Digital Signal Processing
Lab1
Getting Started with MATLAB

Variables

Unlike many programming languages, Matlab does not require prior definition of the Variables. You can simply write: **variable name = expression**.
For example:
a = sin(64) + 2;
If you do not specify the name of the variable, Matlab automatically creates the
Variable **ans**. Type : 3+2. You will get ans = 5.

Vectors and matrices

```
> x=[1:10]
> x = [1 3 7 15]
> y=[1:0.1:10]
> z=[1:3;4:6;7:9]
> [m,n]=size(z)
```

Arithmetic operations

- Arithmetic operators: +, -, *, /, \, ^.
- Mathematical functions available: ABS, SQRT, LOG, SIN, and COS.

	Mathematical Function	**Matlab Syntax for Function**
Arithmetic and algebraic operation		**f1 = a1 + b1*x + c1*x^2**
		f2 = a2 + b2*x + c2*x^2 + d2*x^3
		g = exp(A*t)*(C1*cos(B*t)+C2*sin(B*t))
		u = 2*x*y^2 + sin(x+y)

Control Flow in Matlab

Loops	**FOR Loops**	**WHILE Loops**	**IF ... ELSE ...**

Syntax	for
	v=expression
	statements
	end
		while expression
		statements
		end
			if expression
			statements
			elseif expression
			statements
			else
			statements
			end

A FOR loop allows a statement to be repeated a fixed, predetermined number of times. Let's look at the following problem. We would like to fill the vector b with square roots of 1 to 1000. One way to do so, is by using a for loop

We will calculate the time required for this operation for comparing it with the more efficient version of this calculation.

Write this code in an m-file and save it under the name *tictoc.m*.

clear ; To clear all previous variables, and to free memory.

tic ; This function initializes an internal clock

for i = 1:1000

b(i) = sqrt(i);

end

t=toc;

The time required was: _________________.

Tasks	**Commands**	Example
<u>Writing your own functions</u>	function [output1,output2,...] = cmd_name(input1,input2,...)	Example of a function to compute $f(x) = sin(x^2)$ function y = fcn(x) y = sin(x.^2); %Create in m file
<u>2D Plotting</u>	**Plot,Subplot,Figure Hold,Stem,Axis,title**	>t=[-2:0.01:2]; >x=sin(t*10); >plot(t,x); >axis([-1 1 -1 1]); >zoom >xlabel(`Time'); >title(`My first plot'); >specgram(x);
<u>Polynomial roots</u>	**r = roots(p)**	>>p = [1 2 1] %polynomial x^2 +2x +1 >>r = roots(p) %roots r = -1 -1
<u>Dealing with sound files</u>	**wavread, wavwrite, auread, auwrite, sound(y,fsamp)**	>y=wavread('C:\sound.wav');%file must be valid >sound(y,44100);
<u>Complex numbers</u>	**j, real, imag, abs, angle,**	> real(j) % locate a complex number in cartesian form

		> imag(j) > abs(j) % locate a complex number in polar form; > angle(j)
<u>Signal processing and Image processing</u>	• **Fft(),dft(),conv()** • **dither(),gray2ind(),ind2gray(),ind2rgb()** • **imread() ,imwrite(,)**	>>clear >>A=imread('my_pic.jpg'); %file must be valid >>whos >>imshow(A)
<u>Transfer function representation and frequency response</u>	• **<u>Tf2zp</u>** • **<u>Zp2tf</u>** • **<u>Freqs()</u>** • **<u>Semilogx()</u>** • **<u>bode()</u>**	Given H(s)=(2s+3)(s3+4s2+5) >num[2 3] >den=[1 4 0 5] > [z,p,k]=tf2zp(num,den) > [num.den]=zp2tf(z,p,k) %one way of plotting >T=0:0.1:1; >Y=step(num,den,t); >Plot(t,y) % another way of plotting >Bode(num,den) > [mag,phase]=bode(num,den,w); >Magdb=20*log10(mag) >Semilogx(w,magdb) >Semilogx(w,phase)

