# EEHE5666 (DSP) : Homework-1

**Due on January 28, 2022 by 11:59 pm via submission portal.**

**NAME**: McKean, Tyler

### Table of Contents



## Instructions

1. You are required to complete this assignment using Live Editor.
2. Enter your MATLAB script in the spaces provided. If it contains a plot, the plot will be displayed after the script.
3. All your plots must be properly labeled and should have appropriate titles to get full credit.
4. Use the equation editor to typeset mathematical material such as variables, equations, etc.
5. After completeing this assignment, export this Live script to PDF and submit the PDF file through the provided submission portal.
6. You will have only one attempt to submit your assignment. Make every effort to submit the correct and completed PDF file the first time.
7. Please submit your homework before the due date/time. A late submission after midnight of the due date will result in loss of points at a rate of 10% per hour until 8 am the following day, at which time the solutions will be published.

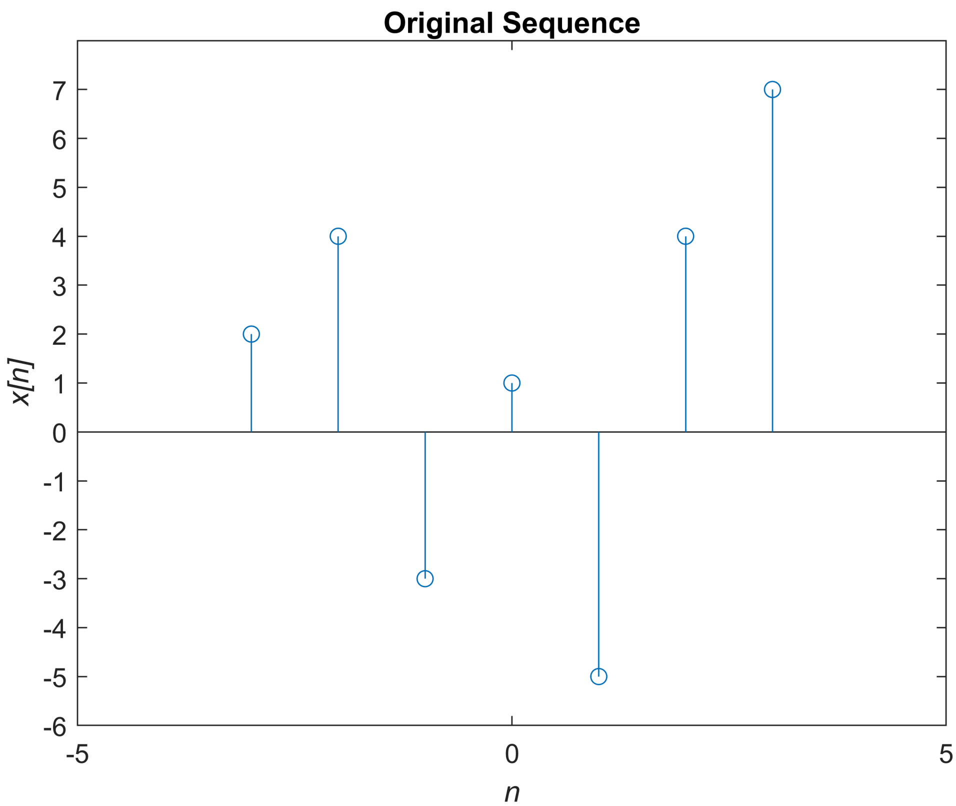
## Default Plot Parameters

```
set(0,'defaultfigurepaperunits','points','defaultfigureunits','points');
set(0,'defaultaxesfontsize',10); set(0,'defaultaxeslinewidth',1.5);
set(0,'defaultaxestitlefontsize',1.4,'defaultaxeslabelfontsize',1.2);
```

## Problem P1.1

Let $x[n] = \{2, 4, -3, \underset{\uparrow}{1}, -5, 4, 7\}$. Using the **timealign,shift,** and **stem** functions, generate and plot samples of the following sequences.

```
clc; close all; clear;
n = -3:3;
x = [2, 4, -3, 1, -5, 4, 7];
figure
stem(n,x)
title("Original Sequence")
xlabel("\it{n}")
ylabel("\it{x[n]}")
yticks(-6:1:7)
xlim([-5 5])
```

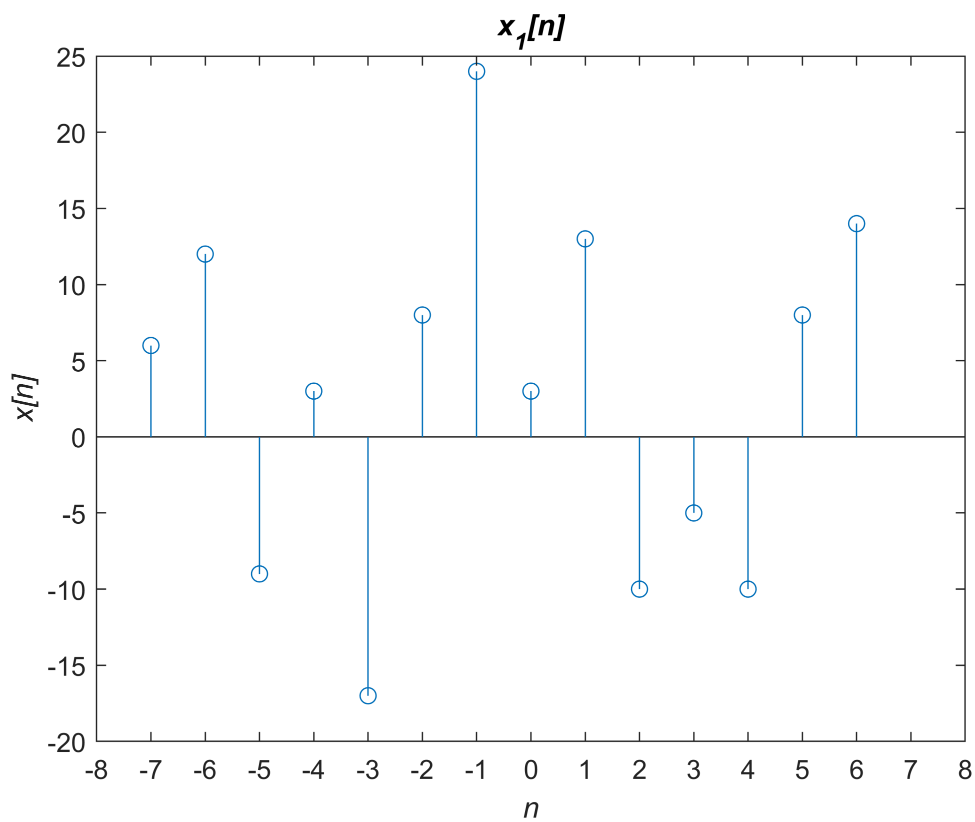



**(a)** $x_1[n] = 2x[n-3] + 3x[n+4] - x[n]$

**MATLAB script**:

```
[x1,n1] = shift(2.*x,n,3);
[x2,n2] = shift(3.*x,n,-4);
xB = timealign(x2,n2,x1,n1);
nB = -7:6;
xC = timealign(x,n,xB,nB);
```

```
xA = timealign(x1,n1,x2,n2);
X1 = xA + xB - xC;
N1 = -7:6;
figure
stem(N1,X1)
xlim([-8 8])
xticks(-8:1:8)
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x_1[n]}")
```



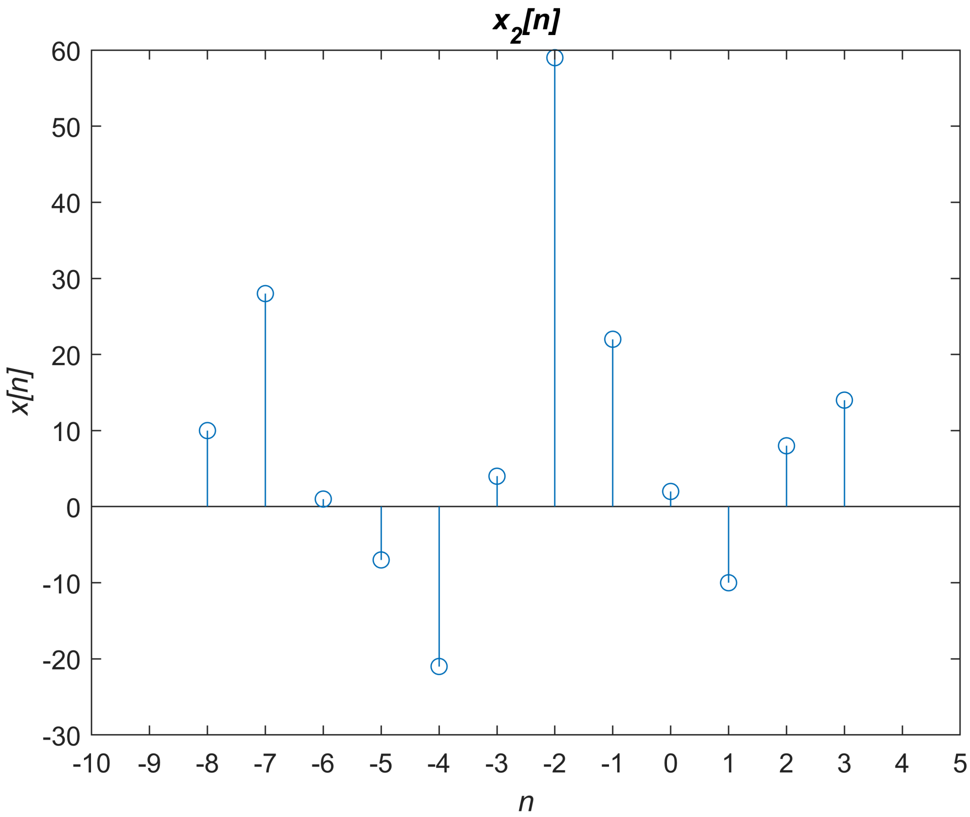

---

**(b)** $x_2[n] = 4x[n+4] + 5x[n+5] + 2x[n]$

**MATLAB script**:

```
[x4,n4] = shift(4.*x,n,-4);
[x5,n5] = shift(5.*x,n,-5);
x4 = timealign(x4,n4,x,n);
x5 = timealign(x5,n5,x,n);
N2 = -8:3;
x = timealign(x,n,x5,N2);
x4 = [0 x4];
X2 = x4 + x5 + 2.*x;
figure
stem(N2,X2)
xlim([-10 5])
xticks(-10:1:5)
```

```
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x_2[n]}")
```




