DEPARTMENT OF ELECTRONICS AND COMMUNICATION

ATRIA INSTITUTE OF TECHNOLOGY

ATRIA INSTITUTE OF TECHNOLOGY

(Affiliated To Visvesvaraya Technological University, Belgaum)

Anandanagar, Bangalore-24

DSP LAB MANUAL

5th SEMESTER ELECTRONICS AND COMMUNICATION

SUBJECT CODE: 18ECL57

2020-21

Introduction to MATLAB

MATLAB stands for matrix laboratory. It is a technical computing environment for high performance numeric computation and visualization.

MATLAB integrates numeric analysis, matrix computation, signal processing and graphics in an easy to use environment.

MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

Introduction to Digital Signal Processing

A digital signal processing system uses a computer or a digital processor to process the signals. The real life signals are analog and therefore must be converted to digital signals before they can be processed with a computer. To convert a signal from analog to digital, an analog to digital (A/D) converter is used. After processing the signal digitally, it is usually converted to an analog signal using a device called a digital-to-analog(D/A) converter. The below block diagram shows the components of a DSP scheme. This figure contains two additional blocks, one is the antialiasing filter for filtering the signal before sampling and the second is the reconstruction filter placed after the D/A converter. The antialiasing filter ensures that the signal to be sampled does not contain any frequency higher than half of the sampling frequency. If such a filter is not used, the high frequency contents sampled with an inadequate sampling rate generate low frequency aliasing noise. The reconstruction filter removes high-frequency noise due to the "staircase" output of the D/A converter.

The signals that occur in a typical digital signal processing scheme are: continuous-time or analog signal, sampled signal sampled-data signal, quantized or digital signal, and the D/A output signal. An analog signal is a continuous-time, continuous-amplitude signal that occurs in real systems. Such a signal is defined for any time and can have any amplitude within a given range. The sampling process generates a sampled signal. A sampled signal value is held by a hold circuit to allow an A/D converter to change it to the corresponding digital or quantized signal. The signal at the A/D converter input is called a ***sampled data*** signal and at the output is the digital signal. The processed digital signal, as obtained from the digital signal processor , is the input to the D/A converter. The analog output of a D/A converter has "staircase" amplitude due to the conversion process used in such a device. The signal, as obtained from the D/A, can be passed through a reconstruction low pass filter to remove its high-frequency contents and hence smoothen it.

Block diagram of Digital Signal Processing system

There are several toolboxes available from MATLAB. These tool boxes are collections of functions written for special application such as symbolic computation, image processing, statistics and control system design.

In Matrix laboratory there exist 3 major windows which is termed as basic windows as listed 1. Command window.
2. Graphics or figure window.
3. Editor window.

COMMAND WINDOW

This is the main window It is characterized by the Matlab command prompt'>>'. When you launch an application program, MATLAB puts you in this window, all commands including those for running user written programs are typed in this window at the MATLAB prompt.

GRAPHICS or FIGURE WINDOW

The output of all the graphics commands typed in the command window is flushed to the graphics or figure window. The user can create as many as many figure windows, as the system memory will allow.

EDITOR WINDOW

This is where you write, edit, create, and save your own programs in files called 'm-files'. MATLAB provides its own built in editor.

MATLAB FILE TYPES

Mat lab has three types of files for storing.

1. M-files
2. MAT –files
3. MEX-files.

M-files

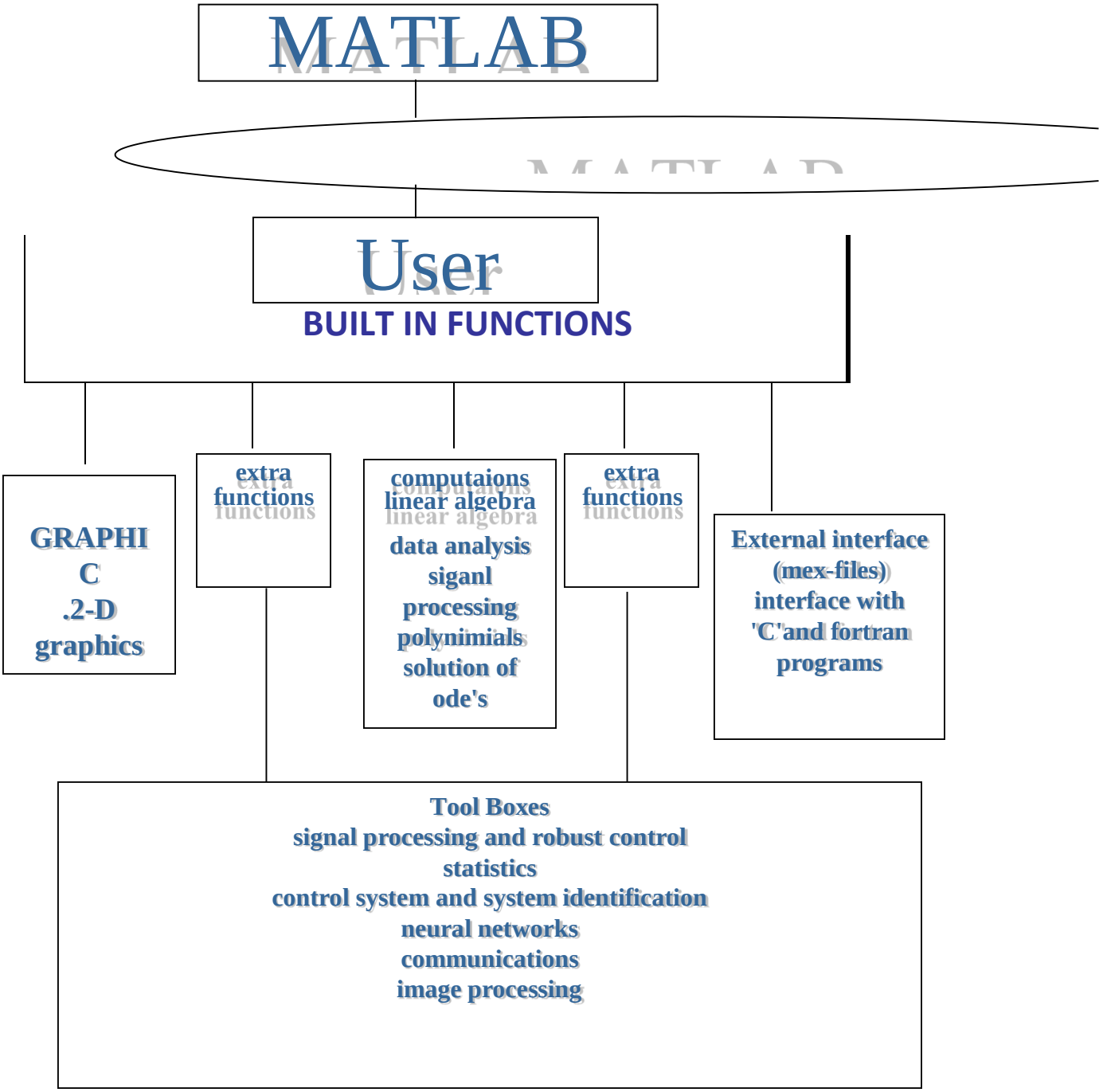
They are standard ASCII text files with a .m extension to the file name. There are two types of m-files namely script file and function file.

MAT-files

These are binary data files, with a .m extension to the file name. MAT files are created by MATLAB when you save the data with the save command. The data is written in a special format that only MATLAB can read.

MEX-files

These are MATLAB callable FORTRAN and C programs with a . mex extension to the file name.


MATLAB
User
BUILT IN FUNCTIONS
GRAPHI
C
.2-D
graphics
extra
functions
computaions
linear algebra
data analysis
siganl
processing
polynimials
solution of
ode's
extra
functions
External interface
(mex-files)
interface with
'C'and fortran
programs
Tool Boxes
signal processing and robust control
statistics
control system and system identification
neural networks
communications
image processing

Part A : MATLAB Programs

Program 1:Verification of the Sampling theorem

Theory: Sampling is a process of converting a continuous time signal (analog signal) x(t) into a discrete time signal x[n], which is represented as a sequence of numbers. (A/D converter). Converting back x[n] into analog (resulting in $x(t)$) is the process of reconstruction.(D/A converter)

Aliasing-A high frequency signal is converted to a lower frequency, results due to under sampling. Though it is undesirable in ADCs, it finds practical applications in stroboscope and sampling oscilloscopes.

Algorithm:

1. Generate original First Sine signal.
2. Generate the second sine signal
3. Add both signal
4. Sample the combined signal at different rates.
5. Display the sampled signal
6. Recover the continuous signal from the samples.

Aim : Sample a Bandlimited continuous time signal bandlimited to Fm Hz under the conditions
1. (i)Nyquist Rate (ii)Twice the Nyquist rate (iii)Half the Nyquist rate
Find the Effect in each of the above case and reconstruction of the signal in time domain
2. Reconstruct the sampled Signal

```
clear all                                          %removes all variables from the
workspace
clc                                                 %clears the command window
close all                                           %closes all open figure
windows
f1=input('The Frequency of the first sine wave in Hz :');%Frequency of the first sine wave
A1=input('Amplitude of first sine wave:');          %Amplitude of first sine wave
f2=input('The frequency of the second sine wave in Hz:'); frequency of the second sine wave
A2=input('amplitude of second Sine wave:');         %amplitude of first Sine wave
fs=input('enter the sampling frequency in Hz:');    %the sampling frequency
p=input('the number periods for display=');         %number periods for display
teta=input('Phase Shift for First Wave:');          %Phase Shift for First Wave

t=0:0.0001:p*(1/f1);                                %time index
xf1=A1*cos(2*pi*f1*t + teta);
subplot(2,2,1);                                     %first sine wave plot
plot(t,xf1);
xlabel('time in seconds-->');
ylabel('amplitude-->');
title('plot of first cosine wave');
```

```
axis([0 (p*(1/f1)) -1.2 1.2]);
grid on;

xf2=A2*cos(2*pi*f2*t);
subplot(2,2,2);                                          %second sine wave plot
plot(t,xf2,'r -',t,xf1)
xlabel('Time in seconds-->');
ylabel('Amplitude-->');
title('plot of Second Cosine wave');
axis([0 (p*(1/f1)) -1.2 1.2]);
grid on;

xsum=xf1+xf2;                                            %summation of two sine
waves
subplot(2,2,3);                                          %plot of summed wave
plot(t,xsum);
xlabel('Time in seconds-->');
ylabel('Amplitude-->');
title('Plot of summed cosine waves');
axis([0 (p*(1/f1)) -2.2 2.2])
grid on;

ts=0:1/fs: p*(1/f1);                                     %sampling
xs=A1*cos(2*pi*f1*ts+teta)+A2*cos(2*pi*f2*ts);
nt=0:length(ts)-1;
subplot(2,2,4)
stem(nt,xs);
axis([0 (length(nt)-1) -2.2 2.2]);
title('plot of sampled signal');
grid on;

ts=0:1/fs:1;
xs=A1*cos(2*pi*f1*ts)+A2*cos(2*pi*f2*ts);
xfftmag2=abs(fft(xs));                                   % Frequency domain representation
figure(2),subplot(2,1,1)
plot(xfftmag2);
subplot(2,1,1)
xlabel('frequency in hz-->');
ylabel('Amplitude-->');
title('the plot of cosine wave in the frequency domain');
grid on;

T=(1/fs);                                                % Reconstruction of original signal
n=(0:T:1-T);
xs=A1*cos(2*pi*f1*n)+A2*cos(2*pi*f2*n);
t=linspace(-0.5,1.5,2000);
ya=sinc((1/T).*t(:,ones(size(n)))-(1/T).*n(:,ones(size(t)))).*xs;
subplot(2,1,2)
plot(n,xs)
xlabel('frequency in hz-->');
```

```
ylabel('Amplitude-->');
title('the plot of reconstruction of original signal');
axis([0 0.12 -2 2]);
grid on;
```

Result : Output will be displayed on the command window and figure window .

PROGRAM 2A

Aim: I: To find the linear convolution of the two discrete sequences and verification of its properties.

(1) Commutative Property : A Conv B =B Conv A
(2) Distributive Property : A×(B conv C) = A×(B) conv A×(C)
(3) Associative Property : A Conv(B + C) = (A conv B) +(A conv C)

II: :To find the Circular convolution of the two discrete sequences and verification of its properties.

Procedure:-

1. Read the input sequence, x[n] and plot
2. Read the impulse response of the system, h[n] and plot
3. Convolve the two results and plot them

Description:-

Linear Convolution involves the following operations.

1. Folding
2. Multiplication
3. Addition
4. Shifting

These operations can be represented by a Mathematical Expression as follows:

$$\mathbf{y[n] = \sum x[k]h[n-k]} \text{ where}$$

x[ ]= Input signal Samples **h[ ]=** Impulse response co-efficient. **y[ ]=** Convolution output.

n = No. of Input samples **h =** No. of Impulse response co-efficient.

Eg: x[n] = {1, 2, 3, 4}

h[k] = {1, 2, 3, 4}

Where: n=4, k=4. : Values of n & k should be a multiple of 4.
If n & k are not multiples of 4, pad with zero's to make multiples of 4

r= n+k-1 : Size of output sequence.
= 4+4-1
= 7

r=	0	1	2	3	4	5	6
n= 0	x[0]h[0]	x[0]h[1]	x[0]h[2]	x[0]h[3]			
1		x[1]h[0]	x[1]h[1]	x[1]h[2]	x[1]h[3]		
2			x[2]h[0]	x[2]h[1]	x[2]h[2]	x[2]h[3]	
3				x[3]h[0]	x[3]h[1]	x[3]h[2]	x[3]h[3]

Output: y[r] = { 1, 4, 10, 20, 25, 24, 16}.