<u>Getting help from Matlab</u>

```
> doc fft
> help help
> help cos
> help fft
> lookfor  filter
```

<u>Demonstration of scripting</u>

1. To invoke scripts from matlab: write your own matlab file using emacs, xemacs or the Matlab editor and save it as myfile.m
2. Type in the Matlab prompt myname
3. Use % as comments
4. ; To suppress output

Exercise:

1. Calculate for $\left(1+\frac{2}{n^2}\right)^n$ n= 3, 7.

2. Plot the function: $y = e\text{-}^{at} * \cos(\omega t)$, for $a = 2$, $\omega = 5$, and $t = 0\text{-}10$.

3. Try using the WHILE and the IF statements to calculate all the Fibonacci numbers sothat the sum of two consecutive numbers is smaller than 10,000. How many are even?How many are odd? Try to plot them.

Hints:

1. Matlab can increase the size of a vector as it is being created.
2. To determine whether a number n is even or odd you can use the function **rem(n,2).** If rem(n,2) equals 0 then the number is even, otherwise it is odd.

4. Given $f(x)= (x^2+2x+3)/(x+3)$.Plot f(x) for $0<=x<=100$

Digital Signal Processing
Lab2
Familiarization with basic CT/DT functions

Functions Used:
Understand the following functions using Matlab online-help feature.
who, whos, input(), disp(), subplot(), figure(), clear all, close all, home, hold on, grid on, grid off, grid, demo, ver, lookfor, length(), pause, plot(), stem(), real(), imag(),zeros(),ones(),exp() function, for statement, if-else statement etc.

Background:
Being the very first lab, the main objective is to familiarize you with the various Matlab functions and use the same to plot the basic signals and display the results.

Problems:

1. Plot the basic signal using Matlab
   a) Impulse response
   b) Unit step
   c) Ramp
   d) Rectangular

2. Plot the following continuous-time signals.
   a) $x(t)=Ce^{at}$ where **C** and **a** are real numbers and choose **C** and **a** both positive and negative.

   b) Plot the same signal taking **a** as pure imaginary number.

   c) Consider complex exponential signal as specified in **b)** where **C** is expressed in polar form i.e., ($C=|C|e^{j\theta}$) & **a** in rectangular form i.e., ($a=r+j\omega_o$). Then your function $x(t)$, on simplification, becomes

   $$x(t)= |C|e^{rt}[cos(\omega_o t+\theta)+jsin(\omega_o t+\theta)]$$

   Now, plot the signal for different values of **r** and comment on the results.
   i $r=0$
   ii. $r<0$
   iii $r>0$

3. Plot the DT exponential function
   $x[n] = a^n, a = |a|e^{j\theta}$
   Choose the suitable value of $|a|$ and θ , then plot the function $x[n]$.

4. Synthesize the signal from the FS Coefficients as $C_0=1, C_1=C_{-1}=1/4, C_2=C_{-2}=1/2, C_3=C_{-3}=1/3$.

5. Plot fundamental sinusoidal signal, its higher harmonics up to 5th harmonics and add all of them to see the result. Comment on the result.

Digital Signal Processing
Lab3
Convolution

Objective:

- To be able to perform convolution of two given signal using basic formula
- To be able to perform convolution of two given signals using Matlab function.

Functions Used:

Conv, *Sinc* etc.

Background:

Convolution Sum:

The output of any Linear Time Invariant (**LTI**) system is some sort of operation between input and system response; the operation is nothing but convolution, denoted by symbol '$*$', and defined as

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau).h(t-\tau)\,d\tau \quad \text{----For continuous time}$$

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k] \quad \text{----For discrete time}$$

For a causal LTI system, convolution sum is given by

$$y(n)= \sum_{k=0}^{n} x(k)h(n-k)$$

The process of computing the convolution between x(k) and h(k) involves the following four steps:

1. Folding : Fold h(k) about k=0 to obtain h(-k)
2. Shifting: Shift h(-k) by n_0 to the right (left if n_0 is positive (negative), to obtain h(n_0-k).
3. Multiplication: Multiply x(k) by h(n_0 –k) to obtain the product sequence V_{n0}(k)=x(k) h(n_0-k).
4. Summation: Sum all the values of the product sequence V_{n0}(k) to obtain the value of the output at times n=n_0.

