Chapter 2 - Linear Time-Invariant Systems

2.1 Tutorial Convoluation

(a) Consider the finite-length signal

$$x[n] = \begin{cases} 1 & 0 \le n \le 5 \\ 0 & \text{otherwise} \end{cases} \qquad (2.5)$$

Analytically determine $y[n] = x[n] * x[n]$

Get output of y[n] with the convolution sum ($*$).

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

$$y[n] = \sum_{n=-5}^{5} x(k)h(n-k)$$

==

REVISION:

1. By the first term, x[k], the non-zero values are k=0 to 5.

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]x[n-k] = \sum_{k=0}^{5} x[k]x[n-k]$$

2. By the second term, x[n-k], the non zero values are at n=0 to 10. So we have y[n] for n in [0, 10].

$$y[0] = \sum_{0}^{5} x[k]x[-k] = 1;$$

$$y[1] = \sum_{0}^{5} x[k]x[1-k] = 2;$$

$$y[5] = \sum_{0}^{5} x[k]x[5-k] = 6;$$

$$y[6] = \sum_{0}^{5} x[k]x[6-k] = 5;$$

$$y[7] = \sum_{0}^{5} x[k]x[7-k] = 4;$$

$$y[10] = \sum_{0}^{5} x[k]x[10-k] = 1$$

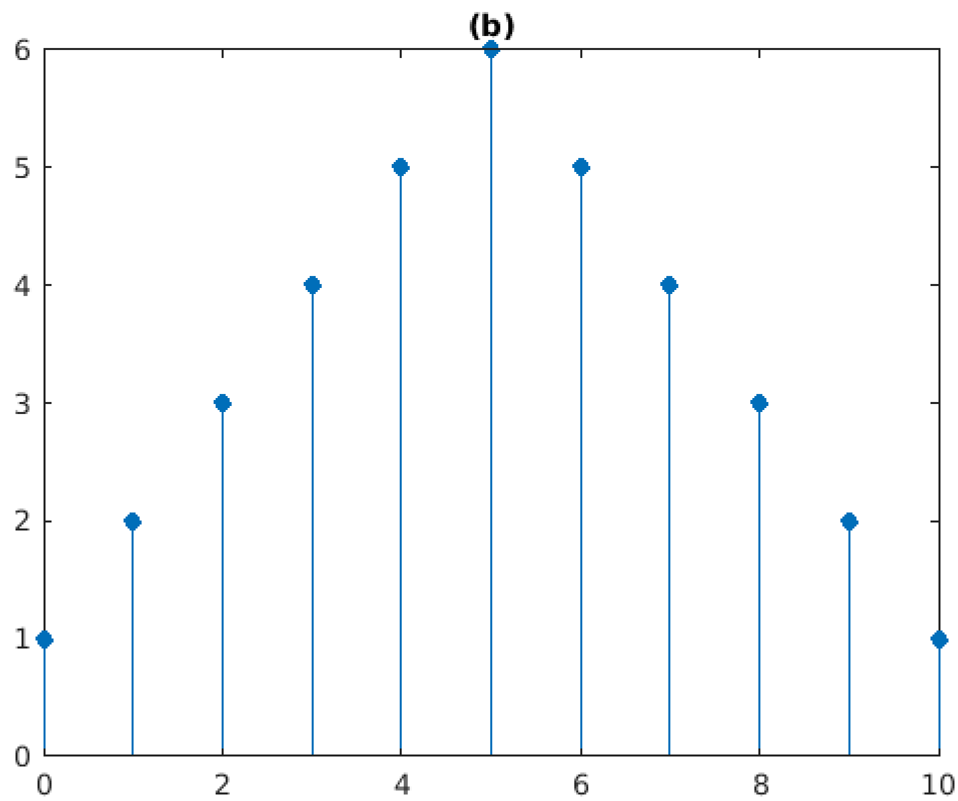
Then

$y[11] = \sum_0^5 x[k]x[11-k] = 0$, the same for y[n]=0 where n>10.

==

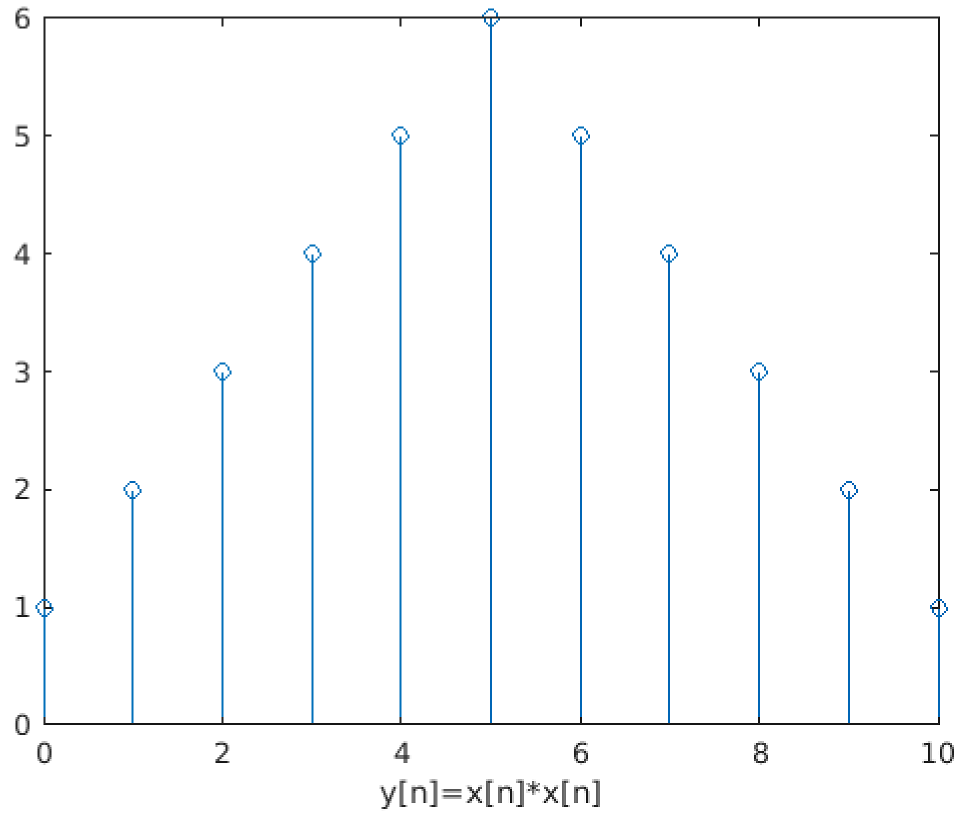
(b) Compute the non-zero samples of $y[n] = x[n] * x[n]$ using ***conv***, *and store these samples in the vectory y.* plot the results and check you plot to see if it agrees what you got in (a) and it shold also agrees with Figure 2.1.

```
% L2_1b.m
n=[0:5];
x=ones(1, length(n));
y=conv(x, x);
ny=[0:10];
figure(1);
stem(ny, y, 'filled');
title('(b)');
```




```
% =====================================================================
```

```
% REVISION
clf;

x=[0:5];
x(1:end)=1;
ny=[0:10];

y=conv(x, x);
stem(ny, y);
xlabel('y[n]=x[n]*x[n]');
```




```
% ====================================================================
```

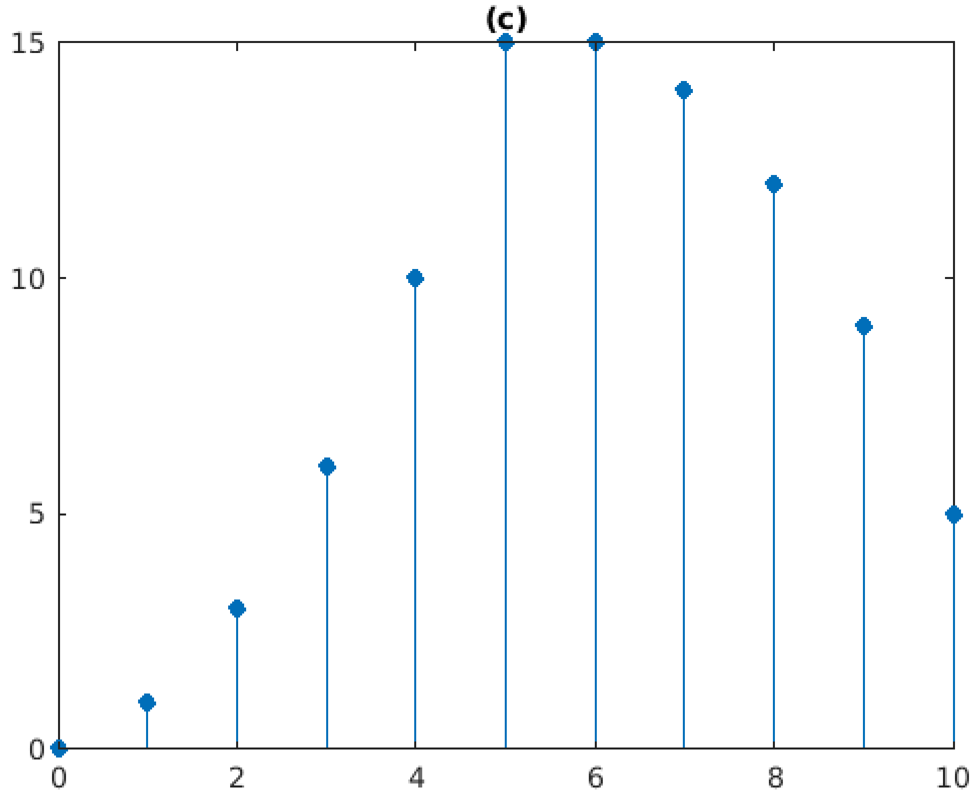
(c) Consider the finite-length signal

$$h[n] = \begin{cases} n, & 0 \le n \le 5 \\ 0, & \text{otherwise} \end{cases} \qquad (2.6)$$

Analytically compute $y[n] = x[n] * h[n]$. Next, compute ***y*** using ***conv,*** then plot your results., As a check you plot should agree with Figuree 2.2 and your analytical derivation.

