PROCESSAMENTO DE SINAL E IMAGEM EM FÍSICA MÉDICA

2019/2020 – 2º Semestre (F4012)

Discrete-Time Signals

- Time-Domain Representation
- Basic Sequences
- The Sampling Process

Discrete-Time Systems

Discrete-Time Signals: Time-Domain Representation

Signals may be represented as sequences of numbers, called samples. The sample values are denoted as x[n], with n being an integer in the range $-\infty$ to $+\infty$.

x[n] is defined only for integer values of n, and undefined for non-integer values of n.

A discrete time signal may be also written as a sequence of numbers inside braces:

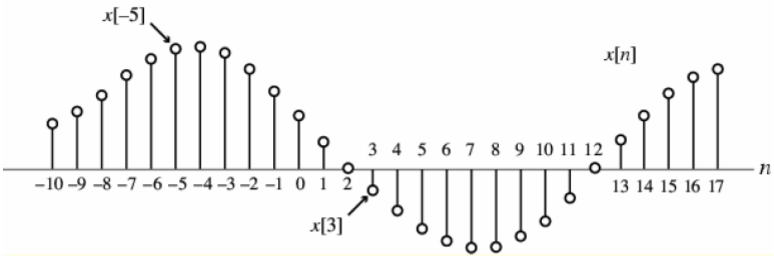
$$\{x[n]\} = \{\ldots, -0.2, \underset{\uparrow}{2.2}, 1.1, 0.2, -3.7, 2.9, \ldots\}$$

The arrow (↑) indicates where n=0

In this example,
x[-1] = −0.2
x[0] = 2.2
x[4] = 2.9

Discrete-Time Signals: Time-Domain Representation

Example of the graphical representation of a discrete-time signal {x[n]}, with real valued samples.

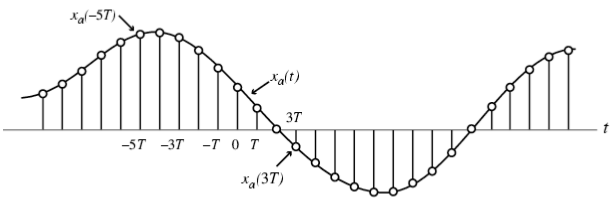


Often the braces {} are omitted when denoting a sequence, as long as there is no ambiguity in the representation.

{x[n]} can be represented simply as x[n]

Discrete-Time Signals: Time-Domain Representation

A Discrete time signal is often obtained by sampling a continuous-time signal, at uniform intervals of time



Sequence $x[n] = x_a(nT)$, $n = \ldots,-1,0,1,2,\ldots$

T= sampling period

$1/T$= sampling frequency

Discrete-Time Signals: Time-Domain Representation

The n-th sample of a signal is given by

$x[n] = x_a(t)|_{t=nT} = x_a(nT)$, n = ...,−2,-1,0,1,2,...

where T is the sampling period, and $x_a(t)$ the analog signal.

The sampling frequency unit is **Hertz (Hz)**, cycles per second, and the time unit, for the period T, is in seconds (s).

x[n] is a **real sequence** if the n-th sample x[n] is real for all values of n.

Otherwise, x[n] is a **complex sequence**.

Discrete-Time Signals: Time-Domain Representation

A complex sequence x[n] can be written as

$x[n] = x_{re}[n] + j\, x_{im}[n]$

where $x_{re}[n]$ and $x_{im}[n]$ are the real and imaginary parts of x[n].

In DSP, the imaginary unit is usually represented as **j**, instead of **i**, because **I** (i) is used as a symbol for the electric current intensity.

The complex conjugate sequence of x[n] is given by

$x^*[n] = x_{re}[n] - j\, x_{im}[n]$

Discrete-Time Signals: Time-Domain Representation

EXAMPLES:

$x[n] = \{\cos 0.2n\}$ is a real sequence

$y[n] = \{e^{j0.3n}\}$ is a complex sequence

we can write

$y[n] = \{\cos 0.3n + j \sin 0.3n\} = y_{re}[n] + j\, y_{im}[n]$

where $y_{re}[n] = \{\cos 0.3n\}$ and $y_{im}[n] = \{\sin 0.3n\}$.

$w[n] = \{\cos 0.3n - j \sin 0.3n\} = \{e^{-j0.3n}\}$

is the complex conjugate sequence of $y[n]$

$w[n] = y^*[n]$

Discrete-Time Signals: Time-Domain Representation

A discrete-time signal may be a **finite-length** or an **infinite-length** sequence.

A finite-length (also called finite-duration or finite-extent) sequence is defined only for a finite time interval, $N_1 \le n \le N_2$, where

$-\infty < N_1$, $N_2 < \infty$ and $N_1 \le N_2$.

The **length** (duration) of the finite-length sequence is $N = N_2 - N_1 + 1$

EXAMPLES:

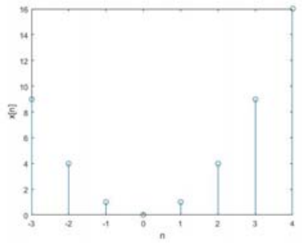
$x[n] = n^2$, $-3 \le n \le 4$ is finite-length sequence of length 4-(-3)+1=8

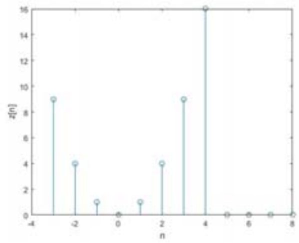
$y[n] = \cos 0.15n$ is an infinite-length sequence

Sequences – zero-padding

A **length-N** sequence is often referred to as a **N-point** sequence.

The length of a finite-length sequence can be increased by **zero-padding**, i.e. by appending it with zeros.





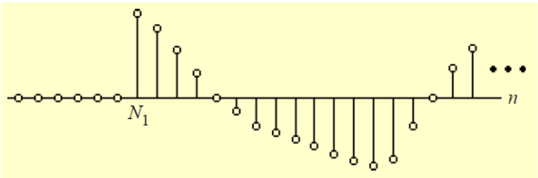
EXAMPLE:

$x[n] = n^2$, $-3 \leq n \leq 4$

$z[n] = \begin{cases} n^2, & -3 \leq n \leq 4 \\ 0, & 5 \leq n \leq 8 \end{cases}$ is finite-length sequence of length 8-(-3)+1=12

A **right-sided sequence** x[n] has zero-valued samples for $n < N_1$.

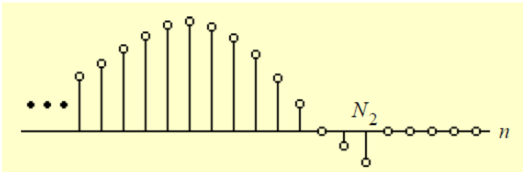
If $N_1 > 0$, a right-sided sequence is called a **causal sequence**.



Example of a right-sided sequence

A **left-sided sequence** x[n] has zero-valued samples for $n > N_2$.

If $N_2 \leq 0$, a left-sided sequence is called a **anti-causal sequence**.



