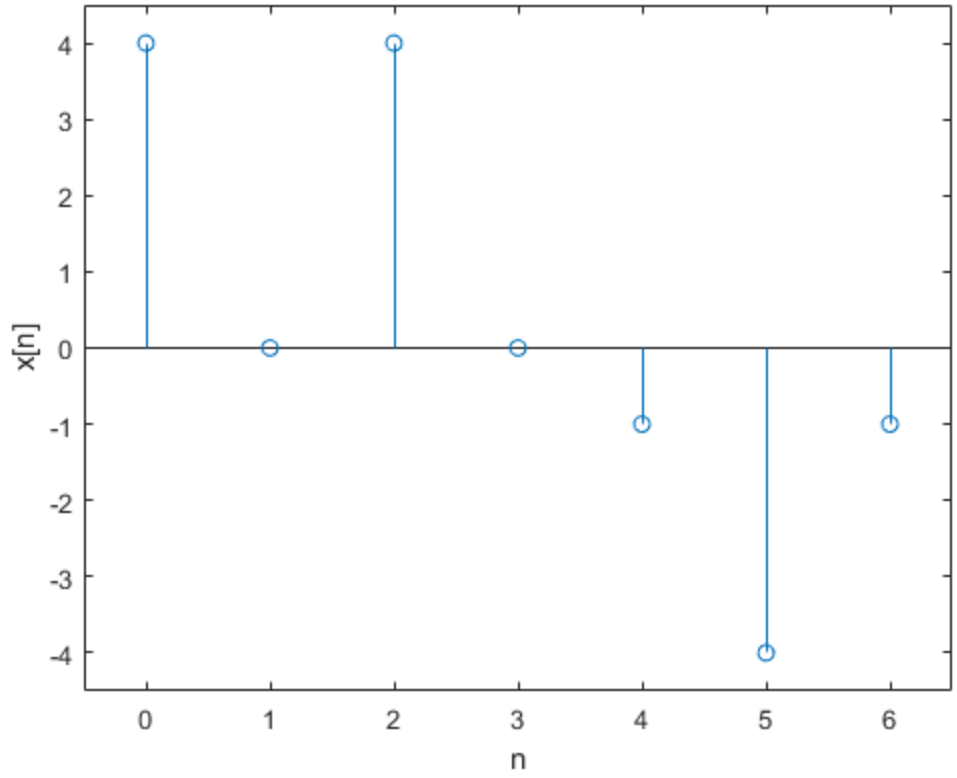
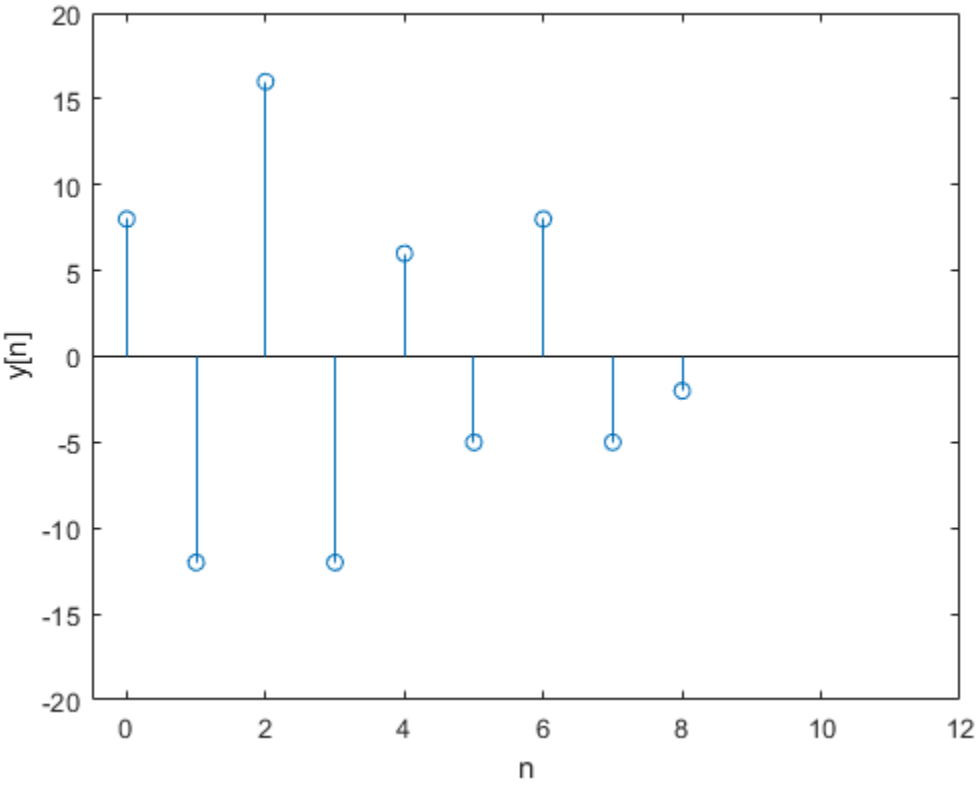
```
clc
close all
```