Don't worry! You can use Matlab's built in function to calculate those.

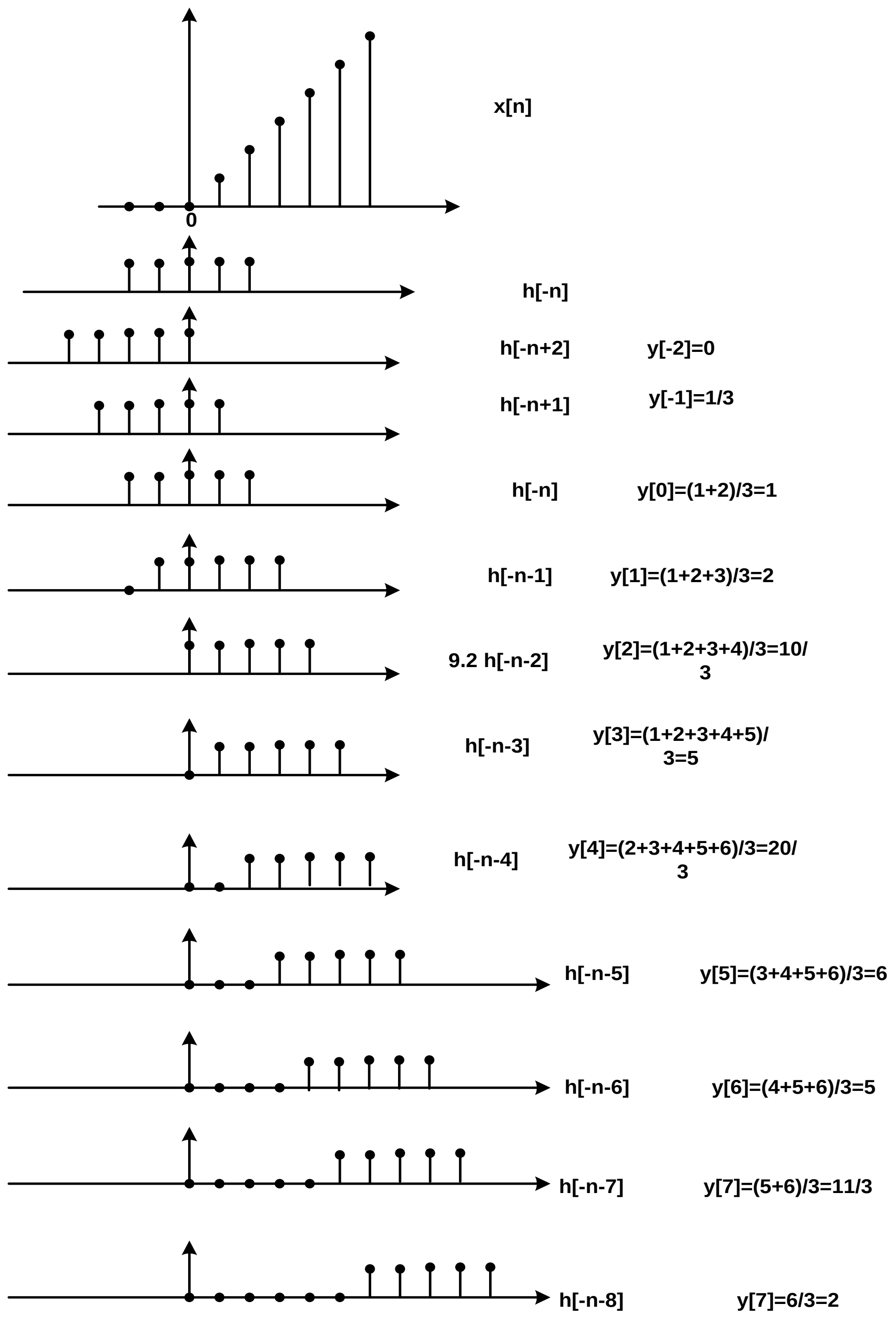
Illustration of convolution :