## Problem P1.2

**Text Problem 2.37 (Page 85)**

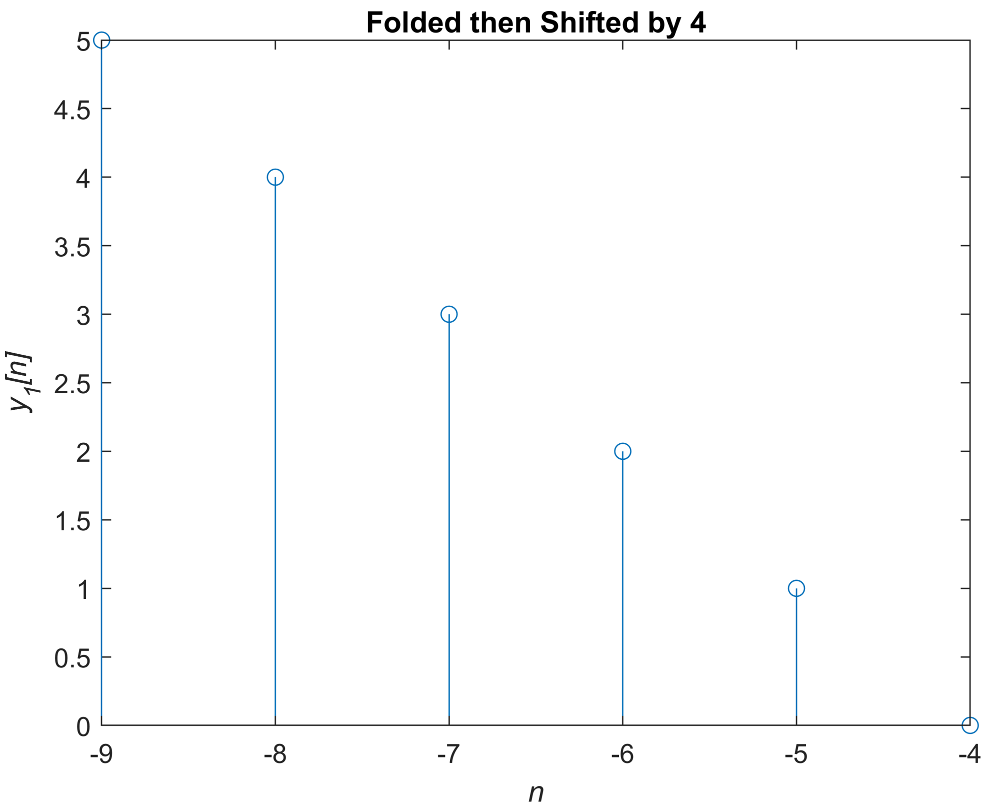
Let $x[n] = \{\underset{\uparrow}{0}, 1, 2, 3, 4, 5\}$. Consider a new sequence $x[-4-n] = x[-(n+4)]$.

**(a)** Let $y_1[n]$ be obtained by first folding $x[n]$ and then shifting the result to the left by four samples. Determine and **`stem`** plot $y_1[n]$.

**MATLAB script**:

```
clc; close all; clear;
x = [0:5];
n = 0:5;
[x1,n1] = fold(x,n);
[y1,ny] = shift(x1,n1,-4);
figure
stem(ny,y1)
xlabel("\it{n}")
```

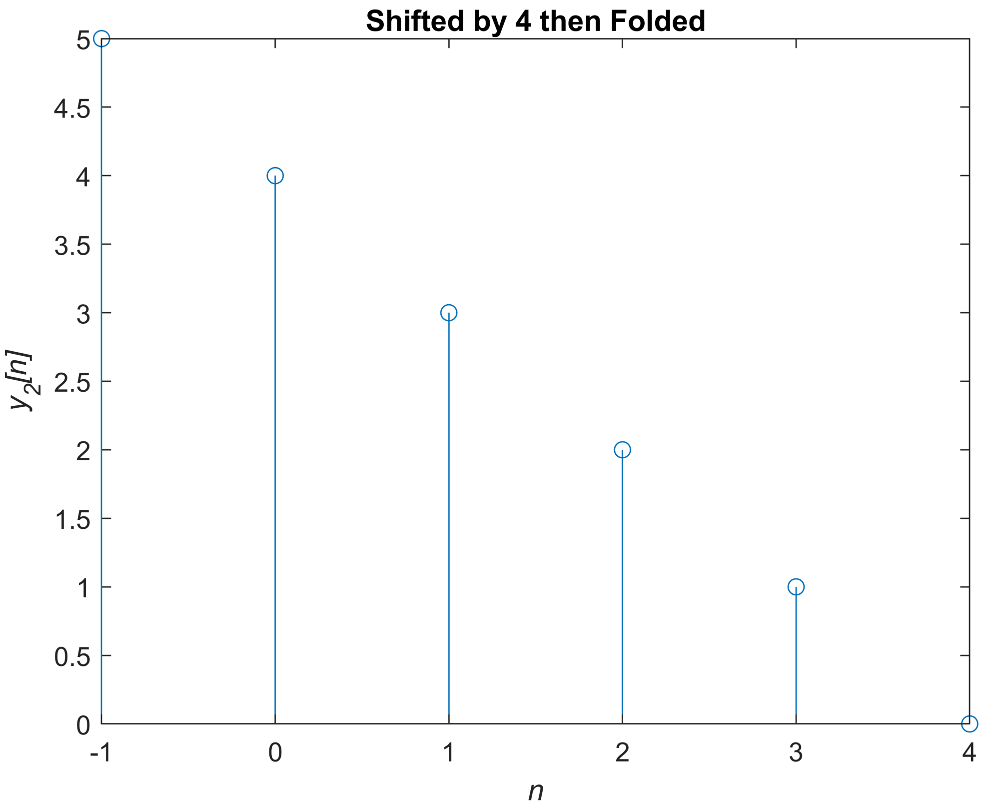
```
ylabel("\it{y_{1}[n]}")
title("Folded then Shifted by 4")
```




---

**(b)** Let $y_2[n]$ be obtained by first shifting $x[n]$ to the left by four samples and then folding the result. Determine and **`stem`** plot $y_2[n]$.

```
x = [0:5];
n = 0:5;
[x1,n1] = shift(x,n,-4);
[y2,ny] = fold(x1,n1);
figure
stem(ny,y2)
xlabel("\it{n}")
ylabel("\it{y_{2}[n]}")
title("Shifted by 4 then Folded")
```




**(c)** From your plots, are $y_1[n]$ and $y_2[n]$ the same signals? Which signal represents the correct $x[-n-4]$ signal?

**Answer:**

From comparing the two outputs, $y_1[n]$ and $y_2[n]$, these signals are not the same because the time shift and folding operations are not commutative.

The correct signal operation applied to $x[n]$ would be $y_1[n]$ because it was Folded then Shifted.

Checking the operations we observe:

$x[n]$ folded = $x[-n]$ and then shifted by 4 = $x[-n-4]$ or $x[-4-n]$

The operations of Shifting and then Folding would result in $x[-n+4]$ or $x[4-n]$ which is the incorrect signal.

## Problem 1.3

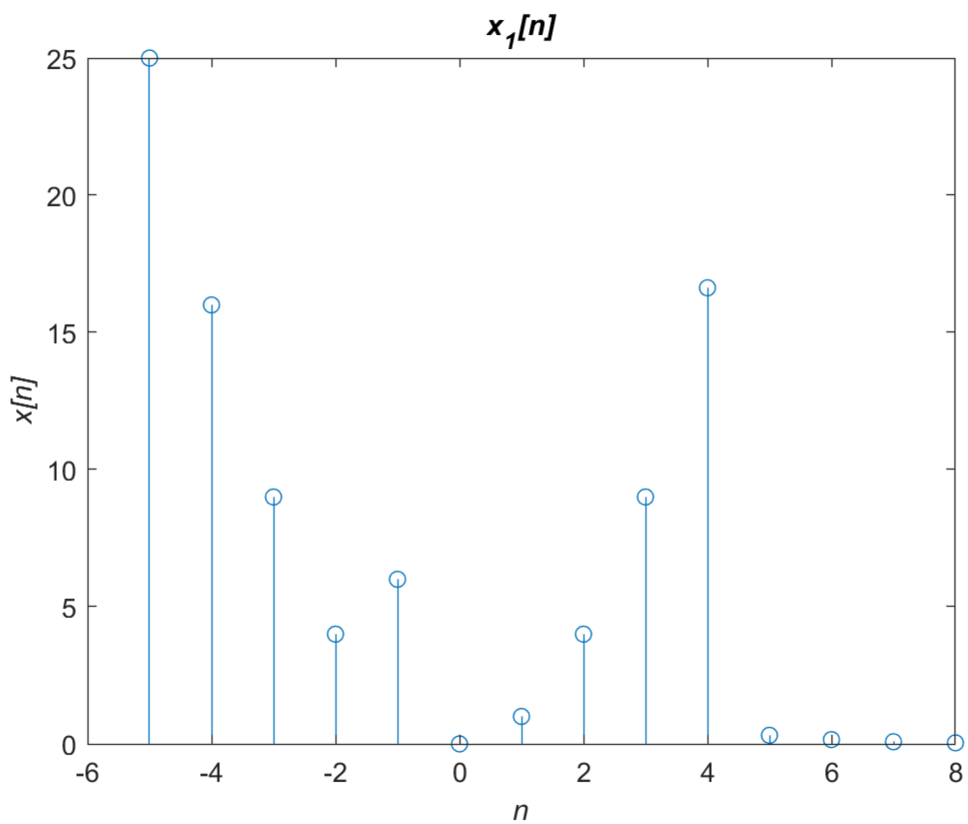
### Text Problem 2.38 (Page 85)

Generate and `stem` plot samples of the following signals.

