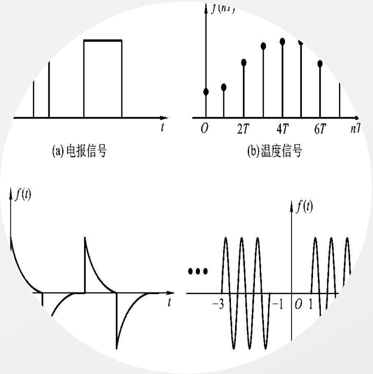
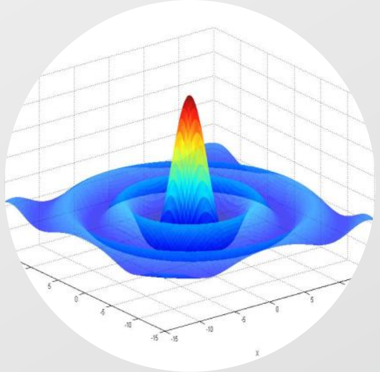
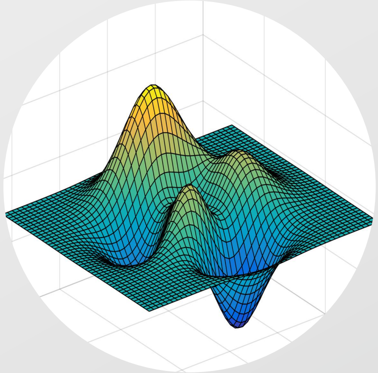
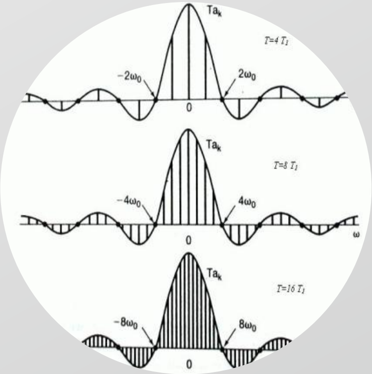




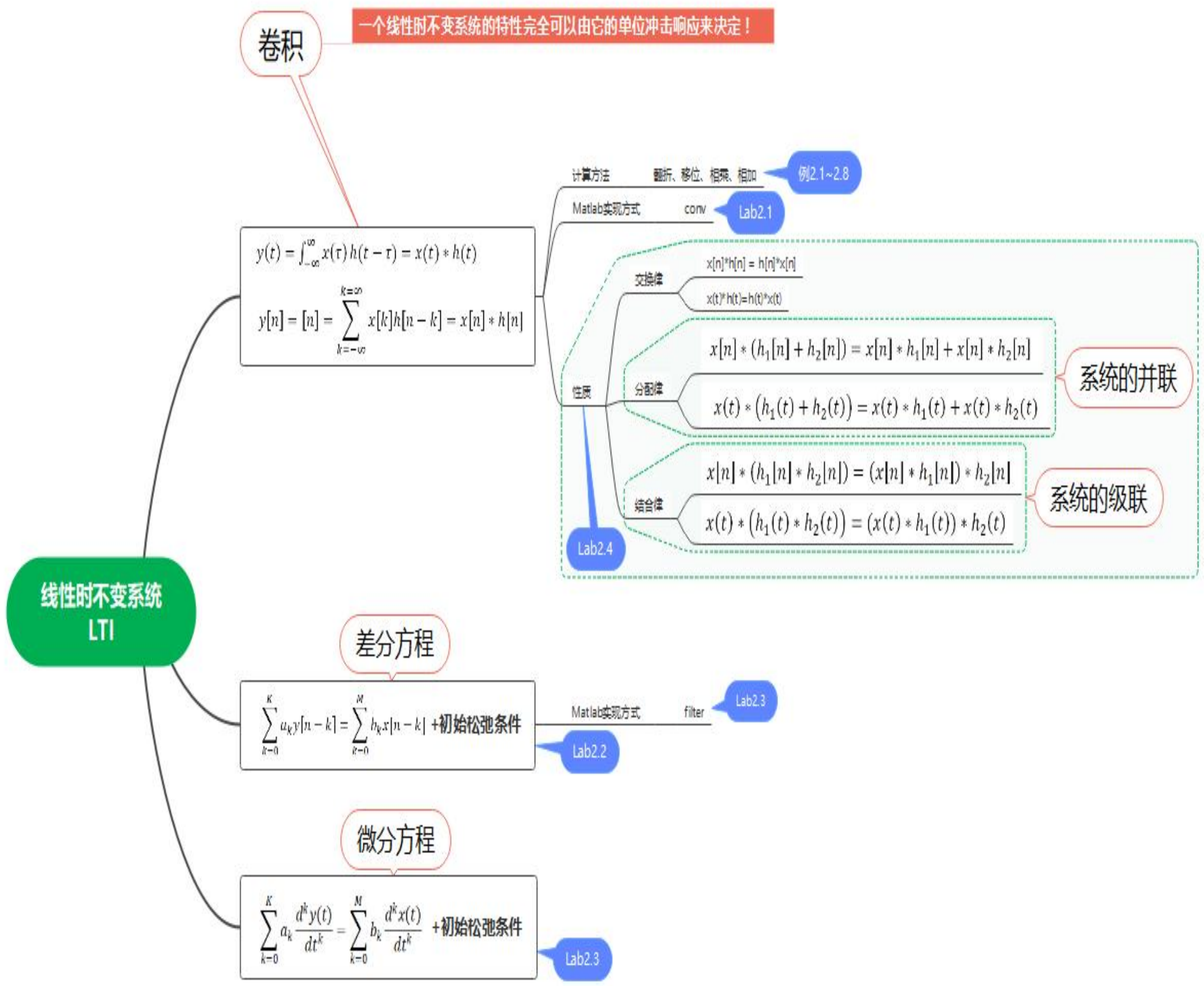




Lab2 Linear Time-Invariant Systems

Introducer : Wang Xiaojing

Objectives



- 1. Verify the property of convolution
- 2. Verify the property of LTI systems
- 3. Design a LTI system for echo cancellation

Convolution-Sum

Arbitrary DT signal can be written as a linear combination of impulse functions with different time shifting

$$x[n] = \sum_{k=-\infty}^{+\infty} x[k]\delta[n-k]$$

LTI System

$$x[n] = x[n] * \delta[n]$$

Convolution-Sum

$$y[n] = x[n] * h[n]$$

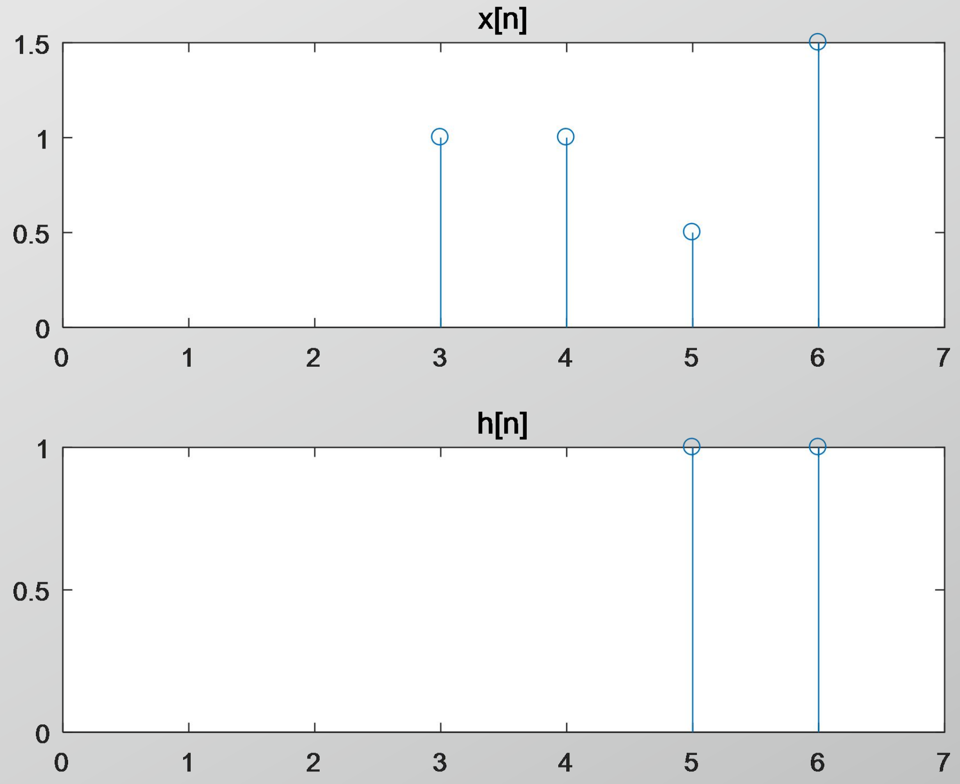
$h[n]$: the output as unit impulse response for unit impulse input $\delta[n]$ $$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

The output of any DT LTI system is a convolution of the input signal with the unit impulse response.

➢Convolution——Flip,Slide, Multiply, Sum

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

```
x=[1 1 0.5 1.5];
nx=3:6;
h=[1 1];
nh=5:6;
figure
subplot(2,1,1),
stem(nx,x),xlim([0 7]),title('x[n]')
subplot(2,1,2),
stem(nh,h),xlim([0 7]),title('h[n]')
```



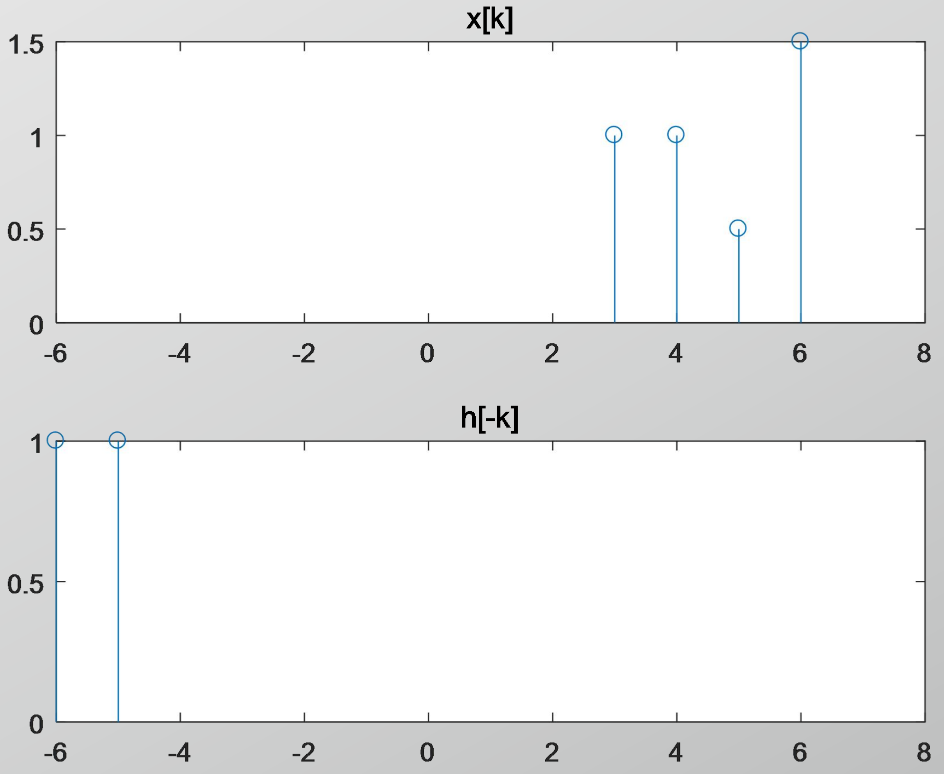
Flip

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

Flip

```
figure
subplot(2,1,1),
stem(nx,x),xlim([-6 8]),title('x[k]')
subplot(2,1,2),
stem(-nh,h),xlim([-6 8]),title('h[-k]')
```

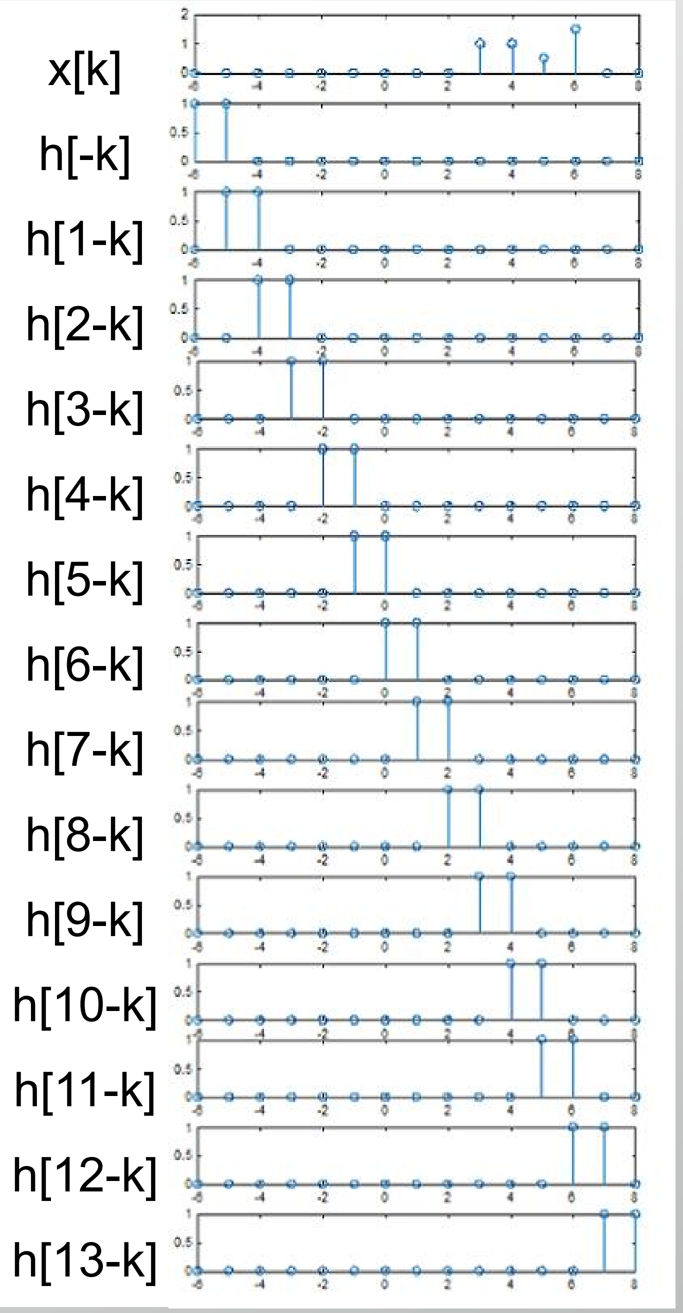
Flip



Slide

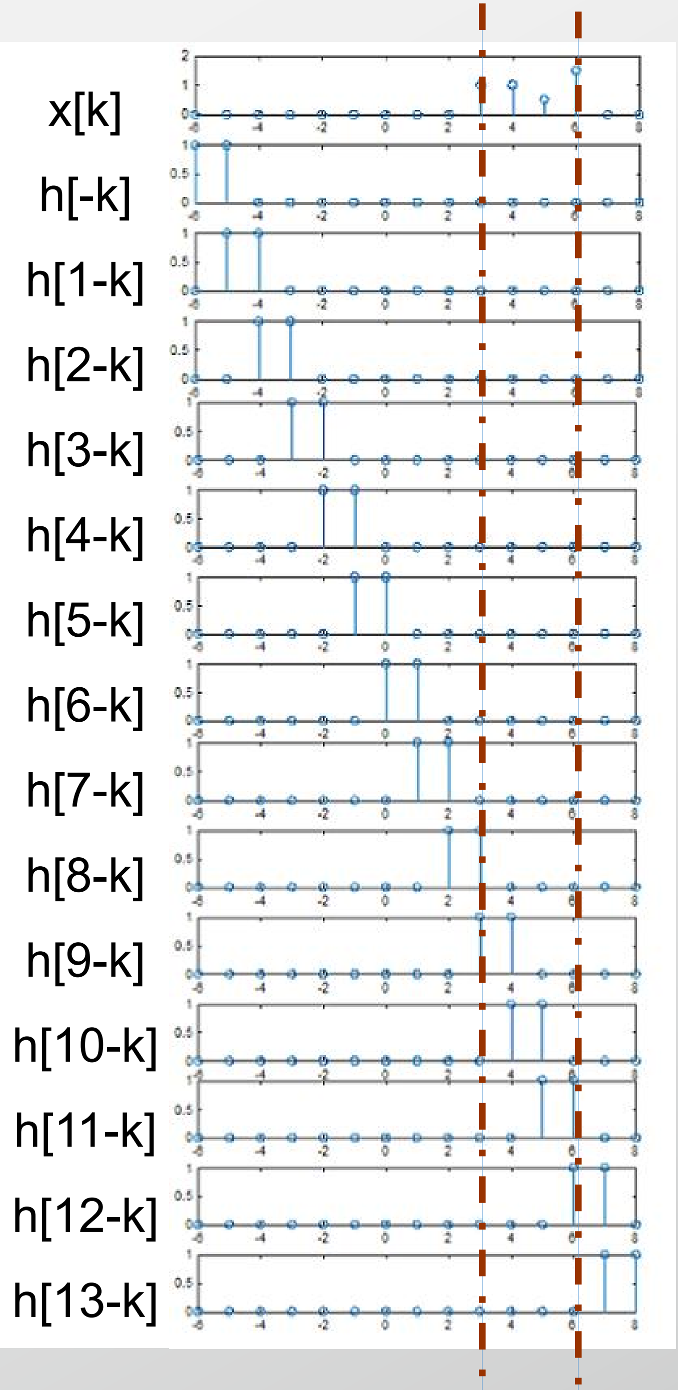
$$y[n]=\sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

```
figure
%subplot(14,1,1),stem(nx,x),xlim([-6 8]),title('x[k]')
subplot(14,1,2),stem(-nh+1,h),xlim([-6 8]),title('h[1-k]')
subplot(14,1,3),stem(-nh+2,h),xlim([-6 8]),title('h[2-k]')
subplot(14,1,4),stem(-nh+3,h),xlim([-6 8]),title('h[3-k]')
subplot(14,1,5),stem(-nh+4,h),xlim([-6 8]),title('h[4-k]')
subplot(14,1,6),stem(-nh+5,h),xlim([-6 8]),title('h[5-k]')
subplot(14,1,7),stem(-nh+6,h),xlim([-6 8]),title('h[6-k]')
subplot(14,1,8),stem(-nh+7,h),xlim([-6 8]),title('h[7-k]')
subplot(14,1,9),stem(-nh+8,h),xlim([-6 8]),title('h[8-k]')
subplot(14,1,10),stem(-nh+9,h),xlim([-6 8]),title('h[9-k]')
subplot(14,1,11),stem(-nh+10,h),xlim([-6 8]),title('h[10-k]')
subplot(14,1,12),stem(-nh+11,h),xlim([-6 8]),title('h[11-k]')
subplot(14,1,13),stem(-nh+12,h),xlim([-6 8]),title('h[12-k]')
subplot(14,1,14),stem(-nh+13,h),xlim([-6 8]),title('h[13-k]')
```



Multiply, Sum

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$



Observing the figure, we only need to calculate y[n] when n is between 8 and 12.

y(8)=1*1=1;

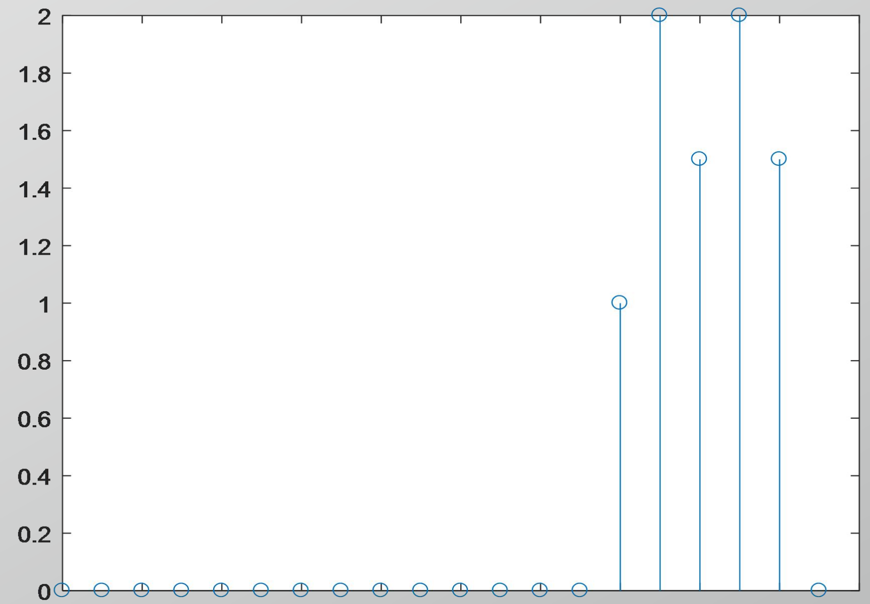
y(9)=1*1+1*1=2;

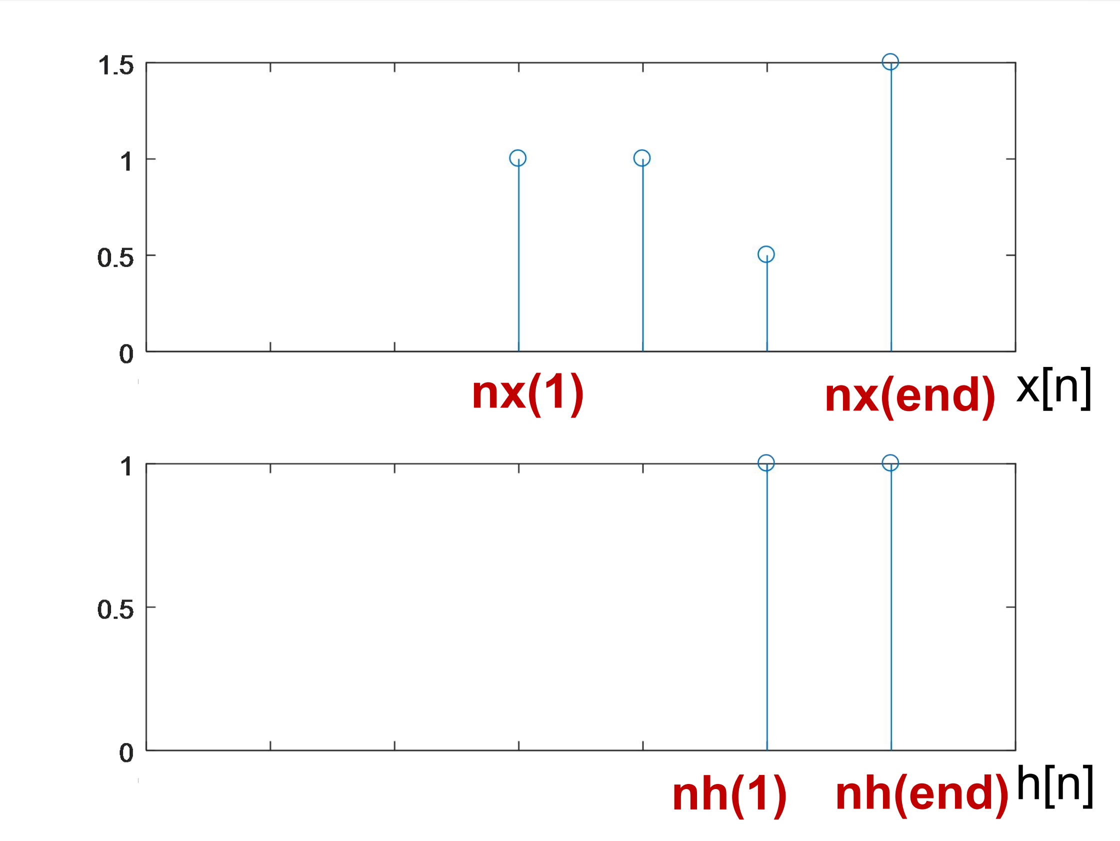
y(10)=1*1+1*0.5=1.5;

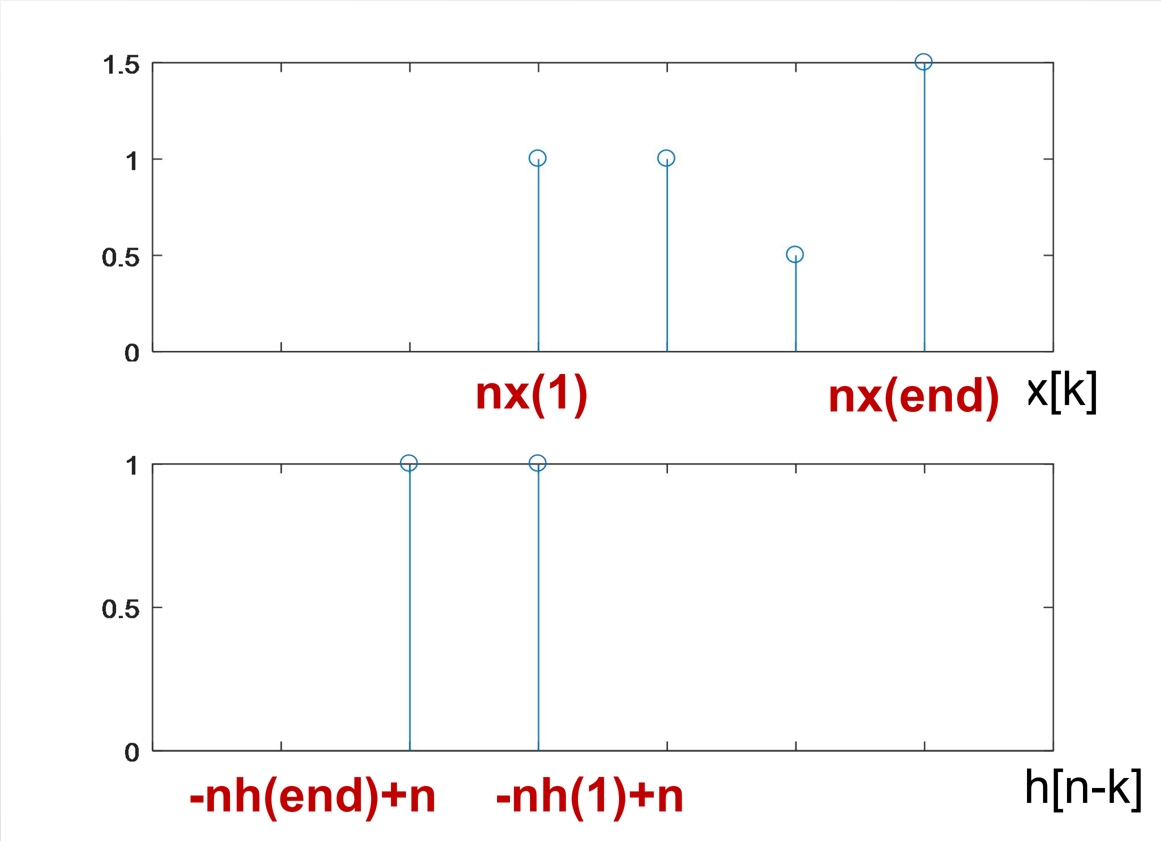
y(11)=1*0.5+1*1.5=2;

y(12)=1*1.5=1.5

otherwise, y(n) is always 0；







$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

Non-zero interval of y[n]: **[ny(1) ny(end)]**

ny(1): when **nx(1)=-nh(1)+n ， n=nx(1)+nh(1);**

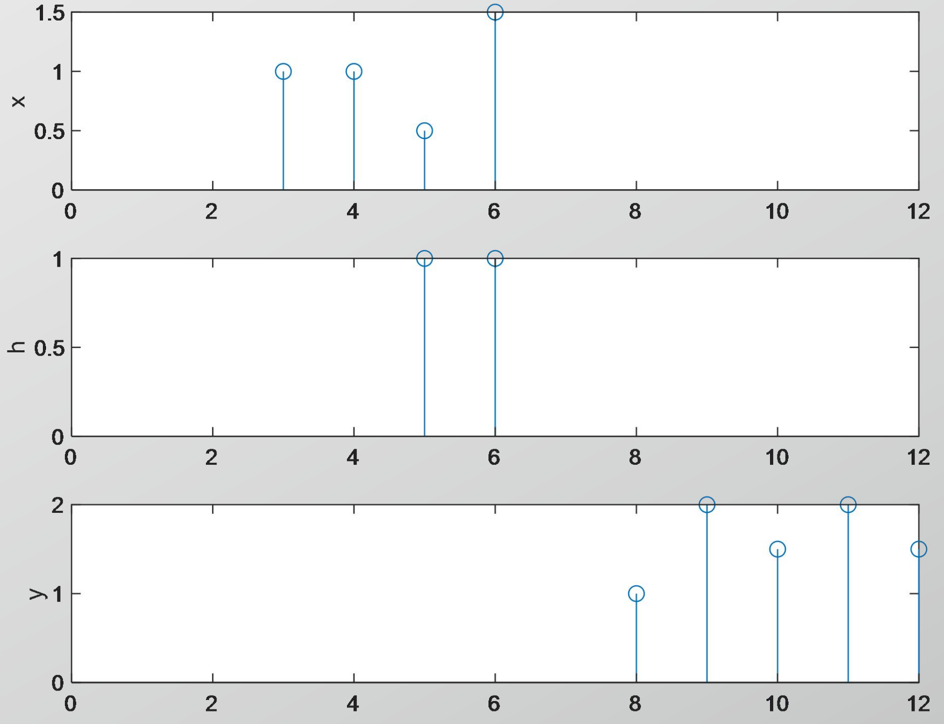
ny(end): when **nx(end)=-nh(end)+n , n= nx(end)+nh(end);**

Part I：Calculating Convolution via Matlab

Convolution——matlab function:conv

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

```
x=[1 1 0.5 1.5];
nx=3:6;
h=[1 1];
nh=5:6;
y=conv(x,h);
ny=nx(1)+nh(1):nx(end)+nh(end);
subplot(3,1,1),stem(nx,x),ylabel('x');
xlim([0 nx(end)+nh(end)]);
subplot(3,1,2),stem(nh,h),ylabel('h');
xlim([0 nx(end)+nh(end)]);
subplot(3,1,3),stem(ny,y),ylabel('y');
xlim([0 nx(end)+nh(end)]);
```




```
y =

    1.0000    2.0000    1.5000    2.0000    1.5000
```

■ 2.1 Tutorial: conv

The MATLAB function **conv** computes the convolution sum

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m] \tag{2.3}$$

assuming that $x[n]$ and $h[n]$ are finite-length sequences. If $x[n]$ is nonzero only on the interval $n_x \leq n \leq n_x + N_x - 1$ and $h[n]$ is nonzero only on the interval $n_h \leq n \leq n_h + N_h - 1$, then $y[n]$ can be nonzero only on the interval

$$(n_x + n_h) \leq n \leq (n_x + n_h) + N_x + N_h - 2\,, \tag{2.4}$$

meaning that **conv** need only compute $y[n]$ for the $N_x + N_h - 1$ samples on this interval. If **x** is an N_x–dimensional vector containing $x[n]$ on the interval $n_x \leq n \leq n_x + N_x - 1$ and **h** is an N_h–dimensional vector containing $h[n]$ on the interval $n_h \leq n \leq n_h + N_h - 1$, then **y=conv(h,x)** returns in **y** the $N_x + N_h - 1$ samples of $y[n]$ on the interval in Eq. (2.4). However, **conv** does not return the indices of the samples of $y[n]$ stored in **y**, which makes sense because the intervals of **x** and **h** are not input to **conv**. Instead, you are responsible for keeping track of these indices, and will be shown how to do this in this tutorial.

Part II: Causal DT LTI Systems Described by Difference Equations

Causal DT LTI Systems Described by Difference Equations

Tutorial: Textbook 2.4.2

$$\sum_{k=0}^{K} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]$$

Initial Rest——i.e.,if $x[n]=0$ for $n<n_0$, then $y[n]=0$ for $n<n_0$;

$$\sum_{k=0}^{K} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]$$

+

Initial rest

Causal DT LTI System is uniquely specified by two coefficient vectors:

$A=[a_0,a_1,a_2\ldots , a_K]$, $B=[b_0,b_1,b_2\ldots , b_M]$.

Caculate the output of Causal DT LTI System by

Matlab function filter()

$$\sum_{k=0}^{K} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]$$

+ Initial rest

Syntax: y=filter(B, A, x) ------> doc filter

Coefficients vectors: $A=[a_0,a_1,a_2\ldots, a_K]$,$B=[b_0,b_1,b_2\ldots, b_M]$.

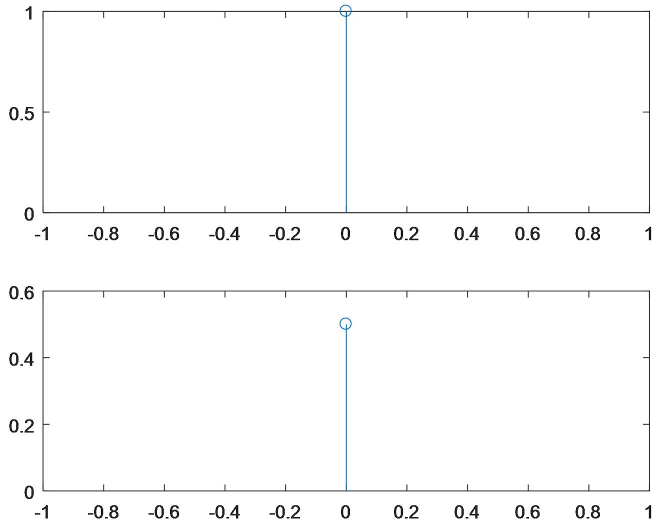
Notes： x and y share the same range of time indices.

——**Output signal y may be truncated.**

System1 :   y[n]=0.5x[n]+x[n-1]+2x[n-2];

```
a=[1];
b=[0.5 1 2];
x=[1];
nx=0;
y=filter(b,a,x)
subplot(2,1,1),stem(nx,x);
subplot(2,1,2),stem(nx,y);
```

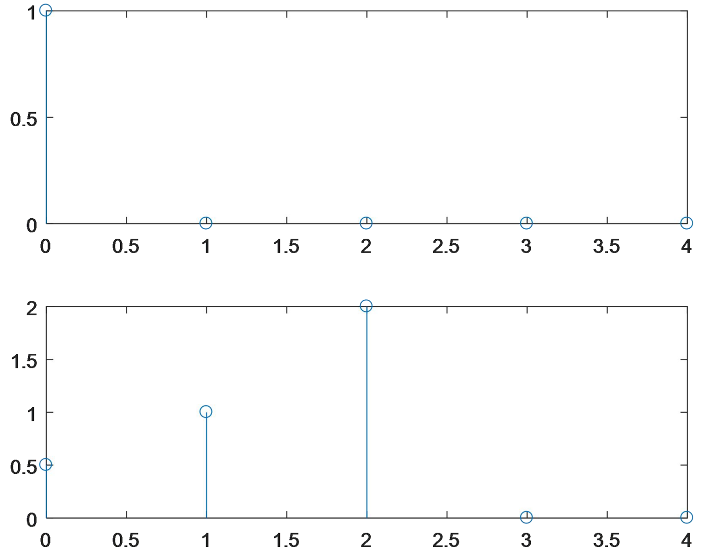
```
y=
     0.5000
```



VS

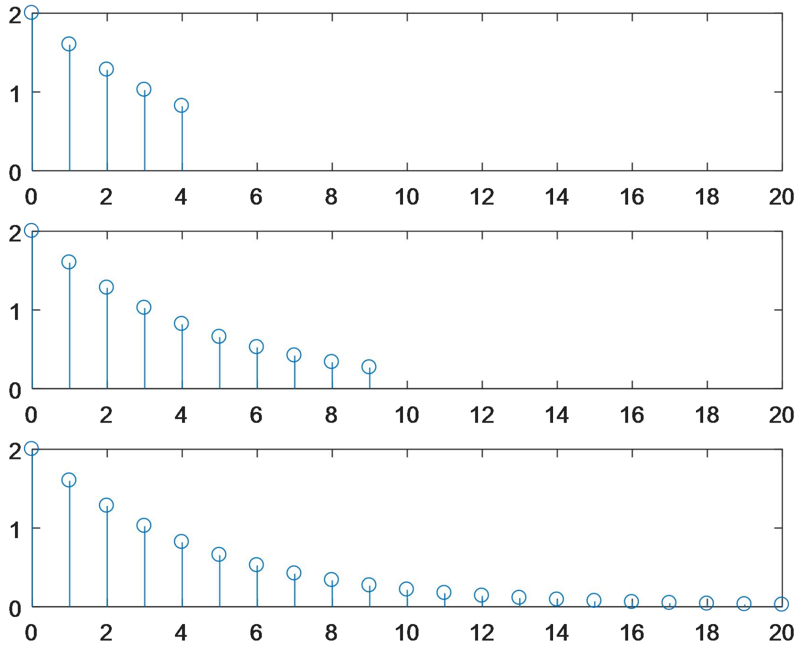
```
a=[1];
b=[0.5 1 2];
x=[1 0 0 0 0];
nx=0:4;
y=filter(b,a,x)
subplot(2,1,1),stem(nx,x);
subplot(2,1,2),stem(nx,y);
```

```
y=
     0.5000  1.0000   2.0000   0   0
```



system2： y[n]-0.8y[n-1]=2x[n];

```
a=[1 -0.8];
b=[2];
x1=[1 0 0 0 0];
nx1=0:4;
y1=filter(b,a,x1)
x2=[1 0 0 0 0 0 0 0 0 0];
nx2=0:9;
y2=filter(b,a,x2)
x3=[1 zeros(1,20)];
nx3=0:20;
y3=filter(b,a,x3)
subplot(3,1,1),stem(nx1,y1),axis([0 20 0 2]);
subplot(3,1,2),stem(nx2,y2),axis([0 20 0 2]);
subplot(3,1,3),stem(nx3,y3),axis([0 20 0 2]);
```




```
y1 =    2.0000    1.6000    1.2800    1.0240    0.8192
y2 =    2.0000    1.6000    1.2800    1.0240    0.8192    0.6554    0.5243    0.4194    0.3355   0.2684
Y3=    2.0000    1.6000    1.2800    1.0240    0.8192    0.6554    0.5243    0.4194    0.3355   0.2684 ……
```

$$\sum_{k=0}^{K} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]$$

+

Initial rest

y[n]=0.5x[n]+x[n-1]+2x[n-2]; h[n]=? ---- nonzero only over a finite time interval

→ Finite Impulse Response **(FIR)**)

y[n]-0.8y[n-1]=2x[n]; h[n]= ? ---- an impulse response of infinite duration

→ Infinite Impulse Response **(IIR)**)

■ 2.2 Tutorial: `filter`

The **`filter`** command computes the output of a causal, LTI system for a given input when the system is specified by a linear constant-coefficient difference equation. Specifically, consider an LTI system satisfying the difference equation

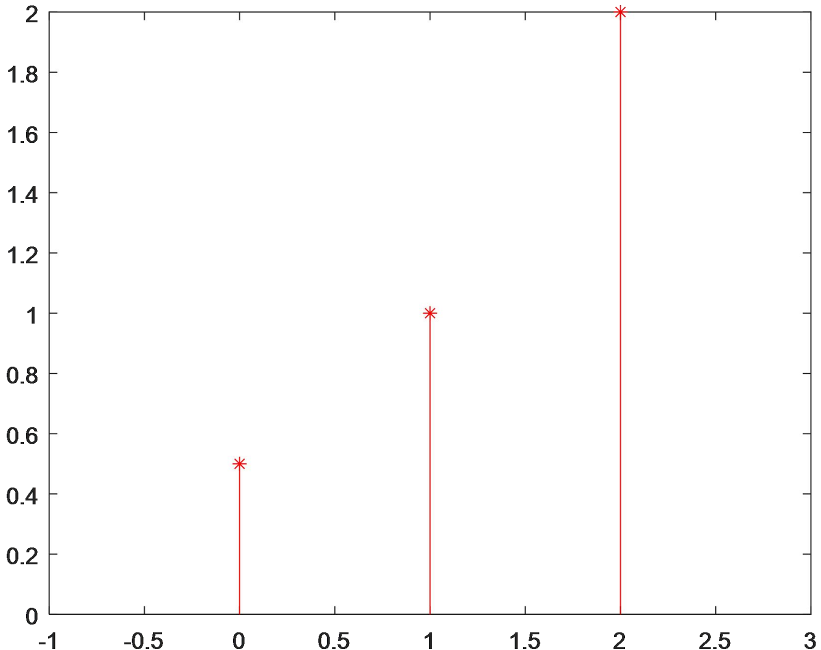
$$\sum_{k=0}^{K} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]\,, \tag{2.7}$$

Filter can also be used to implement an noncausal LTI system.

conv vs filter

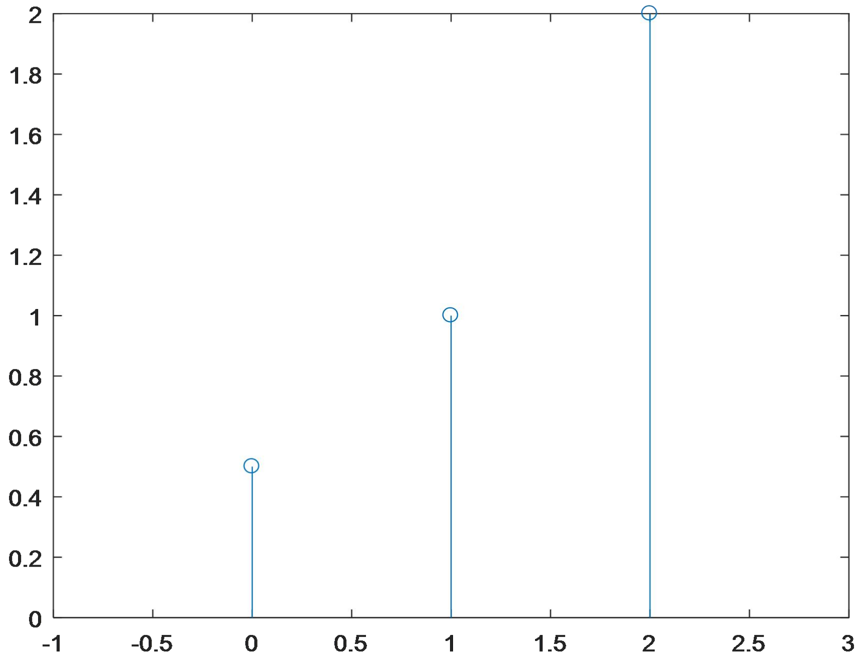
y[n]=0.5x[n]+x[n-1]+2x[n-2];

```
h=[0.5 1 2];
nh=0:2;
x=1;
nx=0;
y=conv(x,h);
ny=0:2;
figure,stem(ny,y,'*','r'),xlim([-1 3])
```



||

```
a=[1];
b=[0.5 1 2];
x=[1 0 0 ];
nx=0:2;
y=filter(b,a,x)
figure,stem(nx,y),xlim([-1 3])
```



	conv	filter
Syntax	w=conv(u,v);	y=filter(B,A,x)
Scenario	finite-length input finite-length impulse response	finite-length input finite-length or infinite-length impulse response
Notes	nw=nu(1)+nv(1):nu(end)+nv(end);	x and y share the same range of time indices **——Output signal y may be truncated.**

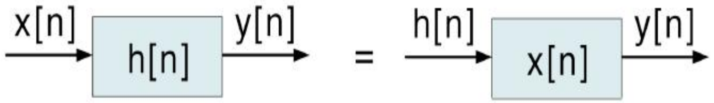
Any DT LTI system are completely characterized by its unit impulse response!

Causality:	$h[n] = 0 \qquad \text{for all } n < 0$
Stability:	$\sum_{k=-\infty}^{+\infty} \lvert h[k] \rvert < \infty$
Memory/Memoryless:	$h[n] = K\delta[n]$
Invertibility:	$h[n] * h1[n] = \delta[n]$

➢ Properties of DT LTI Systems

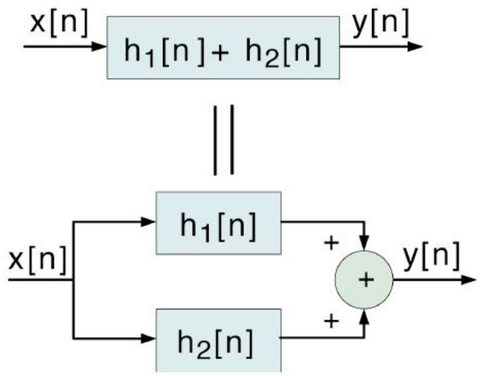
$$y[n] = x[n] * h[n] \quad = \quad h[n] * x[n]$$

Commutative



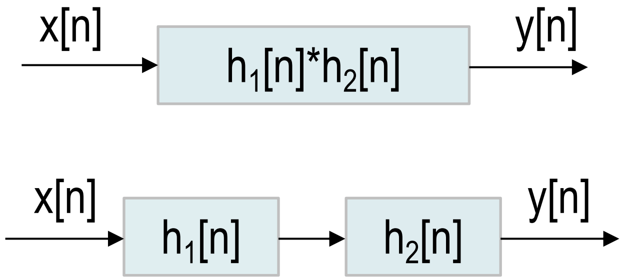
$$x[n] * \{h_1[n] + h_2[n]\} = x[n] * h_1[n] + x[n] * h_2[n]$$

Distributive



$$x[n] * (h_1[n] * h_2[n]) = (x[n] * h_1[n]) * h_2[n]$$

Associative



■ 2.4 Properties of Discrete-Time LTI Systems

In this exercise, you will verify the commutative, associative and distributive properties of convolution for a specific set of signals. In addition, you will examine the implications of these properties for series and parallel connections of LTI systems. The problems in this exercise will assume that you are comfortable and familiar with the `conv` function described in Tutorial 2.1. Although the problems in this exercise solely explore discrete-time systems, the same properties are also valid for continuous-time systems.

Lab homework !

Lab2 Assignments

- Read tutorial 2.1、2.2、2.3
- homeworks: **2.4，** 2.10.

Tips：

1 Download **lineup.mat** for 2.10

2 2.10 f , there is an obvious error in the sample code, fix it yourself.