Signals and Systems
E-623

Lecture 4

- Signals Simulation and Plotting
- Systems Properties

د. باسم ممدوح الحلوانى

Digital Signal Processing Using MATLAB
Third Edition
Vinay K. Ingle
John G. Proakis

CHAPTER 1

Introduction

Vector Manipulation and Various signals plotting techniques.

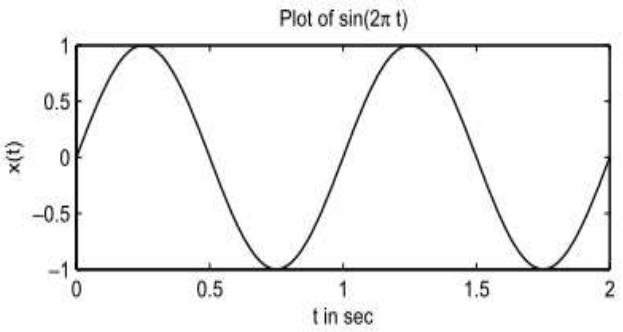
Vector Manipulation and Various signals plotting techniques.

- ✓ MATLAB provides several types of plots, starting with simple two-dimensional (2D) graphs to complex, higher-dimensional plots with full-color capability.
- ✓ The 2D plotting is plotting of one vector versus another in a 2D coordinates

1. CT plotting

- ✓ The basic command is the plot(t,x) command, which generates a plot of x values versus t
- ✓ The arrays t and x should be the same length and orientation.
- ✓ The following set of commands creates a list of sample points, evaluates the sine function at those points, and then generates a plot of a simple sinusoidal wave

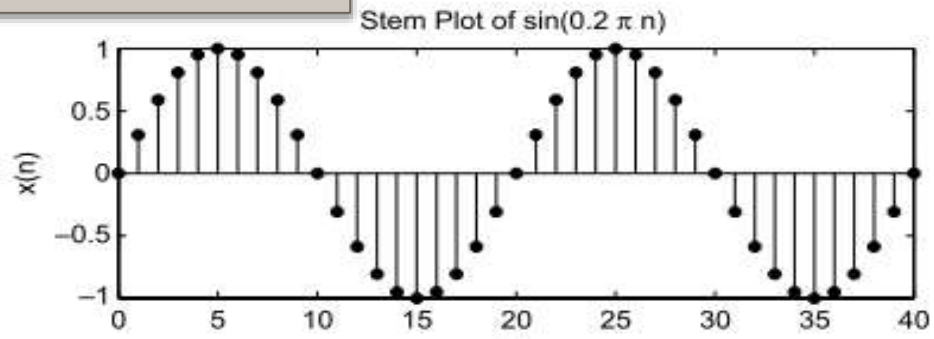
```
>> t = 0:0.01:2;   % sample points from 0 to 2 in steps of 0.01
>> xt = sin(2*pi*t);   % Evaluate sin(2 pi t)
>> plot(t,xt,'b');   % Create plot with blue line
>> xlabel('t in sec'); ylabel('x(t)');    % Label axis
>> title('Plot of sin(2\pi t)');   % Title plot
```



2. DT plotting

- ✓ For plotting a set of discrete numbers (or discrete-time signals), we will use the **stem command** which displays data values as a stem, that is, a small circle at the end of a line connecting it to the horizontal axis.
- ✓ The circle can be open (default) or filled (using the option 'filled').
- ✓ Using Handle Graphics, we can resize circle markers.
- ✓ The following set of commands displays a discrete-time sine function using these constructs.

