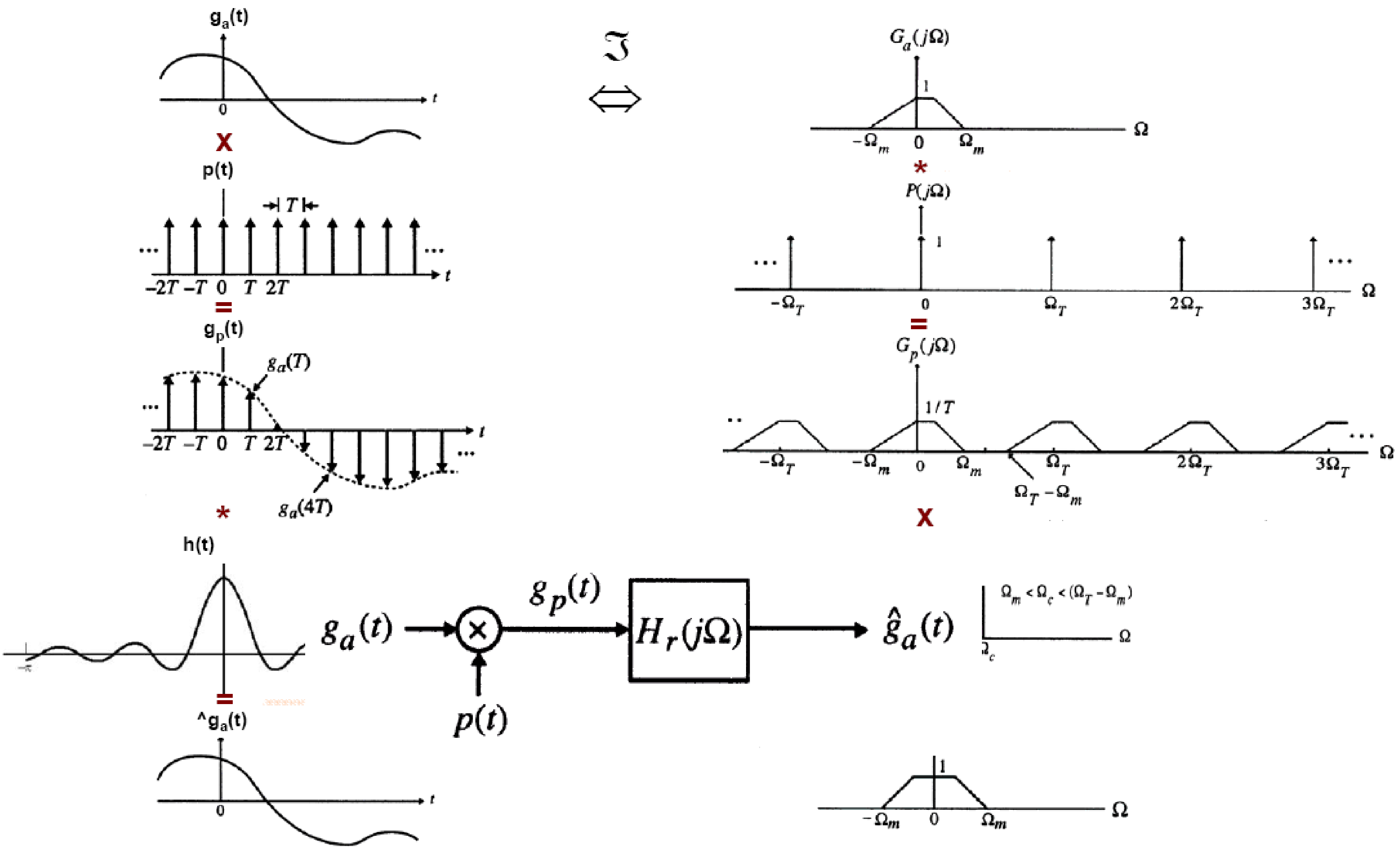
- In sampling, identical discrete-time signals may result from the sampling of more than one distinct continuous-time function. In fact, there exists an infinite number of continuous-time signals, which when sampled lead to the same discrete-time signal