Example of a left-sided sequence

L_p-norm

The L_p-norm $||x||_p$ of a signal x[n] is computed by

$$\|x\|_p = \left(\sum_{n=-\infty}^{\infty} |x[n]|^p \right)^{1/p}$$

where p is a positive integer (typically p is 1 or 2 or ∞).

The L_1-norm $||x||_1$ is linked with the **mean absolute value** of x[n].

The L_2-norm $||x||_2$ to the **root-mean-squared (rms) value** of x[n].

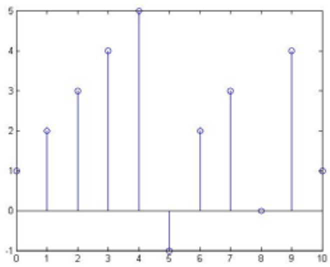
The L_∞-norm $||x||_\infty$ is the **peak absolute value** of x[n], i.e. $||x||_\infty = |x|_{max}$.

L_p-norm

EXAMPLE

Consider the sequence of length 11, defined for 0≤n≤10:

x[n] = { 1 2 3 4 5 -1 2 3 0 4 1 }



$L_1 = 26$

$L_2 = 9.2736$

$L_3 = 6.8824$

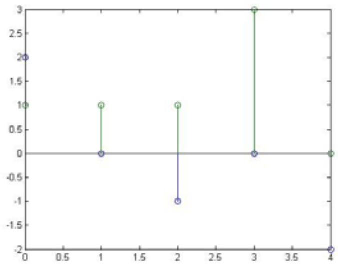
$L_\infty = 5$

L_p-norm

EXAMPLE

Consider the complex sequence of length 5, defined for 0≤n≤4:

x[n] = { 1+2j 1 1-j 3 -2j }



$L_1 = 9.6503$

$L_2 = 4.5826$

$L_3 = 3.6842$

$L_\infty = 3$

Relative Error using L2

Let y[n], 0 ≤ n ≤ N-1, be an approximation of x[n], 0 ≤ n ≤ N-1

An estimate of the relative error is given by the ratio of

the L2-norm of the differences signal

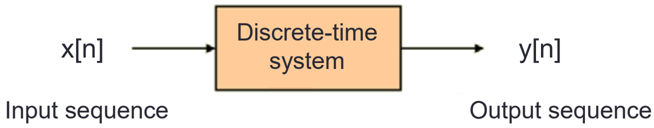
and

the L2-norm of x[n]

$$E_{rel} = \left(\frac{\sum_{n=0}^{N-1} \left| y[n] - x[n] \right|^2}{\sum_{n=0}^{N-1} \left| x[n] \right|^2} \right)^{1/2}$$

Operations on Sequences

A single-input, single-output, discrete-time system operates on a sequence according to some prescribed rules, producing an output sequence with more desirable properties than the input.



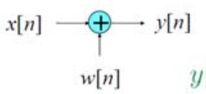
For example, the input may be corrupted with additive noise.

A discrete-time system can be designed to generate an output, removing the noise component from the input signal.

In most cases, the procedures defining a particular discrete-time system are composed of some basic-operations.

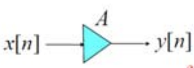
Basic Operations on Sequences

Addition:
- Adder



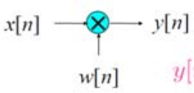
$$y[n] = x[n] + w[n]$$

Multiplication:
- Multiplier



$$y[n] = A \times x[n]$$

Product (modulation):
- Modulator



$$y[n] = x[n] \times w[n]$$

e.g. **Windowing** – multiplying an infinite-leght sequence by a finite-length sequence to extract a region

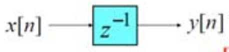
[Source: Dan Ellis, 2010]

Basic Operations on Sequences

Time-shifting: y[n]=x[n-N] , where N is an integer

if N>0, it is a **delaying** operation

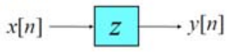
- Unit delay



$y[n] = x[n-1]$

if N<0, it is a **advance** operation

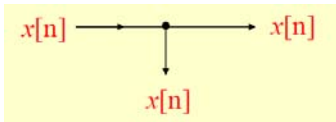
- Unit advance



$y[n] = x[n+1]$

Time-reversal (folding) operation: y[n] = x[-n]

Branching operation: Used to provide multiple copies of a sequence



[Source: Dan Ellis, 2010]

Basic Operations on Sequences

EXAMPLE

Consider the 2 following sequences, of length 5, defined for 0≤n≤4:

a[n] = { 3 4 6 -9 0 }

b[n] = { 2 -1 4 5 -3 }

New sequences can be produced from these two sequences, by applying basic operations, such as:

c[n] = a[n] × b[n] = { 6 -4 24 -45 0 }

d[n] = a[n] + b[n] = { 5 3 10 -4 -3 }

e[n] = 3/2 a[n] = { 4.5 6 9 -13.5 0 }

Basic Operations on Sequences

As pointed out in the example of the previous slide,

operations on two or more sequences can be carried out if all sequences have the same length and defined for the same range of the time index n.

If the sequences are not of the same length, such operation cannot be performed.

However, in some cases this problem can be circumvented by appending zero-valued samples to the smaller length sequence(s), to make all sequences have the same range of the time index.

Basic Operations on Sequences

EXAMPLE

Consider the sequence of length 3, defined for $0 \le n \le 2$:

$f[n] = \{ -2 \;\; 1 \;\; 3 \}$

We cannot add the length-3 sequence f[n] to the length-5 sequence

$a[n] = \{ 3 \;\; 4 \;\; 6 \;\; -9 \;\; 0 \}$

We can however append 2 zero-valued samples to f[n], thus producing a length-5 sequence $f_e[n]$

$f_e[n] = \{ -2 \;\; 1 \;\; 3 \;\; 0 \;\; 0 \}$

Which can be for example added to a[n]

$g[n] = a[n] + f_e[n] = \{ 1 \;\; 5 \;\; 9 \;\; -9 \;\; 0 \}$

Basic Operations on Sequences

Ensemble Averaging

- A very simple application of the addition operation in improving the quality of measured data corrupted by an additive random noise
- In some cases, actual uncorrupted data vector **s** remains essentially the same from one measurement to next
- While the additive noise vector is random and not reproducible

Basic Operations on Sequences

Let $\mathbf{d}_i$ denote the noise vector corrupting the i-th measurement of the uncorrupted data vector **s**:

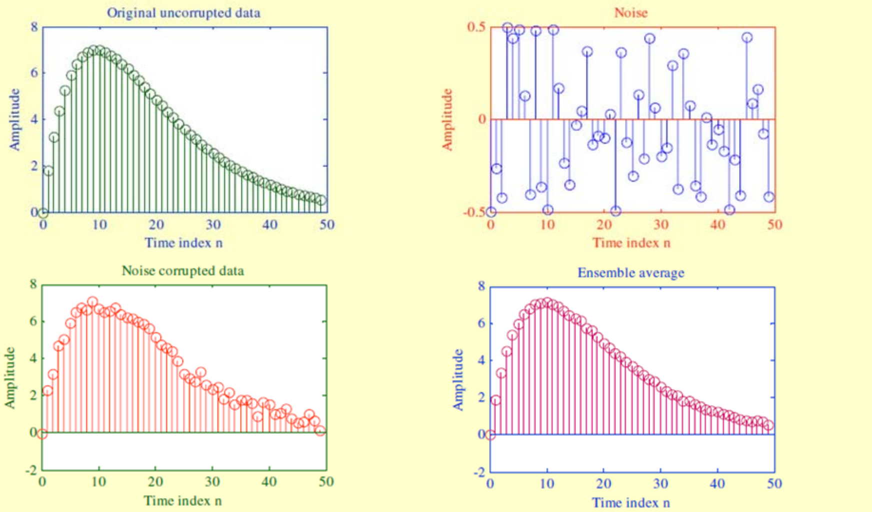
$$\mathbf{x}_i = \mathbf{s} + \mathbf{d}_i$$

