EECE5666 (DSP) : Homework-1 Solutions

Table of Contents

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

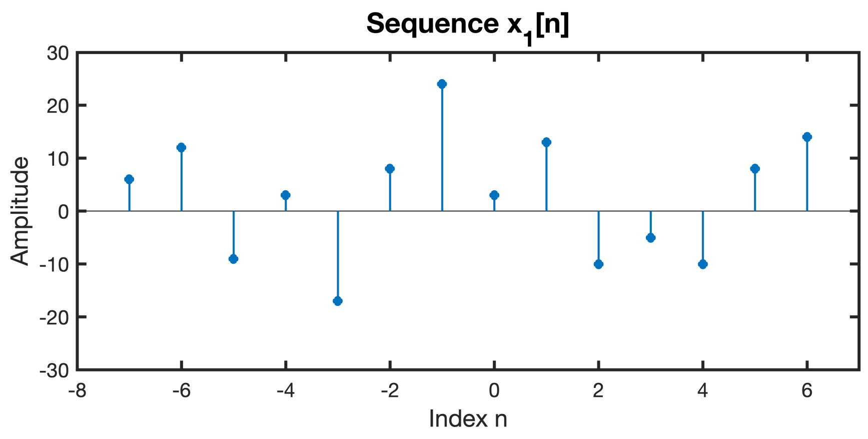
Let $x[n] = \{2, 4, -3, \underset{\uparrow}{1}, -5, 4, 7\}$. Using the **timealign,shift,** and **stem** functions, generate and plot samples of the following sequences.

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

MATLAB script:

```
clc; close all; clear;
n = (-3:3); x = [2,4,-3,1,-5,4,7];
[x11,n11] = shift(x,n,3); % shift by 3
[x12,n12] = shift(x,n,-4); % shift by -4
[x11,x12,n112] = timealign(x11,n11,x12,n12); % Time-align first two components
x112 = 2*x11+3*x12; % Add first two components
[x112,x13,n1] = timealign(x112,n112,x,n); % Time-align with the last component
x1 = x112-x13; % Final sequence
figure('position',[1,1,7,3]*72);
stem(n1,x1,'filled','markersize',4,'linewidth',1);
```

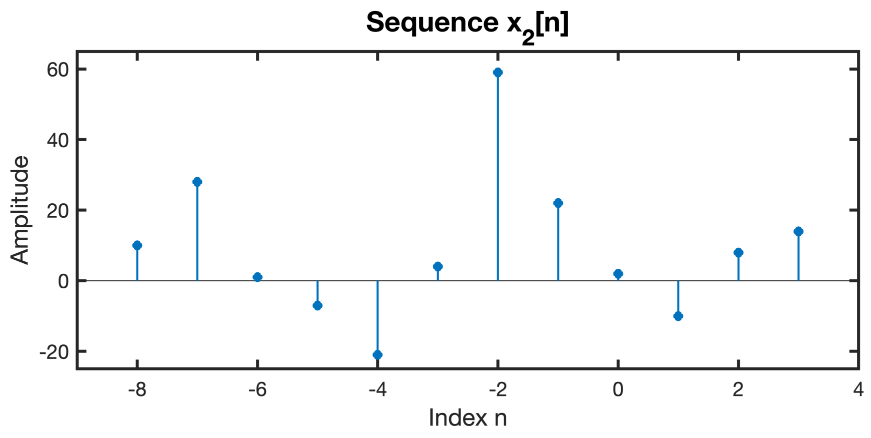
```
xlabel('Index n'); ylabel('Amplitude');
title('Sequence x_1[n]'); axis([-8,7,-30,30]);
```



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

MATLAB script:

```
[x21,n21] = shift(x,n,-4); % shift by -4
[x22,n22] = shift(x,n,-5); % shift by -5
[x21,x22,n212] = timealign(x21,n21,x22,n22); % Time-align first two components
x212 = 4*x21+5*x22; % Add first two components
[x212,x23,n2] = timealign(x212,n212,x,n); % Time-align with the last component
x2 = x212+2*x23; % Final sequence
figure('position',[1,1,7,3]*72);
stem(n2,x2,'filled','markersize',4,'linewidth',1);
axis([min(n2)-1,max(n2)+1,min(x2)-4,max(x2)+6]);
xlabel('Index n'); ylabel('Amplitude');
title('Sequence x_2[n]');
```



Problem P1.2

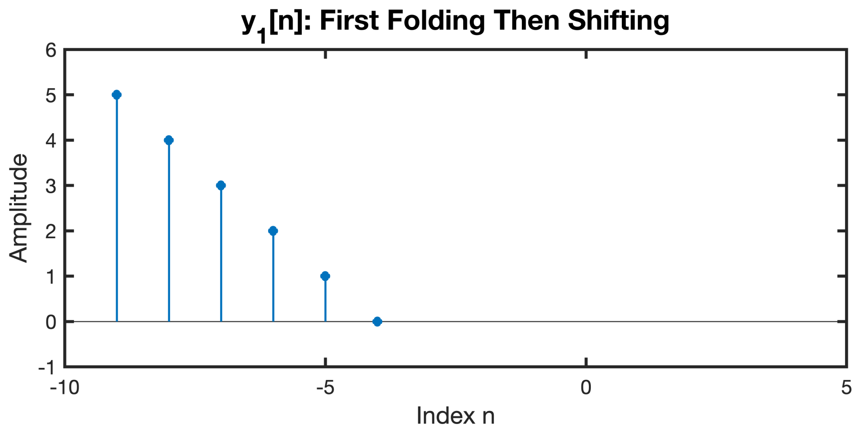
Text Problem 2.37 (Page 85)

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:

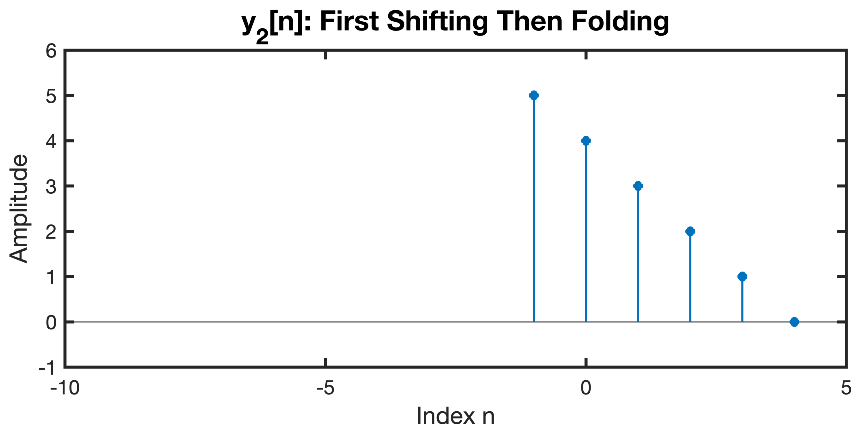
```
clc; close all; clear;
nx = 0:5; x = 0:5; n0 = -4;
[y1,ny1] = fold(x,nx);
[y1,ny1] = shift(y1,ny1,n0);
figure('position',[1,1,7,3]*72);
stem(ny1,y1,'fill','markersize',4,'linewidth',1); axis([-10,5,-1,6]);
xlabel('Index n'); ylabel('Amplitude');
title('y_1[n]: First Folding Then Shifting');
```



(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 $y_2[n]$.

MATLAB script:

```
[y2,ny2] = shift(x,nx,n0);
[y2,ny2] = fold(y2,ny2);
figure('position',[1,1,7,3]*72);
stem(ny2,y2,'fill','markersize',4,'linewidth',1); axis([-10,5,-1,6]);
xlabel('Index n'); ylabel('Amplitude');
title('y_2[n]: First Shifting Then Folding');
```



(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: Clearly, signals $y_1[n]$ and $y_2[n]$ are not the same signals. The correct $x[-n-4]$ signal can be obtained by inserting appropriate values for n. Since $x[n]$ is nonzero for $0 \le n \le 5$, $x[-n-4]$ is nonzero when $0 \le -n-4 \le 5$ or $-9 \le n \le -4$ which is also the support of $y_1[n]$. Thus $y_1[n]$ is the correct signal.

Problem 1.3

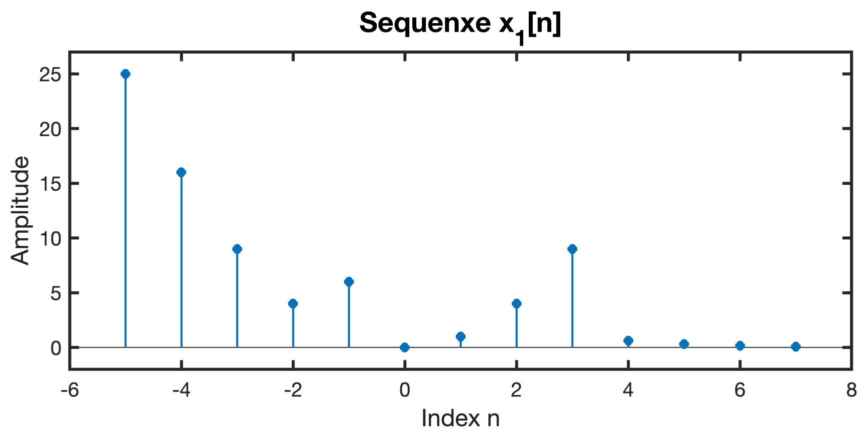
Text Problem 2.38 (Page 85)

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:

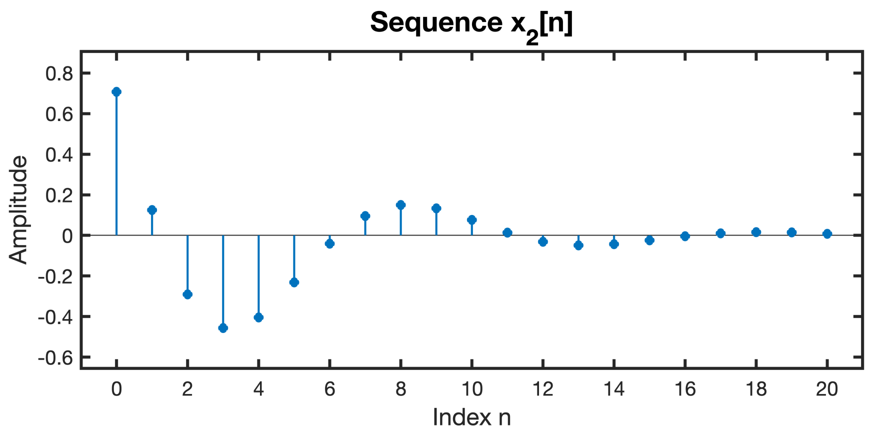
```
[x1a,nx1a] = delta(-1,-1,-1); x1a = 5*x1a; nx1b = -5:3;
x1b = nx1b.^2; nx1c = 4:7; x1c = 10*0.5.^nx1c;
[x1a,x1b,nx1] = timealign(x1a,nx1a,x1b,nx1b);
x1 = x1a + x1b;
[x1,x1c,nx1] = timealign(x1,nx1,x1c,nx1c);
x1 = x1 + x1c;
figure('position',[1,1,7,3]*72);
stem(nx1,x1,'fill','markersize',4,'linewidth',1);
axis([min(nx1)-1,max(nx1)+1,min(x1)-2,max(x1)+2]);
xlabel('Index n'); ylabel('Amplitude');
title('Sequenxe 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:

```
nx2 = 0:20; x2 = 0.8.^nx2.*cos(0.2*pi*nx2+pi/4);
figure('position',[1,1,7,3]*72);
stem(nx2,x2,'fill','markersize',4,'linewidth',1);
axis([min(nx2)-1,max(nx2)+1,min(x2)-0.2,max(x2)+0.2]);
xlabel('Index n'); ylabel('Amplitude');
title('Sequence x_2[n]');
```

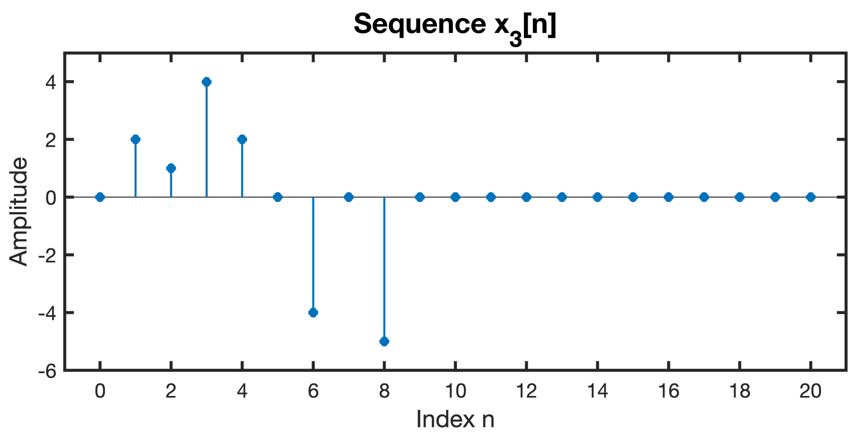


(c) $x_3[n] = \sum_{m=0}^{4}(m+1)\{\delta[n-m] - \delta[n-2m]\}$, $0 \le n \le 20$

MATLAB script:

```
M = 4; m = (0:M)'; N = 20; nx3 = (0:N)';
d1 = toeplitz([1;zeros(N,1)],[1,zeros(1,M)]);
d2 = zeros(N+1,M+1); d2(1:2:N+1,:) = d1(1:1:(N/2+1),:);
x3 = (d1-d2)*(m+1);
figure('position',[1,1,7,3]*72);
stem(nx3,x3,'fill','markersize',4,'linewidth',1);
```

```
axis([min(nx3)-1,max(nx3)+1,min(x3)-1,max(x3)+1]);
xlabel('Index n'); ylabel('Amplitude');
title('Sequence 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 $(1.4.1)$.

Solution: Adding $x_e[n]$ and $x_o[n]$ from $(1.4.1)$ we obtain

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

which proves the result.

(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
%
if ~isreal(x)
    error('*** x must be a real-valued sequence ***');
end
[xfold,nfold] = fold(x,n);
[y1,y2,m] = timealign(x,n,xfold,nfold);
xe = 0.5*(y1+y2);
xo = 0.5*(y1-y2);
end
```

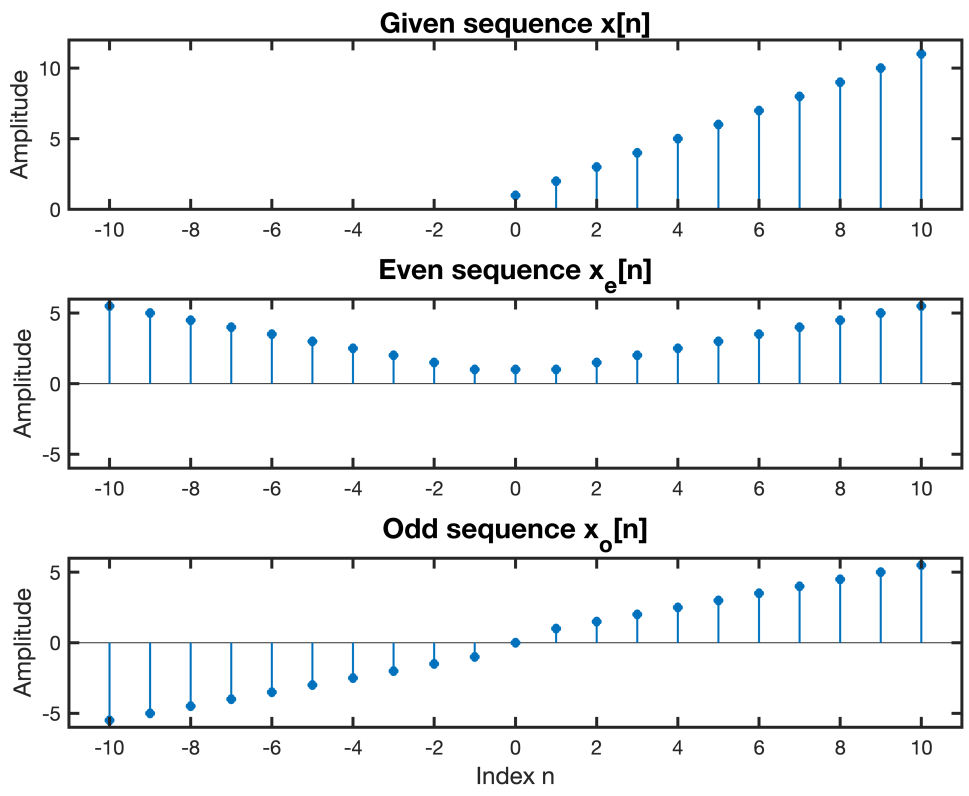
(c) Using your **EvenOdd** function, decompose the following sequence

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

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 3×1 **subplot**s.

MATLAB script:

```
clc; close all; clear;
x = (1:11); n = (0:10);
[xe,xo,m] = EvenOdd(x,n);
figure('position',[0,0,8,6]*72);
subplot(3,1,1); stem(n,x,'filled','markersize',4,'linewidth',1); axis([-11,11,0,12]);
ylabel('Amplitude'); title('Given sequence x[n]');
subplot(3,1,2); stem(m,xe,'filled','markersize',4,'linewidth',1); axis([-11,11,-6,6]);
ylabel('Amplitude'); title('Even sequence x_e[n]');
subplot(3,1,3); stem(m,xo,'filled','markersize',4,'linewidth',1); axis([-11,11,-6,6]);
ylabel('Amplitude'); xlabel('Index n'); title('Odd sequence 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 $h[n] = n\left(u[n] - u[n-M]\right),\ M \le N.$

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

Solution: The convolution sequence $y[n]$ is given by

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m] = \sum_{m=-\infty}^{\infty} m\left(u[m] - u[m-M]\right)\left(u[n-m] - u[n-m-N]\right)$$

$$= \sum_{m=0}^{M-1} m\left(u[n-m] - u[n-m-M]\right)$$

Consider

- If $0 \le n \le (M-1)$ then $y[n] = \sum_{m=0}^{n} m = \dfrac{n(n+1)}{2}$.

- If $(M-1) \leq n \leq (N-1)$ then $y[n] = \sum_{m=0}^{M-1} m = \dfrac{M(M-1)}{2}$.
- If $(N-1) \leq n \leq (M+N-2)$ then

$$y[n] = \sum_{m=n-(N-1)}^{M-1} m = \sum_{m=0}^{M-1} m - \sum_{m=0}^{n-N} m = \frac{M(M-1)}{2} - \frac{(n-N+1)(n-N)}{2}.$$

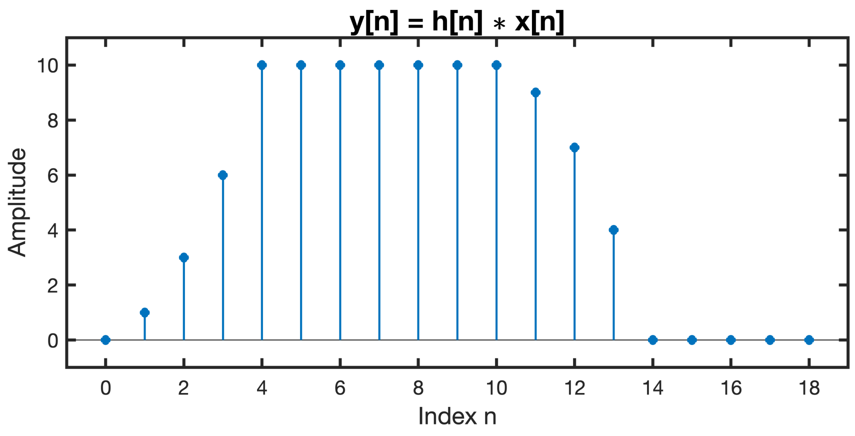
- For all other values of n, $y[n] = 0$.

(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.

MATLAB script:

```
clc; close all; clear;
N = 10; M = 5; n = 0:N-1; x = unitpulse(0,0,N-1,N-1)';
h = n.*unitpulse(0,0,M-1,N-1)'; [y,ny] = conv0(h,n,x,n);
figure('position',[0,0,7,3]*72);
stem(ny,y,'fill','markersize',4,'linewidth',1);
xlabel('Index n'); ylabel('Amplitude');
title('y[n] = h[n] \ast x[n]'), axis([-1,19,-1,11]);
```



Problem 1.6

Text problem 2.34 (Page 84)

A system is described by the difference equation

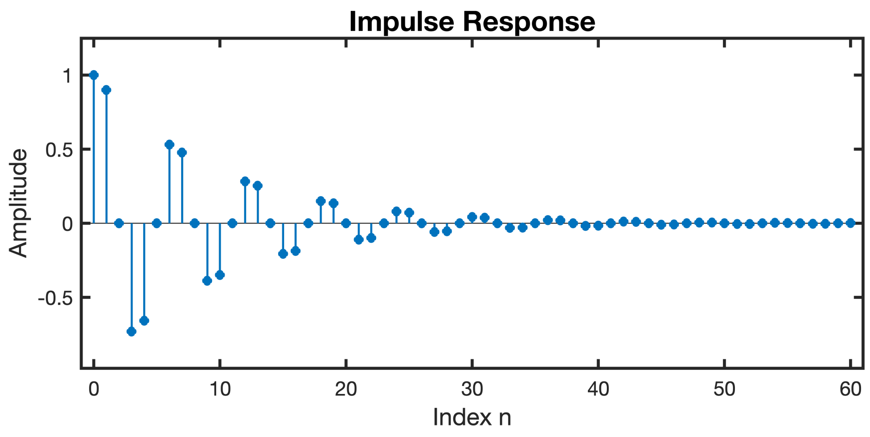
$$y[n] = x[n] + 0.9y[n-1] - 0.81y[n-2].$$

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

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

MATLAB script:

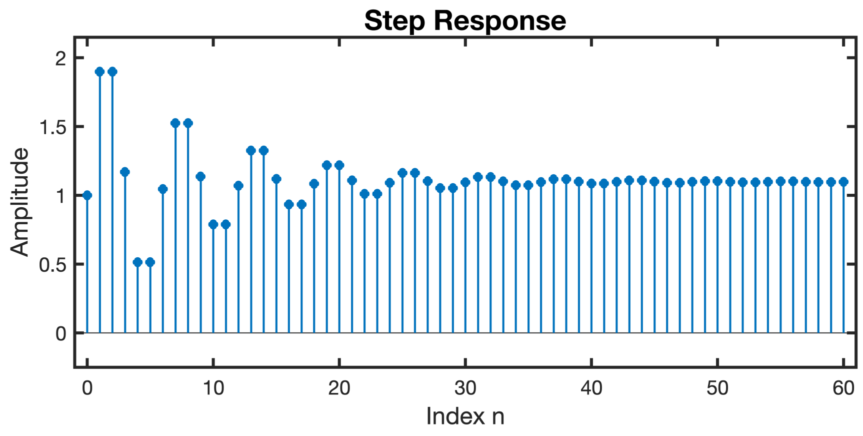
```
clc; close all; clear;
N = 60; n = 0:N;
b = 1; a = [1, -0.9, 0.81];
[d,nd] = delta(n(1),0,n(N+1));
h = filter(b,a,d);
figure('position',[0,0,7,3]*72);
stem(n,h,'fill','markersize',4,'linewidth',1);
axis([n(1)-1,n(N+1)+1,min(h)-0.25,max(h)+0.25])
xlabel('Index n'); ylabel('Amplitude'); title('Impulse Response');
```



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

MATLAB script:

```
[u,nu] = unitstep(n(1),0,n(N+1));
s = filter(b,a,u);
figure('position',[0,0,7,3]*72);
stem(n,s,'fill','markersize',4,'linewidth',1);
axis([n(1)-1,n(N+1)+1,-0.25,max(s)+0.25])
xlabel('Index n'); ylabel('Amplitude'); title('Step Response');
```



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

Answer: From the plot above, approximately the first 50 samples constitute the transient response while the rest is the steady-state response.

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 θ is a constant.

(a) Determine if the system is linear.

Solution: Consider the response to a linear combination of inputs:

$$\begin{aligned}H\{a_1x_1[n] + a_2x_2[n]\} &= 10\big(a_1x_1[n] + a_2x_2[n]\big)\cos(0.25\pi n + \theta)\\ &= a_1\big(10x_1[n]\cos(0.25\pi n + \theta)\big) + a_2\big(10x_2[n]\cos(0.25\pi n + \theta)\big)\\ &= a_1y_1[n] + a_2y_2[n].\end{aligned}$$

which results in a linear combination of corresponding outputs. Hence the system is linear.

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

Solution: The output due to a shifted input $x[n-n_0]$ is

$$H\{x[n-n_0]\} = 10x[n-n_0]\cos(0.25\pi n + \theta)$$

while the shifted output $y[n-n_0]$ is

$$y[n-n_0] = 10x[n-n_0]\cos\left(0.25\pi[n-n_0]+\theta\right).$$

Clearly, $y[n-n_0] \neq H\{x[n-n_0]\}$. Hence the system is not time-invariant.

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.

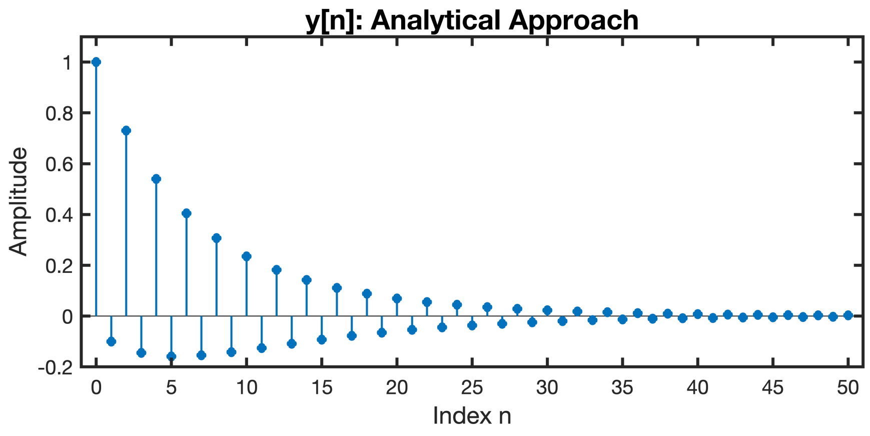
Solution: The convolution $y[n]=h[n]*x[n]$ is given by

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = \sum_{k=0}^{\infty}(-0.9)^k u[k](0.8)^{n-k}u[n-k]$$

$$= \left[\sum_{k=0}^{n}(-0.9)^k(0.8)^n(0.8)^{-k}\right]u[n] = (0.8)^n\left[\sum_{k=0}^{n}\left(-\frac{9}{8}\right)^k\right]u[n] = \frac{0.8^{n+1}-(-0.9)^{n+1}}{1.7}u[n].$$

MATLAB script:

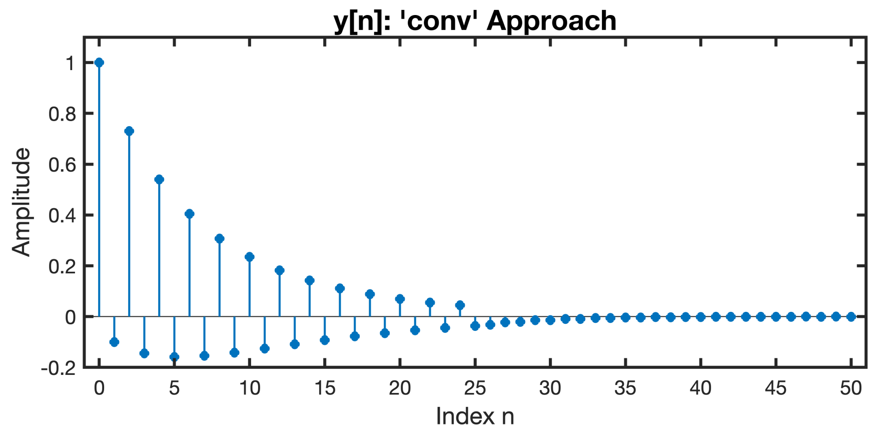
```
clc; close all; clear;
n = 0:50; x = 0.8.^n; h = (-0.9).^n;
figure('position',[0,0,7,3]*72);
y1 = ((0.8).^(n+1) - (-0.9).^(n+1))/(0.8+0.9);
stem(n,y1,'filled','markersize',4,'linewidth',1); axis([-1,51,-0.2,1.1]);
title('y[n]: Analytical Approach');
xlabel('Index n'); ylabel('Amplitude');
```



(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:

```
x2 = x(1:26); h2 = h(1:26); y2 = conv(h2,x2);
figure('position',[0,0,7,3]*72);
stem(n,y2,'filled','markersize',4,'linewidth',1); axis([-1,51,-0.2,1.1]);
title("y[n]: 'conv' Approach");
xlabel('Index n'); ylabel('Amplitude');
```



(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: The impulse response $h[n]$ can be expressed as

$$(-0.9)^n u[n] = \delta[n] + (-0.9)^n u[n-1] = \delta[n] - 0.9\big((-0.9)^{n-1}u[n-1]\big)$$
$$\Rightarrow h[n] = \delta[n] - 0.9h[n-1]$$

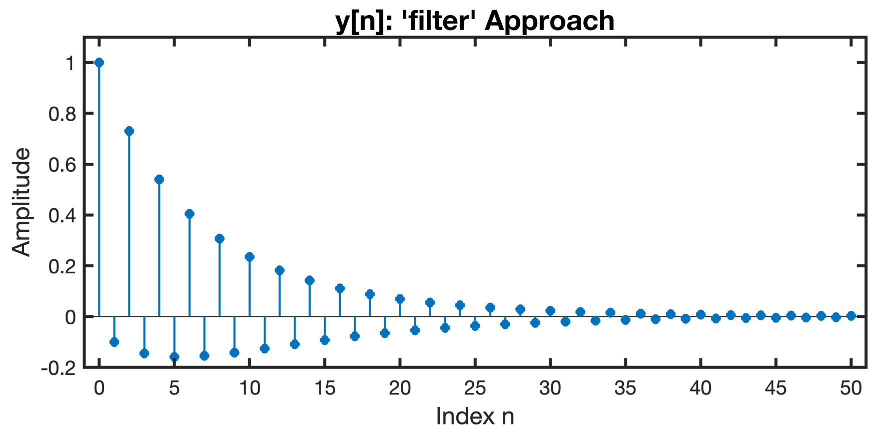
Since $x[n] = \delta[n] \Rightarrow y[n] = h[n]$, the corresponding difference equation is

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

Thus the corresponding filter coefficients are **b = 1;** and **a = [1,0.9];**.

MATLAB script:

```
y3 = filter(1,[1,0.9],x);
figure('position',[0,0,7,3]*72);
stem(n,y3,'filled','markersize',4,'linewidth',1); axis([-1,51,-0.2,1.1]);
title("y[n]: 'filter' Approach");
xlabel('Index n'); ylabel('Amplitude');
```



(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 analytical solution to the convolution in (a) is the exact answer. In the **`filter`** function approach (c), the infinite-duration sequence $h[n]$ is exactly represented by coefficients of an equivalent filter. Therefore, the **`filter`** solution should also be exact except that it is evaluated up to the length of the input sequence. The **`conv`** approach needs finite-duration sequences. Therefore, both $x[n]$ and $h[n]$ have to be truncated to 26 samples to obtain 51 samples of the result. Thus computation in (b) is correct up to the first 26 samples and then it degrades rapidly. Clearly, approach in (b) gives an incorrect result. Note that if we wanted the first 51 correct samples then we should have considered 51 samples of both $x[n]$ and $h[n]$ and then truncated the result to 51 samples.

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.

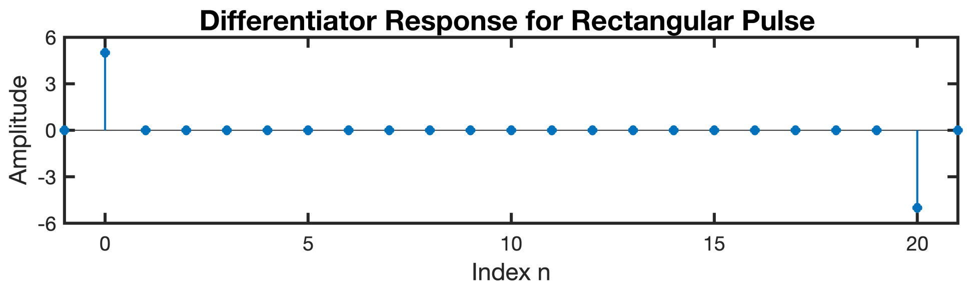
Implement this differentiator on the following sequences, and plot the corresponding results.

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

MATLAB script:

```
clc; close all; clear;
a = 1; b = [1 -1]; % Simple differentiator parameters
[x1,n1] = unitpulse(-5,0,19,25); x1 = 5*x1;
y1 = filter(b,a,x1);
figure('position',[1,1,8,2]*72);
stem(n1,y1,'filled','markersize',4,'linewidth',1); axis([-1,21,-6,6]);
```

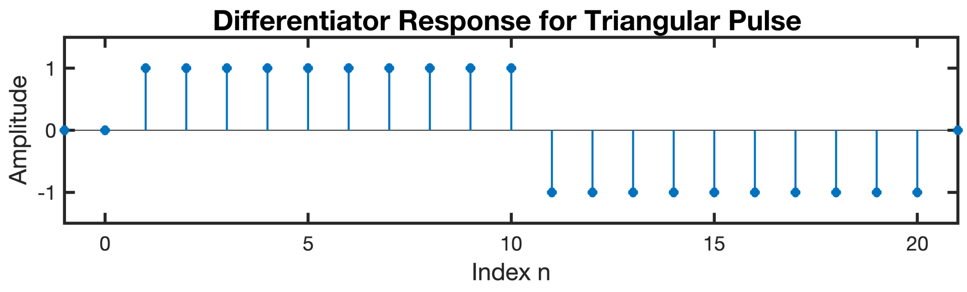
```
xlabel('Index n'); ylabel('Amplitude');
title('Differentiator Response for Rectangular Pulse ');
set(gca,'XTick',(0:5:20),'YTick',(-6:3:6));
```



(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:

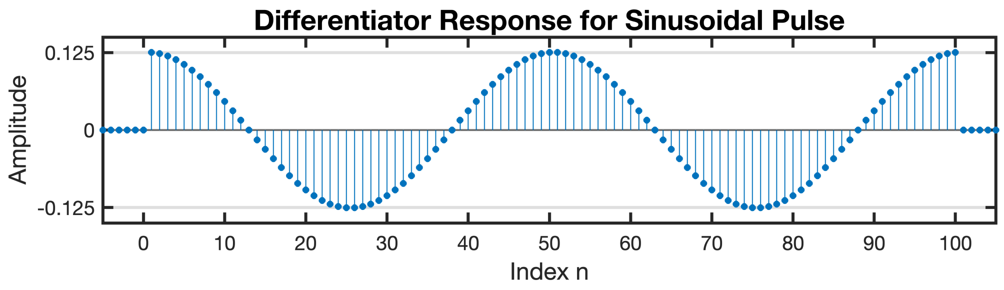
```
[x11,nx11] = unitstep(-5,0,25); [x12,nx12] = unitstep(-5,10,25);
[x13,nx13] = unitstep(-5,20,25); n2 = nx11;
x2  =  n2.*(x11 - x12) + (20 - n2).*(x12 - x13);
y2 = filter(b,a,x2);
figure('position',[1,1,8,2]*72);
stem(n2,y2,'filled','markersize',4,'linewidth',1); axis([-1,21,-1.5,1.5]);
xlabel('Index n'); ylabel('Amplitude');
title('Differentiator Response for Triangular Pulse ');
set(gca,'XTick',(0:5:20),'YTick',(-1:1));
```



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

MATLAB script:

```
[x3,n3] = unitpulse(-10,0,99,110); x3 = sin(pi*n3/25).*x3;
y3 = filter(b,a,x3);
figure('position',[1,1,8,2]*72);
stem(n3,y3,'filled','markersize',3); axis([-5,105,-0.15,0.15]);
xlabel('Index n'); ylabel('Amplitude');
title('Differentiator Response for Sinusoidal Pulse');
ytick = [round(min(y3),3),0,round(max(y3),3)];
set(gca,'XTick',(0:10:100),'YTick',ytick,'ygrid','on');
```



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

Answer: Differentiation of a rectangular pulse, in part (a), should result in a positive impulse at the rising-edge and a negative impulse at the falling-edge. The digital differentiator gives us, appropriately, positive and negative samples at those edges. Differentiation of a triangular pulse, in part (b), should result in a positive pulse for the linearly-increasing side and a negative pulse for the linearly-decreasing side which is what we get in the corresponding plot. Differentiation of a sinusoidal pulse, in part (c), should result in a co-sinusoidal pulse. If we assume that the discrete-time sinusoidal pulse was obtained from a continuous-time sinusoidal pulse by sampling and letting $t = nT = n$ where $T = 1\text{second}$ is the sampling interval, then we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\sin\left(\frac{\pi t}{25}\right)\right] = \frac{\pi}{25}\cos\left(\frac{\pi t}{25}\right) = 0.125\cos\left(\frac{\pi t}{25}\right) \quad \Rightarrow \quad 0.125\cos\left(\frac{\pi n}{25}\right)$$

after sampling. The plot in (c) confirms this calculation. Thus the simple digital differentiator is a good derivative operator or system.

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: Consider the convolution of $h[n]$ and $g[n]$.

$$h[n] * g[n] = \left(\sum_{k=0}^{\infty} a_k \delta[n-kD]\right) * \left(\sum_{\ell=0}^{\infty} b_\ell \delta[n-\ell D]\right)$$

$$\delta[n] = \sum_{k=0}^{\infty}\sum_{\ell=0}^{\infty} a_k b_\ell \big(\delta[n-kD] * \delta[n-\ell D]\big)$$

$$\delta[n] = \sum_{k=0}^{\infty}\sum_{\ell=0}^{\infty} a_k b_\ell \delta[n-kD-\ell D] = \sum_{k=0}^{\infty}\sum_{\ell=0}^{\infty} a_k b_\ell \delta[n-(k+\ell)D]$$

$$\delta[n] = \sum_{k=0}^{\infty}\left(\sum_{m=0}^{k} a_{k-m} b_m\right)\delta[n-kD], \quad \begin{Bmatrix}(k+\ell) & \to & k \\ \ell & \to & m\end{Bmatrix} \Rightarrow \begin{Bmatrix}0 \le k < \infty \\ 0 \le \ell < \infty\end{Bmatrix} \to \{0 \le m \le k < \infty\}$$

$$\delta[n] = (a_0 b_0)\delta[n] + \sum_{k=1}^{\infty}\left(\sum_{m=0}^{k} a_{k-m} b_m\right)\delta[n-kD]$$

