The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

Digital Signal Processing (EIE413 / EIE4413)
Lab 2: FIR Filter Design

Purpose

This laboratory work explores the implementation of various fundamental concepts in digital finite impulse response filter (FIR) design using Matlab. It serves as the supplement to the discussion in class.

Things to do

For each question in this lab sheet, give your answer right under the question. Submit this lab sheet (with your answers) through the Blackboard to your lecturer before the deadline.

Equipment

PC with Windows 7 or above
PC with MATLAB 7 or above

Part 1. FIR filter design

Q.1.1 The following specifications are obtained for the design of an FIR low pass filter:

- passband edge frequency: f_p = 4960 Hz ±50Hz
- passband ripple $\delta_p < 0.005$
- transition width for both edges: ≤ 0.75 kHz
- sampling frequency = 16 kHz

Assume that the filter is designed using the windowing method. Determine

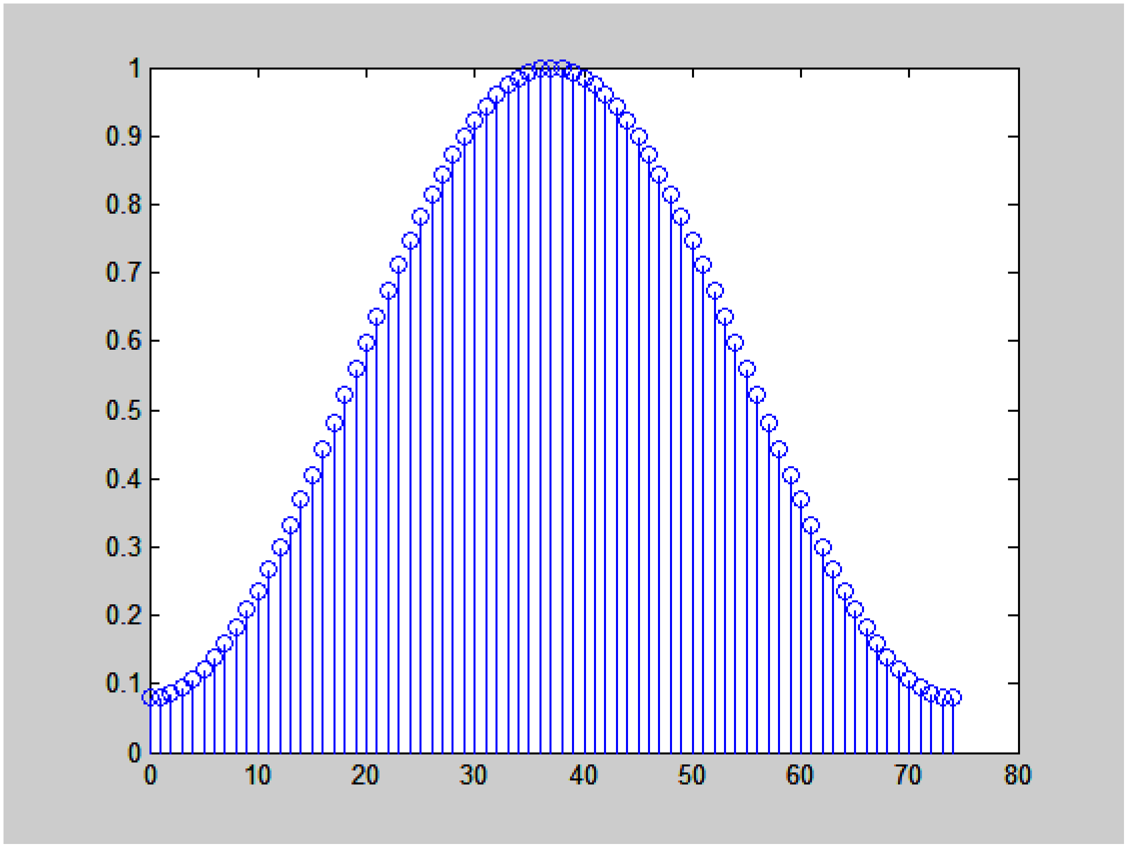
(a) The window to be chosen. Give the reason why.
We need to choose Hamming Window because the passband ripple δ_p < 0.0433dB and Hamming Window's passband ripple is 0.0194. We need to use the smallest M to design an FIR low pass filter, so the Hamming Window is a suitable window method.