The average data vector, called the ensemble average, obtained after K measurements is given by

$$\mathbf{x}_{ave} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{x}_i = \frac{1}{K}\sum_{i=1}^{K}(s + \mathbf{d}_i) = \mathbf{s} + \frac{1}{K}\sum_{i=1}^{K}\mathbf{d}_i$$

- For large values of K, $\mathbf{x}_{ave}$ is usually a reasonable replica of the desired data vector **s**

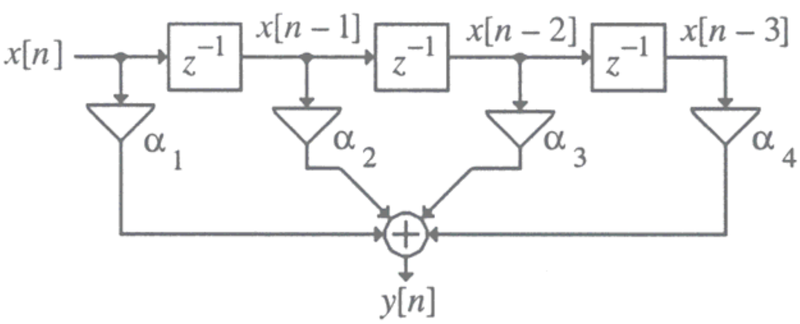
Basic Operations on Sequences



Basic Operations on Sequences

Combination of Basic Operations

$$y[n] = \alpha_1 x[n] + \alpha_2 x[n-1] + \alpha_3 x[n-2] + \alpha_4 x[n-3]$$



[Source: Dan Ellis, 2010]

Sampling Rate Modification

Employed to generate a new sequence $y[n]$ with a sampling rate F_T' higher or lower than the sampling rate F_T of a given sequence $x[n]$

Sampling rate alteration ratio is $R = \dfrac{F_T'}{F_T}$

- If $R > 1$, the process called **interpolation**
- If $R < 1$, the process called **decimation**

Sampling Rate Modification

In **up-sampling** by an integer factor $L > 1$, $L-1$ equidistant zero-valued samples are inserted by the **up-sampler** between each two consecutive samples of the input sequence $x[n]$:

$$x_u[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \cdots \\ 0, & \text{otherwise} \end{cases}$$

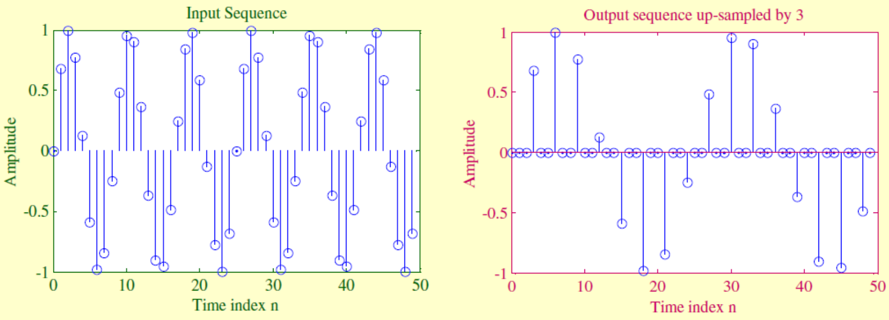
$$x[n] \longrightarrow \boxed{\uparrow L} \longrightarrow x_u[n]$$

Sampling Rate Modification

Up-sampling = adding more samples = interpolation

Example of an up-sampling operation

$$x[n] \longrightarrow \boxed{\uparrow 3} \longrightarrow x_u[n]$$



Sampling Rate Modification

In **down-sampling** by an integer factor $M > 1$, every M-th samples of the input sequence are kept and $M-1$ in-between samples are removed:

$$y[n] = x[nM]$$

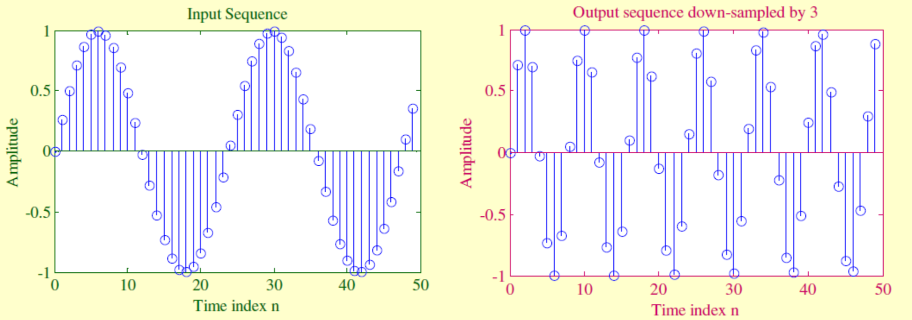
$$x[n] \longrightarrow \boxed{\downarrow M} \longrightarrow y[n]$$

Sampling Rate Modification

Down-sampling = discarding samples = decimation

Example of a down-sampling operation

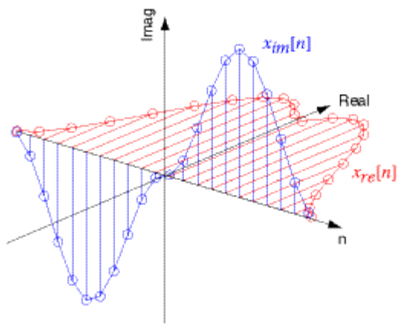
$$x[n] \longrightarrow \boxed{\downarrow 3} \longrightarrow y[n] = x[3n]$$



Classification of Sequences

Conjugate symmetric sequence $x_{cs}[n] = x_{cs}^*[-n]$

$$x_{re}[n] + j \cdot x_{im}[n] = x_{cs}[n] = x_{cs}^*[-n] = x_{re}[-n] - j \cdot x_{im}[-n]$$



If x[n] is real, then it is an **even** sequence

[Source: Dan Ellis, 2010]

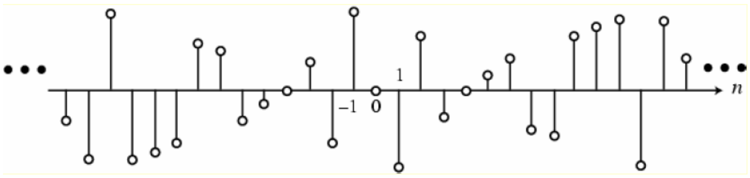
Classification of Sequences

Conjugate anti-symmetric sequence

$x_{ca}[n] = -\, x_{ca}^*[-n]$

$x_{re}[n] + j\cdot x_{im}[n] = x_{ca}[n] = -\, x_{ca}^*[-n] = -\, x_{re}[-n] + j\cdot x_{im}[-n]$

