Techno India NJR Institute of Technology

Department of Electronics & Communication Engineering

B.Tech. VII Semester

Lab: Advance Communication Lab (7EC4-22)

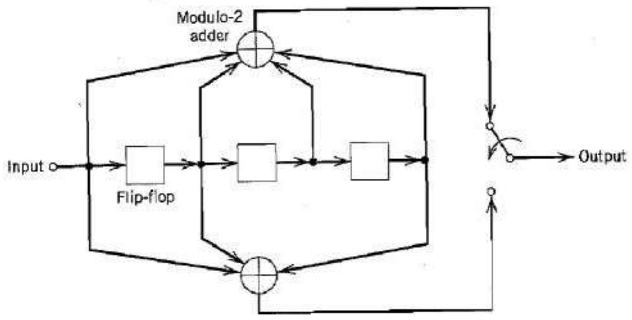
7EC4-22: Advance Communication Lab (MATLAB Simulation)

Credit: 1 **Max. Marks: 50 (IA:30, ETE:20)**

0L+0T+2P

SN	Contents
1	Introduction: Objective, scope and outcome of the course.
Part-A	**Analog-to-digital conversion** 1. Generate a sinusoidal signal. Sample and reconstruct a signal through interpolation. Vary the sampling rate below and above the Nyquist rate and hence verify the Sampling theorem. 2. Generate a sequence of length 500 of zero-mean, unit variance Gaussian random variables. Using a uniform PCM scheme, quantize this sequence to 16, 64 and 128 levels. (a). Find and compare the resulting signal-to-quantization noise ratios. (b). Find the first ten values of the sequence, the corresponding quantized values and the corresponding code words for each case. (c). Plot the quantization error and the quantized value as a function of the input value for each case. **Digital modulation techniques** 3. Simulate the transmitter and receiver for QPSK. Plot the signal and signal constellation diagram. Plot the average probability of symbol error as a function of SNR E_b/N_o, where E_b is the transmitted energy per bit and $N_o/2$ is the double sided power spectral density of additive white Gaussian noise (AWGN) with zero mean. 4. Simulate the transmitter and receiver for 16-QAM. Plot the signal and signal constellation diagram.Plot the average probability of symbol error as a function of SNR E_b/N_o, where E_b is the transmitted energy per bit and $N_o/2$ is the double sided power spectral density of additive white Gaussian noise (AWGN) with zero mean.
PART-B Attempt any four experiment	1. Find all the code words of the (15,11) Hamming code and verify that its minimum distance is equal to 3. 2. Generate an equiprobable random binary information sequence of length 15. Determine the output of the convolutional encoder shown below for this sequence.

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

	 3. Generate the L=31 Gold sequences. Consider a time-synchronous CDMA system (direct sequence spread spectrum) having four users, each employing a distinct Gold sequence of length L=31 and the binary (±1) modulation of their representative Gold sequences. The receiver for each user correlates the composite CDMA received signal, which is corrupted by AWGN (added on a chip-by-chip basis) with each user's respective sequence. Using 10000 information bits, estimate and plot the probability of error for each user as a function of SNR. 4. Consider a MIMO (multiple-input, multiple-output) system with $N_T = 2$ transmit antennas and NR = 2 receive antennas. Generate the elements of the channel matrix **H** for a Rayleigh fading (frequency nonselective) AWGN channel and the corresponding inputs to the detectors for the two receive antennas. 5. Perform feature extraction from a given Image and use Principal Components as image descriptors. 6. By using an image dataset, train a Neural Network to recognize a given Image. Apply this in context to face/object recognition and calculate recognition accuracy of the training set. 7. Develop a Fuzzy Inference System (FIS) by using a set of fuzzy rule base between some key image parameters and calculate output after defuzzification. 8. Design a Fuzzy PID controller using Matlab for a Dc Motor. 9. Classify ECG signals using Neural networks.

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

Course Outcomes:

Course Code	**Course Name**	**Course** Outcomes	**Details**
7EC4-22	**Advance Communication Lab (MATLAB Simulation)**	**CO1**	Understand the features of an communication system and perform basic functions on signals.
		CO2	Explain various methods of generating and detecting different types of code words.
		CO3	Compute various digital communication parameters with the help of graphical representation.
		CO4	Implement fuzzy system and neural networks for different applications.
		CO5	Analyze the effects of sampling on a continuous time signal.

Course Outcome Mapping with Program Outcome:

Course Outcome	**Program Outcomes (PO's)**											
CO. NO.	**Domain Specific**					**Domain Independent**						
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	2	2	1	2	2				2	1		2
CO2	2	2	1	2	2				1	2		3
CO3	2	3	2	1	2				2	1		2
CO4	2	2	1	2	2				1	1		1
CO5	2	2	2	2	1				2	2		2
1: Slight (Low) , 2: Moderate (Medium), 3: Substantial (High)												

INSTRUCTIONS OF LAB

DO’S

1. Student should get the record of previous experiment checked before starting the new experiment.
2. Read the manual carefully before starting the experiment.
3. Before starting the experiment, system checked by the teacher.
4. Get your readings checked by the teacher.
5. Apparatus must be handled carefully.
6. Maintain strict discipline.
7. Keep your mobile phone switched off or in vibration mode.
8. **Students should get the experiment allotted for next turn, before leaving the lab.**

DON’TS

1. Do not touch or attempt to touch the mains power supply wire with bare hands.
2. Do not overcrowd the tables.
3. Do not tamper with equipments.
4. Do not leave the lab without permission from the teacher.

SAFETY MEASURES

1. Antivirus software is installed for protection against viruses and malwares.
2. External storage devices are not allowed to use in lab.
3. At all the times the right procedures while starting and shutting down the computer therefore abrupt switching on and off the computer should be avoided since this can lead to damaging the computer.
4. Any repairs to the computer should be done by someone who has knowledge regarding computer repairs.

INTRODUCTION TO MATLAB

MATLAB is a high performance language for technical computing .It integrates computation visualization and programming in an easy to use environment, MATLAB stands for MATrix LABoratory. It was written originally to provide easy access to matrix software developed by LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is therefore built on a foundation of sophisticated matrix software in which the basic element is matrix that does not require pre dimensioning.

Typical uses of MATLAB-

1. Math and computation
2. Algorithm development
3. Data acquisition
4. Data analysis, exploration and visualization
5. Scientific and engineering graphics

The main features of MATLAB are-

1. Advance algorithm for high performance numerical computation, especially in the field of matrix algebra.
2. A large collection of predefined mathematical functions and the ability to define one's own functions.
3. Two-and three dimensional graphics for plotting and displaying data
4. A complete online help system
5. Powerful, matrix or vector oriented high level programming language for individual applications.
6. Toolboxes available for solving advanced problems in several application areas.

The MATLAB System-

The MATLAB system consists of five main parts:

Development Environment-

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.

The MATLAB Mathematical Function Library-

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix Eigen values, Bessel functions, and fast Fourier transforms.

The MATLAB Language-

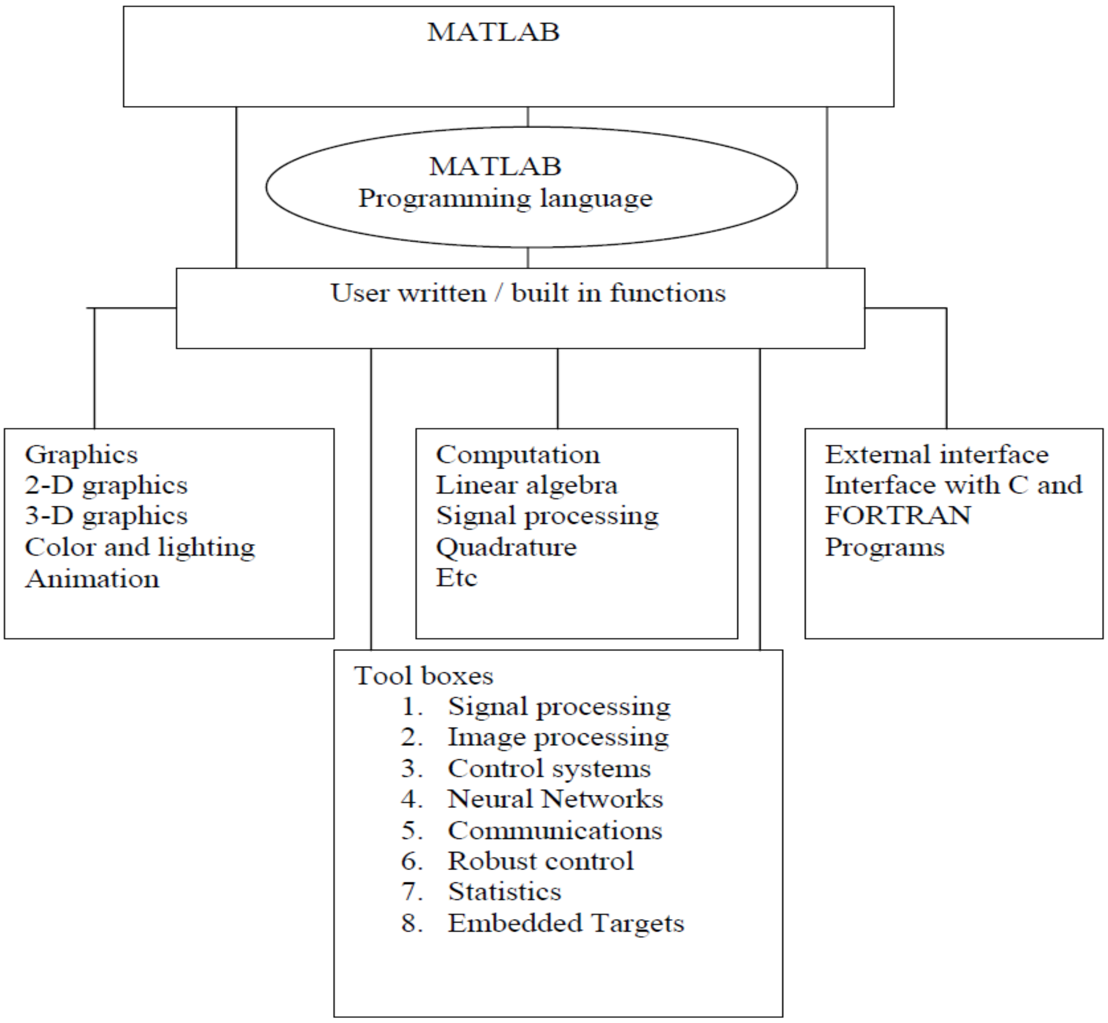
This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs.

Graphics-

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as
Well as to build complete graphical user interfaces on your MATLAB applications.

The MATLAB Application Program Interface (API)-

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.



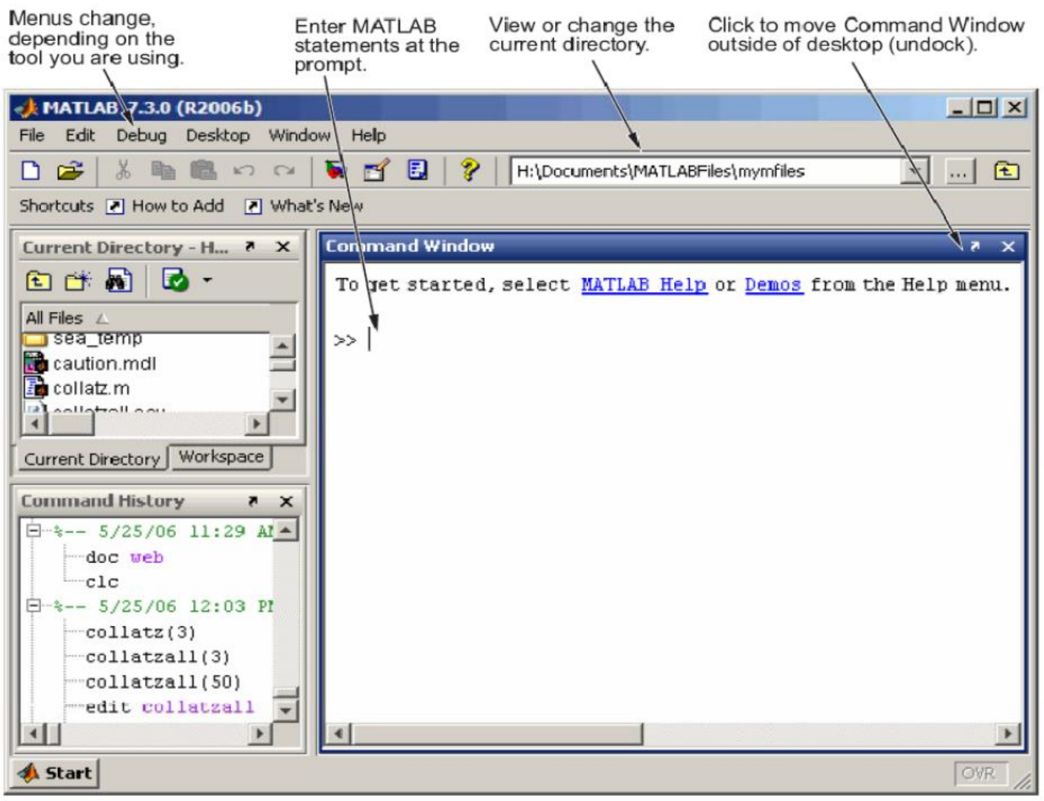
Features and capabilities of MATLAB-

Starting MATLAB-

On Windows platforms, start MATLAB by double-clicking the MATLAB shortcut icon on your Windows desktop. On UNIX platforms, start MATLAB by typing mat lab at the perating system prompt. You can customize MATLAB startup. For example, you can change the directory in which MATLAB starts or automatically execute MATLAB statements in a script file named startup.m.

MATLAB Desktop-

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB. The following illustration shows the default desktop. You can customize the arrangement of tools and documents to suit your needs.


Menus change, depending on the tool you are using.
Enter MATLAB statements at the prompt.
View or change the current directory.
Click to move Command Window outside of desktop (undock).
MATLAB 7.3.0 (R2006b)
File Edit Debug Desktop Window Help
H:\Documents\MATLABFiles\mymfiles
Shortcuts How to Add What's New
Current Directory - H...
All Files
sea_temp
caution.mdl
collatz.m
Current Directory
Workspace
Command History
Command Window
Start
OVR

```
To get started, select MATLAB Help or Demos from the Help menu.

>> |
```

```
%-- 5/25/06 11:29 AM
doc web
clc
%-- 5/25/06 12:03 PM
collatz(3)
collatzall(3)
collatzall(50)
edit collatzall
```

M-Files-

You can create your own matrices using *M-files*, which are text files containing MATLAB code. Use the MATLAB Editor or another text editor to create a file containing the same statements you would type at the MATLAB command Line. Save the file under a name that ends in .m.

File->New->M-File
Write on your Program
Save the M-File

Evaluate the Compilation

OBJECTIVE OF LAB:

It is important to understand the processing of signals. Processing of signals means make suitable form of signal/information according to the need or application. This can be done with the help of filter, converter, multiplier, amplifiers etc. "**Digital Signal Processing**" is the part of this signal processing in which the signal is processed in the digital domain or we can say that in 0 and 1 form. This makes the processing fast and immune to the noise. Digital Signal Processors (DSP) take real-world signals like voice, audio, video, temperature, pressure, or positions that have been digitized and then mathematically manipulate them. A DSP is designed for performing mathematical functions like "add", "Subtract", "multiply" and "divide" very quickly. Signals need to be processed so that the information that they contain can be displayed, analyzed, or converted to another type of signal that may be of use. In the real-world, analog products detect signals such as sound, light, temperature or pressure and manipulate them. Converters such as an Analog-to-Digital converter then take the real-world signal and turn it into the digital format of 1's and 0's. From here, the DSP takes over by capturing the digitized information and processing it. It then feeds the digitized information back for use in the real world. It does this in one of two ways, either digitally or in an analog format by going through a Digital-to-Analog converter. All of this occurs at very high speeds.

This lab will teach the basics concepts involve in the digital processing of signals. It includes many different topics, such as:

- filters
- analysis of signals and systems
- synthesis of signals
- Behavior of LTI systems
- Random sequence like Rayleigh distribution, Gaussian distribution etc.

SCOPE:

In the UG level projects one can design the system by applying the knowledge of DSP. This lab helps to understand all the conditions of processing the signal.

Also in higher studies, the dissertation work includes the area of digital signal processing and image processing speech processing, FPGA programming etc. so strong knowledge of DSP is needed.

Once the basic DSP background is in place, many application-specific DSP areas are accessible. This includes speech processing (recognition, synthesis, and compression), image processing, audio processing (analysis, synthesis, etc.), and all types of biomedical signal processing, and on and on.

ADVANTAGES OF DSP :

The advantages of DSP over Analog Signal Processing are:

1. High Accuracy: The accuracy of the analog filter is affected by the tolerance of the circuit components used for designing the filter, but DSP has superior control of accuracy.

2. Cheaper: The digital realization is much cheaper than the analog realization in many applications.

3. Flexibility in Configuration: For reconfiguring an analog system, we can only do it by redesign of system hardware; where as a DSP System can be easily reconfigured only by changing the program.

4. Ease of Data Storage

5. Time Sharing: The cost of the processing signal can be reduced in DSP by the sharing of a given processor among a number of signals.

APPLICATIONS OF DSP : DSP can be applicable in variety of fields such as

1. Telecommunication

2. Consumer Electronics

3. Image Processing

4. Instrumentation and Control

5. Military Applications

6. Speech Processing

7. Seismology

8. Medicine

LEARNING OBJECTIVES:

The student will be able to:

1. Generate elementary signals/ waveforms and perform arithmetic operations on signals.
2. Calculate and plot DFT / IDFT of given DT signal.
3. Plot frequency response of a given system and verify the properties of LTI system.
4. Implement FFT of given sequence.
5. Implement LP FIR filter for a given sequence and calculate the filter coefficients.
6. Implement HP FIR filter for a given sequence and plot the response of the same.
7. Generate amplitude modulated signal.
8. Find correlation between given sequences.
9. Implement convolution
10. Generate the random signals having different distributions, mean and variance.

PART A

Analog-to-Digital Conversion

EXPERIMENT NO.1

AIM: Generate a sinusoidal signal. Sample and reconstruct a signal through interpolation. Vary the sampling rate below and above the Nyquist rate and hence verify the Sampling theorem

SOFTWARE USED: SCILAB 6.1.0

THEORY: The sampling is extremely important and useful in signal processing and digital communication. The sampling process is usually described in the time domain. With the help of sampling process, a continuous time signal may be completely represented and recovered from the knowledge of samples taken uniformly. A continuous time signal is first converted to discrete time signal by sampling process. The sufficient number of samples of the signal must be taken so that the original signal is represented in its samples completely. Also, it should be possible to recover or reconstruct the original signal completely from its samples.

Sampling Theorem Statement: A continuous time signal may be completely represented in its samples and recovered back if the sampling frequency is $f_s >= 2f_m$. Here f_s is the sampling frequency and f_m is the maximum frequency present in the signal.

The output sample signal is represented by the samples. These samples are maintained with a gap, these gaps are termed as sample period or sampling interval (Ts). And the reciprocal of the sampling period is known as "sampling frequency" or "sampling rate". The number of samples is represented in the sampled signal is indicated by the sampling rate.

Consider a continuous time signal x(t) whose spectrum is band limited to f_m Hz. Sampling of x(t) at a rate of f_s Hz (Fs samples per second) may be achieved by multiplying x(t) by an impulse train. Therefore it is called as ideal sampling. The impulse train consists of unit impulse repeating periodically every T_s seconds, where $T_s = 1/f_s$.
The sampled signal may be written as

$$g(t) = x(nT_s) = x(t).\delta_{Ts}(t)$$

The sampled signal in frequency domain is represented as

$$G(\omega) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} X(\omega - n\omega_s)$$

The spectrum $G(\omega)$ consist of $X(\omega)$ repeating periodically with period ω_s, where

$$\omega_s = \frac{2\pi}{T_s}\ rad/sec\ sec \quad or \quad f_s = \frac{1}{T_s}\ Hz$$

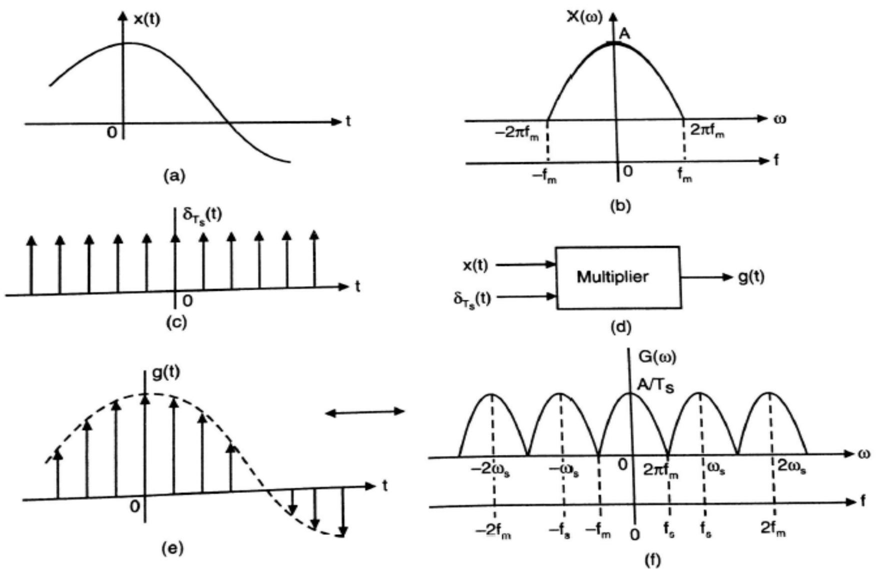


Fig. 1.1 Sampling Process

When the sampling rate exactly equal to $2f_m$ samples per second, then it is called Nyquist rate. Nyquist rate is also called the minimum sampling rate.

$$f_s = 2f_m \quad Hz$$

Similarly, maximum sampling interval is called Nyquist interval

$$Ts = 1/(2f_m) \quad Seconds$$

Signal Reconstruction: The process of reconstructing a continuous time signal x(t) from its samples is known as interpolation. Interpolation gives either approximate or exact recovery of the continuous time signal. A signal x(t) band limited to f_m Hz can be reconstructed (interpolated) completely from its samples, by passing the sampled signal through an ideal low pass filter of cut-off frequency f_m Hz. Therefore the filter output to g(t), which is x(t), may be expressed as

$$x_r(t) = x(t) = \sum_{n=-\infty}^{\infty} x(nT_s) sinc(f_s(t - nT_s))$$

This is known as interpolation formula, which provides values of x(t) between samples as a weighted sum of all the sample values.