(b) The value of *M*.
ΔF = 0.75/16 = 0.046875
$2M = 3.3 / \Delta F = 70.4 \Rightarrow M$ is selected as 37 which is a suitable and proper value.

(c) The passband frequency $f_{p(ideal)}$ of $h_D[n]$ used to design the filter.

$f_{p\,(ideal)}$ = 4960+ 750 / 2 = 5335 Hz

Q.1.2 Use the command **window** to generate the window function you have chosen. Plot the window of your choice below (remember the window size is 2M+1).

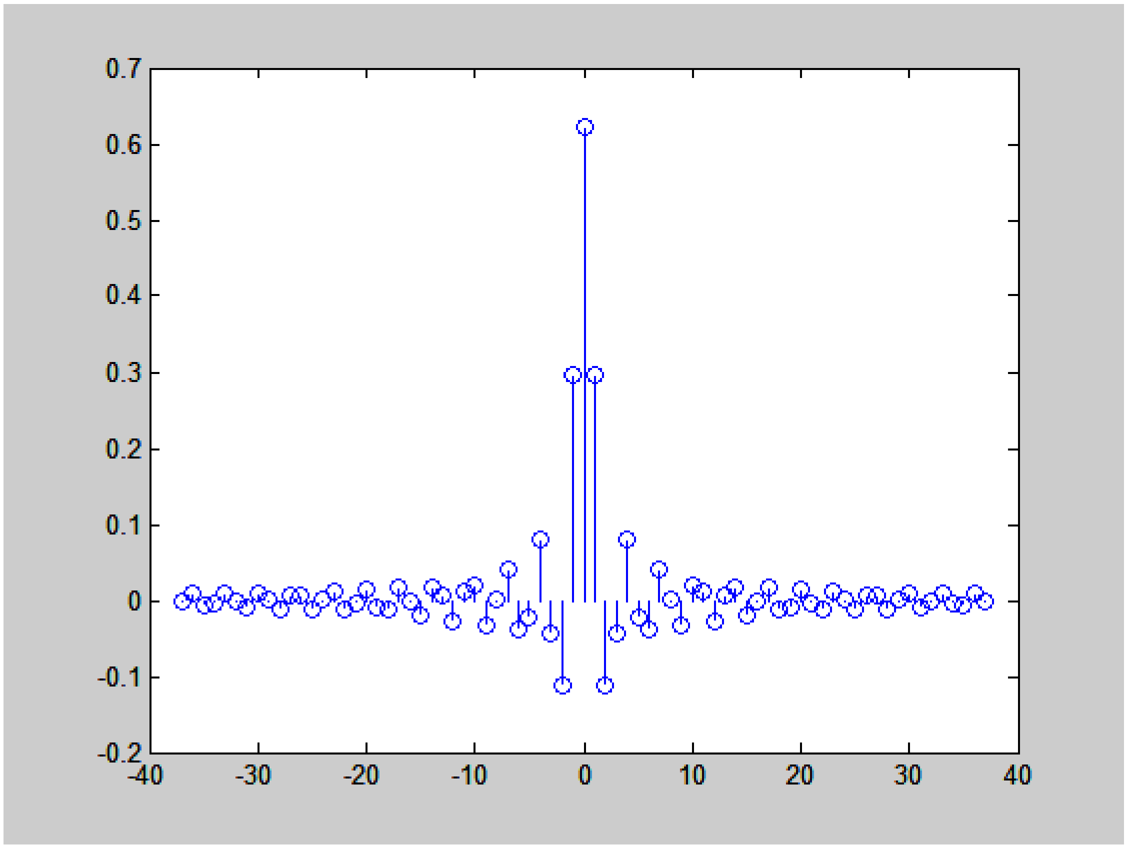


Q.1.3 Note that the impulse response of an ideal low-pass filter is a sinc function as follows:

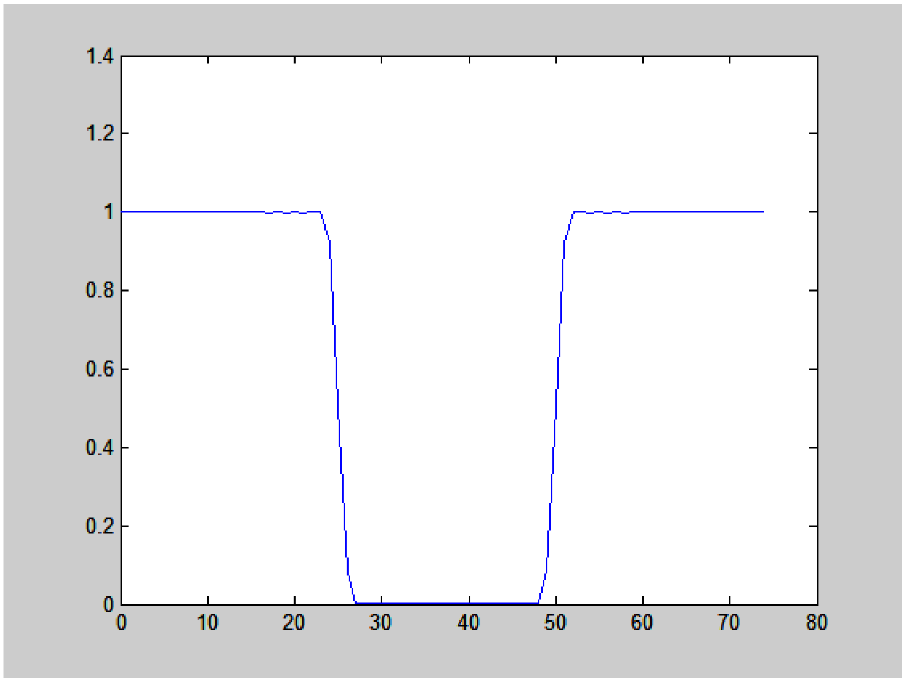
$$h_D(nT_s) \equiv h_D[n] = \frac{T_s}{2\pi}\int_{-\omega_p}^{\omega_p} 1e^{j\omega nT_s}d\omega$$

$$= \begin{cases} \dfrac{\sin \omega_p nT_s}{n\pi} & n = \pm 1, \pm 2, \ldots \pm \infty \\ \omega_p T_s / \pi & n = 0 \end{cases}$$