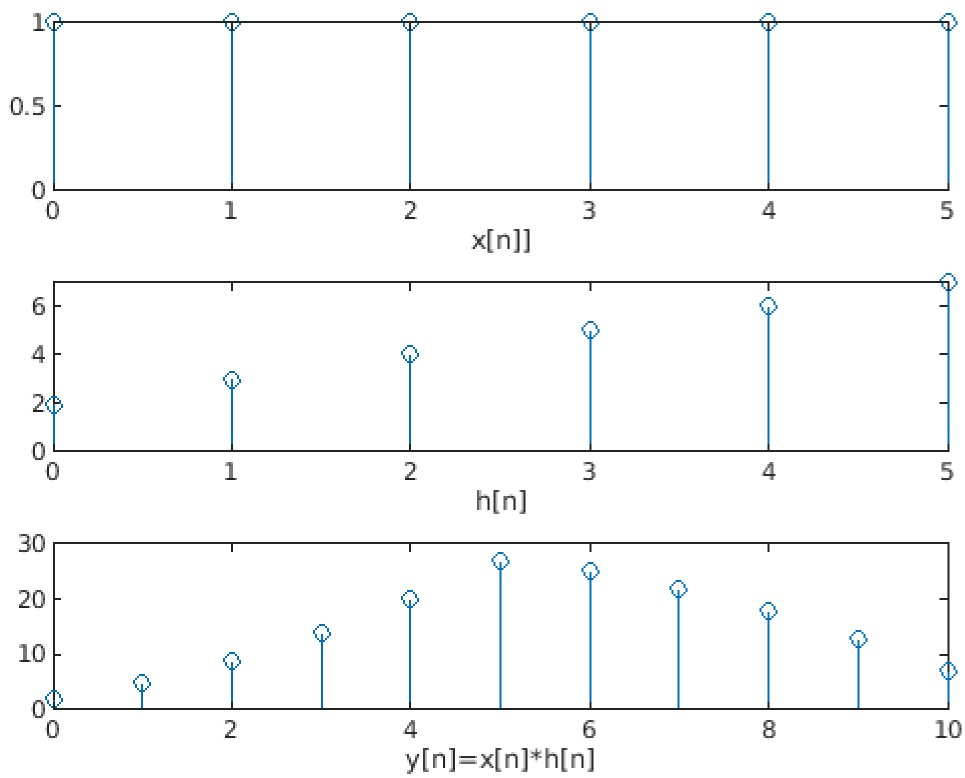
```
n=[0:5];
x=ones(1,length(n));
h=[0:5];
y=conv(x, h);
ny=[0:10];
figure(2);
```

```
stem(ny, y, 'filled');
title('(c)')
```




```
% ==============================================================================
% REVISION

clf;

n=[0:5];
x=n*0;
x(1:end)=1;
subplot(3, 1, 1);
stem(n, x);
xlabel('x[n]]');

h=n*0;
for k=1:6
    h(k)=k+1;
end
subplot(3, 1, 2);
stem(n, h);
xlabel('h[n]');

ny=[0:10];
y=conv(x, h);
subplot(3, 1, 3);
stem(ny, y);
```

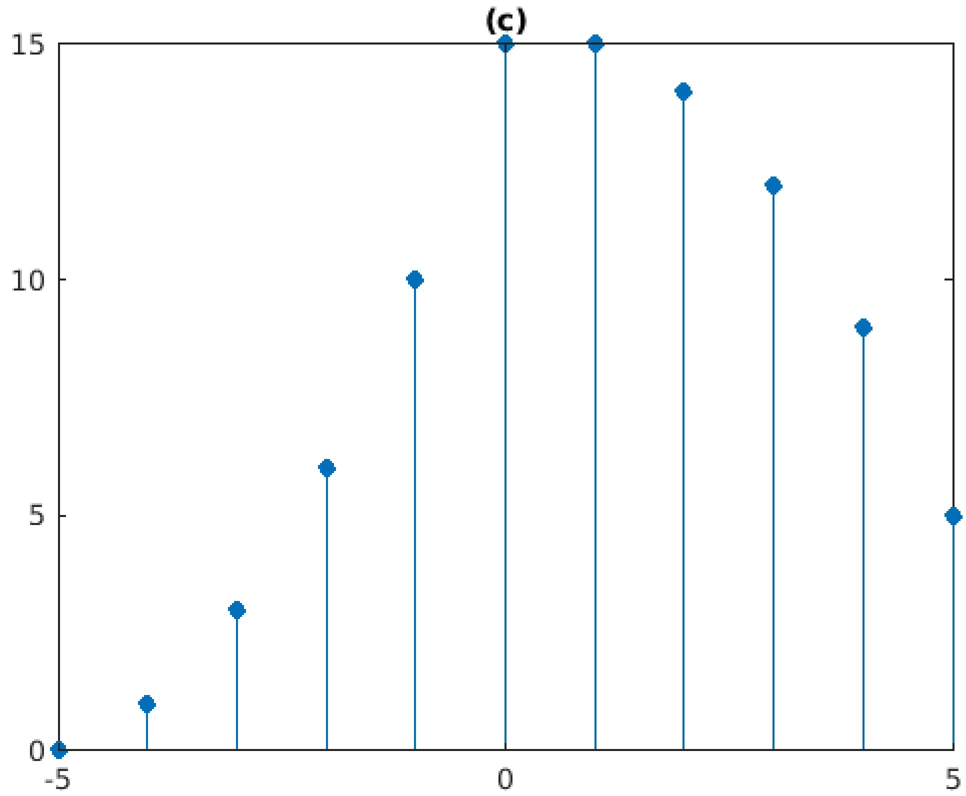
```
xlabel('y[n]=x[n]*h[n]');
```




```
% ==========================================================================
```

Your Analytical Solution:

```
% L2_1c.m
```



(d) Let $y_2[n] = x[n] * h[n+5]$, compare to the signal y[n] derived in Part-C, what is your conclusion?

Put your Analysis Here:

The difference between y[n] and y2[n] is that with the [n+5] shifts the graph over to the left.

===

REVISION

As we see in Part c, $y[n] = x[n] * h[n]$ is a Linear Time-Invarient system. Based on LTI property,

$y_2[n] = x[n] * h[n+5]$ => $y[n+5]$

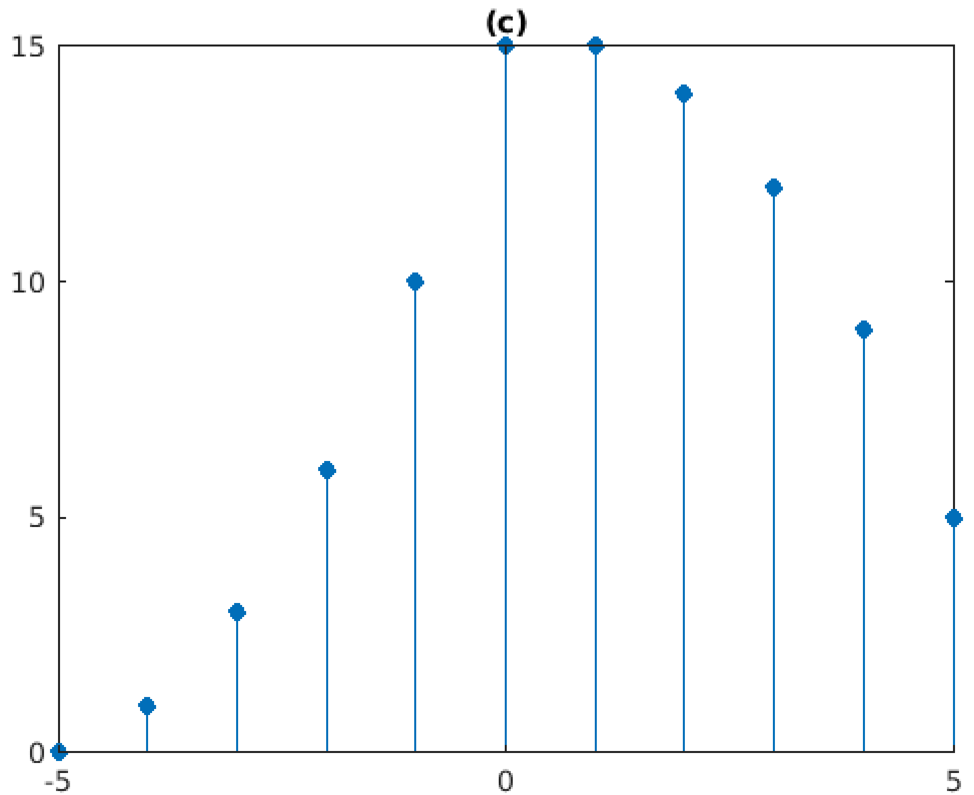
which results aare the same as y[n] with shift 5 units to left (or advance 5 unite). We can verifty these results in Part e below

===

(e) Use conv to compute the $y_2[n]$, then plot the results which should agree with Figure 2.3.

