**EXPERIMENT-8**

**Name of the experiment:**

To study and implement Low pass filtering, Discrete Fourier transform and inverse discrete Fourier transform.

**Theory:**

Ideal LPF:

A low-pass filter (LPF) is a filter that passes signals with a frequency lower than a certain cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency.

Impulse response h=∫𝑒𝑗2𝜋𝑓𝑡𝑑𝑡 (B to –B) =2𝐵 𝑆𝑖𝑛𝑐 (2𝐵𝑡)



Time domain Frequency domain

The discrete Fourier transform transforms a sequence of N complex numbers x0,x1,x2……xN−1 into another sequence of complex numbersX0,X1,X2……XN−1.

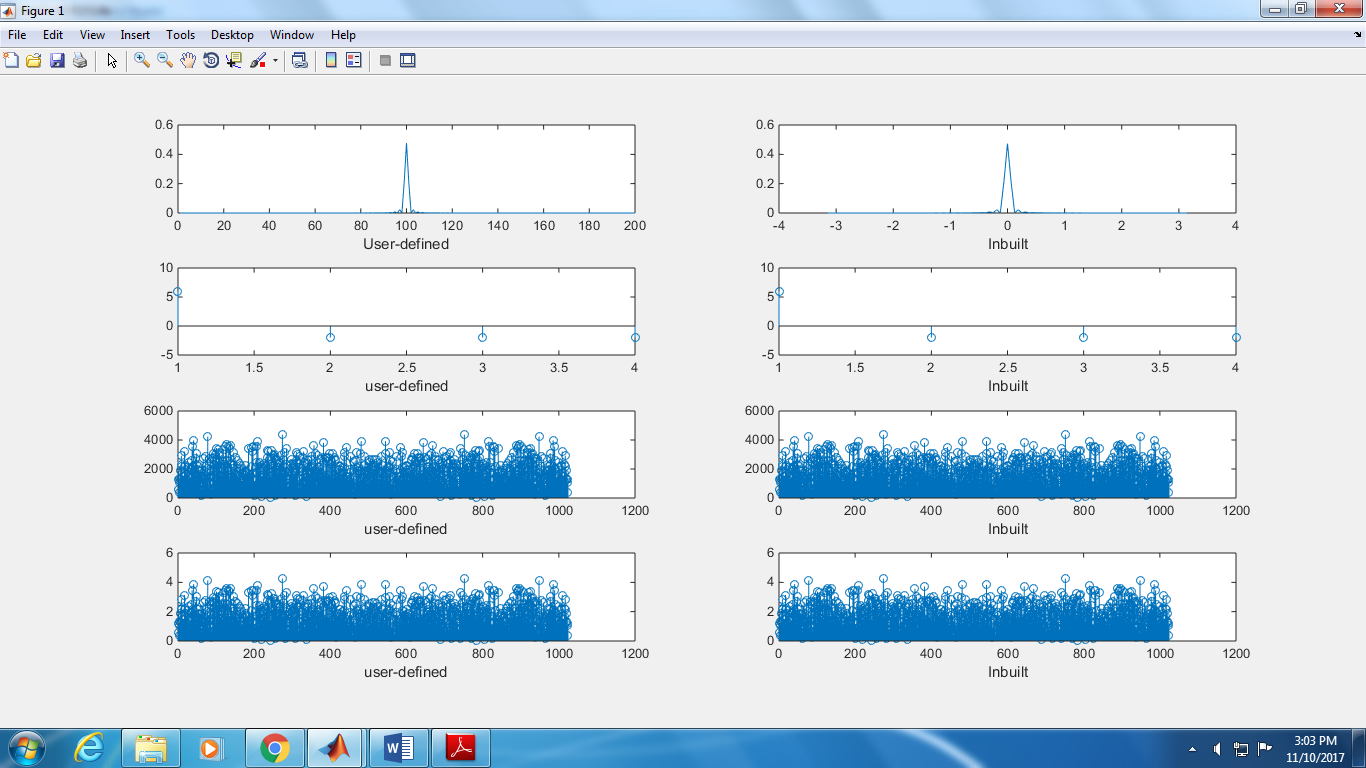
X( k)=Σ𝑥(𝑛)𝑒-j(2\*pi/N)\*(k\*n) k=0, 1, 2………N-1