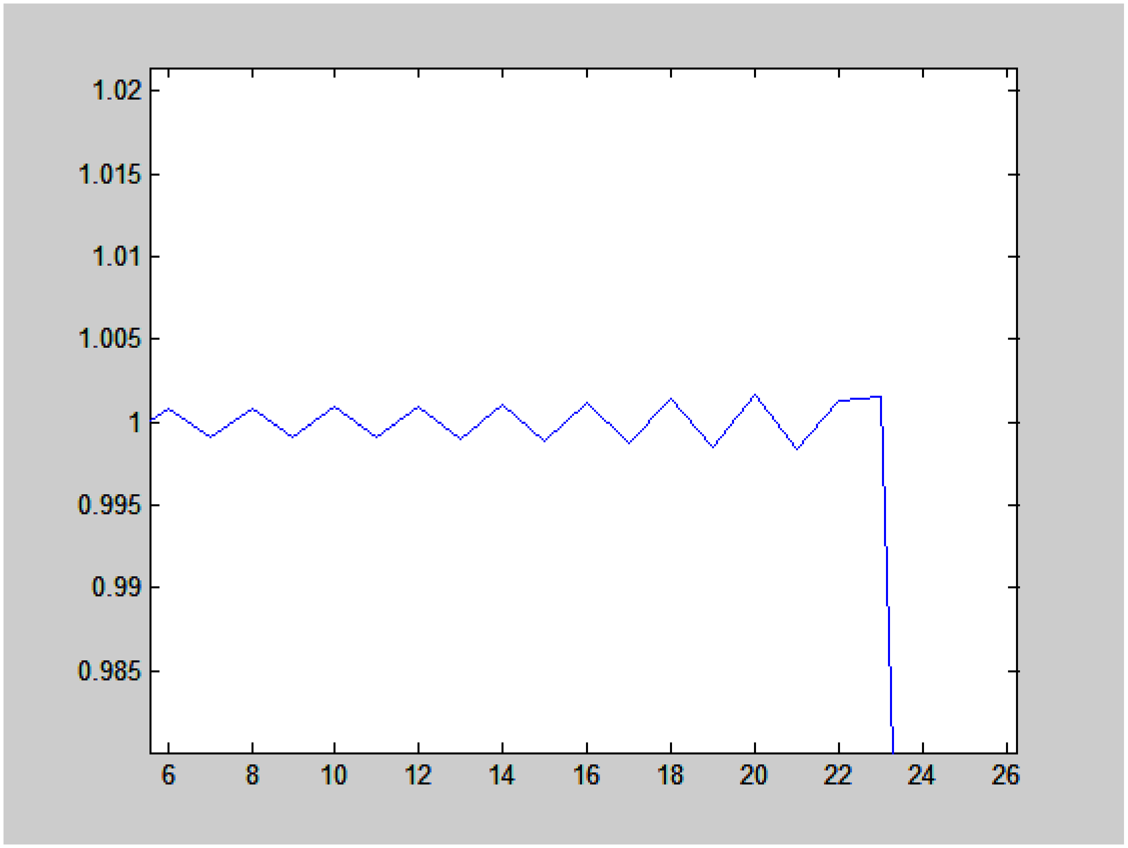
where ω_p is the edge radian frequency and T_s is the sampling period. Based on the ideal low-pass filter you have selected, plot the corresponding sinc functions below. Note that your sinc function should also have the length of 2M+1 so as to multiple with the window in Q.1.2 later.



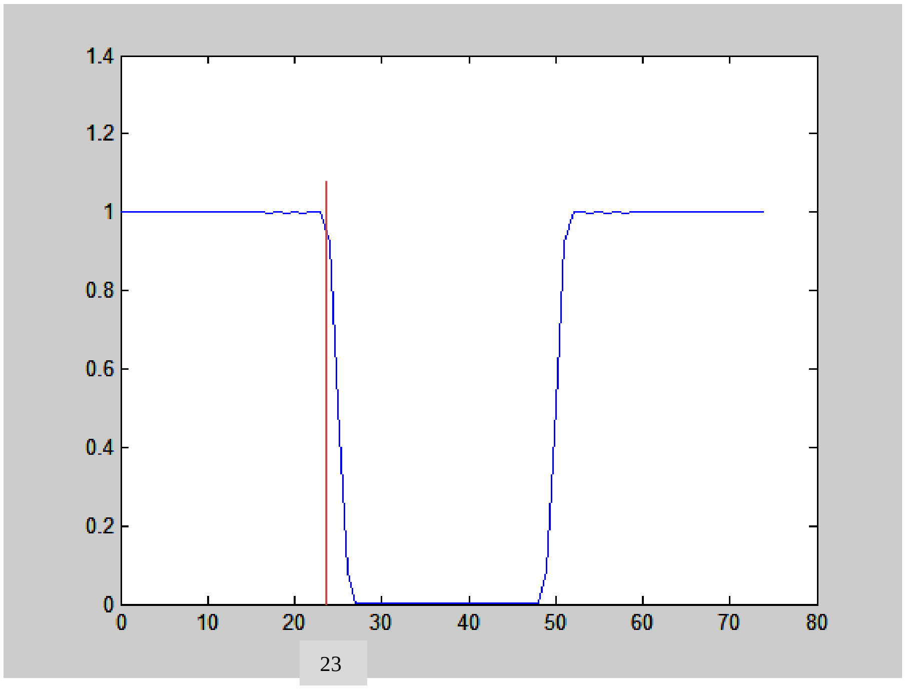
<u>Compute the coefficients of the FIR low pass filters by multiplying the selected window functions with the sinc function in Q.1.3. Plot the magnitude response of the resulting band pass filter below.</u>



