The L.N.M. Institute of Information Technology, Jaipur

Digital Signal Processing Project

Topic of the Project: ECHO CANCELLATION FROM A SIGNAL

(HYBRID ECHO CANCELLATION TYPE)

OBJECTIVE: 1> Echo cancellation using Adaptive Filter only.

2> Hybrid echo cancellation.

***INTRODUCTION*:**

There are various key terms that we need to understand before we start the project. Firstly, we describe a few terms that will help for further proceedings. The terms are described as follows:

ECHO- In audio signal processing and acoustics, an **echo** (plural **echoes**) is a reflection of sound, arriving at the listener some time after the direct sound. Typical examples are the echo produced by the bottom of a well, by a building, or by the walls of an enclosed room and an empty room. A true echo is a single reflection of the sound source. The time delay is the extra distance divided by the speed of sound.

In terms of project, when we refer to a signal, so while recording the signal unintentionally some noise is already present. So, when two waves like these superimpose, one acts as original signal and the other as feedback or echo. Thus for clear interpretation of the signal, we need to filter the unwanted signals as much as possible.

Hence, we now understand echo cancellation.

ECHO CANCELLATION - 'The term *echo cancellation'* is used in telephony to describe the process of removing echo from a voice communication in order to improve voice quality on a telephone call. In addition to improving subjective quality, this process increases the capacity achieved through silence suppression by preventing echo from traveling across a network.

Two sources of echo have primary relevance in telephony: **acoustic echo** and **hybrid echo**.

Echo cancellation involves first recognizing the originally transmitted signal that re-appears, with some delay, in the transmitted or received signal. Once the echo is recognized, it can be removed by 'subtracting' it from the transmitted or received signal. This technique is generally implemented using a digital signal processor (DSP), but can also be implemented in software. Echo cancellation is done using either echo suppressors or echo cancellers, or in some cases both.

Now, we discuss about Acoustic echo and Hybrid echo.

Acoustic echo

Acoustic echo arises when sound from a loudspeaker—for example, the earpiece of a telephone handset—is picked up by the microphone in the same room—for example, the mic in the very same handset. The problem exists in any communications scenario where there is a speaker and a microphone. Examples of acoustic echo are found in everyday surroundings such as:

- Hands-free car phone systems
- A standard telephone or cellphone in speakerphone or hands-free mode
- Dedicated standalone "conference phones"
- Installed room systems which use ceiling speakers and microphones on the table
- Physical coupling (vibrations of the loudspeaker transfer to the microphone via the handset casing)

In most of these cases, direct sound from the loudspeaker (not the person at the far end, otherwise referred to as the Talker) enters the microphone almost unaltered. This is called direct acoustic path echo. The difficulties in cancelling acoustic echo stem from the alteration of the original sound by the ambient space. This colours the sound that re-enters the microphone. These changes can include certain frequencies being absorbed by soft furnishings, and reflection of different frequencies at varying strength. These secondary reflections are not strictly referred to as echo, but rather are "reverb".

Acoustic echo is heard by the far end talkers in a conversation. So if a person in Room A talks, they will hear their voice bounce around in Room B. This sound needs to be cancelled, or it will get sent back to its origin. Due to the slight round-trip transmission delay, this acoustic echo is very distracting.

Hybrid echo

Hybrid echo is generated by the public switched telephone network (PSTN) through the reflection of electrical energy by a device called a hybrid (hence the term hybrid echo). Most telephone local loops are two-wire circuits while transmission facilities are four-wire circuits. Each hybrid produces echoes in both directions, though the far end echo is usually a greater problem for voiceband.

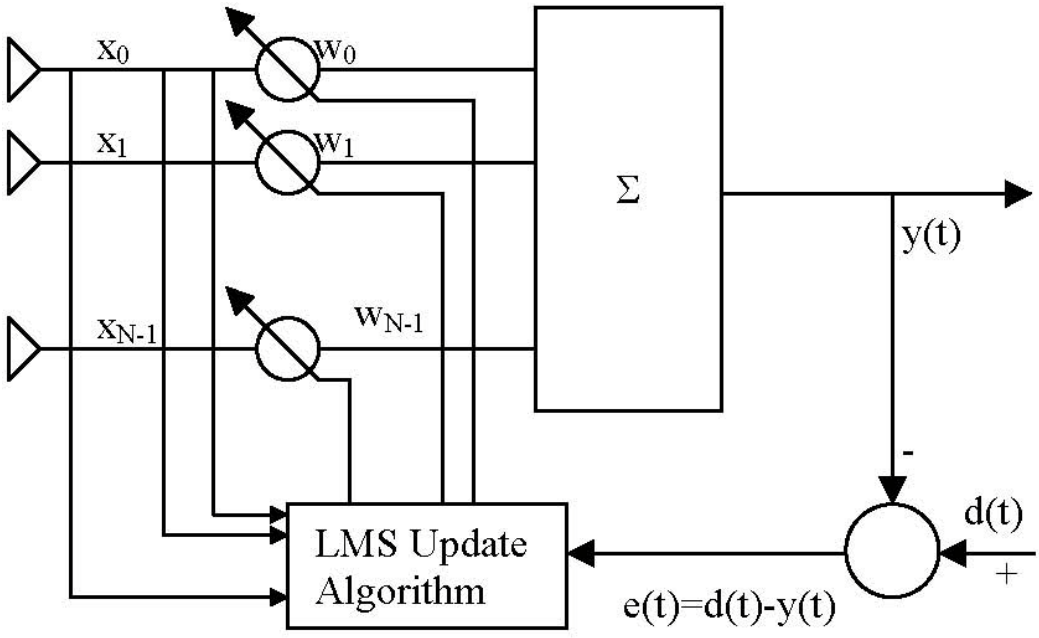
ADAPTIVE FILTER

An **adaptive filter** is a filter that self-adjusts its transfer function according to an optimization algorithm driven by an error signal. Because of the complexity of the optimization algorithms, most adaptive filters are digital filters. By way of contrast, a non-adaptive filter has a static transfer function. Adaptive filters are required for some applications because some parameters of the desired processing operation (for instance, the locations of reflective surfaces in a reverberant space) are not known in advance. The adaptive filter uses feedback in the form of an error signal to refine its transfer function to match the changing parameters.

Generally speaking, the adaptive process involves the use of a cost function, which is a criterion for optimum performance of the filter, to feed an algorithm, which determines how to modify filter transfer function to minimize the cost on the next iteration.

As the power of digital signal processors has increased, adaptive filters have become much more common and are now routinely used in devices such as mobile phones and other communication devices, camcorders and digital cameras, and medical monitoring equipment.

LMS ALGORITHM - The Least Mean Square (LMS) algorithm, introduced by Widrow and Hoff in 1959 is an adaptive algorithm, which uses a gradient-based method of steepest decent . LMS algorithm uses the estimates of the gradient vector from the available data. LMS incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which eventually leads to the minimum mean square error. Compared to other algorithms LMS algorithm is relatively simple; it does not require correlation function calculation nor does it require matrix inversions.



MATLAB CODE FOR ADAPTIVE FILTER BY LMS ALGORITHM

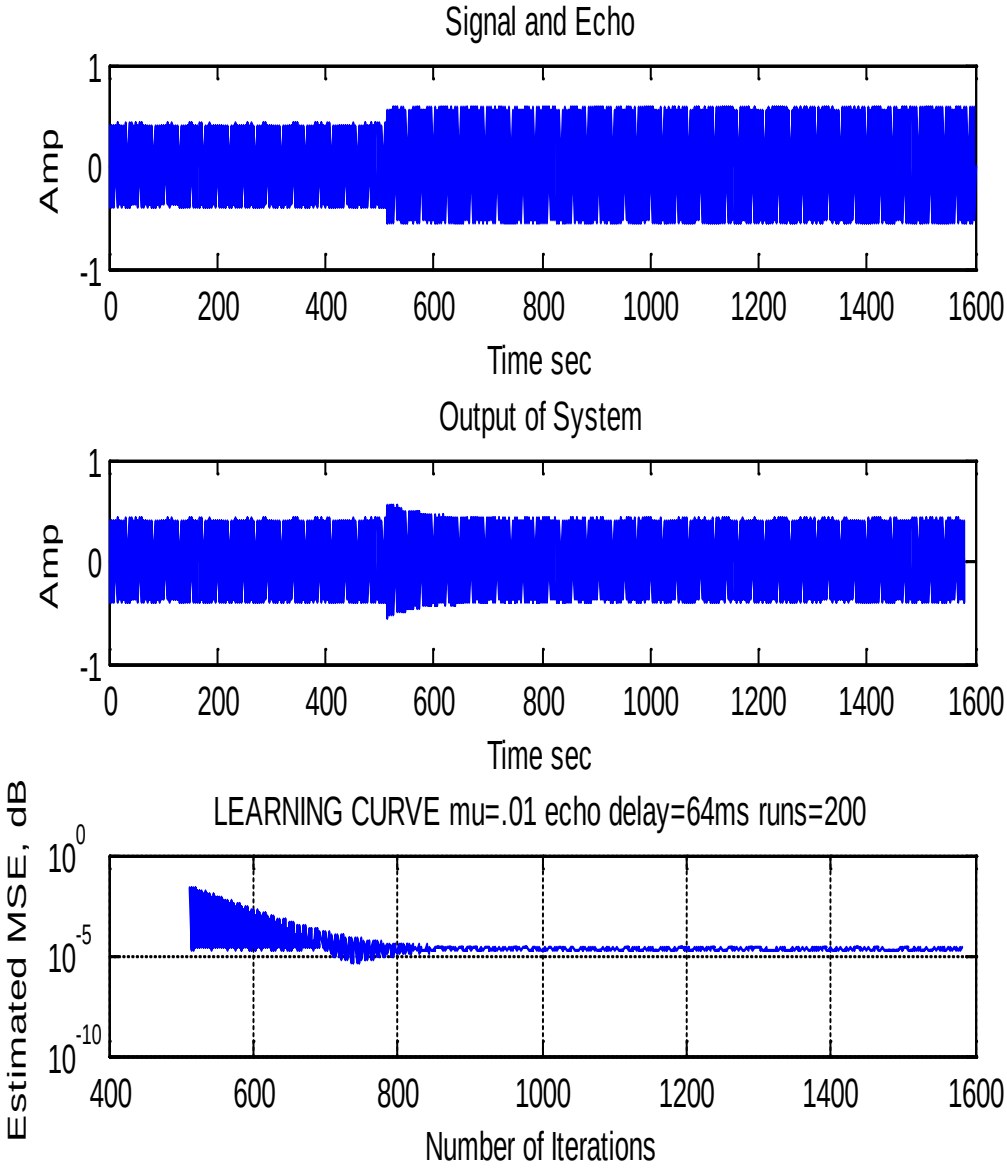
```
clear all;
clc;
clf;
mule = .01;             % Larger values for fast conv
max_run = 200;

for run=1:max_run;
taps = 20;              %Adaptive Filter Taps #
freq = 2000;            %Signal Freq
w = zeros(1,taps);      %state of adaptive filter
time = .2;              %lenght of simulation (sec)
samplerate = 8000;      %samples/sec
samples = time*samplerate;
max_iterations = samples-taps+1;
iterations = 1:max_iterations;%Vector of iterations
t=1/samplerate:1/samplerate:time;
rand('state',sum(100*clock));%Reset Randome Generator

noise=.02*rand(1,samples);%noise added to signal
s=.4*sin(2*pi*freq*t);%Pure Signal

x=noise+s;%input to adaptive filter
echo_amp_per = .4; %Echo percent of signal

%rand('state',sum(100*clock));%Reset Randome Generator
echo_time_delay = .064;
echo_delay=echo_time_delay*samplerate;
echo = echo_amp_per*[zeros(1,echo_delay) x(echo_delay+1:samples)];

%LMS
```

```
for i=1:max_iterations;

   y(i)=w*x(i:i+taps-1)';
   e(run,i)=echo(i)-y(i);

   %mule(i) = .5/(x(i:i+taps-1)*x(i:i+taps-1)'+ .01);

   w = w + 2*mule*e(run,i)*x(i:i+taps-1);

end


end

%%Mean Square Error

mse=sum(e.^2,1)/max_run;

b=x+echo;

%Ouput of System

out=b(1:length(y))-y;

subplot(3,1,1),plot(b);

title('Signal and Echo');

ylabel('Amp');

xlabel('Time sec');

subplot(3,1,2),plot(out);

title('Output of System');

ylabel('Amp');

xlabel('Time sec');

subplot(3,1,3),semilogy(mse);

grid

title('LEARNING CURVE mu=.01 echo delay=64ms runs=200');

ylabel('Estimated MSE, dB');

xlabel('Number of Iterations');
```

EXPLANATION TO THE ABOVE MATLAB CODE

- Clear the command window.
- Define the variables: mule=0.01 (larger values for fast convolution) and max_run=200.
- Start a 'for' loop : run =1:max_run;
- Define taps=20; freq=2000 (signal frequency);w=0(1:taps);time=0.2(length of simulation);samplerate=8000 (samples/sec).
- Calculate the number of samples using: samples=time*samplerate;
- Now, maximum iteration limit: max_iterations=sample-taps+1;
- Vector of iteration: iteration=1:max_iterations and t=1/samplerate:1/samplerate:time.
- Reset Randome Generator using rand() function.
- Set noise= 0.2 * rand(1,samples); (Noise added to the signal)
- Pure Signal: s=0.4*sin(2*pi*freq*t);
- Input to adaptive filter : x=noise+s;
- ECHO percent of the signal : echo_amp_per=0.4;
- echo_time_delay=0.064;
- Define : echo_delay=echo_time_delay*samplerate;
- Calculate ECHO: echo=echo_amp_per*[zeros(1,echo_delay) x[echo_delay +1 : samples)]
- Implement LMS Algorithm.
- Find out the Mean Square Error
- Get the desired output.
- Analyze the plot.

The Output of the above code is as follows:



MATLAB CODE FOR HYBRID ECHO CANCELLATION

```
clear all;
clc;
clf;
close all;

%*************************************************************************
% Initialize all the values for the block implementation
N = 64;
L = 16;
P = L;
M = 32;
R = 10;
B = N+M-L;
%*************************************************************************

% Find the FFT and IFFT matrices
F = fft(eye(M)); % MxM matrix
figure(1);plot(F);
figure(2);plot(abs(F));

Finv = ifft(eye(M)); % MxM matrix
figure(3);plot(Finv);
figure(4);plot(abs(Finv));

G1 = [zeros(M-L,M-L) zeros(M-L,L)
zeros(L,M-L) eye(L,L)];
figure(5);plot(G1);
%*************************************************************************

% Intialize the constarined or unconstarined condition
G2 = F*[eye(P) zeros(P,M-P)
zeros(M-P,P) zeros(M-P,M-P)]*Finv; % size = MxM , constrained
figure(6); plot(G2);
%*************************************************************************

% Intialize the coefficients vector
K = 5; % Gain factor
w(1:N,1) = K*[ones(N,1)]; % Nx1 vector , intialize the coefficients vector

for (p = 0 : (N/P)-1)
W_p(1:M,p+1) = F*[w((p*P)+1 : (p+1)*(P),1) % Mx1 vector , find the frequency
domain equivalent
zeros(M-P,1)]; % Split into N/P parts , length is Px1 and zero pad with (M-
P)x1 zeros
end % Size of W_p = Mx1

figure(7);plot(W_p);
%*************************************************************************

% Time sampling for collecting data
% %noise signal
% noise = .2*randn(1,length(t));
```

```
% x1 = randn(1,length(t));
% b = fir1(63,[.35 .65]);
% %sample original signal x
% f1 = filter(b,1,x1);

[x1,fs] = wavread('C:\signalplusnoise.wav',64);


[s] = wavread('C:\noise.wav',64);
figure(9);plot(s);
t = 0:length(x1);

%assume additive white gaussian noise and add to s
[xw] = 10*[x1];
k = 5;
figure(8);plot(xw);
%here s = sin(3*t)
%s = k*(10*sin(.5*t) + 20*cos(5*t));
% d = xw + s
d = xw + s ;
figure(19);plot(d);
%************************************************************************

% Taking data , size = the lowest power of 2 available data
TB = floor(log2(length(t)));
TBL = 2^TB;
%************************************************************************

%plot x d and s to be able to see them together
figure(10);
plot(t(1:TBL),x1(1:TBL));
hold on;
plot(t(1:TBL),d(1:TBL),'r-');
xlabel ('NUMBER OF SAMPLES');
ylabel('AMPLITUDE');
axis tight;
plot(t(1:TBL),s(1:TBL),'g+');
legend('ORIGINAL SIGNAL, x','d = s + (w*x)','NEW SIGNAL, s');
%close ;
%plot noise
figure(11);
plot(xw(1:TBL));
axis tight;
title('SIGNALS FROM LOUDSPEAKER');
xlabel ('NUMBER OF SAMPLES');
ylabel('AMPLITUDE');
%close ;

%************************************************************************
x = [zeros(1,(N + M - 2*L))' x1]; % Zero pad with N + M - 2*L zeros
%************************************************************************

for a = 1:TBL/L
dd(1:L,1) = d(((a-1)*L + 1):a*L)'; % L samples of desired element
d_l(1:M,1) = [zeros(M-L,1)
```

```
dd ];% Desired data matric Mx1 matrix , fixed for the
end
figure(12);plot(dd);
xhold(1:B,1) = [x(((a-1)*L +1) : ((a-1)*L + B))]' ; % every L sample (N+M-
L)x1
figure(13);plot(xhold);

for p = 0:1:((N/P)-1)
x_p(1:M,p+1) = xhold((B - p*P - M +1):(B - p*P),1); % take the first L
samples and the previous N samples
X_p(1:M,p+1) = F*x_p(1:M,p+1); % create blocks of length M starting from the
latest sample
% and going backwards.
end
figure(14);plot(x_p);
figure(15);plot(X_p);
figure(16);plot(abs(X_p));

%Del = calculate_step_size(X_p,M,N/P,type);
Del = size(X_p,M);

for r = 1:R % Start the recursion
if r == 1
W_p_r(1:M,1:(N/P),r) = W_p(1:M,1:(N/P)); % store the filter weights in a new
array indexed
end % for including the R iterations
end
figure(17);plot(abs(W_p_r));

temp = 0; % Initialize for summation

for p = 0:1:((N/P)-1)
temp = temp + [((diag(X_p(1:M,p+1)))*W_p_r(1:M,p+1))]; % Find the summation
of X_p*w_p
end
% figure(17);plot(temp);

y(1:M,r) = real(G1*Finv*temp)./P; % estimate of
figure(18);plot(y);
```

FIGURE1: Fast Fourier Transform(fft) of 32X32 identity matrix.(F)

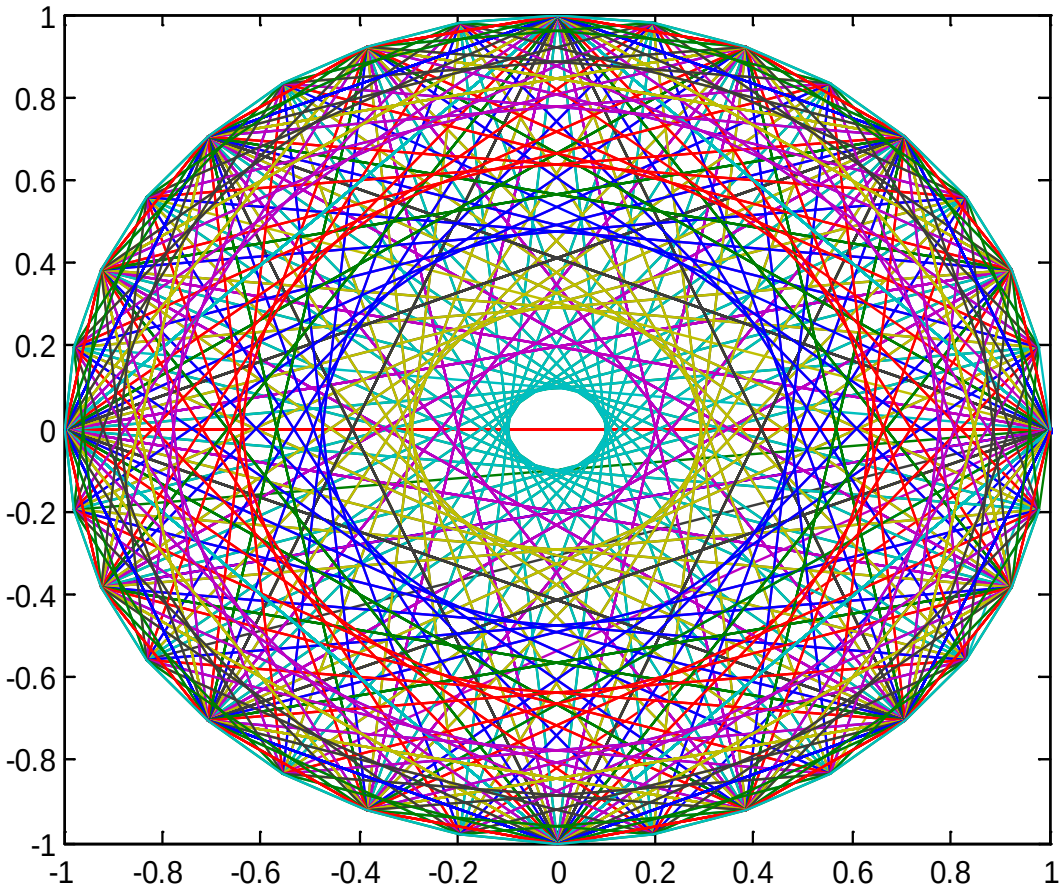


FIGURE2: Absolute value of fft of 32X32 identity matrix.

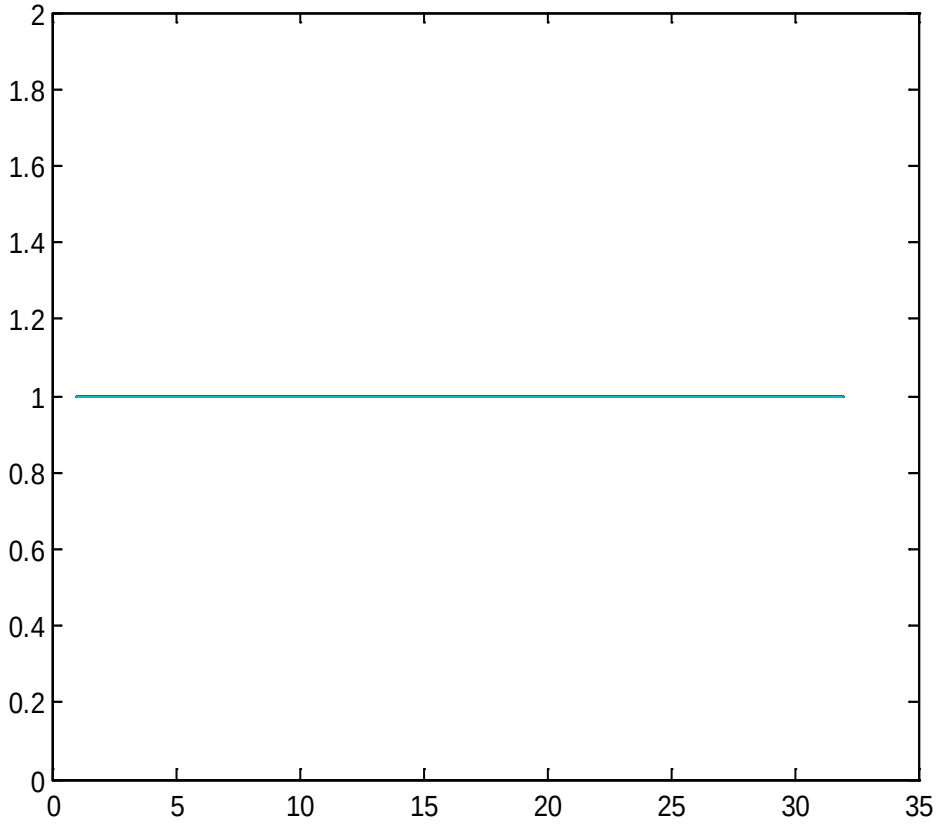


FIGURE3: Inverse fft of 32X32 identity matrix.(Finv)

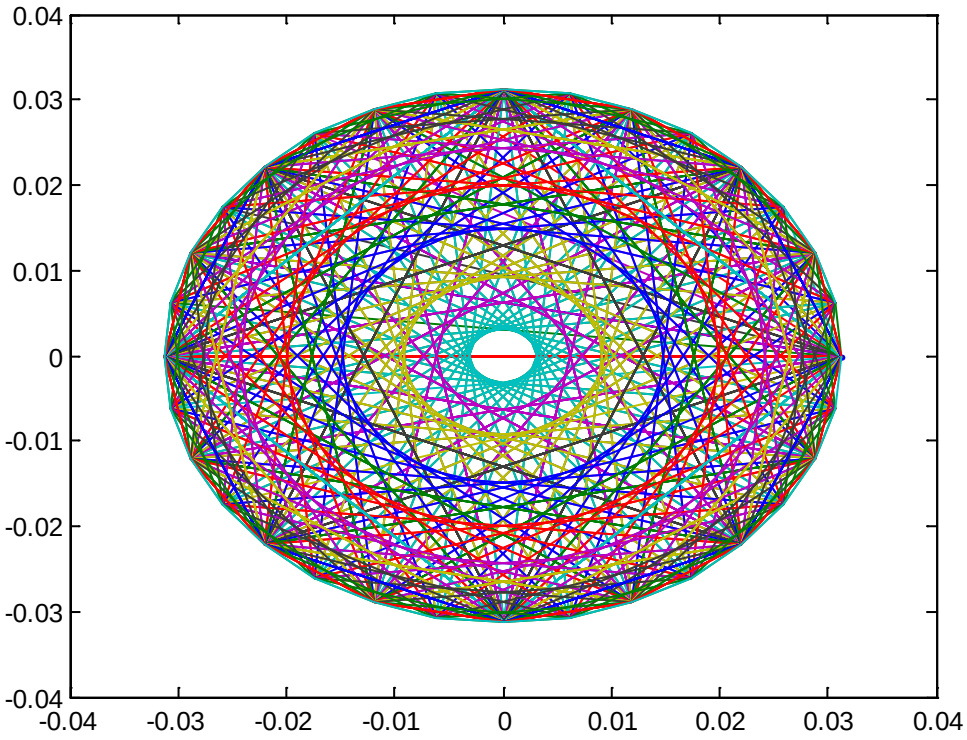


FIGURE4: Absolute value of ifft of 32X32 identity matrix.

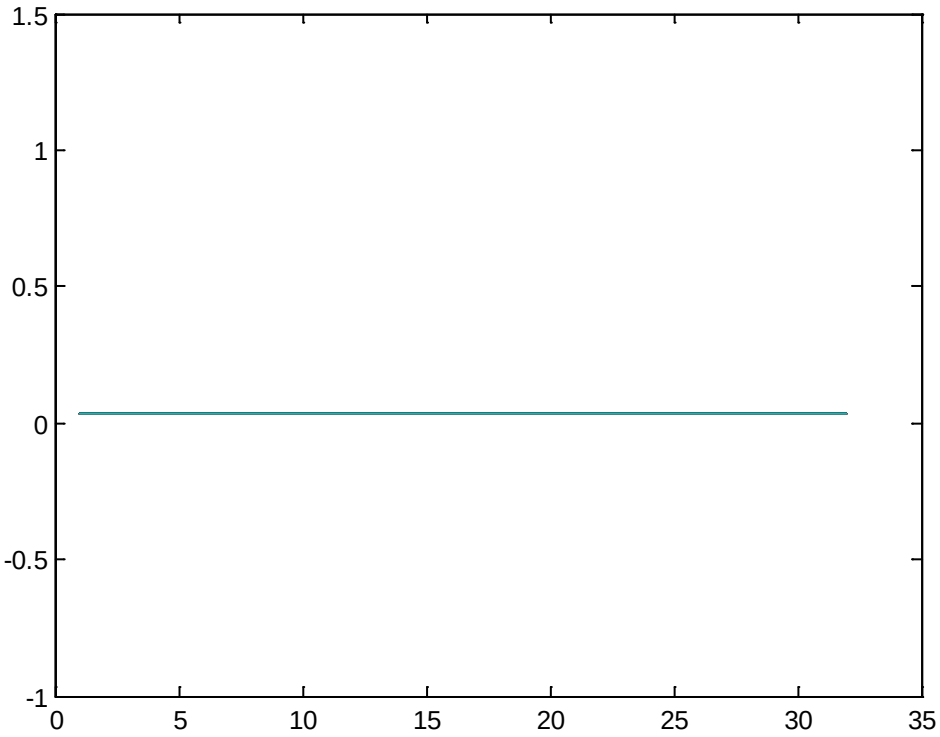


FIGURE5: Bottom right matrix 16X16 is an identity matrix. it is a sub matrix of 32X32 matrix , else all values are zero.(G1)

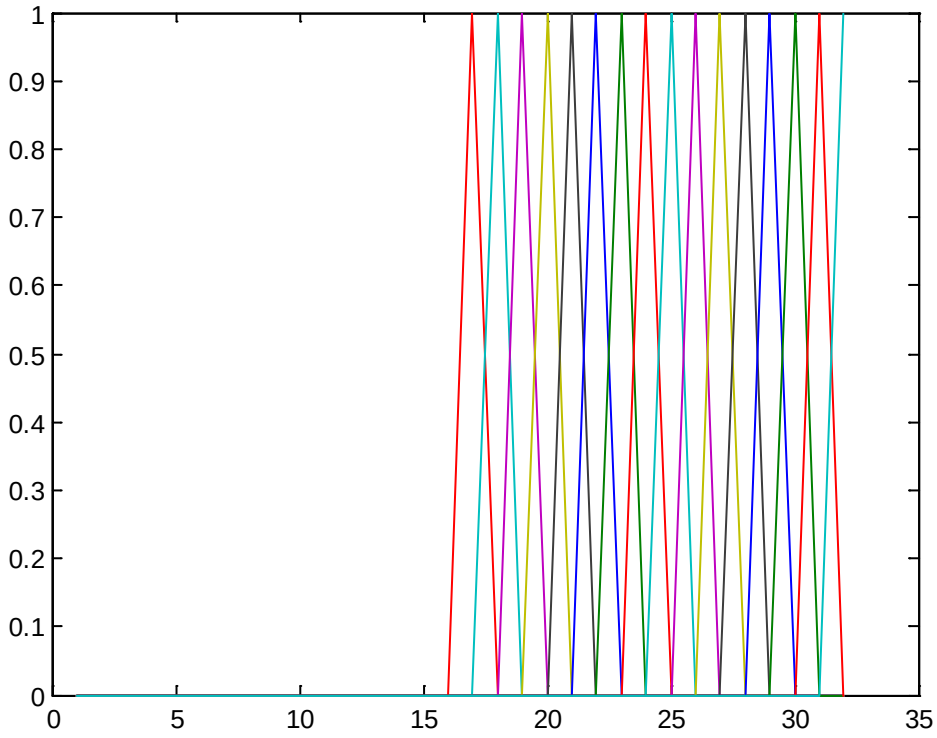


FIGURE6: F multiplied to (Top left matrix 16X16 is an identity matrix. it is a sub matrix of 32X32 matrix , else all values are zero.) and Finv : (G2)

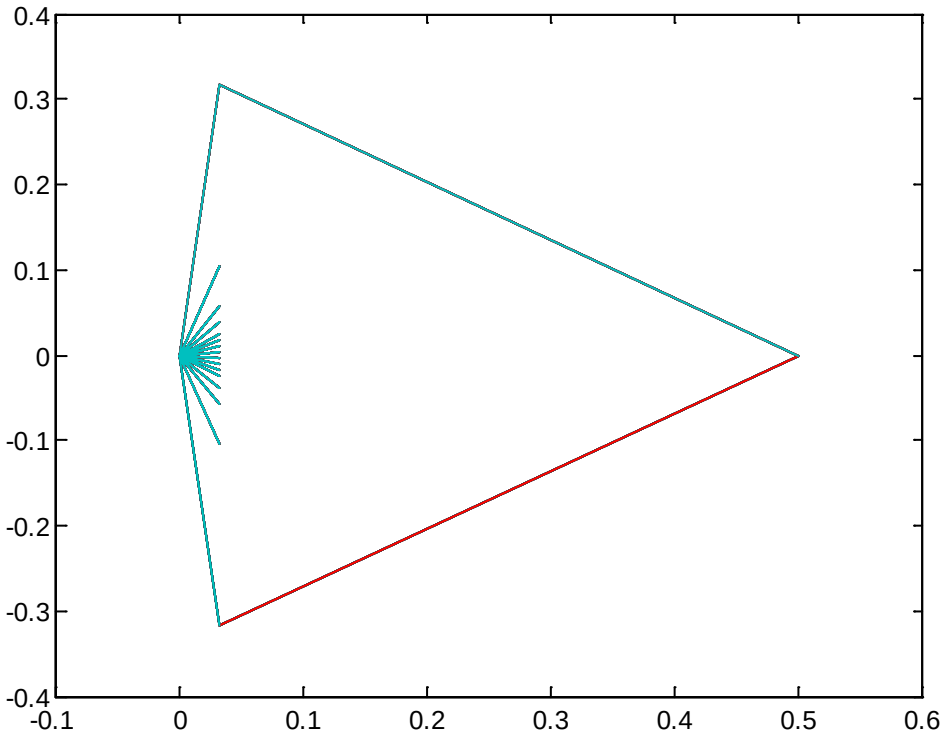


FIGURE7: Frequency Domain Equivalent : (w_p)

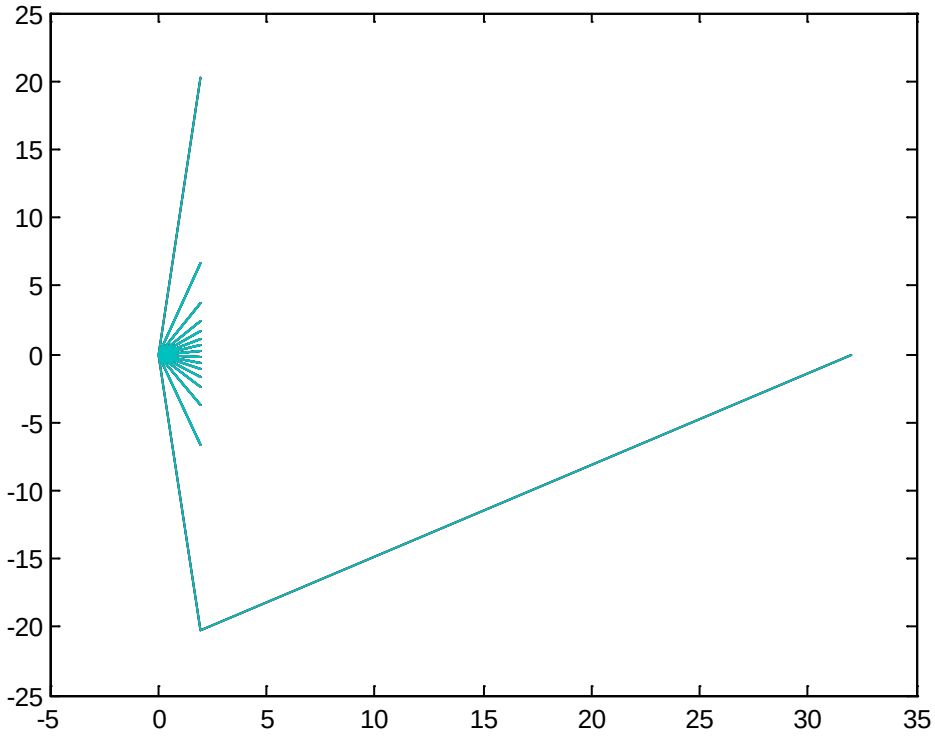


FIGURE8: Original signal ‘noise.wav’ : (s)

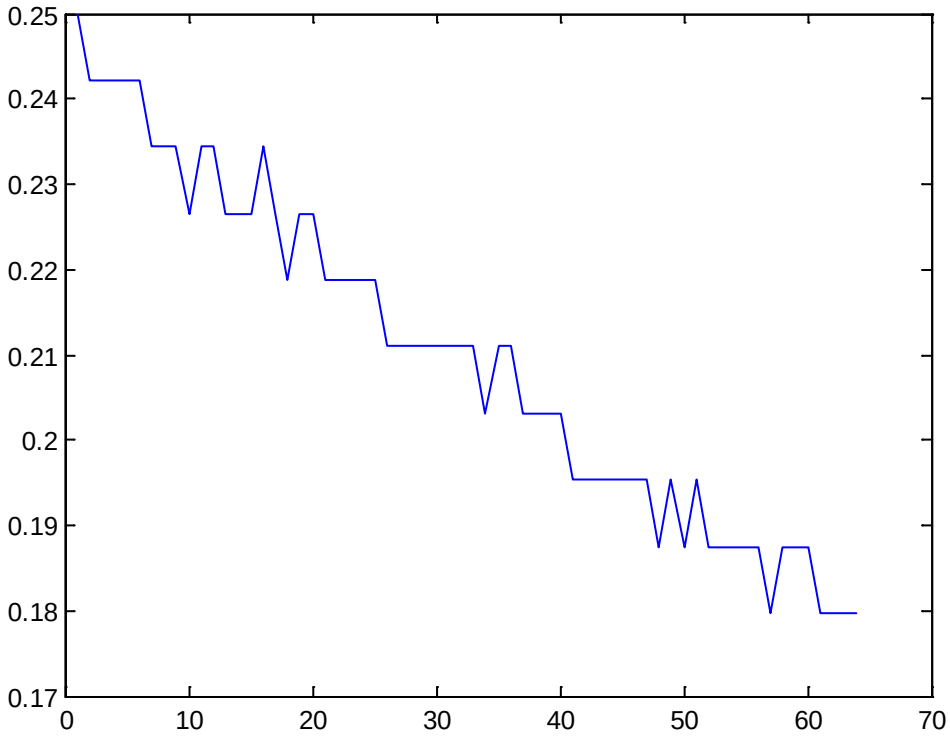


FIGURE9: Additive white Gaussian noise : ‘signalplusnoise’ (xw)

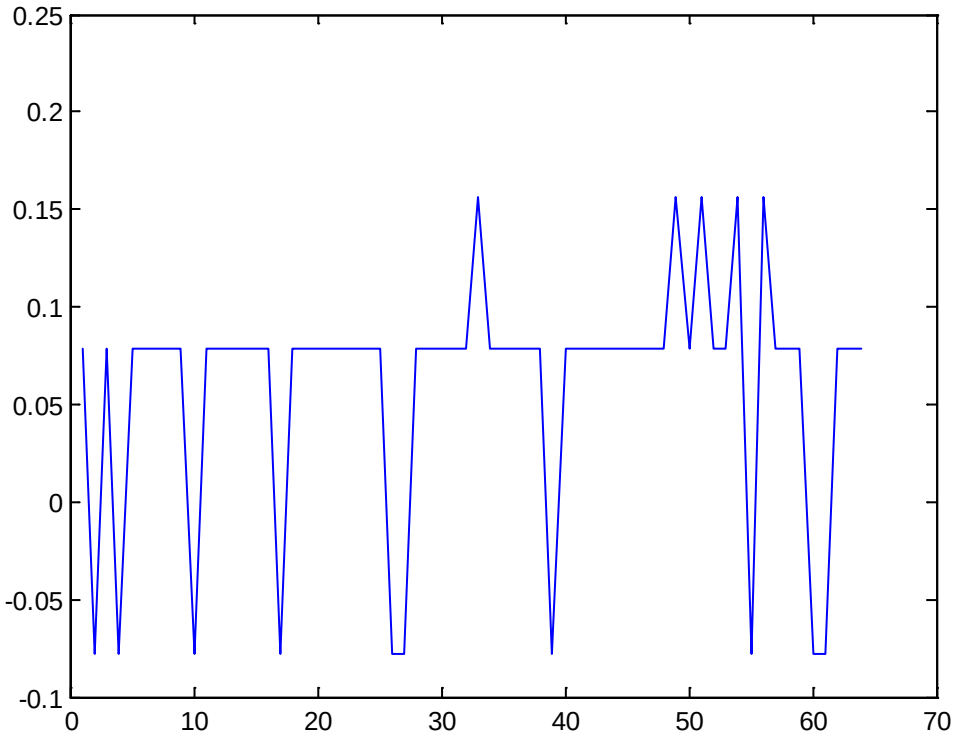


FIGURE10: concatenated signals : d=xw+s

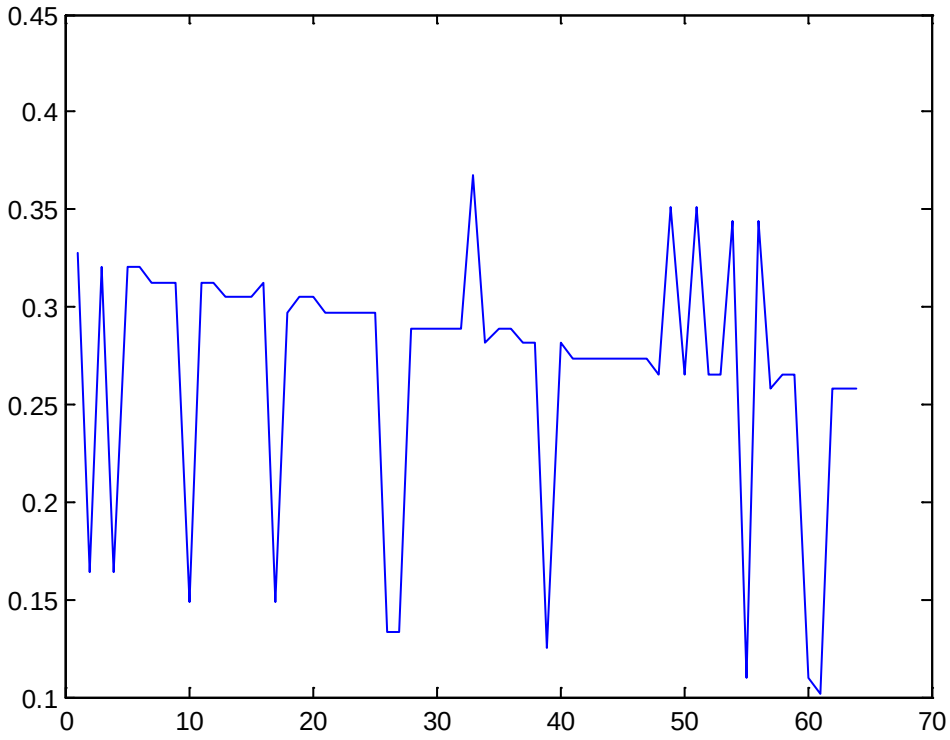


FIGURE11: signal in blue is original signal.

signal in red is concatenated signal.

signal in green is new signal.

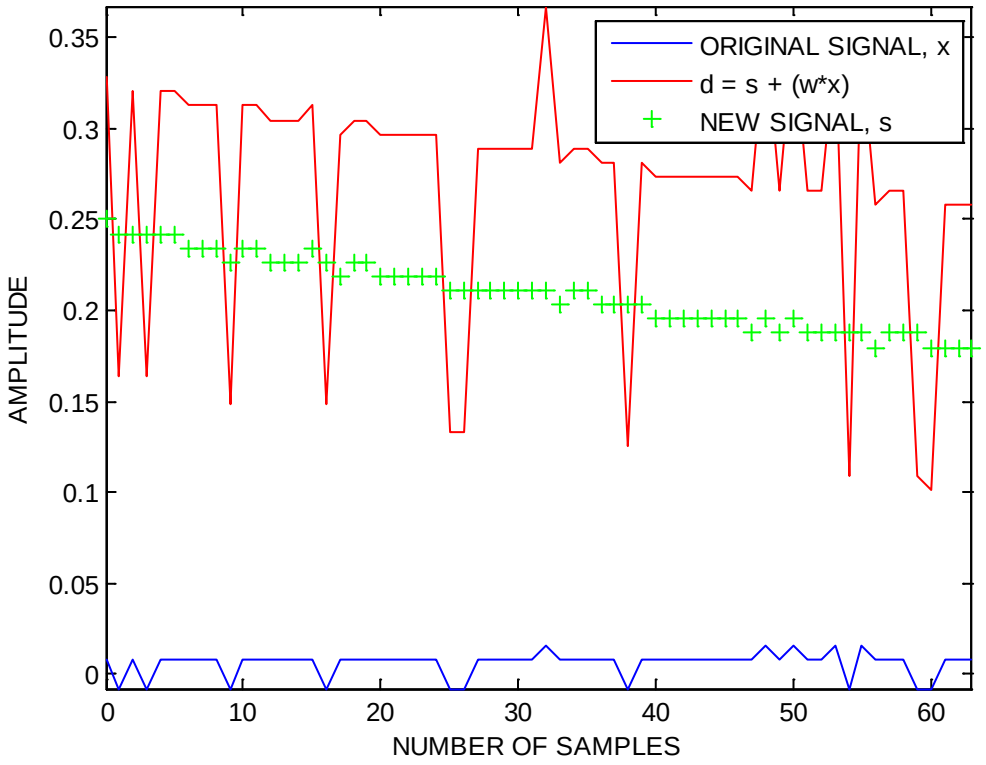


FIGURE12: plot of xw

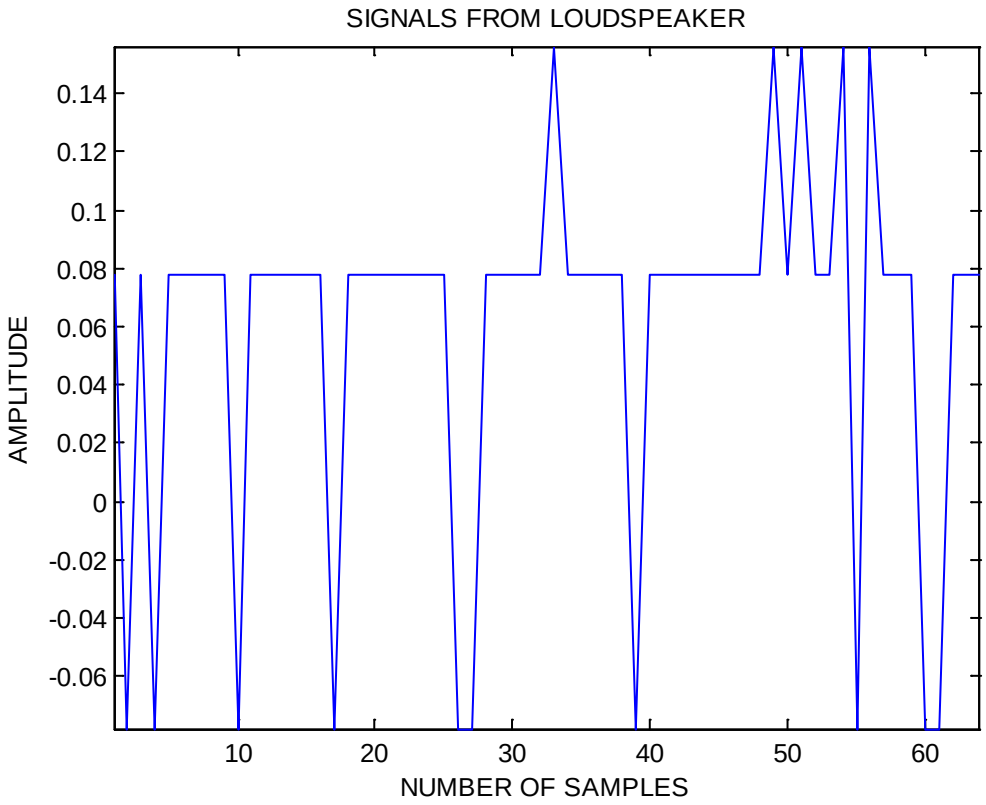


FIGURE13:

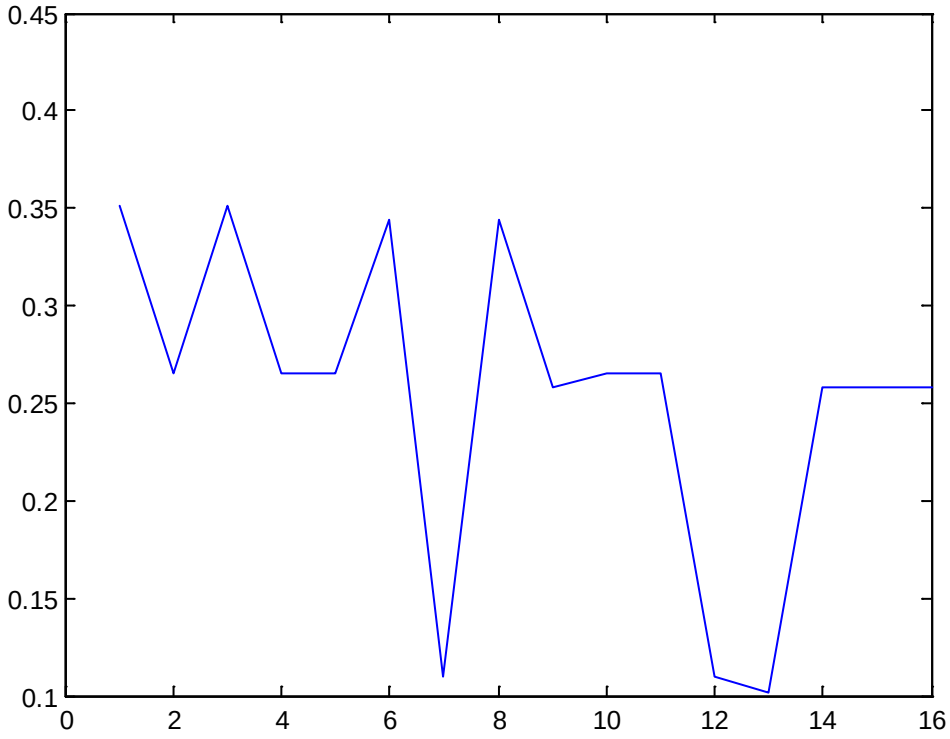


FIGURE14:

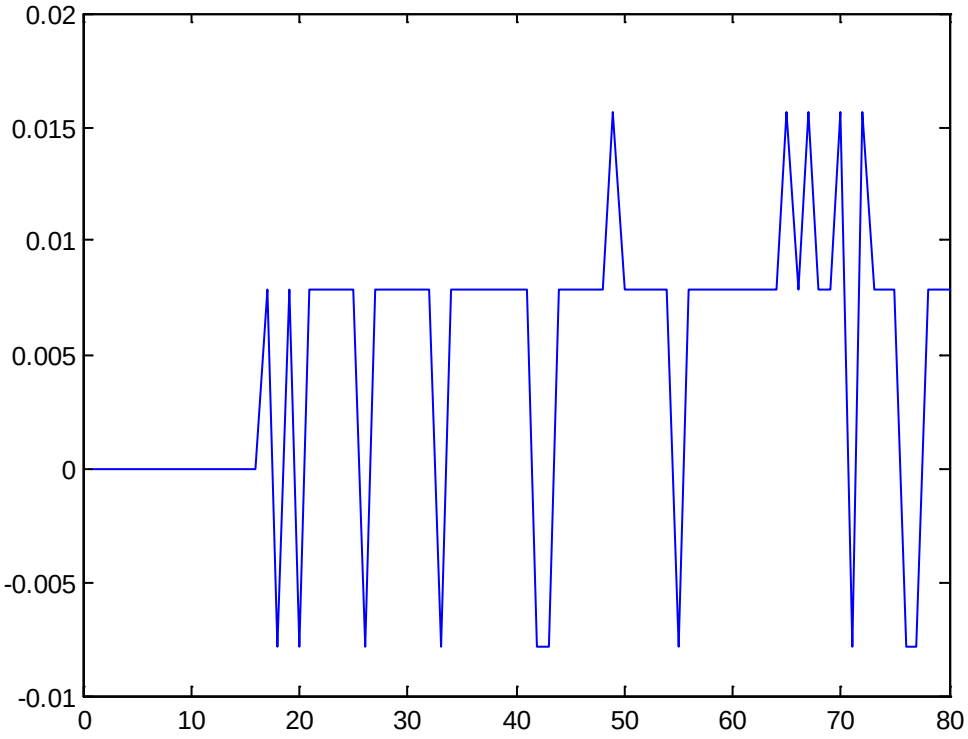


FIGURE15: plot(x_p)

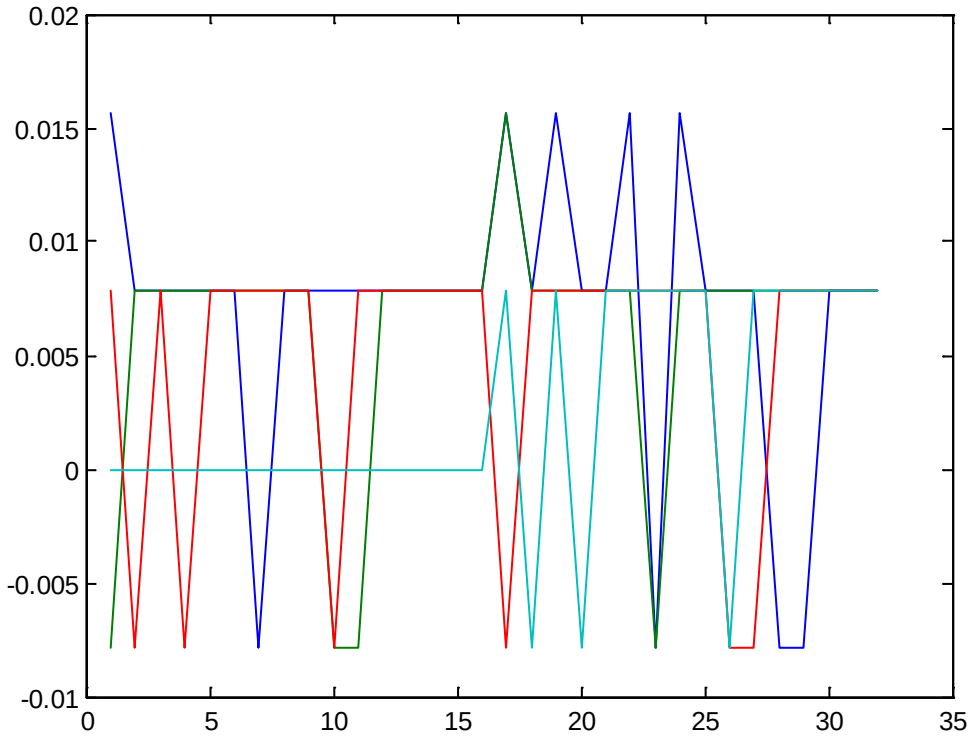


FIGURE16: plot (X_p)

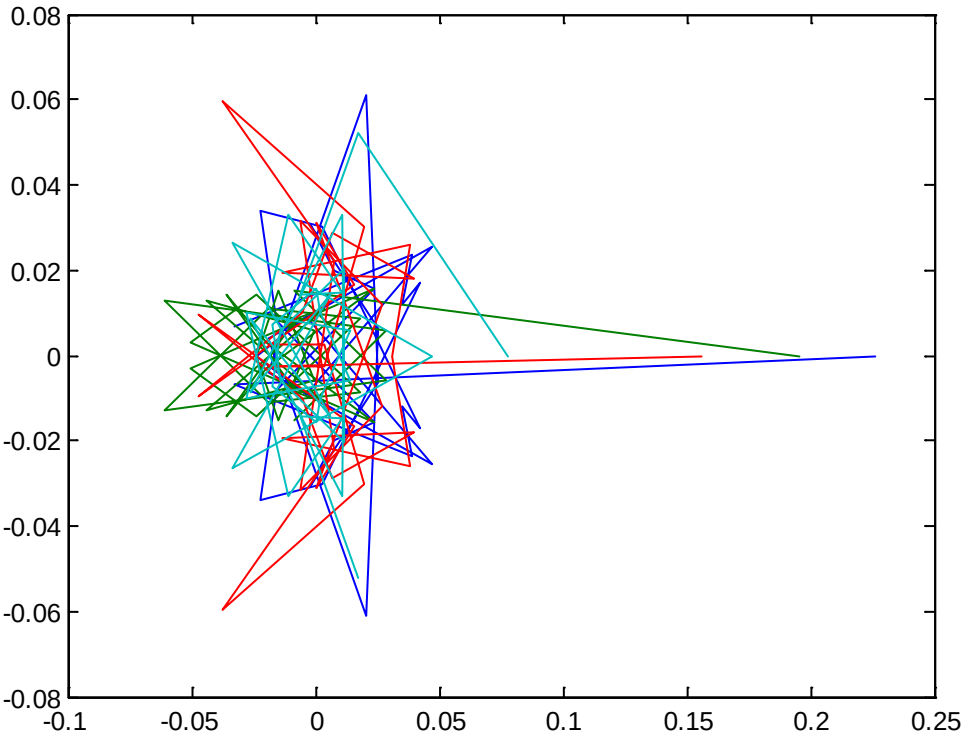


FIGURE 17: plot (abs(X_p))