And inverse DFT is calculated by using following equation

𝑥(𝑛)=(1/N)Σ𝑋(𝑘) j(2\*pi/N)\*(k\*n) =0 n=0, 1, 2………N-1

**Observation:**

The graphs for all the functions are plotted in below figure (left hand side showing my implementation and right-hand side, the MATLAB inbuilt plot). All the functions were validated by this procedure.



**Conclusion:**

This experiment clarified our concepts about discrete Fourier and discrete Inverse Fourier transforms. We also learnt how to implement the functionality of a low pass filter and understood how to implement different forms of filter too. Learning about inbuilt MATLAB functions like fplot(), fft() and ifft() was also educational and informative. All in all, this experiment helped us to learn a lot about signal processing.

**Appendix:**

MATLAB code:

clc;

clear all;

close all;

k=zeros(1,200);

% t=linspace(-3\*pi,3\*pi,1000);

% figure;

% plot(x1(t));

for i=1:1:200

sig=@(w) x1(w/(3)).\*rect(w).\*exp(1j\*w\*(i-100));

k(i)=integral(sig,-20,20)/(2\*pi);

end;

subplot(4,2,1);

t=linspace(-pi,pi,101);

plot(abs(k));

xlabel('User-defined');

subplot(4,2,2);

plot(t,fftshift(abs(ifft(x1(t/3).\*rect(t)))));

xlabel('Inbuilt');

mat=zeros(1,4);

rix=[0 1 2 3];

for i=1:1:4

for r=1:1:4

mat(i)=mat(i)+rix(r).\*exp(-2j\*(pi/4)\*(i-1)\*(r-1));

end;

end;

subplot(4,2,3);

disp(mat);

stem(((mat)));

xlabel('user-defined');

subplot(4,2,4);

stem((fft(rix)));

xlabel('Inbuilt');

rix = randi([-100 100],1,1024);

mat=zeros(1,1024);

for i=1:1:1024

for r=1:1:1024

mat(i)=mat(i)+rix(r).\*exp(-2j\*pi/1024\*(i-1)\*(r-1));

end;

end;

idft=zeros(1,1024);

subplot(4,2,5);

stem((abs(mat)));

xlabel('user-defined');

subplot(4,2,6);

stem(abs(fft(rix)));

xlabel('Inbuilt');

for i=1:1:1024

for r=1:1:1024

idft(i)=idft(i)+rix(r).\*exp(2j\*pi/1024\*(i-1)\*(r-1));

end;

idft(i)=idft(i)/1024;

end;

subplot(4,2,7);

stem(abs(idft));

xlabel('user-defined');

subplot(4,2,8);

stem(abs(ifft(rix)));

xlabel('Inbuilt');

dif=abs((idft))-abs(ifft(rix));

%stem(dif);

Code for rect():

function [ a ] = rect( t )

%UNTITLED Summary of this function goes here

% Detailed explanation goes here

a=abs(t)<=pi;

end

Code for x1() : [Tri]

function [y ] = x1( t )

%UNTITLED2 Summary of this function goes here

% Detailed explanation goes here

y=ramp(t+1)-2\*ramp(t)+ramp(t-1);

%plot(t,r1-2\*ramp+r2);

%plot(y);

end

Code for ramp():

function [ ramp ] = ramp(t )

%UNTITLED4 Summary of this function goes here

% Detailed explanation goes here

%t=(-4:0.01:4)';

%unitstep=t>=0;

ramp=(t).\*unitstep(t);

end