Shannon's Sampling Teorem

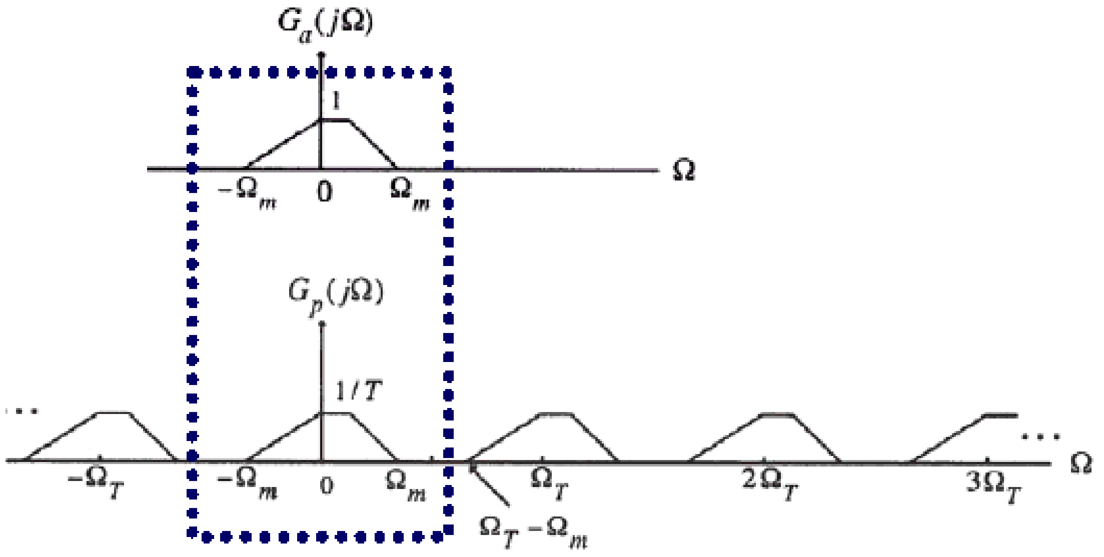
- The solution to this complicated and perplexing phenomenon comes from the amazingly simple **Shannon's Sampling Theorem**, one of the cornerstones of the modern communications, signal processing and control
  - A continuous time signal $x(t)$, with frequencies no higher then $\Omega_{\max} = 2\pi f_{\max}$ can be reconstructed exactly, precisely and uniquely from its samples $x[n] = x(nT_S)$, if the samples are taken at a sampling rate (frequency) of $f_S = 1/T_S$ or ($\Omega_S = 2\pi/T_S$) that is greater then $2f_{\max}$. The frequency $\Omega_S/2$ (or $f_S/2$ or $f_{\max}$) is called the **Nyquist frequency** (or **folding frequency**), as it determines the minimum sampling frequency required. The minimum required sampling frequency is then called the **Nyquist rate**
  - In other words, *if a continuous time signal is sampled at a rate that is at least twice as high (or higher) as the highest frequency in the signal, then it can be uniquely reconstructed from its samples*
  - **Aliasing** can be avoided if a signal is sampled at or above the **Nyquist rate**

Sampling (Graphically)



Nyquist Rate

- Note that the key requirement for the $G_a(\Omega)$ recovered from $G_p(\Omega)$ is that $G_p(\Omega)$ should consist of non-overlapping replicas of $G_a(\Omega)$