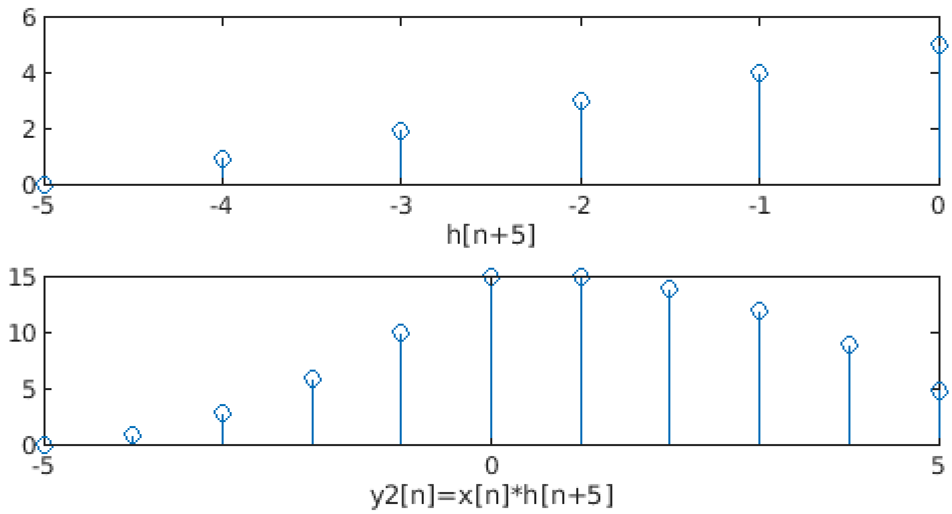
```
% L2_1e.m
n=[0:5];
x=ones(1,length(n));
h=[0:5];
```

```
y=conv(x, h);
ny=[-5:5];
figure(1);
stem(ny, y, 'filled');
title('(c)')
```




```
% ============================================================
% REVISION

clf;

nx=[0:5];
x=nx*0;
x(1:end)=1;

nh=[-5:0];
h=nh*0;
for k=1:6
    h(k)=k-1;
end
subplot(3, 1, 1);
stem(nh, h);
xlabel('h[n+5]');

ny=[-5:5];
y=conv(x, h);
subplot(3, 1, 2);
stem(ny, y);
```

```
xlabel('y2[n]=x[n]*h[n+5]');
```




```
% ================================================================
```

2.2 Tutorial Filter

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

$$\sum_{k=0}^{k} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m] \qquad (2.7)$$

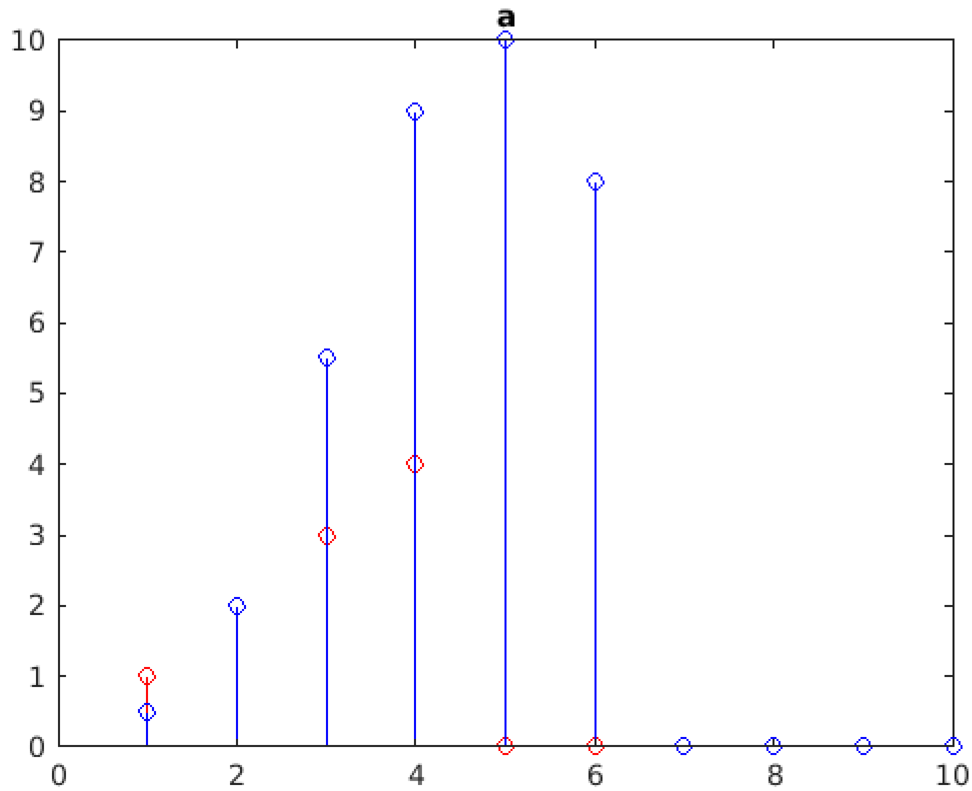
where x[n] is the system input and [n] is the system output.

(a) Define coffeicient vectors a1 and b1 t describe the causal LTI system specified by $y[n] = 0.5x[n] + x[n-1] + 2x[n-2]$.

Your Answer:

```
clf;
n=1:1:10;
x=1*(n==1)+2*(n==2)+3*(n==3)+4*(n==4);
num=[0.5 1 2];
den=1;
y=filter(num, den, x);
figure(1);
```

```
stem(n, x, 'r');
hold on
stem(n, y, 'b');
title('a');
```



==

REVISION

Here a1=(1) and b1=(0.5, 1, 2)

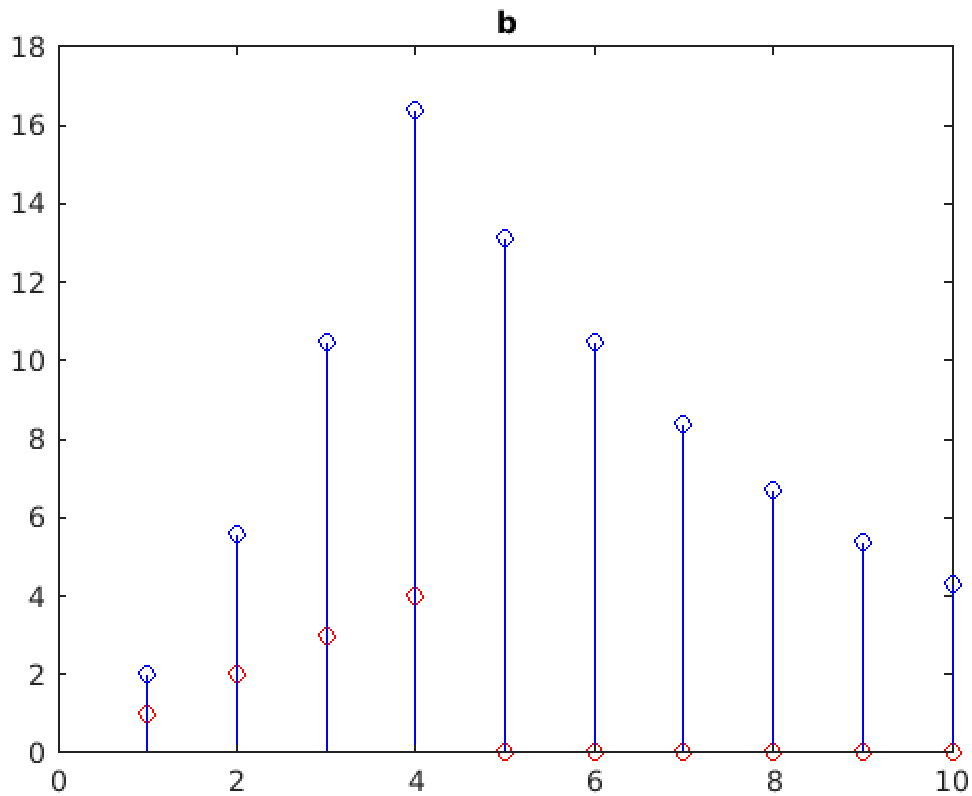
==

(b) Define coefficient vector a2 and b2 to describe the causal LTI system sepecified by $y[n] = 0.8y[n-1] + 2x[n]$.