Hence the successive b_k's must satisfy

$$a_0 b_0 = 1 \text{ at } k=0 \quad \text{and} \quad \sum_{m=0}^{k} a_{k-m} b_m = 0, \text{ for } k > 0.$$

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

Solution: Consider

- $k=0: \; a_0 b_0 = 1 \quad \Rightarrow \quad b_0 = a_0^{-1}.$
- $k=1: \; a_0 b_1 + a_1 b_0 = 0 \quad \Rightarrow \quad b_1 = -a_1 a_0^{-2}.$
- $k=2: \; a_0 b_2 + a_1 b_1 + a_2 b_0 = 0 \quad \Rightarrow \quad b_2 = -a_2 a_0^{-2} + a_1^2 a_0^{-3}.$

(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: Combining conditions and results of previous parts, we have

$$b_0 = 1, b_1 = -0.5, b_2 = 0 \quad \text{and} \quad b_k + 0.5 b_{k-1} + 0.25 b_{k-2} = 0, \quad k \ge 3.$$

or

k	0	1	2	3	4	5	6	7	$\cdots$
b_k	1	-0.5	0	0.5^3	-0.5^4	0	0.5^6	-0.5^7	$\cdots$

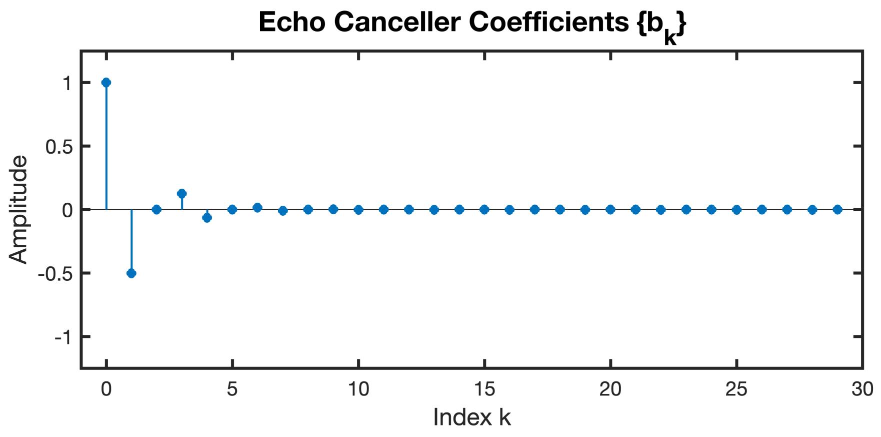
Thus we conclude that

$$b_k = \begin{cases} 0.5^k, & k = 3l \\ -0.5^k, & k = 3l+1 \\ 0, & k = 3l+2 \end{cases} \qquad l = 0, 1, 2, \ldots$$

Plot of the first 30 samples of $\{b_k\}$ is given below. This plot was not required in the problem.

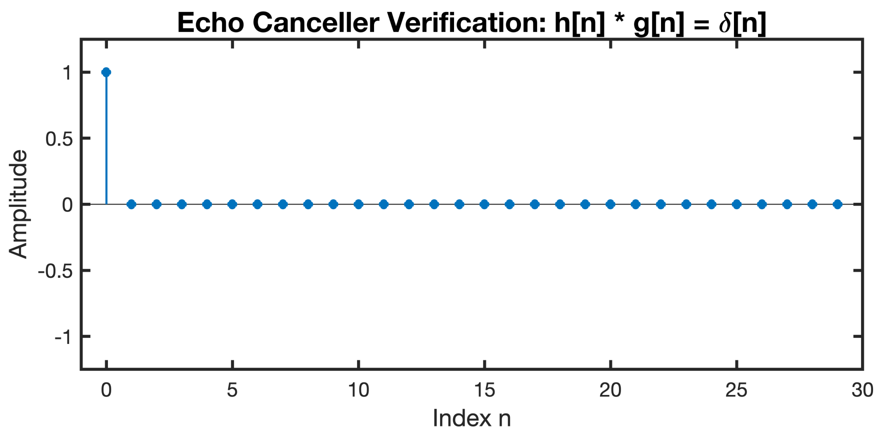
MATLAB script:

```
clc; close all; clear;
l = (0:9); L = 3*(l(end)+1); b = zeros(1,L);
b(3*l+1) = (0.5).^(3*l); b(3*l+2) = -(0.5).^(3*l+1); b(3*l+3) = 0;
figure('position',[1,1,7,3]*72);
stem((0:L-1),b,'filled','markersize',4,'linewidth',1);
xlabel('Index k'); ylabel('Amplitude'); axis([-1,L,-1.25,1.25]);
title('Echo Canceller Coefficients \{b_k\}');
```



Verification: We can validate the design of the echo canceller $g[n]$ by performing convolution between the echo impulse response $h[n]$ and $g[n]$ and check if it is equal to the impulse $\delta[n]$. The following plot verifies it. Again, it was not required.

```
a = [1,0.5,0.25]; % given echo impulse response
check = conv(a,b); % verification
figure('position',[1,1,7,3]*72);
stem((0:L-1),check(1:L),'filled','markersize',4,'linewidth',1);
xlabel('Index n'); ylabel('Amplitude'); axis([-1,L,-1.25,1.25]);
title('Echo Canceller Verification: h[n] * g[n] = \delta[n]');
```



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 ~isreal(x)
    error('*** x must be a real-valued sequence ***');
end
[xfold,nfold] = fold(x,n);
[y1,y2,m] = timealign(x,n,xfold,nfold);
xe = 0.5*(y1+y2);
xo = 0.5*(y1-y2);
end
```