NOTE: At the end of input sequences pad ‘n’ and ‘k’ no. of zero’s

MATLAB Code

I: To find the linear convolution of the two discrete sequences and verification of its properties.

```
clear  all                                  %removes all variables from the workspace
clc                                         %clears the command window
close all                                    %closes all open figure windows
x=[2 1 2 1 ];                                %first sequence is x with time index nx
nx=0:3;
h=[1 2 3 4 ];                                %second sequence is x with time index nh
nh=0:3;
b=[5 6 7 8 ];                               %second sequence is x with time index nh
nb=0:3;
y=conv(x,h);                                %convolution result
ny=[nx(1)-nh(1):nx(length(x))+nh(length(h))];        %corresponding time index calculation
                        %plot the two sequences and the corresponding convolution

%Output
subplot(3,1,1);
stem(nx,x);
xlabel('n-->');
ylabel('x-->');
title('first sequence');
grid on;
subplot(3,1,2);
stem(nh,h);
xlabel('n-->');
ylabel('h-->');
title('second sequence');
grid on;
subplot(3,1,3);
stem(ny,y);
xlabel('n-->');
ylabel('y-->');
title('Linear Convolved sequence');
grid on;
disp('First Sequence is :') ; x
disp('Second sequence is :') ; h
disp('Linear convolution is :');y
```

(1) Commutative Property : A Conv B =B Conv A

```
m=conv(x,h);                                    %Left hand convolution result
nm=[nx(1)-nh(1):nx(length(x))+nh(length(h))]; %corresponding time index calculation

o= conv(h,x);                                    %Right hand convolution result
```

```
nm=[nx(1)-nh(1):nx(length(x))+nh(length(h))];    %corresponding time index calculation
disp('Left hand Sequence'); m
disp('Left hand Sequence'); o
 if(m==o)
    disp('Commutative Property is proved')
 else
    disp('Commutative Property is not proved')
end
```

(2) Distributive Property : A×(B conv C) = (A×B) conv (A×C)

```
A=[2 1 2 1 ];                          %first sequence is A with time index nA
B=[1 2 3 4 ];                          %second sequence is B with time index nB
C=[5 6 7 8 ];                         %Third sequence is C with time index nC
A1=4;
LHS1=conv(B,C);
LHS= A1.*LHS1
RHS1= A1.*B;
RHS2 = A1.*C;
RHS = conv(RHS1,RHS2)

if(LHS==RHS)
disp('Distributive Property is proved')

else
disp('Distributive Property is not proved');
end
```

(3)Associative Property : A conv(B +C) = (A conv B) +(A Conv C)

```
LHS1= B+C;
LHSA=conv(A,LHS1);
RHS1= conv(A,B);
RHS2= conv(A,C);
RHSA= RHS1+RHS2;

if(LHSA==RHSA)
disp('Associative Property is proved')
 else
disp('Associative Property is not proved');
end
```

Result : Output will be displayed on the command window and figure window .

PROGRAM 2B: CIRCULAR CONVOLUTION OF TWO SEQUENCES.

Procedure:-

1. Enter the sequence x[n]
2. Enter the sequence y[n]
3. Find the lengths of x[n] and y[n] ie; Nx and Ny respectively

4. Check Nx=Ny : proceed if equal
5. Initialize a loop variable number of output points
6. For each out sample , access the samples of y[n] in the cyclic order
7. Find the sum of products of x[n] and cyclically folded and shifted y[n]

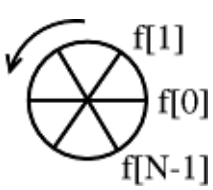
Description:-

Steps for Cyclic Convolution

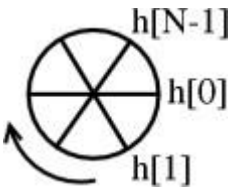
Steps for cyclic convolution are the same as the usual convolution, except all index calculations are done "mod N" = "on the wheel"

Steps for Cyclic Convolution

Step1: "Plot f[m] and h[−m]



Subfigure 1.1



Subfigure 1.2

Step 2: "Spin" h[−m] *n* times Anti Clock Wise (counter-clockwise) to get h[n-m] (i.e. Simply rotate the sequence, *h*[*n*], clockwise by *n* steps)

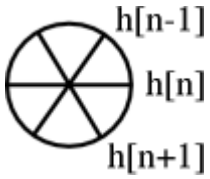
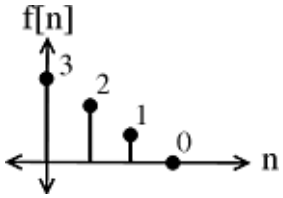


Figure 2: Step 2

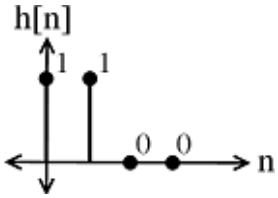
Step 3: Point wise multiply the *f*[*m*] wheel and the *h*[n−m] wheel. Sum=*y*[*n*]

Step 4: Repeat for all 0≤*n*≤*N*−1

Example 1: Convolve (n = 4)



Subfigure 3.1



Subfigure 3.2

*Fi*gure 3: Two discrete-time signals to be convolved.

- *h*[−*m*] =

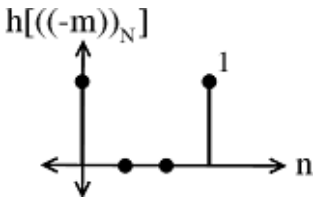


Figure 4

Multiply $f[m]$ and sum to yield: $y[0] = 3$

- $h[1-m]$

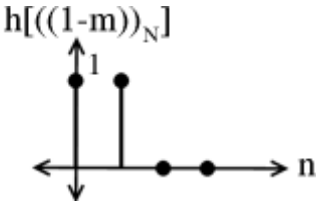


Figure 5

Multiply $f[m]$ and sum to yield: $y[1] = 5$

- $h[2-m]$

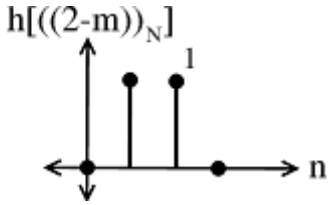


Figure 6

Multiply $f[m]$ and sum to yield: $y[2] = 3$

- $h[3-m]$

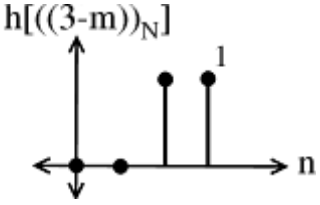


Figure 7

Multiply $f[m]$ and sum to yield: $y[3] = 1$

MATLAB Code

```
clear all                                   %removes all variables from the workspace
clc                                         %clears the command window
close all                                   %closes all open figure windows

x1=[2 1 2 1];                                       %first sequence x1
nx1=0:length(x1)-1;                         %time index for first sequence
subplot(3,1,1);                                     %plot first sequence
stem(nx1,x1);
xlabel('n-->');
ylabel('x1--');
title('First sequence');
grid on;

x2=[1 2 3 4 ];                                      %second sequence x2
nx2=0:length(x2)-1;                   %time index for second sequence
subplot(3,1,2);                                      %plot second sequence
stem(nx2,x2);
xlabel('n-->');
ylabel('x2-->');
title('First sequence');
grid on;

N=4;                                        %circular convolution order is N

if length(x1)>N                         %check for length of x1 and x2 with respect to n
error('N must be >=the length of x1');
end;

if length(x2)>N
error('N must be >=the length of x2');
end;

x1=[x1,zeros(1,N-length(x1))];              %if N >length(x1) or length(x2)Zeropad
x2=[x2,zeros(1,N-length(x2))];
m=[0:N-1];
for  n=0:N-1,
      y(n+1)=sum(x2(mod(n-m, N)+1).*x1);                    %plot convolved sequence
end;                                    %circularly fold x2,x2(-m,mod n)     ,for  m=0,1,...,n-1

%convolve the circular shifted folded sequence with x1
ny=0:3;%y(n)=sum(x1(m)*x2((n-m)mod n))
subplot(3,1,3);
stem(ny,y);
xlabel('n-->');%x2=x2(mod(-m,n)+1);
ylabel('y-->');
title('First sequence');
grid on;
disp('First seq is:');x1
disp('second seq is :');x2
```

disp('circular convolution is :');y

Properties

(1) Commutative Property : A Conv B =B Conv A
(2) Distributive Property :A×(B conv C) = A×(B) conv A×(C)
(3) Associative Property :A Conv(B +C) = (A conv B)+(A conv C)

Note: code to check the properties Circular Convolution same as Linear Convolution

Sl. No	Property	Linear Convolution	Circular Convolution
1	Commutative	Yes	Yes
2	Distributive	Yes	Yes
3	Associative	yes	Yes

Result : Output will be displayed on the command window and figure window .

PROGRAM 3A

Aim: Find the Autocorrelation and Cross correlation of given two sequences and verify their Properties

Theory: Correlation is mathematical technique which indicates whether 2 signals are related and in a precise quantitative way how much they are related. A measure of similarity between a pair of energy signals x[n] and y[n] is given by the cross correlation sequence $r_{xy}[l]$ defined by :

$$r_{xy}[l]=\sum_{n=-\infty}^{\infty} x[n]y[n-l];\ l=0,\pm1,\pm2,\ldots$$

The parameter '*l*' called 'lag' indicates the time shift between the pair.
Autocorrelation sequence of x[n] is given by :-

$$r_{xx}[l]=\sum_{n=-\infty}^{\infty} x[n]x[n-l];\ l=0,\pm1,\pm2,\ldots$$

Some of the properties of autocorrelation are enumerated below –
1. The autocorrelation sequence is an even function i.e., $r_{xx}[l]=r_{xx}[-l]$
2. At zero lag, i.e., at l=0, the sample value of the autocorrelation sequence has its maximum value (equal to the total energy of the signal ex) i.e.,

$$r_{xx}[l]\le r_{xx}[0]=\varepsilon_x=\sum_{n=-\infty}^{\infty} x^2[n]$$

This is verified in Fig. where the autocorrelation of the rectangular pulse (square) has a maximum value at l=0. All other samples are of lower value. Also the maximum value = 11 = energy of the pulse [$1^2+1^2+1^2$..].

3. A time shift of a signal does not change its autocorrelation sequence. For example, let y[n]=x[n-k]; then $r_{yy}[l] = r_{xx}[l]$ i.e., the autocorrelation of x[n] and y[n] are the same regardless of the value of the time shift k. This can be verified with a sine and cosine sequences of same amplitude and frequency will have identical autocorrelation functions.

4. For power signals the autocorrelation sequence is given by :

$$r_{xx}[l]=\lim_{k\to\infty}\frac{1}{2k+1}\sum_{n=-k}^{k}x[n]x[n-l];\ l=0,\pm1,\pm2,\ldots$$

and for periodic signals with period N it is :

$$r_{xx}[l]=\frac{1}{N}\sum_{n=0}^{N-1}x[n]x[n-l];\ l=0,\pm1,\pm2,\ldots$$

and this **$r_{xx}[l]$ is also periodic** with N. This is verified in Fig. where we use the periodicity property of the autocorrelation sequence to determine the period of the periodic signal y[n] which is x[n] (=cos(0.25*pi*n)) corrupted by an additive uniformly distributed random noise of amplitude in the range [-0.5 0.5]

Procedure:-

1. Read the input sequence
2. Auto correlate the signal using xcorr(x,x)
3. Display the Autocorrelation result in suitable axis
4. verify the correlation property: Rxx(0)=energy(x)
5. verify the property : Rxx is an even function

Part A : Auto Correlation-MATLAB Code

```
Clear all                          %removes all variables from the workspace
Clc                                %clears the command window
close all                          %closes all open figure windows

x=[1 2 1 1];                       %read the input signal
n=0:1:length(x)-1;                 %define the axis
subplot(2,1,1);                    %plot the signal
stem(n,x);
xlabel('n-->');
ylabel('x-->');
title ('sequence x');
grid on;

Rxx=xcorr(x,x)                     %autocorrelate the signal
nRxx=1:length(x)+length(x)-1;      %axis for the autocorrelation result
subplot(2,1,2);                    %Display the result
stem(nRxx,Rxx);
xlabel('nRxx-->');
```

```
ylabel('autocorrelation of x-->');
title('Autocorrelated sequence of x');
grid on;
```

%Verification of the autocorrelation properties
Property-1:Rxx(0) gives the energy of the signal

```
energy=sum(x.^2);                    %Energy Of the signal = SUM(Squares of x)
center_index=ceil(length(Rxx)/2);          %get the index of the center value
Rxx_0 = Rxx(center_index)                %access center value rxx(0)
 %check if the rxx(0)=energy
if Rxx_0==energy;
disp('Rxx(0) gives energy--property-1 is proved');
else
disp('Rxx(0) gives energy--property-1 is not proved');
end
  %property-2:Rxx is even
Rxx_Right=Rxx(center_index:1:length(Rxx))
Rxx_Left=Rxx(center_index:-1:1)
ifRxx_Right==Rxx_Left
disp('Rxx is even');
else
disp('Rxx is not even');
end
```

Result : Output will be displayed on the command window and figure window .

PROGRAM 3B
Part B: To find the Crosscorrelation of given sequences

Theory:

Cross Correlation has been introduced in the last experiment. Comparing the equations for the linear convolution and cross correlation we find that

$$r_{xy}[l]=\sum_{n=-\infty}^{\infty} x[n]y[n-l]=\sum_{n=-\infty}^{\infty} x[n]y[-(l-n)]=x[l]*y[-l] \quad . \qquad \text{i.e.,}$$

convolving the reference signal with a folded version of sequence to be shifted (y[n]) results in cross correlation output. (Use 'fliplr' function for folding the sequence for correlation). The properties of cross correlation are

1) The cross correlation sequence sample values are upper bounded by the inequality

$$r_{xx}[l] \le \sqrt{r_{xx}[0]r_{yy}[0]} = \sqrt{\varepsilon_x \varepsilon_y}$$

2) The cross correlation of two sequences x[n] and y[n]=x[n-k] shows a peak at the value of k. Hence cross correlation is employed to compute the exact value of the delay k between the 2 signals. Used in radar and sonar applications, where the received signal reflected from

the target is the delayed version of the transmitted signal (measure delay to determine the distance of the target).