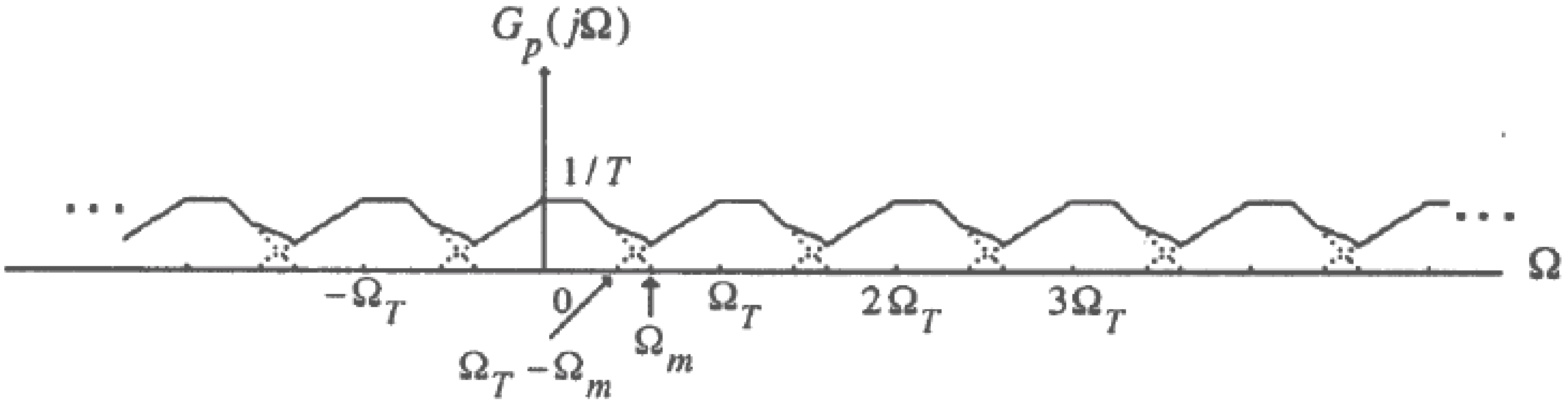


- Under what conditions would this be satisfied…?

If $\Omega_S \geq 2\Omega_m$, $g_a(t)$ can be recovered exactly from $g_p(t)$ by passing it through an ideal lowpass filter $H_r(\Omega)$ with a gain T_S and a cutoff frequency Ω_C greater than Ω_m and less than Ω_S - Ω_m. For simplicity, a half-band ideal filter is typically used in exercises

Aliasing - Revisited

- On the other hand, if $\Omega_S < 2\Omega_m$, due to the overlap of the shifted replicas of $G_a(\Omega)$ in the spectrum of $G_p(\Omega)$, the signal cannot be recovered by filtering
  - This is simply because the filtering of overlapped sections will cause a distortion by folding, or **aliasing**, the areas immediately outside the baseband back into the baseband



  - The frequency $\Omega_S /2$ is known as the **folding frequency**

Summary

- Given the discrete samples $g_a(nT_S)$, we can recover $g_a(t)$ exactly by generating the impulse train

$$g_p(t) = \sum_{n=-\infty}^{\infty} g_a(nT_S)\delta(t - nT_S)$$

  and then passing it through an ideal lowpass filter $H_r(\Omega)$ with a gain T_S and a cutoff frequency Ω_C satisfying $\Omega_m \leq \Omega_C \leq \Omega_S - \Omega_m$

- The highest frequency Ω_m contained in $g_a(t)$ is usually called the **Nyquist frequency** since it determines the minimum sampling frequency $\Omega_S = 2\,\Omega_m$ that must be used to fully recover $g_a(t)$ from its sampled version

Summary

- Sampling over or below the **Nyquist rate** is called **oversampling** or **undersampling**, respectively. Sampling exactly at this rate is **critical sampling**.
- A pure sinusoid may not be recoverable from **critical sampling**.
- Some amount of **oversampling** is usually used to allow some tolerance
- e.g. In phone conversations, 3.4 kHz is assumed to be the highest frequency in the speech signal, and hence the signal is samples at 8 kHz
- In digital audio applications, the full range of audio frequencies of 0 ~ 20 kHz is preserved. Hence, in CD audio, the signal is sampled at 44.1 kHz