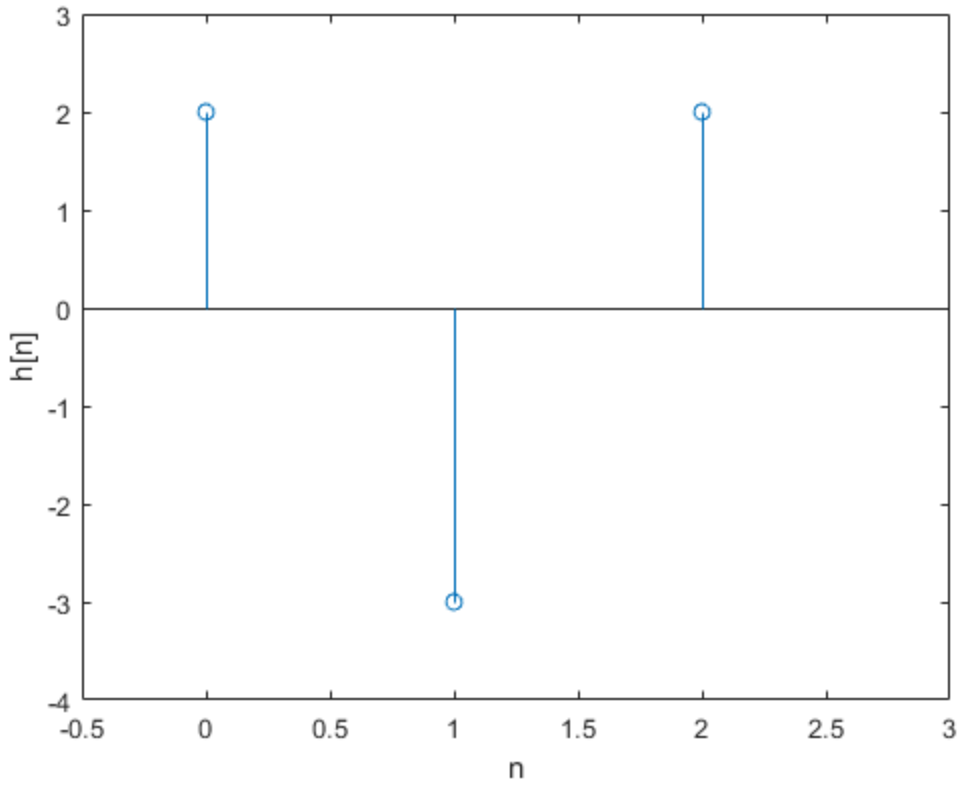
2. Finute Impulse Response (FIR) Filter

```
%Part a)
x = [ 4 0 4 0 -1 -4 -1 ];
n = 0 : length(x) - 1;
stem(n, x);
xlim( [-0.5, 6.5] );
ylim( [-4.5, 4.5] );
xlabel( 'n' );
ylabel( 'x[n]' );

b = [ 2 -3 2 ];
y = conv(x, b);

fprintf('y: [')
fprintf('%d, ', y(1:end-1))
fprintf('%d]\n', y(end))

%Part b)
n = 0 : length(y) - 1;
figure;
stem(n, y);
xlim( [-0.5, 12] );
ylim( [-20, 20] );
xlabel( 'n' );
ylabel( 'y[n]' );

%Part c)
h = [ 2 -3 2 ];
n = 0 : length(h) - 1;
figure;
stem(n, h);
xlim( [-0.5, 3] );
ylim( [-4, 3] );
xlabel( 'n' );
ylabel( 'h[n]' );
```

y: [8, -12, 16, -12, 6, -5, 8, -5, -2]


x[n]
n
4
3
2
1
0
-1
-2
-3
-4
0
1
2
3
4
5
6


y[n]
n
20
15
10
5
0
-5
-10
-15
-20
0
2
4
6
8
10
12



System Identification:

c)

```
x = [ 1 2 3 4 5 ];
y = [ 1 1 1 1 1 -5 ];

% Determine Nmax based on input signal
%   Finite-length   length(y) - length(x) + 1
%   Infinite-length length(x)
Nmax = length(y) - length(x) + 1;   %% finite-length input signal
if ( Nmax < 2 )
    Nmax = length(x);
end

b = zeros(1, Nmax);
b(1) = y(1) / x(1); % Compute the first b value

% Compute the rest of b value
for k = 2:Nmax
    numer = y(k);
    n = k;
    for m = 1:(k-1)
 if (n >= 1)
            numer = numer - b(m) * x(n);
 end
```

```
        n = n - 1;
    end
    b(k) = numer / x(1);
    % Avoid possible division by zero error
    if (abs(b(k) - b(k-1)) <= (1e-7)*abs(b(k)))
        break;
    end
end

% utdeconvolve.m. implements the above algorithm for deconvolution.

% Part (a).  Give the vectors for x and y that you used when running
% utdeconvolve.m. and the filter coefficients in vector b that the code
 computes.

fprintf('y: [')
fprintf('%d, ', y(1:end-1))
fprintf('%d]\n', y(end))

fprintf('Here is the x vector: [');
fprintf('%d, ', x(1:end-1));
fprintf('%d]\n', x(end))

fprintf('Here is the y vector: [');
fprintf('%d, ', y(1:end-1));
fprintf('%d]\n', y(end))

% Part (b). Verify that the filter coefficients by using them in the
 difference
% equation for the LTI FIR filter.  You can use the Matlab command conv(x, b).
c = conv(x,b);

fprintf('Here are the filter coefficients: [');
fprintf('%d, ', c(1:end-1));
fprintf('%d]\n', c(end))
```

```
y: [1, 1, 1, 1, 1, -5]
Here is the x vector: [1, 2, 3, 4, 5]
Here is the y vector: [1, 1, 1, 1, 1, -5]
Here are the filter coefficients: [1, 1, 1, 1, 1, -5]
```

Published with MATLAB® R2023a