**(a)** $x_1[n] = 5\delta[n+1] + n^2\big(u[n+5] - u[n-4]\big) + 10(0.5)^n\big(u[n-4] - u[n-8]\big)$

**MATLAB script**:

```
clear; close all; clc;
n = -5:8;
x1 = 5*unitpulse(-5,-1,-1,8)';
x2 = (n.^2).*(unitstep(-5,-5,8) - unitstep(-5,5,8))';
x3 = 10*((0.5).^n).*(unitstep(-5,4,8)');
X1 = x1+x2+x3;
figure
stem(n,X1)
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x_{1}[n]}")
```

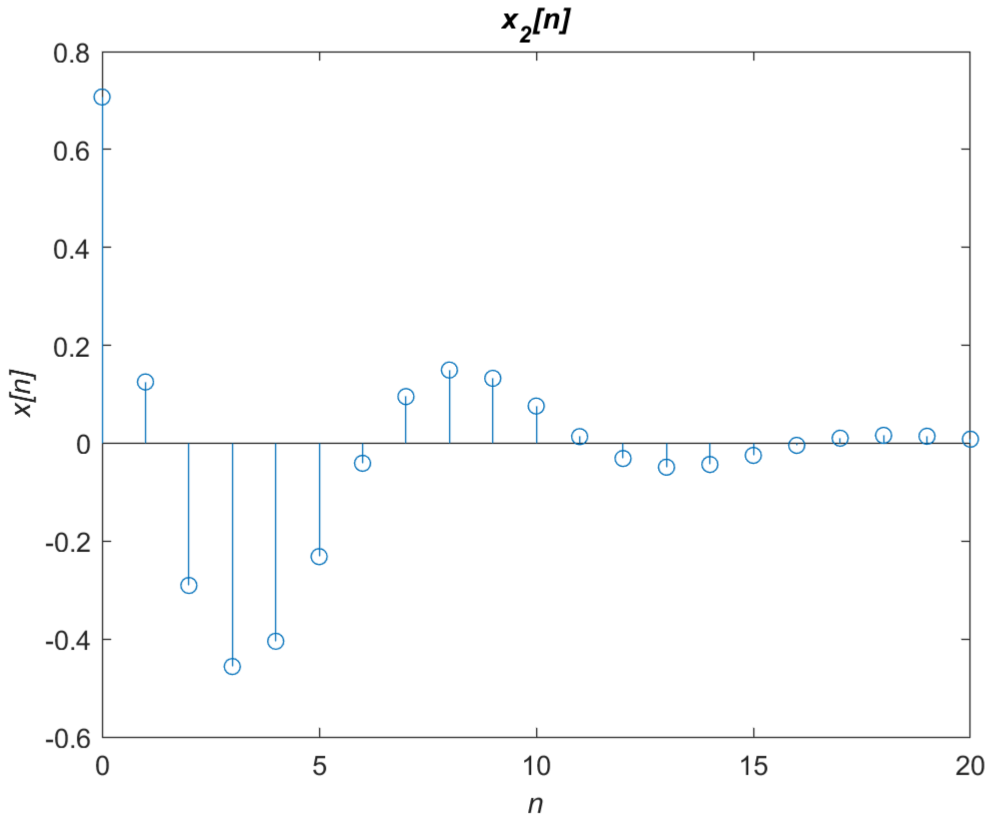



**(b)** $x_2[n] = \begin{cases} (0.8)^n \cos(0.2\pi n + \pi/4), & 0 \le n \le 20 \\ 0, & \text{elsewhere} \end{cases}$

**MATLAB script**:

```
n = 0:20;
x = ((0.8).^n).*cos(0.2.*pi.*n + pi/4);
figure
stem(n,x)
xlabel("\it{n}")
```

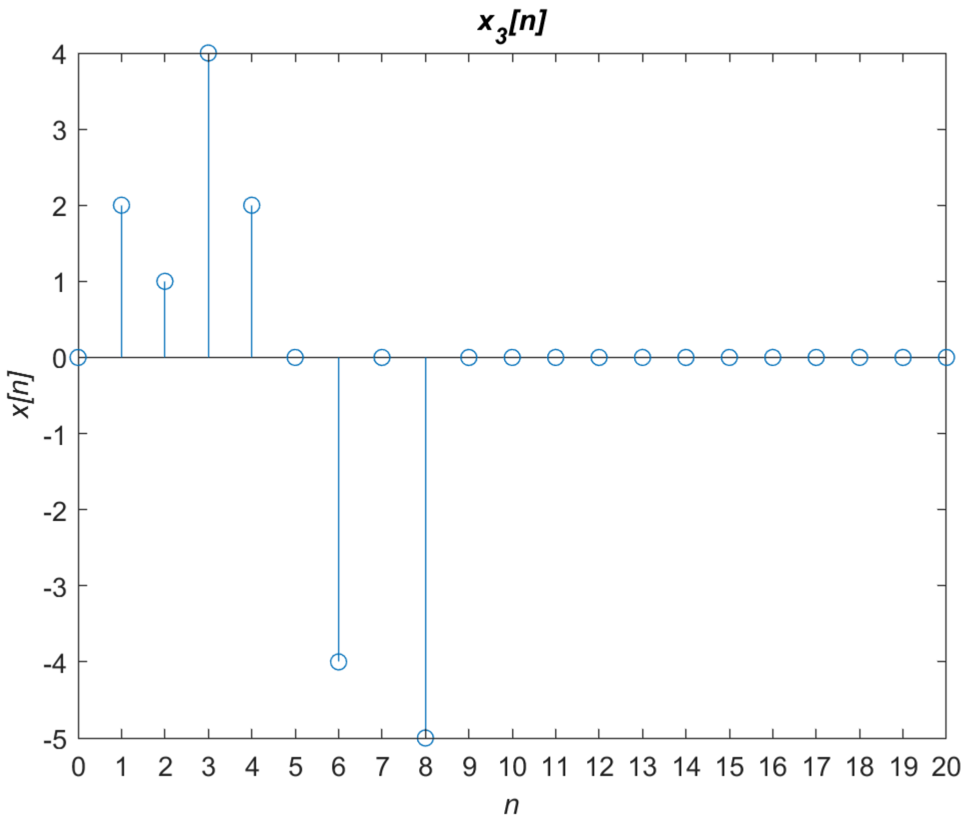
```
ylabel("\it{x[n]}")
title("\it{x_{2}[n]}")
```




(c) $x_3[n] = \sum_{m=0}^{4}(m+1)\{\delta[n-m] - \delta[n-2m]\}$, $0 \le n \le 20$

**MATLAB script**:

```
n = 0:20;
x3 = zeros(5,length(n));
for m = 0:4
    x3(1+m,:) = (m+1)*(delta(0,m,20)'-delta(0,2*m,20)');
end
x3 = x3(1,:) + x3(2,:) + x3(3,:) + x3(4,:) + x3(5,:);
figure
stem(n,x3)
xticks(0:1:20)
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x_{3}[n]}")
```




## Problem 1.4

A real-valued sequence $x_e[n]$ is called an *even* (or *symmetric*) sequence if $x_e[n] = x_e[-n]$, and a real-valued sequence $x_o[n]$ is called an *odd* (or *antisymmetric*) if $x_o[n] = -x_o[-n]$. Then any arbitrary real-valued sequence $x[n]$ can be decomposed into its even and odd parts, that is, $x[n] = x_e[n] + x_o[n]$, where

$$x_e[n] = \frac{1}{2}\left(x[n] + x[-n]\right) \quad \text{and} \quad x_e[n] = \frac{1}{2}\left(x[n] - x[-n]\right) \qquad (1.4.1)$$

**(a)** Prove the above result in $(1.4.1)$.

**Proof**:

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

$$= \frac{1}{2}x[n] + \frac{1}{2}x[-n] + \frac{1}{2}x[n] - \frac{1}{2}x[-n]$$

After cancelling the $\frac{1}{2}x[-n]$ and $-\frac{1}{2}x[-n]$ terms we get

$$x[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n]$$

Thus,

$$x[n] = x[n]$$

---

**(b)** Design a MATLAB function **EvenOdd** that accepts an arbitrary real-valued sequence and decomposes it into its even and odd parts by implementing $(1.4.1)$.

**MATLAB function**: Enter your function code below after the comments for the TA to evaluate and grade. Create your function at the end of this file for it to execute properly.

```
function [xe,xo,m] = EvenOdd(x,n)
% Real-valued sequence decomposition into even and odd parts
% [xe,xo,m] = EvenOdd(x,n)
% Output variables:
% xe: Even part of x[n]
% xo: Odd part of x[n]
%  m: Index support of even and odd parts
% Input variables:
%  x: Real-valued input sequence
%  n: Index support of x
%
% Enter your code below
if any(imag(x) ~= 0)
    error("x is not a real-valued sequence")
end

m = -fliplr(n);
m1 = min([m,n]);
m2 = max([m,n]);
m = m1:m2;
nm = n(1)-m(1); n1 = 1:length(n);
x1 = zeros(1,length(m));
x1(n1+nm) = x;
x = x1;
xe = (1/2)*(x + fliplr(x));
xo = (1/2)*(x - fliplr(x));

end
```

---

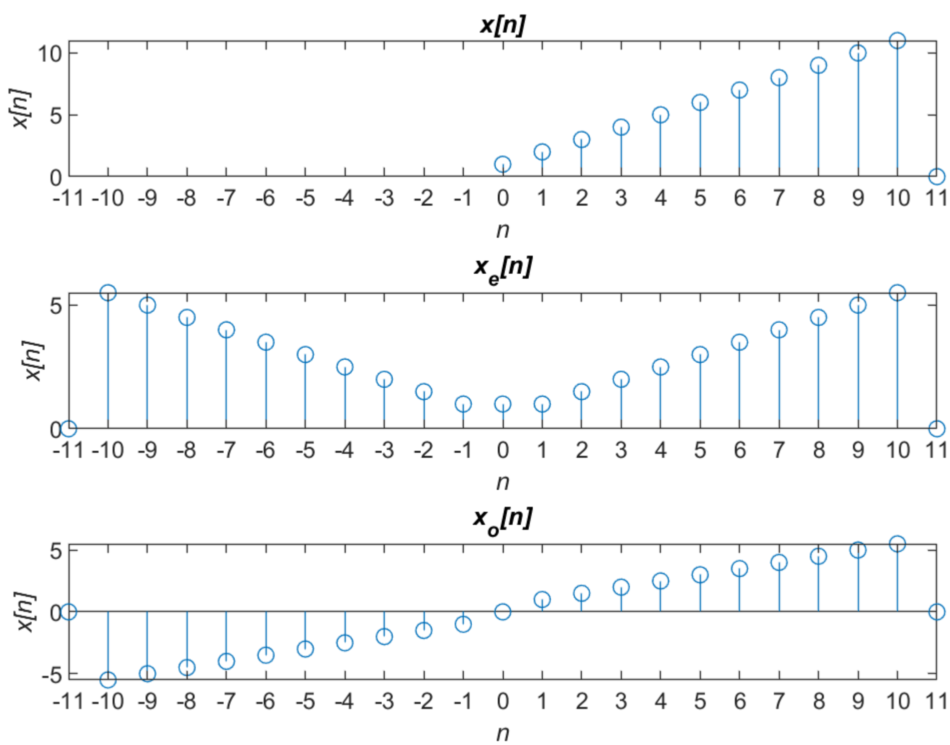
**(c)** Using your **EvenOdd** function, decompose the following sequence

$$x[n] = \{\underset{\uparrow}{1}, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$$

into its even and odd parts and **stem**-plot $x[n]$, $x_e[n]$, and $x_o[n]$ over $-11 \le n \le 11$ in one figure window using $3 \times 1$ **subplot**s.