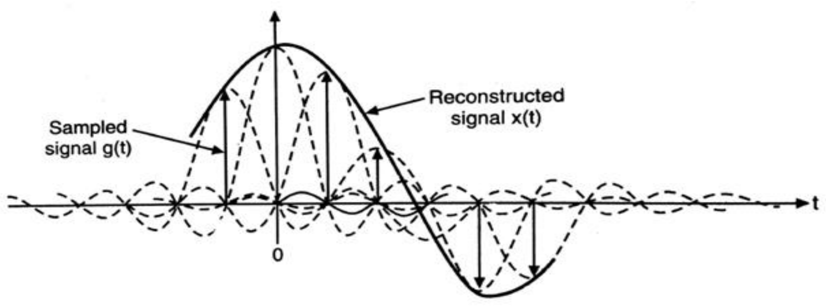


Figure. 1.2: Signal Reconstruction

PROGRAM:

//sampling

clear all;

mtlb_close ;

clc;

t=-10:0.01:10; **// define time t to be a vector running between -10 and 10 in steps of 0.01**

T=4; **//sampling**

clear all;

mtlb_close ;

clc;

```
t=-10:0.01:10; // define time t to be a vector running between -10 and 10 in steps of
0.01

T=4; //time period of message signal

fm=1/T; //frequency of message signal
//generate sinusoidal signal
x=cos(2*%pi*fm*t);
subplot(2,2,1);plot2d(t,x);
xlabel('time');ylabel('amplitude');
mtlb_hold;
subplot(2,2,1);plot(t,x,'linewidth',3);
xlabel('time');ylabel('amplitude');
mtlb_hold;
title('input signal');
// sampling frequencies
fs1=1.6*fm; //less than nyquist rate
fs2=2*fm; //equal to nyquist rate
fs3=10*fm; //greater than nyquist rate
//Sampling theorem verification, fs≥2fm
//CASE 1: UNDERSAMPLING
n1=-4:1:4; //no. of samples
```

```
x1=cos(2*%pi*fm/fs1*n1);

subplot(2,2,2);plot2d3('gnn',n1,x1);


xlabel('number of samples');ylabel('amplitude');

subplot(2,2,2);plot(n1,x1,'linewidth',3);

mtlb_hold;

xgrid;

title('under sampling');
```

//CASE 2: UNIFORM SAMPLING

```
n2=-5:1:5;

x2=cos(2*%pi*fm/fs2*n2);

subplot(2,2,3);plot2d3('gnn',n2,x2);

xlabel('number of samples');ylabel('amplitude');

subplot(2,2,3);plot(n2,x2,'linewidth',3);

mtlb_hold;

xgrid;

title('uniform sampling');
```

//CASE 3: OVERSAMPLING

```
n3=-20:1:20;

x3=cos(2*%pi*fm/fs3*n3);
```

```
subplot(2,2,4);plot2d3('gnn',n3,x3);

//hold on;



xgrid;

subplot(2,2,4);plot(n3,x3,'linewidth',3);

xlabel('number of samples');ylabel('amplitude');

mtlb_hold;

xgrid;

title('over sampling');

//Sample and reconstruct a signal through interpolation

clear all;

mtlb_close ;

clc;

Fs = 5000;

Ts = 1/Fs;

n = -25:1:25; // need length (n) for proper matrix multiplication

nTs = n*Ts;

x = exp(-1000*abs(nTs));

//discrete signal to be reconstructed

//to reconstruct, use an interpolating function (sinc)

//form of xa(t) = summmation from n = n1 -> n2 of x[n]*sinc[Fs(t-n*Ts

Dt = 0.00005;
```

```
t = -0.005:Dt:0.005; // define time interval

xa_t = x * sinc(Fs*(ones(length(n),1)*t-nTs'*ones(1,length(t))));

figure(2);

subplot(2,1,1)

plot2d3(n,x);

xlabel('number of samples');

ylabel('amplitude');

subplot(2,1,2)
```

//approximate analog signal

```
plot2d(t*1000,xa_t);

xlabel('time in ms');

ylabel('xa(t)');

title('signal reconstructed using sinc function');

mtlb_hold;
```

OUTPUT:

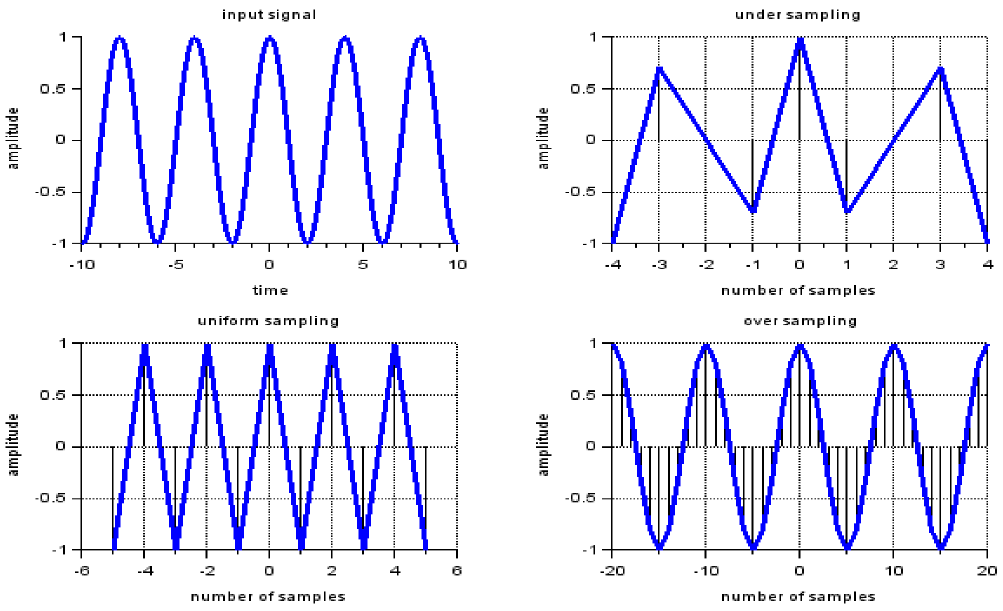


Figure 1.2 a) Input Signal b) Under Sampling c) Uniform Sampling d) Over Sampling

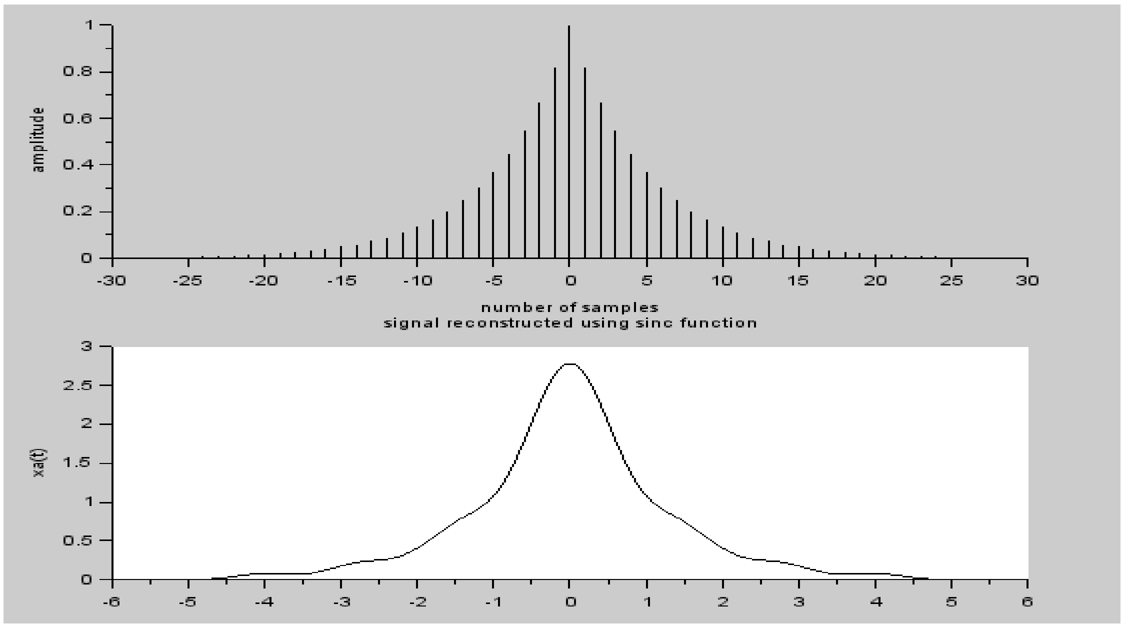


Figure1.3 Signal Reconstruction using Interpolation

Result: Sinusoidal signal is generated as shown in Figure 1.2 and is sampled and reconstructed through interpolation. Sampling theorem is verified by varying the sampling rate below and above the Nyquist rate.

Discussion:

Q 1. Explain aliasing effect?

Q 2. Define Nyquist rate and Nyquist interval.

Q 3. Explain demerits of Ideal sampling.

Q 4. What is aperture effect?

EXPERIMENT NO.2

AIM: Generate a sequence of length 500 of zero-mean, unit variance Gaussian random variables. Using a uniform PCM scheme, quantize this sequence to 16, 64 and 128 levels. (a). Find and compare the resulting signal-to-quantization noise ratios. (b). Find the first ten values of the sequence, the corresponding quantized values and the corresponding code words for each case. (c). Plot the quantization error and the quantized value as a function of the input value for each case.

SOFTWARE USED: SCILAB 6.1.0

THEORY:

Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. It is the standard form of digital audio in computers, compact discs, digital telephony and other digital audio applications. In a PCM stream, the amplitude of the analog signal is sampled regularly at uniform intervals, and each sample is quantized to the nearest value within a range of digital steps. the three steps for developing an equivalent PCM digital signal from an analog signal are

1. Sampling: Sampling allows for the conversion of an analog signal into a digital signal after a quantization is made on the amplitude of the samples.
2. Quantization: Quantizing is the process of rounding off the sample value to the closest permissible discret value (quantizing level). Quantization is done by dividing the range of possible values of the analog samples into some different levels and assigning the center value of each level to any sample in the quantization interval. Quantization approximates the analog sample values with the nearest quantization values. So almost all the quantized samples will differ from the original samples by a small amount. That amount is called quantization error.
3. Coding The encoder encodes the quantized samples. Each quantized sample is encoded into codeword

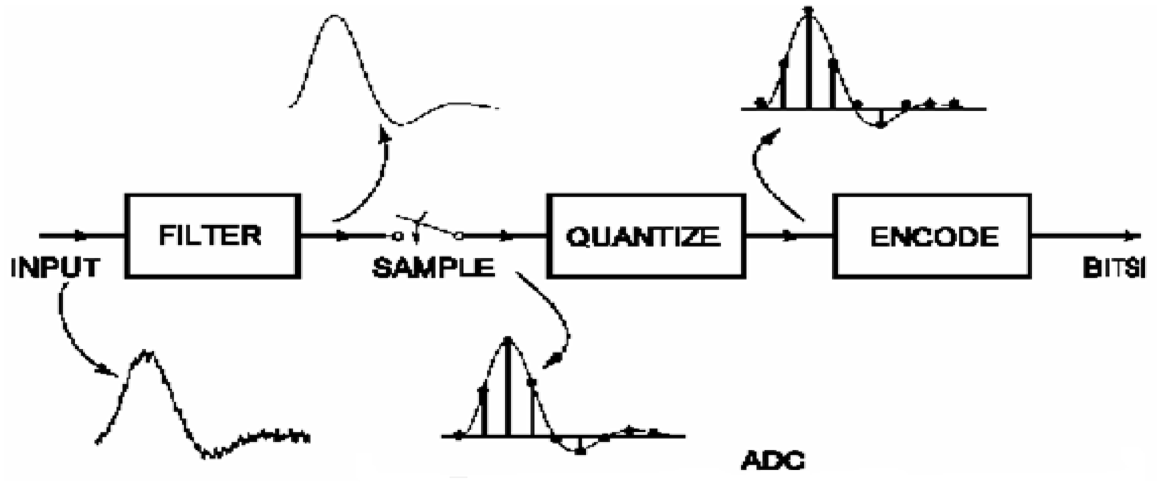


Figure 2.1 Block Diagram of PCM

PROGRAM

```
clc;
clear all ;
close;
x=rand(1,500, "normal");//---PART 1
n=input("Enter the value of n=")//--PARTII
n= 4,6,7
figure(1)
histplot (100 , x);
SNQ =1.76+(6.02* n);//---PART-3
printf("\n\n a.Signal to Noise ratio = %.2f ",SNQ);
disp(SNQ);
figure(2)//-----PART3
//part4
x1= x(1:10);
printf("\n\n b.First 10 random signal ",x1);
disp(x1);
//pcm
xmin=min(x1);
```

```
xmax=max(x1);
l=2^n;//16
l1=l-1,    //15
printf("\n\n c.number of Quantize Level",l);
s=(xmax-xmin)/l1;  //step size
v=log10(l)/log10(2);//no.of bits
x2=[];
for i=0:l1
    x2=[x2, xmin+i*s ]; //0-15:   [0,2,3,4,1,5,7,8,]
x3=gsort(x1, 'g', 'i')//sorting of levels  0,1,2,3,4,-15
xq = x1/xmax;//normalization
en_code = xq;//codewords
d = 2/l;   // 1-2,2-3
q = d*[0:l-1];
q = q-((l-1)/2)*d;
for i = 1:l
    xq(find(((q(i)-d/2)<= xq)&(xq<=(q(i)+d/2))))=q(i).*ones(1,length(find(((q(i)-
d/2)<=xq)&(xq<=(q(i)+d/2)))))
    en_code(find(xq == q(i)))= (i-1).*ones(1,length(find(xq == q(i))))
end
  xq = xq*xmax//normal data
  for i=1:length(x1)
  SQNR(i) = 20*log10(abs((x1(i))/(x1(i)-xq(i))))
  disp(SQNR);
  end
  plot2d3(x1, SQNR);     //----part-5
end
```

Output:

```
Enter the value of n=5

 a.Signal to Noise ratio = 25.84

 b.First 10 random signal

column 1 to 5

-1.3762437  -0.3237197  -0.2674413   0.7234039  -0.9106784

column 6 to 10

 -0.9222671   1.4572251  -0.8909827   0.9867483  -0.5393308
```

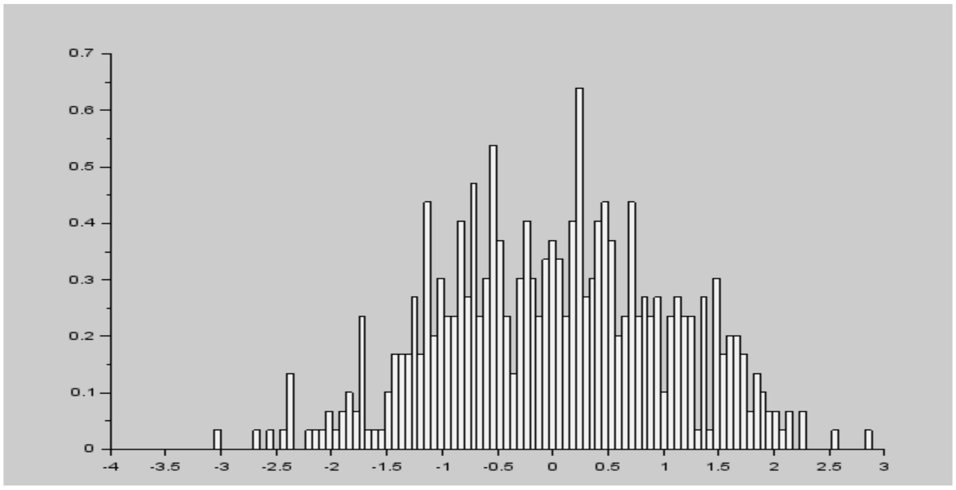


Figure 2.1

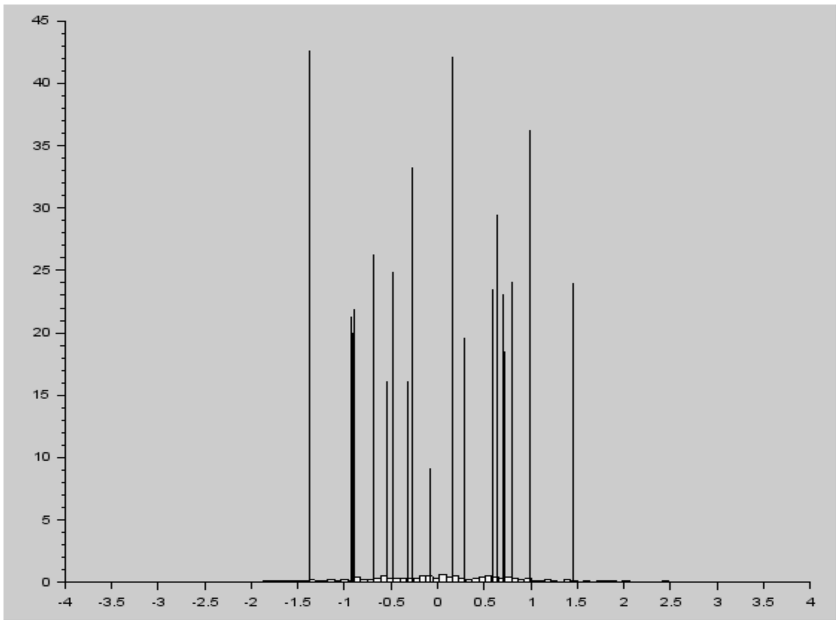


Figure 2.2

RESULT: A sequence of length 500 of zero-mean, unit variance Gaussian random variables is generated and a uniform PCM scheme is quantized to sequence 16, 64 and 128 levels and their resulting signal-to-quantization noise ratios are compared . The quantization error and the quantized value as a function of the input value for each case has been plot.

DISCUSSION:

Q1. Define PCM?

Q2. What are the applications of PCM?

Q3. What are the advantages and disadvantages of PCM?

Q4.What is Signal to Quantization noise ratio of PCM

Q5. Differentiate Uniform and Non uniform Quantization techniques.

Digital Modulation Techniques

EXPERIMENT NO.3

AIM: Simulate the transmitter and receiver for QPSK. Plot the signal and signal constellation diagram. Plot the average probability of symbol error as a function of SNR Eb/No, where Eb is the transmitted energy per bit and No/2 is the double sided power spectral density of additive white Gaussian noise (AWGN) with zero mean.

SOFTWARE USED: MATLAB 7.1

THEORY:

Quadrature Phase Shift Keying (QPSK) is a form of Phase Shift Keying in which two bits are modulated at once, selecting one of four possible carrier phase shifts (0, 90, 180, or 270 degrees). QPSK allows the signal to carry twice as much information as ordinary PSK using the same bandwidth.

QPSK is used for satellite transmission of MPEG2 video, cable modems, videoconferencing, cellular phone systems, and other forms of digital communication over an RF carrier.

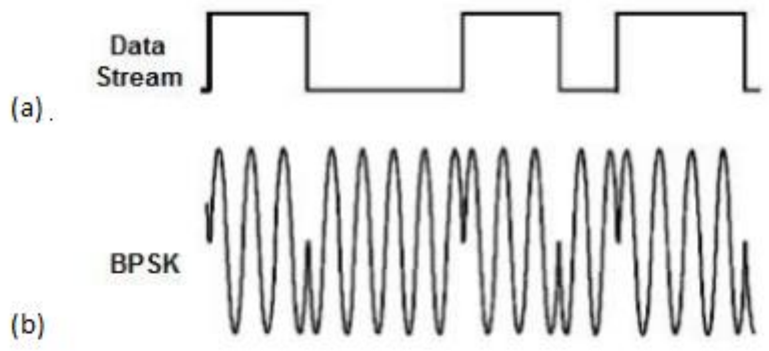


Figure 3.1 (a) Data Stream (b) BPSK signal

A QPSK signal can be generated by independently modulating two carriers in quadrature as shown in Figure 3.2.

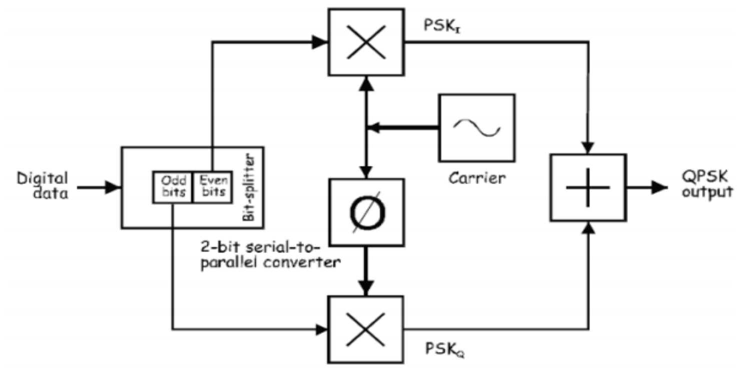


Figure 3.2 Block Diagram of Mathematical Implementation of QPSK

At the input to the modulator, the digital data's even bits (that is, bits 0, 2, 4 and so on) are stripped from the data stream by a "bit-splitter" and are multiplied with a carrier to generate a BPSK signal (called PSKI). At the same time, the data's odd bits (that is, bits 1, 3, 5 and so on) are stripped from the data stream and are multiplied with the 90° phase-shifted carrier to generate a second BPSK signal (called PSKQ). The two BPSK signals are then simply added together for transmission. Figure 3.3 illustrates this procedure to generate a QPSK signal.

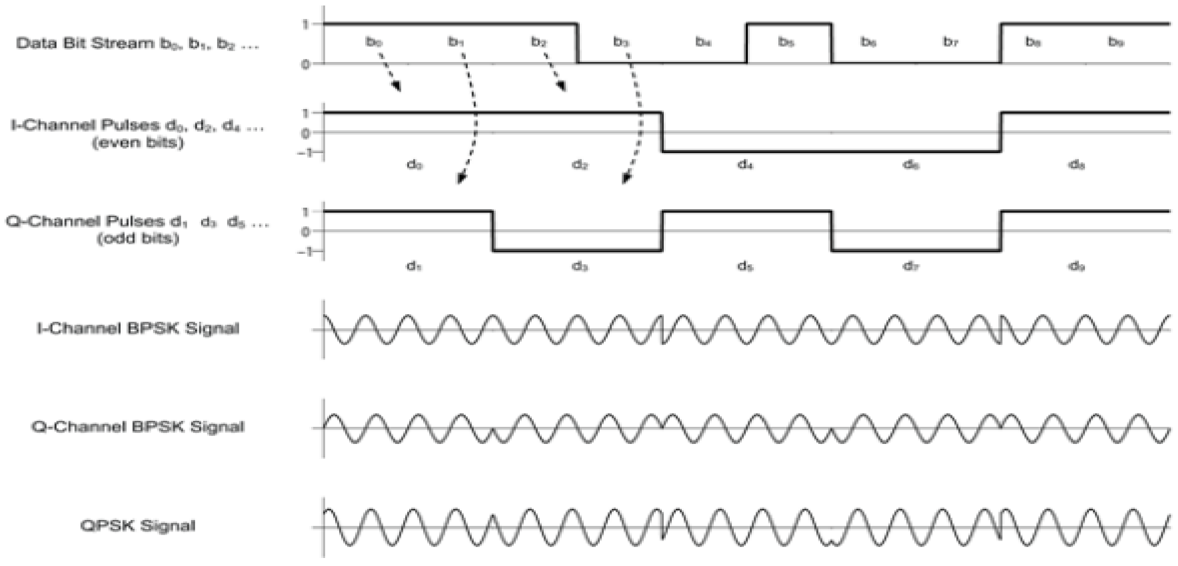


Figure 3.3 QPSK signal generation from two BPSK signals.

The 90^o phase separation between the carriers allows the sidebands to be separated by the receiver using phase discrimination. Figure 3.4 shows the block diagram of the mathematical implementation of QPSK demodulation.

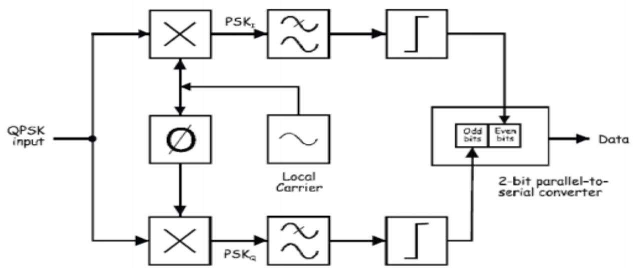


Figure 3.4 Block Diagram of Mathematical Implementation of QPSK demodulation

The arrangement uses two product detectors to simultaneously demodulate the two BPSK signals. This simultaneously recovers the pairs of bits in the original data. The two signals are cleaned-up using a comparator or some other signal conditioner then the bits are put back in order using a 2-bit parallel-to-serial converter.