3) The ordering of the subscripts xy specifies that x[n] is the reference sequence That remains fixed in time, whereas the sequence y[n] is shifted w.r.t x[n]. If y[n] is the reference sequence then $r_{yx}[l]=r_{xy}[-l]$. Hence $r_{yx}[l]$ is obtained by time reversing the sequence $r_{xy}[l]$.

Procedure:-

1. Read the input sequence
2. Cross correlate the signal
3. Display the cross correlate result in suitable axis
4. Verify the correlation property : Rxx(0)=energy(x)
5. Verify the property : Rxx is an even function

MATLAB code

```
clear  all                  %removes all variables from the workspace
clc                         %clears the command window
close all                   %closes all open figure windows
x=[2 -1 3 7 1 2 -3]         %read the input sequences
y=[1 -1 2 -2 4 1 -2 5]
n1=-4:1:2;                  %define the axis
n2=-4:1:3;
subplot(3,1,1);             %plot the signal
stem(n1,x);
title('sequence-x');
grid on;
subplot(3,1,2);
stem(n2,y);
title('sequence-y');
grid on;

r=conv(x,fliplr(y))         %crosscorrelate the sequences using convolution
nr=-8:1:5
subplot(3,1,3)
stem(nr,r)
title('cross correlation sequence')
grid on;

c=xcorr(x,y)                %verification by correlation function
```

Result : Output will be displayed on the command window and figure window.

PROGRAM 4

Aim: Part A: Solving the given difference Equation

Procedure:-

1. Rewrite the given difference equation to have only the output terms on the LHS and input terms to be on RHS
2. Create a Matrix of Y coefficients, a
3. Create a Matrix of X coefficients, b
4. Generate the input sequence x(n)
5. Find the output of the system for the input sequence x(n)
6. Plot the input and the output

The Difference Equation is y(n)-0.9y(n-1)=x(n).
The Input Sequence is x(n)=U(N)-U(n-10).
To Find The Impulse Response h(n) and The Output y(n) And Find The Transfer Function of The Given Difference Equation $H(Z)=\frac{X(Z)}{Y(Z)}$. Numerator Polynomial is 'A' and Denominator Polynomial is 'B'. For In this Example B=[1] AND A=[1 -0.9]

MATLAB CODE:

```
Clear all                                   %removes all variables from the workspace
clc                                         %clears the command window
close all                                   %closes all open figure windows
b=[1];
a=[1 -0.9];

 %find the impulse response h(n) generate impulse sequence
n=[-5:50];                                  %time index
x=[(n==0)];                                 %x is the impulse sequence
h=filter(b,a,x) ;                  %find the output of the system(impulse response) figure(1)

%plot the input and the output
subplot(2,1,1);
stem(n,x);
xlabel('n-->');
ylabel('x-->');
title('impulse sequence');
grid on;
subplot(2,1,2);
stem(n,h);
xlabel('n-->');
ylabel('h-->');
title('Impulse response');
grid on;

x1=[(n>=0)];                        %generate the input sequence x(n)
x2=(-1)*[(n-10)>=0];
x=x1+x2;
y=filter(b,a,x);                    %find the output of the system for the input sequence x(n)
figure(2)                           %plot the input and the output
subplot(2,2,1);
stem(n,x1);
```

```
xlabel('n-->');
ylabel('x-->');
title('Input sequence x1(n)');
grid on;
subplot(2,2,2);
stem(n,x2);
xlabel('n-->');
ylabel('x-->');
title('output sequence y(n)');
subplot(2,2,3);
stem(n,y)
```

Result : Output will be displayed on the command window and figure window .

PROGRAM 4B

Part B: Solve The Given Difference Equation with initial conditions

%y(n)-1.5y(n-1)+0.5y(n-2)=x(n),n>=0, x(n)= $(\frac{1}{4})^n$ *u(n) subject to y(-1)= 4 and y(-2)=10

Calculate and Plot The Output Of The System And Find The Transfer Function of The Given Difference Equation H(Z)=$\frac{X(Z)}{Y(Z)}$Numerator Polynomial is B And Denominator Polynomial is A For The Above Example %b=[1] AND a=[1 -1.5 0.5]

```
Clear all                          %removes all variables from the workspace
clc                                %clears the command window
close all                          %closes all open figure windows

b=[1];
 a=[1 -1.5 0.5];
yic=[4 10];                        %initial conditions for y are obtained
xic=filtic(b,a,yic);               %find the initial conditions for the input x

n=0:5;                             %generate the discrete time index
   x=(1/4).^n;                     %generate the given input x
   y=filter(b,a,x,xic);            %find the output of the system for the input sequence x
figure(1)                          %plot the input and the output
subplot(2,1,1);
stem(n,x);
grid on;
xlabel('n-->');
ylabel('x-->');
title('Input Sequence');
subplot(2,1,2);
stem(n,y);
xlabel('n-->');
ylabel('y-->');
title('Output sequence');
grid on;
```

Result : Output will be displayed on the command window and figure window.

PROGRAM 5

Aim: Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum(using DFT Equation and verify it by built-in routine).

Procedure:-

1) Enter the number of points, N
2) Enter the input sequence elements, x[n]
3) Create a vector for the sample index, 'n'
4) Initialize loop variable, 'k' for the DFT samples X(k)
5) Calculate the twiddle factor for each 'k'
6) Multiply x[n] and the twiddle factors , elements-by-element
7) Sum all the products, assign to X(k)
8) Plot the magnitude and phase spectrum
9) Verify the results with built in function

COMPUTATION OF DFT

Note:-

$$x(n) \xrightarrow[\text{DFT}]{N} x(k)$$

$$x(k) = \sum_{n=0}^{N-1} x(n) w_N^{nk} \qquad k = 0,1, \text{---} N-1$$

where, $w_N = e^{-j\,2\,\pi\,/\,N}$

$$x(k) = \sum_{n=0}^{N-1} x(n) e^{-j\,2\,\pi\,k\,n\,/\,N} \qquad k = 0,1, \text{-----} N-1$$

Calculations:-

x(n) = [2 3 4 5] **N = 4**

$$x(k) = \sum_{n=0}^{3} x(n) e^{-j\,2\,\pi\,k\,n\,/\,4} \qquad k = 0,1,2,3$$

$$x(k) = x(0)e^{-j2\,\pi\,k\,*\,0\,/\,4} + x(1)\, e^{-j\,2\,\pi\,k\,*1\,/\,4} + x(2)\, e^{-j\,2\,\pi\,k\,*2\,/\,4} + x(3)\, e^{-j\,2\,\pi\,k*3\,/\,4}$$

$x(k) = 2 + 3e^{-j\,\pi\,k/\,2} + 4e^{-j\,\pi\,k} + 5e^{-j\,3\,\pi\,k/\,2}$

x(0) = 2 + 3 + 4 +5 = **14**

$x(1) = 2 + 3e^{-j\,\pi\,/\,2} + 4e^{-j\,\pi} + 5e^{-j3\pi\,/2}$ **$\{e^{j\theta} = \cos\theta + j\sin\theta\}$**

$= 2 + 3\,[\cos \pi/2 - j \sin \pi/2] + 4[\cos \pi - j \sin \pi] + 5[(0)\,3\,\pi/2 - j\sin\,3\,\pi/2]$ **x(1) = -2 +2j**

$\{e^{-j\theta} = \cos\theta - j\sin\theta\}$

x(2) $= 2 + 3e^{-j\,\pi\,/\,2} + 4e^{-j2\,\pi} + 5e^{-j3*2\,\pi\,/\,2}$

$= 2 + 3[e^{-j\,\pi} + 4e^{-j2\,\pi} + 5e^{-j3\,\pi}]$

$= 2 + 3\,[\cos \pi/2 - j \sin \pi/2] + 4[\cos \pi - j \sin \pi] + 5[(0)\,3\,\pi/2 - j\sin\,3\,\pi/2]$

= 2 +3 [-1-0] + 4 [1-0] + 5[-1-0]
= 2 -3 + 4 -5

x(2) = -2

x(3) = $2 + 3e^{-j\pi/2} + 4e^{-j3\pi} + 5e^{-j3*3\pi/2}$
= 2 + 3j -4-5j

x(3) = -2-2j

x(k) = [14, -2+2j, -2, -2-2j] % rectangular form

x(k) = [14, 2.82∟135^0, 2 ∟-180^0, 2.82∟-135^0] % polar form

| x(k)| = [14, 2.82, 2, 2.82]

<x(k) = [0, 2.35, 3.143, -2.35] % in radians

MATLAB Code

```
Clear all                          %removes all variables from the workspace
clc                                %clears the command window
close all                          %closes all open figure windows

N=8 ;                              %enter the number of points
x=[1 2 3 4 5 6 7 8];               %enter the sequence is x with time index n
n=0:n-1;                           %compute dft
for k=0:n-1
        w=exp(-j*2*pi*k*n/N);
        dotprod=x.*w;
        x(k+1) = sum(dotprod);
end
ceil(x)

subplot(2,1,1);                    %plot the magnitude spectrum
stem(n,abs(x));
xlabel('n-->');
ylabel('magnitude-->');
grid on;

subplot(2,1,2);                    %plot the magnitude spectrum
stem(n,angle(X));
xlabel('n-->');
ylabel('angle-->');
grid on;

verify=fft(x,N)                    %verification of the implementation
figure                             %plot the magnitude spectrum
subplot(2,1,1);
stem(n,abs(verify));
xlabel('n-->');
ylabel('magnitude-->');
```

```
grid on;
subplot(2,1,2);                    %plot the phase spectrum
stem(n,angle(verify));
xlabel('n-->');
ylabel('angle-->');
grid on;
```

Result : Output will be displayed on the command window and figure window .

PROGRAM 6
Aim:

(i) Verification of DFT properties (Like Linearity and Parsevals Theorem,etc.)

(ii)DFT computation of Square Pulse and Sinc function etc.

Part B: Parsevals Theorem: $\sum_{n=0}^{N-1}\{x(n)^2\} = \frac{1}{N}\sum_{K=0}^{N-1}\{X(K)^2\}$

Part A: Linearity Property:

If x(n)------> X(K) Then DFT{$ax_1(n)+bx_2(n)$} ----------> a×DFT{$x_1(n)$}+ b×DFT{$x_2(n)$}

MATLAB Code

```
Clear all                          %removes all variables from the workspace
clc                                %clears the command window
close all                          %closes all open figure windows

x1=[2  1 2 1];                     %first sequence x1
nx1=0:length(x1)-1;                %time index for first sequence
n1=length(x1);                     %length of first sequence
subplot(3,1,1);                    %plot first sequence
stem(nx1,x1);
xlabel('n-->');
ylabel('x1-->');
title('First Sequence');
grid on;

x2=[1 2 3 4];                      %second sequence x2
nx2=0:length(x2)-1;                %time index for second sequence
n2=length(x2);                     %length of second sequence
subplot(3,1,2);                    %plot second sequence
stem(nx2,x2);
xlabel('n-->');
ylabel('x2-->');
title('Second sequence');
grid on;
```

%Linearity Property

```
N=max(length(x1),length(x2));
newx1=[x1,zeros(1,N-n1)];          %IF N >length((x1) or length(x2)zero pad
```

```
newx2=[x2,zeros(1,N-n2)];
x1dft=fft(newx1);                    %take dft of first sequence
x2dft=fft(newx2);                    %take dft of second sequence
a=6; b=4;
 LHS= a.*x1+b.*x2;
LHSDFT=fft(LHS);
RHS=a.*(fft(x1))+b.*(fft(x2));
LHSDFT
RHS

Disp('Linearity Property is Proved')
```

Result : Output will be displayed on the command window and figure window .