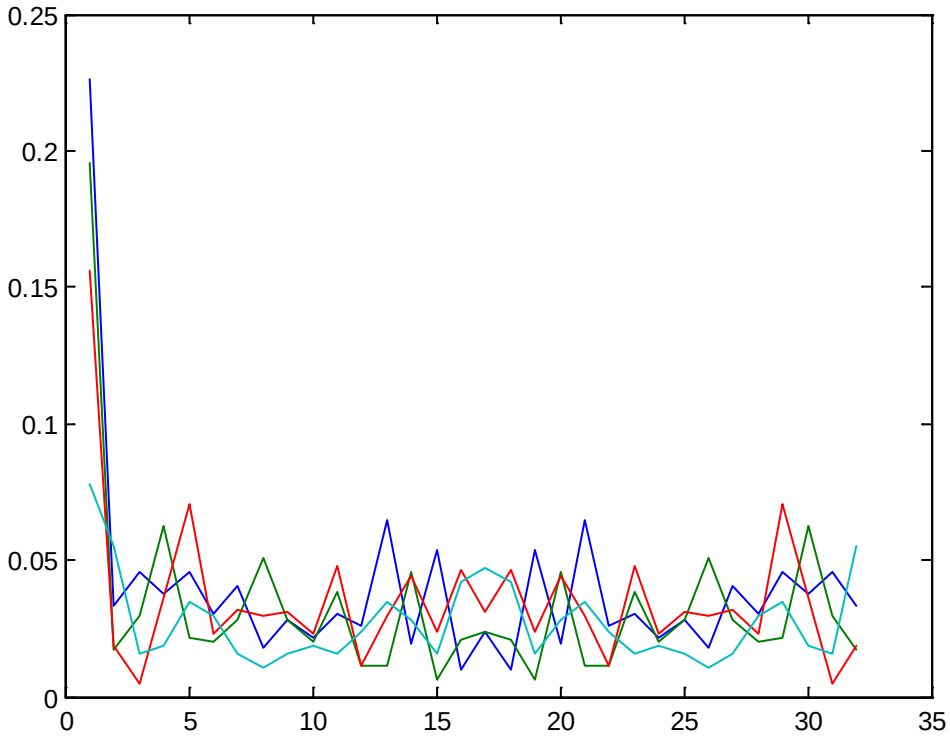


FIGURE18: plot(abs(W_p_r))

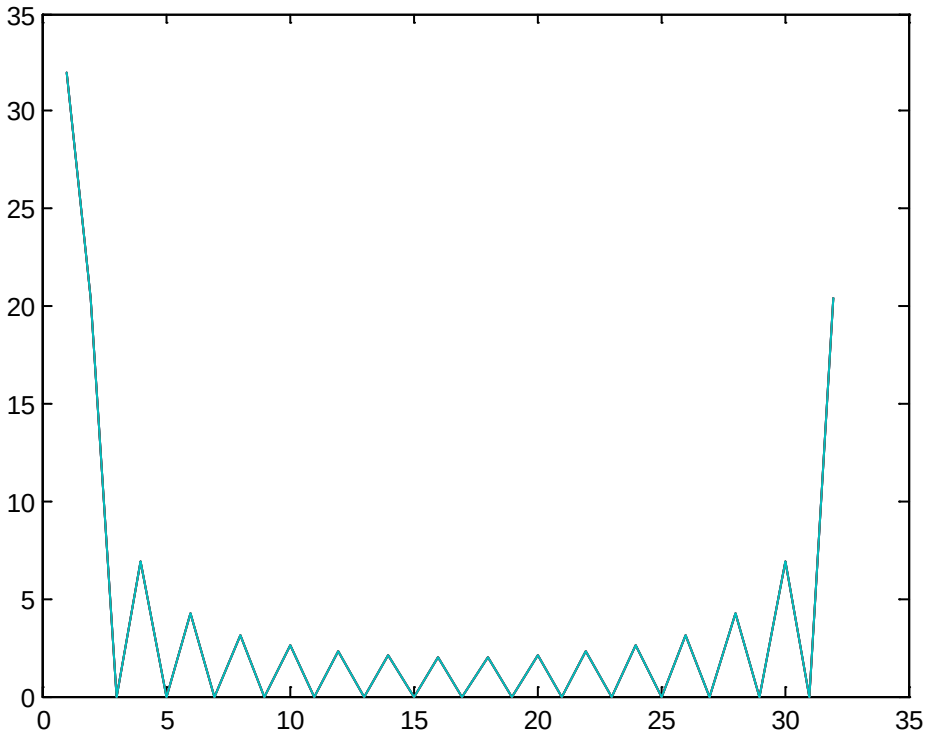
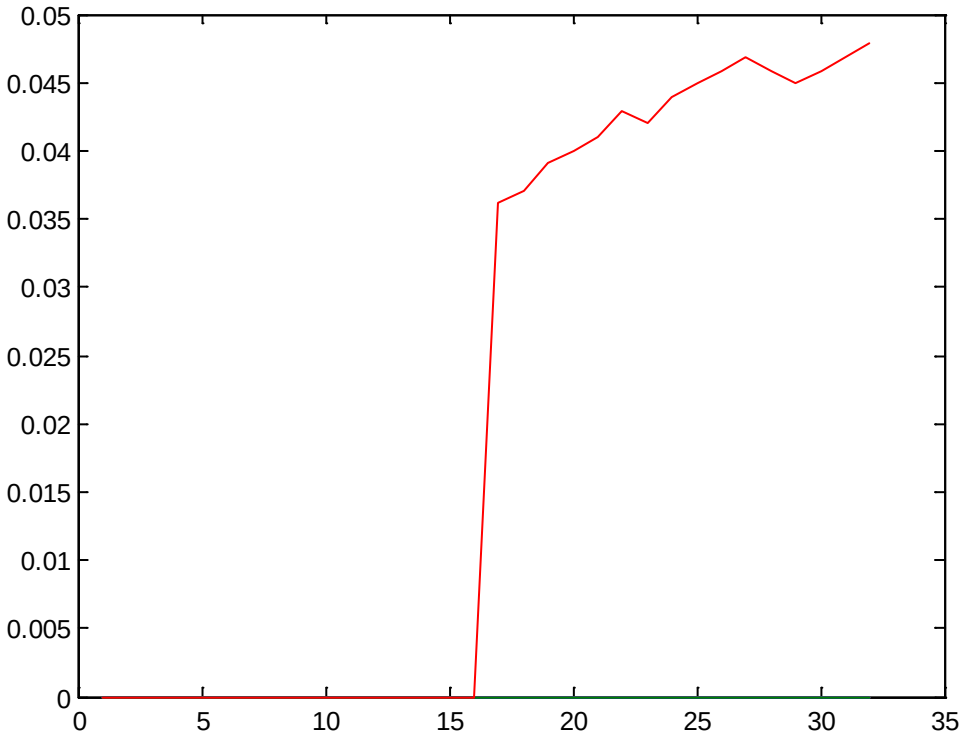


FIGURE19: plot (Y)



SUBMITTED BY:

NEHA GOYAL (Y09UC255)

SUMIT GAUTAM(Y09UC285)