You Answer:

```
n=1:1:10;
x=1*(n==1)+2*(n==2)+3*(n==3)+4*(n==4);
num=2;
den=[1 -0.8];
y=filter(num, den, x);
figure(2);
stem(n, x, 'r');
hold on
stem(n, y, 'b');
```

```
title('b')
```



==

REVISION

Here a2=(1, -0.8) and b2=(2)

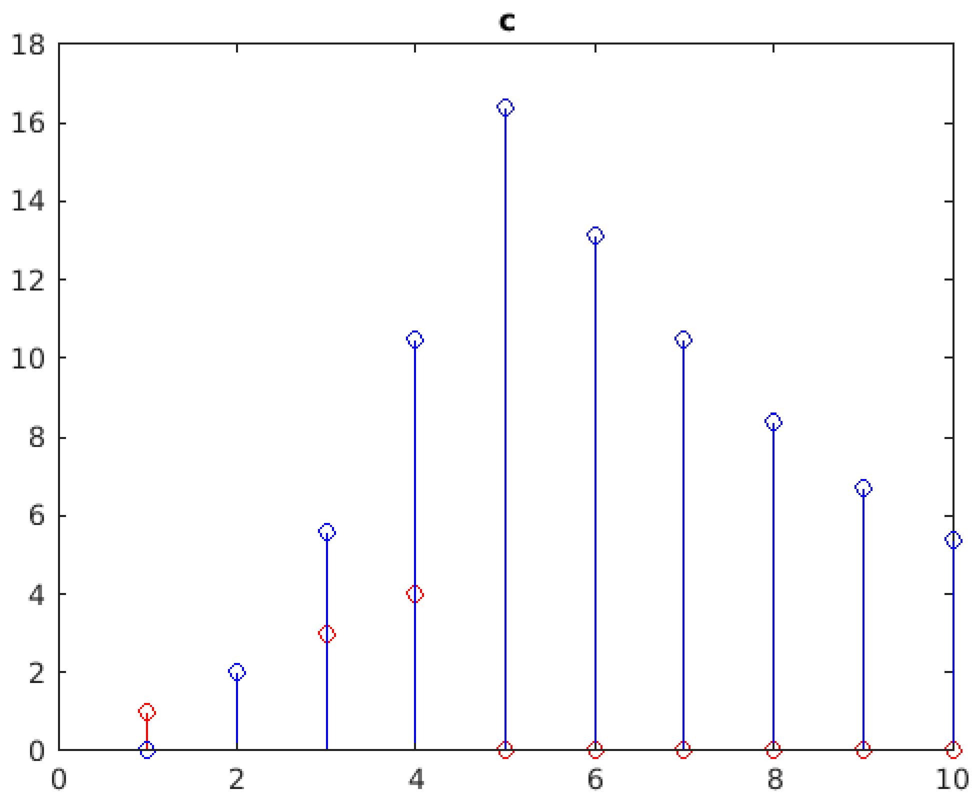
==

(c) Define coefficient vector a3 and b3 to describe the causal LTI system specifed by

$y[n] - 0.8y[n-1] = 2x[n-1]$

Your Answer:

```
n=1:1:10;
x=1*(n==1)+2*(n==2)+3*(n==3)+4*(n==4);
num=[0 2];
den=[1 -0.8];
y=filter(num, den , x);
figure(3);
stem(n, x, 'r');
hold on
stem(n, y, 'b');
title('c');
```



==

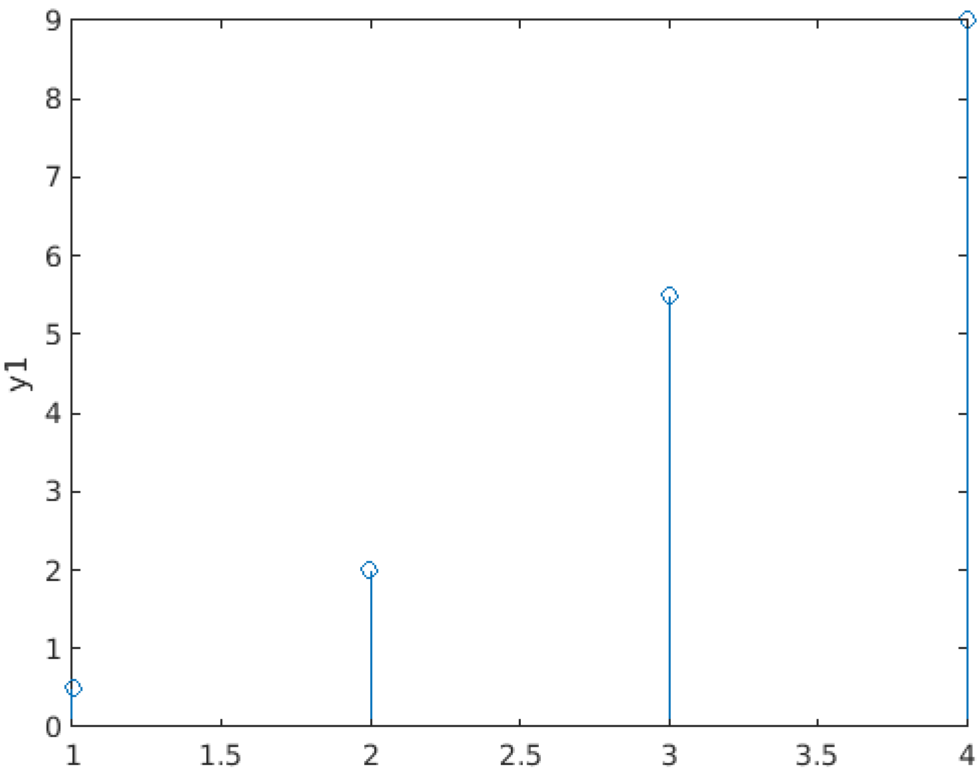
REVISION

Here a3=(1, -0.8) and b3=(0, 2)

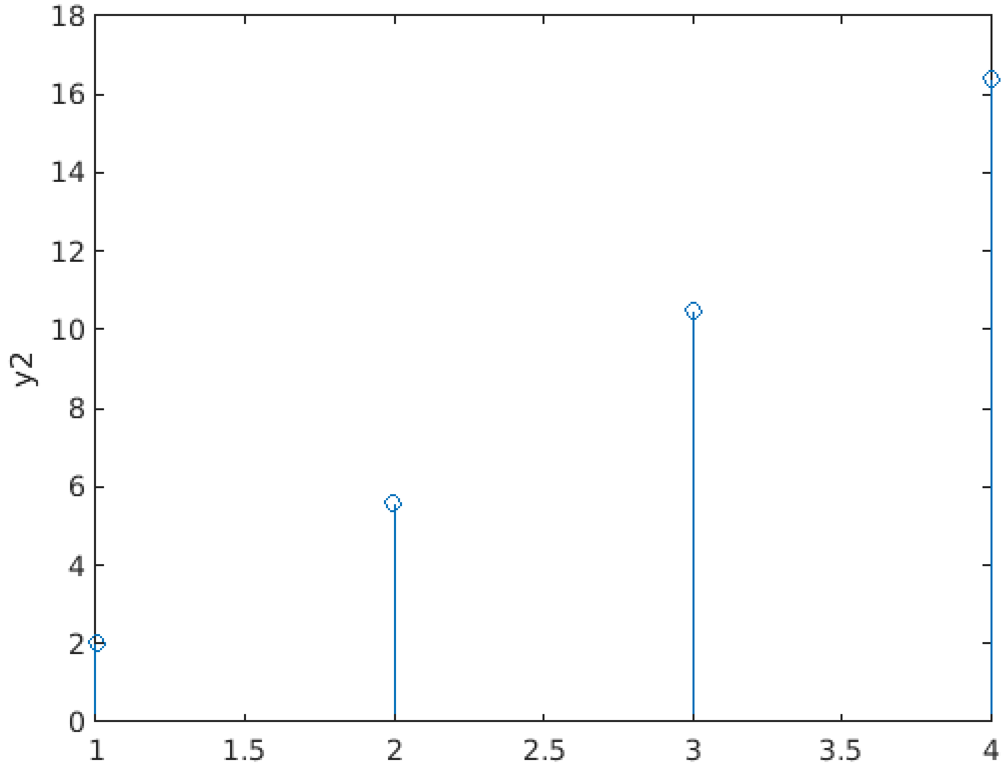
==

(d) For each these three systems, use ***filter*** to compute the response ***y[n]*** on the interval $1 \le n \le 4$ to the imput signal $x[n] = nu[n]$. You should begin by defining the vector x = [1 2 3 4] which contains x[n] on the interval $1 \le n \le 4$. The result of using fitler for each system is shown below:

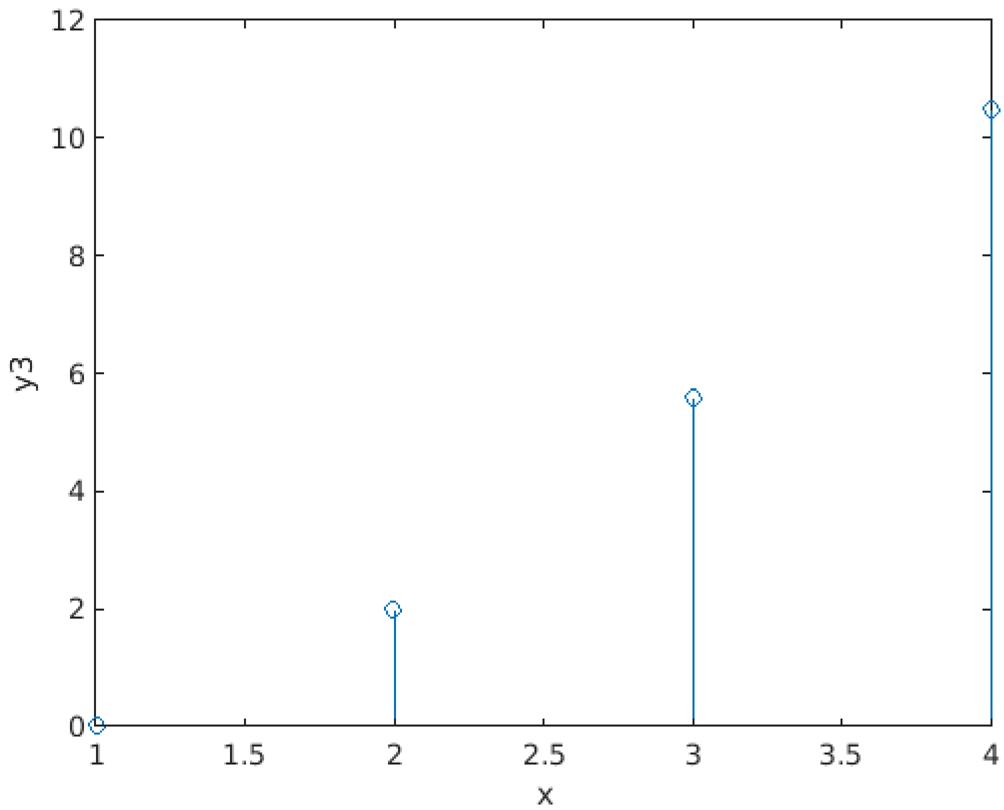
```
% ==========================================================================
% REVISION

clf;
x=[1 2 3 4];

a1=[1 0 0];
b1=[0.5 1 2];
y1=filter(b1, a1, x);
figure;
stem(x, y1);
ylabel('y1')
```




```
a2=[1 -0.8];
b2=[2];
y2=filter(b2, a2, x);
figure;
stem(x, y2);
ylabel('y2')
```




```
a3=[1 -0.8];
b3=[0 2];
y3=filter(b3, a3, x);
figure;
stem(x, y3);
ylabel('y3')
xlabel('x')
```




```
% ==========================================================================
```

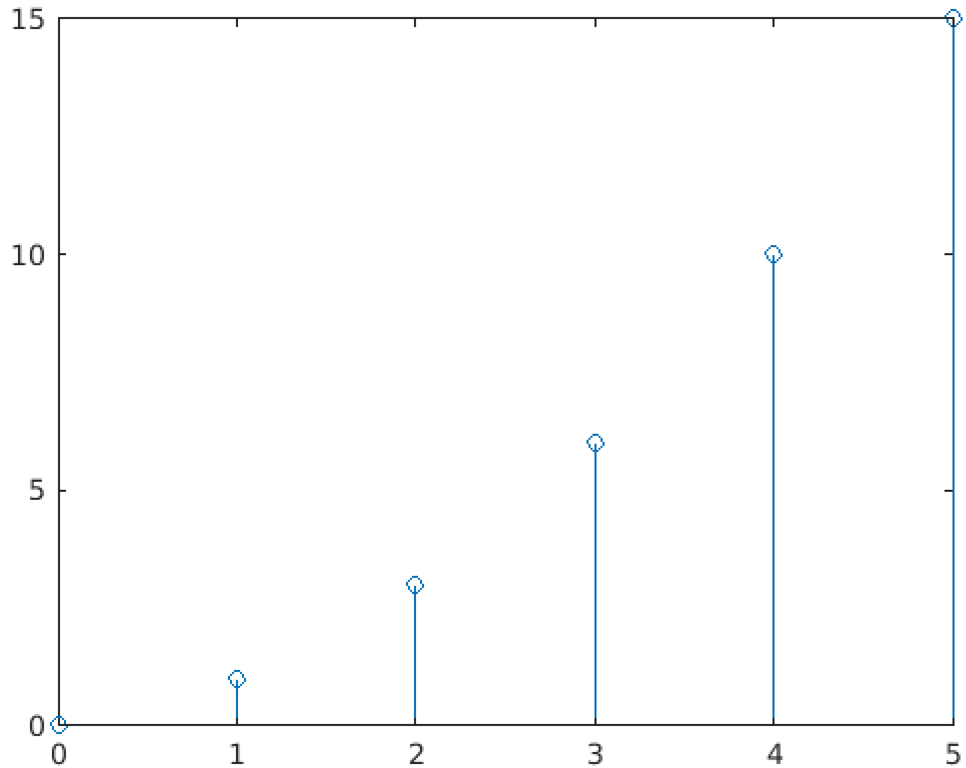
```
% L2_2.m
% MATLAB code for 2.2 Tutorial filter for problem a), b), c) and d) on the Companion bo
```

2.2 (e) and (f):

The function filter can also be used to perform discrete-time convolution. To Illustrate how to use filter to implement a discrete-time convolution, consider the convolution of x[n] in Eq. (2.5) and h[n] in Eq. (2.6).

```
% L2_2e_f.m
% use the filter function to perform convoluation for discrete-time systems.

% ==========================================================================
% REVISION

clf;

%store x[n] and h[n] as eq 25 and eq 26
x=[0:5];
x(1:end)=1;
h=x*0;
for k=1:6
```

```
    h(k)=k-1;
end

y=filter(h, 1, x);
ny=[0:5];
stem(ny, y);
```



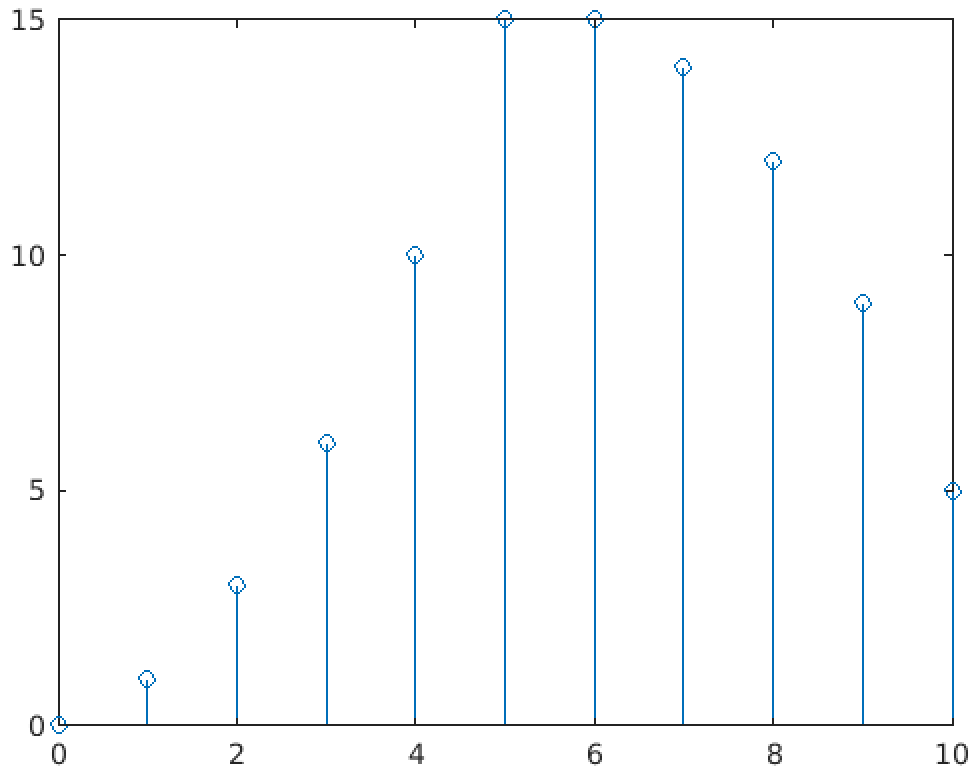

```
% ==========================================================================
```