PROGRAM

PART 1: QPSK TRANSMITTER AND RECEIVER

```
clc;
clear all;
close all;
data=[0 0 1 1 0 1 1 0 1 1 1 0]; % information
figure(1)
stem(data, 'linewidth',3), grid on;
title('  Information before Transmiting ');
axis([ 0 11 0 1.5]);
data_NZR=2*data-1; % Data Represented at NZR form for QPSK modulation
s_p_data=reshape(data_NZR,2,length(data)/2);  % S/P convertion of data


br=10.^6; %Let us transmission bit rate  1000000
```

```
f=br; % minimum carrier frequency
T=1/br; % bit duration
t=T/99:T/99:T; % Time vector for one bit information
y=[];
y_in=[];
y_qd=[];
d=zeros(1,length(data)/2);
for i=1:length(data)/2
   p=data(2*i);
   imp=data(2*i - 1);
   y1=s_p_data(1,i)*cos(2*pi*f*t); % inphase component
   y2=s_p_data(2,i)*sin(2*pi*f*t) ;% Quadrature component
   y_in=[y_in y1]; % inphase signal vector
   y_qd=[y_qd y2]; %quadrature signal vector
   y=[y y1+y2]; % modulated signal vector
   if (imp == 0) && (p == 0)
     d(i)=exp(j*pi/4);%45 degrees
   end
   if (imp == 1)&&(p == 0)
      d(i)=exp(j*3*pi/4);%135 degrees
   end
   if (imp == 1)&&(p == 1)
      d(i)=exp(j*5*pi/4);%225 degrees
   end
   if (imp == 0)&&(p == 1)
      d(i)=exp(j*7*pi/4);%315 degrees
   end
end
Tx_sig=y; % transmitting signal after modulation



qpsk=d;
```

```
tt=T/99:T/99:(T*length(data))/2;
figure(2)
subplot(3,1,1);
plot(tt,y_in,'linewidth',3), grid on;
title(' wave form for inphase component in QPSK modulation ');
xlabel('time(sec)');
ylabel(' amplitude(volt0');
subplot(3,1,2);
plot(tt,y_qd,'linewidth',3), grid on;
title(' wave form for Quadrature component in QPSK modulation ');
xlabel('time(sec)');
ylabel(' amplitude(volt0');
subplot(3,1,3);
plot(tt,Tx_sig,'r','linewidth',3), grid on;
title('QPSK modulated signal (sum of inphase and Quadrature phase signal)');
xlabel('time(sec)');
ylabel(' amplitude(volt0');
Rx_data=[];
Rx_sig=Tx_sig; % Received signal
for(i=1:1:length(data)/2)
   %%XXXXXX inphase coherent dector XXXXXXX
   Z_in=Rx_sig((i-1)*length(t)+1:i*length(t)).*cos(2*pi*f*t);
   % above line indicat multiplication of received & inphase carred signal
   Z_in_intg=(trapz(t,Z_in))*(2/T);% integration using trapizodial rull
   if(Z_in_intg>0) % Decession Maker
      Rx_in_data=1;
   else
     Rx_in_data=0;
   end

   %%XXXXXX Quadrature coherent dector XXXXXX
   Z_qd=Rx_sig((i-1)*length(t)+1:i*length(t)).*sin(2*pi*f*t);
```

```
    %above line indicat multiplication ofreceived & Quadphase carred signal
    Z_qd_intg=(trapz(t,Z_qd))*(2/T);%integration using trapizodial rull
        if (Z_qd_intg>0)% Decession Maker
        Rx_qd_data=1;
        else
       Rx_qd_data=0;
        end
        Rx_data=[Rx_data  Rx_in_data  Rx_qd_data]; % Received Data vector
end
figure(3)
stem(Rx_data,'linewidth',3)
title('Information after Receiveing ');
axis([ 0 11 0 1.5]), grid on;
figure(4);
plot(d,'o');%plot constellation without noise
axis([-2 2 -2 2]);
grid on;
xlabel('real'); ylabel('imag');
title('QPSK constellation');
```

OUTPUT:

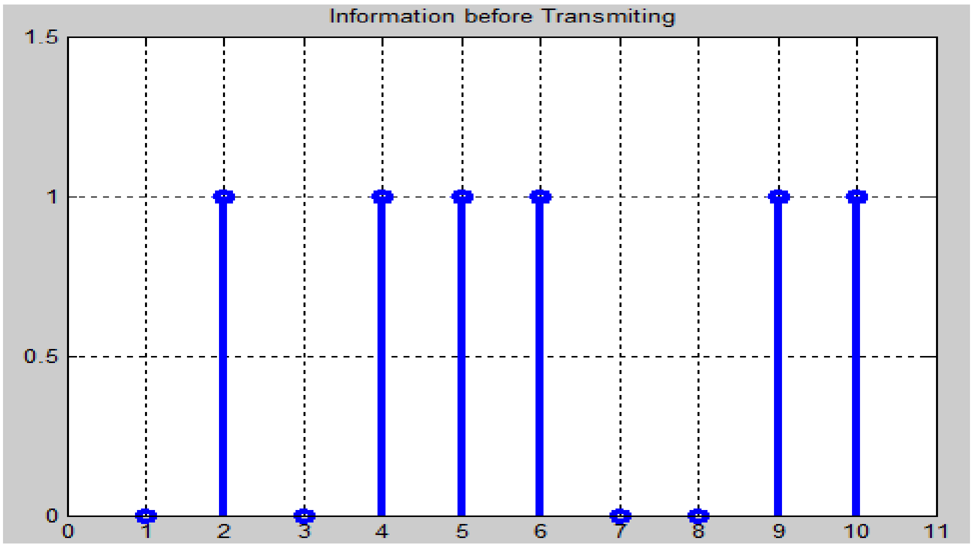


Figure3.5 Information before Transmitting

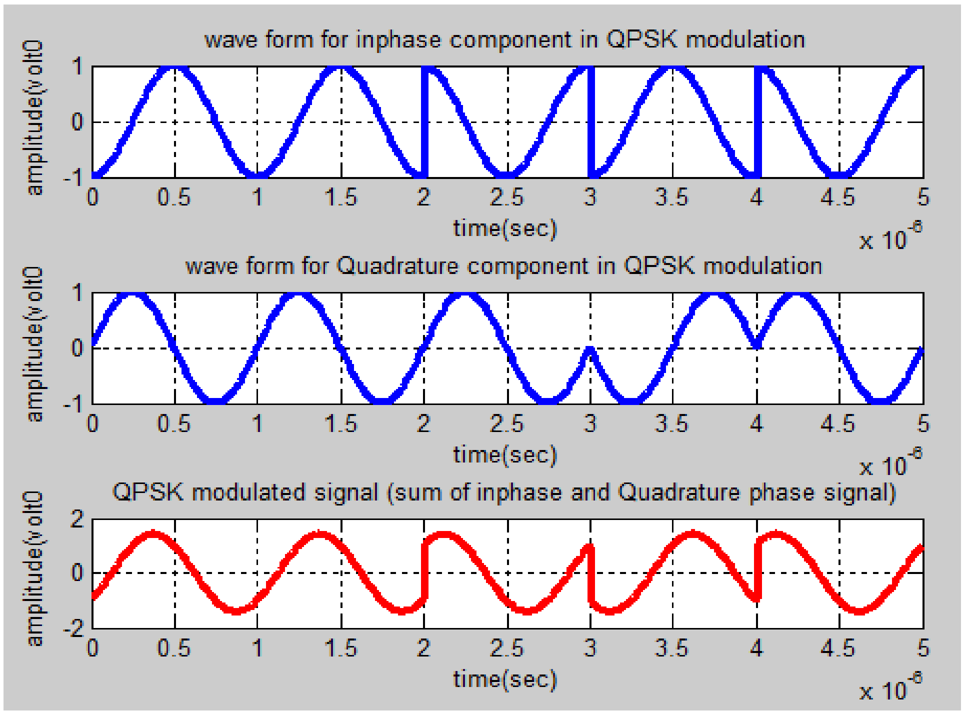


Figure3.6 QPSK Modulation

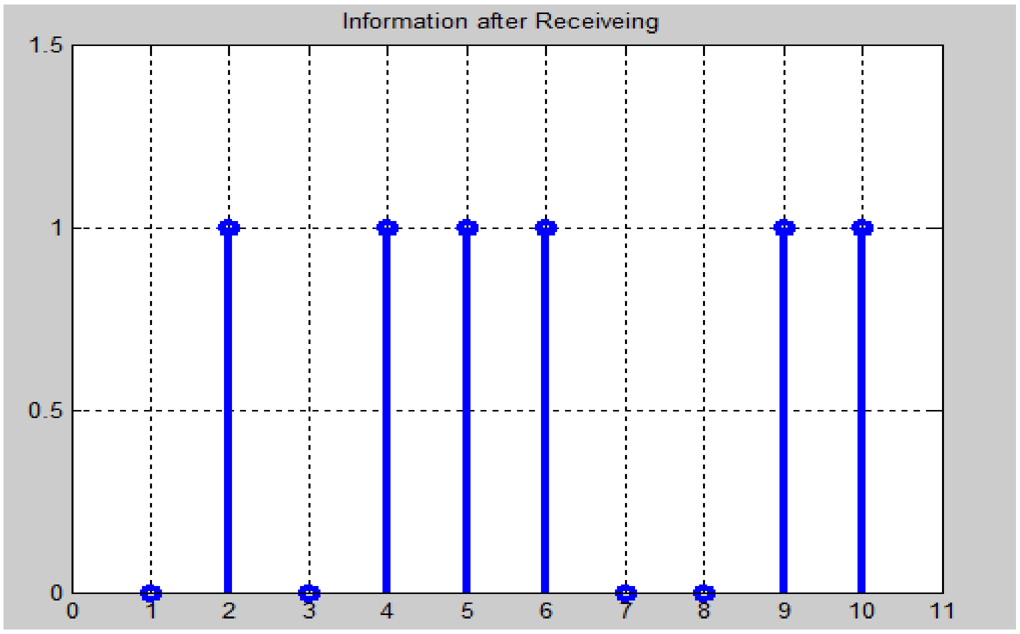


Figure3.7 Information after Receving

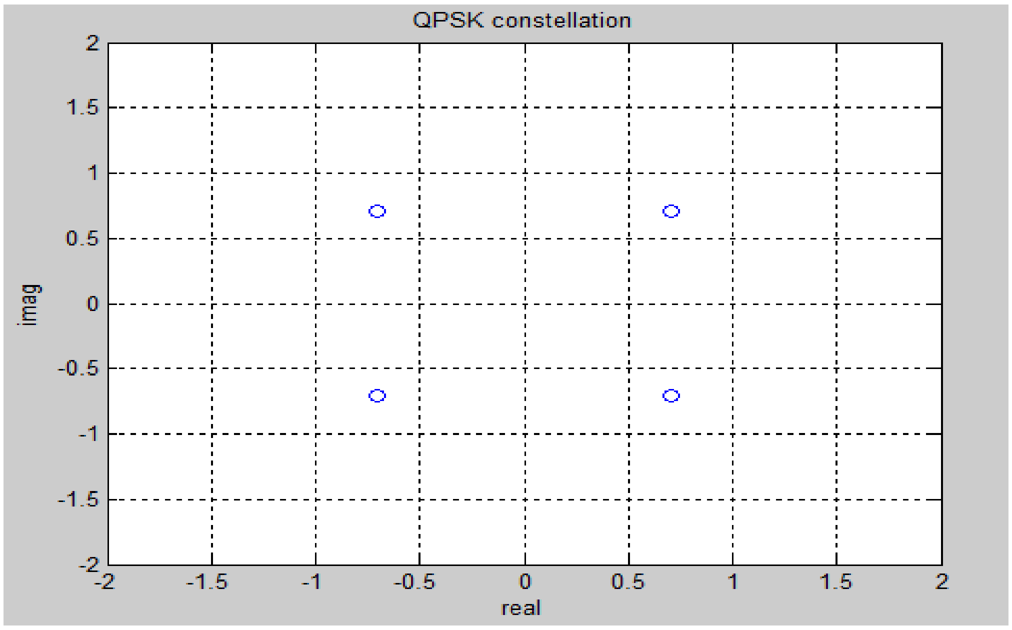


Figure3.8: QPSK Constellation

Part II: Average probability of symbol error as a function of SNR Eb/No

PROGRAM

```
clc;
clear;
close all;
% Definition of system parameters
Fs = 44100; % Sampling frequency
Fc = 4000; % Carrier frequency
Ts = 2.3*(10^-3); % Symbol time
Nb = 50000; % Number of input bits
beta = 3/100; % Attenuation factor
Ns = round(Fs*Ts); % Number of samples
Tsam = 1/Fs; % Sampling time
T = 0:Tsam:(Ns-1)*Tsam; % Total time
mho = 23*10^-5; % Time delay
N = 0; % Initial noise

%================================ Transmitter Part
==============================

% Pulse, bits and symbols generation
p = sin(pi/Ts*T); % Pulse
norm = 1/sqrt(1/Fs*sum(p.^2)); % Pulse normalization
P = norm*p; % Normalized pulse
B = randi([0 1],1,Nb); % Random bits
S = zeros(1,length(B)/2); % Symbols
% Mapping bits into QPSK symbols by Gray Coding method
for n = 1:length(B)/2
   if (B(2*n-1) == 0) && (B(2*n) == 0)

      S(n) = 1+1i;
   elseif (B(2*n-1) == 1) && (B(2*n) == 0)
```

```
        S(n) = -1+1i;
    elseif (B(2*n-1) == 1) && (B(2*n) == 1)
        S(n) = -1-1i;
    elseif (B(2*n-1) == 0) && (B(2*n) == 1)
        S(n) = 1-1i;
    end
end
% Pulse train production
PtI = zeros(Nb/2,length(P)); % create box of zeros to input real values
PtQ = zeros(Nb/2,length(P)); % create box of zeros to input imaginary values
for n = 1:length(B)/2
    PtI(n,1:length(P)) = P*real(S(n)); % a matrix which has different pulses
    PtQ(n,1:length(P)) = P*imag(S(n));
end
PtI1 = reshape(vec2mat(PtI,1),[1,(length(S)*length(P))]);
PtQ1 = reshape(vec2mat(PtQ,1),[1,(length(S)*length(P))]);
Tc = 0:Tsam:Ts*length(S); % Time of carrier signal
% Carrier modulation
CsI = sqrt(2).*cos(2*pi*Fc*Tc);  % Real part
CsQ = sqrt(2).*-sin(2*pi*Fc*Tc); % Imaginary part
CsI = CsI(1:length(PtI1));
CsQ = CsQ(1:length(PtQ1));
SgI = CsI.*PtI1;
SgQ = CsQ.*PtQ1;
Tx = SgI+SgQ; % Transmitting signal

%================================ Channel Part
================================

Mho = round(Fs*mho);
for n = Mho+1:Ns*Nb/2
    H(n) = (Tx(n)*(sqrt(1-beta^2))+Tx(n-Mho)*beta+N);
```

```
end
% SNR calculations
SNR_dB = 0:0.1:10;
SNR_lin = 10.^(SNR_dB/10);
Es = sum(abs(H.^2))./length(H); % Energy of transmitted signal
Eb = Es*length(P)/2; % Energy per bit
for n = 1:length(SNR_lin)
   No = Eb/SNR_lin(n);
   % Adding White Gaussian Noise into the channel
   WGN = sqrt(No/2).*randn(1,length(H));
   Tr = H+WGN; % Transmitted signal

%================================== Receiver Part
==================================
   % Demodulation
   RsI = CsI.*Tr; % Real part
   RsQ = CsQ.*Tr; % Imaginary part
   % Match filtering
   MfI = conv(RsI,P);
   MfQ = conv(RsQ,P);
   % Sampling
   SamI = MfI(100:Ns:end);
   SamQ = MfQ(100:Ns:end);
   % Building symbols
   SymI = sign(SamI(1:1:end));
   SymQ = 1i*sign(SamQ(1:1:end));
   Sym = SymI+SymQ;

   % Mapping symbols to bits
   R = zeros(1,Nb); % Create a box of zeros to put the bits inside it
   for m = 1:Nb/2
      if Sym(m) == 1+1i
```

```
          R(2*m-1) = 0;
          R(2*m) = 0;
        elseif Sym(m) == -1+1i
          R(2*m-1) = 1;
          R(2*m) = 0;
        elseif Sym(m) == -1-1i
          R(2*m-1) = 1;
          R(2*m) = 1;
        elseif Sym(m) == 1-1i
          R(2*m-1) = 0;
          R(2*m) = 1;
        end
    end
    Be = sum(R~=B); % Number of erorrs into received bits
    Pb_simulated(n) = Be/length(B); % Bit error probabilty
    Pb_theoretical(n) = qfunc(sqrt(2*SNR_lin(n))); % Bit error probabilty
end
figure;
semilogy(SNR_dB,Pb_simulated,'-x',SNR_dB,Pb_theoretical);
xlabel('Eb/No in dB');
ylabel('Pb');
title('Bit Error Probability, Pb vs Eb/No');
legend('Pb simulated','Pb theoretical');
grid on;
```

OUTPUT:

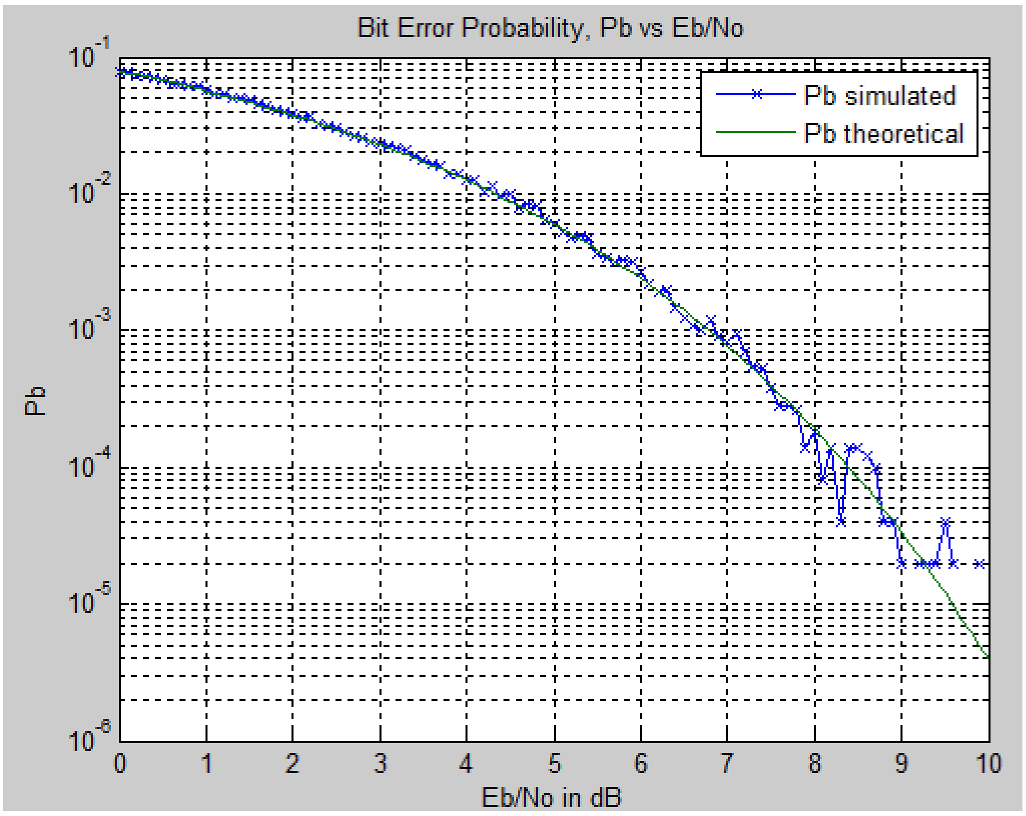


Figure3.9 Bit Error Probabity vs Eb/No

Result: Transmitter and receiver for QPSK is generated as shown in Figure3.5,3.6,3.7 and QPSK constellation diagram is generated as shown in Figure 3.8. The average probability of symbol error as a function of SNR Eb/No is shown in Fig 3.9.

Discussion

Q1. What is QPSK?

Q2. Which type of synchronization is used in QPSK?

Q3. What are the advantages of QPSK?

Q4.How many phases are transmitted in QPSK.

Q5. What is the phase difference between adjacent messages in QPSK

EXPERIMENT NO.4

AIM: Simulate the c. Plot the signal and signal constellation diagram. Plot the average probability of symbol error as a function of SNR Eb/No, where Eb is the transmitted energy per bit and No/2 is the double sided power spectral density of additive white Gaussian noise (AWGN) with zero mean.

SOFTWARE USED: MATLAB 7.1

THEORY:

Quadrature Amplitude Modulation (QAM) conveys two bit streams by changing the amplitude of two carrier waves that have the same frequency and a 90° shift. The most common type of QAM modulation is rectangular QAM, were the constellation points are arranged in a square grid. Depending on the desired number of bits per symbol (4, 5, 6 …), we have 16QAM, 32QAM, 64 QAM, etc.…

The constellation for 16 QAM is shown in Figure 4.1

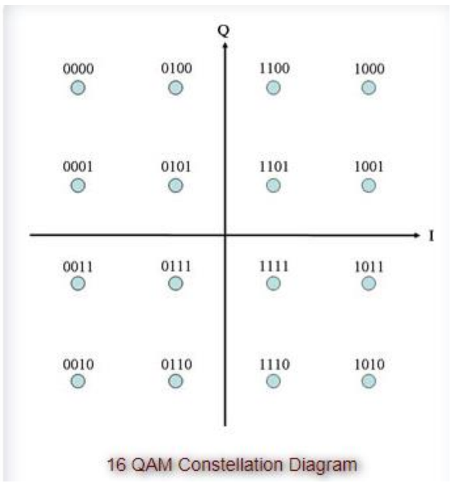


Figure 4.1

As in QPSK, there are two ways (codes) to map the symbols to the constellation points: Binary code and Gray code. In Gray code, two adjacent symbols differ in one bit, while in Binary code, two adjacent symbols may differ in 2 bits. Therefore, Gray code is preferable over Binary code, since if a receiver maps a symbol to one of its adjacent symbols (due to noise or errors), it will lead to 1 wrong bit instead of 2.

The demodulator maps the received signal (possibly distorted due to noise in the channel) back to bit streams.

For 16 QAM, the Bit Error Rate (BER) is the same as BPSK.

$$BER = \frac{3}{4} Q\left(\sqrt{\frac{4E_b}{N_0}}\right)$$

Since in QAM modulation two carriers are used, the Symbol Error Rate per carrier is given by:

$$P_{sc} = \frac{6}{4} Q\left(\sqrt{\frac{E_s}{5N_0}}\right)$$

And the total Symbol Error Rate is given by:

$$P_s = 1 - (1 - P_s c)^2$$

Where N0/2 is the noise power spectral density, and Q(.) is the Q function of the Gaussian distribution.