Parsevals Theorem

```
x1=[2  1 2 1];                       %first sequence x1
N=4
Y1=fft(x1)

Left= energy=sum(x1.^2);
Right= (1/N)*(sum(Y1.^2);
Disp('Energy of signal in time domain'); Left
Disp('Energy of signal in time domain'); Right
if(Left==Right)
        disp('Parsevals Property is proved')
end
```

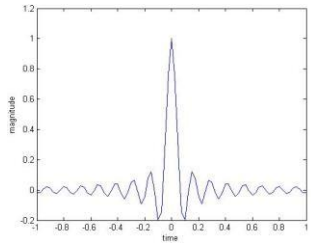
Program 6B
(i) DFT computation of Square Pulse and Sinc function

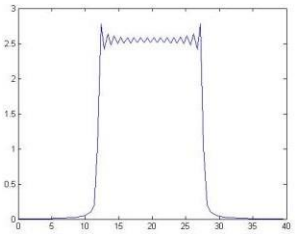
Procedure:

1. Define sampling frequency, number of cycles, plot function.
2. Compute DFT, Plot the function in frequency domain

```
clear all
close all

Fs = 40;
Ts = 1/Fs;
t = -1: Ts : 1;
f = 5;

y = sinc(pi*t*f);
figure, plot(t,y), xlabel('time'), ylabel('magnitude'), title('Sinc Function');

 N = size(t);
```

```
Ydft = fft(y,N);
ff = (0:N-1)*Fs/N;
figure,    plot(ff, fftshift(abs(Ydft)));
%width = pi*f
```

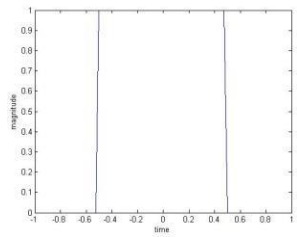


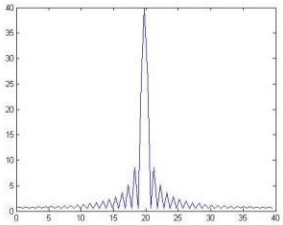


(i) DFT computation of Square function

```
clear  all
close all
Fs = 40;
Ts = 1/Fs;
t = -1:Ts:1;
width = 1;
 x = rectpuls(t, width);
figure, plot(t,x), xlabel('time'), ylabel('magnitude');
[M N] = size(t);

F = fft(x,N);

ff = (0:N-1)*Fs/N;
figure, plot(ff,fftshift(abs(F)));
```

Result

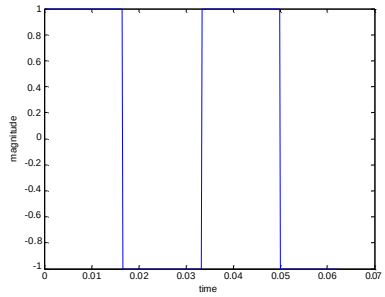


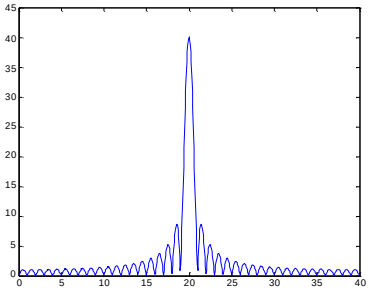


Alternate code:

```
t = 0:.0001:.0625;
y = square(2*pi*30*t);
figure, plot(t,y), xlabel('time'), ylabel('magnitude');
[M N] = size(t);
F = fft(x,N);
ff = (0:N-1)*Fs/N;
figure, plot(ff,fftshift(abs(F)));
```

Result:





PROGRAM 7

Aim: Design and implementation of FIR filter to meet given specifications(using different Window techniques)

Procedure:-

1) Get the sampling frequency
2) Get the pass band frequency
3) Get the stop band frequency or transition width
4) Get the pass band ripple and stop band attenuation
5) Select the window suitable for the stop band attenuation
6) Calculate the order ,N, based on the Transition width
7) Find the N window coefficients
8) Find truncated impulse response of h[n]
9) Verify the frequency response of h[n]

```
Clear all                          %removes all variables from the workspace
Clc                                %clears the command window
close all                          %closes all open figure windows

fp=input('enter the pass band frequency:');
fs=input('enter the stop band frequency:');
rp=input('enter the pass band ripple:');
rs=input('enter the stop band ripple:');
Fs=input('enter the sampling frequency:');

FN=Fs/2;
wp=2*fp/Fs;
ws=2*fs/Fs;

numa=-20*(log10(sqrt(rp*rs))-13);
dena=14.6*(fs-fp)/Fs;

N=ceil(numa/dena);
  N1=N+1;

if(rem(N,2)~=0)
    N1=N;
    N=N-1;
```

```
end;

%Rectangular Window
y=boxcar(N1);
b=fir1(N,wp,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);plot(o/pi,m);xlabel('normalized frequency-->');ylabel('gain in db-);title('LPF');

% Low Pass Filter
b=fir1(N,wp,'low',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);plot(o/pi,m);xlabel('normalized frequency->');ylabel('gain in db-');title(‘HPF');
grid on;

% High Pass Filter
b=fir1(N,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2),plot(o/pi,m);xlabel('normalized frequency->');ylabel('gain in db-');title(‘HPF’);
grid on;

% Band Pass Filter
wn=[wp ws];
b=fir1(N,wn,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);plot(o/pi,m);xlabel('normalized frequency->');ylabel('gain in db-');title(‘BPF’);
grid on;

% Band Stop filter
b=fir1(N,wn,'stop',y);
[h, o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4)
plot(o/pi,m);xlabel('normalized frequency->');ylabel('gain in db->');title('bandstop filter');
```

Input:
Fp=200; fs =500; Sampling frequency=2000;passband ripple=0.001, stop band Ripple=0.005

Result : Output will be displayed on the command window and figure window .

PROGRAM 8

Aim: Design and implementation of IIR filter to meet given specifications.

Procedure:-

1) Get the pass band and stop band edge frequencies

2) Get the pass band and stop band ripples
3) Get the sampling frequency
4) Get the order of the filter
5) Find the filter coefficients
6) Plot the magnitude response

MATLAB Code

```
Clear all                          %removes all variables from the workspace
Clc                                %clears the command window
close all                          %closes all open figure windows

fp = input('enter the pass band frequency:');
fs = input('enter the stop band frequency:');
rp = input('enter the pass band ripple:');
rs = input('enter the stop band ripple:');
Fs=input('enter the sampling frequency:');

fs=Fs/2;

wp =2*fp/Fs;
ws =2*fs/Fs;

[n,wc]=buttord(wp,ws,rp,rs);
[b,a]=butter(n,wc);
disp('Numerator Coefficients'); b
disp('Denominator Coefficients'); a
[y,t]=impz(b,a,60);
Figure;stem(t,y); xlabel(Time index n');ylabel(Amplitude->');title('Impulse Response of
Butter worth Low Pass Filter');

Pole Zero Plot++++++++++++++++++++++++++++++++

w=0:0.01:pi;
[h,om]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(4,2,1);plot(om/pi,m);xlabel('normalized frequency->');ylabel('gain in db-
>');subplot(4,2,2)
plot(om/pi,an)
xlabel('normalized angle--->');
ylabel('gain in db--->');

% High Pass Filter

[b,a]=butter(n,wc,'high');
w=0:0.01:pi;
[h,om]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
```

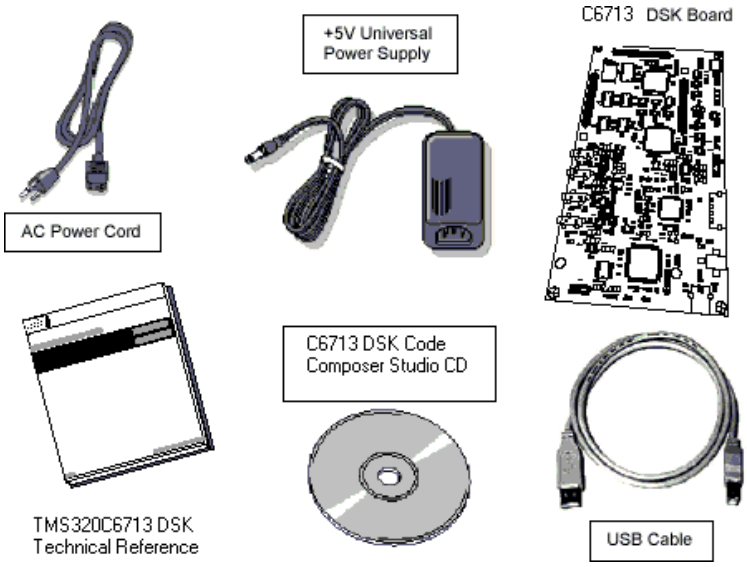
```
subplot(4,2,3)
plot(om/pi,m)
xlabel('normalized frequency--->');
ylabel('gain in db--->');
subplot(4,2,4)
plot(om/pi,an)
xlabel('normalized angle--->');
ylabel('gain in db--->');
% Band Pass filter

wc=[0.4426,0.1];
[b,a]=butter(n,wc,'bandpass');
w=0:0.01:pi;
[h,om]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(4,2,5)
plot(om/pi,m)
xlabel('normalized frequency--->');
ylabel('gain in db--->');
subplot(4,2,6)
plot(om/pi,an)
xlabel('normalized angle--->');
ylabel('gain in db--->');

%Band Stop filter

wc=[0.4426,0.1];

[b,a]=butter(n,wc,'stop');
w=0:0.01:pi;
[h,om]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(4,2,7)
plot(om/pi,m)
xlabel('normalized frequency--->');
ylabel('gain in db--->');
subplot(4,2,8)

plot(om/pi,an)
xlabel('normalized angle--->');
ylabel('gain in db--->');
```

Result : Output will be displayed on the command window and figure window .

Part B: Introduction To Code Composer Studio(CCS)

TMS320C6713 DSK BOARD FEATURES

Package Contents



The C6713™ DSK builds on TI's industry-leading line of low cost, easy-to-use **DSP Starter Kit** (DSK) development boards. The high-performance board features the TMS320C6713 floating-point DSP. Capable of performing 1350 million floating-point operations per second (MFLOPS), the C6713 DSP makes the C6713 DSK the most powerful DSK development board.

The DSK is USB port interfaced platform that allows to efficiently develop and test applications for the C6713. The DSK consists of a C6713-based printed circuit board that will serve as a hardware reference design for TI's customers' products. With extensive host PC and target DSP software support, including bundled TI tools, the DSK provides ease-of-use and capabilities that are attractive to DSP engineers.

The following checklist details items that are shipped with the C6713 DSK.

- TMS320C6713 DSK TMS320C6713 DSK development board
- Other hardware — External 5VDC power supply
  IEEE 1284 compliant male-to-female cable
- CD-ROM — Code Composer Studio DSK tools
- Technical reference manual

The C6713 DSK has a TMS320C6713 DSP onboard that allows full-speed verification of code with Code Composer Studio. The C6713 DSK provides:

- A USB Interface

- ❖ SDRAM and Flash ROM
- ❖ An analog interface circuit for Data conversion (AIC)
- ❖ An I/O port
- ❖ Embedded JTAG emulation support

Connectors on the C6713 DSK provide DSP external memory interface (EMIF) and peripheral signals that enable its functionality to be expanded with custom or third party daughter boards. The DSK provides a C6713 hardware reference design that can assist you in the development of your own C6713-based products. In addition to providing a reference for interfacing the DSP to various types of memories and peripherals, the design also addresses power, clock, JTAG, and parallel peripheral interfaces. The C6713 DSK includes a stereo codec. This analog interface circuit (AIC) has the following characteristics:

High-Performance Stereo *Codec*

- 90-dB SNR Multibit Sigma-Delta ADC (A-weighted at 48 kHz)
- 100-dB SNR Multibit Sigma-Delta DAC (A-weighted at 48 kHz)
- 1.42 V – 3.6 V Core Digital Supply: Compatible With TI C54x DSP Core Voltages
- 2.7 V – 3.6 V Buffer and Analog Supply: Compatible Both TI C54x DSP Buffer Voltages
- 8-kHz – 96-kHz Sampling-Frequency Support

Software Control Via TI McBSP-Compatible Multiprotocol Serial Port

- I 2 C-Compatible and SPI-Compatible Serial-Port Protocols
- Glueless Interface to TI McBSPs

Audio-Data Input/Output Via TI McBSP-Compatible Programmable Audio Interface

- I 2 S-Compatible Interface Requiring Only One McBSP for both ADC and DAC
- Standard I 2 S, MSB, or LSB Justified-Data Transfers
- 16/20/24/32-Bit Word Lengths

The TMS320C6713™ DSP compose the floating-point DSP generation in the TMS320C6000™ DSP platform. The C6713 device is based on the high-performance, advanced very-long-instruction-word (VLIW) architecture developed by Texas Instruments (TI), making this DSP an excellent choice for multichannel and multifunction applications.

The 6713 DSK is a low-cost standalone development platform that enables customers to evaluate and develop applications for the TI C67XX DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market.

Operating at 225 MHz, the C6713 delivers up to 1350 million floating-point operations per second (MFLOPS), 1800 million instructions per second (MIPS), and with dual fixed-/floating-point multipliers up to 450 million multiply-accumulate operations per second (MMACS).The DSK uses the 32-bit EMIF for the SDRAM (CE0) and daughter card expansion interface (CE2 and CE3). The Flash is attached to CE1 of the EMIF in 8-bit mode.

An on-board AIC23 codec allows the DSP to transmit and receive analog signals. McBSP0 is used for the codec control interface and McBSP1 is used for data. Analog audio I/O is done through four 3.5mm audio jacks that correspond to microphone input, line input, line output and headphone output. The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP1 can be re-routed to the expansion connectors in software.

A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD has a register based user interface that lets the user configure the board by reading and writing to the CPLD registers. The registers reside at the midpoint of CE1.

The DSK includes 4 LEDs and 4 DIP switches as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers. An included 5V external power supply is used to power the board. On-board voltage regulators provide the 1.26V DSP core voltage, 3.3V digital and 3.3V analog voltages.

A voltage supervisor monitors the internally generated voltage, and will hold the boards in reset until the supplies are within operating specifications and the reset button is released. If desired, JP1 and JP2 can be used as power test points for the core and I/O power supplies.

Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

TMS320C6713 DSP Features

- Highest-Performance Floating-Point Digital Signal Processor (DSP):
  - Eight 32-Bit Instructions/Cycle
  - 32/64-Bit Data Word
  - 300-, 225-, 200-MHz (GDP), and 225-, 200-, 167-MHz (PYP) Clock Rates
  - 3.3-, 4.4-, 5-, 6-Instruction Cycle Times
  - 2400/1800, 1800/1350, 1600/1200, and 1336/1000 MIPS /MFLOPS
  - Rich Peripheral Set, Optimized for Audio
  - Highly Optimized C/C++ Compiler
  - Extended Temperature Devices Available
- Advanced Very Long Instruction Word (VLIW) TMS320C67x™ DSP Core
  - Eight Independent Functional Units:
    - Two ALUs (Fixed-Point)
    - Four ALUs (Floating- and Fixed-Point)
    - Two Multipliers (Floating- and Fixed-Point)
  - Load-Store Architecture With 32 32-Bit General-Purpose Registers

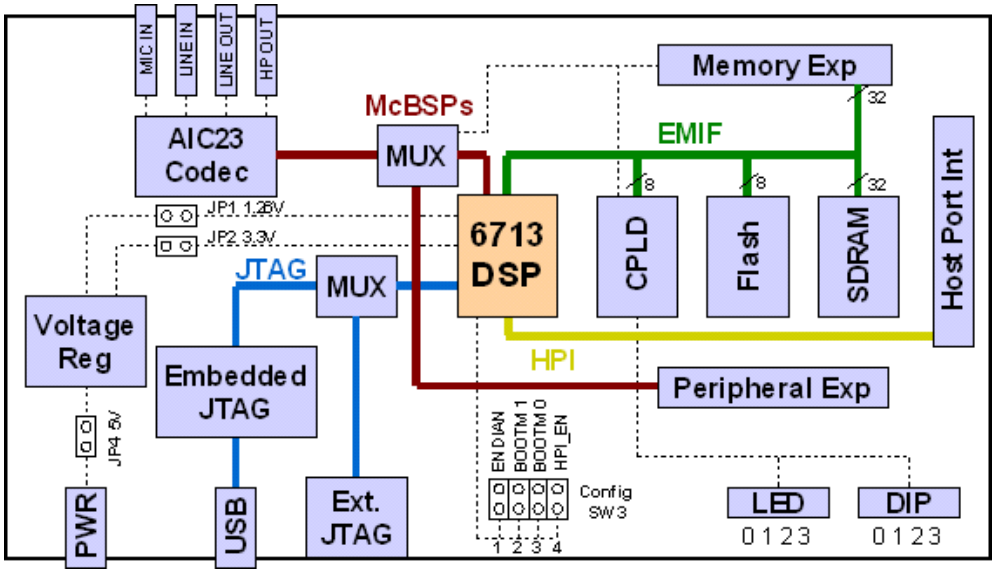
- Instruction Packing Reduces Code Size
- All Instructions Conditional

- Instruction Set Features
  - Native Instructions for IEEE 754
    - Single- and Double-Precision
  - Byte-Addressable (8-, 16-, 32-Bit Data)
  - 8-Bit Overflow Protection
  - Saturation; Bit-Field Extract, Set, Clear; Bit-Counting; Normalization
- L1/L2 Memory Architecture
  - 4K-Byte L1P Program Cache (Direct-Mapped)
  - 4K-Byte L1D Data Cache (2-Way)
  - 256K-Byte L2 Memory Total: 64K-Byte L2 Unified Cache/Mapped RAM, and 192K-Byte Additional L2 Mapped RAM
- Device Configuration
  - Boot Mode: HPI, 8-, 16-, 32-Bit ROM Boot
  - Endianness: Little Endian/Big Endian
- 32-Bit External Memory Interface (EMIF)
  - Glueless Interface to SRAM, EPROM, Flash, SBSRAM, and SDRAM
  - 512M-Byte Total Addressable External Memory Space
- Enhanced Direct-Memory-Access (EDMA) Controller (16 Independent Channels)
- 16-Bit Host-Port Interface (HPI)
- Two Multichannel Buffered Serial Ports (McBSPs)
  - Two Independent Clock Zones Each (1 TX and 1 RX)

Eight Serial Data Pins Per Port: Individually Assignable to any of the Clock Zones

  - Each Clock Zone Includes:
    - Programmable Clock Generator
    - Programmable Frame Sync Generator
    - TDM Streams From 2-32 Time Slots
    - Support for Slot Bits Size8, 12, 16, 20, 24, 28, 32:
    - Data Formatter for Bit Manipulation
- Wide Variety of I2S and Similar Bit Stream Formats
  - Integrated Digital Audio Interface Transmitter (DIT) Supports:
    - S/PDIF, IEC60958-1, AES-3, CP-430 Formats
    - Up to 16 transmit pins
    - Enhanced Channel Status/User Data
  - Extensive Error Checking and Recovery
- Two Inter-Integrated Circuit Bus (I^2C Bus™) Multi-Master and Slave Interfaces
- Two 32-Bit General-Purpose Timers
- Dedicated GPIO Module With 16 pins (External Interrupt Capable)
- Flexible Phase-Locked-Loop (PLL) Based Clock Generator Module
- IEEE-1149.1 (JTAG †) Boundary-Scan-Compatible
- Package Options:
  - 208-Pin Power PAD™ Plastic (Low-Profile) Quad Flat pack (PYP)
  - 272-BGA Packages (GDP and ZDP)
- 0.13-µm/6-Level Copper Metal Process
  - CMOS Technology
- 3.3-V I/Os, 1.2-V Internal (GDP & PYP)

❖ 3.3-V I/Os, 1.4-V Internal (GDP)(300 MHz only)



TMS320C6713 DSK Overview Block Diagram

INTRODUCTION TO CODE COMPOSER STUDIO

Code Composer Studio (CCS) is the first fully industrially development environment (IDE) with DSP specific functionalities. This led us to edit, build, debug profile and manages projects from a single unified environment. Other features are

Graphical signal analysis
Injection and extraction of data signals by a file IO
Multiprocessor debugging
Automatic testing
Customization via 'C' interpretive scripting language.
TMS Processor: This is TI industry leading line of low cost easy to use. DSP start kit development boards with Floating point DSP, Capable of performing 1350 million floating point operations ina second.

CODE COMPOSER FEATURES INCLUDE:

- IDE
- Debug IDE
- Advanced watch windows
- Integrated editor
- File I/O, Probe Points, and graphical algorithm scope probes
- Advanced graphical signal analysis
- Interactive profiling
- Automated testing and customization via scripting
- Visual project management system
- Compile in the background while editing and debugging

- Multi-processor debugging
- Help on the target DSP

Note:-

Launch the DSK help file by opening the following file using Windows Explorer.

C:\CCStudio_v3.1\docs\hlp\c6713dsk.hlp

Documents for Reference:

spru509 → Code Composer Studio getting started guide.
spru189 → TMS320C6000 CPU & Instruction set guide
spru190 → TMS320C6000 Peripherals guide
slws106d →Codec(TLV320AIC23) Data Manual.
spru402 → Programmer's Reference Guide.
sprs186j → TMS320C6713 DSP

Soft Copy of datasheets are available at **: C:\CCStudio_v3.1\docs\pdf.**

Procedure for Simulation of the C Programs with CCS

To Create New Project:

Project--->New Project Name : Proj
Location : C:\Ccs
Project Type : Executable
Target : Tms32067xx
& Then Click Finish

To Create New Source File:
File -----> New--- >Source File
And Then Type C Program And Save The File In Your Project With '.C' Extension

To Add Source File To The Project:
Project----> Add Files To The Project -->Open Your Respective File & Then Click Open

To Add Command & Linkage Files:
Project----> Add Files To The Project -->C:\Ccstudio-V3.1\Tutorials\Dsk6713\Hello1\Select File Type As Linker Command File

To Add Library Files:
Project----> Add Files To The Project -->C:\Ccstudio-V3.1\C6000\Cgtools\Lib\Rts6700.Lib

To Compile:

Project --->Compile File

To Build A Link:

Project---- > Build

Procedure To Load:

File--->Load Programs--->Project Name.Out From Debug Folder

To Execute The Project:

Debug ----> Run

PROGRAM 1

```
/* Program to implement linear convolution */
#include<stdio.h>

main()
{       int m=6;                                        /*Length of i/p samples sequence*/
        int n=6;                                /*Length of impulse response Co-efficients */
        inti=0,j;
        int x[15]={1,2,3,4,5,6,0,0,0,0,0,0};            /*Input Signal Samples*/
        int h[15]={1,2,3,4,5,6,0,0,0,0,0,0};            /*Impulse Response Co-efficients*/
        int y[20];

        for(i=0;i<m+n-1;i++)
        {
                y[i]=0;
                for(j=0;j<=i;j++)
                y[i]+=x[j]*h[i-j];
        }

        for(i=0;i<m+n-1;i++)
        printf("%d\n",y[i]);

}
```

PROGRAM 2 /* Program to implement circular convolution */

```
#include<stdio.h>
void main()
{
  int m,n, x[30],h[30],y[30],i,j, temp[30],k,x2[30],a[30];
  printf(" enter the length of the first sequence\n");
  scanf("%d", &m);
  printf(" enter the length of the second sequence\n");
  scanf("%d", &n);
  printf(" enter the first sequence\n");
  for(i=0;<m;i++)
    scanf("%d",&x[i]);
    printf(" enter the second sequence\n");
 for(j=0;j<n;j++)
    scanf("%d",&h[j]);
        if(m-n!=0)
```

```
        {
                if(m>n)
            {
                for(i=n;i<m;i++)
                h[i]=0;
                n=m;
             }
          for(i=m;i<n;i++)
          x[i]=0;
          m=n;
        }
        y[0]=0;
        a[0]=h[0];
        for(j=1;j<n;j++)
            a[j]=h[n-j];
            for(i=0;i<n;i++)
                y[0]+=x[i]*a[i];
          /* convolution*/
        for(k=1;k<n;k++)
         {
            y[k]=0;
            /*circular shift*/
                for(j=1;j<n;j++)
                        x2[j]=a[j-1];
                        x2[0]=a[n-1];
                for(i=0;i<n;i++)
              {
                  a[i]=x2[i];
                  y[k]+=x[i]*x2[i];
               }
           }
      /*displaying the result*/
     printf(" the circular convolution is\n");
       for(i=0;i<n;i++)
        printf("%d \t",y[i]);
}
```

PROGRAM 3 % N- DFT USING CCS

```
#include<stdio.h>
#include<math.h>
short x[8];
void dft(short *x, short k, int *out);
#define N 8
float pi=3.1416;
int sumre, sumim;
short x[N]={1,2,3,4,5,6,7,8};
int out[2]={0,0};
int real[8],imag[8],k=0;

void dft(short *x, short k, int *out)
```

```
   {
      int sumre=0,sumim=0,i=0;
      float cs=0,sn=0;
      for(i=0;i<N;i++)
      {
         cs=cos(2*pi*k*i/N);
         sn=sin(2*pi*k*i/N);
         sumre = sumre +x[i]*cs;
         sumim = sumim -x[i]*sn;
        }
     out[0]=sumre;
      out[1]=sumim;
      real[k]=sumre;
      imag[k]=sumim;
      k++;

       if(k>N)
       k=0;
        printf("\n%d",out[0]);
     }

void main()
   {
         int j;
         for(j=0;j<N;j++)
         {
         dft(x,j,out);
         }
     }
```

PROGRAM 4: IMPULSE RESPONSE USING Code Composer Studio

```
#include<stdio.h>
#define order 2
#define len 10
float y[len]={0,0,0},sum;
void main()
{

       int k;
       float a[order+1]={0.1311,0.2622,0.1311};
       float b[order+1]={1,-0.7478,0.2722};
          for(i=0;i<len;i++)
          {
            sum=0;
             for(k=1;k<=order;k++)
             {
```

```
                if((i-k)>0)
                sum=sum+(b[k]*y[i-k]);
            }
            if(i<=order)
              {
                y[i]=a[i]-sum;
              }
           else y[i]=-sum;
                printf("response[%d]=%f\n",i,y[i]);
        }
}
```

PROCEDURE:

STEPS FOR INTERFACING:

1. Connect the kit to USB(metal pin). power(black pin) supply,line in, line out.
2. Go to setup icon select family 67xx, platform DSK and all, save and quit, don't open CCS studio
3. Run diagnostics--->start --->if everything is correct you will get "pass".
4. Open CCS studio ,go to debug---->connect -- >check for green color.
5. Projet---> new--->type project name-->select TMS320067XX,executable file .out----> finish.
6. File --->new--->DSP/BIOS configuration--->select DSK67XX---->select DSK6713.cdb---->ok.
7. file---->save as---->file name.cdb in your project folder.
8. Project--->add files to the project---->select files of type .cdb with icon DSK6713.c---> double click on.
9. To create new source file: file--->new--->source file--->type FIR program--- >save as filename.c.
10. Right click on source--->add files to the project -->select filename.c and open.
11. Open file with path CCSstudio v 3.1--->c6000--->DSK6713--->include--->file type *.*---> we get a set of files. In that select DSK6713_aic 23,DSK6713 files-->copy both. Gotoyour project folder, paste these two files and close it.
12. Right click on libraries--->add files to the project-->ccstudio v 3.1-->c6000--> DSK6713-->lib--->DSK6713bsl in *.o or *.l-- >select and open.
13. Go to generated files folder-- >open .c file copy the first line

```
#include<"configuration.h">
```

Go to source file paste it as first line of your c program remove the line

```
#include"XYZCGG.h".
```

14. Save the source program.
15. Compile,build and check for zero errors.
16. Switch on CRO and signal generator set the signal generator output up to 2V and less than 10KHz (around 5KHz) connect right channel, left channel and neutral properly.
17. Run the program.

PROGRAM 18

%FIR FILTER RESPONSE USING DSK KIT(INTERFACING PROGRAM)

```
#include"stdio.h"

#include"dsk6713.h"
#include"dsk6713_aic23.h"

float filter_coeff[]={0.000000,-0.001591,-0.002423,0.000000,0.005728,0.011139,0.010502,
-0.000000,-0.018003,-0.033416,-
0.031505,0.000000,0.063010,0.144802,0.220534,0.26448,0.220534,
0.144802,0.063010,0.000000,-0.031505,-0.033416,-0.018003,-
0.000000,0.010502,0.011139,0.005728,
0.000000,-0.002423,-0.001591,0.000000};

static shortin_buffer[100];
DSK6713_AIC23_Config
config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};

void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Unit32l_input,r_input,l_output,r_output;
DSK6713_init();
hCodec=DSK6713_AIC23_openCodec(0,&config);
DSK6713_AIC23_setFreq(hCodec,1);
while(1)
{
while(!DSK6713_AIC23_read(hCodec,&l_input));
while(!DSK6713_AIC23_read(hCodec,&r_input));
l_output=(int16)FIR_FILTER(&filter_coeff,l_input);
r_output=l_output;

while(!DSK6713_AIC23_write(hCodec,&l_output));
while(!DSK6713_AIC23_write(hCodec,&r_output));
}
DSK6713_AIC23_closeCodec(hCodec);
}
signedint FIR_FILTER(float *h,signed int x)
{
inti=0;
signed long output=0;
in_buffer[0]=x;
for(i=51;i>0;i--)
in_buffer[i]=in_buffer[i-1];
for(i=0;i<51;i++)
output=output+h[i]*in_buffer[i];
return(output);
}
```

VTU Question Bank

1. Write a Mat lab program to sample a band limited continuous signal band limited to fm= --------Hz under the following conditions (i)Nyquist rate (ii)TWICE THE NYQUIST RATE and to find the effect in each of the above case.
2. Obtain Impulse and step response of the given X(n),Y(n) using Matlab.The difference equation is ---------- --.
3. Write and verify the Mat lab code to find the linear convolution of the two discrete sequences. X(n)= ---------- Y(n)= ----------
4. Write and verify the Mat lab code to find the circular convolution of the two discrete sequences. X(n)= ---------- Y(n)= ----------
5. Using Mat lab find the autocorrelation of the given discrete sequence and verify its properities. X(n)= ---------- Y(n)= ----------
6. Using Mat lab find the cross correlation of the given discrete sequence and verify its properities. X(n)= ---------- Y(n)= ----------
7. Given a difference equation, calcu;late h(n) and obtain the response to input sequence x(n) --------using Mat lab.(Without intial conditions)
8. Given a difference equation, calcu;late h(n) and obtain the response to input sequence x(n) --------using Mat lab.(With intial conditions)
9. Using Mat lab compute N-point DFT of a given sequence ---------- and plot magnitude and phase spectrum.
10. Using Mat lab find the Linear convolution of the two finite length sequences using DFT and IDFT.
11. Using Mat lab find the Circular convolution of the two finite length sequences using DFT and IDFT.
12. Design and implement Analog FIR filter to meet the given specification Filter Type ----------
    Pass Band Edge frequency ---------- Stop Band Edge frequency----------.
    PassBand ripple ---------- StopBand ripple -------- Sampling frequency ----------.
13. Design and implement Analog IIR filter to meet the given specification using Mat lab. Pass Band Edge frequency ---------- Stop Band Edge frequency----------.PassBand ripple ---------- StopBand ripple -------- Sampling frequency----------.
14. Realize a Digital IIR filter with the given specification using bilinear transformation.Use Chebyshev –I prototype design Pass Band Edge frequency ---------- Stop Band Edge frequency ----------.PassBand ripple ---------- StopBand ripple ---------- Sampling frequency----------.
15. Using CCS find the Linear convolution of the two discrete sequences. X(n)= ------- Y(n)= -------
16. Using CCS find the Circular convolution of the two discrete sequences. X(n)= ------- Y(n)= -------
17. Using CCS Computation of N-point DFT of a given sequence. Using CCS find the impulse response of the first order and second order system
18. Realize a FIR filter (ant type) to meet the following specifications. The input can be a signal generator/speech emulate the FIR filter using DSK 6713 kit.

DSP Lab viva Questions

1. What are the commonly used computer architecture explain?
2. What are the advantages of DSP?
3. What is the difference between fixed point and floating point devices?
4. What do you mean by time domain and frequency domain ?
5. What do you understand by Real signals, Bandwidth, Spectrum ?
6. What is Sampling?
7. What is the effect of Sampling at 1)Very high frequency,
   2)Sampling at the Nyquist rate
   3)Sampling below the Nyquist rate ?
8. What do you mean by Quantizing?
9. What do you mean by Quantization error?
10. What is the way by which Quantization error can be reduced?
11. What do you mean by filtering?
12. What are the Different types of filters?
13. Explain about i) High pass
    ii) Low pass
    iii) Bandpass and Bandstop filters.
14. What do you understand by Phase spectrum and Frequency spectrum?
15. What is Impulse?
16. What is impulse response of the filter?
17. What do you understand by FIR filter?
18. What do you understand by IIR filter?
19. What are the advantages of digital filter?
20. What do you mean by decimation?
21. Why do we need to use ADC & DAC with DSP's?
22. Linear phase response means?
23. Two sine waves have the amplitude 'A' and frequency'f'.They are out of phase by 180. If these two signals are added, the resultant waveform will be ?
24. What do you mean by aliasing?
25. What type of filter should be used in front of the ADC?
26. On what Factors does the anti aliasing filter depend on?
27. What do you mean by sampling theorem?
28. What is the MATLAB?What arethe applications of MATLAB?
29. State sampling theorem?
30. What is meant by Nyquist rate and nyquist criteria?
31. Explain scaling and superposition properities of a system.
32. What is meant by linearity of a system and how it is related to scaling and superposition?
33. What is impulse function ?
34. What is meant by impulse response?
35. What is energy signal?How to calculateenergy of signal?
36. What is power signal? How to calculatepower of signal?
37. Differentiate between even and odd signals/

38. Explain time invariance property of a system with an example.
39. What is memory less system?
40. When a system is said to have memory?
41. What is meant by causality?
42. Explain linear convolution and circular convolution.
43. What is the length of linear and circular convolutions if the two sequences are having the length n1 and n2?
44. What are the Fourier series and Fourier transform.
45. What are the advantages and special applications of Fourier series, Fourier transform, Z transform and Laplace transform?
46. Differentiate between DTFT and DFT. Why it is advantageous to use DFT in computers rather than DTFT?
47. How to perform linear convolution using circular convolution?
48. What is meant by correlation?
49. What is auto correlation?
50. What is cross correlation?
51. What are the advantages of using autocorrelation and cross correlation properities in signal processing fields?
52. How autocorrelation can be used to detect the presence of noise?
53. Differentiate between IIR filters and FIR filters?
54. What is the procedure to design a digital Butterworth filter?
55. What are the difference equations and differential equations?
56. What is non real time processing?
57. What is meant by real time processing?
58. What is Digital signal processor(DSP)?
59. Differentiate between RISC and CISC architectures.
60. Differentiate between General purpose MPU(Micro Processor Unit) and DSP Processor.
61. What is pipelining?
62. What is parallel processing?
63. What is MAC?
64. What is Barrel shifter? Why it is advantageous to use it in DSP processor?
65. Differentiate between floating point DSP and fixed point DSP.
66. Explain Fixed point/Floating point?
67. What is code composer studio?
68. Explain Von-Neumann and Harvard architectures.
69. What are Line-in,line-out,Mic-in,Mic-out?

VIVA QUESTION AND ANSWERS FOR DSP LAB[15ECL 57]

1. Features of TMS320C6713 Starter kit :-

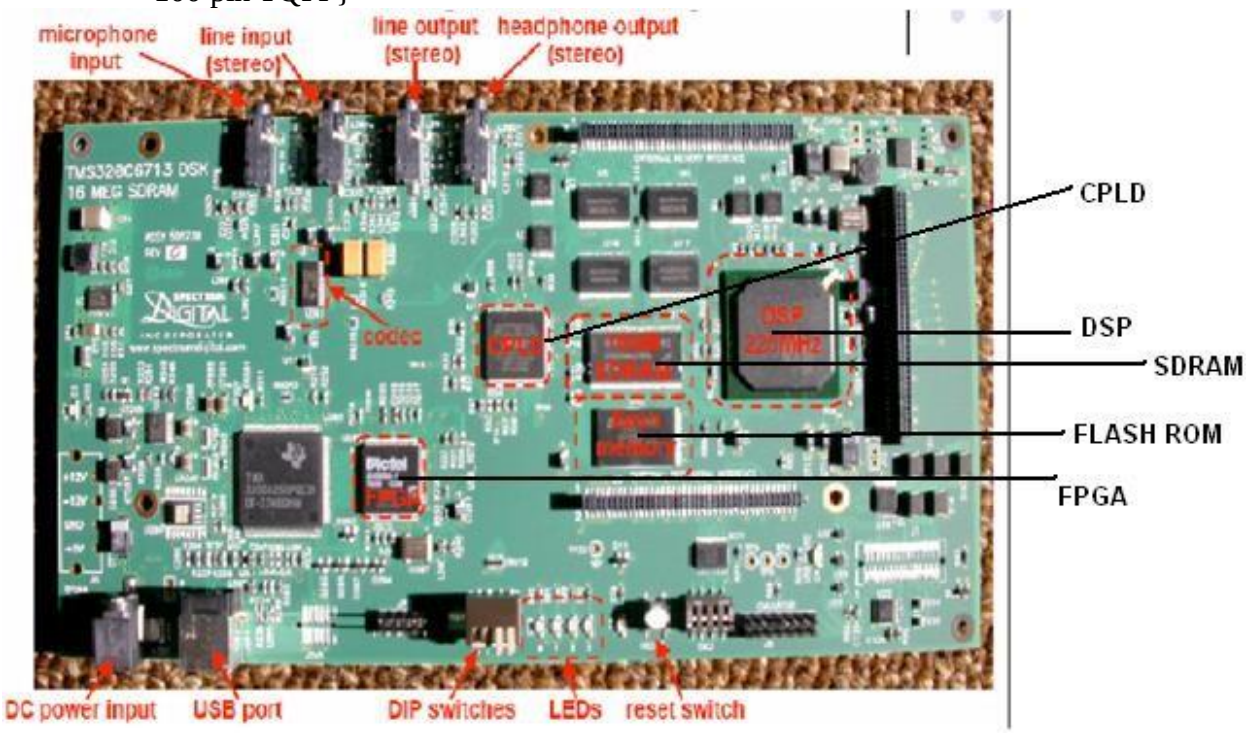
- TMS320C6713 DSP operating at 225MHz device delivering up to 1800 million instructions per second (MIPs) and 1350 MFLOPS.
- Embedded USB JTAG controller with plug and play drivers, USB cable included
- TI TLV320AIC23 codec → This IC is used to transmit and receive analog signals.

- 16MB SDRAM **I.C No. 48LC4M32B2 → 4MEGX32(1MEGX32X4banks)**
- 512K bytes of on board Flash ROM **I.C No. M29W400T**
- Expansion connectors (Memory Interface, Peripheral Interface, and Host Port Interface)
- On board IEEE 1149.1 JTAG connection for optional emulator debug
- +5V universal power supply
- High-quality 24-bit stereo codec
- user definable LEDs
- position dip switch, user definable
- Size: 8.25" x 4.5" (210 x 115 mm), 0.062" thick, 6 layers
- Compatible with Spectrum Digital's DSK Wire Wrap Prototype Card
- RoHS Compliant
- CPLD **I.C.No. MAX EPM3128A TC100-10N**

{MAX3000A, 128 macro cells, 100TQFP, 8logic array blocks, 96 I/O pins and 10ns.}

- FPGA **I.C.No. ACTEL A54SX08A-TQG100**

{FPGA SX-A Family, 8K gates, 512 cells, 238MHz,0.25um/0.22um(CMOS) tech.,2.5v and 100 pin TQFP}



2. Expansion of TMS320C6713 DSK ?

T	→	TEXAS INSTRUMENTS
MS	→	MIXED SIGNALS
32	→	It's a 32 bit processor
0	→	Floating point

C → CMOS Technology
6713 → The series of the processor kit.
DSK → Digital Starter Kit.

3. Difference between DSP processor and microprocessor ?

Functions of DSP:-

DSPs are designed specifically to perform large numbers of complex arithmetic calculations as quickly as possible, usually in applications such as image processing, speech recognition and telecommunications. DSPs are more efficient than general-purpose microprocessors at performing basic arithmetic operations, especially multiplication. They are found in devices that require rapid processing of audio or video data in real time, such as cell phones, DVD players and digital cameras.

Functions of microprocessor:-

In the computing world, faster is always better. However, microprocessors are general-purpose devices. They are designed to run software applications such as word processors, spreadsheets and web browsers. These are applications where speed is important but not critical. Microprocessors are at the core of desktops, laptops, netbooks and tablet PCs.

4. Difference between fixed point and floating point ?

Digital signal processing can be separated into two categories - fixed point and floating point. These designations refer to the format used to store and manipulate numeric representations of data. Fixed-point DSPs are designed to represent and manipulate integers – positive and negative whole numbers – via a minimum of 16 bits, yielding up to 65,536 possible bit patterns (2^{16}). Floating-point DSPs represent and manipulate rational numbers via a minimum of 32 bits in a manner similar to scientific notation, where a number is represented with a mantissa and an exponent (e.g., $A \times 2^B$, where 'A' is the mantissa and 'B' is the exponent), yielding up to 4,294,967,296 possible bit patterns (2^{32}).

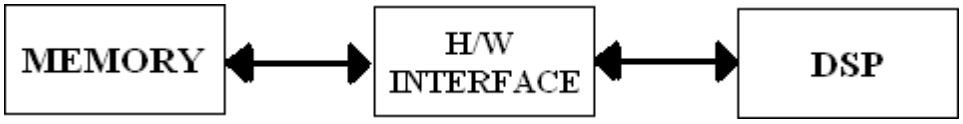
The term 'fixed point' refers to the corresponding manner in which numbers are represented, with a fixed number of digits after, and sometimes before, the decimal point. With floating-point representation, the placement of the decimal point can 'float' relative to the significant digits of the number. For example, a fixed-point representation with a uniform decimal point placement convention can represent the numbers 123.45, 1234.56, 12345.67, etc, whereas a floating-point representation could in addition represent 1.234567, 123456.7, 0.00001234567, 1234567000000000, etc. As such, floating point can support a much wider range of values than fixed point, with the ability to represent very small numbers and very large numbers.