If x[n] is real, then it is an **odd** sequence



Classification of Sequences

$x_{cs}[n] = x_{cs}^*[-n]$ $x_{ca}[n] = -\, x_{ca}^*[-n]$

It follow from the definitions that:

- For a conjugate-symmetric sequence x[n]
  x[0] must be a real number
- For a conjugate anti-symmetric sequence y[n]
  y[0] must be an imaginary number
- For an odd real sequence w[n], w[0]=0

Classification of Sequences

Any complex sequence can be expressed as a sum of its conjugate-symmetric and conjugate anti-symmetric parts.

$$x[n] = x_{cs}[n] + x_{ca}[n]$$

where

$$x_{cs}[n] = (x[n] + x^*[-n]) / 2$$

$$x_{ca}[n] = (x[n] – x^*[-n]) / 2$$

An exemple is presented next...

Consider the sequence: g[n]= { 0, 1+4j, -2+3j, 4-2j, -5-6j, -2j, 3 }

Its conjugate seq. is: g*[n]= { 0, 1-4j, -2-3j, 4+2j, -5+6j, 2j, 3 }

And its time-revered is: g*[-n]= {3, 2j, -5+6j, 4+2j, -2-3j, 1-4j, 0 }

$$g_{cs}[n] = (g[n] + g^*[-n]) / 2 =$$
$$= \{ 1.5, 0.5+3j, -3.5+4.5j, 4, -3.5-4.5j, 0.5-3j, 1.5 \}$$

$$g_{ca}[n] = (g[n] - g^*[-n]) / 2 =$$
$$= \{ -1.5, 0.5+j, 1.5-1.5j, -2j, -1.5-1.5j, -0.5+j, 1.5 \}$$

Pode verificar-se que $g[n] = g_{cs}[n] + g_{ca}[n]$

e que $g_{cs}[n] = g^*_{cs}[-n]$ e $g_{ca}[n] = -g^*_{ca}[-n]$

Classification of Sequences

Any real sequence can be expresses as a sum of its even and odd parts:

$$x[n] = x_{ev}[n] + x_{od}[n]$$

where

$$x_{ev}[n] = (x[n] + x[-n]) / 2$$

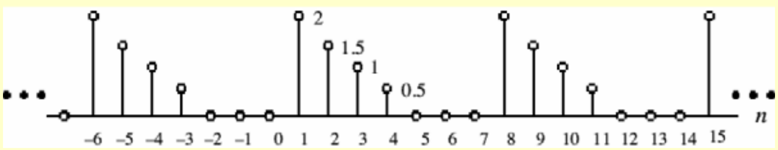
$$x_{od}[n] = (x[n] – x[-n]) / 2$$

Classification of Sequences

A sequence $\tilde{x}[n]$ satisfying $\tilde{x}[n] = \tilde{x}[n + kN]$ is called a **periodic sequence** with a **period** N where N is a positive integer and k is any integer

- Smallest value of N satisfying $\tilde{x}[n] = \tilde{x}[n + kN]$ is called the **fundamental period**

Example



- A sequence not satisfying the periodicity condition is called an **aperiodic sequence**

Sequences – Energy and Power Signals

Total **energy** of a sequence $x[n]$ is defined by

$$\mathcal{E}_x = \sum_{n=-\infty}^{\infty} |x[n]|^2$$

- An infinite length sequence with finite sample values may or may not have finite energy
- A finite length sequence with finite sample values has finite energy

Sequences – Energy and Power Signals

The **average power** of an aperiodic sequence is defined by

$$P_x = \lim_{K\to\infty} \frac{1}{2K+1} \sum_{n=-K}^{K} |x[n]|^2$$

Define the **energy** of a sequence $x[n]$ over a finite interval $-K \le n \le K$ as

$$\mathcal{E}_{x,K} = \sum_{n=-K}^{K} |x[n]|^2$$

Then $P_x = \lim_{K\to\infty} \frac{1}{2K+1} \mathcal{E}_{x.K}$

Sequences – Energy and Power Signals

The **average power** of a periodic sequence $\tilde{x}[n]$ with a period N is given by

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1}\left|\tilde{x}[n]\right|^2$$

- The average power of an infinite-length sequence may be finite or infinite

Sequences – Energy and Power Signals

- A finite energy signal with zero average power is called an **energy signal**

Example - An infinite-length sequence which has finite energy but zero average power

$$x[n] = \begin{cases} \dfrac{1}{n}, & n \geq 0 \\ 0\ , & n < 0 \end{cases}$$

Classification of Sequences

A sequence $x[n]$ is said to be **bounded** if

$$|x[n]| \le B_x < \infty$$

Example: The sequence $x[n] = \cos 0.3\pi n$ is a bounded sequence as

$$|x[n]| = |\cos 0.3\pi n| \le 1$$

Classification of Sequences

A sequence $x[n]$ is said to be **absolutely summable** if

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty$$

Example: The sequence

$$y[n] = \begin{cases} 0.3^n, & n \ge 0 \\ 0, & n < 0 \end{cases}$$

is an absolutely summable sequence as

$$\sum_{n=0}^{\infty} |0.3^n| = \frac{1}{1-0.3} = 1.42857 < \infty$$

Classification of Sequences

A sequence $x[n]$ is said to be **square-summable** if

$$\sum_{n=-\infty}^{\infty} \left| x[n] \right|^2 < \infty$$

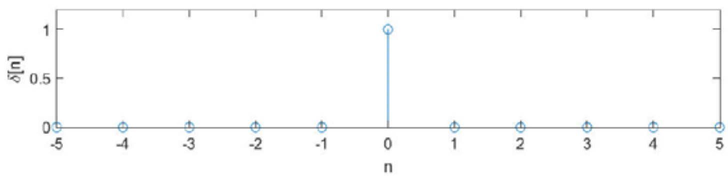
Example: The sequence

$$h[n] = \frac{\sin 0.4n}{\pi n}$$

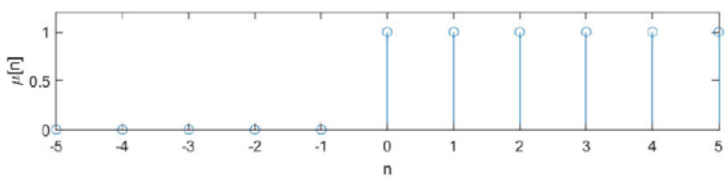
is square-summable but not absolutely summable

Basic Sequences

$\delta[n] = \begin{cases} 1\ , & n=0 \\ 0\ , & n\neq 0 \end{cases}$ is a **Unit sample sequence** (impulso unitário)



$\mu[n] = \begin{cases} 1\ , & n\geq 0 \\ 0\ , & n<0 \end{cases}$ is a **Unit step sequence** (degrau unitário)



Basic Sequences

$x[n] = A \cos(\omega_0 n + \phi)$ is a **Real sinusoidal sequence**,

where

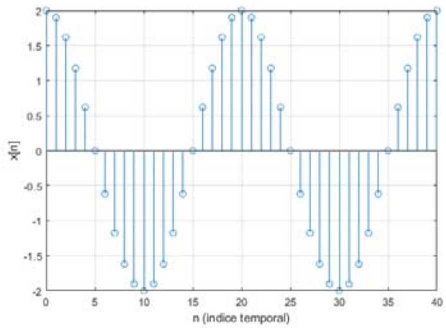
A – Amplitude
ω_0 – angular frequency
ϕ – phase

Example:

$A = 2$

$\omega_0 = 0.1\pi$

$\phi = 0$



Basic Sequences

$x[n] = A\,\alpha^n$, $-\infty < n < +\infty$ is an **Exponential sequence**,

where A and α are real or complex numbers.