Part I Transmitter & Receiver of 16-QAM

PROGRAM:

```
clc;
close all;
clear all;
M = 16; % Modulation order (alphabet size or number of points in signal constellation)
k = log2(M); % Number of bits per symbol
n = 30000; % Number of bits to process
sps = 1; %  Number of samples per symbol (oversampling factor)
rng default;% Controls the random number generation.
dataIn = randi([0 1],n,1);% Generates a random binary data stream.
stem(dataIn(1:40),'filled');%stem plot to show the binary values for the first 40 bits of the
random binary data stream
title('Random Bits');
xlabel('Bit Index');
ylabel('Binary Value')
%Perform a bit-to-symbol mapping by first reshaping the data into binary k-tuples, where
k is the number of bits per symbol defined by k=log2(M).
%Then, use the bi2de function to convert each 4-tuple to an integer value.
dataInMatrix = reshape(dataIn,length(dataIn)/k,k);
dataSymbolsIn = bi2de(dataInMatrix);
figure;
stem(dataSymbolsIn(1:10));%Plot the first 10 symbols in a stem plot
title('Random Symbols');
xlabel('Symbol Index');
ylabel('Integer Value');
%Use the qammod function to apply 16-QAM modulation to the dataSymbolsIn column
vector for natural-encoded and Gray-encoded binary bit-to-symbol mappings.
dataMod = qammod(dataSymbolsIn,M,'bin'); % Binary coding with phase offset of zero

dataModG = qammod(dataSymbolsIn,M);% Gray coding with phase offset of zero
```

```
%Calculate the SNR when the channel has an Eb/N0 of 10 dB
EbNo = 10;
snr = EbNo+10*log10(k)-10*log10(sps);
%Pass the signal through the AWGN channel for the binary and Gray coded symbol
mappings.
receivedSignal = awgn(dataMod,snr,'measured');
receivedSignalG = awgn(dataModG,snr,'measured');
sPlotFig = scatterplot(receivedSignal,1,0,'g.');
hold on
scatterplot(dataMod,1,0,'k*',sPlotFig)%constellation diagram of 16 qam
%Receiver of QAM
dataSymbolsOut = qamdemod(receivedSignal,M,'bin');%demodulation of qam
dataSymbolsOutG = qamdemod(receivedSignalG,M);
dataOutMatrix = de2bi(dataSymbolsOut,k);
dataOut = dataOutMatrix(:); % Return data in column vector
dataOutMatrixG = de2bi(dataSymbolsOutG,k);
dataOutG = dataOutMatrixG(:); % Return data in column vector
%The biterr function calculates the bit error statistics from the original binary data
stream, dataIn, and the received data streams, dataOut and dataOutG.
%Gray coding significantly reduces the BER.
[numErrors,ber] = biterr(dataIn,dataOut);
fprintf('\nThe binary coding bit error rate is %5.2e, based on %d errors.\n',ber,numErrors)
[numErrorsG,berG] = biterr(dataIn,dataOutG);
fprintf('\nThe Gray coding bit error rate is %5.2e, based on %d errors.\n',
berG,numErrorsG);
```

OUTPUT

The binary coding bit error rate is 2.40e-03, based on 72 errors.

The Gray coding bit error rate is 1.33e-03, based on 40 errors.

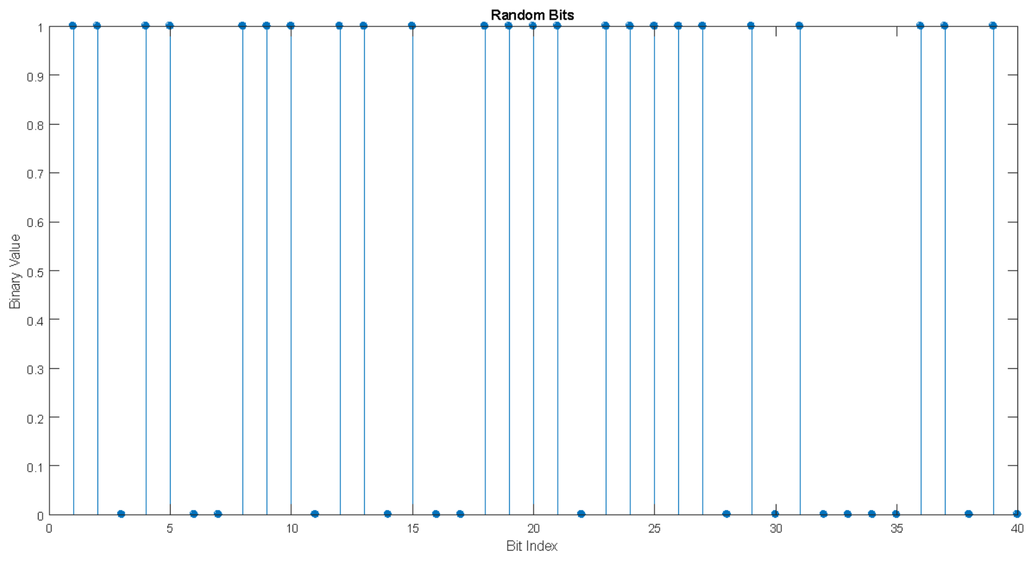


Figure 4.2: **first 40 bits of the random binary data stream**

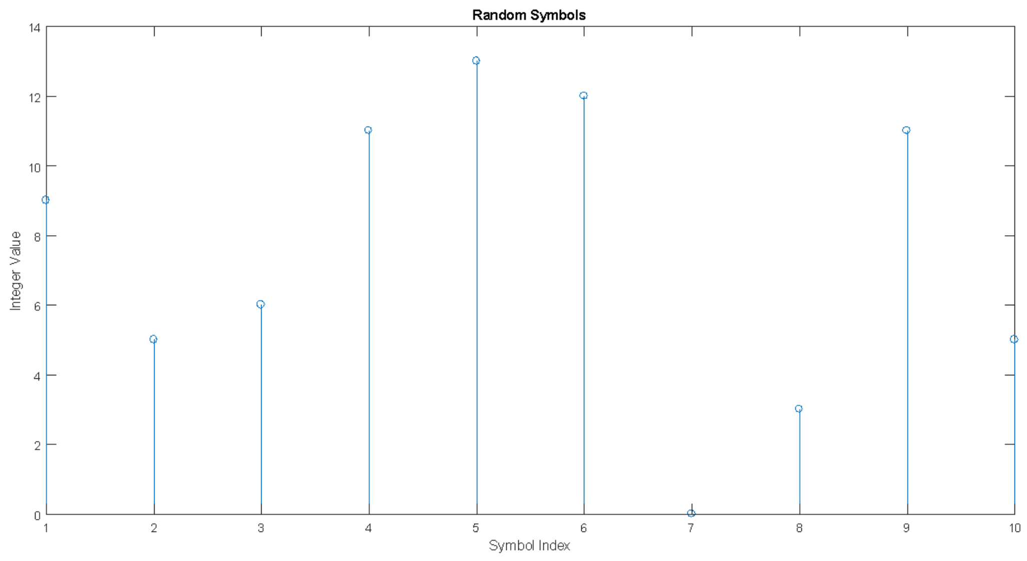


Figure 4.3: **first 10 symbols**

Constellation Diagram:

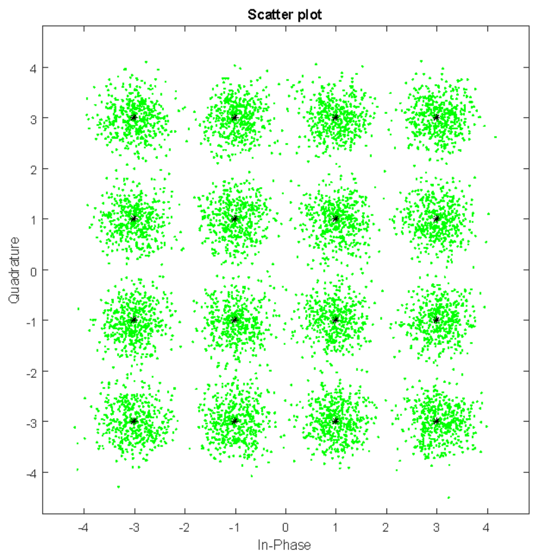


Figure 4.4 Constellation Diagram

Part II: Symbol error rate for 16-QAM modulation

PROGRAM

```
clear
N = 2*10^5; % number of symbols
alpha16qam = [-3 -1 1 3]; % 16-QAM alphabets
Es_N0_dB = [0:20]; % multiple Es/N0 values
ipHat = zeros(1,N);
for ii = 1:length(Es_N0_dB)
   ip = randsrc(1,N,alpha16qam) + j*randsrc(1,N,alpha16qam);
   s = (1/sqrt(10))*ip; % normalization of energy to 1
   n = 1/sqrt(2)*[randn(1,N) + j*randn(1,N)]; % white guassian noise, 0dB variance
   y = s + 10^(-Es_N0_dB(ii)/20)*n; % additive white gaussian noise
```

```
    % demodulation
    y_re = real(y); % real part
    y_im = imag(y); % imaginary part
    ipHat_re(find(y_re< -2/sqrt(10)))          = -3;
    ipHat_re(find(y_re > 2/sqrt(10)))          =  3;
    ipHat_re(find(y_re>-2/sqrt(10) & y_re<=0))  = -1;
    ipHat_re(find(y_re>0 & y_re<=2/sqrt(10)))   =  1;
    ipHat_im(find(y_im< -2/sqrt(10)))          = -3;
    ipHat_im(find(y_im > 2/sqrt(10)))          =  3;
    ipHat_im(find(y_im>-2/sqrt(10) & y_im<=0))  = -1;
    ipHat_im(find(y_im>0 & y_im<=2/sqrt(10)))   =  1;
    ipHat = ipHat_re + j*ipHat_im;
    nErr(ii) = size(find([ip- ipHat]),2); % couting the number of errors
end
simBer = nErr/N;
theoryBer = 3/2*erfc(sqrt(0.1*(10.^(Es_N0_dB/10))));
close all
figure
semilogy(Es_N0_dB,theoryBer,'b.-','LineWidth',2);
hold on
semilogy(Es_N0_dB,simBer,'mx-','Linewidth',2);
axis([0 20 10^-5 1])
grid on
legend('theory', 'simulation');
xlabel('Es/No, dB')
ylabel('Symbol Error Rate')
title('Symbol error probability curve for 16-QAM modulation')
```

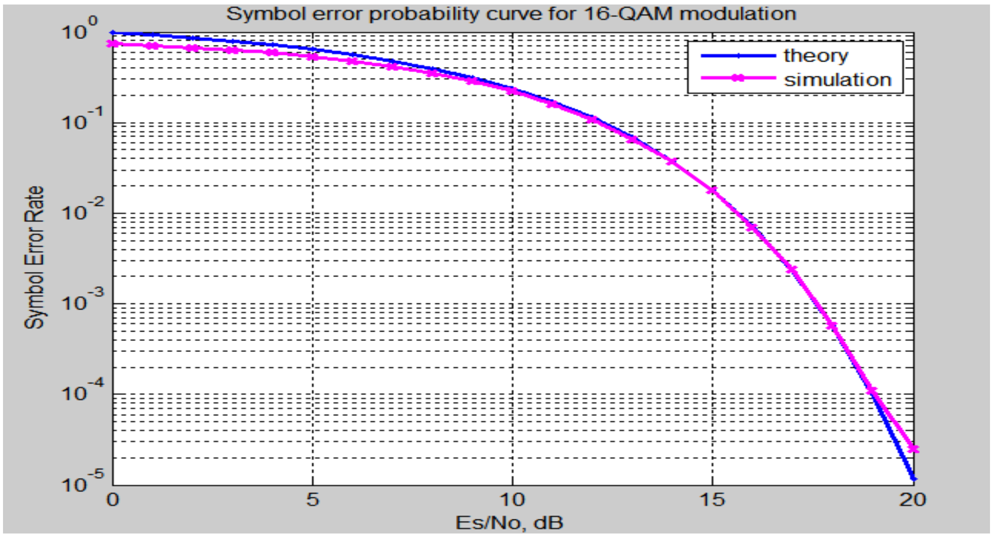


Figure 4.5 Symbol error probability vs Eb/No

RESULT: Transmitter and Receiver of 16-QAM is generated and its constellation is shown in Figure4.4. Average probability of symbol error as a function of SNR Eb/No is generated and is shown in Figure4.5

DISCUSSION:

Q1. What is QAM?

Q2.What is Euclidean distance for 16-QAM system?

Q3. Compare 16 QAM with QPSK on the basis of error rate.

Q4. How many bits are represented by each symbol in 64-QAM?

PART B

EXPERIMENT NO.5

AIM: Find all the code words of the (15, 11) Hamming code and verify that its minimum distance is equal to 3.

SOFTWARE USED: SCILAB 6.1.0

THEORY: Hamming code is a set of error-correction codes that can be used to **detect and correct the errors** that can occur when the data is moved or stored from the sender to the receiver.

Redundant bits –

Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the following formula:

$2^r \geq m + r + 1$

where, r = redundant bit, m = data bit

Suppose the number of data bits is 7, then the number of redundant bits can be calculated using:= $2^4 \geq 7 + 4 + 1$

Thus, the number of redundant bits= 4

Parity bits –

A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd. Parity bits are used for error detection. There are two types of parity bits:

1. **Even parity bit:**

   In the case of even parity, for a given set of bits, the number of 1's are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1's an even number. If the total number of 1's in a given set of bits is already even, the parity bit's value is 0.

2. **Odd Parity bit –**

   In the case of odd parity, for a given set of bits, the number of 1's are counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1's

3. an odd number. If the total number of 1's in a given set of bits is already odd, the parity bit's value is 0.

PROGRAM

(7,4) Block Code

```
k=4;//message bits length
n=7;//block length
m=n-k;//Number of parity bits
I=eye(k,k);//identity matrix
disp(I,'identity matrix Ik')
P=[1,1,0;0,1,1;1,1,1;1,0,1];//coefficient matrix
disp(P,'coefficient matrix P')
G=[P I];//generator matrix
disp(G,'generator matrix G')
H=[eye(k-1,k-1)P'];//parity check matrix
disp(H,'paritychechk matrix H')
//message bits
m=[0,0,0,0;0,0,0,1;0,0,1,0;0,0,1,1;0,1,0,0;0,1,0,1;0,1,1,0;0,1,1,1;1,0,0,0;1,0,0,1;1,0,1,0;1,
0,1,1;1,1,0,0;1,1,0,1;1,1,1,0;1,1,1,1];
//Code words
C=m*G;
C=modulo(C,2);
disp(C,'Code words of (7,4) Hamming code')
```

OUTPUT:

identity matrix Ik

```
  1.   0.   0.   0.
  0.   1.   0.   0.

  0.   0.   1.   0.

  0.   0.   0.   1.
```

coefficient matrix P

```
  1.   1.   0.

  0.   1.   1.

  1.   1.   1.

  1.   0.   1.
```

generator matrix G

```
  1.   1.   0.   1.   0.   0.   0.

  0.   1.   1.   0.   1.   0.   0.

  1.   1.   1.   0.   0.   1.   0.

  1.   0.   1.   0.   0.   0.   1.
```

paritychechk matrix H

```
  1.   0.   0.   1.   0.   1.   1.

  0.   1.   0.   1.   1.   1.   0.

  0.   0.   1.   0.   1.   1.   1.
```

Code words of (7,4) Hamming code

```
  0.   0.   0.   0.   0.   0.   0.

  1.   0.   1.   0.   0.   0.   1.

  1.   1.   1.   0.   0.   1.   0.

  0.   1.   0.   0.   0.   1.   1.

  0.   1.   1.   0.   1.   0.   0.

  1.   1.   0.   0.   1.   0.   1.
```

```
1.   0.   0.   0.   1.   1.   0.

0.   0.   1.   0.   1.   1.   1.

1.   1.   0.   1.   0.   0.   0.

0.   1.   1.   1.   0.   0.   1.

0.   0.   1.   1.   0.   1.   0.

1.   0.   0.   1.   0.   1.   1.

1.   0.   1.   1.   1.   0.   0.

0.   0.   0.   1.   1.   0.   1.

0.   1.   0.   1.   1.   1.   0.

1.   1.   1.   1.   1.   1.   1.
```

(15,11) Block Code

PROGRAM

```
k=11;//message bits length
n=15;//block length
m=n-k;//Number of parity bits
I=eye(k,k);//identity matrix
disp(I,'identity matrix Ik')


P=[1 1 0 0;0 1 1 0; 0 0 1 1;1 1 0 1; 1 0 1 0; 0 1 0 1; 1 1 1 0;0 1 1 1;1 1 1 1;1 0 1 1; 1 0 0
1];//coefficient matrix
disp(P,'coefficient matrix P')
G=[P I];//generator matrix
disp(G,'generator matrix G')
//H=[eye(k-1,k-1) P'];//parity check matrix
//disp(H,'paritychechk matrix H')
//message bits
//m=[0,0,0,0;0,0,0,1;0,0,1,0;0,0,1,1;0,1,0,0;0,1,0,1;0,1,1,0;0,1,1,1;1,0,0,0;1,0,0,1;1,0,1,0
;1,0,1,1;1,1,0,0;1,1,0,1;1,1,1,0;1,1,1,1];
```

```
//Code words
m=[0 0 0 0 0 0 0 1 0 0 1 ]
C=m*G;
C=modulo(C,2);
disp('The codeword for given message is:');
disp(C);
```

OUTPUT

identity matrix Ik

```
1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.

0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.

0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.

0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.

0.   0.   0.   0.   1.   0.   0.   0.   0.   0.   0.

0.   0.   0.   0.   0.   1.   0.   0.   0.   0.   0.

0.   0.   0.   0.   0.   0.   1.   0.   0.   0.   0.

0.   0.   0.   0.   0.   0.   0.   1.   0.   0.   0.


0.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.

0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.

0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.
```

coefficient matrix P

```
1.   1.   0.   0.

0.   1.   1.   0.

0.   0.   1.   1.

1.   1.   0.   1.
```

```
 1.   0.   1.   0.

 0.   1.   0.   1.

 1.   1.   1.   0.

 0.   1.   1.   1.

 1.   1.   1.   1.

 1.   0.   1.   1.

 1.   0.   0.   1.
```

generator matrix G

```
 1.   1.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.

 0.   1.   1.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.

 0.   0.   1.   1.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.

1.   1.   0.   1.   0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.

 1.   0.   1.   0.   0.   0.   0.   0.   1.   0.   0.   0.   0.   0.   0.

 0.   1.   0.   1.   0.   0.   0.   0.   0.   1.   0.   0.   0.   0.   0.


 1.   1.   1.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.   0.   0.

 0.   1.   1.   1.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.   0.

 1.   1.   1.   1.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.

 1.   0.   1.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.

 1.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.
```

The codeword for given message is:

column 1 to 14

1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.

column 15

RESULT: The code words of the (15, 11) Hamming code is generated

DISCUSSION

Q1. What is hamming code?

Q2. how many no. of parity bits are required in hamming code if the msg size is 8 bit

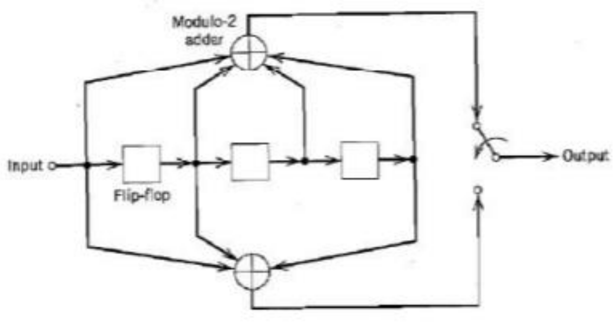
Q3. What is the rate of Hamming codes?

Q4. What is the minimum hamming distance required to detect 3 bit errors hamming code?

Q5. What are application of hamming code?

EXPERIMENT NO.6

AIM: Generate an equiprobable random binary information sequence of length 15.Determine the output of the convolutional encoder shown below for this sequence.



SOFTWARE USED: MATLAB 7.1/SCILAB 6.1.0

THEORY:

In convolutional codes, the message comprises of data streams of arbitrary length and a sequence of output bits are generated by the sliding application of Boolean functions to the data stream.

In block codes, the data comprises of a block of data of a definite length. However, in convolutional codes, the input data bits are not divided into block but are instead fed as streams of data bits, which convolve to output bits based upon the logic function of the encoder. Also, unlike block codes, where the output codeword is dependent only on the present inputs, in convolutional codes, output stream depends not only the present input bits but also only previous input bits stored in memory.

For generating a convolutional code, the information is passed sequentially through a linear finite-state shift register. The shift register comprises of (-bit) stages and Boolean function generators.

A convolutional code can be represented as (n,k, K) where

- *k* is the number of bits shifted into the encoder at one time. Generally, *k = 1.*
- n is the number of encoder output bits corresponding to k information bits.
- The code-rate, $R_c = k/n$.

- The encoder memory, a shift register of size k, is the constraint length.
- *n* is a function of the present input bits and the contents of *K*.
- The state of the encoder is given by the value of (*K - 1)* bits.

PART 1: Equiprobable random binary information sequence of length 15

PROGRAM

```
n = 15; %

numberOfOnes = n/2

% Get a list of random locations, with no number repeating.

indexes = randperm(n)

% Start off with all zeros.

x = zeros(1, n);

% Now make half of them, in random locations, a 1.

x(indexes(1:numberOfOnes)) = 1

stem(x);
```

OUTPUT

```
x =

     0     0     1     0     1     1     1     1     0     0     1     0     0     1     0
```

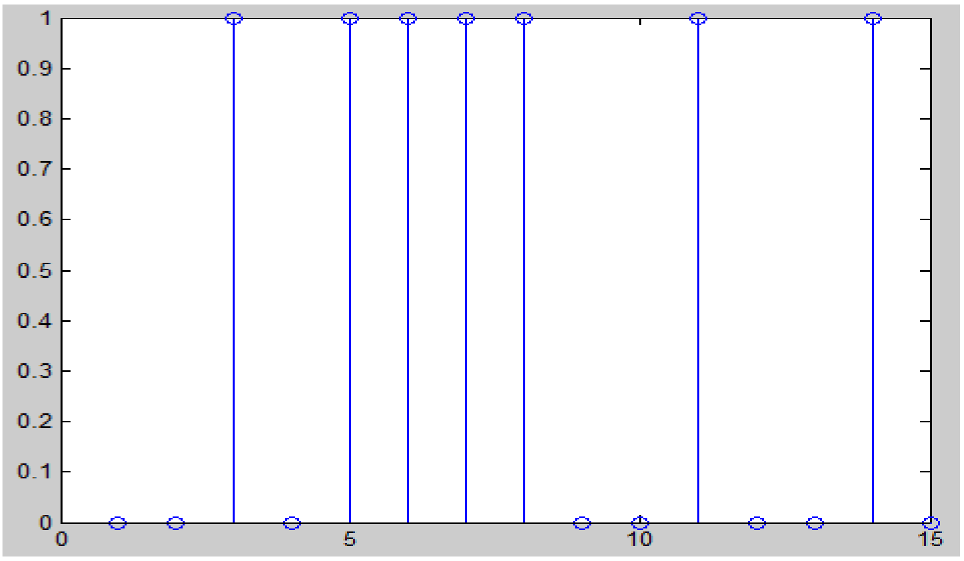


Figure 6.1 Random binary information sequence

PART 2: output of the convolutional encoder

PROGRAM

```
clc;
D=poly(0,'D');
g1D=1+D+D^2+D^3;//generator polynomial 1
disp(g1D);
g2D=1+D+D^3;//generator polynomial 2
mD=1+0+0+D^3+D^4;//message sequence polynomial representation
x1D=g1D*mD;//top output polynomial
x2D=g2D*mD;//bottom output polynomial
x1=coeff(x1D);
x2=coeff(x2D);
disp(modulo(x1,2),'top output sequence')
disp(modulo(x2,2),'bottom output sequence')
```

OUTPUT:

top output sequence

```
1.   1.   1.   0.   0.   0.   0.   1.
```

bottom output sequence

```
   1.   1.   0.   0.   0.   1.   1.   1.
```

RESULT: Generate an Equiprobable random binary information sequence of length 15 is generated as shown Figure 6.1 and output of the convolutional encoder is calculated

DISCUSSION:

Q1. What is code rate in convolutional code?