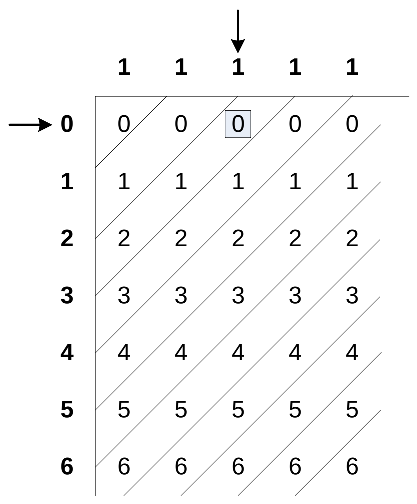
Convolution of signals x[n] and h[n] are obtained by two methods:

$$x[n] = \begin{cases} \frac{1}{3}n & for\, 0 \le n \le 6 \\ 0 & else \end{cases}$$ **and**

$$h[n] = \begin{cases} 1 & for -2 \le n \le 2 \\ 0 & else \end{cases}$$

Graphical Method





y[n]={0,1,3,6,10,15,20,18,15,11,6}/3={0,1/3,1,2,10/3,5,20/3,6,5,11/3,2}

Tabular Method

<u>References:</u>

1. For the folding operation the function is to be formed as follows:

```
function [y,n]=sigfold(x,n)
y=fliplr(x); n=-fliplr(n);
```

2. For shifting operation

```
function[y,n]=sigshift_m(x,m,n0);
n=m+n0; y=x;
```

3. For Multiplication

```
function[y,n]=sigmulti(x1,n1,x2,n2);
n=min(min(n1),min(n2)):
max(max(n1),max(n2));

y1=zeros(1,length(n));
y2=y1;
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
y=y1.*y2;
```

4. Using Conv() function

```
x=[1,0,-1,1,2,1]
n1=[-2,-1,0,1,2,3]
nx=length(x)
h=[1,1,1,1,1]
n2=[0,1,2,3,4]
nh=length(h)
y=conv(x,h)
n=-2:1:-7
% nmin = min(min(n1),min(n2))
% n= nmin:1:nx+nh-2+ nmin
stem(n,y)
```

Problems

1. Find the convolution result of the following signal using basic convolution formula :

   $X_1(n1) = [1,1,1,1,1]$
   n1= [-2,-1,0,1,2]

   X2(n2) = [1,0,0,0,0,0,0,0,0,0]
   n2 = [-4,-3,-2,-1,0,1,2,3,4,5]
   X2 is a periodic signal.
   Y2= X1*X2

2. Find the convolution of using **conv** function

   a. $x[n] = \begin{cases} \frac{1}{3}n & \text{for } 0 \le n \le 6 \\ 0 & \text{else} \end{cases}$ **and**

   $h[n] = \begin{cases} 1 & \text{for } -2 \le n \le 2 \\ 0 & \text{else} \end{cases}$

   b. *x(t)= u(t)*
   $h(t)= e^{-at} u(t)$, where $a>0$

3. Consider two discrete time sequences **x[n]** and **h[n]** given by

   x[n] = 1 for $0 \le n \le 4$, elsewhere 0
   h[n] = 2^n for $0 \le n \le 6$, elsewhere 0

   a. Find the response of the LTI system with impulse response **h[n]** to input **x[n]**.
   b. Plot the signals and comment on the result.

4. If the impulse response of a LTI system is given by **sinc** function as

   $h[n] = 2\tau/T_p \text{ sinc}(k\ 2\tau/T_p)$

   and input signal is a rectangular wave given by

   x(t) = 1 for $1 \le t \le 100$

   0 elsewhere,

   Find output of the system for different values of τ. Comment on the result.