If we write $\alpha = e^{(\sigma_0 + j\omega_0)}$, $A = |A|\,e^{j\phi}$, then we can express

$x[n] = |A|\,e^{j\phi}\,e^{(\sigma_0 + j\omega_0)\,n} = x_{re}[n] + j\,x_{im}[n]$

where

$x_{re}[n] = |A|\,e^{\sigma_0 n} \cos(\omega_0 n + \phi)$

$x_{im}[n] = |A|\,e^{\sigma_0 n} \sin(\omega_0 n + \phi)$

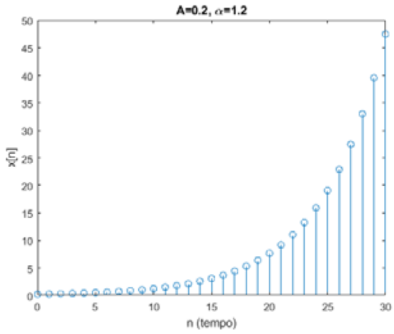
$x_{re}[n]$ and $x_{im}[n]$ of a complex exponential sequence $x[n]$ are real sinusoidal sequences with constant (σ_0=0), growing (σ_0>0) or decaying (σ_0<0) amplitudes for n>0.

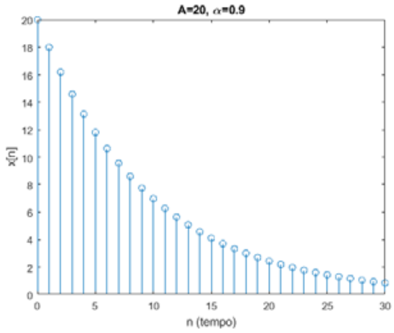
Basic Sequences

Real exponential sequence

$x[n] = A\,\alpha^n$, $-\infty < n < +\infty$, where A and α are real numbers.

Examples:





Basic Sequences

Example:

Complex exponential sequence

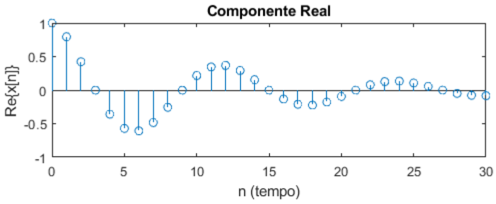
$$x[n] = \exp\left(-\frac{1}{12} + j\,\frac{\pi}{6}\right) n$$

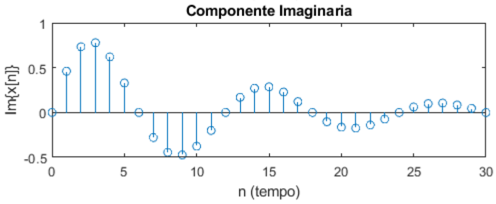
$$x[n] = |A|\, e^{\,j\phi}\, e^{\,(\sigma_0 + j\omega_0)\, n}$$

$$x[n] = x_{re}[n] + j\, x_{im}[n]$$

$$x_{re}[n] = |A| e^{\sigma_0 n} \cos(\omega_0 n + \phi)$$

$$x_{im}[n] = |A| e^{\sigma_0 n} \sin(\omega_0 n + \phi)$$





Basic Sequences

The sinusoidal sequence $A \cos(\omega_0 n + \phi)$

and the complex exponential sequence $B \exp(j\omega_0 n)$

are **periodic sequences** of period N, if $\omega_0 N = 2\pi r$,

where N and r are positive integers.

The smallest values of N satisfying $\omega_0 N = 2\pi r$ is the **fundamental period** of the sequence.

Basic Sequences

Consider $x_1[n] = \cos(\omega_0 n + \phi)$ and $x_2[n] = \cos(\omega_0(n+N) + \phi)$

Now, $x_2[n] = \cos(\omega_0 n+\phi) \cos(\omega_0 N) - \sin(\omega_0 n+\phi) \sin(\omega_0 N)$

which is equal to $\cos(\omega_0 n+\phi)=x_1[n]$ only if $\sin(\omega_0 N)=0$ and $\cos(\omega_0 N)=1$

These two conditions are met if and only if $\omega_0 N=2\pi r$, or $\frac{2\pi}{\omega_0} = \frac{N}{r}$

- If $\frac{2\pi}{\omega_0}$ is a non-integer rational number, , then the period will be a multiple of $\frac{2\pi}{\omega_0}$
- Otherwise, the sequence is **aperiodic**

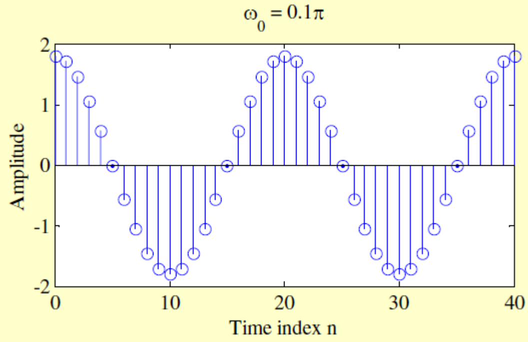
Example: $x[n] = \sin(\sqrt{3}\, n + 2)$ is an aperiodic sequence

Basic Sequences

$$A\cos(\omega_o n + \phi)$$

Example

$$\omega_o = 0.1\pi$$



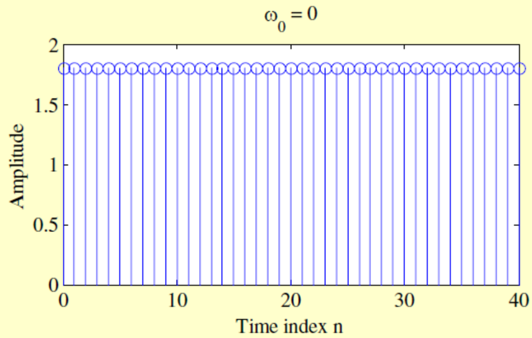
$$N = \frac{2\pi r}{0.1\pi} = 20 \text{ for } r = 1$$

Basic Sequences

$$A\cos(\omega_o n + \phi)$$

Example

$$\omega_o = 0$$



$$N = \frac{2\pi r}{\omega_o} \qquad N = 1 \text{ for } r = 0$$

Basic Sequences

Property 1

Consider $x[n] = \exp(j\omega_1 n)$ and $y[n] = \exp(j\omega_2 n)$
with $0 \le \omega_1 < \pi$ and $2\pi k \le \omega_2 < 2\pi(k+1)$
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

Basic Sequences

Property 2

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
$\omega = 0$ are called **low frequencies**, whereas,
frequencies in the neighborhood of $\omega = \pi$ are
called **high frequencies**

Basic Sequences

- Because of Property 1, a frequency ω_o in the neighborhood of $\omega = 2\pi\, k$ is indistinguishable from a frequency $\omega_o - 2\pi k$ in the neighborhood of $\omega = 0$

  and a frequency ω_o in the neighborhood of $\omega = \pi(2k+1)$ is indistinguishable from a frequency $\omega_o - \pi(2k+1)$ in the neighborhood of $\omega = \pi$

Basic Sequences

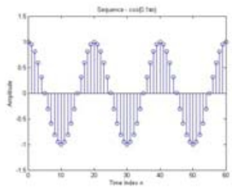
Signals with frequencies in the neighbourhood of ω=2πk are called **low frequencies**.