2.2 (g):

Define a vector x2 to contain x[n] on a interval $0 \le n \le 10$, and use $y2 = f\,lter(h, 1, x2)$; to compare the convolution on this interval. Plot your results and very that it agrees with Figure 2.2.

```
% L2_2g.m
% use the filter function to perform convoluation for discrete-time
% systems.

% ==========================================================================
% REVISION

clf;

ny=[0:10];
x2=ny*0;
```

```
x2(1:6)=1;
h=ny*0;
for k=1:6
    h(k)=k-1;
end
y2=filter(h, 1, x2);
stem(ny, y2);
```




```
% ==========================================================================
```

2.2 (h) and (i):

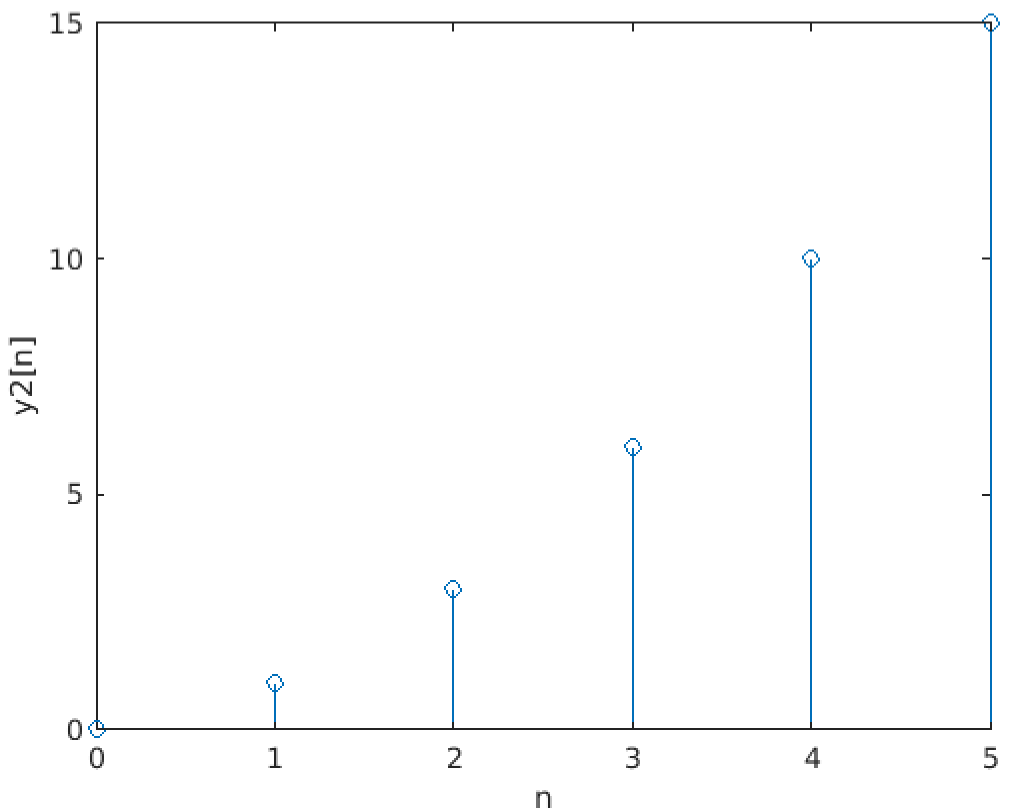
Like *conv*, *filter* can also be used to implement a LTI system which has a noncausal impulse response.

(h) Consider the impulse response $h_2[n] = h[n+5]$, h[n] is defined in Eq. (2.6). Store $h_2[n]$ on the interval $-5 \leq n \leq 0$ in the vector h2.

(i) Excute the command $y_2 = f\,lter(h2, 1, x)$ and create a vector ny2 which contains indices of the samples of $y_2[n] = h_2[n] * x[n]$ stored in y2. Plot your result and check how does this plot compare eith Figure2.4?

```
%L2_2h_i.m
% ==========================================================================
% REVISION

clf;
```

```
%(h) store h2[n]
h2=[-5:0];
for k=1:6
    h2(k)=k-1;
end
%(i) using filter function to compute y2
ny2=[0:5];
x=ny2*0;
x(1:end)=1;
y2=filter(h2, 1, x);
stem(ny2, y2);
xlabel('n');
ylabel('y2[n]')
```




```
% =========================================================================
```

2.2 (j) Create a vector x2 such that ***filter(h2,1,x2)*** returns all the nonzero samples of y2[n].

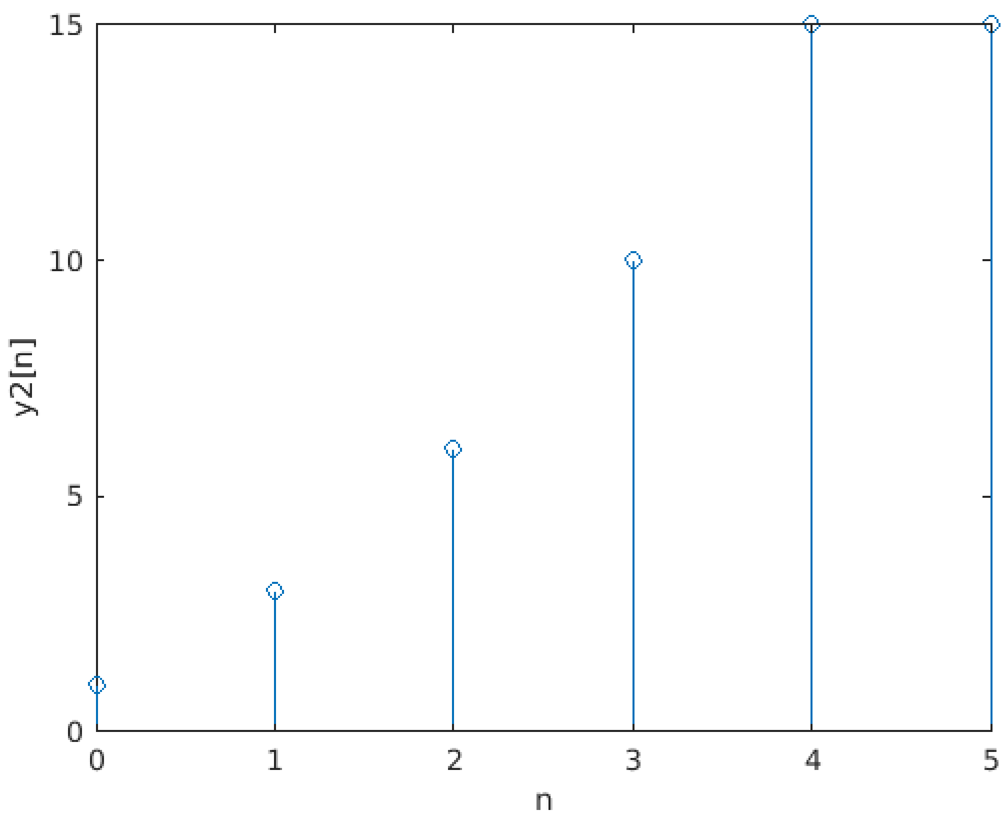
===

REVISION

To get out of one non zero value of y2[n], we need to shift y2[n] left by one unit, which is y2[n+1]

```
% L2_2j.m

clf;
% h(h) store h2[n]
```

```
h2=[-5:0];
for k=1:6
    h2(k)=k-1;
end
% (i) using filter function to compute y2
nx2=[0:6];
x2=nx2*0;
x2(1:6)=1;
y=filter(h2, 1, x2);
for k=1:5
    y2(k)=y(k+1);
end
ny2=[0:5];
stem(ny2, y2);
xlabel('n');
ylabel('y2[n]');
```



