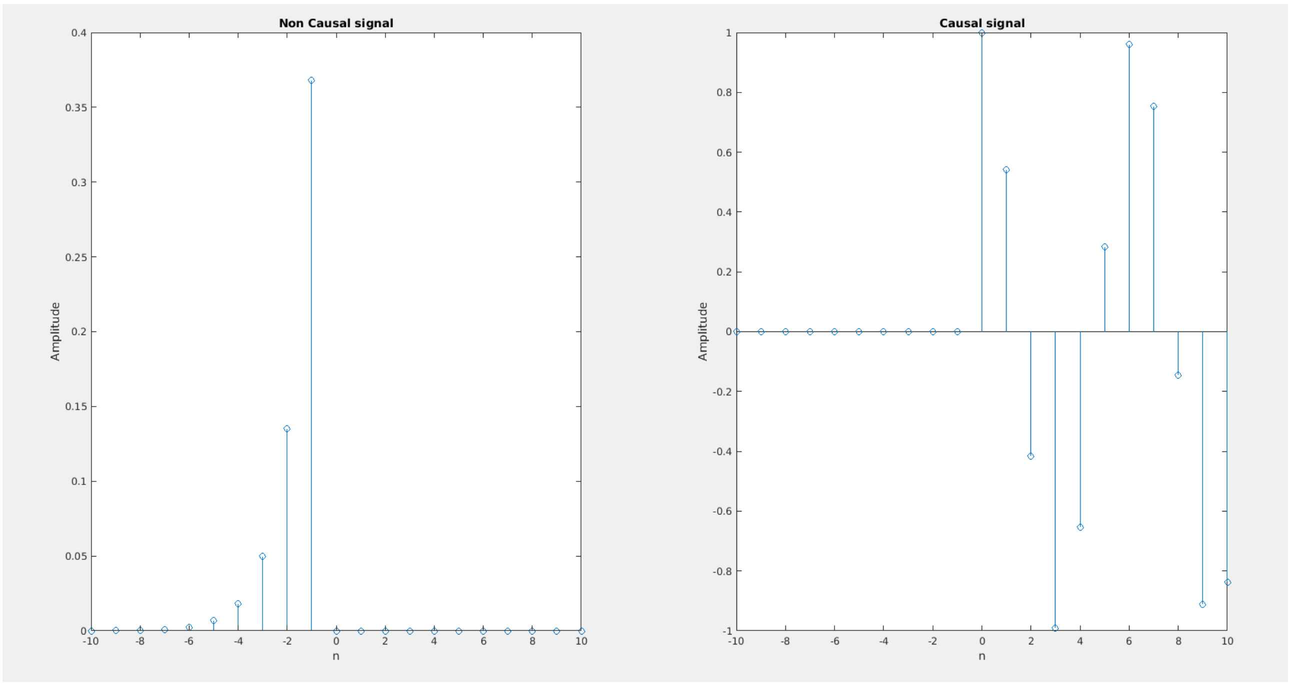
AV 336- Digital Signal Processing Lab

Labsheet 3

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

1. Review what causal and non-causal discrete time impulse responses are. Using Matlab plot
examples of non-causal and causal discrete time impulse responses (2 each).
Ans:

```
clear
clc
% range of n
n = -10 : 10 ;
% defining the signals
causal = cos(n).*double(n>=0 & n<50);
noncausal = exp(n).*double(n<3 & n>-10);
% plotting
figure()

subplot(1,2,1)
stem(n,noncausal)
title('Non Causal signal')
xlabel('n')
ylabel('Amplitude')

subplot(1,2,2)
stem(n,causal)
title('Causal signal')
xlabel('n')
ylabel('Amplitude')
```



2. Let h[n] as defined below be the impulse response of a discrete time LTI system

h[n] =n, for n ∈ {0, 1, 2, . . . , 10} ,

0, otherwise.

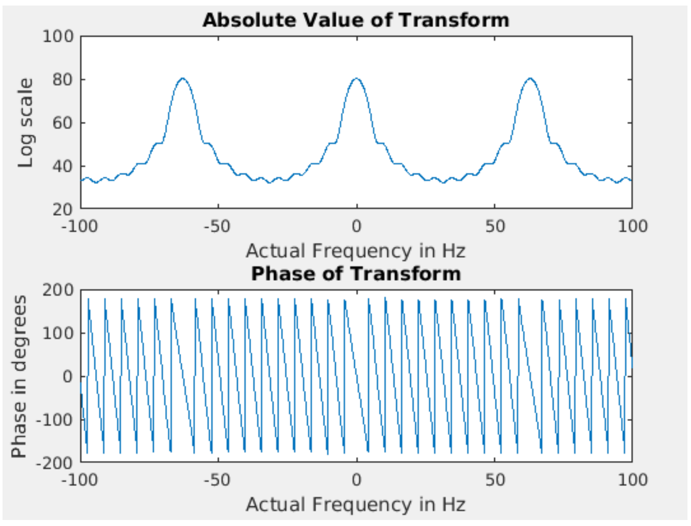
Using Matlab, plot the frequency response of this system (plot the magnitude and phase spectra separately). Note that the frequency response is the DTFT which is a continuous function of the frequency ω. Therefore, a sufficiently "smooth" discretization of ω is needed for obtaining these plots. Observe whether the frequency responses are periodic with ω or not.

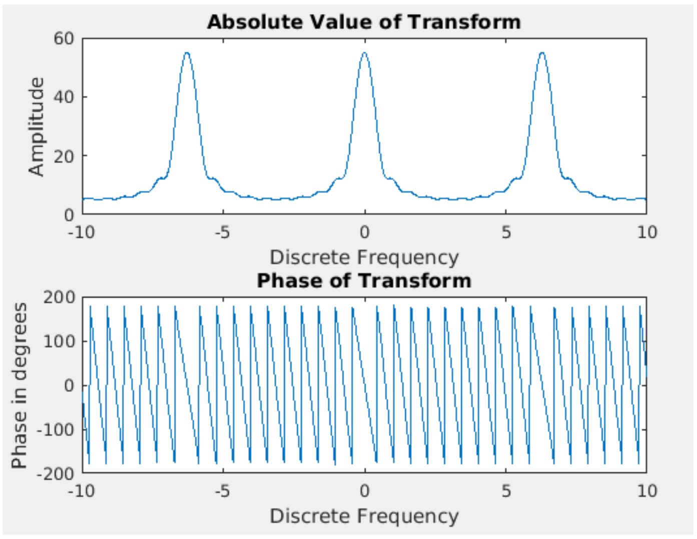
3. Write a Matlab function to plot the frequency response (magnitude and phase) corresponding
to any h[n], rather than the specific case above. The function should take h[n] and two variables "dbscale", "actualfreq" as input. The function should produce the magnitude and phase response as output. If the "dbscale" variable is 1 then the magnitude should be plotted in decibels. If the "actualfreq" variable is not −1, then the ω axis should be plotted in actual frequencies by interpreting the value of the "actualfreq" variable as the sampling frequency used to produce the discrete time signal (i.e., the time duration between two samples is 1/actualf req). Test this function with example inputs (you are free to choose non-trivial inputs) and demonstrate that your implementation is correct. The inputs should be chosen carefully to test the function.

Ans:

```
% This code is for both questions 2 and 3

clear
clc
```

```matlab
% n and h can be any arbitary signal
% n can start and end at any point in Real line
% Corresponding value of h is given
% Here the signal for problem 2 is used
n = -10:10;
h = n.*double(n>=0 & n<=10);
%h = sin(n); % Just an other arbitary h
%
% User defined Inputs
dbscale = 1 ;  % will be True if 1
ActualFreqScale = -1 ;  % will be True if -1
SampleFreq = 10;

% Just plotting
stem(n,h)
title('Impulse response')
xlabel('n')
ylabel('Amplitude')


%Making variable names smaller
AF = ActualFreqScale;
DB = dbscale;

%Range of Discreet frequencies for which Transform is calculated
w = -10:0.01:10;

% Defining Transform variable F with same number of elements as w
F = w*0;

% First for loop to calculate F at each w
for i = 1:numel(w)
    % Second for loop to calculate summation of h[n]*exp(-iwn) at different
    % n
    for j = 1:numel(n)
        F(i) = F(i) + h(j)*exp(-1i*w(i)*n(j));
    end
end

% depending on the user input scale if frequency is changed
if AF == -1
    w = w*SampleFreq;
end

% Amplitude and Phase of Transform are plotted accordingly depending on the
% user input of scale

figure()
subplot(2,1,1)
if DB == 1
    plot(w,20*log(abs(F)))
    title('Absolute Value of Transform')
    ylabel('Log scale')
else
    plot(w,abs(F))
    title('Absolute Value of Transform')
    ylabel('Amplitude')
end

if AF == -1
    xlabel('Actual Frequency in Hz')
else
    xlabel('Discrete Frequency')
end

subplot(2,1,2)
plot(w,(angle(F)*180/pi))
title('Phase of Transform')
ylabel('Phase in degrees')
if AF == -1
    xlabel('Actual Frequency in Hz')
else
    xlabel('Discrete Frequency')
end
```





4. We know that if a discrete time LTI system has an impulse response h[n] and if the input to the system is x[n], then the output (defined as y[n]) is given by the discrete time convolution of x[n] and h[n].
(a) Write a Matlab function \discreteTimeConvolve" that takes as inputs the impulse response and the input signal and produces the output signal as output. Please note that this function should not use any inbuilt Matlab functions to do convolution but rather implement convolution using basic array operations.

(b) Now find out whether any inbuilt Matlab function can be used to directly implement convolution. Demonstrate the use of this function using 3 examples.

Ans)

```
function y= discreteTimeConvolve(x,h)

for i=1:length(h)
   j=length(h)+1-i;
   x2(i)=h(j);
end;
x1=[zeros(1,length(h)) x];
y=[];
for n=1:length(x)+length(h)-1;
    sum=0;
    x1=[x1 0];
    x2=[0 x2];
    for i=6:length(x2)
        sum=sum + x1(i)*x2(i);

    end;
    y(n)=sum;
end;
```

b) conv() is an inbuilt Matlab function used to convolve 2 signals.

Example:

i)

```
x= [ 2 3 4 7 9];
>> h= [4 3 6 8 12];
>> y= discreteTimeConvolve(x,h)

y =

     8    18    37    74   129   137   158   156   108

>> conv(x,h)

ans =

     8    18    37    74   129   137   158   156   108
```

ii)

```
x= [7 3 5 2];
>> h=[4 7 9 32 15 1 4];
>> y= discreteTimeConvolve(x,h)

y =

    28    61   104   294   260   230   170    47    22     8

>> conv(x,h)

ans =
    28    61   104   294   260   230   170    47    22     8
```

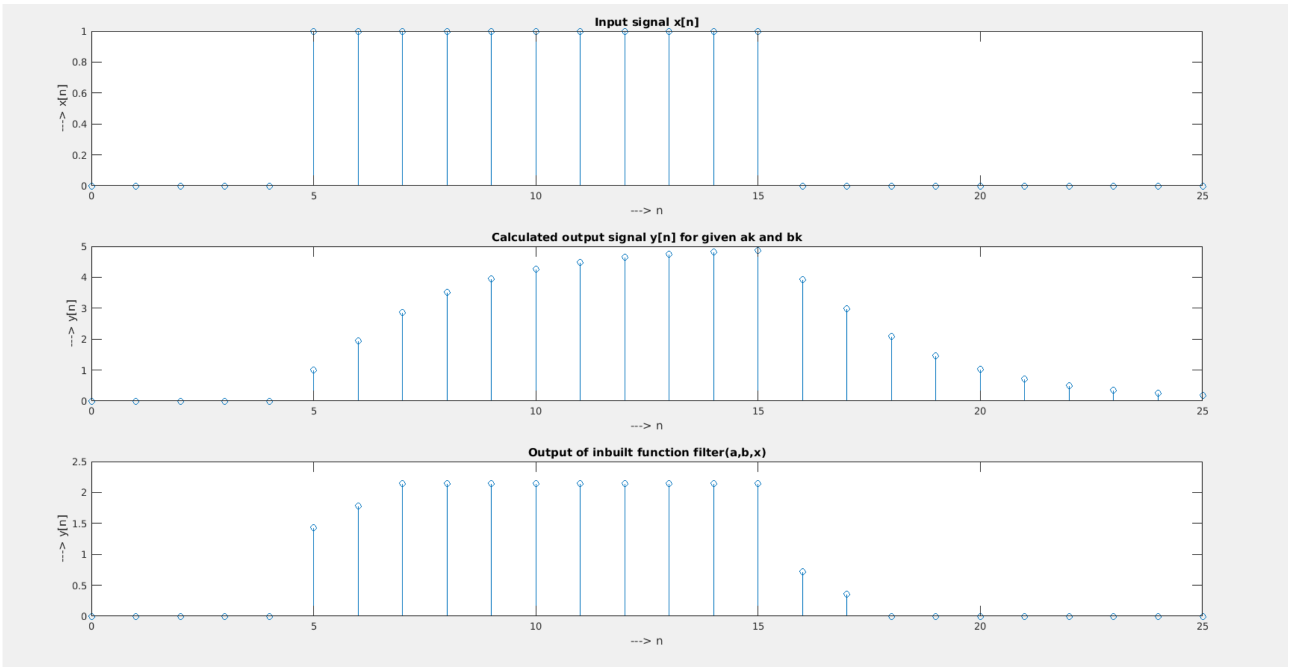
```
iii)	x=[4 7 9 1 4 11];
>> h=[2 1 5 4 3];
>> y= discreteTimeConvolve(x,h)

y =

     8    18    45    62    94    88    62    74    56    33

>> conv(x,h)

ans =

     8    18    45    62    94    88    62    74    56    33
```

5. A discrete time LTI system can also be specified by a constant coefficient difference equation
as follows:

y[n] = sigma(Ak x[n-k]) + sigma(Bk y[n-l])

(a) Implement a Matlab function “discreteTimeCCDE” that takes as input the set of N + 1 coefficients (a k) and M coefficients (b l) and an input signal x[n] and produces as output the signal y[n]. Please note that any initial conditions can be taken as input or assumed. Do not use any inbuilt Matlab functions for this part.
(b) Now find out whether any inbuilt Matlab function can be used to directly the above operation. Demonstrate the use of this function using 3 examples. (Hint: we have seen this in class when we studied IIR filters.)
Ans:

```
clc;
clear;

% define the aks and bks here
ak = [ 1 0.25 0.25 ];
bk = [ 0.7  0 ];

% x is the input signal
n = 0:25;
x = 1*double(n>=5 & n<=15);

% y is output initially 0
y = 0*x;

% a)

% calculating each term of y which has same number of elements as x
for i = 1:numel(x)
    % calculating y(i)
    % first considering aks
    for j = 1:numel(ak)
        % elements of y initially zero in all range and before and after the range
        % so elements with index of y < 0 in not required to be added
        if i-j+1>0
            y(i) = y(i) + ak(j)*x(i-j+1);
        end
    end
    % calculating for bk
    for j = 1:numel(bk)
        if i-j>0 % elements initally are zeros or elements before the y range of y
            y(i) = y(i) + bk(j)*y(i-j);
        end
    end
end


% b

% filter(a,b,x) is the inbuilt function of matlab which takes aks, bks and
% x[n] and calculates the output signal with same number of elements as x[n]
% fourth aurgument is for the initial conditions of y[n] which is
% considered zero
y1 = filter(ak,bk,x,zeros(1,max(length(ak),length(bk))-1));


%plotting te input and output
subplot(3,1,1)
stem(n,x)
title('Input signal x[n]')
xlabel('---> n')
ylabel('---> x[n]')

% plotting calculated output signal
subplot(3,1,2)
stem(n,y)
title('Calculated output signal y[n] for given ak and bk')
xlabel('---> n')
ylabel('---> y[n]')

% plotting output by inbuilt function
subplot(3,1,3)
stem(n,y1)
title('Output of inbuilt function filter(a,b,x)')
xlabel('---> n')
ylabel('---> y[n]')
```



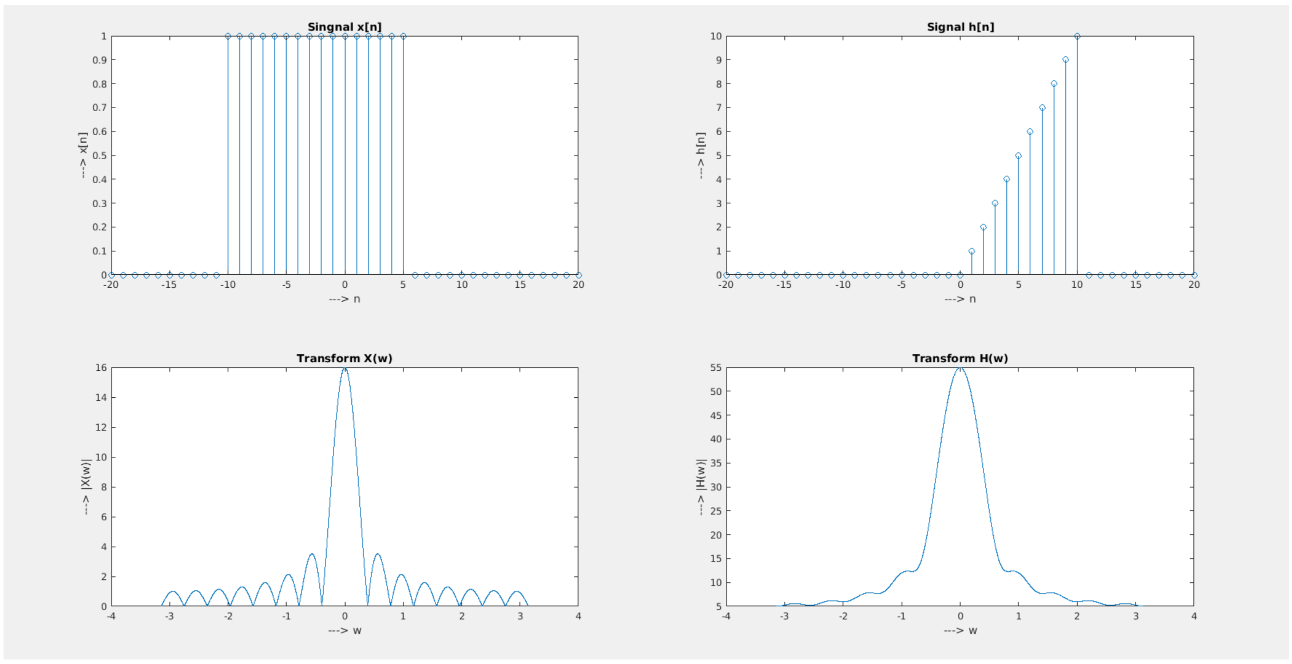
Inference: More input aurguments might me necessory for the inbuilt function to get the required output

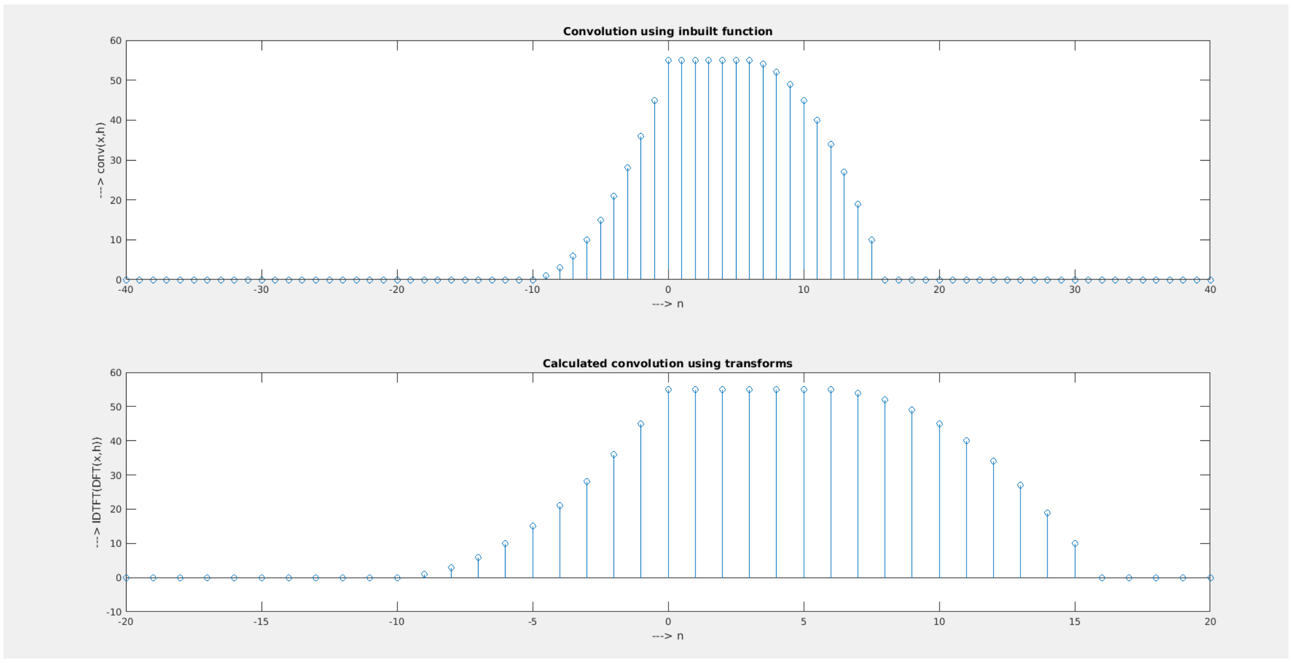
6. In this task, you will explore one of the most important properties of the DTFT. Suppose
h[n] is the impulse response of a discrete time LTI system. The LTI system has output y[n]
when x[n] is the input. Let H(ω), X(ω) and Y (ω) be the DTFTs of h[n], x[n], and y[n]. We
know that Y (ω) = H(ω)X(ω).
(a) Write a Matlab function to compute the DTFT of a signal; note that you have already
done this in (2) in this labsheet. The DTFT that is being computed is a suitably
discretized version of the actual DTFT (defined for the continuous ω).
(b) Using the above function compute H(ω) and X(ω) and their product Y (ω).
(c) Compute y[n] by taking the inverse DTFT (think about how you will implement this on
your computer; are there inbuilt Matlab functions that can help you implement this.)
(d) Compare the y[n] that you have obtained with what is obtained via directly convolving
x[n] and h[n]. Are there any differences? If there are, then why do such differences arise.
Ans:

```
% DTFT, IDTFT and convolution

n = -20:20;    % extent of n for the two signals x[n] and h[n]
```

```
% defining the signals
x = 1*double(n>=-10 & n<=5);
h = n.*double(n>=0 & n<=10);

dw = 0.001;    % resolution of frequency axis
w = -pi:dw:pi; % DTFT from -pi to pi

% calculating the DTFT of x[n] and h[n] simultaniously
for i = 1:numel(w) % at each point on freq axis
    X(i) = 0;
    H(i) = 0;
    % calculating summation at different n
    for j = 1:numel(n)
        X(i) = X(i) + x(j)*exp(-1i*w(i)*n(j));
        H(i) = H(i) + h(j)*exp(-1i*w(i)*n(j));
    end
end

% convolution in time domain is multiplication in frequency domain
Y = X.*H;

% IDTFT of Y
for i = 1:numel(n) % at each n
    y(i)=0;
    % calculating the IDTFT integration at all the w
    for j = 1:numel(w)
        y(i) = y(i) + (0.5/pi)*Y(j)*exp(1j*w(j)*n(i))*dw;
        % 0.5/pi is normalisation factor, dw is resolution
    end
end

% plotting the results

% original signal x[n]
subplot(2,2,1)
stem(n,x)
title('Singnal x[n]')
ylabel('---> x[n]')
xlabel('---> n')

% signal h[n]
subplot(2,2,2)
stem(n,h)
title('Signal h[n]')
xlabel('---> n')
ylabel('---> h[n]')

% DTFT of x[n]
subplot(2,2,3)
plot(w,abs(X))
title('Transform X(w)')
```

```
xlabel('---> w')
ylabel('---> |X(w)|')

% DTFT of h[n]
subplot(2,2,4)
plot(w,abs(H))
title('Transform H(w)')
xlabel('---> w')
ylabel('---> |H(w)|')

figure()

% convolution using inbuilt function
subplot(2,1,1)
% defining the n (x axis)
length = numel(x)+numel(h)-1; % length of convolved signal
start = n(1) + n(1); % one for x and other for h
n1 = start:start+length-1; % x axis
stem(n1,conv(x,h)) % conv(x,h) is the inbuilt function
title('Convolution using inbuilt function')
xlabel('---> n')
ylabel('---> conv(x,h)')

% plotting calculated convolution
subplot(2,1,2)
stem(n,real(y))
title('Calculated convolution using transforms')
xlabel('---> n')
ylabel('---> IDTFT(DFT(x,h))')

%{
Inference:
conv(x,h) gives array as output with number of elements = n(x)+n(h)-1
While calculating with code the range of n which we work on shoud be
carefully selected (should be big enough range.
%}
```


Singnal x[n]
---> x[n]
---> n
Signal h[n]
---> h[n]
---> n
Transform X(w)
---> |X(w)|
---> w
Transform H(w)
---> |H(w)|
---> w


Convolution using inbuilt function
---> conv(x,h)
---> n
Calculated convolution using transforms
---> IDTFT(DFT(x,h))
---> n