Signals with frequencies in the neighbourhood of ω=π(2k+1) are called **high frequencies**.

<u>EXAMPLES:</u>

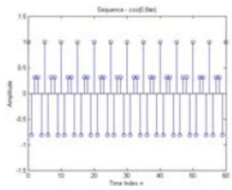
a[n] = cos (0.1πn) = cos (1.9πn)

is a low frequency signal.



b[n] = cos (0.8πn) = cos (1.2πn)

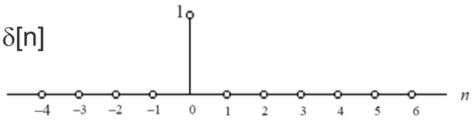
is a high frequency signal.



Basic Sequences

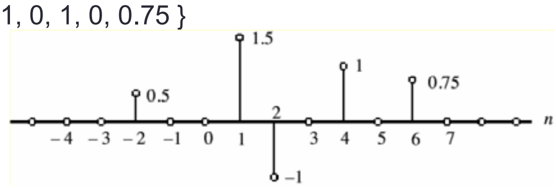
A sequence can be represented in the time domain as a weighted sum of an impulse sequence ($\delta[n]$) and its delayed and advanced versions.

$\delta[n]$



EXAMPLE:

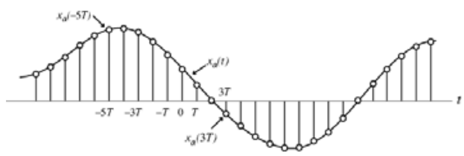
$x[n] = \{ 0.5, 0, 0, 1.5, -1, 0, 1, 0, 0.75 \}$
↑



$x[n] = 0.5\ \delta[n+2] + 1.5\ \delta[n-1] - \delta[n-2] + \delta[n-4] + 0.75\ \delta[n-6]$

The Sampling Process

Often a discrete-time sequence x[n] is created by uniformly sampling a continuous-time signal $x_a(t)$.



The relation between the two signals is

$$x[n] = x_a(t)|_{t=nT} = x_a(nT), \quad n = \ldots,-2,-1,0,1,2,\ldots$$

The time variable t of $x_a(t)$ is related to the variable n of x[n] only at discrete-time instants t_n, given by

$$t_n = nT = n / F_T = 2\pi n / \Omega_T$$

with $F_T = 1/T$ – sampling frequency
$\Omega_T = 2\pi\ F_T$ – sampling angular frequency

The Sampling Process

Consider the continuous-time signal

$x(t) = A \cos(2\pi f_0 t + \phi) = A \cos(\Omega_0 t + \phi)$

The corresponding discrete-time signals is

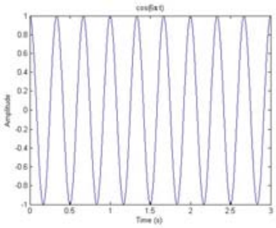
$x[n] = A \cos(\Omega_0 nT + \phi) = A \cos(2\pi n\Omega_0/\Omega_T + \phi) = A \cos(\omega_0 n + \phi)$

where $\omega_0 = 2\pi\Omega_0/\Omega_T = \Omega_0 T$ is the normalized digital angular freq. of x[n]

If the unit of the sampling period (T) is in seconds, then the unit of:

- the normalized digital angular frequency (ω_0) is radians/sample
- the normalized analog angular frequency (Ω_0) is radians/sample
- the analog frequency (f_0) is Hertz (Hz)

The Sampling Process



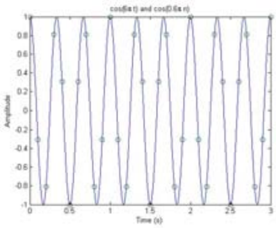
The three continuous-time signals

$g_1(t) = \cos(6\pi t)$

$g_2(t) = \cos(14\pi t)$

$g_3(t) = \cos(26\pi t)$

of frequencies 3 Hz, 7 Hz and 13 Hz, are sampled at a rate of 10 Hz (sampling rate of 10 Hz – T=0.1 s).

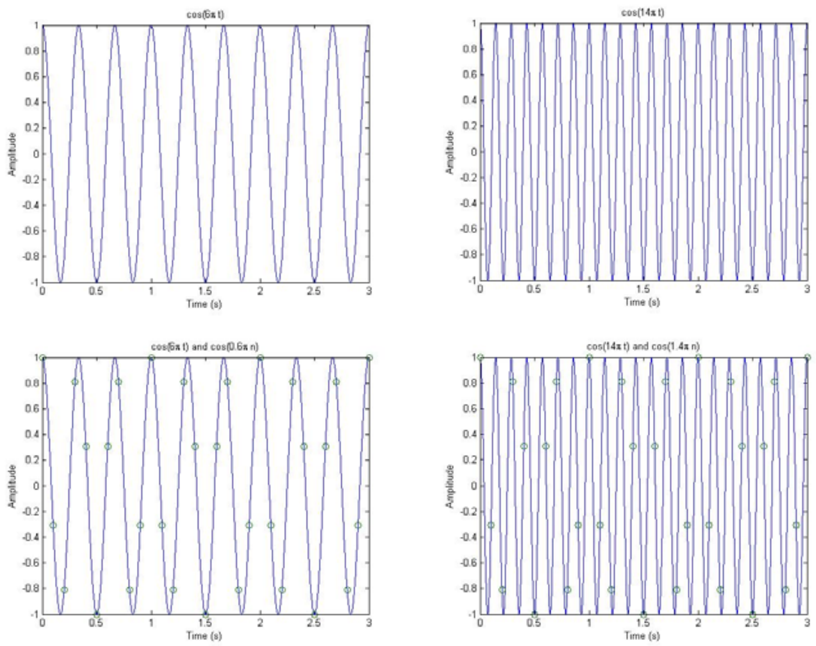


The process generates the following three sequences:

$g_1[n] = \cos(0.6\pi n)$

$g_2[n] = \cos(1.4\pi n)$

$g_3[n] = \cos(2.6\pi n)$



The Sampling Process

Plot of the three continuous-time signals

$g_1(t) = \cos(6\pi t)$

$g_2(t) = \cos(14\pi t)$

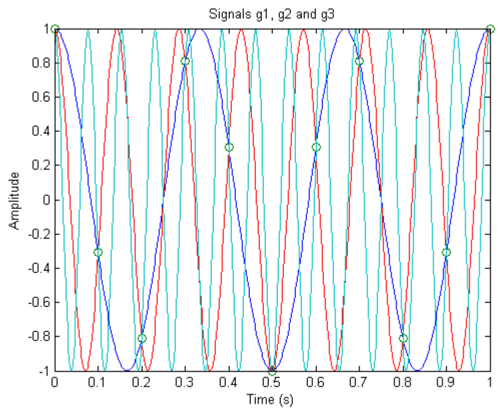
$g_3(t) = \cos(26\pi t)$

and the corresponding sequences, obtained by sampling at 10 Hz

$g_1[n] = \cos(0.6\pi n)$

$g_2[n] = \cos(1.4\pi n)$

$g_3[n] = \cos(2.6\pi n)$



The three sequences, $g_1[n]$, $g_2[n]$ and $g_3[n]$, have exactly the same values for every n.