Q.1.4 Zoom in the passband of the magnitude response to show that the resulting FIR low pass filter meets the specifications on passband ripple and edge frequency. (Make sure that the figure shown is clear enough to illustrate that the filter meets the specifications.)



Passband ripple is 0.005.

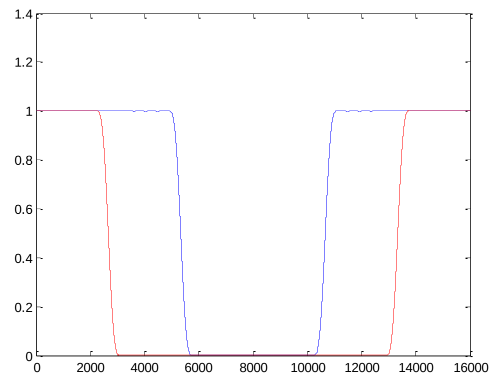


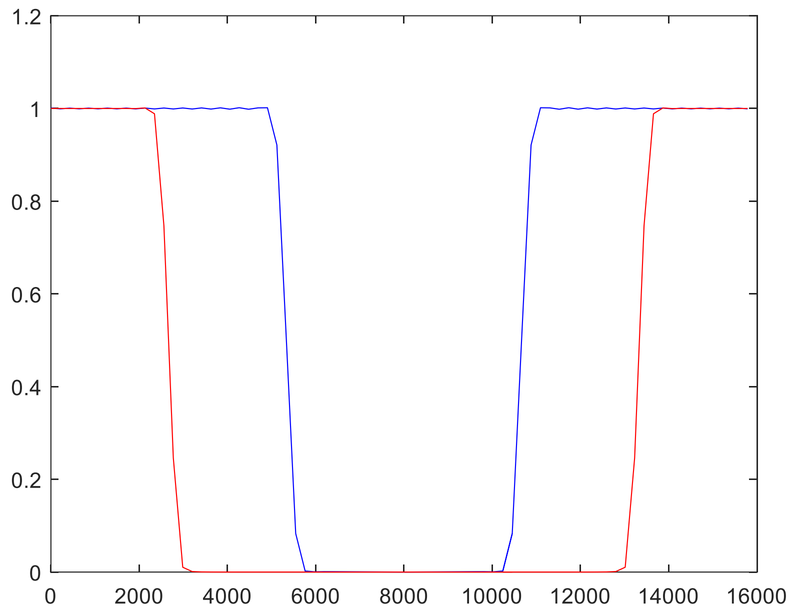
23/75*16000=4906 Hz

Edge frequency is 4906Hz.

Part 2. Hear the effect of low pass FIR filters

Q.2.1 Design another low pass FIR filter which has the same specifications on passband ripple, transition width, and sampling frequency as the one in Q.1. However, the passband edge frequency is changed to 2290 Hz. Plot the magnitude response of both low pass filters you have designed in the same figure using the command **hold on/off**. An example is given below. **<u>Show the result together with that in Q.2.3 to your tutor</u>**.





Q.2.2 Download the music file "music16k.wav" from the Blackboard. Use the command **wavread** to load the music file into a two-dimensional array (since the original music is stereo). Note that the sampling rate of the music is 16 kHz. Play the music using the command **sound**.