**MATLAB script**:

```
clc; close all; clear;
x = [1:11,0]; n = 0:11;
[xe,xo,m] = EvenOdd(x,n);
figure
subplot(3,1,1)
stem(n,x)
xticks(-11:1:11)
xlim(([-11 11]))
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x[n]}")
subplot(3,1,2)
stem(m,xe)
xticks(-11:1:11)
xlim(([-11 11]))
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x_{e}[n]}")
subplot(3,1,3)
stem(m,xo)
xticks(-11:1:11)
xlim(([-11 11]))
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x_{o}[n]}")
```



## Problem 1.5

**Text problem 2.25 (Page 83)**

Consider the finite duration sequences $x[n] = u[n] - u[n-N]$ and $h[n] = n\left(u[n] - u[n-M]\right)$ with $M \le N$.

---

**(a)** Determine an analytical expression for the sequence $y[n] = h[n] * x[n]$.

**Solution**:

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

when $n \le 0$ there is no overlap in the convolution

for $0 \le n \le M-1$ there is overlap from 0 to n, hence

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = \sum_{k=1}^{n} k = \frac{n(n+1)}{2} \quad ,0 < n \le M-1$$

for $M-1 < n \le N-1$ there is overlap from 0 to M - 1, hence

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = \sum_{k=0}^{M-1} k = \frac{(M-1)(M-1+1)}{2} = \frac{M(M-1)}{2} \quad ,M-1 < n \le N-1$$

for $N-1 < n \le N+M-2$, there is overlap from $n-N+1$ to $M-1$, hence

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = \sum_{k=0}^{M-1} k - \sum_{k=0}^{n-N} k = \frac{(M-1)(M-1+1)}{2} - \frac{(n-N)(n-N+1)}{2} = \frac{M(M-1)-(n-N)(n-N+1)}{2} \quad ,N$$

finally, when $n > N+M-2$, there is no overlap.

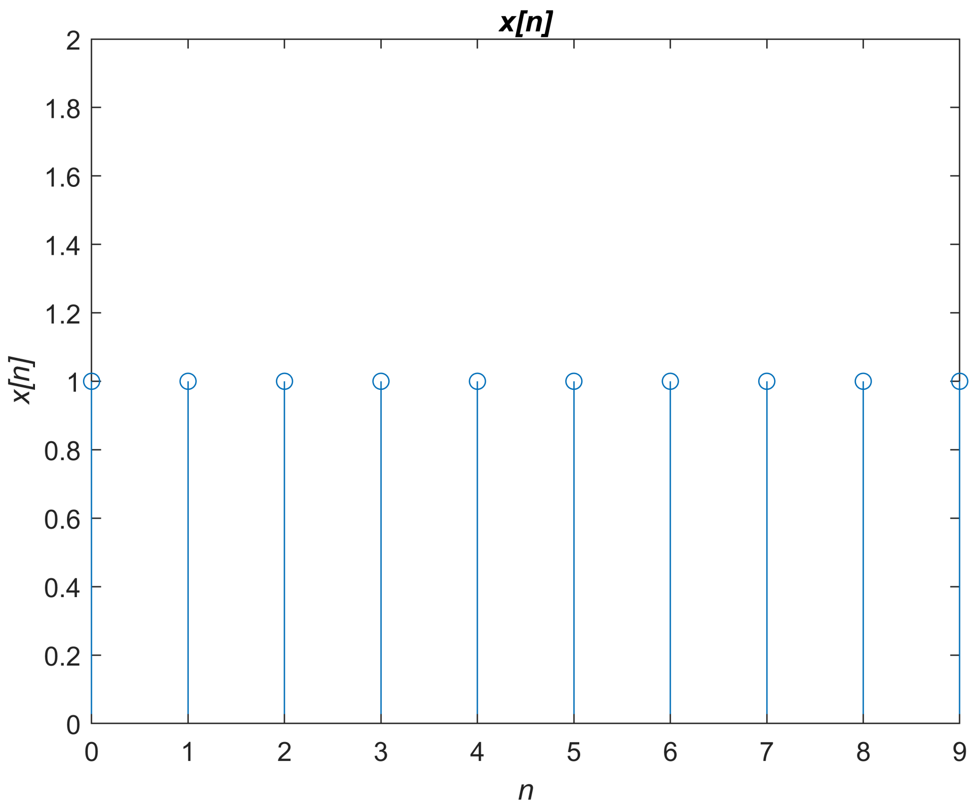
---

**(b)** Verify the result in (a) for $N = 10$ and $M = 5$ using the function **`y = conv(h,x)`**.

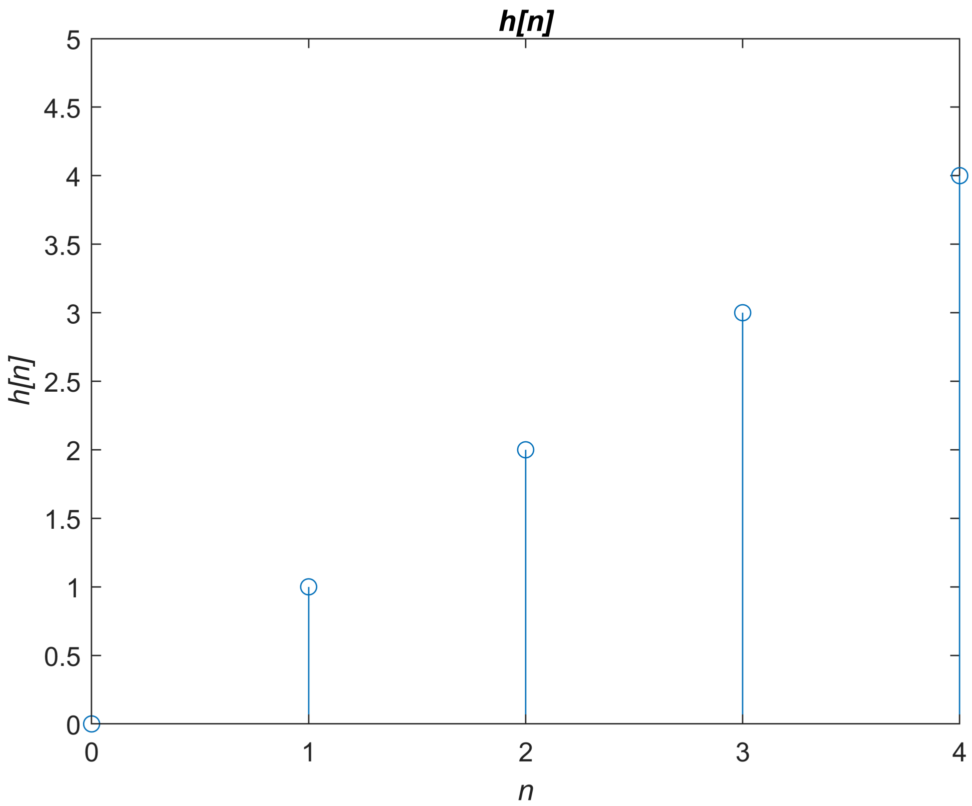
**Solution**: The required MATLAB script and the resulting plot are given below. From this above plot, the result in (a) is verified.

```
clc; close all; clear;
N = 10; M = 5;
for n = 0:N-1
    x(n+1) = 1;
end
for n = 0:M-1
    h(n+1) = n;
end
y = conv(x,h);
n = 0:N-1;
```

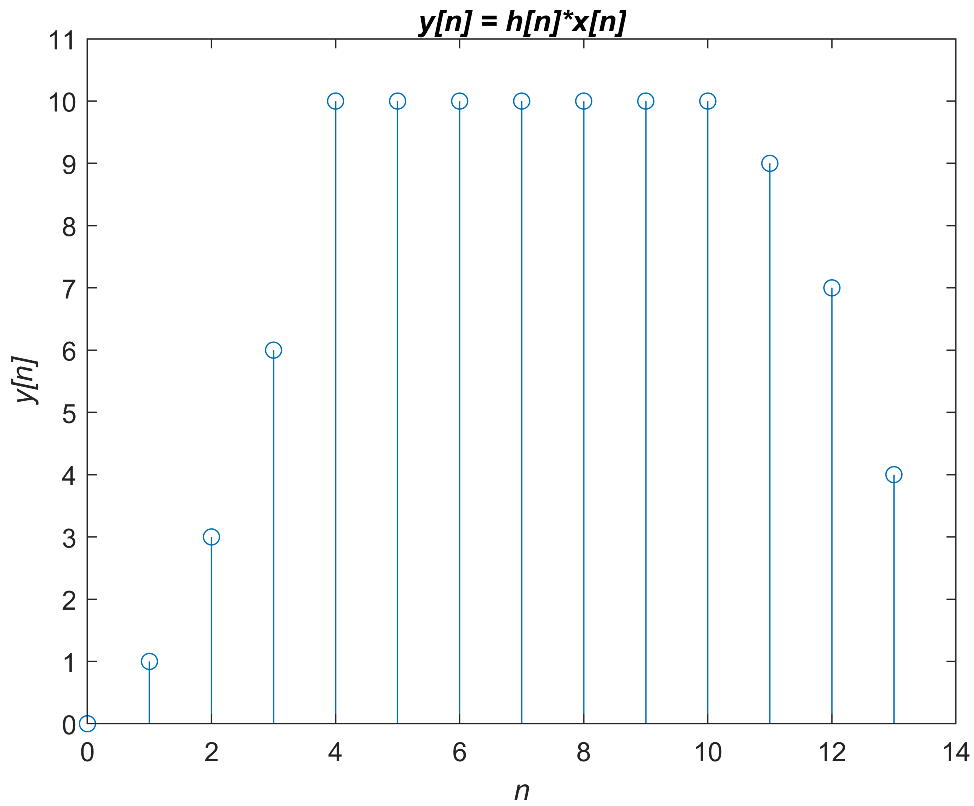
```
n1 = 0:M-1;
n2 = 0:N+M-2;
figure
stem(n,x)
ylim([0 2])
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{x[n]}")
```




```
figure
stem(n1,h)
xticks(0:1:4)
ylim([0 5])
xlabel("\it{n}")
ylabel("\it{h[n]}")
title("\it{h[n]}")
```




```
figure
stem(n2,y)
ylim([0 11])
xlabel("\it{n}")
ylabel("\it{y[n]}")
title("\it{y[n] = h[n]*x[n]}")
```




## Problem 1.6

**Text problem 2.34 (Page 84)**

A system is described by the difference equation

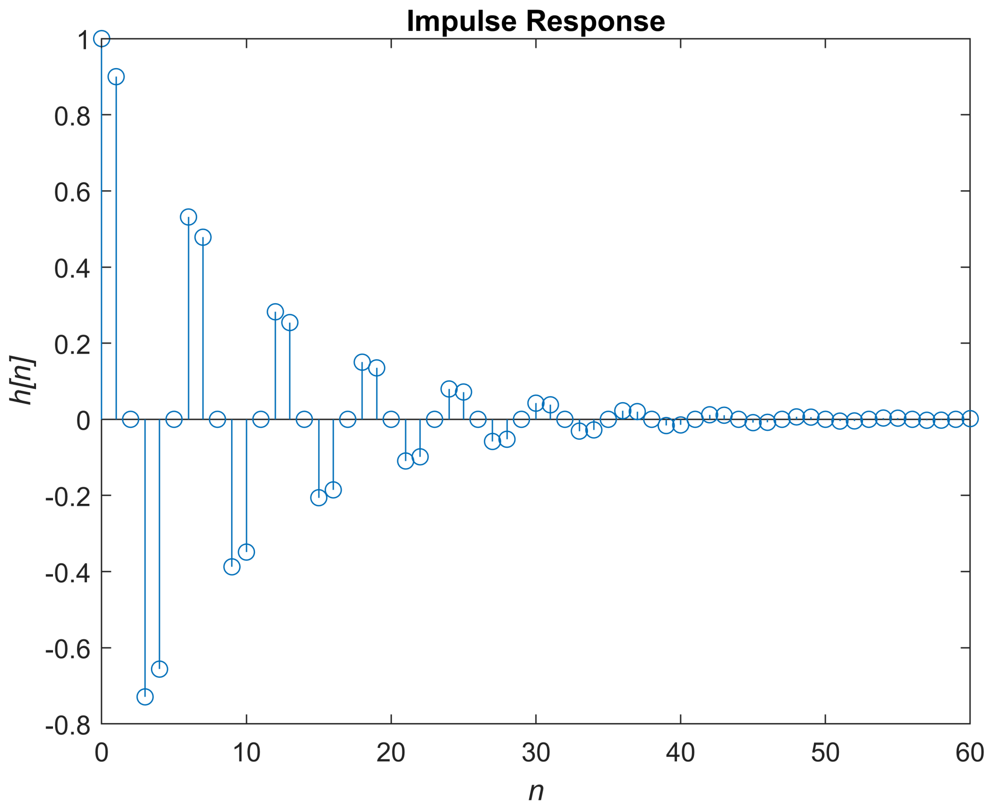
$$y[n] = x[n] + 0.9y[n-1] - 0.81y[n-2].$$

Using MATLAB determine and **`stem`** plot the following responses over $0 \le n \le 60$.

**(a)** Impulse response $h[n] = \mathrm{LTI}\{\delta[n]\}$ of the system.

**MATLAB script**:

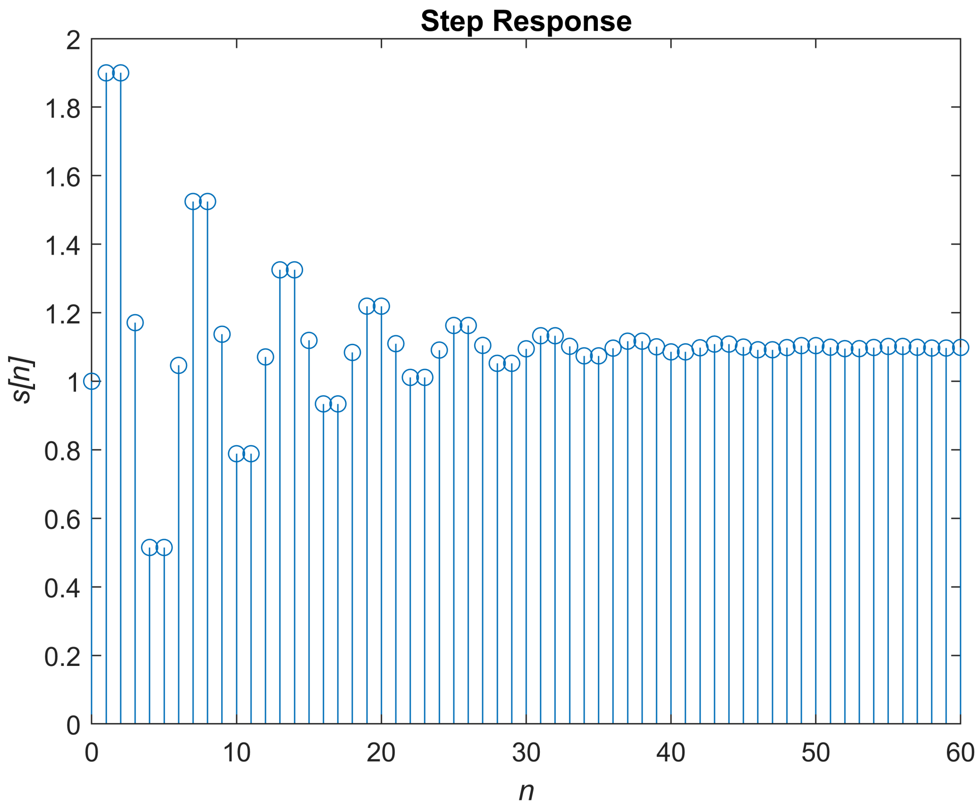
```
clc; close all; clear;
n = 0:60; b = 1; a = [1 -0.9 0.81];
x = (n==0);
h = filter(b,a,x);
figure
stem(n,h)
xlabel("\it{n}")
ylabel("\it{h[n]}")
title("Impulse Response")
```




**(b)** Step response $s[n] = \mathrm{LTI}\{u[n]\}$ of the system.

**MATLAB script**:

```
x = (n>=0);
s = filter(b,a,x);
figure
stem(n,s)
xlabel("\it{n}")
ylabel("\it{s[n]}")
title("Step Response")
```




**(c)** Identify the transient and steady-state responses in (b).

**Answer**:

Looking at the step response of the system in part (b), we can see that the transient response occurs from about $0 \le n \le 50$

The initial oscillations start to settle around $n = 50$ and the remaining part of the step response is the steady-state response of the system.

## Problem 1.7

**Text problem 2.40 (Page 85)**

Consider the following discrete-time system

$$y[n] = H\{x[n]\} = 10x[n]\cos(0.25\pi n + \theta)$$

where $\theta$ is a constant.

**(a)** Determine if the system is linear.

**Solution**:

To determine if the system is linear, we can use the principle of superposition to determine if the output can be represented by a sum of independent inputs

$$y[n] = T\{a_1x_1[n] + a_2x_2[n]\} = 10(a_1x_1[n] + a_2x_2[n])\cos(0.25\pi n + \theta)$$

$$= a_1 10x_1[n]\cos(0.25\pi n + \theta) + a_2 10x_2[n]\cos(0.25\pi n + \theta)$$

$$= a_1T\{x_1[n]\} + a_2T\{x_2[n]\}$$

Hence, this system is linear.

---

**(b)** Determine if the system is time-invariant.

**Solution**:

To determine if the system is time-invarient, we will determine the response $y_k(n)$ to the shifted input sequence

$$y(n) = L[x(n)] = 10x[n]\cos(0.25\pi n + \theta)$$

$$y_k(n) = L[x(n-k)] = 10x(n-k)\cos(0.25\pi n + \theta)$$

then the shifted output is

$$y_k(n-k) = 10x(n-k)\cos(0.25\pi(n-k) + \theta) \neq y_k(n)$$

Which shows the shifted output is not equal to the shifted input

Hence, the system is not time-invarient.

---

## Problem 1.8

Let $x[n] = (0.8)^n u[n]$, $h[n] = (-0.9)^n u[n]$, and $y[n] = h[n] * x[n]$. Since $x[n]$ and $h[n]$ sequences are of semi-infinite duration, we will obtain $y[n]$ using three different approaches.

---

**(a)** Analytical approach: Determine $y[n]$ analytically. Plot the first 51 samples of $y[n]$ using the `stem` function.

**Solution**:

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

$$= \sum_{k=-\infty}^{\infty} (0.8)^k u(k)(-0.9)^{n-k} u(n-k)$$

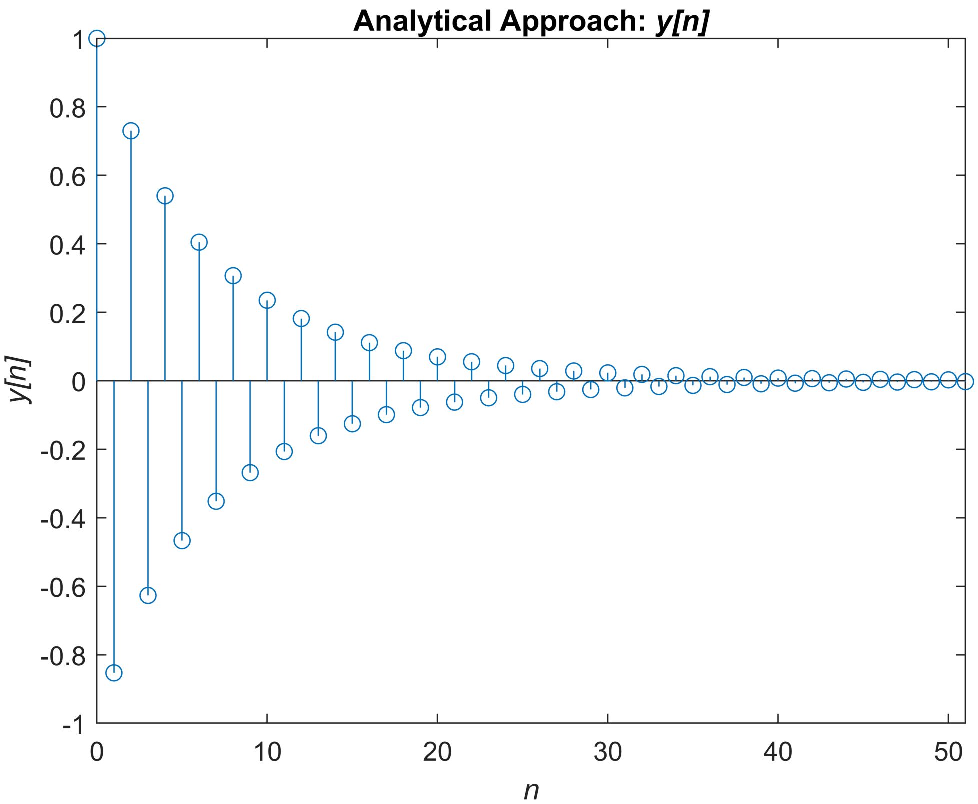
$$= \sum_{k=0}^{n} (0.8)^k (-0.9)^{n-k}$$

$$= (-0.9)^n \left( \sum_{k=0}^{n} (0.8)^k (-0.9)^{-k} \right)$$

$$= (-0.9)^n \sum_{k=0}^{n} \left( \frac{(0.8)}{(-0.9)}^{k} \right) = (-0.9)^n \left( \frac{1 + \left(\frac{(0.8)}{(0.9)}\right)^{n+1}}{1 + \frac{(0.8)}{(0.9)}} \right)$$

**MATLAB script**:

```
clc; close all; clear;
n = 0:51;
u = (n>=0);
x = (0.8).^n.*u;
h = (-0.9).^n.*u;
%y = conv(x,h);
%n = 0:2*length(n)-2;
num = 1 + (0.8/0.9).^(n+1);
denom = 1 + (0.8/0.9);
y = ((-0.9).^n).*(num./denom);
figure
stem(n,y)
xlim([0 51])
xlabel("\it{n}")
ylabel("\it{y[n]}")
title("Analytical Approach: \it{y[n]}")
```

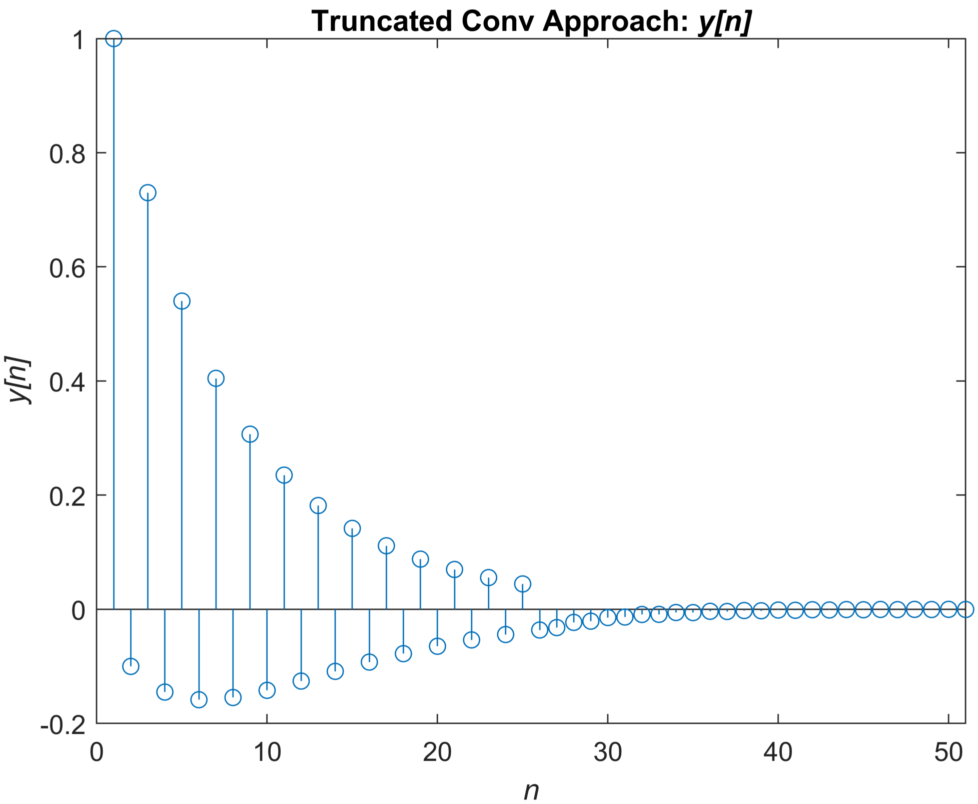



---

**(b)** The **'conv'** approach: Truncate $x[n]$ and $h[n]$ to 26 samples. Use the **conv** function to compute $y[n]$ which should have 51 samples. Plot $y[n]$ using the **stem** function.

**MATLAB script**:

```
x1 = x(1:26); h1 = h(1:26);
y = conv(x1,h1);
n = 1:length(y);
figure
stem(n,y)
xlim([0 51])
xlabel("\it{n}")
ylabel("\it{y[n]}")
title("Truncated Conv Approach: \it{y[n]}")
```




**(c)** The **'filter'** approach: Determine filter coefficients for the given impulse response $h[n]$ and use them in the **filter** function to determine the first 51 samples of $y[n]$. Plot $y[n]$ using the **stem** function.

**Solution**:

This LTI, given the impulse response $h[n]$, can be described by a difference equation from the $h[n]$ expression:

$(-0.9)h(n-1) = (-0.9)(-0.9)^{n-1}u(n-1) = (-0.9)^n u(n-1)$

or $h(n) + (0.9)h(n-1) = (-0.9)^n u(n) + (0.9)^n u(n-1) = (-0.9)^n[u(n) - u(n-1)] = (-0.9)^n \delta(n) = \delta(n)$

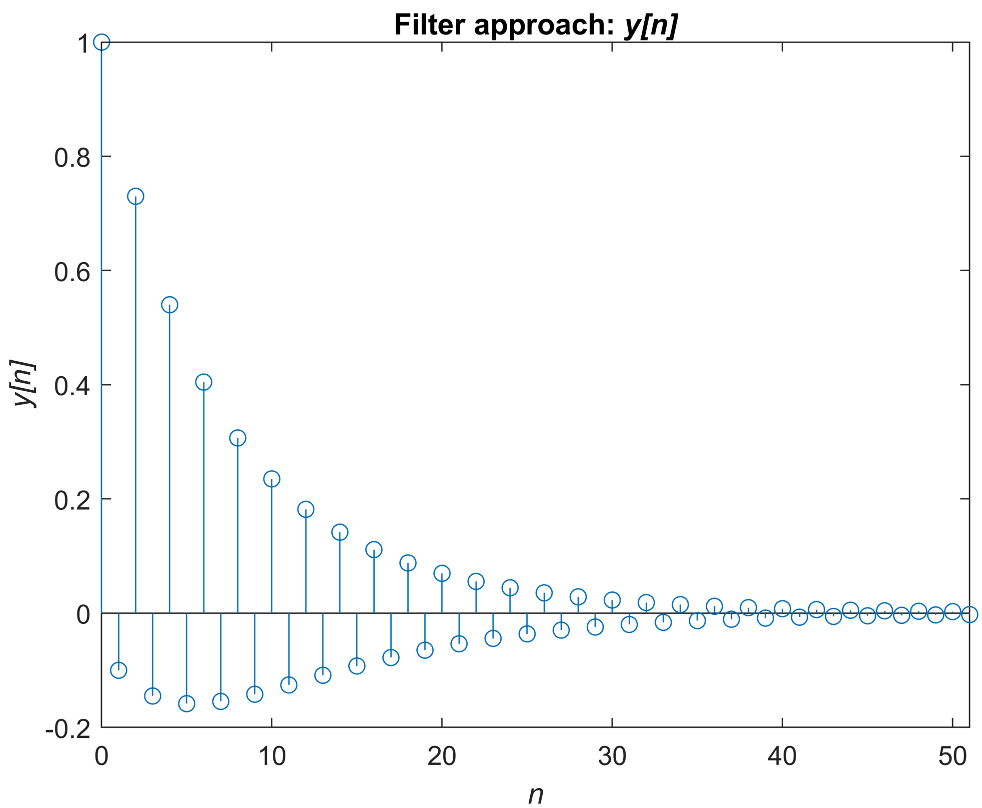
By definition, $h(n)$ is the output of the LTI system when the input is $\delta(n)$. Hence substituting $x(n)$ for $\delta(n)$ and $y(n)$ for $h(n)$, the difference equation is:

$y(n) + 0.9y(n-1) = x(n)$

**MATLAB script**:

```
b = 1; a = [1 0.9];
y1 = filter(b,a,x);
n1 = 0:51;
figure
stem(n1,y1)
xlim([0 51])
xlabel("\it{n}")
```

```
ylabel("\it{y[n]}")
title("Filter approach: \it{y[n]}")
```




**(d)** Compare the three solution approaches over the first 51 samples and comment on your results. Which approach gives an incorrect convolution result and why?

**Solution**:

The incorrect convolution result is observed from part (b) where we truncated both the input signal and impulse response to be half the amount of the sample sequences. The result of the convolution in part (b) has less resolution towards the end of the attentuating exponential signal in the output because of the truncating of the input and impulse response. The signals are short sequences because of the truncation and thus the convolution result reaches zero faster than the other approaches, making it the incorrect approach.

## Problem 1.9

A simple *digital differentiator* is given by

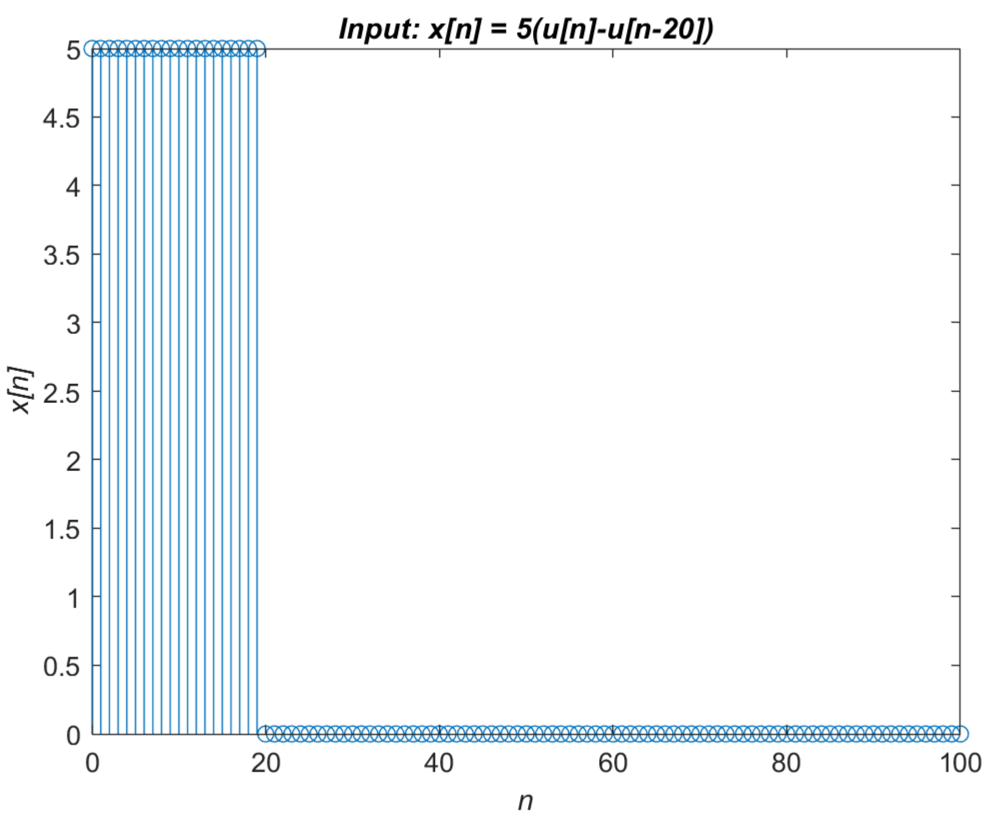
$$y[n] = x[n] - x[n-1]$$

which computes a backward first-order difference of the input sequence. Implement this differentiator on the following sequences, and plot the corresponding results.

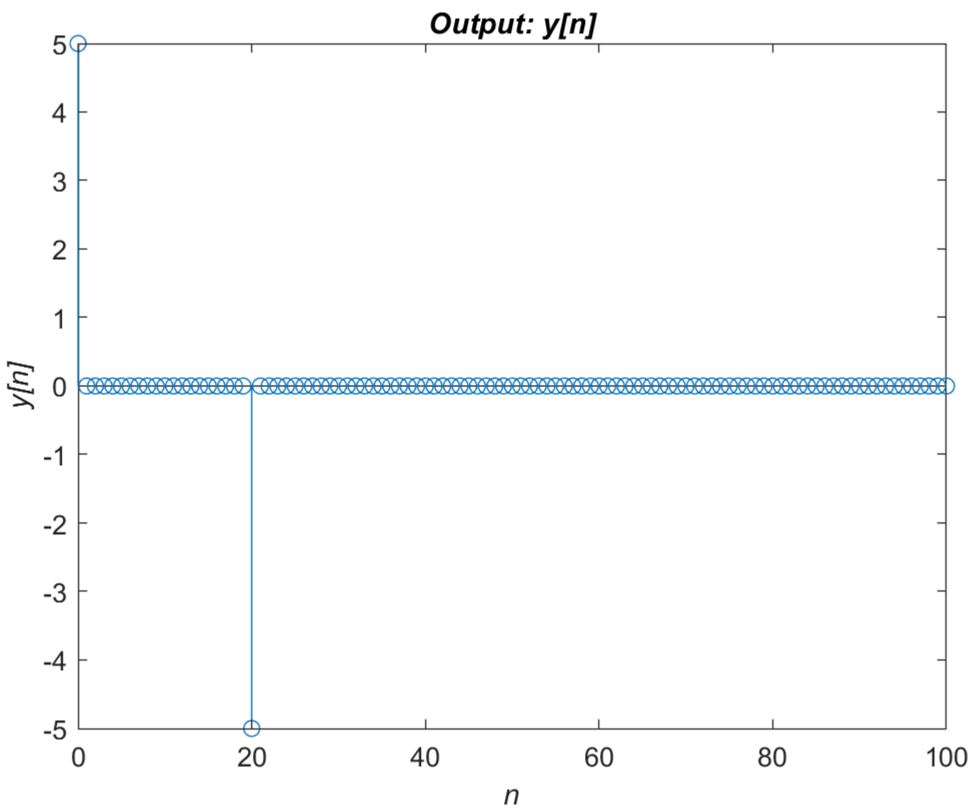
**(a)** A rectangular pulse: $x[n] = 5\left(u[n] - u[n-20]\right)$

**MATLAB script**:

```
clc; close all; clear;
n = 0:100;
b = [1 -1]; a = 1;
x = 5*((n>=0)-(n>=20));
y = filter(b,a,x);
figure
stem(n,x)
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{Input: x[n] = 5(u[n]-u[n-20])}")
```



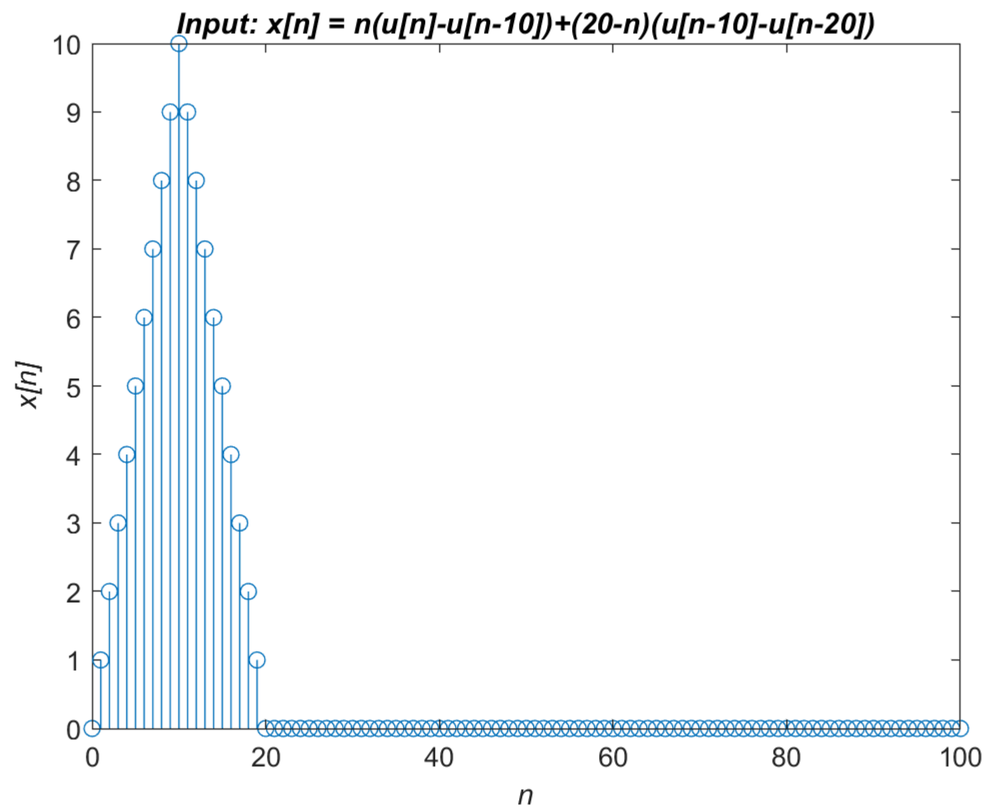

```
figure
stem(n,y)
xlabel("\it{n}")
ylabel("\it{y[n]}")
title("\it{Output: y[n]}")
```



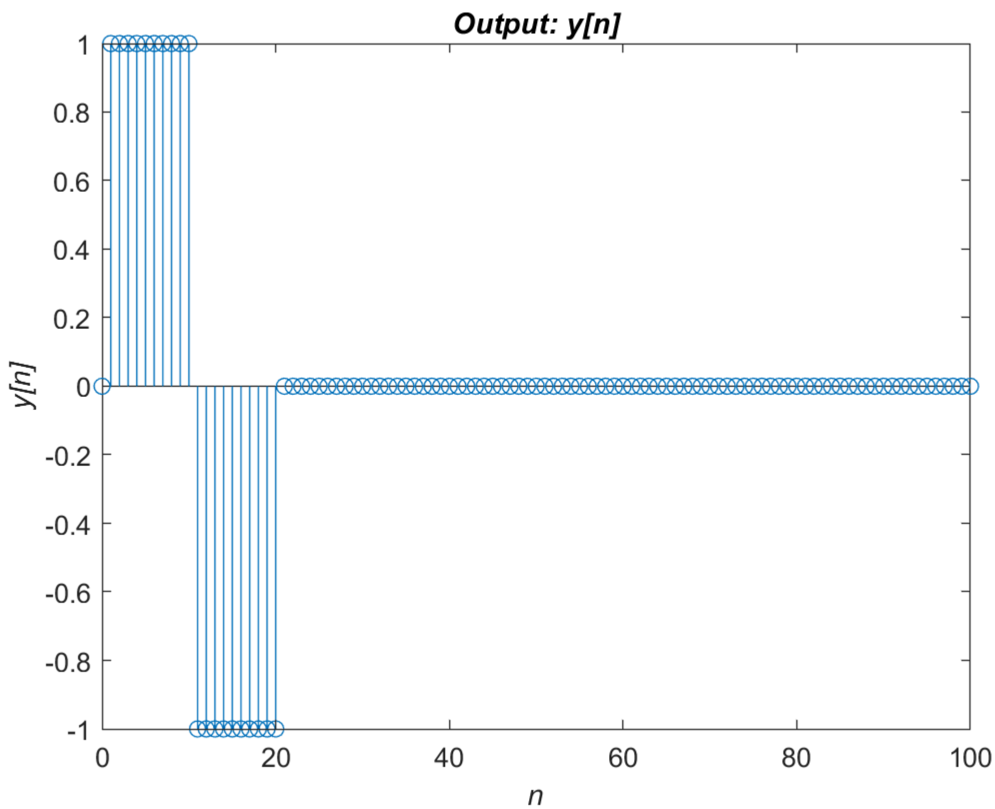

**(b)** A triangular pulse: $x[n] = n\big(u[n] - u[n-10]\big) + (20-n)\big(u[n-10) - u[n-20)\big)$

**MATLAB script**:

```
x1 = n.*((n>=0)-(n>=10)) + (20-n).*((n>=10)-(n>=20));
y1 = filter(b,a,x1);
figure
stem(n,x1)
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{Input: x[n] = n(u[n]-u[n-10])+(20-n)(u[n-10]-u[n-20])}")
```




```
figure
stem(n,y1)
xlabel("\it{n}")
ylabel("\it{y[n]}")
title("\it{Output: y[n]}")
```