5. What is the use of FPFA and CPLD in TMS320C6713 PROCESSOR?

DSPs often have to interface with external memory, typically shared with host processors or with other DSPs. The two main mechanisms available to implement the memory interfacing are to use hardware interfaces already existing on the DSP chip or to provide external hardware that carries out the memory interfacing. These two methods are briefly mentioned below.

Hardware interfaces are often available on TI as well as on ADI DSPs. An example is TI External Memory Interface (EMIF) [38], which is a glue less interface to memories such as SRAM,EPROM, Flash, Synchronous Burst SRAM (SBSRAM) and Synchronous DRAM (SDRAM). On theTMS320C6713 DSP, for instance, the EMIF provides 512 Mbytes of addressable external memory space. Additionally, the EMIF supports memory width of 8 bits, 12 bits and 32 bits, including read/write of both big- and little-endian devices.

When no dedicated on-chip hardware is available, the most common solution for interfacing a DSP to an external memory is to add external hardware between memory and DSP, as shown in below block diagram. Typically this is done by using a CPLD or an FPGA which implements address decoding and access arbitration. Care must be taken when programming the access priority and/or interleaved memory access in the CPLD/FPGA. This is essential to preserve the data integrity. Synchronous mechanisms should be preferred over asynchronous ones to carry out the data interfacing.



6. Which software is used to run TMS320C6713 processor and note about its features ?

The software used is Code Composer Studio V3.1

CCS provides an IDE to incorporate the software tools. CCS includes tools for code generation, such as a C compiler, an assembler, and a linker. It has graphical capabilities and supports real-time debugging. It provides an easy-to-use software tool to build and debug programs.

The C compiler compiles a C source program with extension .c to produce an assembly source file with extension.asm. The assembler assembles an.asm source file to produce a machine language object file with extension.obj. The linker combines object files and object libraries as input to produce an executable file with extension .out.

7. What is compiler ?

A program that translates a high level language into machine level language program is called compiler.

8. What is an assembler ?

Assembler is a software that translates assembly language program to a machine language program.

9. What is linker ?

Linker is a software that joins together several object files and library functions into one large executable file.

10. What is the function of linker command file in TMS320C6713?

The function of linker command file is that it maps sections to memory.

11. What is debugger?

Debugger is a program that allows the execution of a program in single step mode under control of user.

12. What does the building process does in TMS320C6713?

The building process causes all the dependent files to be included.

13. What is convolution and mention it's properties?

Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in digital signal processing. Using the strategy of impulse decomposition, systems are described by a signal called the *impulse response*. Convolution is important because it relates the three signals of interests : the input signals, the output signals and the impulse response.

The three basic properties of convolution as an algebraic operation are that it is commutative, associative, and distributive over addition. The commutative property means simply that x convolved with *h* is identical with *h* convolved with x. The consequence of this property for LTI systems is that for a system with a specified input and impulse response, the output will be the same if the roles of the input and impulse response are interchanged. The associative property specifies that while convolution is an operation combining two signals, we can refer unambiguously to the convolution of three signals without concern about how they are grouped pair wise.

The associative property combined with the commutative property leads to an extremely important property of LTI systems. Specifically, if we have several LTI systems cascaded together, the output generated **by** an input to the overall cascade combination does not depend on the order in which the systems are cascaded. This property of LTI systems plays an extremely important role in system design, implementation, and analysis. It is generally not true for arbitrary systems that are not linear and time-invariant, and it represents one very important consequence of exploiting the properties of linearity and time invariance.

The distributive property states that a signal convolved with the sum of two signals is identical to the result of carrying out the convolution with each signal in the sum individually and then summing the result.

14. What is the function of 'clc' in matlab /octave tool ?

CLC clears the command window and homes the cursor.

15. What is the function of 'clear all' in matlab /octave tool?

CLEAR removes all variables from the workspace.

16. What is the function of 'close all' in matlab /octave tool?

CLOSE ALL closes all the open figure windows.

17. What is the function of 'figure' in matlab /octave tool?

FIGURE, by itself, creates a new figure window, and returns its handle.

18. What is the function of 'subplot' in matlab /octave tool?

Subplot divides the figure window into rows, columns and position

For example subplot(3,1,1) means the figure window divides into 3 rows, 1column and 1st position.

19. What is the function of 'plot' in matlab /octave tool?

PLOT Linear plot.

PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix, then the vector is plotted versus the rows or columns of the matrix, which ever line up. If X is a scalar and Y is a vector, length(Y) disconnected points are plotted.

20. What is the function of 'stem' in matlab /octave tool?

STEM Discrete sequence or "stem" plot.

STEM(Y) plots the data sequence Y as stems from the x axis terminated with circles for the data
value.
STEM(X,Y) plots the data sequence Y at the values specified in X.

21. What is the function of 'xlabel' in matlab /octave tool?

XLABEL X-axis label. XLABEL('text') adds text beside the X-axis on the current axis.

22. What is the function of 'ylabel' in matlab /octave tool?

YLABEL Y-axis label. YLABEL('text') adds text beside the Y-axis on the current axis.

23. What is the function of 'title' in matlab /octave tool?

TITLE Graph title. TITLE('text') adds text at the top of the current axis.

24. What is the function of 'conv' in matlab /octave tool?

CONV Convolution and polynomial multiplication.
C = CONV(A, B) convolves vectors A and B. The resulting vector is LENGTH(A)+LENGTH(B)-1. If A and B are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials.

25. What is the function of 'FFT' in matlab /octave tool ?

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.
FFT(X,N) is the N-point FFT, padded with zeros if X has less than N points and truncated if it
has more.

26. What is the function of 'IFFT' in matlab /octave tool?

IFFT Inverse discrete Fourier transform.
IFFT(X) is the inverse discrete Fourier transform of X.
IFFT(X,N) is the N-point inverse transform.

27. What is the function of 'xcorr' in matlab /octave tool?

XCORR Cross-correlation function estimates.
C = XCORR(A,B), where A and B are length M vectors (M>1), returns the length 2*M-1 cross-
correlation sequence C. If A and B are of different length, the shortest one is zero-padded. C
will be a row vector if A is a row vector, and a column vector if A is a column vector.
XCORR(A), when A is a vector, is the auto-correlation sequence.

28. What is the function of 'conj' in matlab /octave tool?

CONJ Complex conjugate. CONJ(X) is the complex conjugate of X.
For a complex X, CONJ(X) = REAL(X) - i*IMAG(X).

29. What is the function of 'ceil' in matlab /octave tool ?

CEIL Round towards plus infinity. CEIL(X) rounds the elements of X to the nearest integers towards infinity. For example: ceil([-2.7, 2.7]) => -2 3

30. What is the function of 'filter' in matlab /octave tool?

FILTER One-dimensional digital filter.

Y = FILTER(B,A,X) filters the data in vector X with the filter described by vectors A and B to create the filtered data Y. The filter is a "Direct Form II Transposed" implementation of the standard difference equation:

a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
- a(2)*y(n-1) - ... - a(na+1)*y(n-na)

If a(1) is not equal to 1, FILTER normalizes the filter coefficients by a(1).

31. What is the function of 'filtic' in matlab /octave tool?

FILTIC Make initial conditions for 'filter' function.

Z = filtic(B, A, Y, X) converts past input X and output Y into initial conditions for the state variables Z needed in the TRANSPOSED DIRECT FORM II filter structure.

Z = filtic(B, A, Y) assumes that X = 0 in the past.

32. What is the function of 'fliplr' in matlab /octave tool?

FLIPLR Flip matrix in left/right direction.

FLIPLR(X) returns X with row preserved and columns flipped in the left/right direction.

Example:

X =[1 2 3 becomes [3 2 1
4 5 6] 6 5 4]

33. What is the function of 'butter' in matlab /octave tool?

BUTTER Butterworth digital and analog filter design.

[B,A] = BUTTER(N,Wn) designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

34. What is the function of 'buttord' in matlab /octave tool?

BUTTORD Butterworth filter order selection.

[N, Wn] = BUTTORD(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Butterworth filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband.

Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample). For example,

Lowpass: Wp = .1, Ws = .2
Highpass: Wp = .2, Ws = .1
Bandpass: Wp = [.2 .7], Ws = [.1 .8]

Bandstop: Wp = [.1 .8], Ws = [.2 .7]

35. What is the function of 'cheby1' in matlab /octave tool?

CHEBY1 Chebyshev Type I digital and analog filter design.

[B,A] = CHEBY1(N,R,Wn) designs an Nth order lowpass digital Chebyshev filter with R decibels of peak-to-peak ripple in the passband. CHEBY1 returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate. Use R=0.5 as a starting point, if you are unsure about choosing R.

36. What is the function of 'cheb1ord' in matlab /octave tool?

CHEB1ORD Chebyshev Type I filter order selection.

[N, Wn] = CHEB1ORD(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Chebyshev Type I filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample). For example,

Lowpass: Wp = .1, Ws = .2
Highpass: Wp = .2, Ws = .1
Bandpass: Wp = [.2 .7], Ws = [.1 .8]
Bandstop: Wp = [.1 .8], Ws = [.2 .7]

37. What is meant by DFT ?

The discrete Fourier transform (DFT) is the primary transform used for numerical computation

in digital signal processing. It is very widely used for spectrum analysis, fast convolution, and

many other applications. The DFT transforms N discrete-time samples to the same number of

discrete frequency samples, and is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi nk/N}$$

38. What are the applications of DFT?

1. Spectral analysis

When the DFT is used for spectral analysis, the $\{x_n\}$ sequence usually represents a finite set of uniformly spaced time-samples of some signal $x(t)$, where *t* represents time. The conversion from continuous time to samples (discrete-time) changes the underlying fourier transform of x(t) into a discrete-time Fourier transform (DTFT), which generally entails a type of distortion called aliasing.

2. Data compression

The field of digital signal processing relies heavily on operations in the frequency domain (i.e. on the Fourier transform). For example, several lossy image and sound compression methods employ the discrete Fourier transform: the signal is cut into short segments, each is transformed, and then the Fourier coefficients of high frequencies, which are assumed to be unnoticeable, are discarded. The decompressor computes the inverse transform based on this reduced number of Fourier coefficients. (Compression applications often use a

specialized form of the DFT, the discrete cosine transform or sometimes the modified discrete cosine transform.)

3. Partial differential equations

Discrete Fourier transforms are often used to solve partial differential equations, where again the DFT is used as an approximation for the Fourier series (which is recovered in the limit of infinite *N*). The advantage of this approach is that it expands the signal in complex exponentials e^{inx}, which are Eigen functions of differentiation: $d/dx\ e^{inx} = in\ e^{inx}$.

39. What is meant by FFT ?

Fast Fourier Transform is an algorithm used to compute DFTs.

40. What is meant by IDFT ?

The inverse DFT (IDFT) transforms N discrete-frequency samples to the same number of discrete- time samples. The IDFT has a form very similar to the DFT,

$$x(n) = 1/N \sum_{k=0}^{N-1} X(k)\ e^{j2\pi nk/N}$$

41. What is meant by IFFT ?

The IFFT block computes the inverse fast Fourier transform (IFFT) of each row of a sample-based 1-by-*P* input vector, or across the first dimension (*P*) of an *N-D* input array.

42. Difference between analog filter and digital filters ?

Digital Filters:

1. It operates on the digital samples of the signals.
2. These kinds of filters are defined using linear difference equations.
3. While implementing the digital filters in hardware or software (for simulation), we need adders, subtractors, delays, etc which are classified under digital logic components.
4. In this filter, the filter coefficients are designed to meet the desired or expected frequency response.
5. Mathematically the transfer function ***H(z)*** is required to be a rational function of ***z***,where the coefficients of z are real to meet the stability and causality requirement.
6. In order to be stable and causal, the poles of the transfer function should lie inside the unit circle in z-plane.

Analog Filters:

1. Unlike digital, analog filters works on analog signals or the so called actual signals.
2. It is defined by linear differential equations.
3. While implementing the analog filters in hardware or software simulation, electrical components like resistors, capacitors and inductors are used.
4. To achieve the required frequency response, approximation problem is solved.
5. To be stable and causal, the transfer function ***H(s)*** must be a rational function of ***s***, whose coefficients are real.

6. For stability and causality, the poles should lie on the left half of s-plane.

43. How do you convert analog filter prototype to a digital filter?

There are five methods to convert analog filter prototype to a digital filter, they are:

1. Impulse-invariant method
2. Frequency mapping method
3. Bilinear transformation method
4. Matched Z transform
5. Backward difference method

44. What are the steps to be taken while designing a digital filter ?

When designing a digital filter using an analog approximation and the bilinear transform, we follow these steps:

b. Prewarp the cutoff frequencies
c. Design the necessary analog filter
d. apply the bilinear transform to the transfer function
e. Normalize the resultant transfer function to be monotonic and have a unity passband gain (0dB).

Alternatively, if we have an **inverse bilinear transform**, we can follow these steps:

a. use the inverse bilinear transform on the filter specifications in the digital domain to produce equivalent specifications in the analog domain.
b. Construct the analog filter transfer functions to meet those specifications.
c. use the bilinear transform to convert the resultant analog filter into a digital filter.

45. What is prewarping ?

Frequency warping follows a known pattern, and there is a known relationship between the warped frequency and the known frequency. We can use a technique called **prewarping** to account for the nonlinearity, and produce a more faithful mapping.

$$\omega_p = \frac{2}{T}\tan\left(\frac{\omega}{2}\right)$$

The *p* subscript denotes the prewarped version of the same frequency.

46. What is bilinear transform?

The Bilinear transform is a mathematical relationship which can be used to convert the transfer function of a particular filter in the complex Laplace domain into the z-domain, and vice-versa. The resulting filter will have the same characteristics of the original filter, but can be implemented using different techniques. The Laplace Domain is better suited for designing analog filter components, while the Z-Transform is better suited for designing digital filter components.