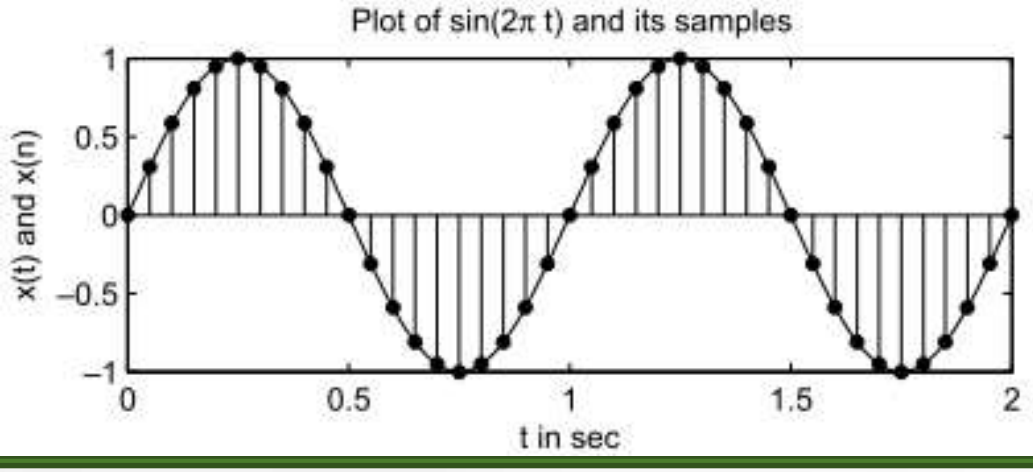
```
>> n = 0:1:40; % sample index from 0 to 20
>> xn = sin(0.1*pi*n); % Evaluate sin(0.2 pi n)
>> Hs = stem(n,xn,'b','filled'); % Stem-plot with handle Hs
>> set(Hs,'markersize',4); % Change circle size
>> xlabel('n'); ylabel('x(n)'); % Label axis
>> title('Stem Plot of sin(0.2 pi n)'); % Title plot
```



3. Plotting more than signal together (Using "Hold on" command)

- ✓ MATLAB provides an ability to display more than one graph in the same figure window.
- ✓ By means of the hold on command, several graphs can be plotted on the same set of axes.
- ✓ The hold off command stops the simultaneous plotting.

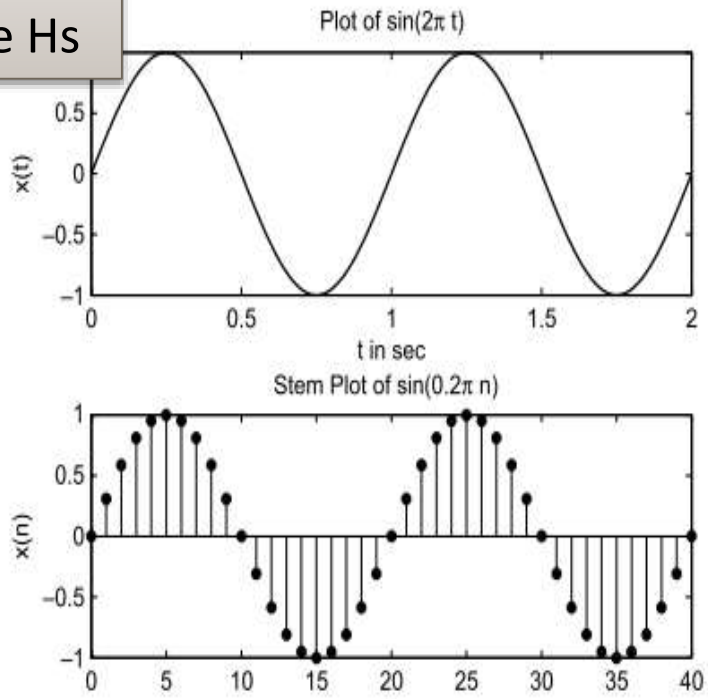
```
>>plot(t,xt,'b'); hold on; % Create plot with blue line
>>Hs = stem(n*0.05,xn,'b','filled'); % Stem-plot with handle Hs
>>set(Hs,'markersize',4); hold off; % Change circle size
```



3. Plotting more than signal together (Using "Subplot" command)

- ✓ Displays several graphs in each individual set of axes arranged in a grid, using the parameters in the subplot command.
- ✓ The following fragment (Figure 1.4) displays graphs in Figure 1.1 and 1.2 as two separate plots in two rows.

```
>> subplot(2,1,1); % Two rows, one column, first plot
>> plot(t,xt,'b'); % Create plot with blue line
>> subplot(2,1,2); % Two rows, one column, second plot
>> Hs = stem(n,xn,'b','filled'); % Stem-plot with handle Hs
```



Digital Signal Processing Using MATLAB
Third Edition
Vinay K. Ingle
John G. Proakis

CHAPTER 2

Discrete-Time Signals and Systems

$$x(n) = \{x(n)\} = \{\ldots, x(-1), \underset{\uparrow}{x(0)}, x(1), \ldots\}$$

- ✓ In MATLAB we can represent a finite-duration sequence by a row vector of appropriate values.
- ✓ However, such a vector does not have any information about sample position n.
- ✓ Therefore a correct representation of x(n) would require two vectors, one each for x and n.

$$x(n) = \{2, 1, -1, \underset{\uparrow}{0}, 1, 4, 3, 7\}$$

```
>> n=[-3,-2,-1,0,1,2,3,4];  x=[2,1,-1,0,1,4,3,7];
```

- ✓ Generally, we will use the x-vector representation alone when the sample position information is not required or when such information is trivial (begin with zero)

2.1.1 TYPES OF SEQUENCES

1. **Unit sample sequence:**

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} = \left\{ \ldots, 0, 0, \underset{\uparrow}{1}, 0, 0, \ldots \right\}$$

For example, to implement

$$\delta(n - n_0) = \begin{cases} 1, & n = n_0 \\ 0, & n \neq n_0 \end{cases}$$

over the $n_1 \leq n_0 \leq n_2$ interval, we will use the following MATLAB function.

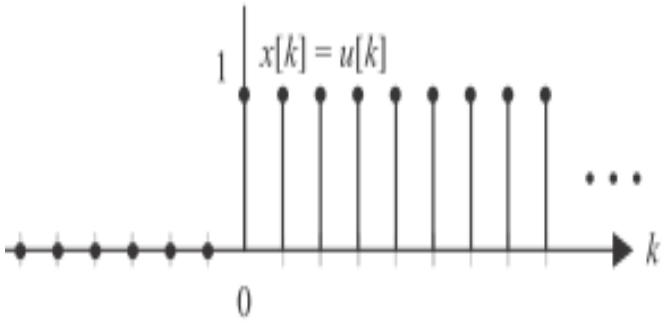
```
function [x,n] = impseq(n0,n1,n2)
  %%  Generates x(n) = delta(n-n0); n1 <= n <= n2
  %  How to call
  %  [x,n] = impseq(5,1,10)

      n = [n1:n2];
      x = [(n-n0) == 0];
```

- We can use the zeros(1,N) built-in function
- We can use the logical relation n==0

2. Unit step sequence:

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases} = \{\ldots, 0, 0, \underset{\uparrow}{1}, 1, 1, \ldots\}$$



How to implement?

$$u(n - n_0) = \begin{cases} 1, & n \geq n_0 \\ 0, & n < n_0 \end{cases}$$

over the $n_1 \leq n_0 \leq n_2$ interval, we will use the following MATLAB function.

```
function [x,n] = stepseq(n0,n1,n2)
  %% Generates x(n) = u(n-n0); n1 <= n <= n2
  %  Call:
   % [x,n] = stepseq(n0,n1,n2)

   n = [n1:n2];
   x = [(n-n0) >= 0];
```

- We can use the ones(1,N) built-in function
- We can use the logical relation n>=0

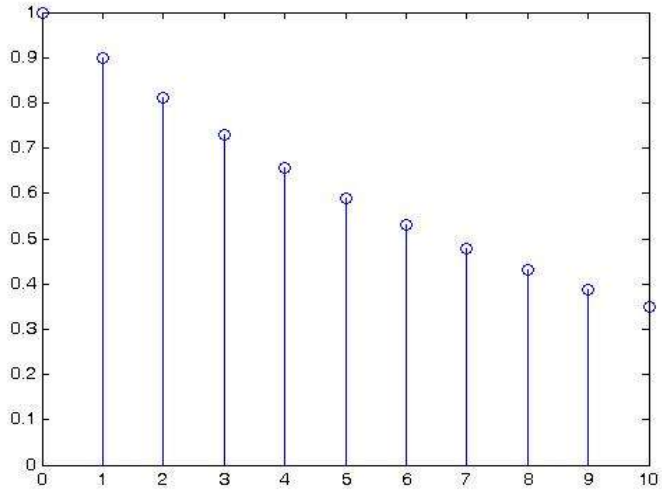
2.1.1 TYPES OF SEQUENCES

3. Real-valued exponential sequence:

$$x(n) = a^n, \forall n; \quad a \in \mathbb{R}$$

```
>> n = [0:10];  x = (0.9).^n;
```

- If a <1 , attenuation
- If a>1, amplification



4. Complex-valued exponential sequence:

$$x(n) = e^{(\sigma + j\omega_0)n}, \forall n$$

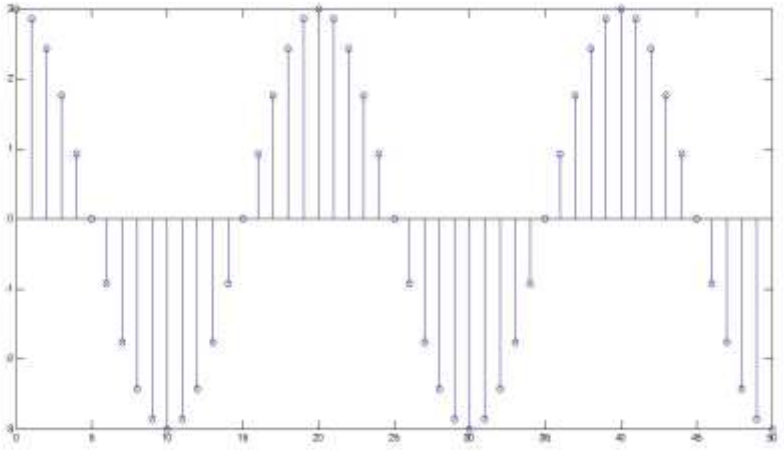
```
>> n = [0:10]; x = exp((2+3j)*n);
>> stem(n,abs(x));
>> figure;
>> stem(n,angle(x));
```

2.1.1 TYPES OF SEQUENCES

5. Sinusoidal sequence:

$$x(n) = A\cos(\omega_0 n + \theta_0), \forall n$$

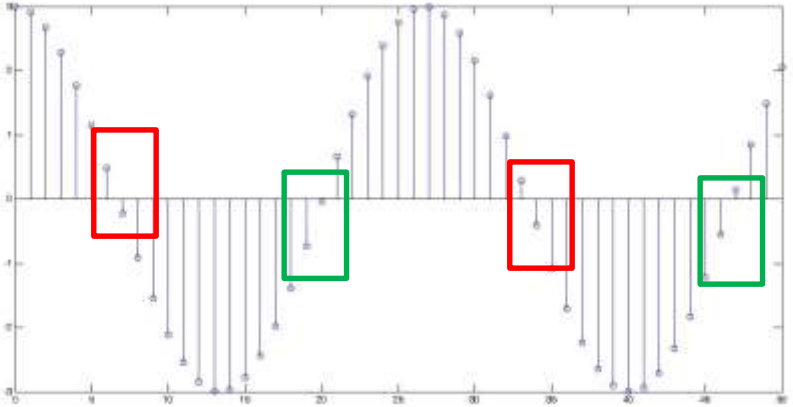
```
>> n = [0:100]; x = 3*cos(0.1*pi*n) % periodic
>> n2 = [0:50] ; x = 3*cos(0.234*n) % aperiodic
```



periodic

$$\frac{\Omega_0}{2\pi} = \frac{m}{K_0}$$

aperiodic



2.1.1 TYPES OF SEQUENCES

7. Periodic sequence:

To generate P periods of $\tilde{x}(n)$ from one period $\{x(n), \quad 0 \leq n \leq N-1\}$, we can copy $x(n)$ P times:

```
>> xtilde = [x,x,...,x];
```

- ✓ But an elegant approach is to use MATLAB's powerful indexing capabilities.
- ✓ First we generate a matrix containing P rows of x(n) values.
- ✓ Then we can concatenate P rows into a long row vector using the construct (:).
- ✓ However, this construct works only on columns. Hence we will have to use the matrix transposition operator '

```
>> xtilde = x' * ones(1,P); % P columns of x; x is a row vector
>> xtilde = xtilde(:);      % long column vector
>> xtilde = xtilde';        % long row vector
```

```
1  1  1  1
2  2  2  2
3  3  3  3
```

```
1
2
3
1
2
3
1
2
3
1
2
3
```

1 2 3 1 2 3 1 2 3 1 2 3

Systems Properties

Linear Systems (1)

- A **linear system** exhibits the **additivity** property:

$$x_1 \longrightarrow y_1 \quad x_2 \longrightarrow y_2 \qquad x_1 + x_2 \longrightarrow y_1 + y_2$$

- It also must satisfy the **homogeneity** or **scaling** property:

$$x \longrightarrow y \qquad kx \longrightarrow ky$$

- These can be combined into the property of **superposition**:

$$x_1 \longrightarrow y_1 \quad x_2 \longrightarrow y_2 \qquad k_1x_1 + k_2x_2 \longrightarrow k_1y_1 + k_2y_2$$

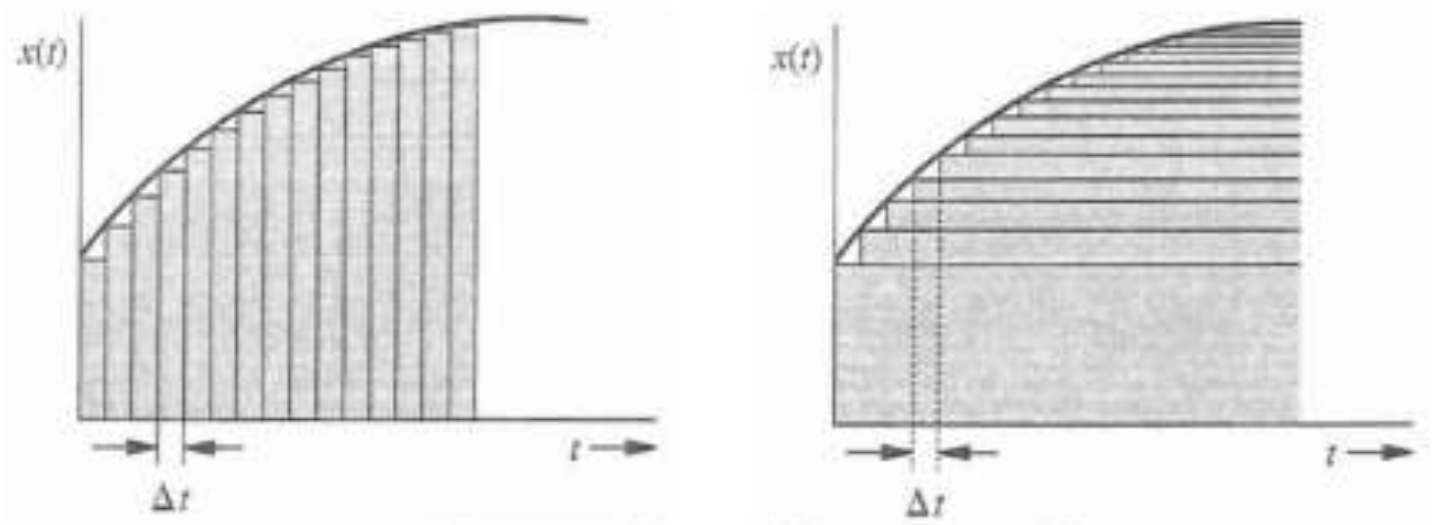
- A non-linear system is one that is NOT linear (i.e. does not obey the principle of superposition)

LINEAR AND NONLINEAR SYSTEMS

- Many systems are nonlinear. For example: many circuit elements (e.g., diodes), dynamics of aircraft, econometric models,…
- However, we focus exclusively on **linear** systems.
- Why?
  - Linear models represent accurate representations of behavior of many systems (e.g., linear resistors, capacitors, other examples given previously,…)
  - Can often linearize models to examine "small signal" perturbations around "operating points"
  - Linear systems are analytically tractable, providing basis for important tools and considerable insight

Linear Systems (5)

- Almost all systems become **nonlinear** when large enough signals are applied
- Nonlinear systems can be **approximated** by linear systems for **small-signal analysis** – greatly simply the problem
- Once superposition applies, analyse system by decomposition into zero-input and zero -state components
- Equally important, we can represent $x(t)$ as a sum of simpler functions (pulse or step)



$$x(t) = a_1 x_1(t) + a_2 x_2(t) + \cdots + a_m x_m(t)$$

$$y(t) = a_1 y_1(t) + a_2 y_2(t) + \cdots + a_m y_m(t)$$

L1.7-1 p

TIME-INVARIANCE (TI)

Informally, a system is time-invariant (TI) if its behavior does not depend on what time it is.

- Mathematically (in DT): A system $x[n] \rightarrow y[n]$ is TI if for any input $x[n]$ and any time shift n_0,

  If $x[n] \rightarrow y[n]$

  then $x[n - n_0] \rightarrow y[n - n_0]$.