Q.2.3 Write a Matlab program to convolve the music with both the low pass filters and play the resulting music individually. Your Matlab program should include the implementation of both filters. List your Matlab programs below. Then listen to the resulting music. Can you find the difference of using the two filters? **Tell the tutor your finding and also show the result of Q.2.1 to your tutor.**

```
N=-37:37;
M=window(@hamming,75);
wp=2*pi*5335;
ts=1/16000;
hD=sin(wp*ts*N)./(N.*pi);
hD(38)=wp*ts/pi;
M=M';
W=M.*hD;
wp1=2*pi*2665;
hD1=sin(wp1*ts*N)./(N.*pi);
hD1(38)=wp1*ts/pi;
W1=M.*hD1;
filename='music16k.wav';
Fs=16000;
[y,Fs]=wavread('music16k.wav');
y1=y(:,1);
y2=y(:,2);
s1=conv(y1,W);
s2=conv(y2,W);
s3=[s1;s2];
sound(s3);
y3=y(:,1);
y4=y(:,2);
s4=conv(y3,W);
s5=conv(y4,W);
s6=[s4;s5];
sound(s6);
```

The first filter allows more high frequencies to pass so we can hear more high-frequency elements in the first music.
The second filter allows less high frequencies to pass so we can hear less high-frequency elements in the second music.

The End

<u>Appendix A : The Function of MATLAB</u>

A. General

help [instruction]	---	Displays help for [instruction]
help	---	Displays a list of available instructions
dir (or ls)	---	Lists directory
cd	---	Change directory
who (whos)	---	Lists current variables
format	---	Set output (display) format, e.g. format compact
; (after command)	---	Suppresses the display of result

B. Matrix

Row matrix : A = [1,2,3] or A = [1 2 3] Column

matrix: A = [1:1:3] or A = [1, 2, 3]' Rectangular

matrix: A = [1 2 3 4: 5 6 7 8]

To access an element: c = [A (row, column)]; e.g. A(2,3)

To access an row: c = [A (row,:)]; e.g. A(1,:) = the first row of A To access an

column: c = [A (:,column)];

Identity matrix: b = eye (n), where n is an integer

Ones matrix: x = ones(m,n) Zeros matrix:

y = zeros(m,n)

Length(x)	---	Shows the length of vector x
Size(A)	---	Shows the size of matrix A in (row, column)

C. Special Variable and Constants

Ans --- Most recent answer

eps --- Floating point relative accuracy i, j ---

Imaginary unit; e.g. c = 2 + 5I inf --- Infinity

pi --- π

Nan --- Not a number

D. Control Flow

if {condition} – else – end

if {condition} – elseif {condition}– else – end

for j = 1:N – end

while {condition} – end

E. Mathematical Operators

+	→ Plus	+	→ Unary plus
-	→ Minus	-	→ Unary minus
*	→ Matrix multiply	*.	→ Array multiply
\	→ Backslash or left matrix divide	/	→ Slash or right matrix divide
.\	→ Left array divide	./	→ Right array divide
^	→ Matrix power	.^	→ Array power

F. Functions

abs	---	Absolute value
angle	---	Phase angle
cos	---	cosine
sin	---	sine
imag	---	Complex imaginary part
real	---	Complex real part

G. Useful Functions

y = fft (x) or y = fft(x, n)	---	FFT of x ; n is number of point
[z, p, k] = buttap(n)	---	butterworth analog prototype
[num, den] = zp2tf (z, p, k)	---	zero-pole to transfer function
[bz, az] = impinvar (num, den, fs)	---	impulse invariant transformation
[zd, pd, kd] = bilinear (z, p, k, fs)	---	bilinear transformation
[h,w] = freqz (num, den, n)	---	Z-transform digital filter frequency response
w = hamming (n)	---	hamming window
stem	---	plot the signal in the impulse form

References

[1] S.K. Mitra, *Digital Signal Processing*, 3rd Edition, McGraw-Hill Education (Asia), 2009.
[2] J.G. Proakis and D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*, 4th Edition, Pearson International Edition, 2007.
[3] McClellan, Schafer and Yoder, *DSP FIRST: A Multimedia Approach*. Prentice Hall, Upper Saddle River, New Jersey, 1998 Prentice Hall.
[4] *Using Matlab*, The Math Works Inc.