Q2. What is the convolutional operation in a convolutional code?

Q3. Where are convolutional codes used?

Q4. What is the application of convolution code?

Q5. What is constraint length?

EXPERIMENT NO.7

AIM: Generate the L=31 Gold sequences. Consider a time-synchronous CDMA system (direct sequence spread spectrum) having four users, each employing a distinct Gold sequence of length L=31 and the binary (±1) modulation of their representative Gold sequences. The receiver for each user correlates the composite CDMA received signal, which is corrupted by AWGN (added on a chip-by-chip basis) with each user's respective sequence. Using 10000 information bits, estimate and plot the probability of error for each user as a function of SNR

SOFTWARE USED: MATLAB 7.1

THEORY:

Gold sequences have been proposed by Gold in 1967 and 1968. These are constructed by EXOR-ing two m-sequences of the same length with each other. Thus, for a Gold sequence of length $m = 2^l-1$, one uses two LFSR, each of length 2^l-1.

If the LSFRs are chosen appropriately, Gold sequences have better cross-correlation properties than maximum length LSFR sequences.

Prefered sequences

Gold (and Kasami) showed that for certain well-chosen *m*-sequences, the cross correlation only takes on three possible values, namely -1, -*t* or *t*-2. Two such sequences are called preffered sequences. Here *t* depends solely on the length of the LFSR used. In fact, for a LFSR with *l* memory elements,

if *l* is odd, $t = 2^{(l+1)/2} + 1$, and

if *l* is even, $t = 2^{(l+2)/2} + 1$.

Thus, a Gold sequence formally is an arbitrary phase of a sequence in the set *G(u,v)* defined by

$G(u,v) = \{u, v, u * v, u * Tv, u * T^2 v, U * T^{(N-1)} v\}$

T^k denotes the operator which shifts vectors cyclically to the left by *k* places, * is the exclusive OR operator and *u*, *v* are *m*-sequences of period generated by different primitive binary polynomials.

It is well known that the "partial crosscorrelation" values can be altered by changing the phases of the code sequences. In theory, then, it is possible to find optimal phases which minimize the interference in the desired data signal. However, for *K* users each employing a sequence of period *N*, there are a total of *N K* different sets of sequence phases possible. For a realistic system, e.g. , direct computation becomes intractable. Even when direct computation is performed, the reduction in interference of the optimal set of phases over the worst set of phases is 30%.

PROGRAM:

```
clear
clc
G=93;  % Code length
x=[];
%..............Generation of first perferred PN sequence...........
sd1 =[0 0 0 0 1];      % Initial State of Register.
PS1=[];
for j=1:G
    PS1=[PS1 sd1(5)];
    if sd1(1)==sd1(4)
        temp1=0;
    else temp1=1;
    end
    sd1(1)=sd1(2);
    sd1(2)=sd1(3);
    sd1(3)=sd1(4);
    sd1(4)=sd1(5);
    sd1(5)=temp1;
end
x=[x PS1];



%................Generation of Second Preferred sequnces.............
```

```
PS2=[];
PS2(1)=x(1);
for i=1:30
    j=(3*i)+1;
    PS2(i+1)=x(j);
end
PS2=[PS2];

%.................Shifting and Storing of PS1 in Matrix 'y'............

for k=1:31
    for j=1:31
        y(k,j)=x(j+k-1);
    end
end

%..................Generation of Gold Sequences........................

for i=1:31
Gold_Seq(1,:)=[PS1(1,(1:31))];
Gold_Seq(2,:)=[PS2];
Gold_Seq(i+2,:)=xor(PS2,y(i,(1:31)));
end
for j=1:33
subplot(11,3,j)
stem(Gold_Seq(j,:))
axis([1 32 0 1.5])
end
```

OUTPUT

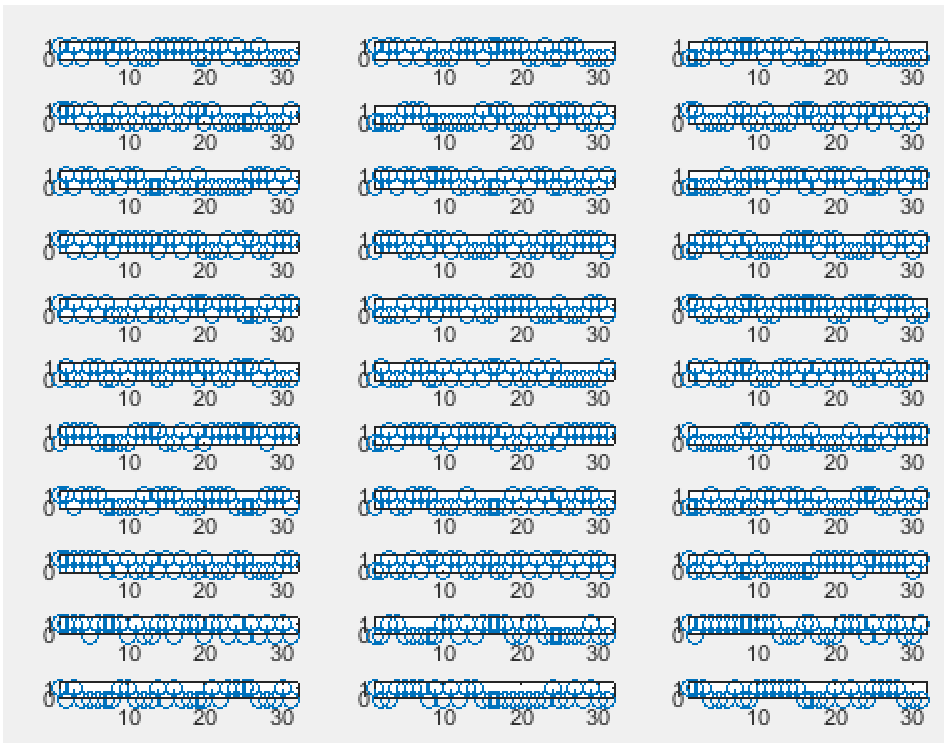


Figure 7.1

RESULT: Gold Code sequence of length 31 is generated as shown in Figure 7.1

DISCUSSION

Q1. How many PN sequences are required for gold sequence?

Q2. How many gold codes are there?

Q3. How do you make a gold code?

Q4. What is DSSS technique?

Q5. What is chip code?

EXPERIMENT NO.8

AIM: Consider a MIMO (multiple-input, multiple-output) system with NT = 2 transmit antennas and NR = 2 receive antennas. Generate the elements of the channel matrix H for a Rayleigh fading (frequency nonselective) AWGN channel and the corresponding inputs to the detectors for the two receive antennas.

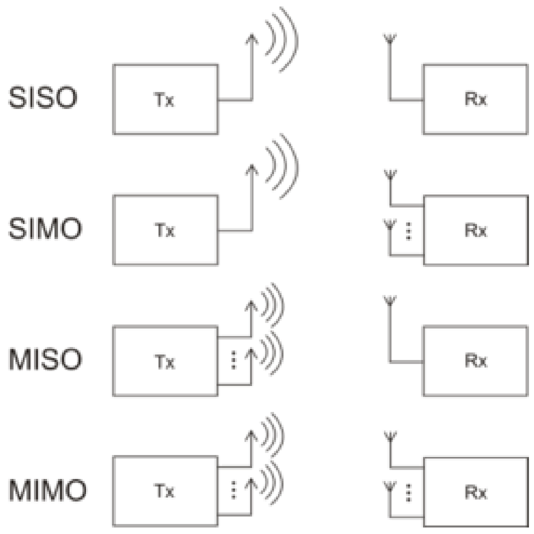
SOFTWARE USED: MATLAB 7.1

THEORY:

MIMO is a new wireless technology conceived in the mid 90’s. It is based on an entirely new paradigm for digital signal processing that multiplies the data rate throughput achievable in wireless communication products. It greatly improves the reliability, range and robustness of the connection providing a much better user experience that is closer to “wired” Ethernet quality.

MIMO is effectively a radio antenna technology as it uses multiple antennas at the transmitter and receiver to enable a variety of signal paths to carry the data, choosing separate paths for each antenna to enable multiple signal paths to be used.

It is found between a transmitter and a receiver; the signal can take many paths. Additionally by moving the antennas even a small distance the paths used will change. The variety of paths available occurs as a result of the number of objects that appear to the side or even in the direct path between the transmitter and receiver. Previously these multiple paths only served to introduce interference. By using MIMO, these additional paths can be used to increase the capacity of a link.

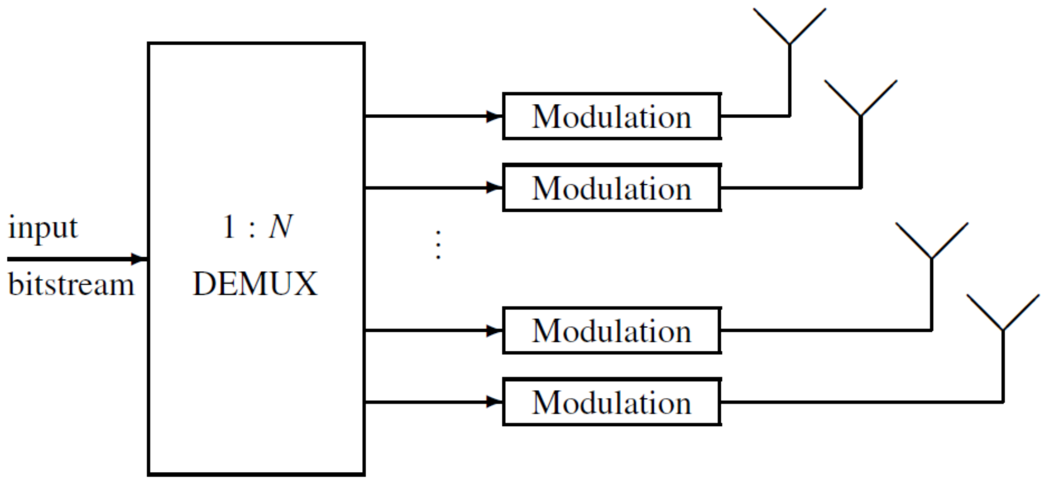


Spatial multiplexing(SM)

It requires MIMO antenna configuration. In SM, a high rate signal is split into multiple lower rate streams and each stream is transmitted from a different transmit antenna in the same frequency channel. If these signals arrive at the receiver antenna array with sufficiently different spatial signatures, the receiver can separate these streams into parallel channels.

SM is a very powerful technique for increasing channel capacity at higher signal-to-noise ratios (SNR). The maximum number of spatial streams is limited by the lesser in the number of antennas at the transmitter or receiver. SM can be used with or without transmit channel knowledge. SM can also be used for simultaneous transmission to multiple receivers, known as space-division multiple accesses. By scheduling receivers with different spatial signatures, good separatibility can be assured.

SM can offer an increase in the transmission rate (or throughput), while using the same bandwidth and power as in a traditional SISO system. In SM, the input is demultiplexed into N separate streams, using a serial-to-parallel converter, and each stream is transmitted from an independent antenna. As a result, the throughput is N symbols per channel use for a MIMO channel with N transmit antennas. This N-fold increase in throughput will generally come at the cost of a lower diversity gain compared to space-time coding. Therefore, spatial multiplexing is a better choice for high rate systems operating at relatively high SNRs while space-time coding is more appropriate for transmitting at relatively low rates and low SNRs.



PROGRAM

```
clear

N = 10 % number of bits or symbols

Eb_N0_dB = 10; % SNR IN dB

nTx = 2

nRx = 2

   % Transmitter

   ip =  rand(1,N)>0.5 % generating 0,1 with equal probability

   ip1=reshape(ip,nRx,N/nTx)

  ipTxA=ip1(1,:)

  ipTxB=ip1(2,:)
```

```
    s = 2*ip-1 % BPSK modulation 0 -> -1; 1 -> 0

    sMod = reshape(s,nRx,N/nTx) % grouping in [nRx,nTx,N/NTx ] matrix

    h = 1/sqrt(2)*[randn(nRx,N/nTx) + j*randn(nRx,N/nTx)]% Rayleigh channel matrix

    n = 1/sqrt(2)*[randn(nRx,N/nTx) + j*randn(nRx,N/nTx)] % white gaussian noise, 0dB
variance

    % Channel and Noise addition

    y = h.*sMod + 10^(- Eb_N0_dB /20)*n

    y1=y(1,:)

    y2=y(2,:)

    Tx1 = kron(y1,ones(nRx,1)) % inputs to detector from antenna 1

    Tx2= kron(y2,ones(nRx,1))  %inputs to detector from antenna 2
```

Output:

```
N = 10

nTx =2

nRx =2

ip =

     0     0     0     1     1     1     1     0     1     1

ip1 =

     0     0     1     1     1

     0     1     1     0     1

ipTxA =

     0     0     1     1     1
```

```
ipTxB =

     0     1     1     0     1

s =

    -1    -1    -1     1     1     1     1    -1     1     1

sMod =

    -1    -1     1     1     1

    -1     1     1    -1     1

h =

   0.5708 + 0.4507i   0.9153 - 0.2852i   0.1546 - 0.3078i  -1.4699 + 0.6209i  -0.7683 - 0.1827i

  -0.7265 - 0.2861i   0.0106 + 0.0595i   1.2114 - 0.3978i   0.0798 - 0.5760i  -1.1019 + 0.3488i

n =

  -0.5676 + 0.0464i   0.4438 - 0.0544i   1.8249 + 1.2039i   0.7237 + 0.0669i  -0.5896 + 0.6501i

  -0.0059 - 0.0087i   0.1092 - 1.1021i  -0.9237 - 0.3316i   0.5500 + 0.2030i  -0.4148 + 0.3607i

y =

  -0.7503 - 0.4360i  -0.7750 + 0.2680i   0.7317 + 0.0729i  -1.2411 + 0.6420i  -0.9547 + 0.0229i

   0.7247 + 0.2833i   0.0451 - 0.2891i   0.9193 - 0.5027i   0.0941 + 0.6402i  -1.2331 + 0.4629i

y1 =
```

-0.7503 - 0.4360i -0.7750 + 0.2680i 0.7317 + 0.0729i -1.2411 + 0.6420i -0.9547 + 0.0229i

y2 = 0.7247 + 0.2833i 0.0451 - 0.2891i 0.9193 - 0.5027i 0.0941 + 0.6402i -1.2331 + 0.4629i

Tx1 = -0.7503 - 0.4360i -0.7750 + 0.2680i 0.7317 + 0.0729i -1.2411 + 0.6420i -0.9547 + 0.0229i

-0.7503 - 0.4360i -0.7750 + 0.2680i 0.7317 + 0.0729i -1.2411 + 0.6420i -0.9547 + 0.0229i

Tx2 = 0.7247 + 0.2833i 0.0451 - 0.2891i 0.9193 - 0.5027i 0.0941 + 0.6402i -1.2331 + 0.4629i

0.7247 + 0.2833i 0.0451 - 0.2891i 0.9193 - 0.5027i 0.0941 + 0.6402i -1.2331 + 0.4629i

RESULT: For a MIMO (multiple-input, multiple-output) system with NT = 2 transmit antennas and NR = 2 receive antennas, the elements of the channel matrix H for a Rayleigh fading (frequency nonselective) AWGN channel and the corresponding inputs to the detectors for the two receive antennas has been calculated

DISCUSSION:

Q1. What is MIMO technology?

Q2. What is the main advantage of MIMO over SISO?

Q3. How does MIMO increase capacity?

Q4. What are the benefits of MIMO?

Q5. Why MIMO is needed?

EXPERIMENT NO.9

AIM: Perform feature extraction from a given Image and use Principal Components as image descriptors.

SOFTWARE USED: MATLAB 7.1

THEORY:

Feature Extraction/Engineering: It is a process of creating/deriving new features from the features which give more information and are less redundant. This is mainly used in pattern recognition and image processing where the dimension of data is high. Feature Extraction is achieved through Principal component analysis

Principal Component Analysis (**PCA**) is a well-known and one of the most successful techniques used in image recognition and compression for extracting feature and representing data. It is technique widely used in the area of pattern recognition, computer vision and signal processing. The purpose of PCA is to reduce the large dimensionality of the data space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which are needed to describe the data economically. This is the case when there is a strong correlation between observed variables. By discarding minor components, the PCA effectively reduces the number of features and displays the data set in a low dimensional subspace.

The other main advantage of PCA is dimension will be reduced by avoiding redundant information, without much loss. Better understanding of principal component analysis is through statistics and some of the mathematical techniques which are Eigen values, Eigen vectors. PCA is a useful statistical and common technique that has found application in fields such as image recognition and compression

PROGRAM:

```
clc;
close all;
clear all;
I = imread(' C:\Users\Desktop\peppers.tiff');
figure, imshow(I);
I = double(imread(' C:\Users\Desktop\peppers.tiff'));
X = reshape(I,size(I,1)*size(I,2),3);
coeff = pca(X);
Itransformed = X*coeff;
Ipc1 = reshape(Itransformed(:,1),size(I,1),size(I,2));  %RED CHANNEL
Ipc2 = reshape(Itransformed(:,2),size(I,1),size(I,2));% GREEN CHANNEL
Ipc3 = reshape(Itransformed(:,3),size(I,1),size(I,2));%BLUE CHANNEL
figure, imshow(Ipc1,[]);
figure, imshow(Ipc2,[]);
figure, imshow(Ipc3,[]);
```

OUTPUT

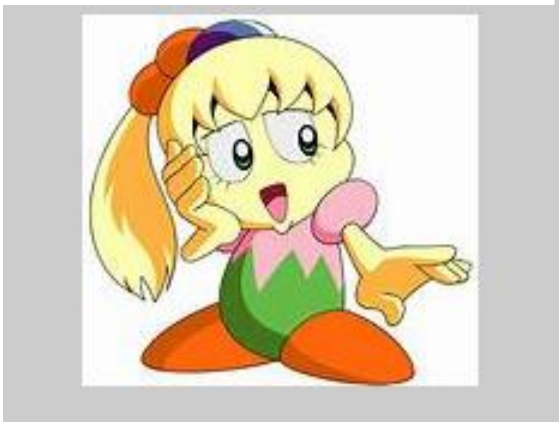


Figure 9.1

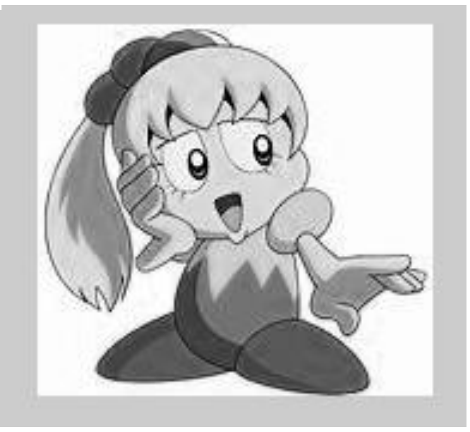
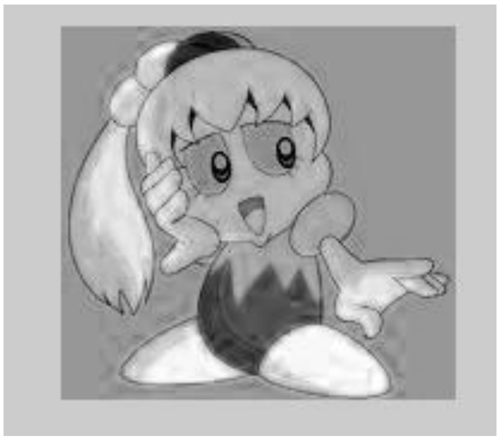


Figure 9.2



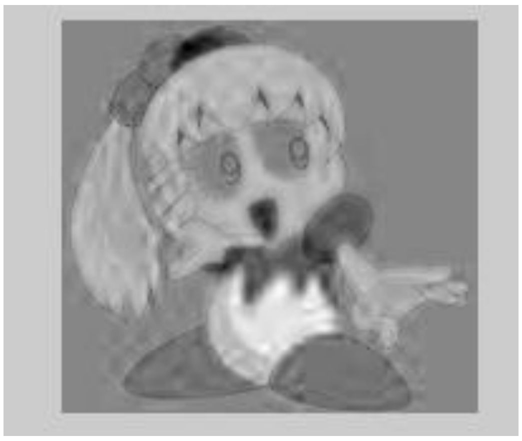


Figure 9.3 **Figure 9.4**

RESULT: Feature extraction from a given Image by use Principal Components as image descriptors is generated as shown in Figure 9.1, 9.2, 9.3, 9.4

DISCUSSION:

Q1**.** What is the feature extraction process in image processing?

Q2. How features are extracted from an image?

Q3. Why do we need feature extraction?

Q4. What are the feature extraction methods?

Q5. What is the use of feature extraction?

EXPERIMENT NO.10

AIM: By using an image dataset, train a Neural Network to recognize a given Image. Apply this in context to face/object recognition and calculate recognition accuracy of the training set.

SOFTWARE USED: MATLAB 7.1

THEORY:

Image classification is an amazing application of deep learning. We can train a powerful algorithm to model a large image dataset. This model can then be used to classify a similar but unknown set of images.

A convolutional neural network (CNN or ConvNet), is a network architecture for deep learning which learns directly from data, eliminating the need for manual feature extraction.

CNNs are particularly useful for finding patterns in images to recognize objects, faces, and scenes. They can also be quite effective for classifying non-image data such as audio, time series, and signal data.

A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.

Feature Learning, Layers, and Classification

Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.

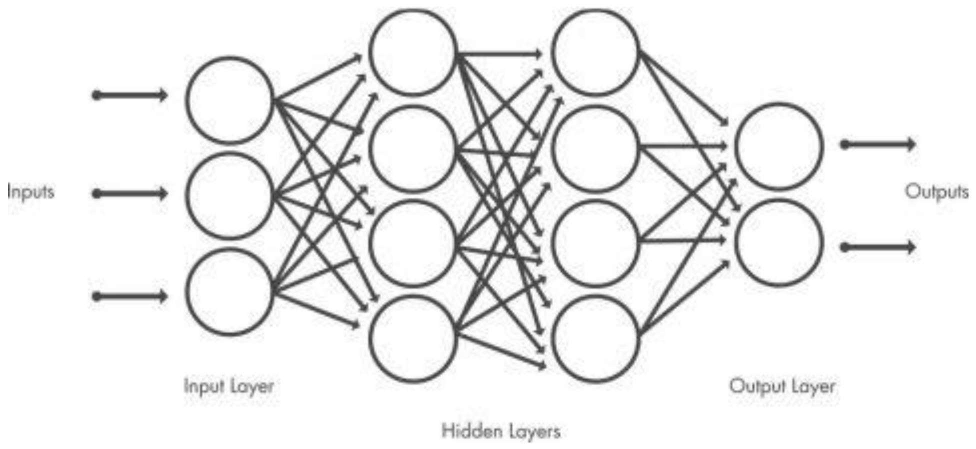


Figure 10.1

These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are: convolution, activation or ReLU, and pooling.

- **Convolution** puts the input images through a set of convolutional filters, each of which activates certain features from the images.
- **Rectified linear unit (ReLU)** allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as *activation*, because only the activated features are carried forward into the next layer.
- **Pooling** simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

These operations are repeated over tens or hundreds of layers, with each layer learning to identify different features.

Classification Layers

After learning features in many layers, the architecture of a CNN shifts to classification.

The next-to-last layer is a fully connected layer that outputs a vector of K dimensions where K is the number of classes that the network will be able to predict. This vector contains the probabilities for each class of any image being classified.

The final layer of the CNN architecture uses a classification layer such as softmax to provide the classification output.

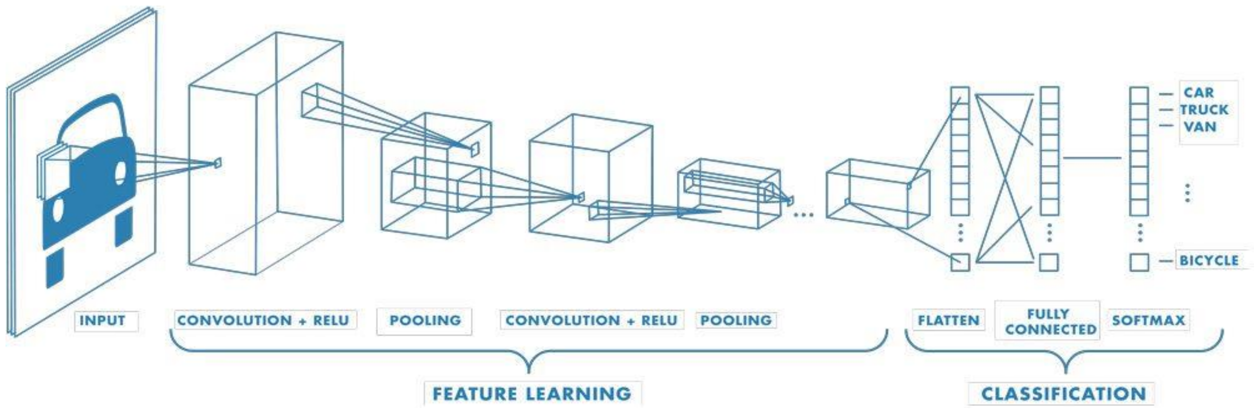


Figure 10.2

Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer.convolutional neural networks are essential tools for deep learning and are especially suited for image recognition.

The example demonstrates how to:

- Load image data.
- Define the network architecture.
- Specify training options.
- Train the network.
- Predict the labels of new data and calculate the classification accuracy.

Load Data

Load the digit sample data as an image datastore. The imageDatastore function automatically labels the images based on folder names.

Divide the data into training and validation data sets, so that each category in the training set contains 750 images, and the validation set contains the remaining images from each label. splitEachLabel splits the image datastore into two new datastores for training and validation.

Define Network Architecture

Define the convolutional neural network architecture. Specify the size of the images in the input layer of the network and the number of classes in the fully connected layer before the classification layer. Each image is 28-by-28-by-1 pixels and there are 10 classes.

Train Network

Specify the training options and train the network.

By default, trainNetwork uses a GPU if one is available (requires Parallel Computing Toolbox™ and a CUDA® enabled GPU with compute capability 3.0 or higher). Otherwise, it uses a CPU. You can also specify the execution environment by using the 'ExecutionEnvironment' name-value pair argument of trainingOptions.

Test Network

Classify the validation data and calculate the classification accuracy.

PROGRAM:

```
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet','nndemos', ...
    'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
 'IncludeSubfolders',true, ...
 'LabelSource','foldernames');
numTrainFiles = 750;
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
inputSize = [28 28 1];
numClasses = 10;
layers = [
 imageInputLayer(inputSize)
 convolution2dLayer(5,20)
 batchNormalizationLayer
reluLayer
fullyConnectedLayer(numClasses)
softmaxLayer
classificationLayer];
options = trainingOptions('sgdm', ...
'MaxEpochs',4, ...
'ValidationData',imdsValidation, ...
'ValidationFrequency',30, ...
'Verbose',false, ...
'Plots','training-progress');
net = trainNetwork(imdsTrain,layers,options);
YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
```

OUTPUT:

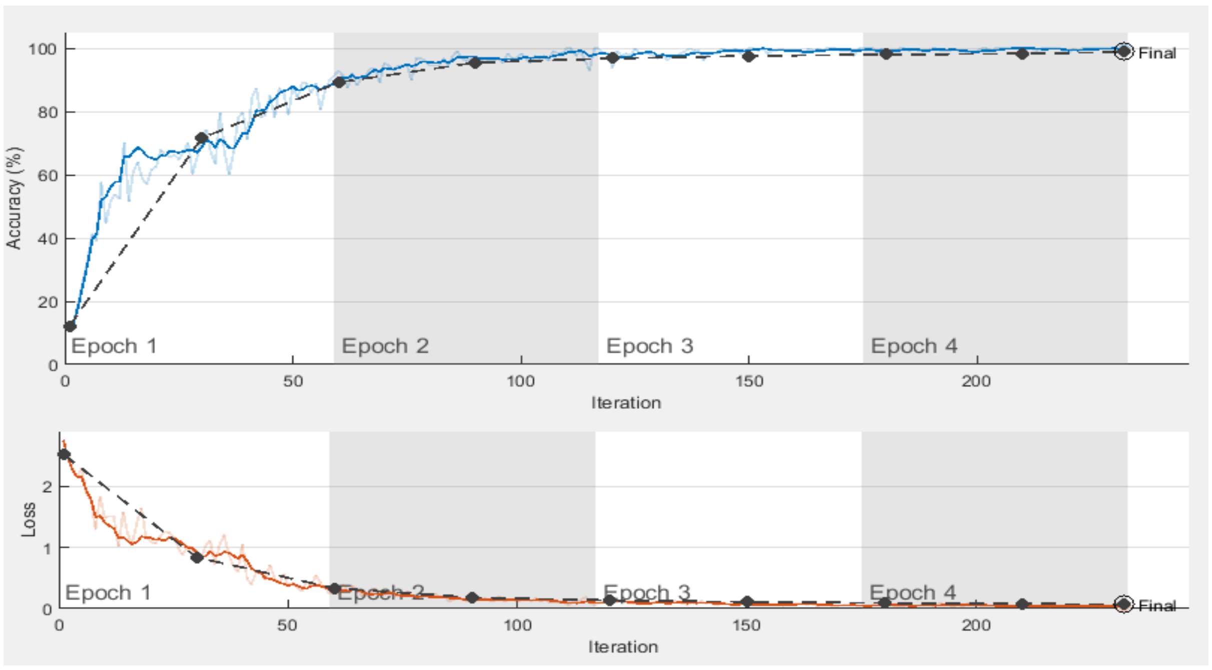
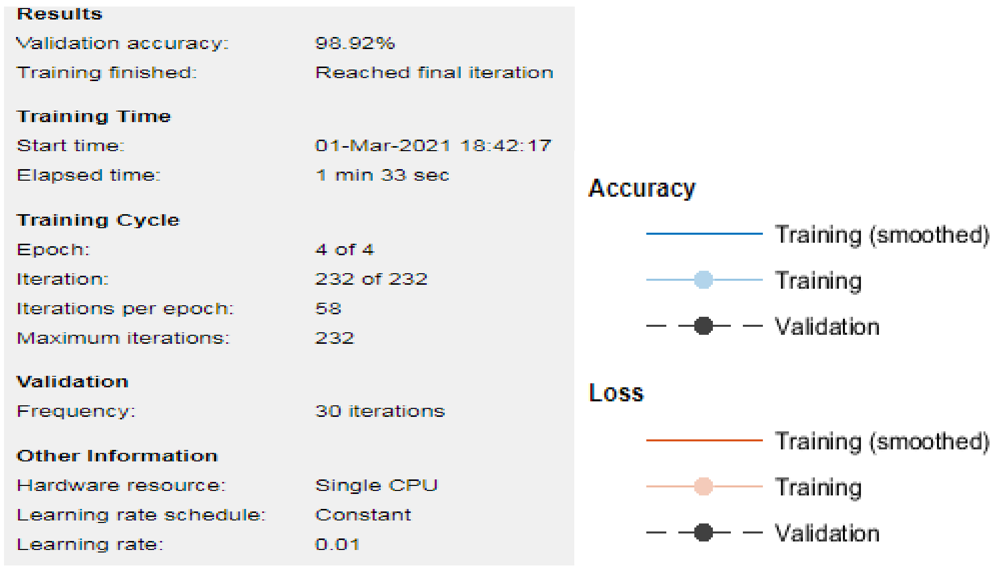


Figure 10.3



Results

Validation accuracy:	98.92%
Training finished:	Reached final iteration

Training Time

Start time:	01-Mar-2021 18:42:17
Elapsed time:	1 min 33 sec

Training Cycle

Epoch:	4 of 4
Iteration:	232 of 232
Iterations per epoch:	58
Maximum iterations:	232

Validation

Frequency:	30 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

Accuracy

Training (smoothed)

Training

Validation

Loss

Training (smoothed)

Training

Validation

Figure 10.4

RESULT: By using an image dataset, face/object recognition using a neural network and recognition accuracy of the training set is calculated as shown in Figure10.3 and Figure 10.4.

DISCUSSION:

Q1. How neural network is used in face recognition?

Q2. What is the purpose of face recognition?

Q3. What are the disadvantages of facial recognition?

Q4. What is difference between face detection and face recognition?

Q5. Where is facial recognition used?

EXPERIMENT NO. 11

AIM: Develop a Fuzzy Inference System (FIS) by using a set of fuzzy rule base between some key image parameters and calculate output after defuzzification.

SOFTWARE USED: MATLAB 7.1

THEORY:

Fuzzy Inference System is the key unit of a fuzzy logic system having decision making as its primary work. It uses the "IF…THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

Characteristics of Fuzzy Inference System

Following are some characteristics of FIS −

- The output from FIS is always a fuzzy set irrespective of its input which can be fuzzy or crisp.
- It is necessary to have fuzzy output when it is used as a controller.
- A defuzzification unit would be there with FIS to convert fuzzy variables into crisp variables.

Functional Blocks of FIS

The following five functional blocks will help you understand the construction of FIS −

- **Rule Base** − It contains fuzzy IF-THEN rules.
- **Database** − It defines the membership functions of fuzzy sets used in fuzzy rules.
- **Decision-making Unit** − It performs operation on rules.
- **Fuzzification Interface Unit** − It converts the crisp quantities into fuzzy quantities.
- **Defuzzification Interface Unit** − It converts the fuzzy quantities into crisp quantities. Following is a block diagram of fuzzy interference system.

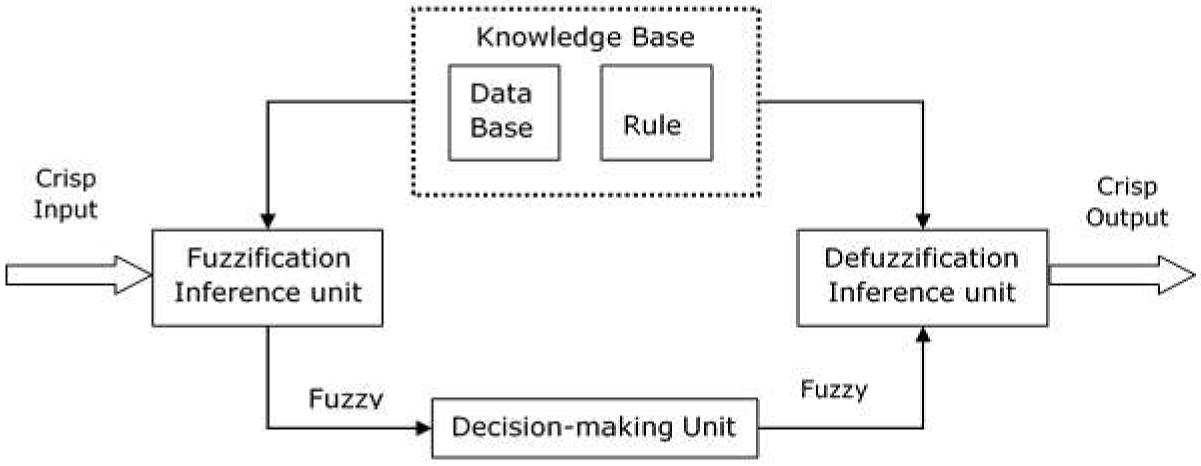


Figure 11.1 Bock diagram of fuzzy interference system.

Working of FIS

The working of the FIS consists of the following steps −

- A fuzzification unit supports the application of numerous fuzzification methods, and converts the crisp input into fuzzy input.
- A knowledge base - collection of rule base and database is formed upon the conversion of crisp input into fuzzy input.
- The defuzzification unit fuzzy input is finally converted into crisp output.

Methods of FIS

Let us now discuss the different methods of FIS. Following are the two important methods of FIS, having different consequent of fuzzy rules −

- Mamdani Fuzzy Inference System
- Takagi-Sugeno Fuzzy Model (TS Method)

Mamdani Fuzzy Inference System

This system was proposed in 1975 by Ebhasim Mamdani. Basically, it was anticipated to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system.

Steps for Computing the Output

Following steps need to be followed to compute the output from this FIS −

- **Step 1** − Set of fuzzy rules need to be determined in this step.
- **Step 2** − In this step, by using input membership function, the input would be made fuzzy.
- **Step 3** − Now establish the rule strength by combining the fuzzified inputs according to fuzzy rules.
- **Step 4** − In this step, determine the consequent of rule by combining the rule strength and the output membership function.
- **Step 5** − For getting output distribution combine all the consequents.
- **Step 6** − Finally, a defuzzified output distribution is obtained.

Following is a block diagram of Mamdani Fuzzy Interface System.

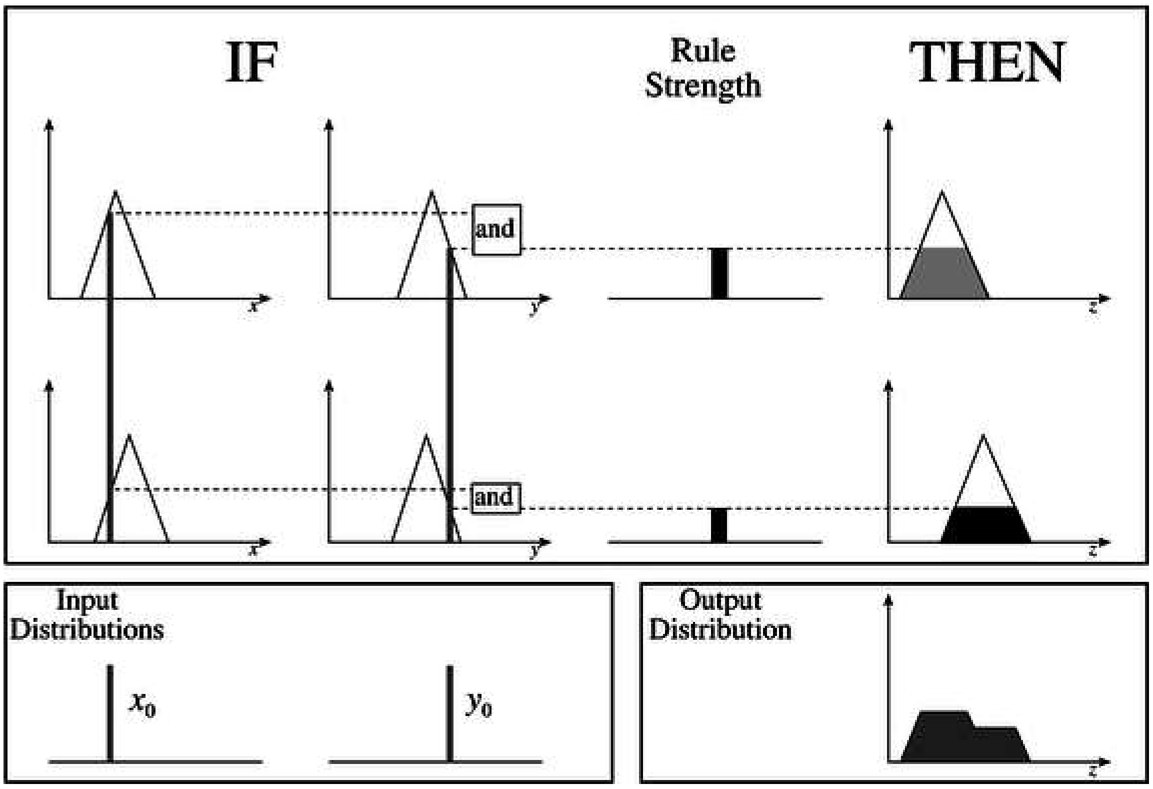


Figure 11.2 Block diagram of Mamdani Fuzzy Interface System.

Takagi-Sugeno Fuzzy Model (TS Method)

This model was proposed by Takagi, Sugeno and Kang in 1985. Format of this rule is given as −

IF x is A and y is B THEN Z = f(x,y)

Here, *AB* are fuzzy sets in antecedents and *z* = *f(x,y)* is a crisp function in the consequent.

Fuzzy Inference Process

The fuzzy inference process under Takagi-Sugeno Fuzzy Model (TS Method) works in the following way −

- **Step 1: Fuzzifying the inputs** − Here, the inputs of the system are made fuzzy.

- **Step 2: Applying the fuzzy operator** − In this step, the fuzzy operators must be applied to get the output.

Rule Format of the Sugeno Form

The rule format of Sugeno form is given by −

if 7 = x and 9 = y then output is z = ax+by+c

Comparison between the two methods

Let us now understand the comparison between the Mamdani System and the Sugeno Model.

- **Output Membership Function** − The main difference between them is on the basis of output membership function. The Sugeno output membership functions are either linear or constant.
- **Aggregation and Defuzzification Procedure** − The difference between them also lies in the consequence of fuzzy rules and due to the same their aggregation and defuzzification procedure also differs.
- **Mathematical Rules** − More mathematical rules exist for the Sugeno rule than the Mamdani rule.
- **Adjustable Parameters** − The Sugeno controller has more adjustable parameters than the Mamdani controller.

PROGRAM:

```
Irgb = imread('C:\Users\admin\Documents\peppers.jpg');
Igray = rgb2gray(Irgb);

figure
image(Igray,'CDataMapping','scaled')
colormap('gray')
title('Input Image in Grayscale')
I = im2double(Igray);
```

```
Gx = [-1 1];
Gy = Gx';
Ix = conv2(I,Gx,'same');
Iy = conv2(I,Gy,'same');
figure
image(Ix,'CDataMapping','scaled')
colormap('gray')
title('Ix')
figure
image(Iy,'CDataMapping','scaled')
colormap('gray')
title('Iy')
edgeFIS = mamfis('Name','edgeDetection');
edgeFIS = addInput(edgeFIS,[-1 1],'Name','Ix');
edgeFIS = addInput(edgeFIS,[-1 1],'Name','Iy');
sx = 0.1;
sy = 0.1;
edgeFIS = addMF(edgeFIS,'Ix','gaussmf',[sx 0],'Name','zero');
edgeFIS = addMF(edgeFIS,'Iy','gaussmf',[sy 0],'Name','zero');


edgeFIS = addOutput(edgeFIS,[0 1],'Name','Iout');
wa = 0.1;
wb = 1;
wc = 1;
ba = 0;
bb = 0;
bc = 0.7;
edgeFIS = addMF(edgeFIS,'Iout','trimf',[wa wb wc],'Name','white');
edgeFIS = addMF(edgeFIS,'Iout','trimf',[ba bb bc],'Name','black');
figure
subplot(2,2,1)
```

```
plotmf(edgeFIS,'input',1)
title('Ix')
subplot(2,2,2)
plotmf(edgeFIS,'input',2)
title('Iy')
subplot(2,2,[3 4])
plotmf(edgeFIS,'output',1)
title('Iout')
r1 = "If Ix is zero and Iy is zero then Iout is white";
r2 = "If Ix is not zero or Iy is not zero then Iout is black";
edgeFIS = addRule(edgeFIS,[r1 r2]);
edgeFIS.Rules
Ieval = zeros(size(I));
for ii = 1:size(I,1)
    Ieval(ii,:) = evalfis(edgeFIS,[(Ix(ii,:));(Iy(ii,:))]');
end
figure
image(I,'CDataMapping','scaled')
colormap('gray')
title('Original Grayscale Image')
figure
image(Ieval,'CDataMapping','scaled')
colormap('gray')
title('Edge Detection Using Fuzzy Logic')
```

OUTPUT:

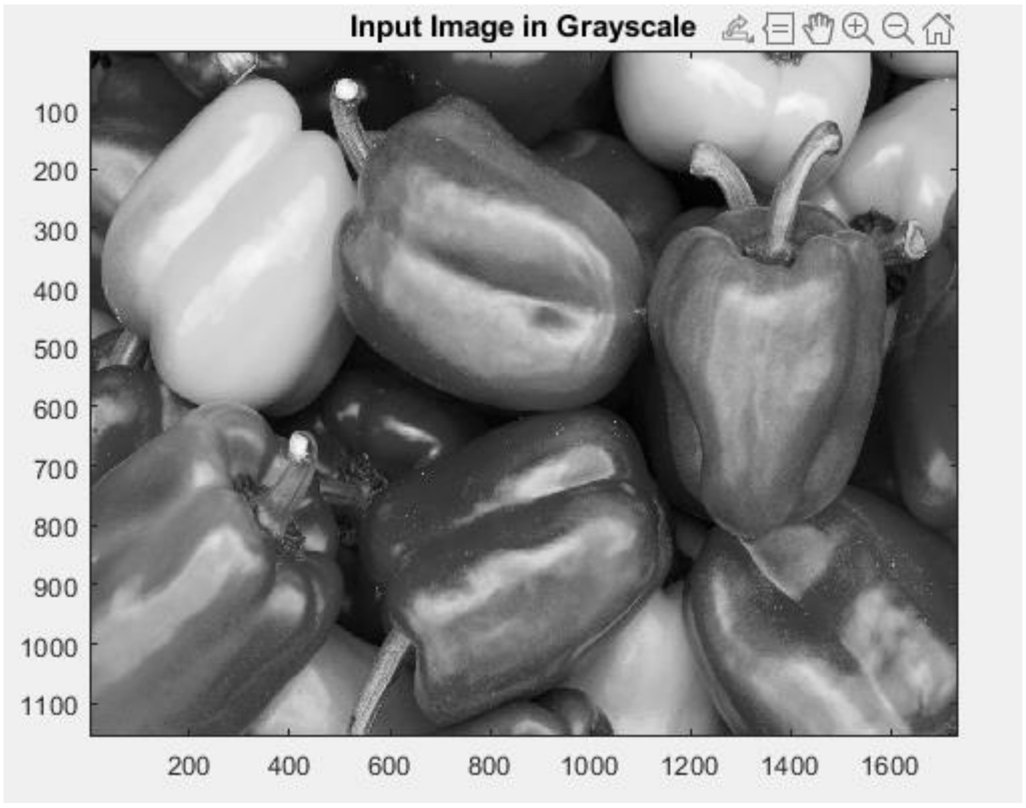


Figure 11.3 Input image in grayscale

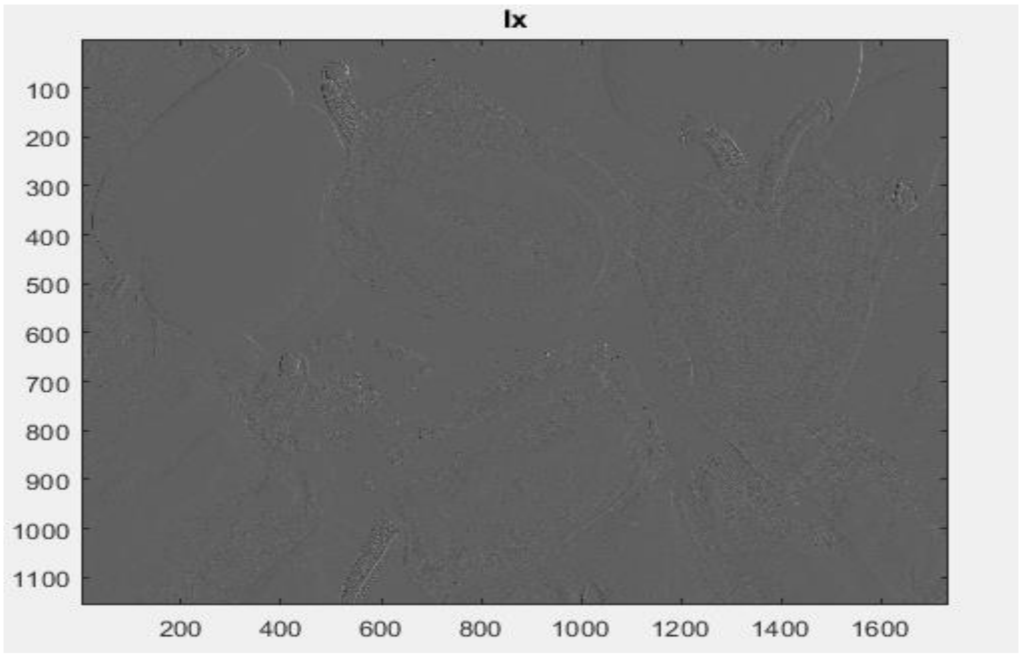


Figure 11.4

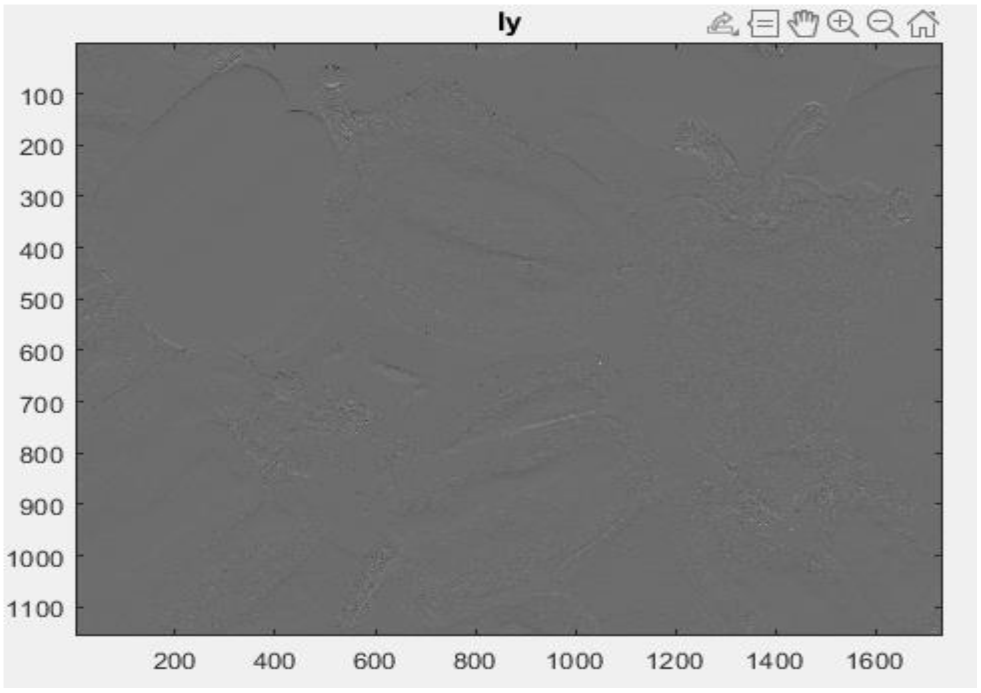


Figure 11.5

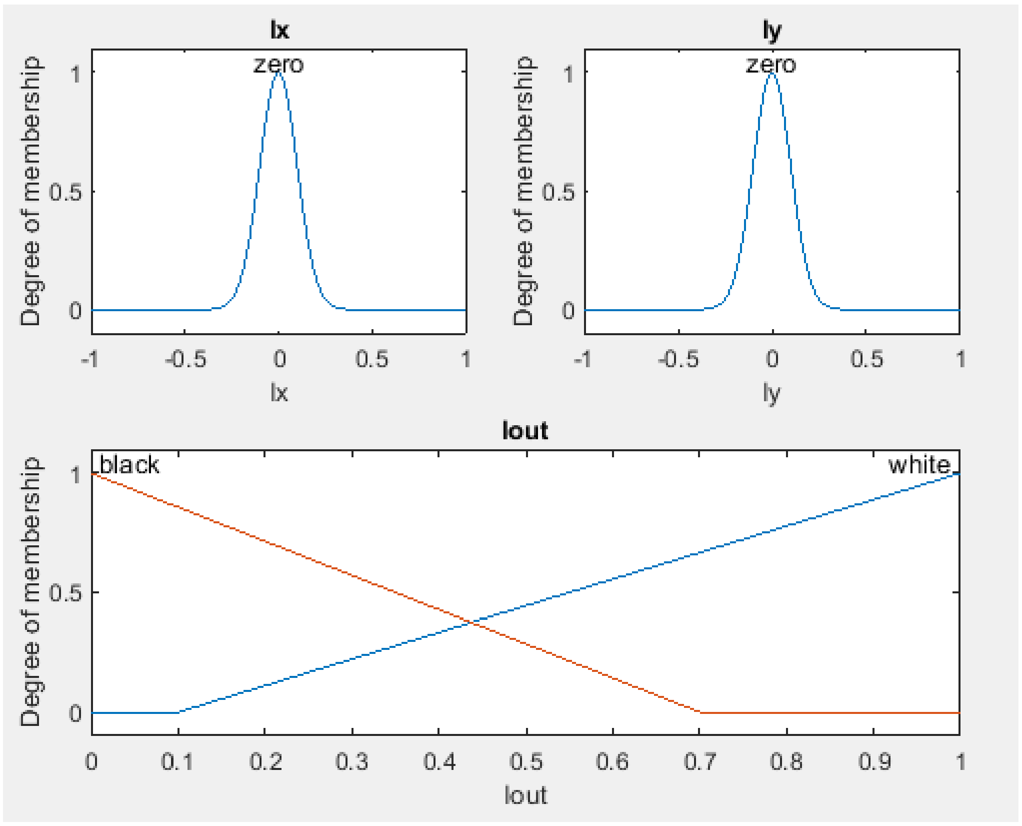


Figure 11.6 Degree of Membership

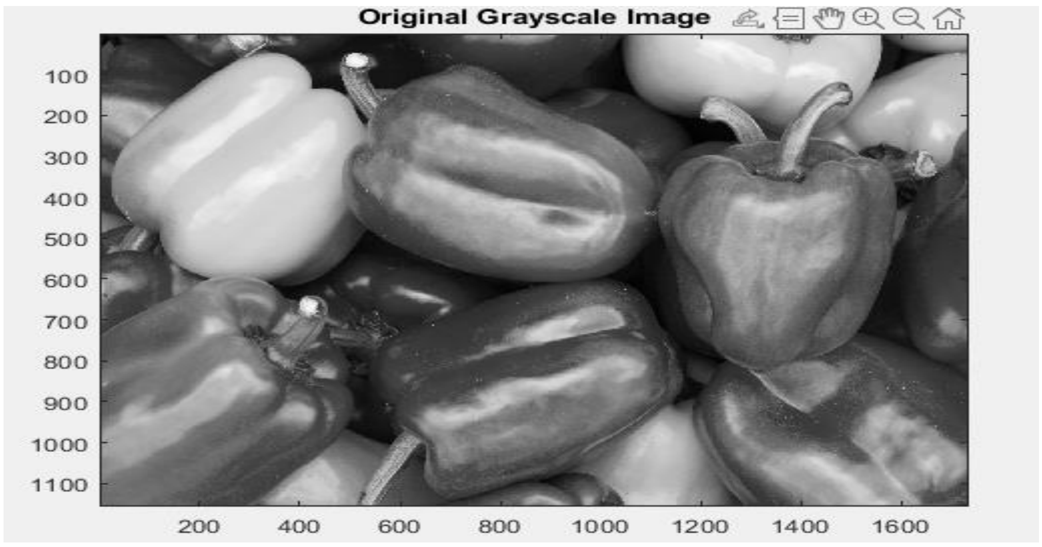


Figure 11.7 Original Grayscale Image

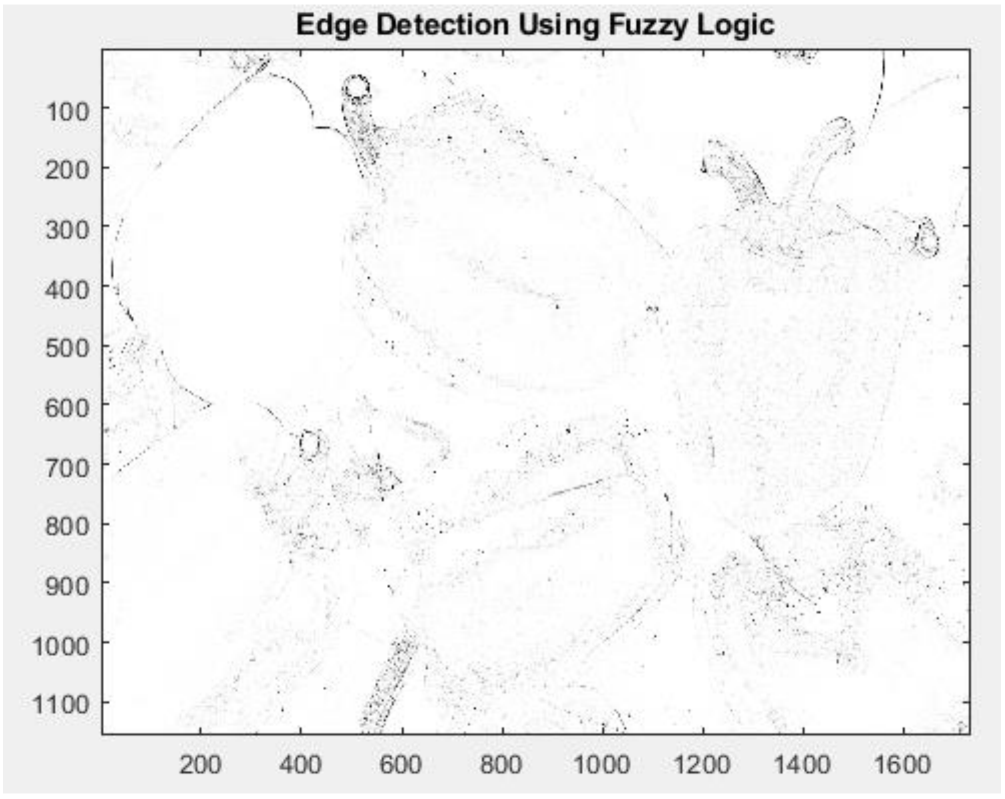


Figure 11.8 Edge Detection Using Fuzzy LOGIC

RESULT: By using Fuzzy Inference System (FIS) a set of fuzzy rule base between some key image parameters has been plotted and output is calculated after defuzzification.

DISCUSSION:

Q1. What is fuzzy inference?

Q2. What is fuzzy rule base?

Q3. What are the applications of fuzzy inference system?

Q4. What is Mamdani fuzzy inference system?

Q5. What is Takagi-Sugeno Fuzzy interference system?

EXPERIMENT NO. 12

AIM: Design a Fuzzy PID controller using Matlab for a Dc Motor.

SOFTWARE USED: MATLAB 7.1

THEORY:

The proportional, integral and derivate (KP, KI, KD) gains of the PID controller are adjusted according to FUZZY LOGIC. First, the fuzzy logic controller is designed according to fuzzy rules so that the systems are fundamentally robust. There are 25 fuzzy rules for self-tuning of each parameter of PID controller. The FLC has two inputs. One is the motor speed error between the reference and actual speed and the second is change in speed error (speed error derivative).Secondly, the output of the FLC i.e. the parameters of PID controller are used to control the speed of the DC Motor. The study shows that both precise characters of PID controllers and flexible characters of fuzzy controller are present in fuzzy selftuning PID controller. The simulation results demonstrate that the designed self-tuned PID controller realize a good dynamic behavior of the DC motor, a perfect speed tracking with less rise and settling time, minimum overshoot, minimum steady state error and give better performance compared to conventional PID controller

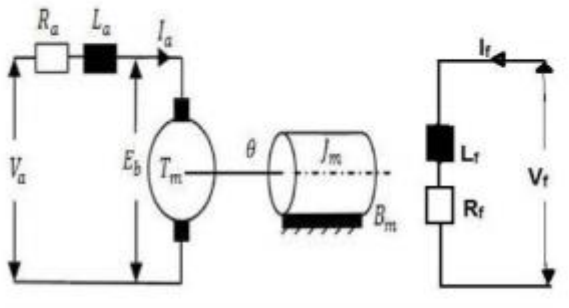


Figure 12.1: Model of DC Motor.

DC motors are most suitable for wide range speed control and are there for many adjustable speed drives. Intentional speed variation carried out manually or automatically to control the speed of DC motors.

Fuzzy Logic Controller

Fuzzy systems are knowledge based or rule based systems. The heart of a fuzzy system is a knowledge base consisting of the so- called If-Then rules. A fuzzy If-Then statement in which some words are characterized by continuous membership functions. After defining the fuzzy sets and assigning their membership functions, rules must be written to describe the action to be taken for each combination of control variables. These rules will relate the input variables to the output variable using If-Then statements which allow decisions to be made. The If (condition) is an antecedent to the Then (conclusion) of each rule. Each rule in general can be represented in the following manner: If (antecedent) Then (consequence). For example: If the speed of the car is high, then apply less force to the accelerator.

In order to define fuzzy membership function, designers choose many different shapes based on their preference and experience. There are generally four types of membership functions used:

1. Trapezoidal MF
2. Triangular MF
3. Gaussian MF
4. Generalized bell MF

Implementation of an FLC requires the choice of four key factors

1. Number of fuzzy sets that constitute linguistic variables.
2. Mapping of the measurements onto the support sets.
3. Control protocol that determines the controller behavior.
4. Shape of membership functions.

PID parameters fuzzy self-tuning is to find the fuzzy relationship between the three parameters of PID and "e" and "de", and according to the principle of fuzzy control, to modify the three parameters in order to meet different requirements for control parameters when "e" and "de" are different, and to make the control object a good dynamic and static performance

In order to improve the performance of FLC, the rules and membership functions are adjusted. The membership functions are adjusted by making the area of membership functions near ZE region narrower to produce finer control resolution. On the other hand, making the area far from ZE region wider gives faster control response. Also the performance can be improved by changing the severity of rules. An experiment to study the effect of rise time (Tr), maximum overshoot (Mp) and steady-state error (SSE) when varying KP, KI and KD was conducted. The results of the experiment were used to develop 25-rules for the FLC of KP, KI and KD are the output variables and from error and change of error are the input variables. Triangular membership functions are selected.