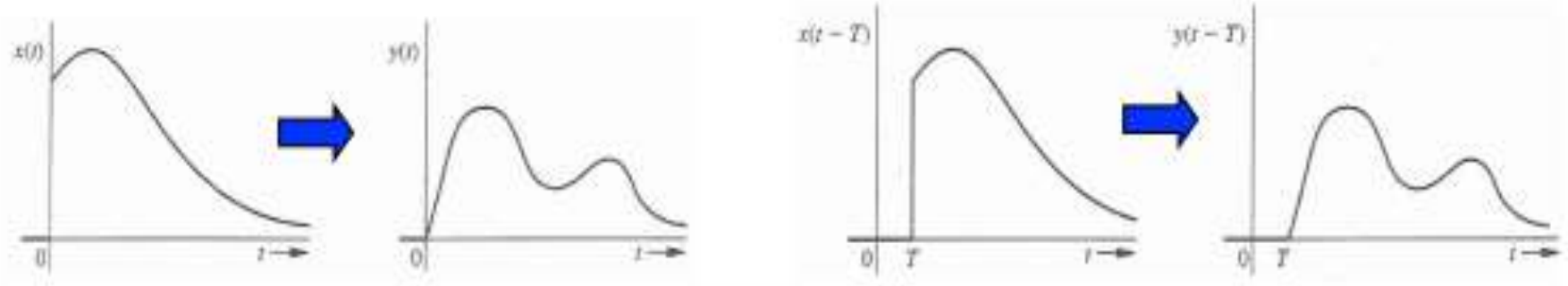
- Similarly for a CT time-invariant system,

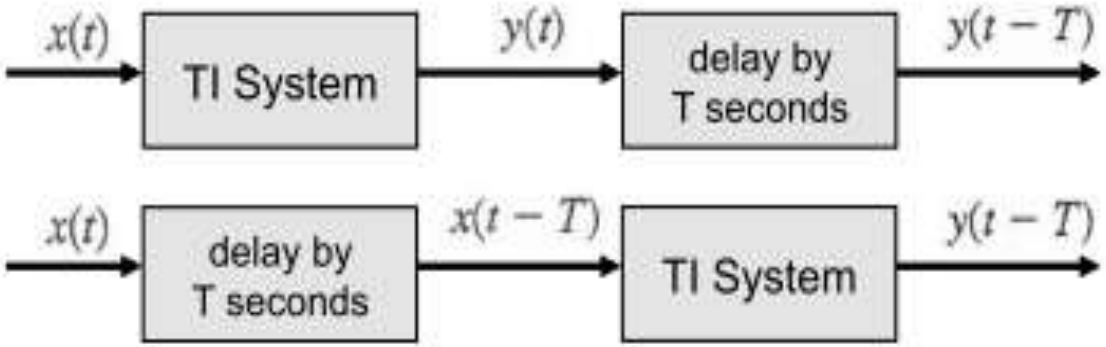
  If $x(t) \rightarrow y(t)$

  then $x(t - t_o) \rightarrow y(t - t_o)$.

Time-Invariant Systems

- **Time-invariant system** is one whose parameters do not change with time:





CAUSALITY

- A system is causal if the output does not anticipate future values of the input, i.e., if the output at any time depends only on values of the input up to that time.
- All real-time physical systems are causal, because time only moves forward. Effect occurs after cause. (Imagine if you own a noncausal system whose output depends on tomorrow's stock price.)
- Causality does not apply to spatially varying signals. (We can move both left and right, up and down.)
- Causality does not apply to systems processing recorded signals, e.g. taped sports games vs. live broadcast.

CAUSAL OR NONCAUSAL

$y(t) = x^2(t-1)$

E.g. $y(5)$ depends on $x(4)$ ... causal

$y(t) = x(t+1)$

E.g. $y(5) = x(6), y$ depends on future $\Rightarrow$ noncausal

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

E.g. $y[5] = x[-5]$ ok, but

$y[-5] = x[5], y$ depends on future $\Rightarrow$ noncausal

$y[n] = \left(\frac{1}{2}\right)^{n+1} x^3[n-1]$

E.g. $y[5]$ depends on $x[4]$ ... causal

Invertible and Noninvertible Systems

- Let a system S produces $y(t)$ with input $x(t)$, if there exists another system S_i, which produces $x(t)$ from $y(t)$, then S is invertible
- Essential that there is **one-to-one mapping** between input and output
- For example if S is an amplifier with gain G, it is invertible and S_i is an attenuator with gain 1/G
- Apply S_i following S gives an **identity system** (i.e. input $x(t)$ is not changed)