The bilinear transform is the result of a numerical integration of the analog transfer function into the digital domain. We can define the bilinear transform as:

$$s = \frac{2(1 - z^{-1})}{T(1 + z^{-1})}$$

47. Differentiate between Butterworth and Chebyshev filter?

Some of the important differences are as follows:

Magnitude response vs frequency curve: The magnitude response |H(jw)| of the

butterworth filter decreases with increase in frequency from 0 to infinity, while the magnitude response of the Chebyshev filter fluctuates or show ripples in the passband and stopband depending on the type of the filter.

Width of Transition band: The width of the transition band is more in Butterworth filter compared to the Chebyshev filter.

Location of the poles: The poles of a Butterworth filter lies only on a circle while that of the Chebyshev filter lies on an ellipse, which can be easily concluded on looking at the poles formula for both types of filters.

No. of Components required for implementing the filter: The number of poles in Butterworth filter is more compared to that of the Chebyshev filter of same specifications, this means that the order of Butterworth filter is more than that of a Chebyshev filter. This fact can be used for practical implementation, since the number of components required to construct a filter of same specification can be reduced significantly.

48. What is correlation?

Correlation is a mathematical operation that is very similar to convolution. Just as with convolution, correlation uses two signals to produce a third signal. This third signal is called the **cross-correlation** of the two input signals. If a signal is correlated with *itself*, the resulting signal is instead called the **autocorrelation**.

49. What are the applications of autocorrelation?

a) One application of autocorrelation is the measurement of optical spectra and the measurement of very-short-duration light pulses produced by lasers, both using optical autocorrelators.

b) Autocorrelation is used to analyze Dynamic light scattering data, which notably enables to determine the particle size distributions of nanometer-sized particles or micelles suspended in a fluid.

c) In signal processing, autocorrelation can give information about repeating events like musical beats (for example, to determine tempo) or pulsar frequencies, though it cannot tell the position in time of the beat.

d) In music recording, autocorrelation is used as a pitch detection algorithm prior to vocal processing, as a distortion effect or to eliminate undesired mistakes and inaccuracies.

50. What are the applications of cross correlation?

The cross-correlation function is used extensively in *pattern recognition* and *signal detection*. We know that projecting one signal onto another is a means of measuring how much of the second signal is present in the first. This can be used to ``detect" the presence of known signals as components of more complicated signals. As a simple example, suppose we record *x(n)* which we think consists of a signal *s(n)* which we are looking for plus some additive measurement noise *e(n)*. Then the projection of x onto is s

$$\mathcal{P}_s(x) = \mathcal{P}_s(s) + \mathcal{P}_s(e) \approx s$$

since the projection of any specific signal s onto random, zero-mean noise is close to zero. Another term for this process is called *matched filtering*. The impulse response of the ``matched filter" for a signal x is given by *Flip(x)*. By time reversing x, we transform the convolution implemented by filtering into a cross-correlation operation.

51. What are the properties of autocorrelation?

a. A fundamental property of the autocorrelation is even symmetry,

b. The continuous autocorrelation function reaches its peak at the origin, where it takes a real value, i.e. for any delay.

c. The autocorrelation of a periodic function is, itself, periodic with the same period.

52. What are the properties of cross correlation?

a. The cross-correlation of functions $f(t)$ and $g(t)$ is equivalent to the convolution of $f^*(-t)$ and $g(t)$.

i.e.: $f \star g = f^*(-t) * g.$

b. Analogous to the convolution theorem, the cross-correlation satisfies:

$$\mathcal{F}\{f \star g\} = (\mathcal{F}\{f\})^* \cdot \mathcal{F}\{g\},$$

53. What is the difference between cross correlation and auto correlation?

Sl No.	Cross correlation	Auto Correlation
01	When two independent signals are compared, the procedure is known as cross-correlation	When the same signal is compared to phase shifted copies of itself, the procedure is known as auto correlation
02	Cross-correlation is the method which basically underlies implementations of the Fourier transformation: signals of varying frequency and phase are correlated with the input signal, and the degree of correlation in terms of frequency and phase represents the frequency and phase spectrums of the input signal.	Autocorrelation is a method which is frequently used for the extraction of fundamental frequency, : if a copy of the signal is shifted in phase, the distance between correlation peaks is taken to be the fundamental period of the signal (directly related to the fundamental frequency)

54. What is impulse response?

In signal processing, the **impulse response**, or **impulse response function (IRF)**, of a dynamic system is its output when presented with a brief input signal, called an impulse. More generally, an impulse response refers to the reaction of any dynamic system in response to some external change. In both cases, the impulse response describes the reaction of the system as a function of time (or possibly as a function of some other independent variable that parameterizes the dynamic behavior of the system).

55. What is sampling?

Sampling is a process of converting a continuous time signal (analog signal) i.e., x(t) into a discrete time signal i.e., x[n] which is represented as a sequence of numbers (Analog to Digital converter).

56. What is quantization?

Quantization is the process of converting a discrete time continuous amplitude signal x(n) into a dicrete-time discrete amplitude signal $z_q(n)$.

57. What is aliasing ?

aliasing is an effect that causes different signals to become indistinguishable (or *aliases* of one another) when sampled. It also refers to the distortion or artifact that results when the signal reconstructed from samples is different from the original continuous signal.

58. State sampling theorem?

The sampling theorem states that a set of samples of a signal can be reconstructed into the original signal if and only if the original system is band limited and the sampling frequency is greater than twice the maximum frequency for non-zero values of the original function.

or

It states that, while taking the samples of a continuous signal, it has to be taken care that the sampling rate is equal to or greater than twice the cut off frequency and the minimum sampling rate is known as the Nyquist rate.

59. What is a system?

A system is defined as a physical device that generates a response or an output signal for a given input signal.

60. What are causal and non-causal system?

A system is said to be causal if the output of the system at any time 'n' depends only at present and past inputs but does not depend on future inputs. If the output of a system depends on future inputs then the system is called non-causal system.

61. What are linear and non-linear system?

A system that satisfies the superposition principle is said to be linear system. Super position principle states that the response of the system to a weighted sum of signals should be equal to the corresponding weighted sum of the output's of the system to each of the individual input signals. A relaxed system that does not satisfy the superposition principle is called non-linear system.

62. What is an FIR system?

If the impulse response of the system is of finite duration then the system is called **F**inite **I**mpulse **R**esponse.

63. What is a *linear phase* filter?

"Linear Phase" refers to the condition where the phase response of the filter is a linear (straight-line) function of frequency (excluding phase wraps at +/- 180 degrees). This results in the *delay* through the filter being the same at all frequencies. Therefore, the filter does not cause "phase distortion" or "delay distortion". The lack of phase/delay distortion can be a critical advantage of FIR filters over IIR and analog filters in certain systems, for example, in digital data modems.

64. What is the condition for linear phase?

FIR filters are usually designed to be linear-phase (but they don't have to be.) A FIR filter is linear-phase if (and only if) its coefficients are symmetrical around the center coefficient, that is, the first coefficient is the same as the last; the second is the same as the next-to-last, etc.

(A linear-phase FIR filter having an odd number of coefficients will have a single coefficient in the center which has no mate.)

65. What is the delay of a linear-phase FIR?

The formula is simple: given a FIR filter which has N taps, the delay is: (N - 1) / (2 * Fs), where Fs is the sampling frequency. So, for example, a 21 tap linear-phase FIR filter operating at a 1 kHz rate has delay: (21 - 1) / (2 * 1 kHz)=10 milliseconds.

66. What is the Z transform of a FIR filter?

For an N-tap FIR filter with coefficients h(k), whose output is described by:
y(n)=h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + ... h(N-1)x(n-N-1),

The filter's Z transform is:
H(z)=h(0)z-0 + h(1)z-1 + h(2)z-2 + ... h(N-1)z-(N-1) , or $H(z) \sum_{n=0}^{N-1} h(n)z^{-n}$

67. Can I calculate the frequency response of a FIR using the Discrete Fourier Transform(DFT)?

Yes. For an N-tap FIR, you can get N evenly-spaced points of the frequency response by doing a DFT on the filter coefficients.

68. How do I scale the gain of FIR filter?

Simply multiply all coefficients by the scale vector.

69. Are FIR filters inherently stable?

Yes. Since they have no feedback elements, any bounded input results in a bounded output.

70. What makes the numerical properties of FIR filters "good"?

The key is the lack of feedback. The numeric errors that occur when implementing FIR filters in computer arithmetic occur separately with each calculation; the FIR doesn't "remember" its past numeric errors. In contrast, the feedback aspect of IIR filters can cause numeric errors to compound with each calculation, as numeric errors are fed back.

71. Why are FIR filters generally preferred over IIR filters in multirate(decimating and interpolating) systems?

Because only a fraction of the calculations that would be required to implement a decimating or interpolating FIR in a literal way actually needs to be done.
Since FIR filters do not use feedback, only those outputs which are actually going to be used have to be calculated. Therefore, in the case of decimating FIRs (in which only 1 of N outputs will be used), the other N-1 outputs don't have to be calculated. Similarly, for interpolating filters (in which zeroes are inserted between the input samples to raise the sampling rate) you don't actually have to multiply the inserted zeroes with their corresponding FIR coefficients and sum the result; you just omit the multiplication-additions that are associated with the zeroes (because they don't change the result anyway.)
In contrast, since IIR filters use feedback, every input must be used, and every input must be calculated because all inputs and outputs contribute to the feedback in the filter.

72. What is an IIR System?

If the impulse response of the system is of infinite duration then the system is called **I**nfinite **I**mpulse **R**esponse.

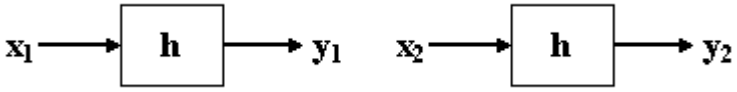
73. Compare FIR and IIR filters?

Sl. No.	IIR	FIR
01	IIR filters are difficult to control and have no particular phase,	FIR filters make a linear phase always possible.
02	IIR can be unstable	FIR is always stable.
03	IIR, when compared to FIR, can have limited cycles,	FIR has no limited cycles.
04	IIR is better for lower-order tapping	The FIR filter is used for higher-order tapping

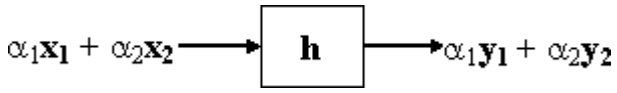
74. Explain the properties of circular convolution?

The properties of circular convolution are:

a) Circular convolution (CC) is ***linear***

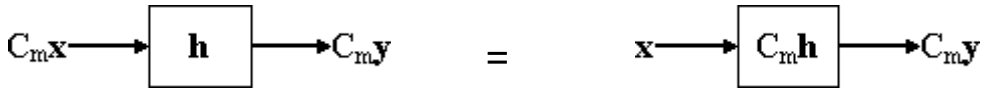


(a) **(b)**



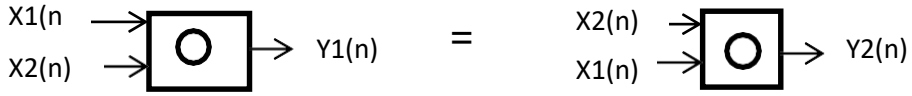
(c)

b) Circular Convolution is ***shift invariant***



(a) **(b)**

c) Circular Convolution is ***commutative***



75. What is the difference between convolution and correlation?

Correlation is a metric for similarity between two different signals. When normalized, so that each of the two signals to be correlated have unitary power and null mean value, the correlation operation shifts to the computation of the correlation coefficient, of the two signals you are comparing. The correlation coefficient, for a given time lag t between the two signals, is always between -1 and +1, clearly giving a measure of the similarity of the shapes of the two signals at that time lag.

Convolution, instead, is the common operation a Linear and Time Invariant system can perform on a given input signal. It is clear that, in specific cases, correlation anc convolution are very similar: the matched filter case makes correlation and convolution identical.

convolution is a technique to find the output of a system of impulse response h(n) for an input x(n) so basically it is used to calculate the output of a system, while correlation is a process to find the degree of similarity between two signals.

76. What is the difference between linear and circular convolution ?

Linear convolution takes two functions of an independent variable, which is called **time**, and convolves them using the convolution sum formula. Basically it is a correlation of one function with the time-reversed version of the other function. I think of it as flip, multiply, and sum while shifting one function with respect to the other. This holds in continuous time, where the convolution sum is an integral, or in discrete time using vectors, where the sum is truly a sum. It also holds for functions defined from -Inf to Inf or for functions with a finite length in time.

Circular convolution is only defined for finite length functions (usually, maybe always, equal in length), continuous or discrete in time. In circular convolution, it is as if the finite length functions repeat in time, periodically. Because the input functions are now periodic, the convolved output is also periodic and so the convolved output is fully specified by one of its periods.

77. How to convert degrees into radians and vice-versa?

To convert degrees into radians:

Example- 200^0 $200^0 * \pi / 180^0 \Rightarrow 10\pi/9 \Rightarrow 3.49$ rad

To convert radians into degrees:

Example- $4\pi/9$ $\Rightarrow \frac{4\pi}{9} * \frac{180^0}{\pi} \Rightarrow \frac{720^0\,\pi}{9\,\pi} \Rightarrow 80^0$

78. What are imaginary numbers ?

An imaginary number is a number that can be written as a real number multiplied by the imaginary unit i, which is defined by its property $i^2 = -1$.

For example, $5i$ is an imaginary number, and its square is -25.

79. What are complex numbers ?

The values which contain both real and imaginary numbers.

For example, $x + iy$, where x and y are real numbers and i is the imaginary unit equal to the square root of -1, $\sqrt{-1}$.