The Sampling Process

This fact can be verified by observing that

$g_2[n] = \cos(1.4\pi n) = \cos((2\pi - 0.6\pi)n) = \cos((0.6\pi)n) = g_1[n]$

$g_3[n] = \cos(2.6\pi n) = \cos((2\pi + 0.6\pi)n) = \cos((0.6\pi)n) = g_1[n]$

As a result, these three sequences are identical.

It is thus impossible to associate a unique continuous-time function to each sequence.

This phenomenon – a continuous-time signal of higher frequency acquiring the identity of a sinusoidal sequence of lower frequency, after the sampling process, is called **aliasing**. (alias – *réplica*)

The Sampling Process

- Since there are an infinite number of continuous-time signals that can lead to the same sequence when sampled periodically, additional conditions need to imposed so that the sequence $\{x[n]\} = \{x_a(nT)\}$ can uniquely represent the parent continuous-time signal $x_a(t)$
- In this case, $x_a(t)$ can be fully recovered from $\{x[n]\}$

The Sampling Process

Example

Determine the discrete-time signal $v[n]$ obtained by uniformly sampling at a sampling rate of 200 Hz the continuous-time signal

$$v_a(t) = 6\cos(60\pi t) + 3\sin(300\pi t) + 2\cos(340\pi t) + 4\cos(500\pi t) + 10\sin(660\pi t)$$

- Note: $v_a(t)$ is composed of 5 sinusoidal signals of frequencies 30 Hz, 150 Hz, 170 Hz, 250 Hz and 330 Hz
  - The sampling period is $T = \frac{1}{200} = 0.005$ sec

The Sampling Process

The generated discrete-time signal $v[n]$ is given by

$$v[n] = 6\cos(0.3\pi n) + 3\sin(1.5\pi n) + 2\cos(1.7\pi n) + 4\cos(2.5\pi n) + 10\sin(3.3\pi n)$$

$$= 6\cos(0.3\pi n) + 3\sin\big((2\pi - 0.5\pi)n\big) + 2\cos\big((2\pi - 0.3\pi)n\big) + 4\cos\big((2\pi + 0.5\pi)n\big) + 10\sin\big((4\pi - 0.7\pi)n\big)$$

$$= 6\cos(0.3\pi n) - 3\sin(0.5\pi n) + 2\cos(0.3\pi n) + 4\cos(0.5\pi n) - 10\sin(0.7\pi n)$$

$$= 8\cos(0.3\pi n) + 5\cos(0.5\pi n + 0.6435) - 10\sin(0.7\pi n)$$

The Sampling Process

$$v[n] = 8\cos(0.3\pi n) + 5\cos(0.5\pi n + 0.6435) - 10\sin(0.7\pi n)$$

- Note: $v[n]$ is composed of 3 discrete-time sinusoidal signals of normalized angular frequencies: 0.3π, 0.5π, and 0.7π
- Note: An identical discrete-time signal is also generated by uniformly sampling at a 200-Hz sampling rate the following continuous-time signals:

$$w_a(t) = 8\cos(60\pi t) + 5\cos(100\pi t + 0.6435) - 10\sin(140\pi t)$$

$$g_a(t) = 2\cos(60\pi t) + 4\cos(100\pi t) + 10\sin(260\pi t) + 6\cos(460\pi t) + 3\sin(700\pi t)$$

The Sampling Process

Recall $\omega_o = \dfrac{2\pi\Omega_o}{\Omega_T}$

$\Omega_T = 2\pi F_T$ sampling angular frequency

Ω_o normalized analog angular frequency

Thus if $\Omega_T > 2\Omega_o$, then the corresponding normalized digital angular frequency ω_o of the discrete-time signal obtained by sampling the parent continuous-time sinusoidal signal will be in the range $-\pi < \omega < \pi$ $\Rightarrow$ No aliasing

The Sampling Process

On the other hand, if $\Omega_T < 2\Omega_o$, the normalized digital angular frequency will foldover into a lower digital frequency $\omega_o = \langle 2\pi\Omega_o / \Omega_T \rangle_{2\pi}$ in the range $-\pi < \omega < \pi$ because of aliasing

Hence, to prevent aliasing, the sampling frequency Ω_T should be greater than 2 times the frequency Ω_o of the sinusoidal signal being sampled

The Sampling Process

Generalization: Consider an arbitrary continuous-time signal $x_a(t)$ composed of a weighted sum of a number of sinusoidal signals

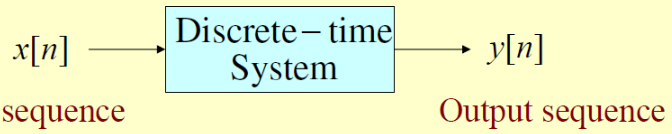
$x_a(t)$ can be represented uniquely by its sampled version $\{x[n]\}$ if the sampling frequency Ω_T is chosen to be greater than 2 times the highest frequency contained in $x_a(t)$

- The condition to be satisfied by the sampling frequency to prevent aliasing is called the **sampling theorem**

Discrete-Time Systems

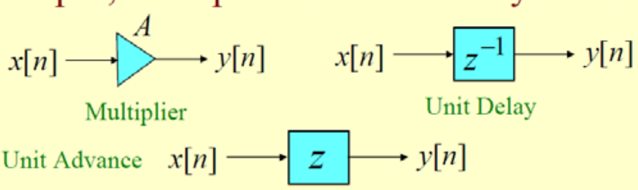
A discrete-time system processes a given **input sequence** $x[n]$ to generates an **output sequence** $y[n]$ with more desirable properties

In most applications, the discrete-time system is a single-input, single-output system:

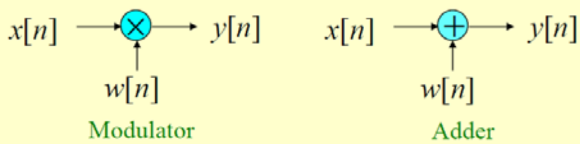


Discrete-Time Systems

- 1-input, 1-output discrete-time systems -



- 2-input, 1-output discrete-time systems -



Discrete-Time Systems

Example – **Accumulator**

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

- The output $y[n]$ at time instant n is the sum of the input sample $x[n]$ at time instant n and the previous output $y[n-1]$ at time instant $n-1$, which is the sum of all previous input sample values from $-\infty$ to $n-1$
- The system cumulatively adds, i.e., it accumulates all input sample values