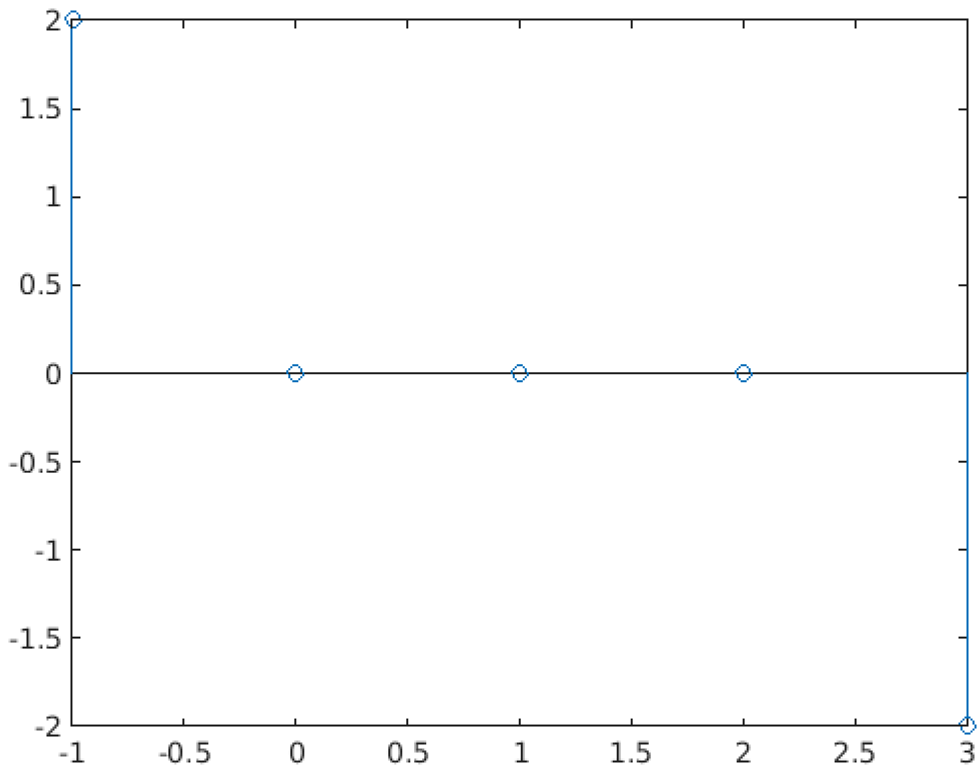
==

2.7 Discrete-Time Convolution

In these problems, you will define some discrete-time signals and the impulse responses of some discrete-time LTI systems. Then the output of the LTI systems can be computed using ***conv.***

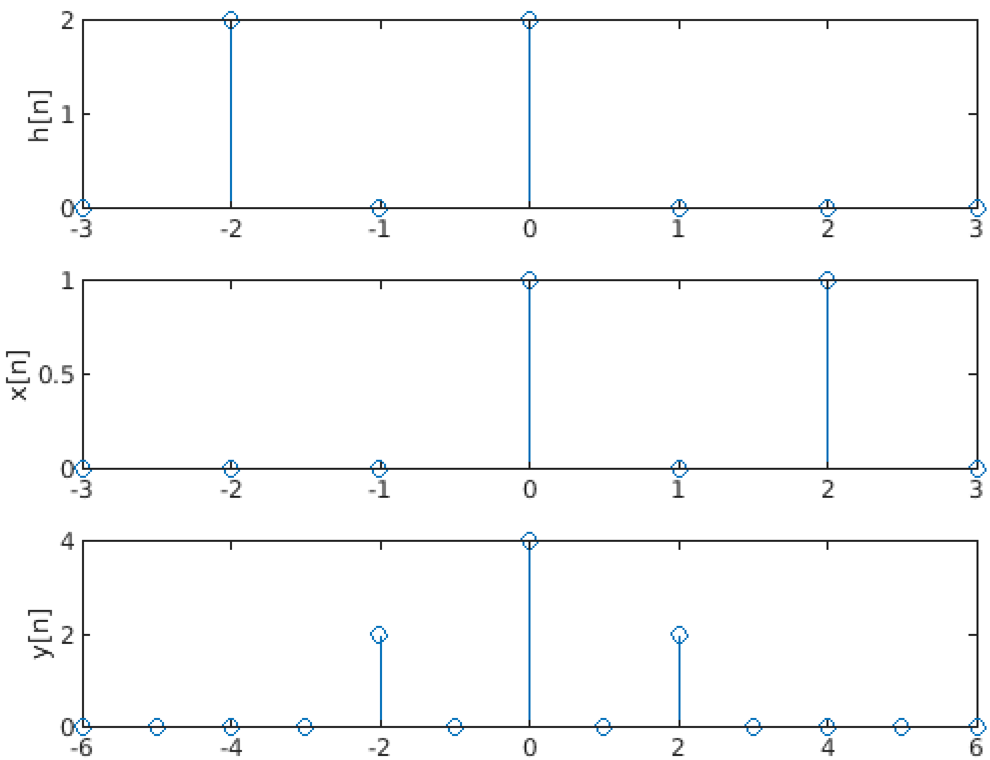
(a). Since the MATLAB function ***conv*** does not keep track of the time indices of the sequences that are convolved, you will have to do some extra bookkeping in order to determine the proper indices for the result of the ***conv*** function. For the sequences $h[n] = 2\delta[n+1] - 2\delta[n-1],\ and\ \ x[n] = \delta[n] + \delta[n-2]$ construct vectors h and x. Define $y[n] = x[n] * h[n]$ and compute $y = conv(h, x)$. Determine the proper time indexing for y and store this set of time indices in the vector ny. Plot y[n] as a function of n using $stem(ny, y)$.

```
clf;
x_n = [1 0 1];
h_n = [2 0 -2];
y = conv(h_n, x_n);
ny = -1:1:3;
stem(ny, y);
```




```
% ==========================================================================
% REVISION

clf;

nh=[-3:3];
h=nh*0;
h(2)=2;
h(3)=0;
h(4)=2;
subplot(3, 1, 1);
stem(nh, h);
ylabel('h[n]');
```

```
nx=[-3:3];
x=nx*0;
x(4)=1;
x(6)=1;
subplot(3, 1, 2);
stem(nx, x);
ylabel('x[n]');
y=conv(h, x);
ny=[-6:6];
subplot(3, 1, 3);
stem(ny, y);
ylabel('y[n]');
```




```
% ======================================================================
```

(b). Consider two finite-length sequences h[n] and x[n] that are represnted in MATLAB using the vectors h and x, with corresponding time indices given by nh=[a:b] and nx=[c:d]. The function call $y = conv(h, x)$ will return in the vector y the proper sequence values of $y[n] = h[n] * x[n]$; however, you must determine a corresponding set of time indices ny. To help you construct ny, consider the sequence $h[n] = \delta[n-a] + \delta[n-b]$ *and* $x[n] = \delta[n = c] + \delta[n-d]$. Determine analyticall the convolution $y[n] = h[n] * x[n]$. From your answer, determine what ny should be in terms of a, b, c, and d. To check your result, verify that y[n] is of length M+N-1 when a=0, b=N-1, c=0, and d=M-1.

$h[n] = \delta[n-a] + \delta[n-b]$

$x[n] = \delta[n-c] + \delta[n-d]$

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

$\delta[n] * \delta[n-k] = \delta[n-k]$

$h[n]x[n] = [\delta(n-a) + \delta(n-b)][\delta(n-c) + \delta(n-d)]$

$= \delta(n-a) * \delta(n-c) + \delta(n-a) * \delta(n-d) + \delta(n-b) * \delta(n-c) + \delta(n-b) * \delta(n-d)$

$y[n] = \delta[n-(a+c)] + \delta[n-(a+d)] + \delta[n-(b+c)] + \delta[n-(b+d)]$

$hy = [a+c : b+d]$

$a = 0, n = N-1, c = 0, d = M-1$

$n, y = [0+0 : M-1+N-1]$

$n, y = [) : M+N-2]$

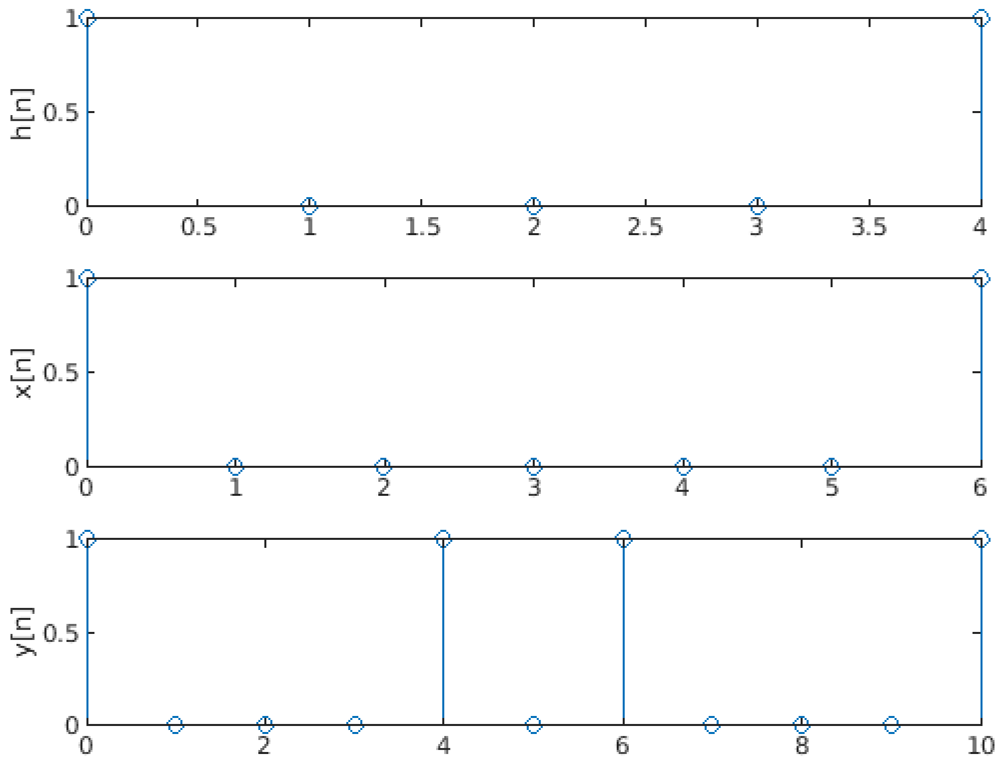
$y[n]$ varies from 0 to M+N-1. y[n] has a length of M+N-1

==

REVISION

```
clf;

N=5;
M=7;
nh=[0:N-1];
h=nh*0;
h(1)=1;
h(5)=1;
subplot(3, 1, 1);
stem(nh, h);
ylabel('h[n]');
nx=[0:6];
x=nx*0;
x(1)=1;
x(7)=1;
subplot(3, 1, 2);
stem(nx, x);
```

```
ylabel('x[n]');
y=conv(h, x);
ny=[0:10];
subplot(3, 1, 3);
stem(ny, y);
ylabel('y[n]');
```



The plots above verifies that y[n] is of length M+N-1 when a=0, b=N-1, c=0, and d=M-1. In this case the length of y[n]=M+N-1 = 7+5-1 = 11.