Digital Signal Processing
Lab 4
LTI system

Background:

$$x[n] \longrightarrow \boxed{H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}} \longrightarrow y[n]$$

A **linear time invariant (LTI)** system is characterized by the transfer function H(z)=Y(z) / X(z), where X(z) and Y(z) are the Z-transforms of the sequences x(n) and y(n) respectively. When the inverse Z-transform of the transfer function H(z) is taken, the resulting difference equation can be written as:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] - a_1 y[n-1] - a_2 y[n-2] - \cdots - a_N y[n-N]$$

The digital signal processing basically deals with the methods of implementation of the above difference equation. The equation can be solved using both hardware and software. It can be implemented using microprocessor based designs in which the assembly language programming plays the vital role. For the fast processing of the signals, considering the improved system performance the digital signal processing chips are preferred. The DSP chips are based on the Harvard architecture rather than the Von-Neumann's architecture, usually found in most of the personal computers.

If the transfer function H(z) is known for the given LTI system the MATLAB signal processing toolbox functions can be used to plot the frequency response of the system. For this there is a function; [H, W]= freqz(b, a, w) which gives the complex values in amplitude H and angle W radians versus w points frequency. Here 'b' & 'a' are the vector sequences representing the numerator and denominator coefficients of H(z).

Linear Systems Transformation

In discrete time systems the transfer function in the Z domain plays the key role in determining the nature of the system. The nature of the system is determined from the number and locations of the poles and zeros in the Z-plane. In lower order systems the locations of the poles and zeros can be easily determined from the transfer function of the system. However the higher order systems possess transfer functions with numerator and denominator polynomials of greater degree. As a result the process of determining the poles and zeros of the system becomes complex and tedious. Besides this in most of the higher order discrete time systems, the transfer function is specified in the form of second order sections. If the transfer function of the system consists of a large number of such second order sections either in cascade or parallel form, the determination of the poles and zeros of the system becomes even harder.

MATLAB signal processing toolbox provides a number of functions for transforming the discrete time linear systems from one form to another. The name of the functions and their purpose are listed as follows:

sos2zp : transforms second order sections into zeros and poles.
sos2tf : performs second order sections to transfer function conversion.
tf2zp : transfer function to pole zero conversion.
zp2sos : zero-poles to second order sections.
zplane : plots the pole-zero diagram in Z-plane.
freqz : determines the magnitude and phase of the transfer function.

For further information on the above functions, please refer to the MATLAB 'help'.

Statements of the Problem:

1. In the given LTI system of fig above, if the coefficients 'b' & 'a' are specified as

   b0=0.0663, b1= 0.1989, b2=0.1989, b3=0.0663
   a0=1, a1= -0.9349, a2=0.5668, a3= -0.1015,

   then the order of the system is 3 i.e. N=3.

   a. Plot the frequency response of the system.
   b. From the magnitude response of the system, find out the cut-off frequency.
   c. Identify the nature of the system analyzing its frequency response.

2. The transfer function of the fourth-order discrete time system is given as:

$$H(z) = \frac{0.0018 + 0.0073\, z^{-1} + 0.011\, z^{-2} + 0.007\, z^{-3} + 0.008\, z^{-4}}{1 - 3.0544\, z^{-1} + 3.8291\, z^{-2} - 2.2925\, z^{-3} + 0.55072\, z^{-4}}$$

   Using MATLAB-
   a. Find out the poles and zeros of the system and plot them in the z- plane.
   b. Use them to determine the second order sections in the cascaded form.
   c. Plot the frequency response of the system and comment on the nature of the system.
   d. After knowing the numerator and denominator coefficients of each second order section, draw the signal flow graph to represent the cascaded structure.