Discrete-Time Systems

Accumulator

Input-output relation can also be written in the form

$$y[n] = \sum_{\ell=-\infty}^{-1} x[\ell] + \sum_{\ell=0}^{n} x[\ell]$$
$$= y[-1] + \sum_{\ell=0}^{n} x[\ell], \; n \geq 0$$

- The second form is used for a causal input sequence, in which case $y[-1]$ is called the **initial condition**

Discrete-Time Systems

M-point moving-average system

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

- Used in smoothing random variations in data
- In most applications, the data $x[n]$ is a bounded sequence

→ M-point average $y[n]$ is also a bounded sequence

Discrete-Time Systems

M-point moving-average system

- If there is no bias in the measurements, an improved estimate of the noisy data is obtained by simply increasing M
- A direct implementation of the M-point moving average system requires $M-1$ additions, 1 division, and storage of $M-1$ past input data samples
- A more efficient implementation is developed next

Discrete-Time Systems

$$y[n] = \frac{1}{M}\left(\sum_{\ell=0}^{M-1} x[n-\ell] + x[n-M] - x[n-M]\right)$$

$$= \frac{1}{M}\left(\sum_{\ell=1}^{M} x[n-\ell] + x[n] - x[n-M]\right)$$

$$= \frac{1}{M}\left(\sum_{\ell=0}^{M-1} x[n-1-\ell] + x[n] - x[n-M]\right)$$

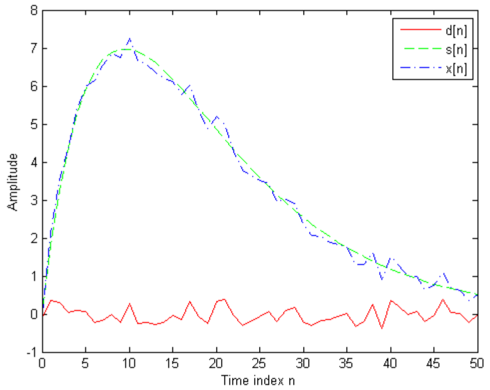
Hence $y[n] = y[n-1] + \frac{1}{M}(x[n] - x[n-M])$

Computation of the modified M-point moving average system using the recursive equation now requires 2 additions and 1 division

Discrete-Time Systems

Consider x[n] = s[n] + d[n]

where x[n] is the signal corrupted by noise d[n]



Example:

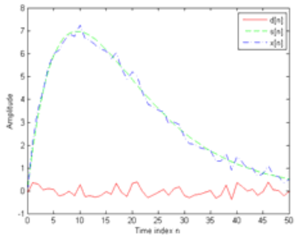
$s[n] = 2n(0.9)^n$

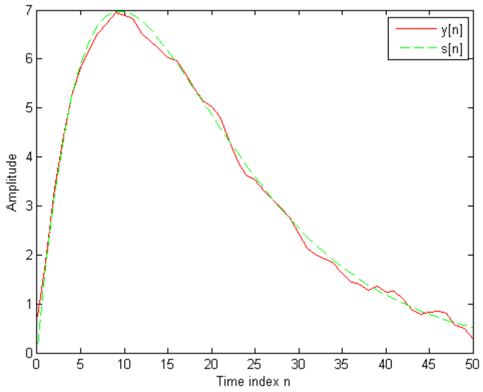
d[n] – random signal (unif. dist. [-0.4,0.4])

Discrete-Time Systems

x[n] = s[n] + d[n]

y[n] filtered x[n]





Example:

$s[n] = 2n(0.9)^n$

d[n] – random signal

(unif. dist. [-0.4,0.4])

Discrete-Time Systems

Exponentially Weighted Running Average Filter

$$y[n] = \alpha y[n-1] + x[n], \quad 0 < \alpha < 1$$

- Computation of the running average requires only 2 additions, 1 multiplication and storage of the previous running average
- Does not require storage of past input data samples

Discrete-Time Systems

$$y[n] = \alpha y[n-1] + x[n], \quad 0 < \alpha < 1$$

For $0 < \alpha < 1$, the exponentially weighted average filter places more emphasis on current data samples and less emphasis on past data samples

$$y[n] = \alpha(\alpha y[n-2] + x[n-1]) + x[n]$$
$$= \alpha^2 y[n-2] + \alpha x[n-1] + x[n]$$
$$= \alpha^2(\alpha y[n-3] + x[n-2]) + \alpha x[n-1] + x[n]$$
$$= \alpha^3 y[n-3] + \alpha^2 x[n-2] + \alpha x[n-1] + x[n]$$

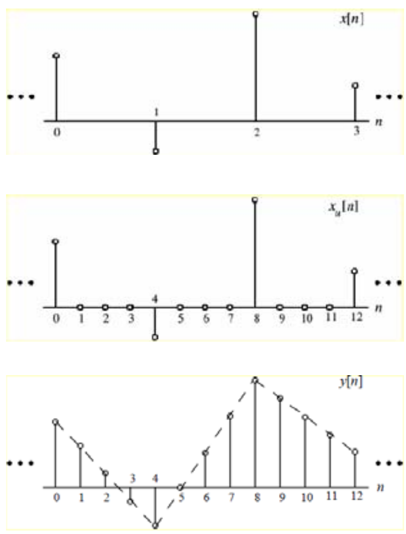
Discrete-Time Systems

Linear Interpolation

Employed to estimate sample values between pairs of adjacent sample values of a discrete-time sequence.

Example:

Factor-of-4 interpolation



Discrete-Time Systems

- **Factor-of-2 interpolator** -

$$y[n] = x_u[n] + \tfrac{1}{2}(x_u[n-1] + x_u[n+1])$$

- **Factor-of-3 interpolator** -

$$y[n] = x_u[n] + \tfrac{1}{3}(x_u[n-1] + x_u[n+1]) + \tfrac{2}{3}(x_u[n-2] + x_u[n+2])$$

Example: Original 512x512 image (top)
Down-sampled 256x256 image (middle)
Linear interpolated image (bottom)

Discrete-Time Systems — Median Filter

- The median of a set of $(2K+1)$ numbers is the number such that K numbers from the set have values greater than this number and the other K numbers have values smaller
- Median can be determined by rank-ordering the numbers in the set by their values and choosing the number at the middle

Example: Consider the set of numbers {2, -3, 10, 5, -1}

Rank-ordered set is given by {-3, -1, 2, 5, 10}

Hence, med {2, -3, 10, 5, -1} = 2

Discrete-Time Systems Median Filter

- Implemented by sliding a window of odd length over the input sequence $\{x[n]\}$ one sample at a time
- Output $y[n]$ at instant n is the median value of the samples inside the window centered at n
- Finds applications in removing additive random noise, which shows up as sudden large errors in the corrupted signal
- Usually used for the smoothing of signals corrupted by impulse noise

Discrete-Time Systems Median Filter

Examples: