Department of Electronics Engineering

Lab Manual

Signal .Processing and Communication.) for M.Tech lab (ECC 507)

Academic Complex, 2nd Floor, Room No.216

INDIAN INSTITUTE OF TECHNOLOGY (Indian School of Mines), DHANBAD

JHARKHAND-826004

Table of Contents

<u>Expt 1(a) : Design of Filters with Fixed Coefficients</u>

1. ***Write a program to generate a square wave of 1 kHz and to pass it through an FIR filter designed to cut off frequencies above 2 kHz. Comment on the results, referring to both time-domain and frequency-domain characteristics of the input as well as the output obtained from the filter designed by you.***

   <u>Program</u>

```
clear all
close all

fs=20000;
t=0:1/fs:0.005;
x= 2*square(2*pi*1000*t);
plot(t,x)
figure;
freqz(x,1,512);

M= max(size(x));
N=61;
N1=(N-1)/2;
n=0:N-1;
hd=0.2*sinc(0.2*(n-N1));
w_ham=hamming(N);
h=hd.*w_ham';
figure;
freqz(h,1,512);

y=filter(h,1,x);
figure;
freqz(y,1,512);

figure;
plot(t,y);
```

Expt 1(b):Study of Adaptive filters (with application in system identification)

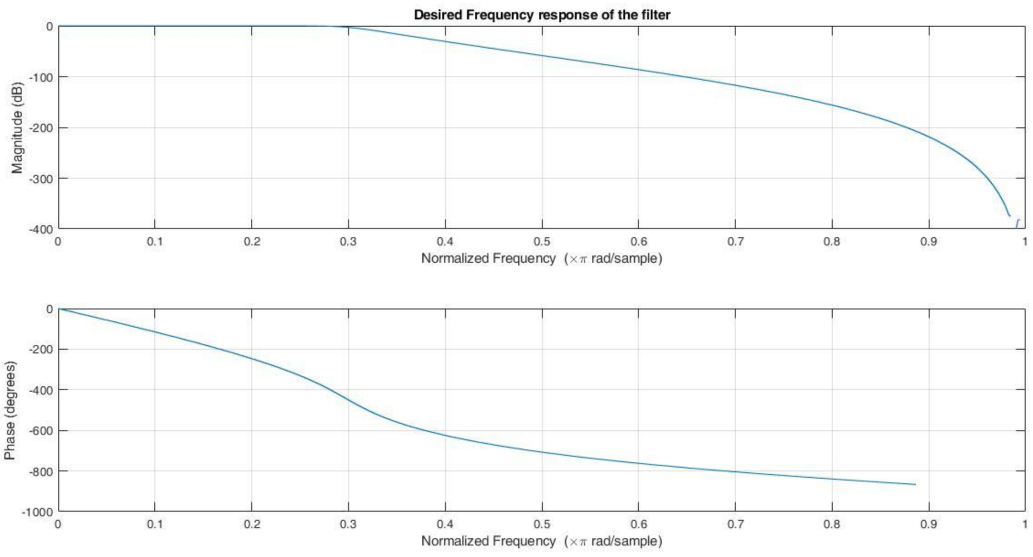
1. ***Work with the program new_sys_ident.m (this is a program for system identification using Adaptive filtering). Verify that the mean square error decreases with time. Also plot the desired signal and adaptive filter output before and after the convergence has occurred.***

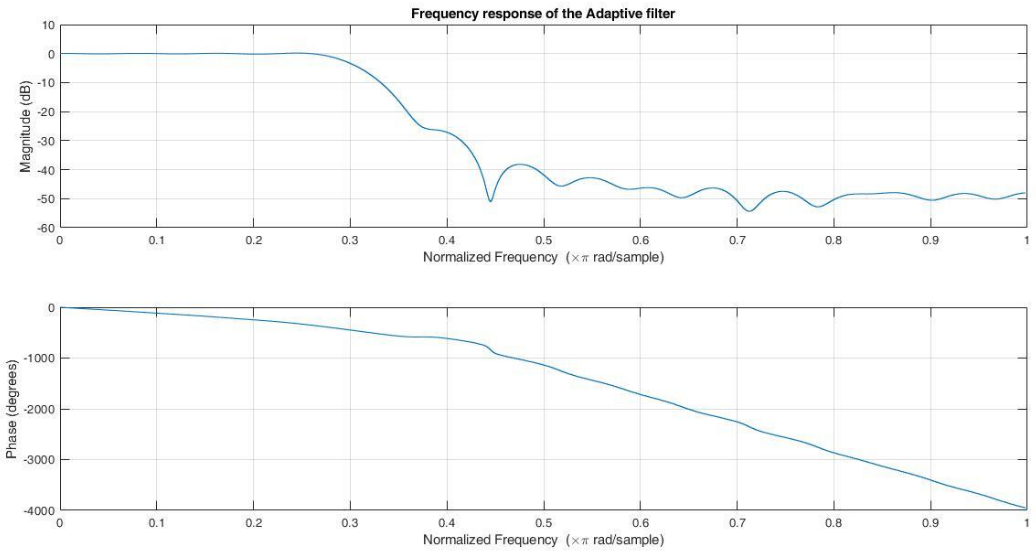
Program

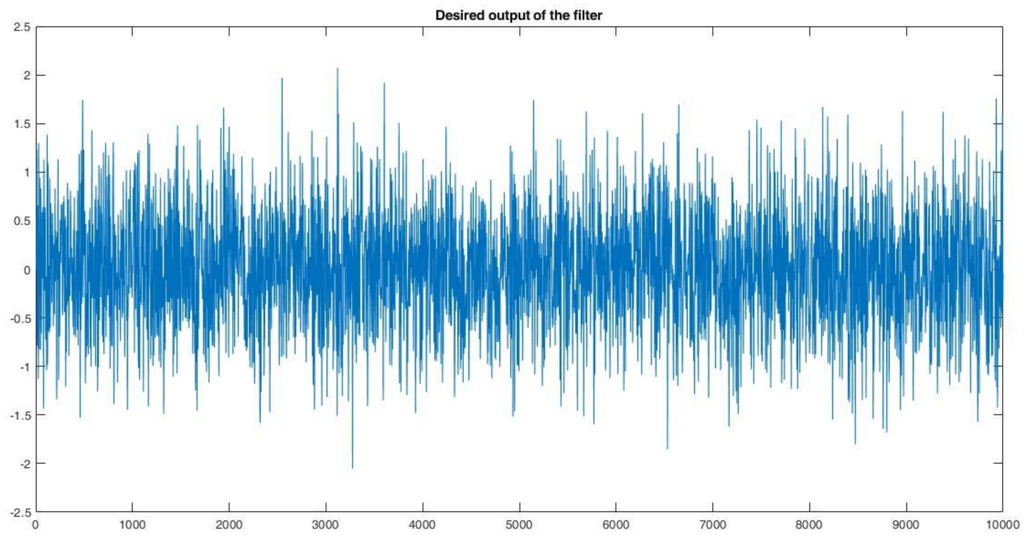
```
    clear all
close all
    % For sinusoidal input
%n=1:10000;
%x=sin(0.4*pi*n);
%x=x';

x = randn(10000,1);%random input signal
    %[B,A] = ellip(4,0.25,10,0.25);%IIR filter

%x=[1 zeros(1,9999)]';  % Input is a unit sample sequence
    [B,A]=butter(10,0.3);

d = filter(B,A,x);%reference signal
    figure;freqz(B,A)%view frequency response
title('Desired Frequency response of the filter');

    N=32;% length of the filter
    M=length(x);%length of the input signal
    y=zeros(1,M);%intializing the response of the adaptive filter
    h=zeros(1,N);%intializing the filter coefficents to zero
    e(1:N-1)=x(1:N-1);%intializing the error array
    % calculating the value of step size
    x2=0;
    for t=1:M
  x2=x2+x(t).^2;
    end
    de=(M+1)/(10*N*x2)%de is the value of step size
    % the lms algorithm calculates the value of output array and error function
    for n =N:M
  x1= x(n:-1:n-N+1);
  y(n)=h*x1;
  e(n)=d(n)-y(n);
  h=h+de*e(n)*x1';
```

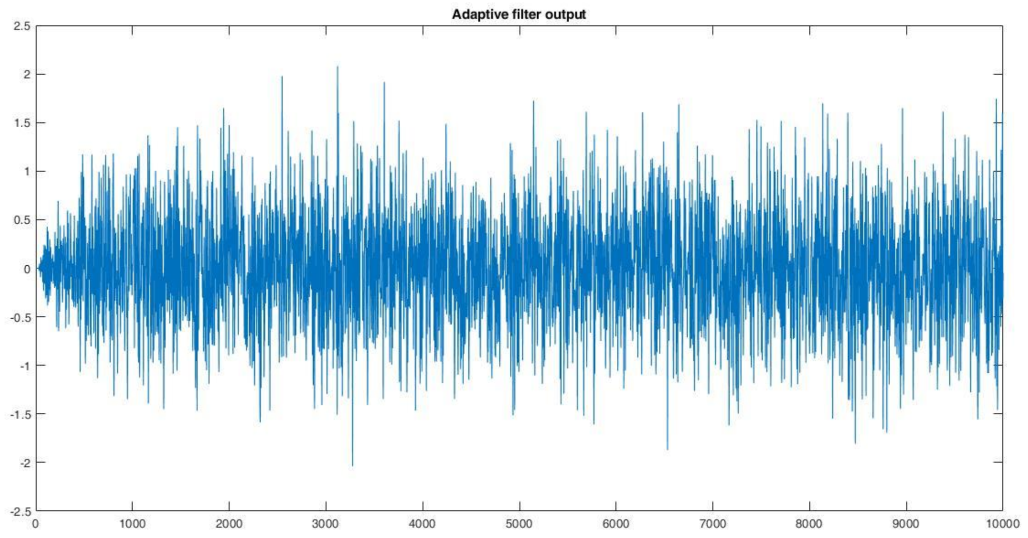
```
end



figure;
freqz(h,1,512);
title('Frequency response of the Adaptive filter');

figure;
plot(d);
title('Desired output of the filter');

figure;
plot(y);
title('Adaptive filter output');

% Testing for error in o/p for Unit sample sequence
%err_impuls=d-y';


%the mean square error is calculated in the following module
      K=10*N
L=fix(M/K);
      for m=1:L-1
   se(m)=0;
   ase(m)=0;
   for t=(m-1)*K+1:m*K
      se(m)=se(m)+e(t).^2;
   end
   ase(m)=se(m)/K;
end

figure;
m1=1:L-1;
stem(m1,ase);
title('Mean square error vs. frame number');

% Plotting the outputs before convergence has occurred. It is seen from
% plot of 'ase' that error is negligible after m=5. Hence the 2nd frame
% is only plotted
figure;
plot(d(0*K+1:0*K+K));
title('Desired output in 1st frame');

figure;
plot(y(0*K+1:0*K+K));
title('Adaptive filter output in 1st frame');

% Plotting the outputs after convergence has occurred. It is seen from
% plot of 'ase' that error is negligible after m=5. Hence the 5th frame
% is only plotted


figure;
plot(d(5*K+1:5*K+K));
title('Desired output in 6th frame');

figure;
plot(y(5*K+1:5*K+K));
title('Adaptive filter output in 6th frame');
```

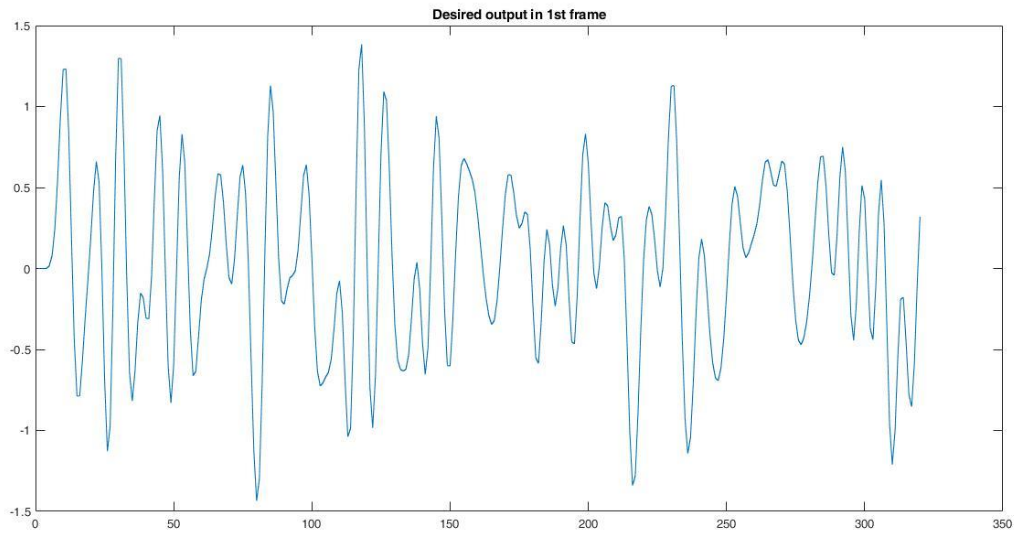
<u>Results</u>

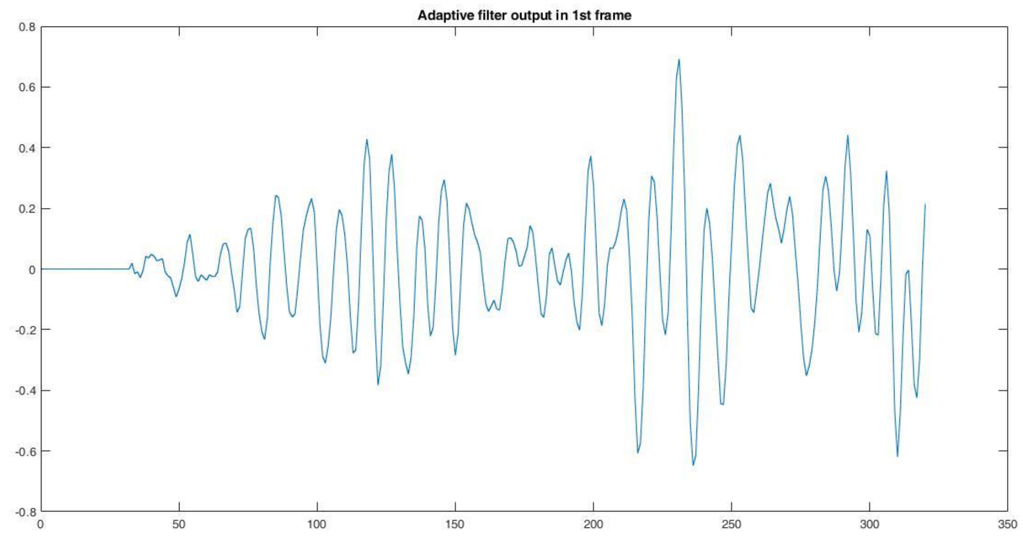


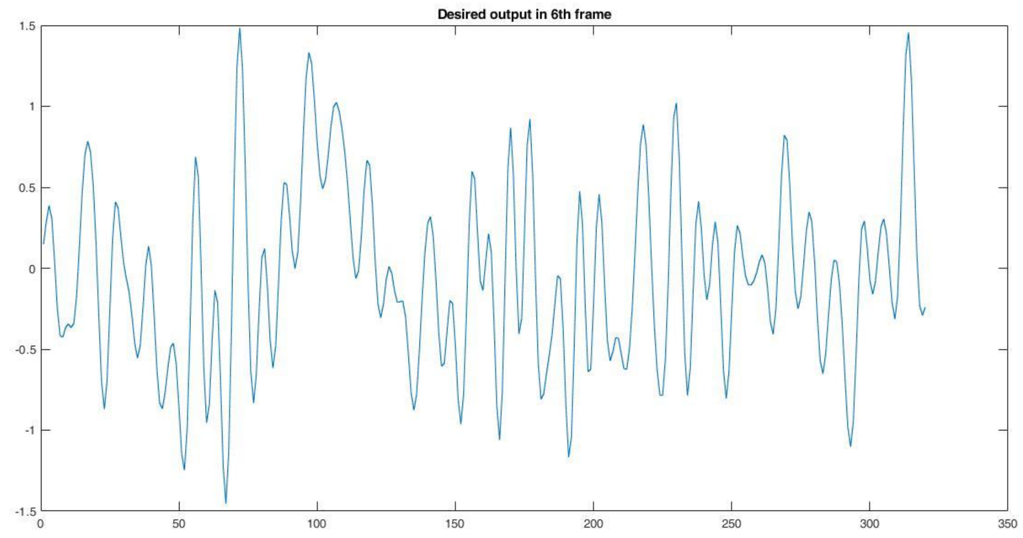


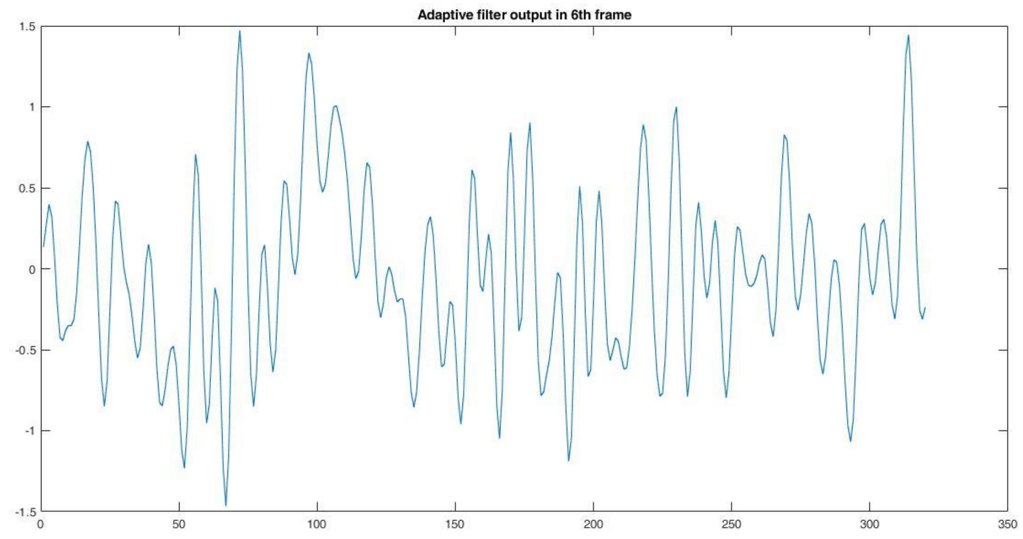

Desired output of the filter

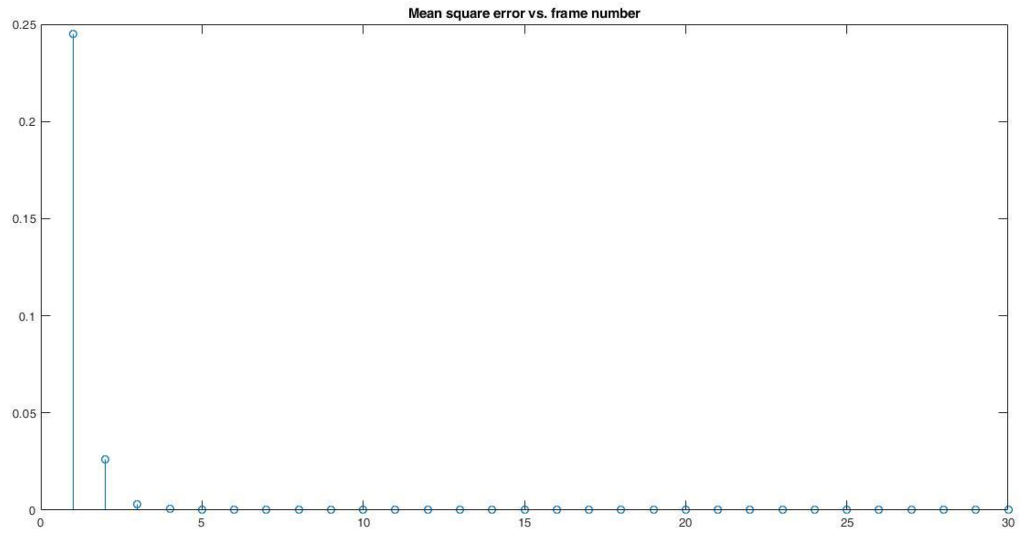

Adaptive filter output


Desired output in 1st frame


Adaptive filter output in 1st frame
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
50
100
150
200
250
300
350


Desired output in 6th frame
1.5
1
0.5
0
-0.5
-1
-1.5
0
50
100
150
200
250
300
350

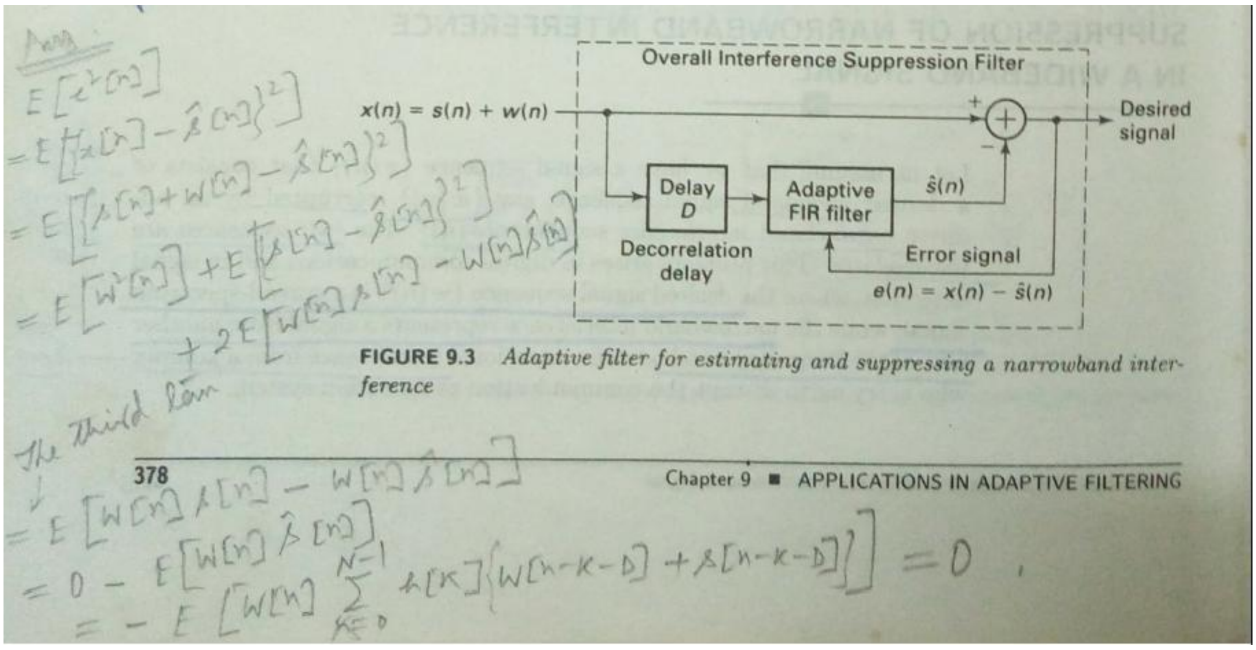

Adaptive filter output in 6th frame
1.5
1
0.5
0
-0.5
-1
-1.5
0
50
100
150
200
250
300
350


Mean square error vs. frame number
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30

Experiments No. 2 (Suppression of narrowband interference in a wideband signal)

1. ***Write a program for noise cancellation using an LMS adaptive filter. Observe (listen to) the output using the adaptive filter.***

Block diagram of the system (from DSP using MATLAB - *Ingle and Proakis*)



Overall Interference Suppression Filter
$x(n) = s(n) + w(n)$
Desired signal
Delay D
Decorrelation delay
Adaptive FIR filter
$\hat{s}(n)$
Error signal
$e(n) = x(n) - \hat{s}(n)$
378
Chapter 9 ■ APPLICATIONS IN ADAPTIVE FILTERING

FIGURE 9.3 *Adaptive filter for estimating and suppressing a narrowband interference*

Ans

$E[e^2[n]]$

$= E[\{x[n] - \hat{s}[n]\}^2]$

$= E[\{s[n] + w[n] - \hat{s}[n]\}^2]$

$= E[w^2[n]] + E[\{s[n] - \hat{s}[n]\}^2]$
$+ 2E[w[n]s[n] - w[n]\hat{s}[n]]$

The third Term ↓

$= E[w[n]s[n] - w[n]\hat{s}[n]]$

$= 0 - E[w[n]\hat{s}[n]]$

$= -E[w[n] \sum_{k=0}^{N-1} h[k]\{w[n-k-D] + s[n-k-D]\}] = 0$,

Program

```
clc;
close all;
clear all;

w=rand(1,1000)-0.5;              % WIDEBAND SIGNAL
figure;
subplot(311);
stem(w(200:300));xlim([0 100]);
title('Wideband Sequence');xlabel('Sample->');ylabel('Amplitude');



w1=0.4*pi;
s=cos(w1*(1:1000));           % NARROWBAND INTERFERENCE SIGNAL
subplot(312);stem(s(200:300));
xlim([0 100]);
title('Narrowband Sequence');xlabel('Sample->');ylabel('Amplitude');
```

```
x=s+w;                    % RECEIVED SIGNAL
subplot(313);stem(x(200:300));
xlim([0 100]);
title('Received Squence x=s+w');xlabel('Sample->');ylabel('Amplitude');


b=[0 1];
a=1;
y1=filter(b,a,x);         % DELAY

N=32;
M=length(y1);
y=zeros(1,M);
h=zeros(1,N);
e(1:N-1)=y1(1:N-1);

x2=0;
for t=1:M
   x2=x2+y1(t).^2;
end

de=(M+1)/(N*x2);
for n=N:M
   x1=y1(n:-1:n-N+1);
   y(n)=h*x1';              % ESTIMATE OF INTERFERENCE SIGNAL.
   e(n)=s(n)-y(n);
   h=h+de*e(n)*x1;
end

z1=x-y;
figure(2);
subplot(211);
stem(z1(200:300)); xlim([0 100]);ylim([0,1]);
title('Generated Wideband Sequence');xlabel('Sample->');ylabel('Amplitude');
subplot(212);
stem(w(200:300));xlim([0 100]);
title(' Given Wideband Sequence');xlabel('Sample->');ylabel('Amplitude');
mse=mean((z1-w).^2)
```

Experiment -3 (LPC Analysis and Synthesis of Speech)

1. ***To implement Linear Predictive Coding (LPC) method for speech analysis and synthesis in non real time.***

Steps involved in non real time Linear Predictive Coding (LPC) method for speech analysis and synthesis using MATLAB.

1) Open Simulink Library Browser in Malab r2007a.
2) Open a new Simulink editor window and save that in your work folder in *My document*.
3) Choose *Signal From Workspace* block and drag it into the editor window from Signal Processing Blockset→ signal management → Signal Processing Sources

Select these options on double clicking on the blocks

For Signal From Workspace block-

Signal- mtlb

Sample time- 1/7418

Samples per frame- 80

From output from final data value by- Setting to zero

4) Choose Buffer block and drag it into the editor window from Signal Processing Blockset→ signal management → Buffer

Select these options on double clicking on the blocks

For Buffer Block-

Output buffer size (per channel)- 80

Buffer overlap- 0

Initial conditions- 0

5) Choose Autocorrelation LPC block and drag it into the editor window from Signal Processing Blockset→ Estimation→ Linear Prediction

Select these options on double clicking on the blocks

For Autocorrelation LPC Block-

Outputs- A

Prediction order (N) – 10

6) Choose *Digital Filter* block and drag it into the editor window from Signal Processing Blockset→ Filtering→ Filter Designs

Select these options on double clicking on the blocks

For Digital Filter Block-

Coefficient source- Input port(s)

In the ***Main*** option-

Transfer function type- FIR (all zeros)

Filter structure- Direct form

Coefficient update rate- One filter per frame

Initial conditions- 0

7) Choose Digital Filter block and drag it into the editor window from Signal Processing Blockset→ Filtering→ Filter Designs

Select these options on double clicking on the blocks

For Digital Filter1 Block-

Coefficient source- Input port(s)

In the ***Main*** option-

Transfer function type- IIR (all poles)

Filter structure- Direct form

Select- First denominator coefficient=1, remove a0 term in the structure

Coefficient update rate- One filter per frame

Initial conditions- 0

8) Choose *To Workspace* block and drag it into the editor window from Simulink→ Sinks
You can rename it as *Re-synthesized Signal*

Select these options on double clicking on the blocks

For Re-synthesized Signal block Block-

Variable name- y_lpc

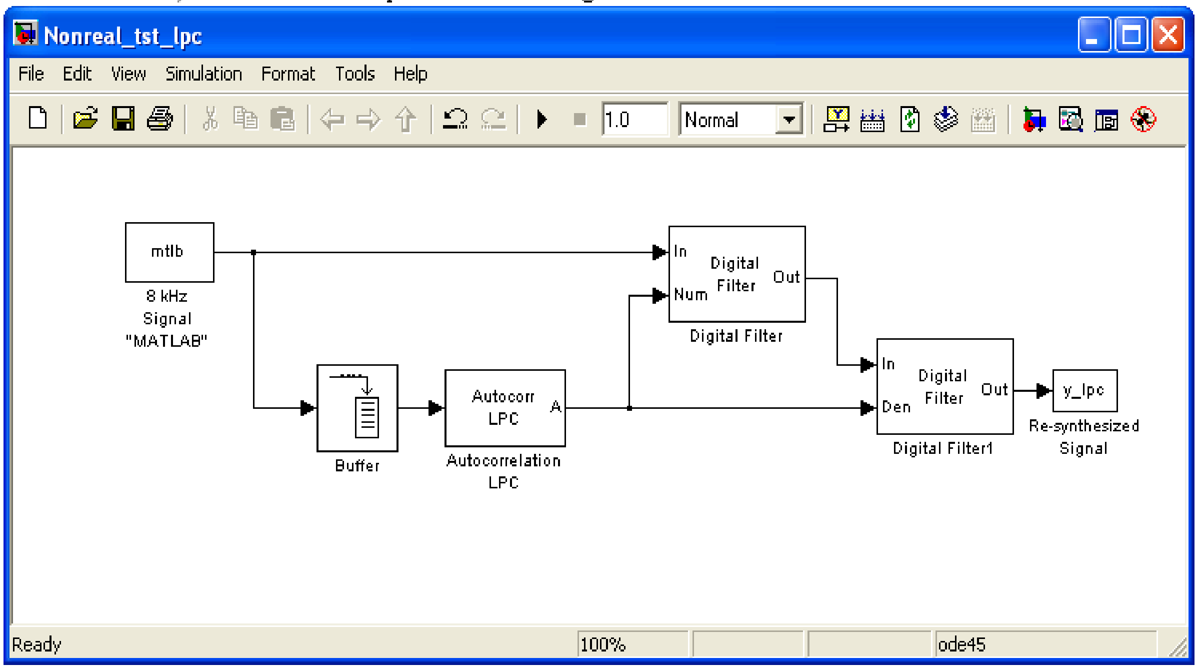
Limit data points to last- inf

Decimation- 1

Sample time (-1 for inherited)- -1

Save format- Array

9) Connect all these block as shown in figure.
10) Save
11) Give the MATLAB program generated input signal.
12) Listen to the input signal using *sound* command.
13) Now Click on *Start Simulation*. (In toolbar you can see the icon for Start Simulation or you can go in simulation→ Start). It will run your model.
14) After simulation is complete, the output of this simulink model will be stored in the form of an array, named as *y_lpc*.
15) Now observe the output *Re-synthesized Signal*. It should be the same as the input signal.
16) You can use *sound* command to listen to the *Re-synthesized Signal*



Expt- 4 (Tasks Related To Digital Image Processing)

1. ***Take the 'eight.tif' image. Obtain FFT of this image. Display this FFT as an intensity image with origin at the centre of the image. Interpret this FFT image. Apply Gaussian smoothing filter, with different variances, to it and observe the output. Comment on the results.***

2. ***Take an image, blur it using a filter (also add noise to it) and then achieve deblurring using inverse filtering.***

Program 1

```
x=imread('eight.tif');
figure;
imshow(x);
[nrow,ncol]=size(x);
fftx=fft2(x);
fftx1=fftshift(fftx);

figure;
mag1=abs(fftx1)+1;
colormap(gray(256));
imagesc(log(mag1));

sigma=input('give variance for Gaussian filter(freq. domain)=');

for i=1:nrow
   for j=1:ncol
      d(i,j)=(i-nrow/2)^2+(j-nrow/2)^2;
      h(i,j)=exp(-d(i,j)/(2*sigma*sigma));
      g(i,j)=fftx1(i,j)*h(i,j);
      g1(i,j)=g(i,j)*((-1)^(i+j));
   end
end

figure;
mag2=abs(g)+1;
colormap(gray(256));
imagesc(log(mag2));

y=ifft2(g);
figure;
z=abs(y);
imshow(z,colormap(gray(256)));

%colormap(gray(256));
%image(abs(y));
```

Program 2

```
%DEBLURRING OF IMAGES IN SPATIAL AND FREQUENCY DOMAIN (WITH
INVERSE FILTER)

%reads image
% fid1=fopen('girl.img','rb');
% img1=fread(fid1,[256,256]);
% x=imrotate(img1,-90);

x1=imread('eight.tif');
x=double(x1);

%FFT of the image
fftx=fft(x);

%displays image
figure;
colormap(gray(256));
image(x);

%blurring with 7 X 7 window
ker=ones(7,7)/49;

%Convolution method
blur1_x=conv2(x,ker);
figure;
colormap(gray(256));
image(blur1_x);


%generate noise
%nois=rand(size(x));


%generate noisy image
%x_nois=double(x)+50*nois;
x_nois=imnoise(x1,'gaussian',0,0.005);
figure;
colormap(gray(256));
image(x_nois);

%generate noisy+blurred image
% nois1=rand(size(blur1_x));
% x_blr_nois=blur1_x+nois1;
x_blr_nois=imnoise(uint8(blur1_x),'gaussian',0,0.005);
figure;
colormap(gray(256));
image(x_blr_nois);


% %Deblurring with Inverse Filtering (without noise)
[blurx,blury]=size(blur1_x);
fftker=fft2(ker,blurx,blury);
fft_deblr1=fft2(blur1_x)./fftker;
deblr_1=ifft2(fft_deblr1);
abs_deblr_1=abs(deblr_1);
figure;
colormap(gray(256));
image(uint8(abs_deblr_1));

% % %Deblurring with Inverse Filtering (with noise)
% fft_deblr2=fft2(x_blr_nois)./fftker;
% deblr_2=ifft2(fft_deblr2);
% figure;
% colormap(gray(256));
% image(uint8(abs(deblr_2)));
```

Home Exercises

1. (a) Read the image *'eight.tif'*. Display it.
   (b) Add Gaussian noise of zero mean and variance 0.2 to the image. (You may use the function *imnoise* available in Matlab). Display the noisy image.

   © Vary the variance of noise from 0.1 to 0.5 and observe changes in the image, if any.

<u>Experiment-5</u>

<u>MINI PROJECT</u>

(more than one lab class may be devoted for this purpose) on applications of signal processing (e.g. speech enhancement, music synthesis and analysis etc.)

<u>On 'speech enhancement'</u> **<u>(reduction of noise in a given speech sample).</u>**

- Though there is no constraint e.g. no specific method/no specific condition (say, SNR of the noisy speech sample) is mentioned here and you can choose the method as well as the speech file, you need to clearly mention these (the method that you choose, descriptions of the speech samples
including the database).

- You should be having clear understanding of the method that you are implementing and each part of the code is to be clearly explained by you. Though a contribution in the form of modification of an existing method (do not underestimate yourself, an apparently simple modification may lead to some improvement. I am more interested in knowing how much you are exploiting your thinking power) is what I would like to have, implementation of a recent method (the code, at least partially, written by you) will be also accepted.

- In the next class, we shall have an assessment of your preparation for this project (your understanding of the fundamentals and knowledge of the simple methods e.g. Spectral subtraction, Wiener filtering etc.). Those who put more effort and get acquainted with the corresponding codes (these are available in the books/different websites including that of Mathworks, the developer of Matlab) will, of course, get more credit.

- There are many good books available on speech processing. One of them that has been mentioned earlier, in the context of another expt. on speech processing (T.F. Quatieri, Discrete time Speech processing) may help you for the fundamentals in this case also.

<u>The report should contain</u>

i) Theory (relevant to the work done only) – precisely written

ii) A working program (the program must be commented); Mention the version of the software that you used.

iii) Your contribution is to be clearly discussed – in the theory and also, the corresponding portion in the program is to be highlighted;
As the task involves working with different blocks e.g. for spectral subtraction, you need to estimate noise (VAD may be required also), then subtract it from the noisy speech which is again possible in different ways, some of the blocks (and the corresponding sections in the programs) may be common among your submissions, esp. if you use the codes from the available resources. This may be true in case you use any other method also. However, you need to clearly tell/highlight your contributions.

iv) Description of the database and the speech files used by you (sampling frequency, no. of bits used for representing samples, male/female voice, text corresponding to the speech etc.)

v) The performance measures (PESQ and other measures that you may use) are to be clearly mentioned.

vi) Send the speech files you used as the input (with the information about them, as mentioned above, in the report), the output (enhanced) speech files.

vii) Mention the references (books, research papers, resources available online etc.) and the website addresses that are relevant in this context (where you got the codes)

viii) Discussions and suggestions for improvement of performance (should be clearly written, based on your experience while executing the project)

Experiment -6

Experiment: Study of channel model

Aim- Study of channel model and plot **pdf** of Rayleigh channel and Rician channel by using MATLAB

Theory

Drive PDF of Rayleigh channel/ Rayleigh random variable

The Rayleigh channel model stated that circular symmetric random variable is of the form **Z = X + jY**, where real and imaginary parts are zero mean independent and identically distributed (iid) Gaussian random variable with magnitude $|Z|$.

The probability density function of Z is

$p(z) = \frac{z}{\sigma^2} e^{\frac{-z^2}{2\sigma^2}}, \quad z \geq 0$ is called Rayleigh random variable

Further, the phase θ is uniformly distributed from [0, 2π]

We will try to derive the expression for **probability density function (PDF)** for $|Z|$ and θ .**Joint probability**

The probability density function of x is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}.$$

Similarly, probability density function of y is

$p(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-y^2}{2\sigma^2}}$ as X and Y are independent random variable

The joint probability is the product of the individual probability, i.e,

$$p(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Simplifying,

$$\begin{aligned} P(z \leq Z + dz, \theta \leq \Theta + d\theta) &= \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} z\,dz\,d\theta \\ &= \frac{z}{\sigma^2} e^{\frac{-z^2}{2\sigma^2}} dz \frac{1}{2\pi} d\theta \end{aligned}.$$

Summarizing the joint probability density function,

$$p(z,\theta) = \frac{z}{2\pi\sigma^2} e^{\frac{-z^2}{2\sigma^2}}.$$

Since z and θ are independent, the individual **probability density functions** are,

$$p(z) = \frac{z}{\sigma^2} e^{\frac{-z^2}{2\sigma^2}},\ z \geq 0,$$

$$p(\theta) = \frac{1}{2\pi},\ -\pi \leq \theta \leq \pi.$$

Simulation Model

A simple MATLAB/Octave simulation model is provided for plotting the probability density of z and θ.

Follow the steps to generate a graph

(a) Generate two independent zero mean, unit variance Gaussian random variables

(b) Using the hist() function compute the simulated probability density for both z and θ

(c) Using the knowledge of the equation (which we just derived), compute the theoretical probability
density function (PDF)

(d) Plot the simulated and theoretical probability density functions (PDF) and show that they are in good agreement.

Output

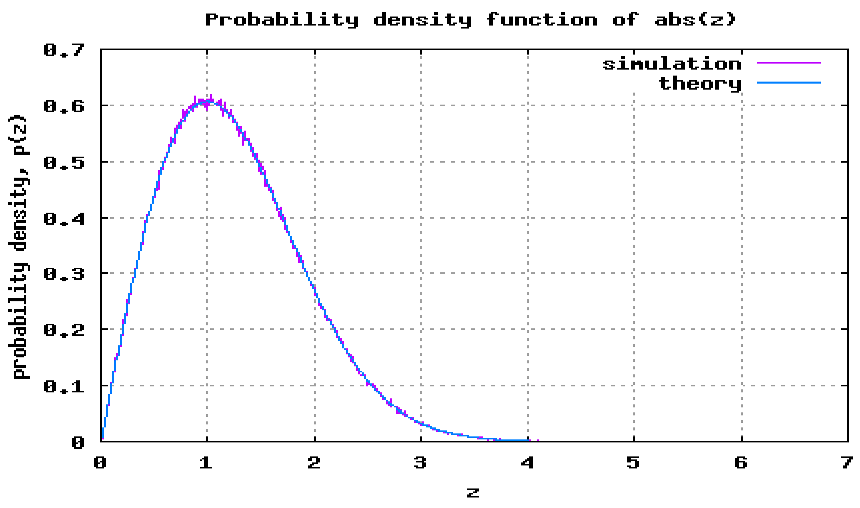


Figure: Simulated/theoretical PDF of Rayleigh random variable

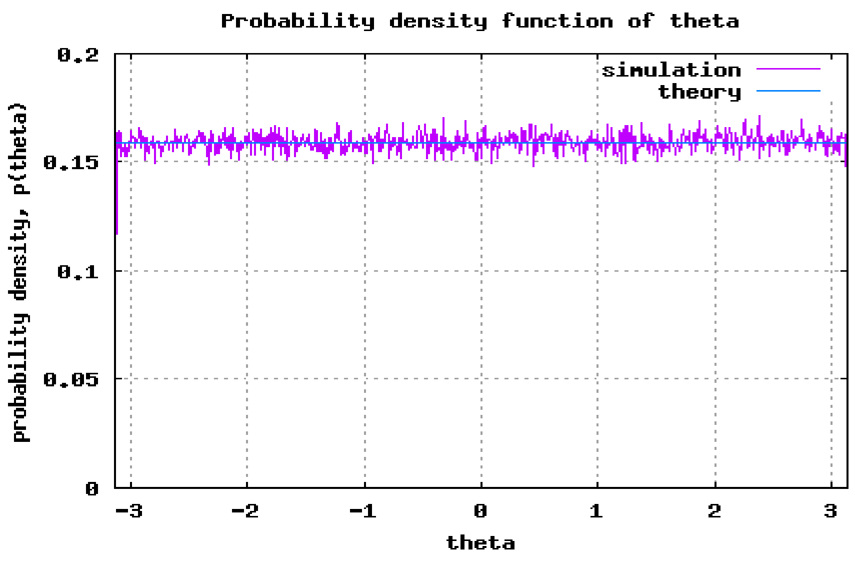


Figure: Simulated/theoretical PDF of uniformly distributed theta random variable

Rician:

If the environment is such that, in addition to the scattering, there is a strongly dominant signal seen at the receiver, usually caused by a LOS, then the mean of the random process will no longer be zero, varying instead around the power-level of the dominant path. Such a situation may be better modeled as Rician fading.

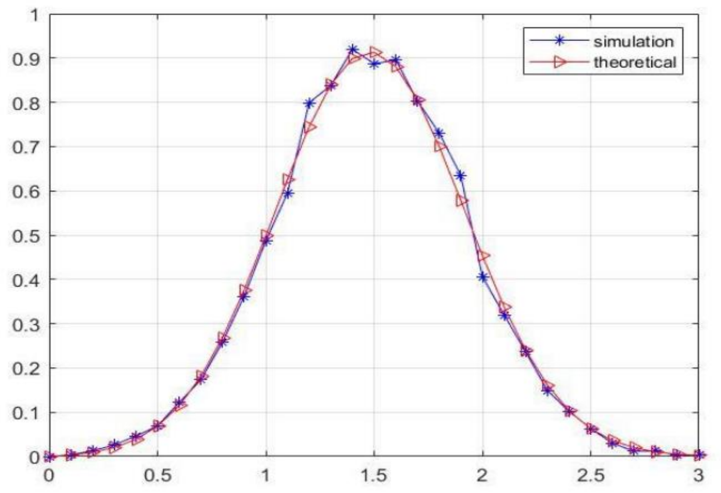
In such a situation, random multipath components arriving at different angles are superimposed on a stationary dominant signal. At the output of an envelope detector, this has the effect of adding a dc component to the random multipath.

The effect of a dominant signal arriving with many weaker multipath signals gives rise to the Rician distribution. As the dominant signal becomes weaker, the composite signal resembles a noise signal which has an envelope that is Rayleigh. Thus, the Rician distribution degenerates to a Rayleigh distribution when the dominant component fades away.

The Rician distribution is given by

$$p(r) = \begin{cases} \frac{r}{\sigma^2} e^{-\left(\frac{r^2+A^2}{2\sigma^2}\right)} I_0\left(\frac{Ar}{\sigma^2}\right), & for\ (A \geq 0, r \geq 0) \\ 0, & (r < 0) \end{cases}$$

Plot:



Power Distributions of Rayleigh channel

We now consider the distribution of the envelope and power for the narrowband received signal $r(t) = r_I(t)\cos(2\pi f_c t) - jr_Q(t)\sin(2\pi f_c t)$. It can be shown that, for any two Gaussian random variables X and Y, both with mean zero and equal variance σ^2, $Z = \sqrt{X^2 + Y^2}$ is Rayleigh distributed and Z^2 is exponentially distributed.

$$P_{z^2}(x) = \frac{1}{2\sigma^2} e^{-\frac{x}{2\sigma^2}}$$

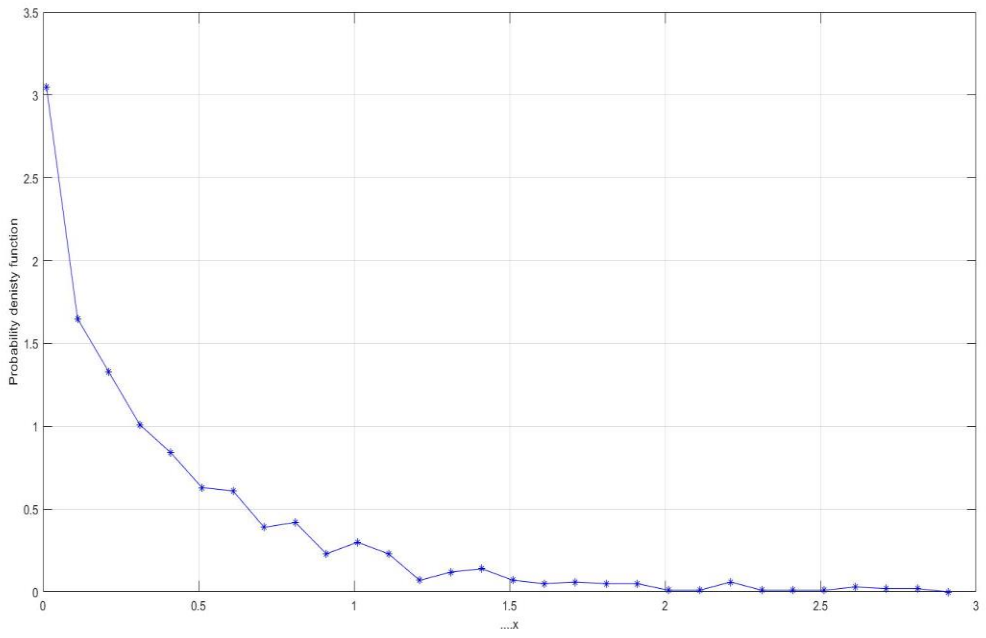


Fig. (Simulation Result)

Experiment – 7

Aim:- Circular convolution and its application in the cyclic prefix of OFDM system by using MATLAB

Theory: The Circular convolution, also known as cyclic convolution, of two periodic functions occurs when one of them is convolved in the normal way with a periodic summation of the other function. Circular convolution is only defined for finite length functions (usually 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. Circular convolution sum can be calculated using the formula:

$$y(n) = x_1(n) * x_2(n) = \sum_{k=0}^{N-1} x_1(k)x_2(n-k)$$

$$n=0,1,2\ldots\ldots\ldots N\text{-}1 \qquad \ldots\ldots.(1)$$

Circular convolution can be performed in different ways:

1. Using the expression for linear convolution sum, but assuming the signal repeats periodically. This can be done by changing the negative indices of *(n-k)* to repetitions of the latter portions of the original aperiodic signal.

2. Convolution in time domain corresponds to multiplication in frequency domain. To make use of this property, we can calculate the DTFT of each of the aperiodic signals, multiply these in the frequency domain, and find the IDFT of the product, to get the periodic convolved signal in time domain.

x[n] and h[n] are two finite sequences of length N with DFTs denoted by X[k] and H[k], respectively. Let us form the product

W[k] = X[k]H[k]

and determine the sequence w[n] of length N for which the DFT is W[k].

Linear convolution:

linear convolution same as the circular convolution except that periodicity in time domain sequence.

OFDM Cyclic Prefix, CP-OFDM:

The cyclic prefix used in Frequency Division Multiplexing schemes including OFDM to primarily act as a guard band between successive symbols to overcome intersymbol interference, ISI.

- The cyclic prefix provides a guard interval to eliminate intersymbol interference from the previous symbol.

- It repeats the end of the symbol so the linear convolution of a frequency-selective multipath channel can be modeled as circular convolution, which in turn may transform to the frequency domain via a discrete Fourier transform. This approach accommodates simple frequency domain processing, such as channel estimation and equalization.
- The cyclic prefix is created so that each OFDM symbol is preceded by a copy of the end part of that same symbol.

SIMULATION:

- Take two sequence of time signal.
- To perform the circular convolution and linear convolution by using MATLAB Function (**cconv conv**) and also do the same without using function.
- Generate the OFDM system and add the cyclic prefix.
- Take the Rayleigh fading channel and do the convolution.
- Do the DFT of convolved signal.
- Take the DFT of OFDM signal and zero padded channel.
- Multiply both the dft.
- Verify your both results should be same.

Experiment- 8

Aim: - BER analysis of OFDM systems for Rayleigh fading channel

Theory:

OFDM (Orthogonal Frequency Division Multiplexing) is a multicarrier digital communication scheme to solve both issues. It combines a large number of low data rate carriers to construct a composite high data rate communication system. Orthogonality gives the carriers a valid reason to be closely spaced, even overlapped, without inter-carrier interference. The low data rate of each carrier implies long symbol periods, which greatly diminishes inter-symbol interference.

This objective is met by developing a MATLAB program to simulate a basic OFDM system. From the process of this development, the mechanism of an OFDM system can be studied and with a completed MATLAB program, the characteristics of an OFDM system can be explored.

OFDM Basic:

In digital communications, information is expressed in the form of bits. The term symbol refers to a collection, in various sizes, of bits. OFDM data are generated by taking symbols in the spectral space using BPSK, QAM, M-PSK, etc, and converting the spectra to the time domain by taking the Inverse Discrete Fourier Transform (IDFT).

Since Inverse Fast Fourier Transform (IFFT) is more cost-effective to implement. Once the OFDM data are modulated to a time signal, all carriers transmit in parallel to fully occupy the available frequency bandwidth. During modulation, OFDM symbols are typically divided into frames. The data will be modulated frame by frame in order for the received signal to be in sync with the receiver. Long symbol periods diminish the probability of having inter-symbol interference, but could not eliminate it.

To make ISI nearly eliminated, a cyclic extension (or cyclic prefix) is added to each symbol period. An exact copy of a fraction of the cycle, typically 25% of the cycle, taken from the end is added to the front.

The key to OFDM is maintaining the orthogonality of the carriers. If the integral of the product of two signals is zero over a time period, then these two signals are said to be orthogonal to each other.

$$BER_{Th} = \frac{1}{2}\left(1 - \sqrt{\frac{SNR}{SNR+1}}\right)$$

Block diagram of OFDM transmitter and receiver:

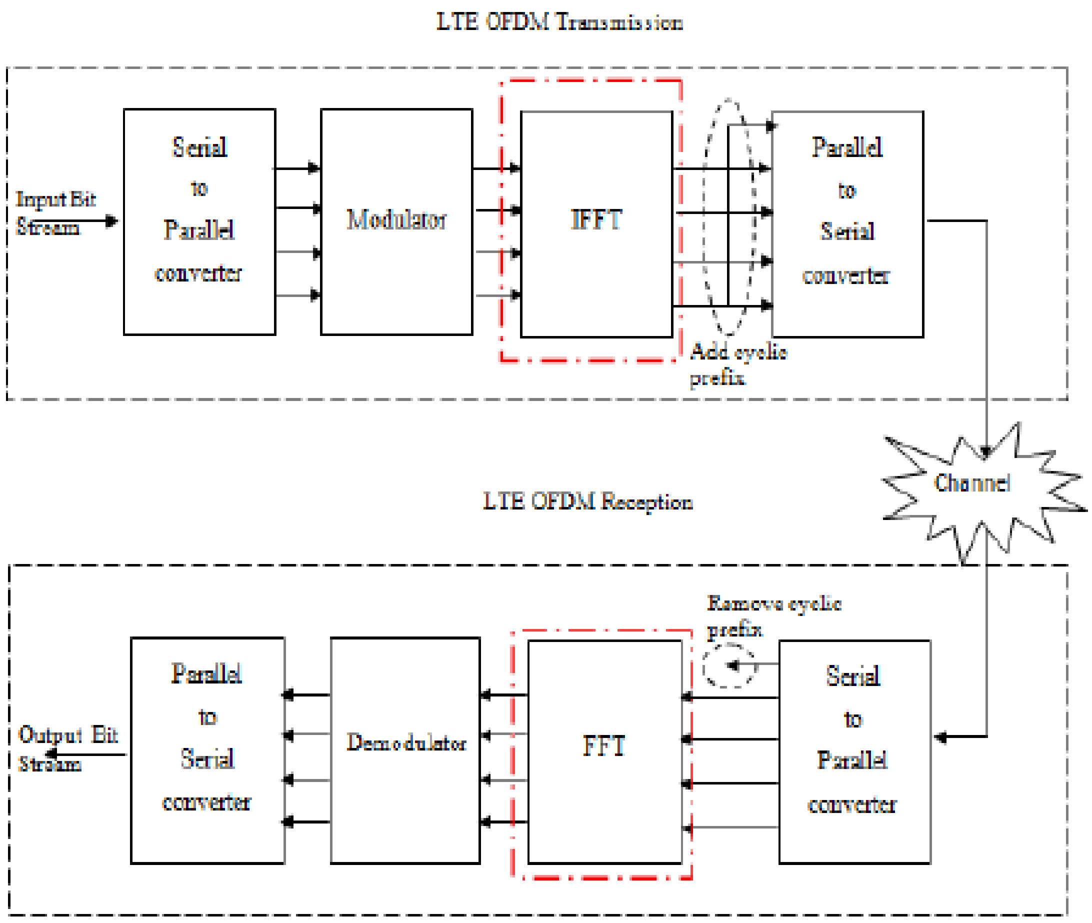


Fig.

MATLAB Steps in OFDM modulation and demodulation:

(a) Generation of random binary sequence

(b) BPSK modulation i.e. bit 0 represented as -1 and bit 1 represented as +1

(c) Convolving each OFDM symbol with a 10-tap Rayleigh fading channel.

(d) Adding White Gaussian Noise

(e) Grouping the received vector into multiple symbols, removing cyclic prefix

(f) Converting the time domain received symbol into the frequency domain

(g) Dividing the received symbol with the known frequency response of the channel

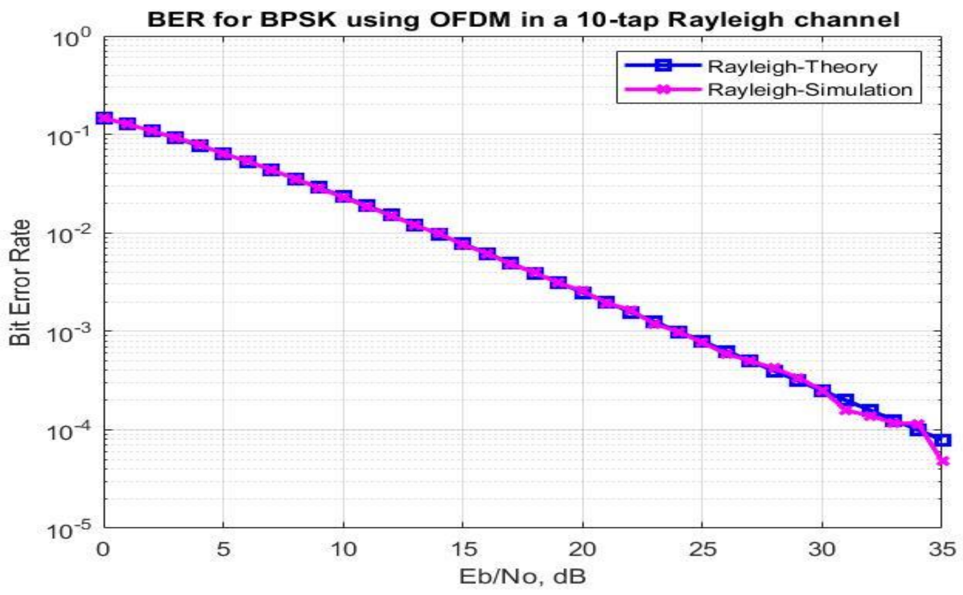
(h) Taking the desired subcarriers

(i) Demodulation and conversion to bits

(j) Counting the number of bit errors

(l) Plot the BER and theoretical BER

Plot



Experiment - 9

Aim:- Study of MIMO system and SVD

- To Generation and Decomposition of MIMO channel matrix (SVD)
- Coefficients of the channel matrix H are independently Rayleigh distributed with equal variance
- To investigate how the number of transmit antennas and number of receive antennas affect the system.

Theory

How to handle the MIMO channel?

- In MIMO systems, the fading channel between each transmit-receiveantenna pair can be modeled as a SISO channel.
- the transmitter knows the N channels and can therefore allocatepower intelligently to maximize capacity
- Let the rank of H is n then the MIMO channel can be decomposed by SVD into n parallel spatial channels where we can decide how to use these channels and how much energy to be allocated to each channel
- Eigenvalues and Eigenvector of a matrix (Please revise this topic ofLinear Algebra)

Assumed that the transmit power of each transmitter is 1.0.

$$\sum_{j=1}^{N_T}(h_{ij})^2 = N_T \qquad \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1N_T} \\ h_{21} & h_{22} & \cdots & h_{2N_T} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N_R,1} & h_{N_R,2} & \cdots & h_{N_R N_T} \end{bmatrix}$$

$$r_1 = h_{11}s + h_{12}s \cdots + h_{1N_T}s$$

If the number of equations is larger than the number of unknowns (i.e. NR > NT) then the solution can be found using a zero-forcing algorithm. When NT = NR, then the solution can be found by (ignoring noise) inverting the H matrix as in

The MIMO Channel matrix

- The system performs best when the H matrix is full rank, with each row/column meeting conditions of independence: only when each path is fully independent of all others.

- This can happen only in an environment that offers rich scattering, fading and multipath.
- We can extract the transmitted information only when the H matrix isinvertible. And the only way it is invertible is if all its rows and columns are uncorrelated
- the scattering, fading and all other effects cause the channels to be completely uncorrelated

What is SVD of a matrix?

- One of the most powerful computational tools in numerical linearalgebra
- Commonly used to solve i) the unconstrained linear least squaresproblems, ii) matrix rank estimation and iii) canonical correlation analysis

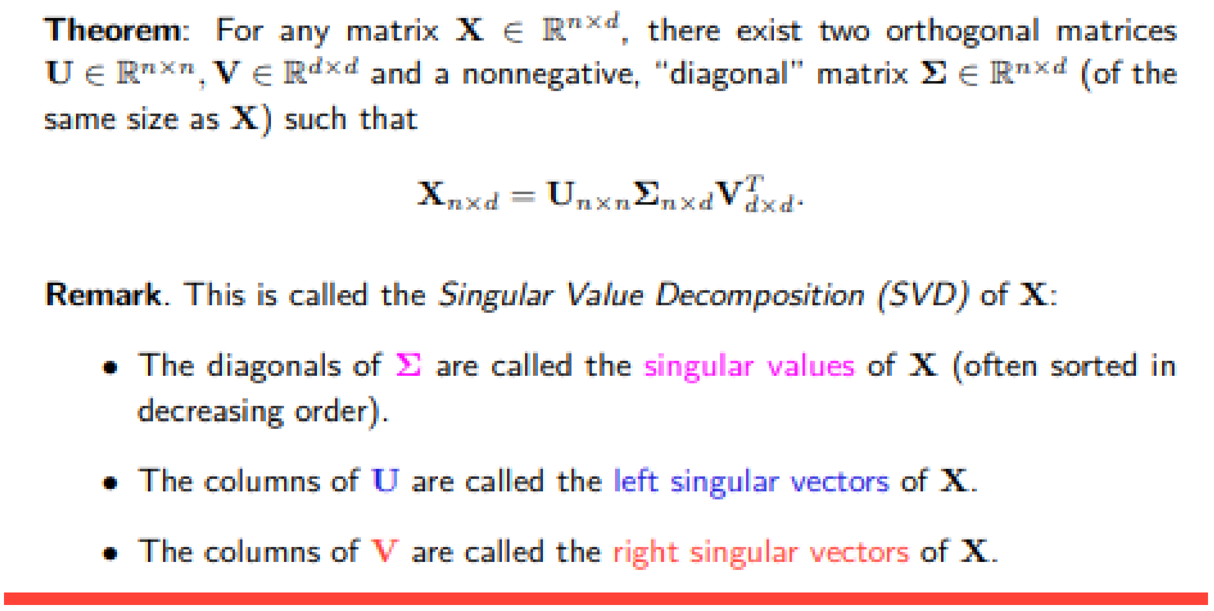
$A = U\Lambda V^*$. In this setting, Λ is a diagonal matrix and it has the same size of A, U and V are square matrices of order m and n, respectively. Λ can be represented as

$$\Lambda = \begin{pmatrix} \Lambda_r & 0 \\ 0 & 0 \end{pmatrix} \tag{4}$$

$$\Lambda_r = \begin{pmatrix} \sigma_1 & & 0 \\ 0 & \sigma_2 & 0 \\ & \cdots & \\ 0 & & \sigma_r \end{pmatrix} \tag{5}$$

and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ are non-negative real values and r determines the rank of a matrix. The columns of U and V are normalized singular vectors satisfying $U^*U = I$ and $V^*V = I$ In otherworld, U and V are orthogonal if they are real or unitary if they are complex. There is a unique Λ for each matrix but U and V are not.

In fact, the diagonal entries of Λ are the non-negative square roots of the eigen values of AA^*, the columns of U are the eigenvectors of AA^* and the columns of V are the eigenvectors of A^*A.



Theorem: For any matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, there exist two orthogonal matrices $\mathbf{U} \in \mathbb{R}^{n\times n}, \mathbf{V} \in \mathbb{R}^{d\times d}$ and a nonnegative, "diagonal" matrix $\mathbf{\Sigma} \in \mathbb{R}^{n\times d}$ (of the same size as $\mathbf{X}$) such that

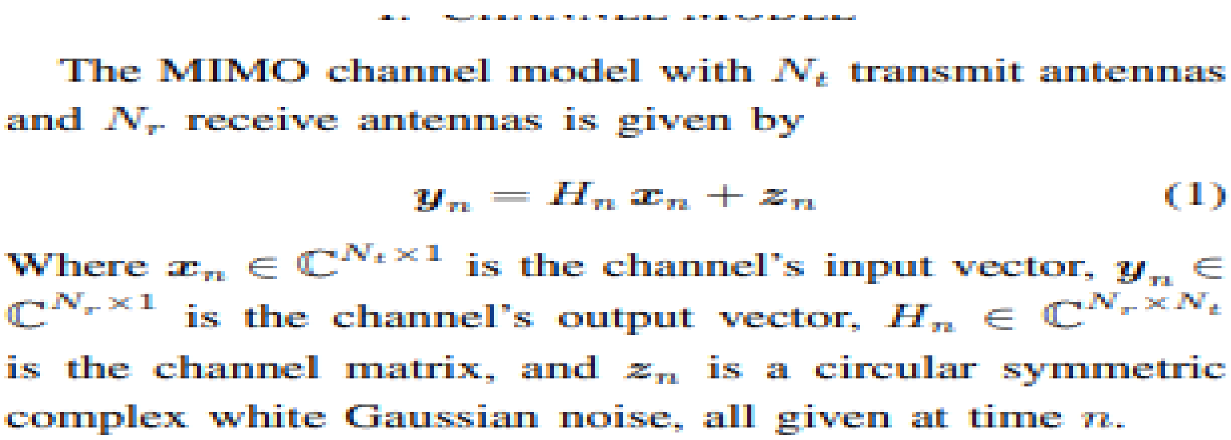
$$\mathbf{X}_{n\times d} = \mathbf{U}_{n\times n}\mathbf{\Sigma}_{n\times d}\mathbf{V}^T_{d\times d}.$$

Remark. This is called the *Singular Value Decomposition (SVD)* of $\mathbf{X}$:

- The diagonals of $\mathbf{\Sigma}$ are called the singular values of $\mathbf{X}$ (often sorted in decreasing order).
- The columns of $\mathbf{U}$ are called the left singular vectors of $\mathbf{X}$.
- The columns of $\mathbf{V}$ are called the right singular vectors of $\mathbf{X}$.

Basics of SVD

- The singular value decomposition of a matrix A is the factorization of A intothe product of three matrices A = UDV where the columns of U and V are orthonormal and the matrix D is diagonal with positive real entries.
- Calculating the SVD consists of finding the eigenvalues and eigenvectorsof AA^T and A^TA. The eigenvectors of A^TA make up the columns of V , the eigenvectors of AA^T make up the columns of U.
- The singular values in **S** are square roots of eigenvalues from AA^T or A^TA. The singular values are the diagonal entries of the S matrix and are arrangedin descending order. The singular values are always real numbers. If the matrix A is a real matrix, then U and V are also real



[illegible]

The MIMO channel model with N_t transmit antennas and N_r receive antennas is given by

$$y_n = H_n x_n + z_n \qquad (1)$$

Where $x_n \in \mathbb{C}^{N_t\times 1}$ is the channel's input vector, $y_n \in \mathbb{C}^{N_r\times 1}$ is the channel's output vector, $H_n \in \mathbb{C}^{N_r\times N_t}$ is the channel matrix, and z_n is a circular symmetric complex white Gaussian noise, all given at time n.

singular value decomposition (SVD) as follows:

$$H_n = U_n S_n V_n^\dagger \tag{2}$$

where $U_n \in \mathbb{C}^{N_r \times N_r}$ and $V_n \in \mathbb{C}^{N_t \times N_t}$ are unitary matrices of the the left and right singular vectors of H_n respectively, and $S_n \in \mathbb{C}^{N_r \times N_t}$ is a diagonal matrix having the nonnegative $\{s_k\}_{k=1}^{\min(N_r,N_t)}$ singular values of H_n on its diagonal. Consequently, the SVD factorization of $H_n^\dagger$ is given by

$$H_n^\dagger = V_n S_n^T U_n^\dagger \tag{3}$$

If the matrix V_n is known on the forward transmission site at time n, and U_n on the reverse site, then the channel in both directions can be diagonalized:

$$\begin{aligned}(U_n^\dagger \boldsymbol{y}_n) &= S_n(V_n^\dagger \boldsymbol{x}_n) + U_n^\dagger \boldsymbol{z}_n \\ (V_n^\dagger \bar{\boldsymbol{y}}_n) &= S_n^T(U_n^\dagger \bar{\boldsymbol{x}}_n) + V_n^\dagger \bar{\boldsymbol{z}}_n\end{aligned}$$

Questions

- Let A = $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$

Find the eigenvalues of matrix A

Do this by hand and verify your results by MATLAB svd command

1. Find the singular values of the matrix $A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$.

Do this using MATLAB and verify if the matrix factorization works as per the theorem discussed in class

Experiment - 10

Aim: To study Software defined radio SDR

Theory: -

Software defined radio is a **rf** communication system where components that have been physically implemented in hardware (e.g modulators/demodulators, mixers, filters, amplifiers, detectors, etc) are instead of implemented by software on a pc or embedded system

Hardware used-

ZCU102 evolution board

AD-FMCOMMS3-EBZ

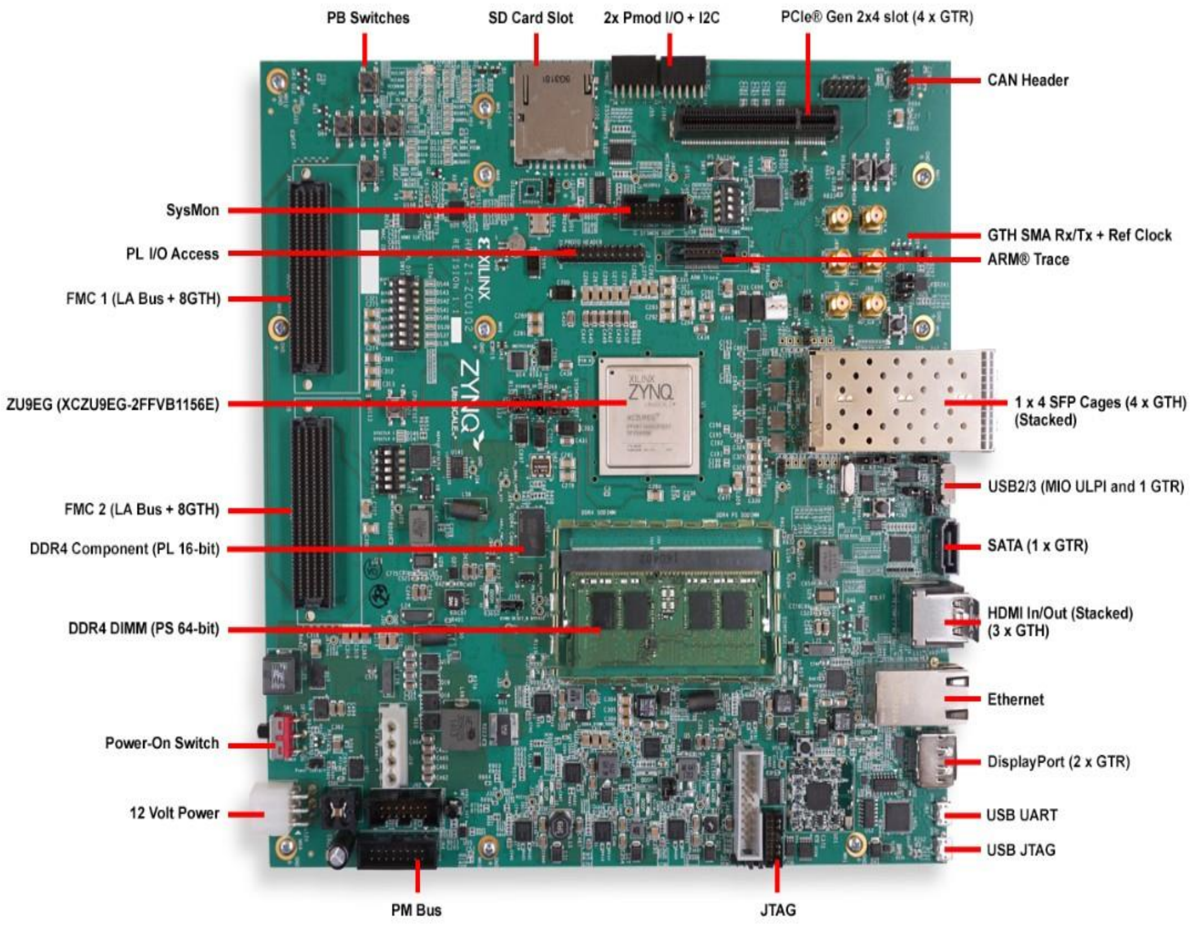


Fig.- ZCU102 board

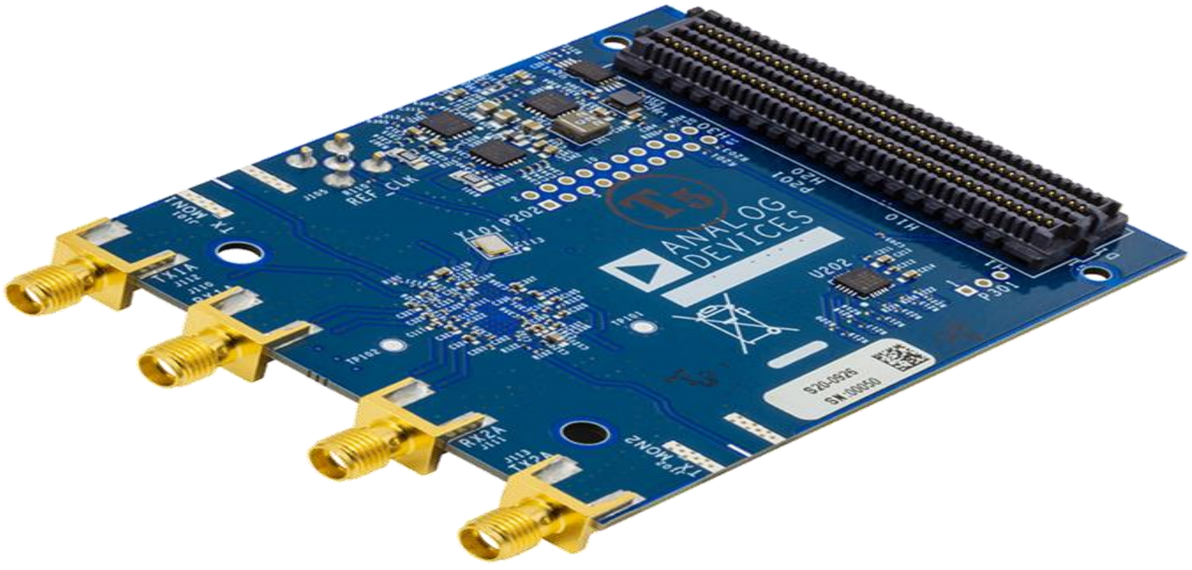


Fig.- FMC board

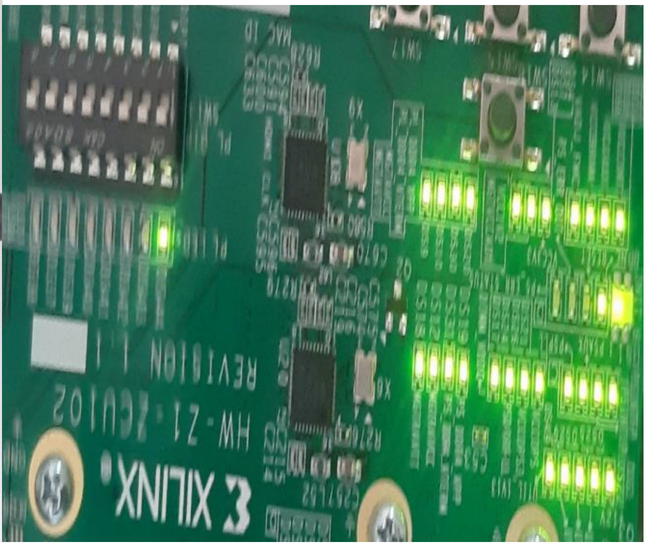
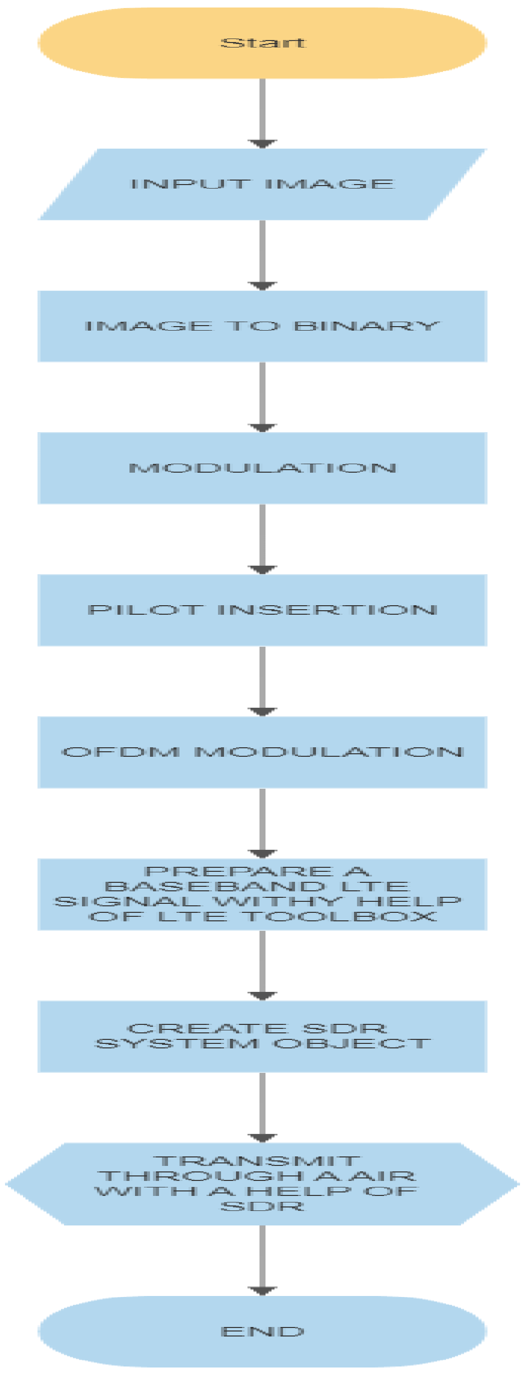


Fig.- connection of ZCU board **Fig.- LED indication of ZCU board**

Transmitter and Receiver Design: -


Start
INPUT IMAGE
IMAGE TO BINARY
MODULATION
PILOT INSERTION
OFDM MODULATION
PREPARE A BASEBAND LTE SIGNAL WITHY HELP OF LTE TOOLBOX
CREATE SDR SYSTEM OBJECT
TRANSMIT THROUGH A AIR WITH A HELP OF SDR
END

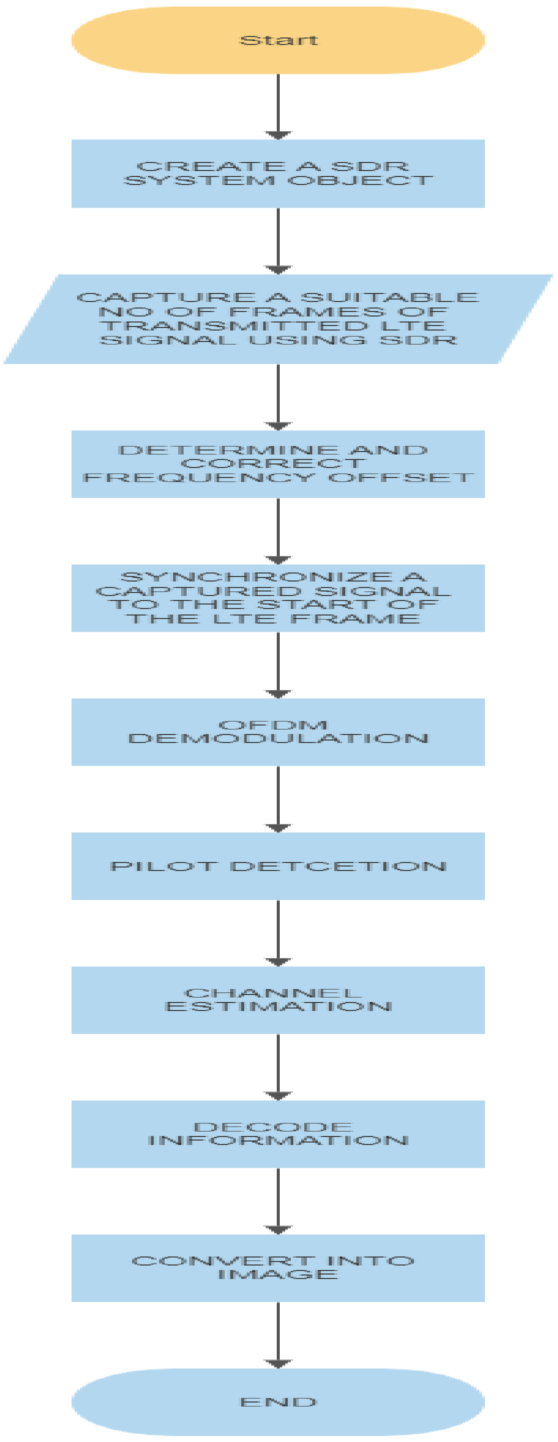

Start
CREATE A SDR SYSTEM OBJECT
CAPTURE A SUITABLE NO OF FRAMES OF TRANSMITTED LTE SIGNAL USING SDR
DETERMINE AND CORRECT FREQUENCY OFFSET
SYNCHRONIZE A CAPTURED SIGNAL TO THE START OF THE LTE FRAME
OFDM DEMODULATION
PILOT DETCETION
CHANNEL ESTIMATION
DECODE INFORMATION
CONVERT INTO IMAGE
END

Fig.- SDR transmitter setup **Fig.-SDR receiver setup**

Transmitter

1. Using LTE Toolbox, create a baseband LTE signal by packaging a binary data stream into the downlink shared channel's transport blocks (DL-SCH).

2. With the help of the SDR device, prepare the baseband signal for transmission.

3. Upsample the baseband data and transmit it at the specified centre frequency using SDR hardware

Receiver

1. Using SDR, capture an appropriate number of frames from the broadcast LTE signal. Determine and correct the frequency offset of the received signal.

2. Line up the collected signal with the beginning of an LTE frame.

3. To obtain an LTE resource grid, OFDM demodulate the received signal.

4. Estimate the received signal's channel strength.

5. From each radio frame's transport blocks, decode the PDSCH and DL-SCH to extract the delivered data.

6. To create the receiving image, recombine the received transport block data.