3. Let a discrete time system be implemented by cascading of the following three second order sections:

   Section 1: $$\frac{0.0007378\,(1 + 2\, z^{-1} + z^{-2})}{(1-1.2686\, z^{-1} + 0.7051\, z^{-2})}$$

   Section 2: $$\frac{1 + 2\, z^{-1} + z^{-2}}{1- 1.0106\, z^{-1} + 0.3583\, z^{-2}}$$

   Section 3: $$\frac{1 + 2\, z^{-1} + z^{-2}}{1 - 0.9044\, z^{-1} + 0.2155\, z^{-2}}$$

   a. Using above three second order sections in cascaded form determine the poles and zeros of the system and plot them in z-plane.
   b. Determine the transfer function of the system, formed by cascading of the above three sections. Determine the poles and zeros from this transfer function and plot them in z-plane. Your result should match with that from 3(a).
   c. Draw the direct form structures-I & II of the system.

Digital Signal Processing
Lab5

Design of IIR Digital Filters

Background:

There are several methods that can be used to design digital filters having an infinite duration unit sample response. One of the popular methods is based on converting an analog filter into a digital filter. In this method we begin the design of digital filter in the analog domain and then convert the design into the digital domain. For this purpose, depending on the specifications of the required digital filter the various approximations like butterworth, chebyshev, chebyshev2 and elliptic filters are used.

Among the different approaches used in the design of digital IIR filters this lab session deals with the,

1. Impulse Invariance method.
2. Bi-Linear Transformation.

In Impulse Invariance method, the objective is to design an IIR filter having an unit sample response **h(n)** that is the sampled version of the impulse response of the analog filter.That is

h(n) = h(nT) n=0,1,2,........,where T is the sampling interval.

In Bi-Linear transformation a conformal mapping from s plane to z plane is carried out with the relation

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

For the design of IIR digital filters, there are built-in functions in MATLAB such as Impinvar(..), bilinear(..), butter(..), cheby1(..), cheby2(..), ellip(..), buttord(..), cheb1ord(..), cheb2ord(..), ellipord(..) etc. For plotting the impulse responses for analog and digital filters refer to the functions impulse(..) and dimpulse(..). For the details of these functions use MATLAB HELP.

Problems:

A.

1. Convert the analog filter $H_a(s) = \dfrac{s+0.1}{(s+0.1)^2+9}$ in to a digital IIR filter by means of the impulse invariance method. Plot the frequency response (magnitude) of the designed filter taking sampling interval (T) of 0.1, 0.5 seconds. Compare the response of the filter designed to that of the analog one. Comment on the effect of T on the response.
2. Compare the unit sample response of the designed digital IIR filter with the impulse response of analog filter for T=0.1 and 0.5.
3. Convert the above analog filter in to a digital IIR filter by means of bilinear transformation and repeat all the procedures as specified in 1.1.

B.

An IIR digital low pass filter is required to meet the following specifications:

Pass band ripple (or peak to peak ripple): ≤ 0.5 dB

Passband edge: 1.2 kHz

Stopband attenuation: ≥ 40 dB

Stopband edge: 2.0 kHz

Sample rate: 8.0 kHz

Use the MATLAB Signal Processing Toolbox functions to determine

- The required filter order,
- The cutoff frequency,
- The numerator and the denominator coefficients

for the digital Butterwoth, digital Chebyshev and digital Elliptic filters. Also plot their frequency responses. Describe the nature of each response.

Digital Signal Processing
Lab6

Design of FIR Digital Filters using Window method

Background:

In previous experiment we discussed the most commonly used techniques for design of IIR filters based on transformations of continuous-time IIR systems into discrete time IIR systems. The major difficulty lies in the implementation of the non-iterative direct design method for IIR filters. However FIR filters are almost entirely restricted to discrete-time implementations. Thus the design techniques for FIR filters are based on directly approximating the desired frequency response of the discrete time system. Furthermore, most techniques for approximating the magnitude response of the FIR system assume a linear phase constraint; thereby avoiding the problem of spectrum factorization that complicates the direct design of IIR filters.