==

(c). Consider an input x[n] and unit impulse response h[n] given by:

$x[n] = (1/2)^{n-2}u[n-2]$,

$h[n] = u[n+2]$

if you wish to compute $y[n] = h[n] * x[n]$ using ***conv***, you must deal appropriately with the infinte length of both x[n] and h[n]. Store in the vector x the values of x[n] for $0 \le n \le 24$ and store in the vector h the values of h[n] for $-2 \le n \le 14$. Now store in the vector y the result of the function call $conv(h, x)$. Since you have truncated both h[n] and x[n], argue that only a portion of the output of **conv** will be valid. Specify which values in the output are valid and which are not. Determine the values of the parameters a, b, c, and d such that nx-[a:b] and nh=[c:d], and use your answer from Part (b) to construct the proper time indices for y. Plot y[n] using ***stem***, and indicate which values of y[n] are correct, and which values are invalid. Be sure to properly label the time axis for y[n].

$x[n] = (1/2)^{n-2}u[n-2]$

$h[n] = u[n+2]$

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

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

$$= \sum_{k=-\infty}^{\infty} (1/2)^{k-2}u[k-2]u[n-k+2]$$

u[k-2] is 0 for k<2

u[n-k+2] is 0 for k>m+2

$$y[n] = \sum_{k=2}^{n+2} (1/2)^{k-2}$$

$$= 1 + (1/2) + (1/2)^2 + (1/2)^3 + ... + (1/2)^n$$

$$= \frac{1-(1/2)^{n+1}}{1-(1/2)}$$

$y[n] = 2[1-(1/2)^{n+1}]u[n]$

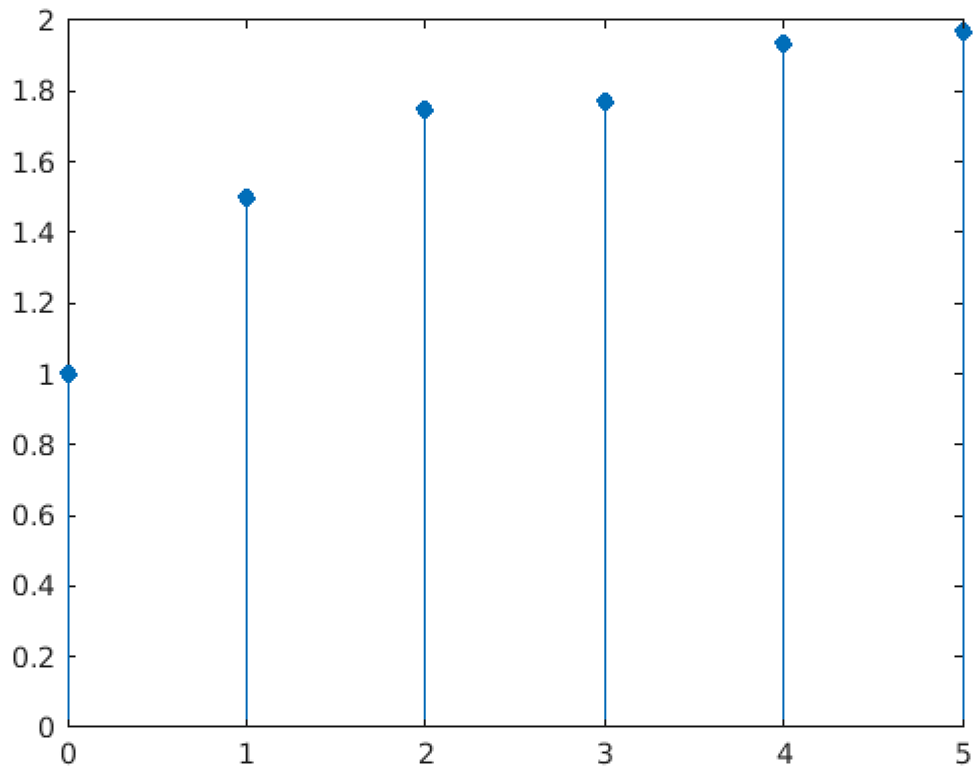
$= [2 - 2(1/2)^{n+1}]u[n]$

$y[n] = [2 - 2^{-n}]u[n]$

```
clf;
n = 0:5;
x = [1 1.5 1.75 1.772 1.9375 1.96875]
```

```
x = 1×6
    1.0000    1.5000    1.7500    1.7720    1.9375    1.9688
```

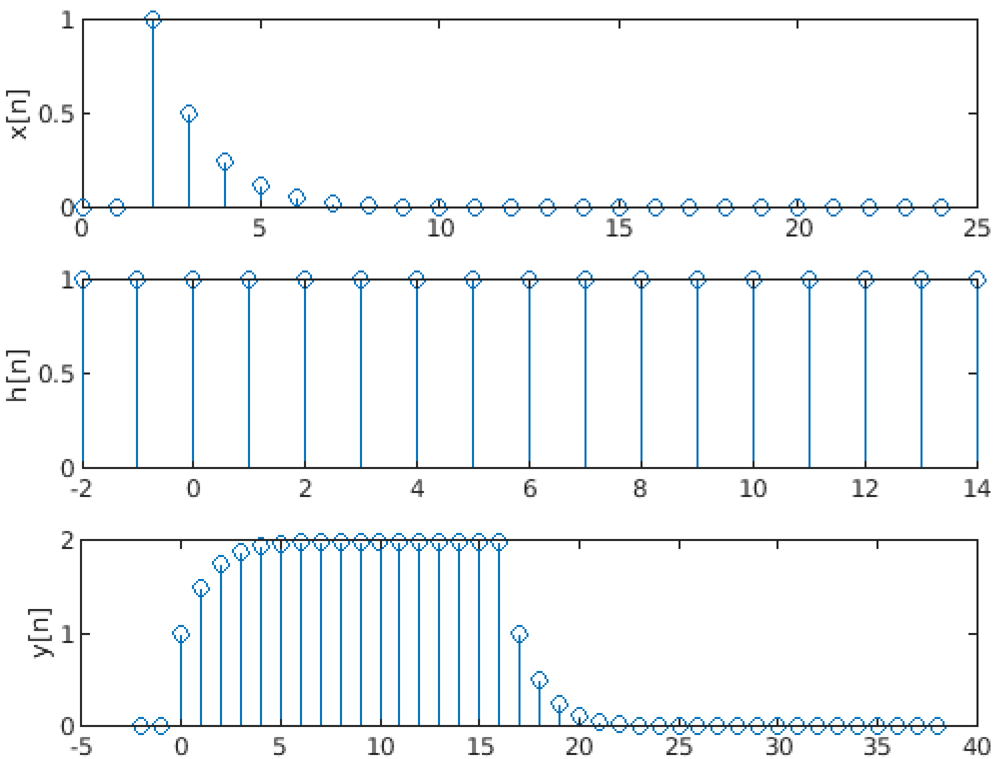
```
stem(n, x, 'filled');
```



===

REVISION

```
clf;
nx=[0:24];
x=nx*0;
for i=3:25
    x(i)=(1/2)^(i-3);
end
subplot(3, 1, 1);
stem(nx, x);
ylabel('x[n]');
nh=[-2:14];
h=nh*0;
h(1:end)=1;
subplot(3, 1, 2);
stem(nh, h);
ylabel('h[n]');
y=conv(x, h);
ny=[-2:38];
subplot(3, 1, 3);
stem(ny, y);
ylabel('y[n]');
```



Given index for the vector x the values of x[n] for 0 <=n<=24 (N=25) and index for the vector h the values of h[n] for -2 <=n<=14 (M=17), we have the values of the parameters a=0, b=24, c=2, and d=14. Using the result for Part b, the size of y[n] should be N+M-1=41, the index for y[n] should be ny=[-2, 38]. y[n] has correct values in the range [-2, 38] and the values outside that range are invalid.

===