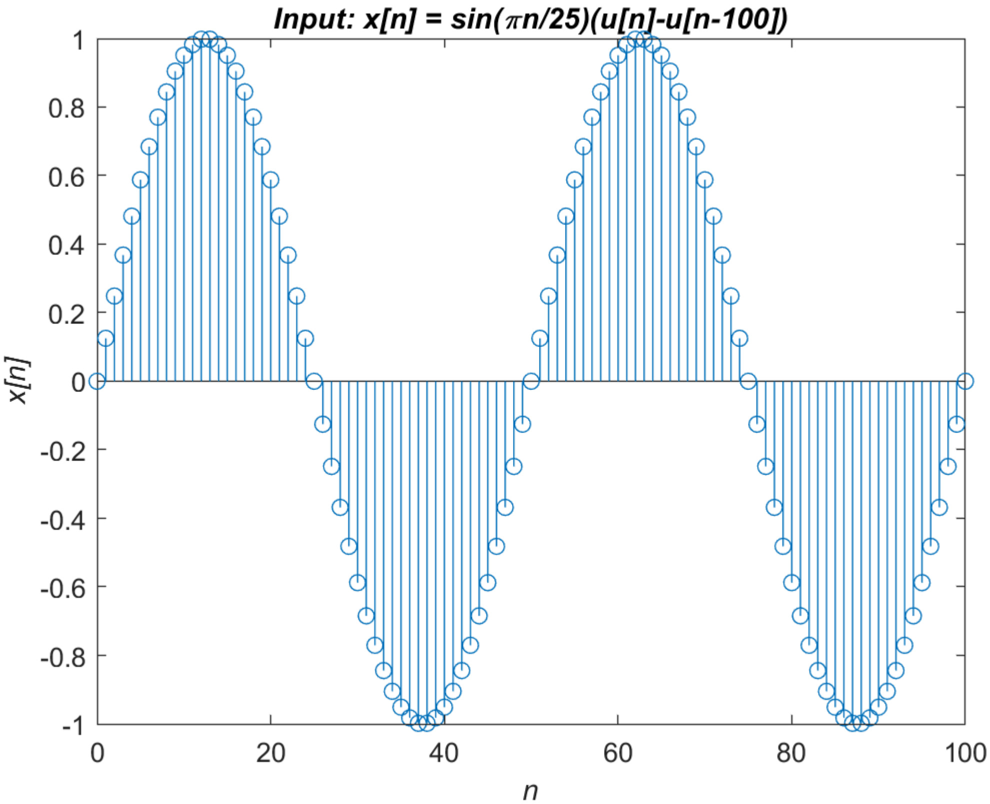



---

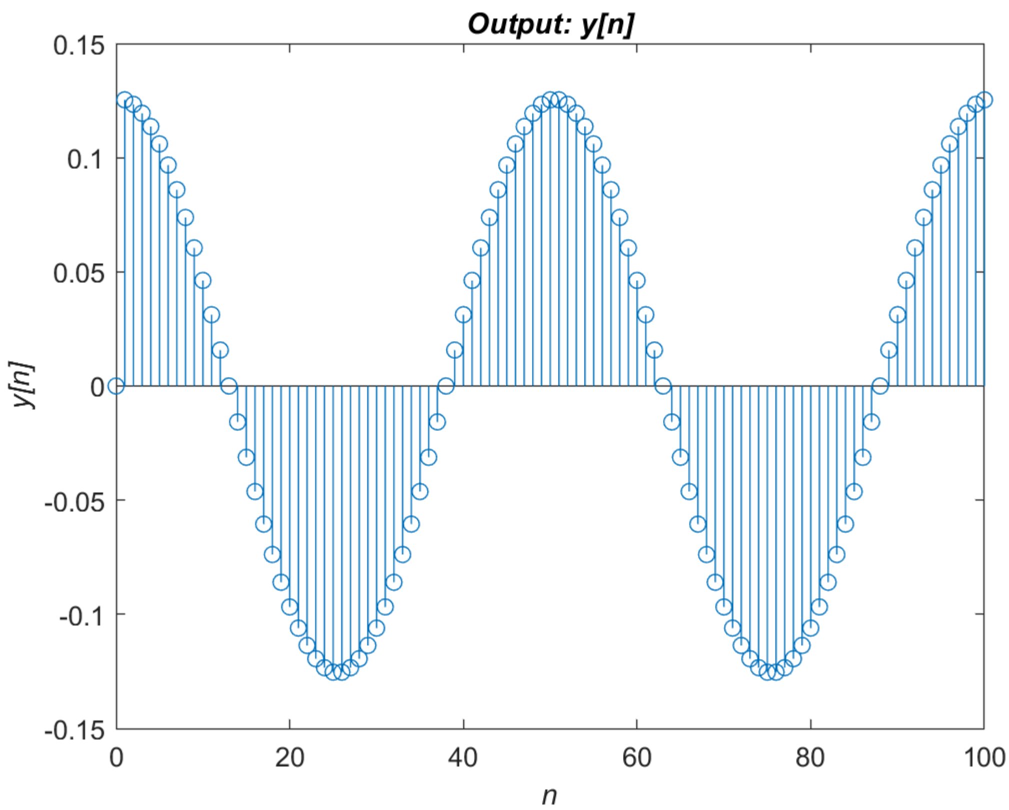
**(c)** A sinusoidal pulse: $\sin\left(\frac{\pi n}{25}\right)(u[n] - u[n-100])$

**MATLAB script**:

```
x2 = sin(pi.*n.*(1/25)).*((n>=0)-(n>=100));
y2 = filter(b,a,x2);
figure
stem(n,x2)
xlabel("\it{n}")
ylabel("\it{x[n]}")
title("\it{Input: x[n] = sin(\pin/25)(u[n]-u[n-100])}")
```




```
figure
stem(n,y2)
xlabel("\it{n}")
ylabel("\it{y[n]}")
title("\it{Output: y[n]}")
```




**(d)** Comment on the appropriateness of this simple differentiator.

**Answer**:

For each case of input sequence the simple differentiator output the slope of the input sequence based on its previously sampled input.

For the rectangluar pulse, the output gave an impulse response of the change in slope when the pulse rose and fell from/to zero.

For the triangular pulse, the differentiator started to output a periodic square-wave oscillation.

Finally, for the sinusoidal input sequence, the simple differentiator provided a scaled and phase shifted sequence of the input as the output signal.

This simple differentiator could be appropriate in situations where the input signal needs to be transformed or reshaped into another signal shape, such as with the case

of the triangle pulse converting to a square-wave pulse. Such a transformation could be appropriate for audio synthesis experimentation.

## Problem 1.10

**Text Problem 2.51 (Page 87)**

The digital echo system described in Example 2.8 can be represented by a general impulse response

$$h[n] = \sum_{k=0}^{\infty} a_k \delta[n - kD]$$

To remove these echoes, an inverse system is needed and one implementation of such a system is given by

$$g[n] = \sum_{k=0}^{\infty} b_k \delta[n - kD]$$

such that $h[n] * g[n] = \delta[n]$.

---

**(a)** Determine the algebraic equations that the successive $b_k$'s must satisfy.

**Solution**:

$$\sum_{k=0}^{\infty} a_k \delta[n - kD] * \sum_{m=0}^{\infty} b_m \delta[n - mD] = h[n] = \sum_{k=0}^{\infty}\sum_{m=0}^{\infty} a_k b_{n-m} \delta[n - kD - mD] = \delta[n]$$

Using this expression for the convolutional sum of the two echo responses, we know that at $n = 0$ the summation should equal to $\delta[n]$ and for everywhere else, the result should be zero. So, we can solve for the first few results of the convolution and then express the result in terms of arbitray $a_k$ and $b_k$ values. Hence,

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

$$h[n] * g[n] = a_0 b_0 + (a_0 b_1 + a_1 b_0) + (a_0 b_2 + a_1 b_1 + a_2 b_0) + ... + a_k b_k$$

---

**(b)** Solve the above equations for $b_0$, $b_1$, and $b_2$ in terms of $a_k$.

**Solution**:

$$a_0 b_0 = 1 \rightarrow b_0 = \frac{1}{a_0}$$

$$a_0 b_1 + a_1 b_0 = 0 \rightarrow a_0 b_1 + \frac{a_1}{a_0} = 0 \rightarrow b_1 = \frac{-a_1}{a_0^2}$$

$$a_0 b_2 + a_1 b_1 + a_2 b_0 = 0 \rightarrow a_0 b_2 - \left(\frac{a_1}{a_0}\right)^2 + \frac{a_2}{a_0} = 0 \rightarrow b_2 = \frac{1}{a_0}\left(-\frac{a_2}{a_0} + \frac{a_1^2}{a_0^2}\right)$$

---

**(c)** For $a_0 = 1$, $a_1 = 0.5$, $a_2 = 0.25$, and all other $a_k$'s equal to zero, determine $g[n]$.

**Solution**:

$$b_0 = \frac{1}{a_0} = 1$$

$$b_1 = \frac{-a_1}{a_0^2} = -\frac{1}{2}$$

$$b_2 = \frac{1}{a_0}\left(-\frac{a_2}{a_0} + \frac{a_1^2}{a_0^2}\right) = \frac{1}{1}\left(-\frac{0.25}{1} + \frac{0.5^2}{1^2}\right) = -0.25 + 0.25 = 0$$

Thus,

$$g[n] = \sum_{k=0}^{\infty} b_k \delta[n-kD] = \delta[n] - \frac{1}{2}\delta[n-D]$$

**MATLAB script**:

```
clc; close all; clear;
a0 = 1; a1 = 0.5; a2 = 0.25;
b0 = 1/a0
```

```
b0 = 1
```

```
b1 = -a1/(a0^2)
```

```
b1 = -0.5000
```

```
b2 = (1/a0)*(-a2/a0 + (a1^2)/(a0^2))
```

```
b2 = 0
```

Create your MATLAB function **EvenOdd** below.

```
function [xe,xo,m] = EvenOdd(x,n)
% Real-valued sequence decomposition into even and odd parts
% [xe,xo,m] = EvenOdd(x,n)
% Output variables:
% xe: Even part of x[n]
% xo: Odd part of x[n]
%  m: Index support of even and odd parts
% Input variables:
%  x: Real-valued input sequence
%  n: Index support of x
%
if any(imag(x) ~= 0)
    error("x is not a real-valued sequence")
end

m = -fliplr(n);
m1 = min([m,n]);
m2 = max([m,n]);
m = m1:m2;
```

```
nm = n(1)-m(1); n1 = 1:length(n);
x1 = zeros(1,length(m));
x1(n1+nm) = x;
x = x1;
xe = (1/2)*(x + fliplr(x));
xo = (1/2)*(x - fliplr(x));
end
```