The simplest method of FIR design is called the window method. This method generally begins with an ideal desired frequency response $H_d(w)$. The impulse response $h_d(n)$ of the filter exhibiting this desired frequency response can be obtained from inverse Fourier transform of $H_d(w)$. However this impulse response exists for n= $-\infty$ to $+\infty$ and hence truncation is needed to make the finite duration impulse response. The truncation is similar to the multiplication of the $h_d(n)$ with the window function w(n). The multiplication in discrete time domain is equivalent to the convolution of the two in frequency domain, which actually gives the frequency response of the truncated FIR filter.

But depending on the tapering of the window to zero at each end, the nature of the window differs. For example, the rectangular window exhibits the most abrupt changes while approaching to zero at each end. For the specific value of length of the window, the rectangular window exhibits main lobe with the greatest width and the lowest side lobe attenuation, than the other windows like Bartlett, Hanning, Hamming, Blackman etc. There are also adjustable windows like Kaiser windows whose windowing function w(n) can be adjusted by changing the value of parameter β according to the stop band attenuation A as given below:

$$\beta = \begin{cases} 0.1102(A-8.7), & \text{if } A>50, \\ 0.5842(A-21)^{0.4}+0.07886(A-21), & \text{if } 21 \le A \le 50, \\ 0.0, & \text{if } A<21, \end{cases}$$

where $A= -20\log_{10} \delta$.

The length of the Kaiser window is given by;

$$M = ((\delta-7.95)/(14.36*\Delta w)) +1, \text{ where } \Delta w= (w_s -w_p)/2$$

MATLAB Signal Processing Toolbox provides built-in functions to determine and plot the different types of windowing functions and their frequency responses. For this the functions like Bartlett( ), Hamming( ), Hanning( ), Blackman( ) , Kaiser( ) etc. are available. There is a function fir1( ) that can be used to obtain the frequency response of the designed FIR filter. Use MATLAB 'Help' for further information of the functions.

Problems:

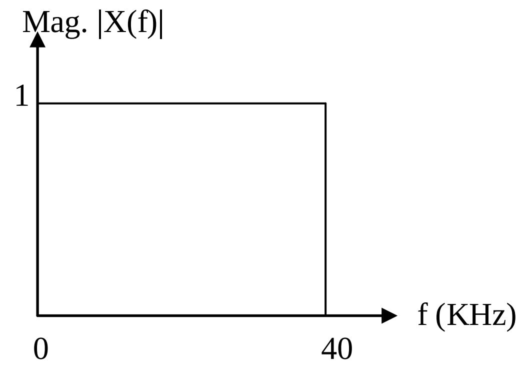
1. Design an FIR linear phase digital filter approximating the ideal frequency response

$$H(w) = \begin{cases} 1, & \text{for } |w| \le \pi/6 \\ 0, & \text{for } \pi/6 < |w| \le \pi \end{cases}$$

   a. Plot the window function for Hamming and its frequency response for length of M=31.
   b. Using the Hamming window plot the frequency response of the truncated FIR filter.
   c. Repeat parts (a) and (b) for the Hanning, Blackman and Bartlett windows.
   d. Repeat parts (a), (b) and (d) for filter length of M=61.

2. Discuss the effects of different types of the windowing functions on the frequency response of the FIR filter. Carry out the comparative study on the basis of the peak side-lobe level, the approximate transition width of the main lobe etc. Also discuss on the effects of increasing value of M.

3. An analog signal x(t) consists of the sum of two components $x_1(t)$ and $x_2(t)$. The spectral characteristics of x(t) is shown in fig 2.1. The signal x(t) is band limited to 40kHz and is sampled at the rate of 100kHz to yield the sequence x(n) .



It is desired to suppress the signal $x_2(t)$ by passing the sequence x(n) through a digital low pass filter. The allowable distortion on $|X_1(f)|$ is ±2% (δ_1=0.02) over the range $0 \le |F| \le 15$ kHz. Above 20 kHz, the filter must have an attenuation of at least 40 dB (δ_2=0.01).

   a. For obtaining the filter with above specifications, use the Kaiser window and determine the length of the required window. Plot the frequency response of the filter and its impulse response also.
   b. If the same filter is to be designed using Hamming window what would be the length of the required window. Plot the frequency and impulse response of the filter.