AV 336- Digital Signal Processing Lab
Labsheet 24

Submitted by: K.V.N.G.Vikram
SC15B148
E.P Roll No- 8

1. Review the following from the textbook (Chapter 10 of Lee & Varaiya)
(a) discrete Fourier transform (DFT) of a discrete time signal x[n]
(b) the inverse DFT

Inference:

Discrete Fourier transform: $X_k = \sum_{m=0}^{p-1} x(m) e^{-imk\omega_0}$

Inverse DFT: $x(n) = \frac{1}{p} \sum_{k=0}^{p-1} X_k e^{-ik\omega_0 n}$

2. Using the internet:
a) https://en.wikipedia.org/wiki/Fast_Fourier_Transform,
b) http://www.dspguide.com/ch12/2.htm
or using the textbook by Oppenheim and Schafer (ch9 and also ch8), get an idea about what fast fourier transform (FFT) is. Then using Matlab's documentation (or internet resources) study what the following inbuilt functions in Matlab do:

a) dftmtx
b) fft
c) ifft
d) fftshift

a)

```
D = dftmtx(5)
D =

  Columns 1 through 2

   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.3090 - 0.9511i
   1.0000 + 0.0000i  -0.8090 - 0.5878i
   1.0000 + 0.0000i  -0.8090 + 0.5878i
   1.0000 + 0.0000i   0.3090 + 0.9511i

  Columns 3 through 4

   1.0000 + 0.0000i   1.0000 + 0.0000i
  -0.8090 - 0.5878i  -0.8090 + 0.5878i
   0.3090 + 0.9511i   0.3090 - 0.9511i
   0.3090 - 0.9511i   0.3090 + 0.9511i
  -0.8090 + 0.5878i  -0.8090 - 0.5878i

  Column 5
```

```
   1.0000 + 0.0000i
   0.3090 + 0.9511i
  -0.8090 + 0.5878i
  -0.8090 - 0.5878i
   0.3090 – 0.9511i
```

b)

```
>>fft([4 5 6])

ans =

  15.0000 + 0.0000i  -1.5000 + 0.8660i  -1.5000 – 0.8660i
```

c)

```
>> ifft(fft([4 5 6]))

ans =

    4     5     6
```

d)

```
>> fftshift([1 2 3 4 5])

ans =

    4     5     1     2     3
```

Inference:
dftmtx Discrete Fourier transform matrix. dftmtx(N) is the N-by-N complex matrix of values around the unit-circle whose inner product with a column vector of length N yields the discrete Fourier transform of the vector. If X is a column vector of length N, then dftmtx(N)*X yields the same result as FFT(X).
fft Discrete Fourier transform. fft(X) is the discrete Fourier transform (DFT) of vector X. For matrices, the fft operation is applied to each column. For N-D arrays, the fft operation operates on the first non-singleton dimension. ifft is the inverse fast fourier transform it returns the vector whose fft is given as input.
fftshift Shift zero-frequency component to center of spectrum. For vectors, fft-shift(X) swaps the left and right halves of X. For matrices, fftshift(X) swaps the first and third quadrants and the second and fourth quadrants. For N-D arrays, fftshift(X) swaps ”half-spaces” of X along each dimension.

3. Implementation of DFT and IDFT. (a) Write a Matlab function named mydft1” that computes the N-point DFT of a discrete time signal x[n] where n=0,1,.....N-1
i. The input x[n] is assumed to extend from 0 to N - 1 and can be a Matlab vector, and
ii. the function should implement DFT using loops, and
iii. the function should return the N-point DFT as another Matlab vector.
(b) Write a Matlab function named myidft1” that computes the N-point IDFT of a DFT X[k] where k=0,1,.....N-1

i. The input X[k] is assumed to extend from 0 to N - 1 and can be a Matlab vector, and
ii. the function should implement IDFT using loops, and
iii. the function should return the N-point signal x[n] as another Matlab vector.
(c) i. Using mydft1() compute the 16-point DFT of x[n] = cos(2π(0:25)n) for n =0,1,.....N-1
ii. Test whether myidft1(mydft1(x[n])) = x[n].
(d) Write a Matlab function named mydft2" that computes the N-point DFT of a discrete time signal x[n] where n =0,1,.....N-1
i. The input x[n] is assumed to extend from 0 to N - 1 and can be a Matlab vector, and
ii. the function should implement DFT using dftmtx,
iii. the function should return the N-point DFT as another Matlab vector.
(e) Write a Matlab function named myidft2" that computes the N-point IDFT of a DFT X[k] where k=0,1,.....N-1.
i. The input X[k] is assumed to extend from 0 to N 1 and can be a Matlab vector, and
ii. the function should implement IDFT using dftmtx, and
iii. the function should return the N-point signal x[n] as another Matlab vector.
(f) i. Using mydft2() compute the 16-point DFT of x[n] = cos(2π(0:25)n) for n =0,1,.....N-1.
iv. Test whether myidft2(mydft2(x[n])) = x[n].

Ans:
Functions:

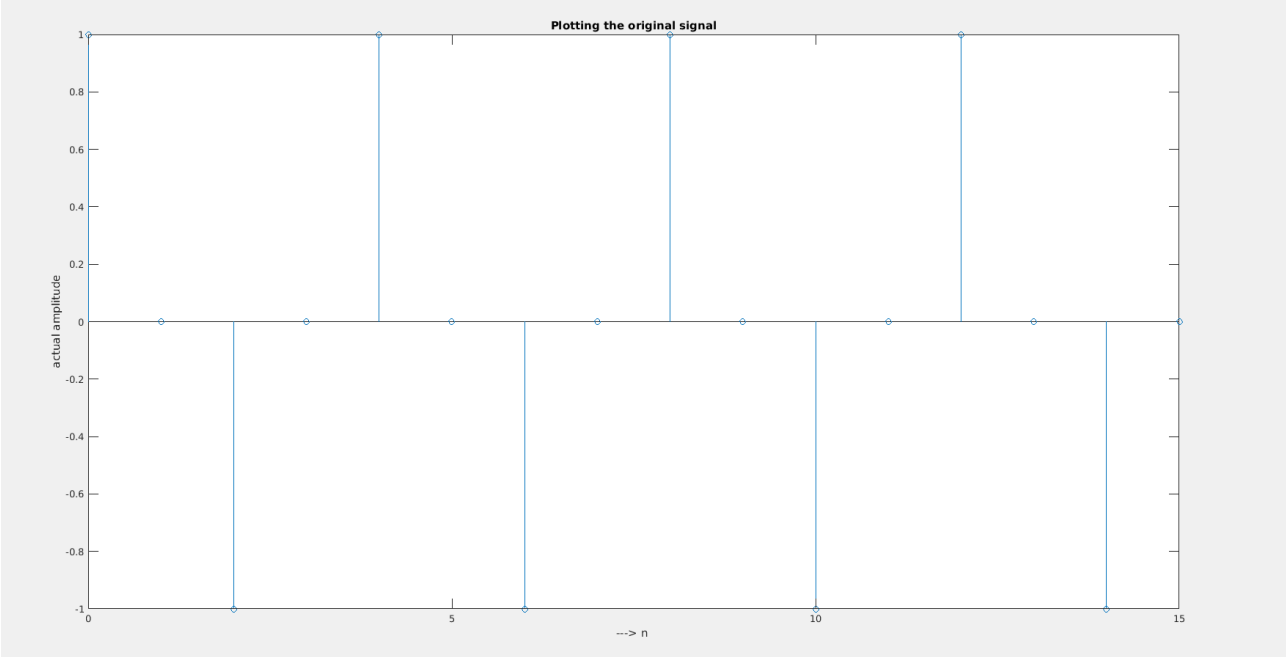
```
% X is the DFT
function X = mydft1(x)
    N = numel(x);

    for k = 1:N   % number of elements in x and X are equal
        X(k) = 0; % for each element of X
        for n = 1:N  % summation for all elements in x
            % k and n should be calculated from 0 but index starts from 1
            X(k)=X(k)+x(n)*exp(-1i*2*pi*(k-1)*(n-1)/N);
        end
    end

end

% x is the IDFT of X
function x = myidft1(X)
    K = numel(X);

    for n = 1:K   % number of elements in x and X are equal
        x(n) = 0;
        for k = 1:K
            x(n)=x(n)+X(k)*exp(1i*2*pi*(n-1)*(k-1)/K);
        end
        x(n) = x(n)/K;
    end

end
```

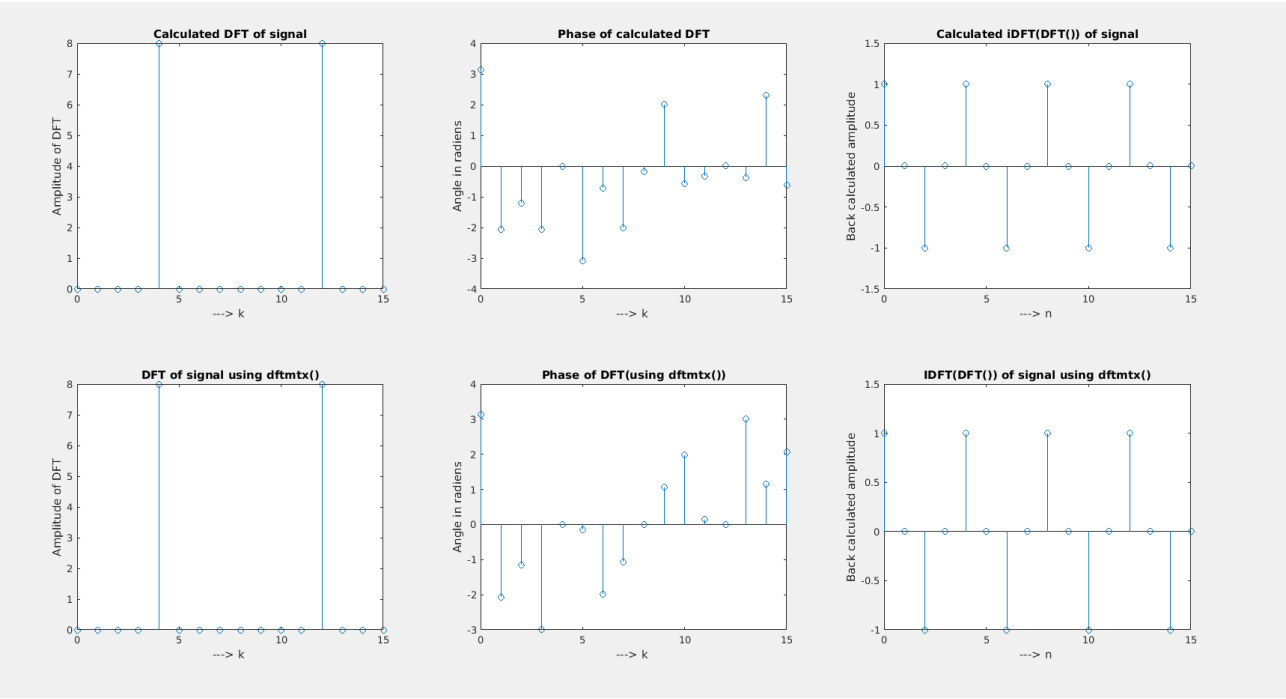
```
function X = mydft2(x)
X = x*dftmtx(numel(x));
end


function x = myidft2(X)
x = X*inv(dftmtx(numel(X)));
end
```

Code:

```
% It is observed that myidft1(mydft1(x[n])) is equal to x[n]

% n values on x axis
b = 0 : 15;
%b = 0 : 720;

% given signal
v = cos(2*pi*0.25*b);
%v = cos(b*rand());

% plotting the signal
stem(b,v)
title('Plotting the original signal')
ylabel('actual amplitude')
xlabel('---> n ')

figure()
% plotting calculated DFT
subplot(2,3,1)
stem(b,abs(mydft1(v)))
title('Calculated DFT of signal')
ylabel('Amplitude of DFT')
xlabel('---> k ')

subplot(2,3,2)
stem(b,angle(mydft1(v)))
title('Phase of calculated DFT')
ylabel('Angle in radiens')
xlabel('---> k ')

% plotting IDFT to get back original signal
subplot(2,3,3)
stem(b,real(myidft1(mydft1(v))))
title('Calculated iDFT(DFT()) of signal')
ylabel('Back calculated amplitude')
xlabel('---> n ')

% plotting calculated DFT using dftmtx
subplot(2,3,4)
```

```
stem(b,abs(mydft2(v)))
title('DFT of signal using dftmtx()')
ylabel('Amplitude of DFT')
xlabel('---> k ')

subplot(2,3,5)
stem(b,angle(mydft2(v)))
title('Phase of DFT(using dftmtx())')
ylabel('Angle in radiens')
xlabel('---> k ')

% plotting IDFT to get back original signal using dftmtx
subplot(2,3,6)
stem(b,real(myidft2(mydft2(v))))
title('IDFT(DFT()) of signal using dftmtx()')
ylabel('Back calculated amplitude')
xlabel('---> n ')
```

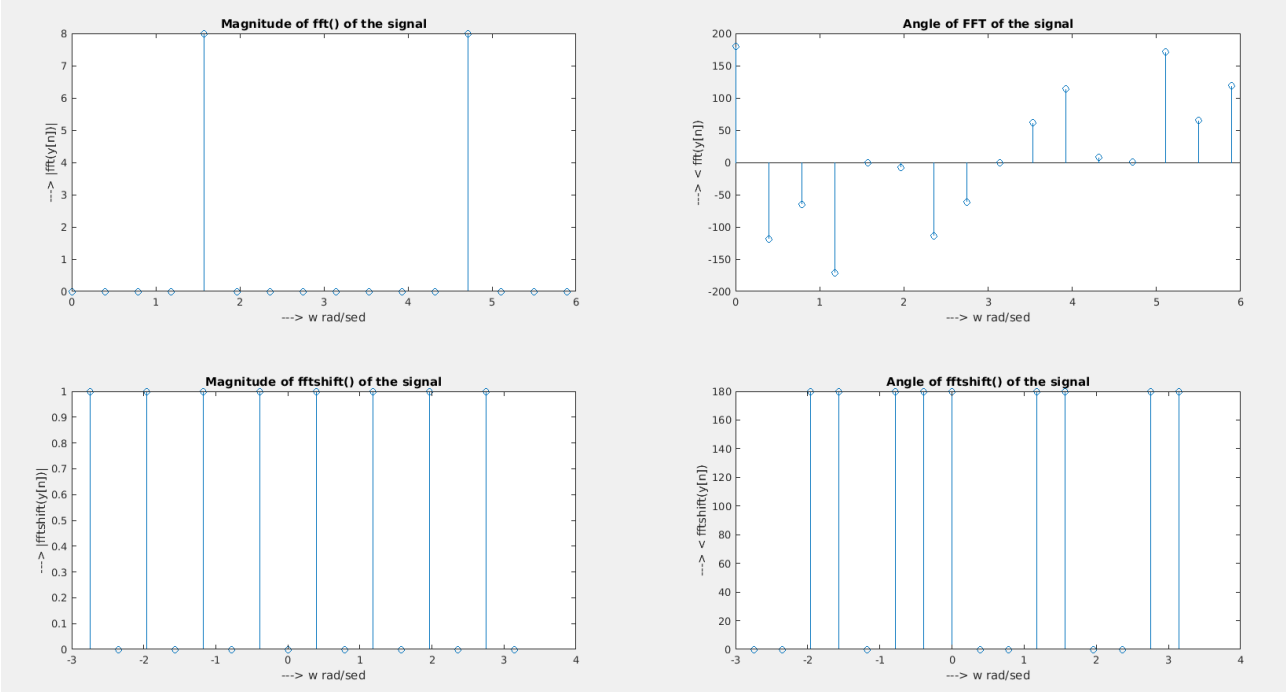


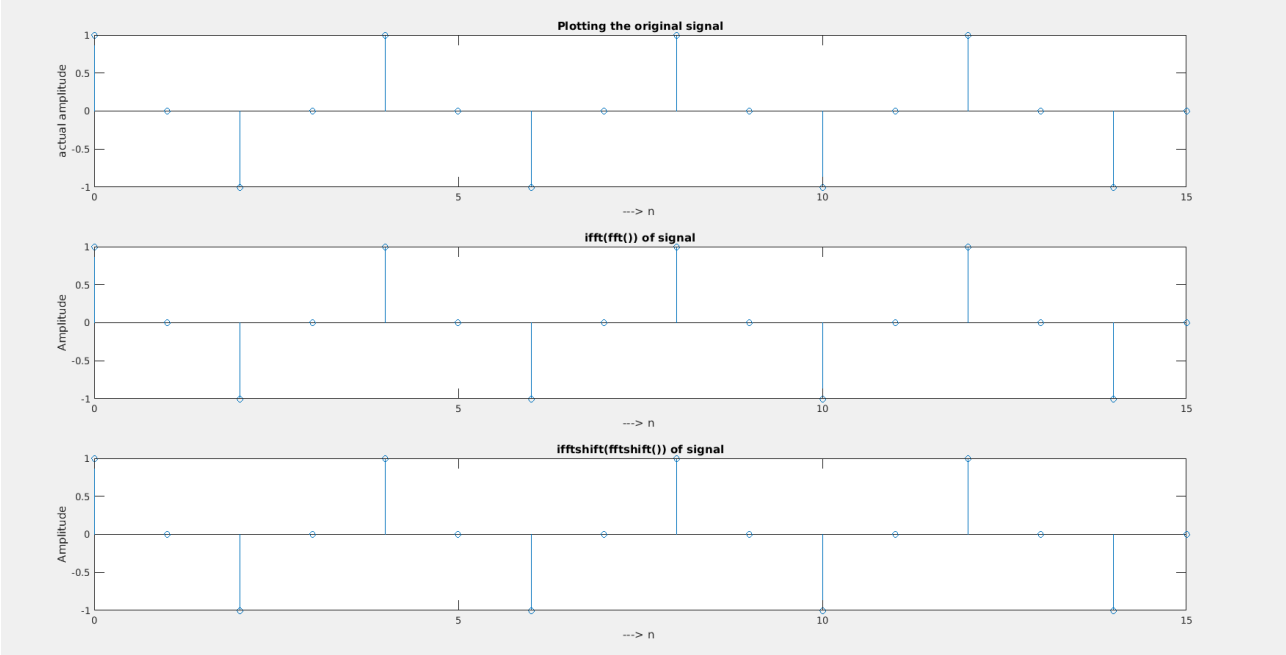


4. Let x[n] = cos(2(0.25)n) for n=(0,1,...15).
(a) Compute the 16-point DFT of x[n] using the fft function
(b) Assuming that x[n] was obtained by sampling at a rate of 1 Hz, plot the DFT (magnitude and phase separately) with the frequency axis in the range [0,2] radians/sec.
(c) Assuming that x[n] was obtained by sampling at a rate of 1 Hz, plot the DFT (magnitude and phase separately) with the frequency axis in the range [0,2] radians/sec. Use the fftshift function for this task.
(d) Check if ifft(fft(x[n]) is x[n].
Ans:

```
clc
clear
% n values on x axis
n = 0 : 15;


% given signal
y = cos(2*pi*0.25*n);


sf = 1;   % sample frequency
ws = 0:sf/numel(n):sf-sf/numel(n); % w axis for fft() from 0 to sf
w = ws*2*pi/sf; % w axis for fft() from 0 to 2pi

shiftws = -sf/2+sf/numel(n):sf/numel(n):sf/2; % w axis for fftshift() from -sf/2 to sf/2
shiftw = shiftws*2*pi/sf; % w axis for fftshift() from -pi to pi

% plotting the signal
figure(1)
```

```
subplot(3,1,1)
stem(n,y)
title('Plotting the original signal')
ylabel('actual amplitude')
xlabel('---> n ')

figure(2)
subplot(2,2,1)
stem(w,abs(fft(y)))
title('Magnitude of fft() of the signal')
ylabel('---> |fft(y[n])|')
xlabel('---> w rad/sed')

subplot(2,2,2)
stem(w,angle(fft(y))*180/pi)
title('Angle of FFT of the signal')
ylabel('---> < fft(y[n])')
xlabel('---> w rad/sed')

subplot(2,2,3)
stem(shiftw,abs(fftshift(y)))
title('Magnitude of fftshift() of the signal')
ylabel('---> |fftshift(y[n])|')
xlabel('---> w rad/sed')


subplot(2,2,4)
stem(shiftw,angle(fftshift(y))*180/pi)
title('Angle of fftshift() of the signal')
ylabel('---> < fftshift(y[n])')
xlabel('---> w rad/sed')

figure(1)
subplot(3,1,2)
stem(n,real(ifft(fft(y))))
title('ifft(fft()) of signal')
ylabel('Amplitude')
xlabel('---> n ')

subplot(3,1,3)
stem(n,real(ifftshift(fftshift(y))))
title('ifftshift(fftshift()) of signal')
ylabel('Amplitude')
xlabel('---> n ')
```





5. Suppose we have two signals defined as follows: x[n] =(0; 1; 2; 3; 4) for n=(0; 1; 2; 3; 4) and; 0 otherwise h[n] = ((1; 1; 1; 1; 1) for n=(0; 1; 2; 3; 4) and; 0 9otherwise

(a) Calculate the circular convolution of x[n] and h[n] manually.

(b) Using Matlab array operations or vector operations compute the circular convolution of x[n] and h[n]

(c) Using Matlab documentation study what the inbuilt function circshift does. Compute the circular convolution of x[n] and h[n] using circshift. Compare with the result obtained in (a) and (b)

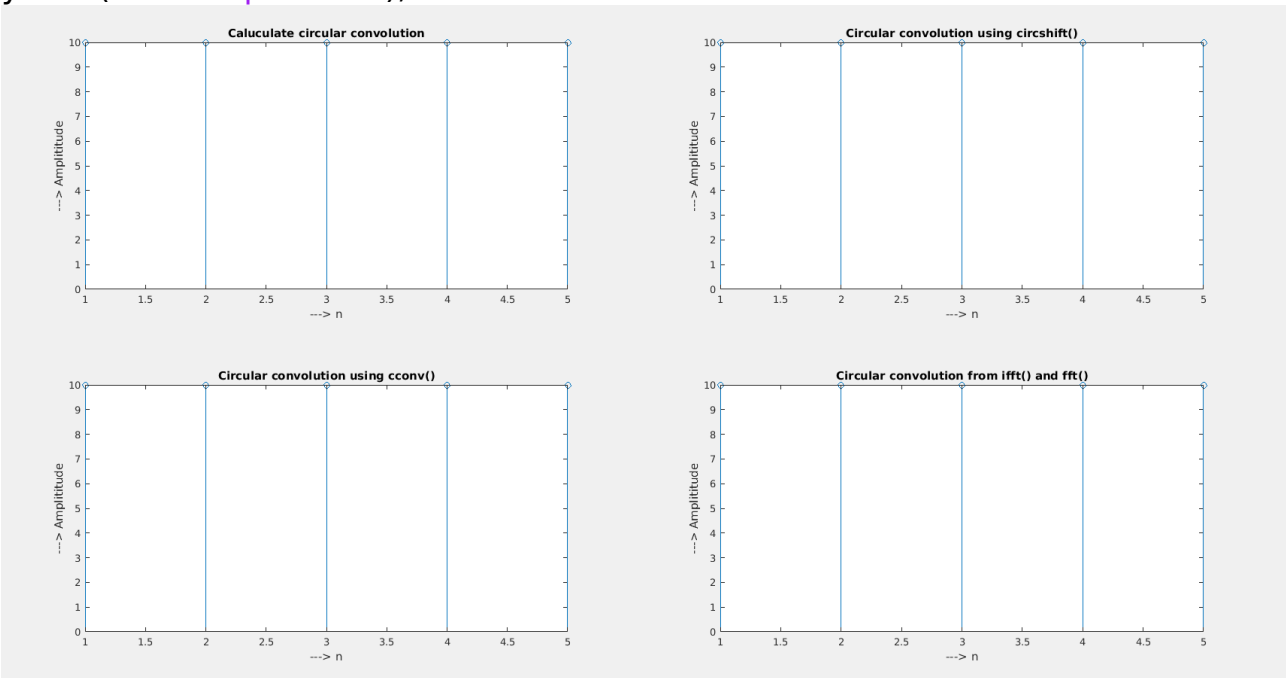
(d) Using Matlab documentation study what the inbuilt function cconv does. Compute the circular convolution of x[n] and h[n] using cconv. Compare with the results obtained in (a), (b), and (c)

(e) Using fft and ifft compute the circular convolution of x[n] and h[n] and

compare with the results obtained in (a), (b), (c), and (d)
Ans:

```
clc;
clear;
% the two signals are h and g
g=[0 1 2 3 4];
h=[1 1 1 1 1];

%a
%{
Manually caluclated circular convolution of two signals is y = [1 1 1 1 1]
for n = [0 1 2 3 4 ]
%}

N1=length(g);
N2=length(h);
N=max(N1,N2);

N3=N1-N2;
% padding zeros to make both the signals of same length
if(N3>0)
    h=[h,zeros(1,N3)];
else
    g=[g,zeros(1,-N3)];
end

%b
% y is the caluclated circular convolution
for n=1:N;
    y(n)=0; % initially assigning zero
    for i=1:N;
        j=n-i+1; % index shifting
        if(j<=0)
            j=N+j; % taking the recoil part for j<0
        end
        y(n)=[y(n)+(g(i)*h(j))]; % summing
    end
end

%c
c = zeros(size(g));
x1 = flip(g)';
x2=h';
for i = 1 : numel(x1)
    c(i) = sum(x2.*circshift(x1,i));
end
% c is the circular convolution by circshift() function which circularly
% shifts the array by given number of times in aurgument

%d
```

```
d = cconv(h,g,N);
% cconv(h,g,N) calculated circular convolution of g,h if N i.e. number of
elements is given

%e
e = ifft(fft(g).*fft(h));


subplot(2,2,1);
stem(y);
title('Caluculate circular convolution')
xlabel('---> n');
ylabel('---> Amplititude');
subplot(2,2,2);
stem(c);
title('Circular convolution using circshift()')
xlabel('---> n');
ylabel('---> Amplititude');
subplot(2,2,3);
stem(y);
title('Circular convolution using cconv()')
xlabel('---> n');
ylabel('---> Amplititude');
subplot(2,2,4);
stem(y);
title('Circular convolution from ifft() and fft()')
xlabel('---> n');
ylabel('---> Amplititude');
```

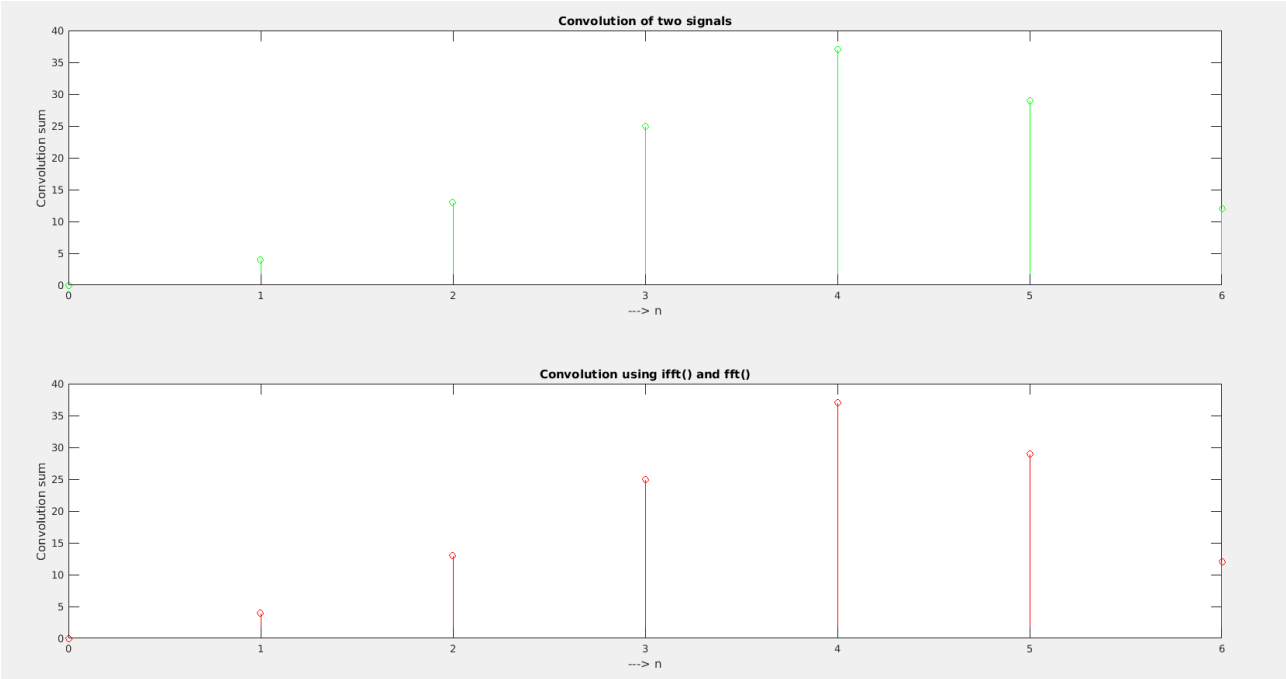


6. Suppose we have two signals de
ned as follows: x[n] = ((0; 1; 2; 3; 4) for n=(0 1 2 3 4) and 0 otherwise h[n] =
((4; 5; 3) for n=(0 1 2) and 0 otherwise
11(a) Calculate the linear convolution of x[n] and h[n] manually.

(b) Using fft and ifft compute linear convolution of x[n] and h[n] and compare with the result obtained in (a)
Ans:

```
clear;
clc;

x = [ 0 1 2 3 4 ] % x1
h = [ 4 5 3 ]   % x2
ax = 0      % starting point of x1
ah = 0      % starting point of x2

% selecting the final start point
a = ax+ah;

% a) manually calculating convolution

m = length(x);
n = length(h);
X = [x,zeros(1,n)];
H = [h,zeros(1,m)];

% number of terms after convolution n + m - 1
for i = 1:m+n-1
    Y(i)=0; % defining the i'th term 0 initially
    for j = 1:m % for all the terms in x
        if(i-j+1>0) % but not the terms which are not overlapped
            Y(i) = Y(i) + X(j)*H(i-j+1); % summation to Y and H is flipped
        end
    end
end

% defining the x axis for the the convolution solution
x1 = a:a+numel(Y)-1;

% plotting
subplot(2,1,1)
stem(x1,Y,'g')
ylabel('Convolution sum')
xlabel('---> n')
title('Convolution of two signals')

% b) second part of solution

% convolution using fft() and ifft()
subplot(2,1,2)
stem([x1 0],ifft(fft(X).*fft(H)),'r')
ylabel('Convolution sum')
xlabel('---> n')
title('Convolution using ifft() and fft()')

%Inference:
```

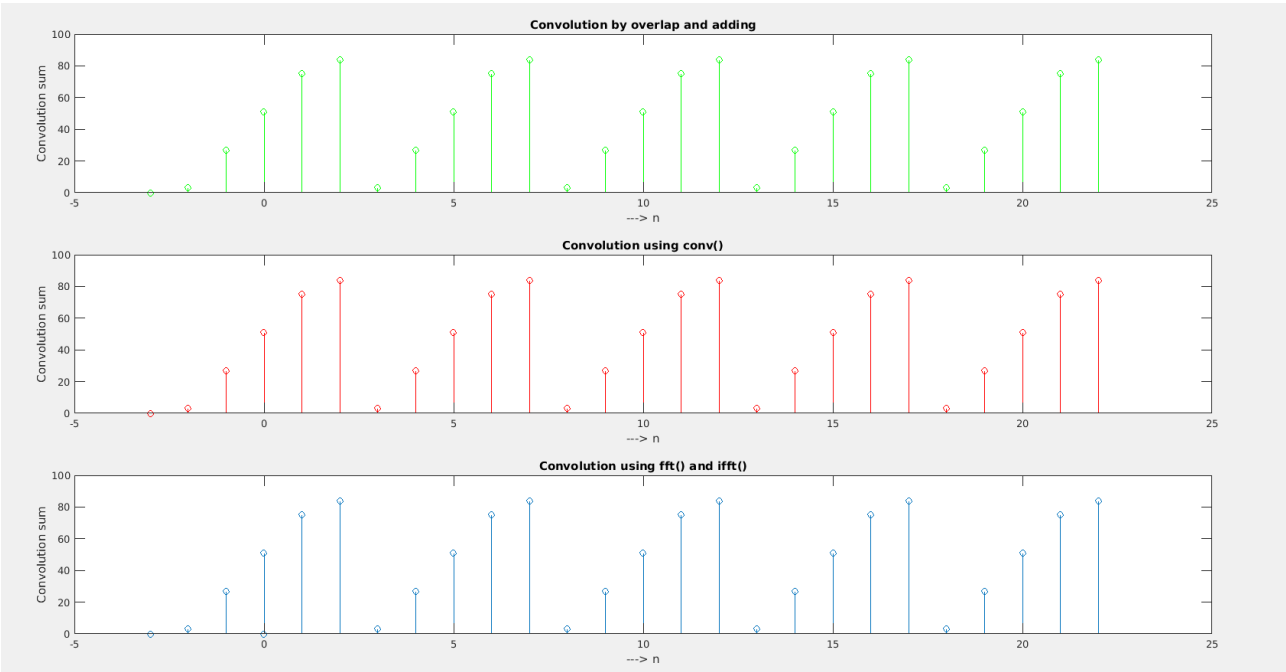
```
% In manual convolution the length of output is n+m-1
% Using fft and ifft length of convolution is n+m
% where n is length of x and m is length of h
```



7. Suppose we have a periodic signal x[n] with period 5 defned as follows: x[n] = (0; 1; 2; 3; 4) for n=(0; 1; 2; 3; 4) So x[n] is (: : : ; 0;1; 2; 3; 4; 0; 1; 2; 3; 4; : : :). Also let h[n] be defined as h[n] = ((3; 2; 1) for n=(0; 1; 2) and; 0 otherwise Let y[n] be x[n] considered for 5 periods from n = 0, i.e., y[n] = x[n] for n (0; 1; 2; 3; 4.....24). Also let y[n] = 0 for all other n.
(a) Compute the linear convolution of y[n] with h[n] in time domain.
(b) Compute the linear convolution of y[n] with h[n] using fft and ifftt. Compare your result with that obtained above.
(c) Review the overlap and add method taught in class from your class notes. Compute the linear convolution of y[n] with h[n] using overlap and add method. Choose at least 3 disjoint segments for the y[n] signal (i.e., express y[n] = y1[n] + y2[n] + y3[n], such that each signal yi[n] has an extent which is disjoint with the extents of the other signals yj [n]) . Check whether the linear convolution that you obtain matches with the results above.
Ans:

```
clear;
clc;
% x[n] = [0 1 2 3 4] from n = 0
% in code y[n]=[x x x x x] is denoted as x
x = [0 1 2 3 4]
x = [ x x x x x ] % y
h = [ 3 21]   % h
ax = -1      % starting point of y
ah = -2      % starting point of h

% selecting the final start point
a = ax+ah;

% a) first part of solution
```

```
m = length(x);
n = length(h);
x1 = [x,zeros(1,n)];
h1 = [h,zeros(1,m)];


% c overlap and adding method
% number of terms after convolution n + m - 1
for i = 1:m+n-1
    Y(i)=0; % defining the i'th term 0 initially
    for j = 1:m % for all the terms in x
        if(i-j+1>0) % but not the terms which are not overlapped
            Y(i) = Y(i) + x1(j)*h1(i-j+1); % summation to Y and H is flipped
        end
    end
end

% defining the x axis for the the convolution solution
xaxis = a:a+numel(Y)-1;

% plotting
subplot(3,1,1)
stem(xaxis,Y,'g')
ylabel('Convolution sum')
xlabel('---> n')
title('Convolution by overlap and adding')

%a

% conv(X,H) is the default function for the convolution of two signals
subplot(3,1,2)
stem(xaxis,conv(x,h),'r')
ylabel('Convolution sum')
xlabel('---> n')
title('Convolution using conv()')


% fft() and ifft() are periodic DFT but since sufficient padding with zeros is done
linear convolution can be calculated. But they have one element extra on the
end
subplot(3,1,3)
stem([xaxis 0],real(ifft(fft(x1).*fft(h1))))
title('Convolution using fft() and ifft()')
xlabel('---> n')
ylabel('Convolution sum')
```


Convolution by overlap and adding
Convolution sum
---> n
Convolution using conv()
Convolution sum
---> n
Convolution using fft() and ifft()
Convolution sum
---> n