OUTPUT

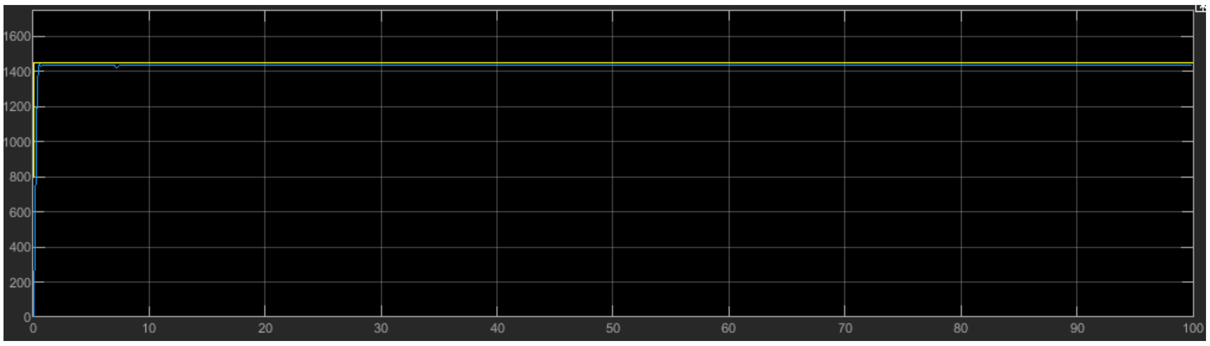


Figure 12.2

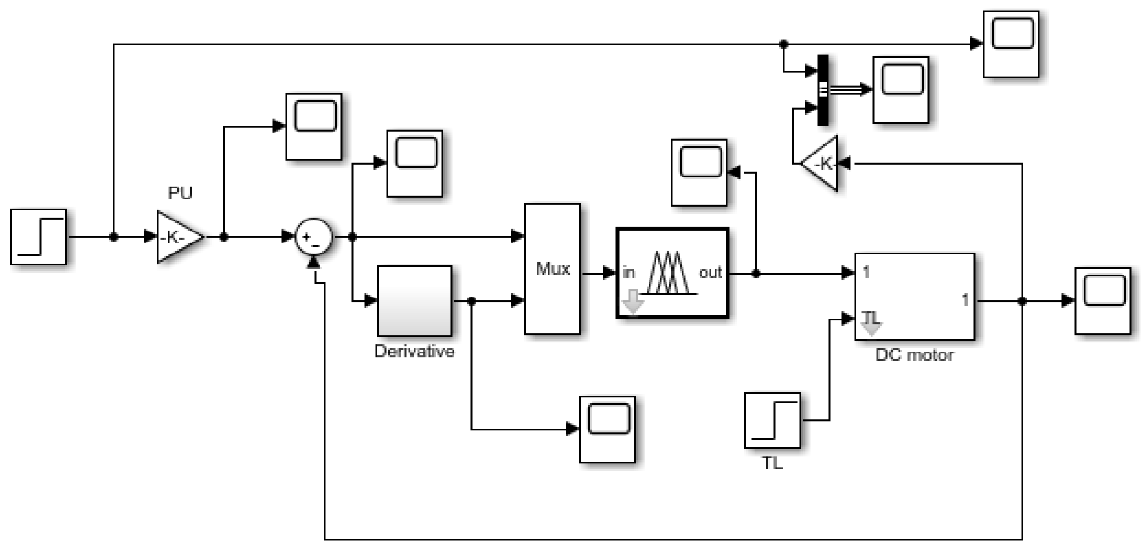


Figure 12.3 Simulink Model

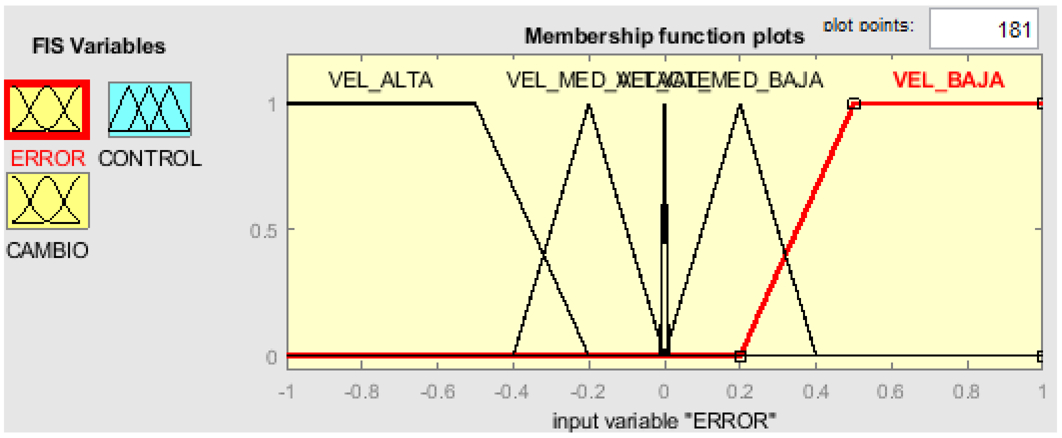


Figure 12.4 Membership functions plot

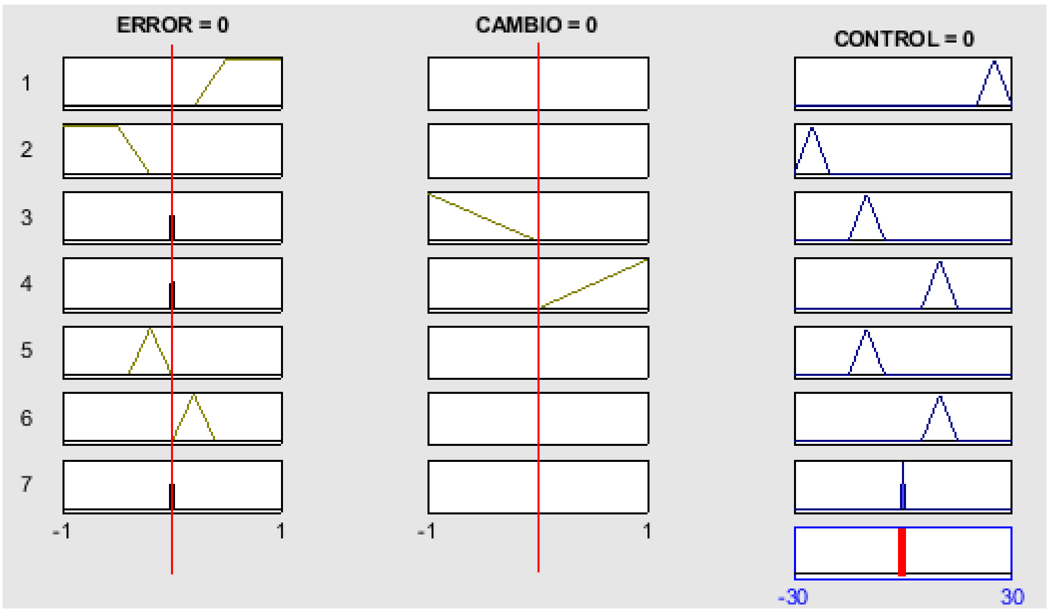


Figure 12.5 Fuzzy Rules

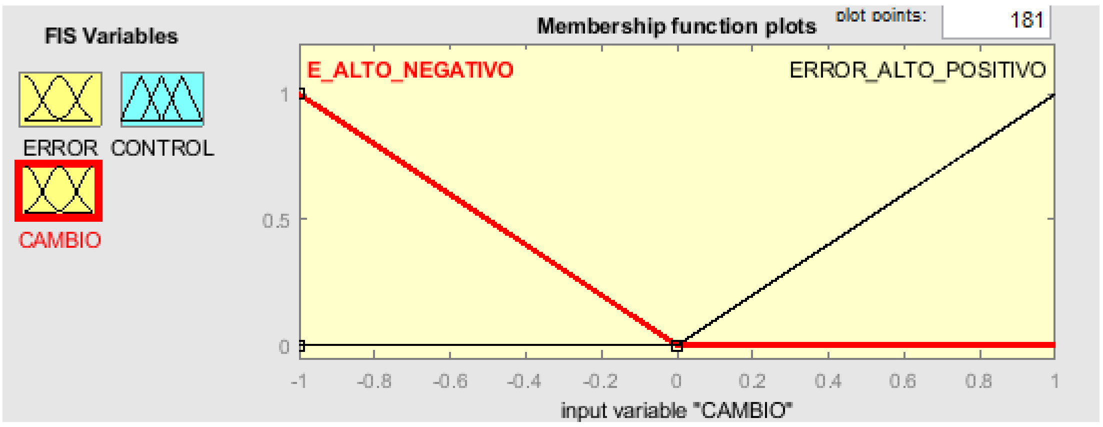


Figure 12.6 Membership Function Plots

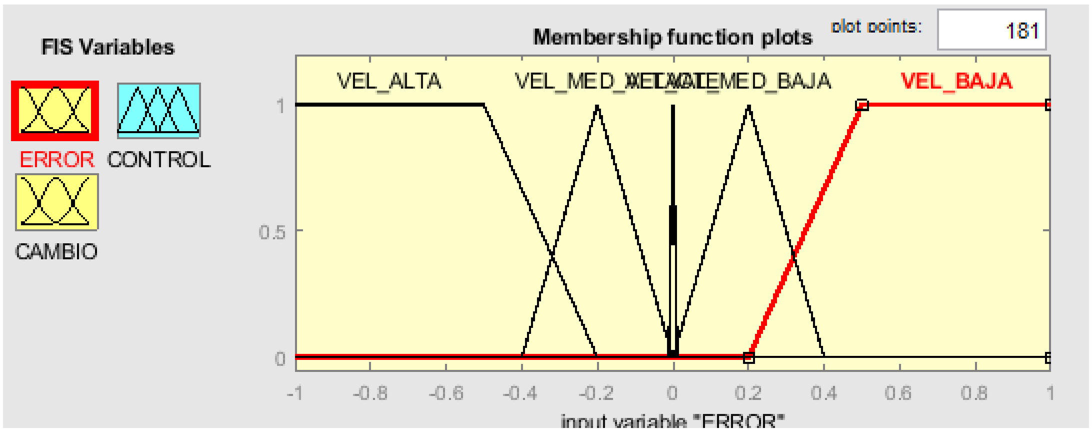
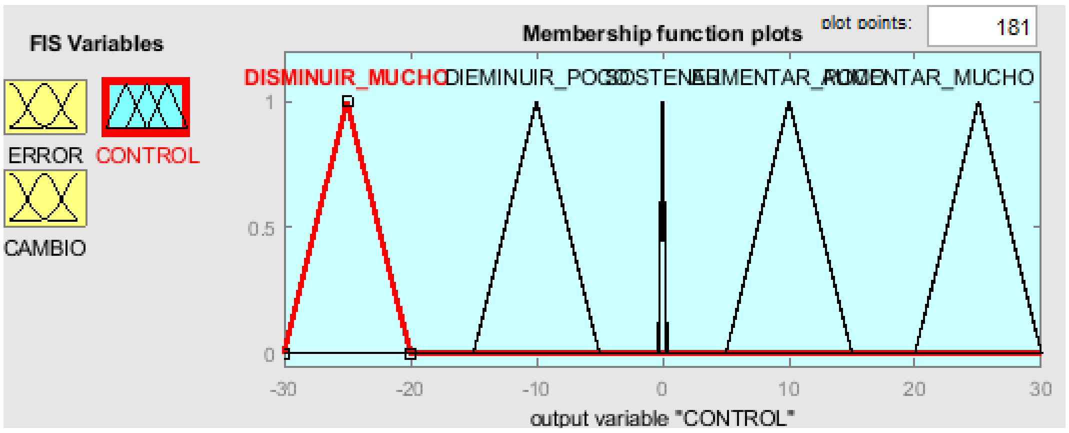
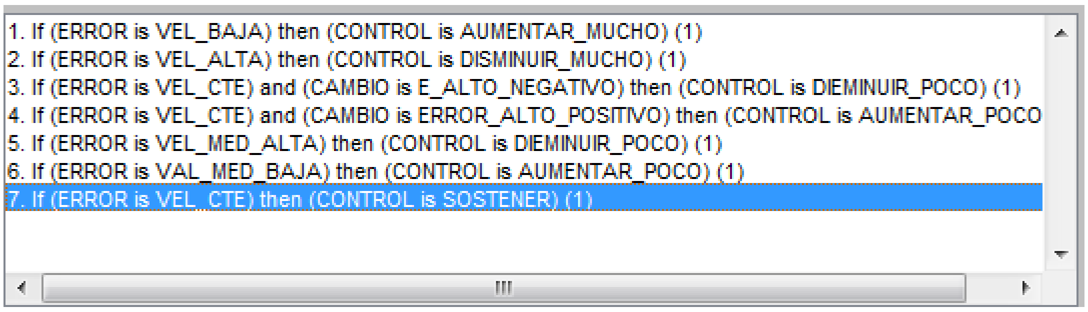


Figure 12.7





1. If (ERROR is VEL_BAJA) then (CONTROL is AUMENTAR_MUCHO) (1)
2. If (ERROR is VEL_ALTA) then (CONTROL is DISMINUIR_MUCHO) (1)
3. If (ERROR is VEL_CTE) and (CAMBIO is E_ALTO_NEGATIVO) then (CONTROL is DIEMINUIR_POCO) (1)
4. If (ERROR is VEL_CTE) and (CAMBIO is ERROR_ALTO_POSITIVO) then (CONTROL is AUMENTAR_POCO
5. If (ERROR is VEL_MED_ALTA) then (CONTROL is DIEMINUIR_POCO) (1)
6. If (ERROR is VAL_MED_BAJA) then (CONTROL is AUMENTAR_POCO) (1)
7. If (ERROR is VEL_CTE) then (CONTROL is SOSTENER) (1)

Figure 12.8

RESULT: The fuzzy controller adjusted the proportional integrator and derivative gains of the PID controller according to speed error and change in speed error.

DISCUSSION:

Q1. What is fuzzy PID controller?

Q2. How is fuzzy logic implemented?

Q3. How can PID be used to control the speed of a DC motor?

Q4. Why is fuzzy logic used?

Q5. What are the types of fuzzy logic sets?

EXPERIMENT NO.13

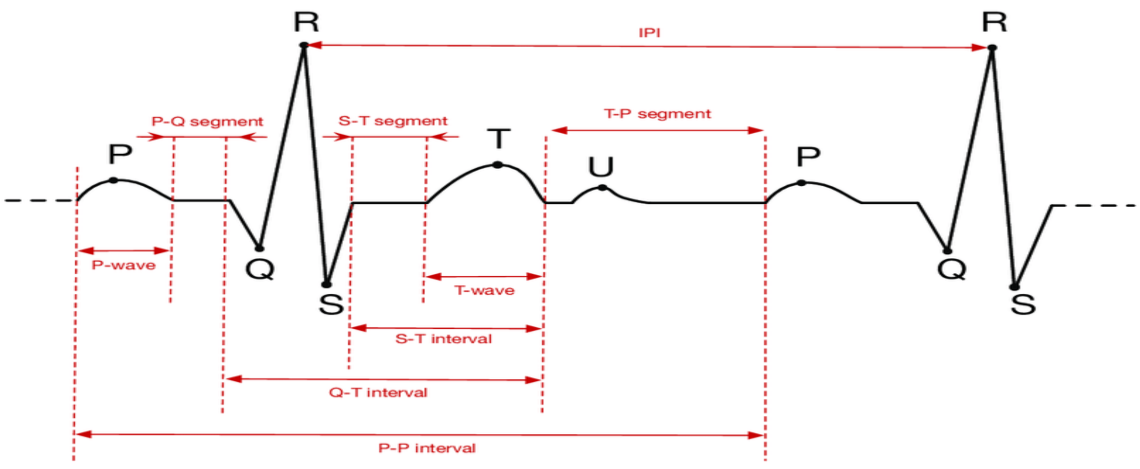
AIM: Classify ECG signals using Neural networks.

SOFTWARE USED: MATLAB 7.1

THEORY: An Electrocardiograph (ECG) is a Cartesian representation of the electrical potential generated by the heart.The early detection gives the information about heart abnormalities and increase life of human. ECG is used to measure the rate and regularity of heartbeats as well as the size and position of the chambers, the presence of any damage to the heart, and the effects of drugs or devices used to regulate the heart. To acquire the signal, ECG devices with varying number of electrodes (3– 12) can be used

ECG is used to measure the rate and regularity of heartbeats as well as the size and position of the chambers, the presence of any damage to the heart, and the effects of drugs or devices used to regulate the heart To acquire the signal, ECG devices with varying number of electrodes (3– 12) can be used

The ECG may roughly be divided into the phases of depolarization and repolarisation of the muscle fibers making up the heart. The depolarization phases correspond to the P-wave (atrial depolarization) and QRS-wave (ventricles depolarization). The repolarisation phases correspond to the T-wave and U-wave (ventricular repolarisation)



PROGRAM:

```
clc;
close all;
clear all;

x=0.01:0.01:2;
default=input('Press 1 if u want default ecg signal else press 2:\n');
if(default==1)
    li=30/72;

    a_pwav=0.25;
    d_pwav=0.09;
    t_pwav=0.16;

    a_qwav=0.025;
    d_qwav=0.066;
    t_qwav=0.166;

    a_qrswav=1.6;
    d_qrswav=0.11;

    a_swav=0.25;
    d_swav=0.066;
    t_swav=0.09;

    a_twav=0.35;
    d_twav=0.142;
    t_twav=0.2;



 a_uwav=0.035;
```

```
      d_uwav=0.0476;
      t_uwav=0.433;
else
    rate=input('\n\nenter the heart beat rate :');
    li=30/rate;

    %p wave specifications
    fprintf('\n\np wave specifications\n');
    d=input('Enter 1 for default specification else press 2: \n');
    if(d==1)
        a_pwav=0.25;
        d_pwav=0.09;
        t_pwav=0.16;
    else
       a_pwav=input('amplitude = ');
       d_pwav=input('duration = ');
       t_pwav=input('p-r interval = ');
       d=0;
    end

        %q wave specifications
    fprintf('\n\nq wave specifications\n');
    d=input('Enter 1 for default specification else press 2: \n');
    if(d==1)
        a_qwav=0.025;
        d_qwav=0.066;
        t_qwav=0.166;
    else
       a_qwav=input('amplitude = ');



  d_qwav=input('duration = ');
  t_qwav=0.166;
  d=0;
end
%qrs wave specifications
fprintf('\n\nqrs wave specifications\n');
d=input('Enter 1 for default specification else press 2: \n');
if(d==1)
   a_qrswav=1.6;
   d_qrswav=0.11;
else
  a_qrswav=input('amplitude = ');
  d_qrswav=input('duration = ');
  d=0;
end
%s wave specifications
fprintf('\n\ns wave specifications\n');
d=input('Enter 1 for default specification else press 2: \n');
if(d==1)
   a_swav=0.25;
   d_swav=0.066;
   t_swav=0.09;
else
  a_swav=input('amplitude = ');
  d_swav=input('duration = ');
  t_swav=0.09
d=0;
end
%t wave specifications
fprintf('\n\nt wave specifications\n');


d=input('Enter 1 for default specification else press 2: \n');
```

```
    if(d==1)
        a_twav=0.35;
        d_twav=0.142;
        t_twav=0.2;
    else
       a_twav=input('amplitude = ');
       d_twav=input('duration = ');
       t_twav=input('s-t interval = ');
       d=0;
    end
    %u wave specifications
    fprintf('\n\nu wave specifications\n');
    d=input('Enter 1 for default specification else press 2: \n');
    if(d==1)
        a_uwav=0.035;
        d_uwav=0.0476;
        t_uwav=0.433;
    else
       a_uwav=input('amplitude = ');
       d_uwav=input('duration = ');
       t_uwav=0.433;
       d=0;
    end
end
 pwav=p_wav(x,a_pwav,d_pwav,t_pwav,li);
 %qwav output
 qwav=q_wav(x,a_qwav,d_qwav,t_qwav,li);
 %qrswav output
 qrswav=qrs_wav(x,a_qrswav,d_qrswav,li);


 %swav output
 swav=s_wav(x,a_swav,d_swav,t_swav,li);
 %twav output
 twav=t_wav(x,a_twav,d_twav,t_twav,li);
 %uwav output
 uwav=u_wav(x,a_uwav,d_uwav,t_uwav,li);
 %ecg output
 ecg=pwav+qrswav+twav+swav+qwav+uwav;
 figure(1)
 plot(x,ecg);
  function [pwav]=p_wav(x,a_pwav,d_pwav,t_pwav,li)
l=li;
a=a_pwav;
x=x+t_pwav;
b=(2*l)/d_pwav;
n=100;
p1=1/l;
p2=0;
for i = 1:n
    harm1=(((sin((pi/(2*b))*(b-(2*i))))/(b-
(2*i))+(sin((pi/(2*b))*(b+(2*i))))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
    p2=p2+harm1;
end
pwav1=p1+p2;
pwav=a*pwav1;
  end
 function [qwav]=q_wav(x,a_qwav,d_qwav,t_qwav,li)
l=li;
x=x+t_qwav;
a=a_qwav;


b=(2*l)/d_qwav;
```

```
n=100;
q1=(a/(2*b))*(2-b);
q2=0;
for i = 1:n
    harm5=(((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b)))*cos((i*pi*x)/l);
    q2=q2+harm5;
end
qwav=-1*(q1+q2);
 end
 function [qrswav]=qrs_wav(x,a_qrswav,d_qrswav,li)
l=li;
a=a_qrswav;
b=(2*l)/d_qrswav;
n=100;
qrs1=(a/(2*b))*(2-b);
qrs2=0;
for i = 1:n
    harm=(((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b)))*cos((i*pi*x)/l);
    qrs2=qrs2+harm;
end
qrswav=qrs1+qrs2;
 end
 function [swav]=s_wav(x,a_swav,d_swav,t_swav,li)
l=li;
x=x-t_swav;
a=a_swav;
b=(2*l)/d_swav;
n=100;
s1=(a/(2*b))*(2-b);


s2=0;
```

```
for i = 1:n
    harm3=(((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b)))*cos((i*pi*x)/l);
    s2=s2+harm3;
end
swav=-1*(s1+s2);
 end
 function [twav]=t_wav(x,a_twav,d_twav,t_twav,li)
l=li;
a=a_twav;
x=x-t_twav-0.045;
b=(2*l)/d_twav;
n=100;
t1=1/l;
t2=0;
for i = 1:n
    harm2=(((sin((pi/(2*b))*(b-(2*i))))/(b-
(2*i))+(sin((pi/(2*b))*(b+(2*i))))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
    t2=t2+harm2;
end
twav1=t1+t2;
twav=a*twav1;
 end
 function [uwav]=u_wav(x,a_uwav,d_uwav,t_uwav,li)
l=li;
a=a_uwav;
x=x-t_uwav;
b=(2*l)/d_uwav;
n=100;
u1=1/l;


u2=0;
```

```
for i = 1:n
    harm4=(((sin((pi/(2*b))*(b-(2*i))))/(b-
(2*i))+(sin((pi/(2*b))*(b+(2*i))))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
    u2=u2+harm4;
end
uwav1=u1+u2;
uwav=a*uwav1;
 end
```

OUTPUT:

```
Press 1 if u want default ecg signal else press 2:

1
```

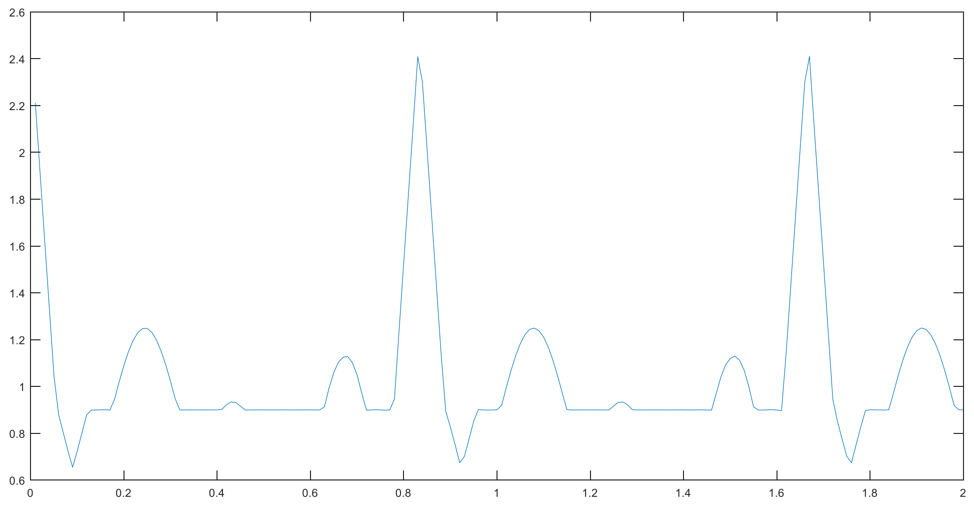


Figure 13.1

RESULT: ECG signals using Neural networks is generated as shown in Figure.13.1

DISCUSSION:

Q1. How do you classify ECG signals?

Q2. What are ECG signals used for?

Q3. How many types of ECG are there?

Q4. What is the principle of ECG?

EXPERIMENT NO.14

AIM: Generate modulation and demodulation of ASK. Plot the average probability of symbol error as a function of SNR Eb/No, where Eb is the transmitted energy per bit and No/2 is the double sided power spectral density of additive white Gaussian noise (AWGN) with zero mean.

SOFTWARE USED: MATLAB 7.1

THEORY:

Amplitude Shift Keying ASKASK is a type of Amplitude Modulation which represents the binary data in the form of variations in the amplitude of a signal.

Any modulated signal has a high frequency carrier. The binary signal when ASK modulated, gives a **zero** value for **Low** input while it gives the **carrier output** for **High** input.

The following figure represents ASK modulated waveform along with its input.

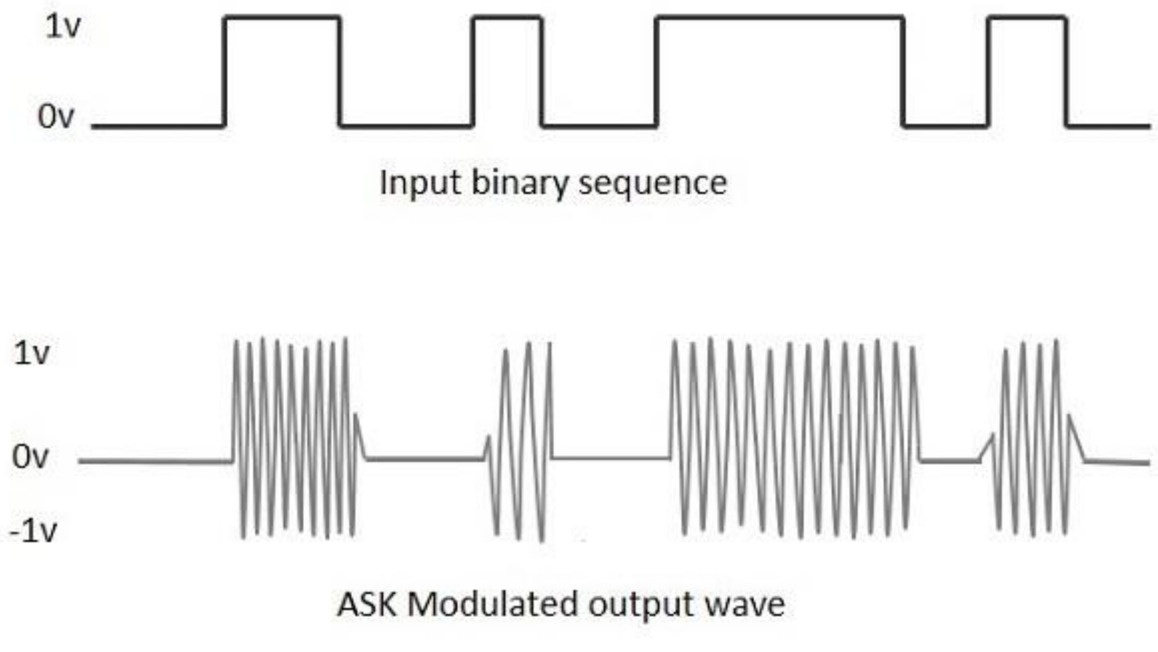


Figure 14.1

To find the process of obtaining this ASK modulated wave, let us learn about the working of the ASK modulator.

ASK Modulator

The ASK modulator block diagram comprises of the carrier signal generator, the binary sequence from the message signal and the band-limited filter. Following is the block diagram of the ASK Modulator.

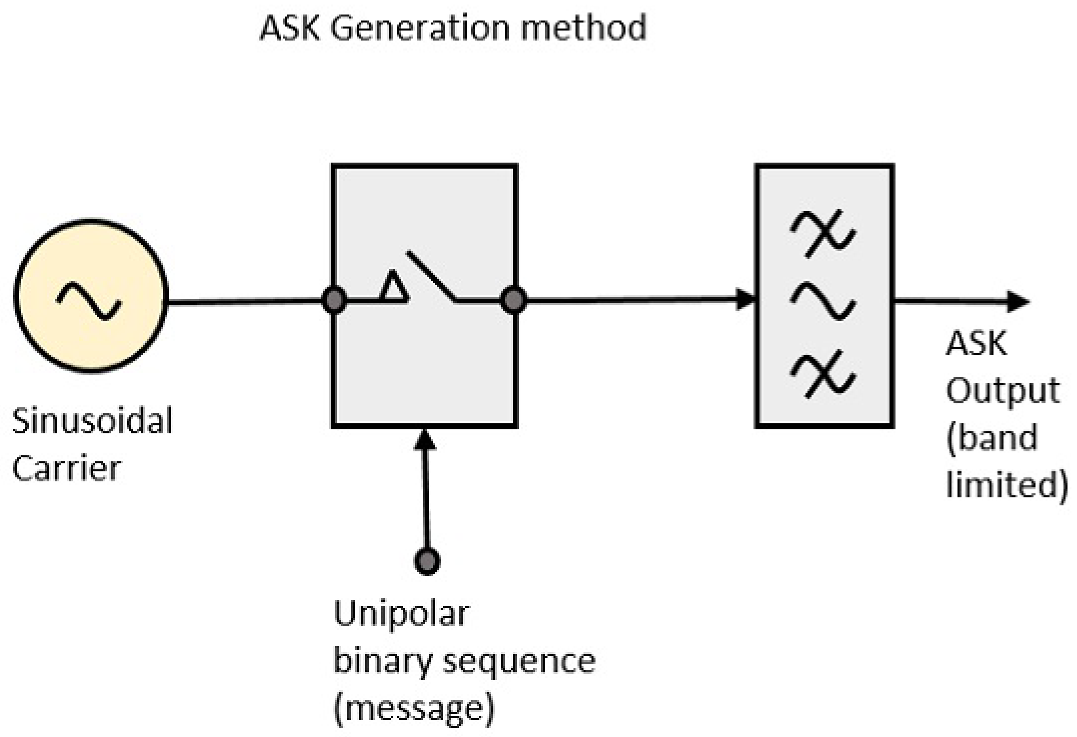


Figure 14.2

The carrier generator, sends a continuous high-frequency carrier. The binary sequence from the message signal makes the unipolar input to be either High or Low. The high signal closes the switch, allowing a carrier wave. Hence, the output will be the carrier signal at high input. When there is low input, the switch opens, allowing no voltage to appear. Hence, the output will be low.

The band-limiting filter, shapes the pulse depending upon the amplitude and phase characteristics of the band-limiting filter or the pulse-shaping filter.

ASK Demodulator

There are two types of ASK Demodulation techniques. They are −

- Asynchronous ASK Demodulation/detection
- Synchronous ASK Demodulation/detection

The clock frequency at the transmitter when matches with the clock frequency at the receiver, it is known as a **Synchronous method**, as the frequency gets synchronized. Otherwise, it is known as **Asynchronous**.

Asynchronous ASK Demodulator

The Asynchronous ASK detector consists of a half-wave rectifier, a low pass filter, and a comparator. Following is the block diagram for the same.

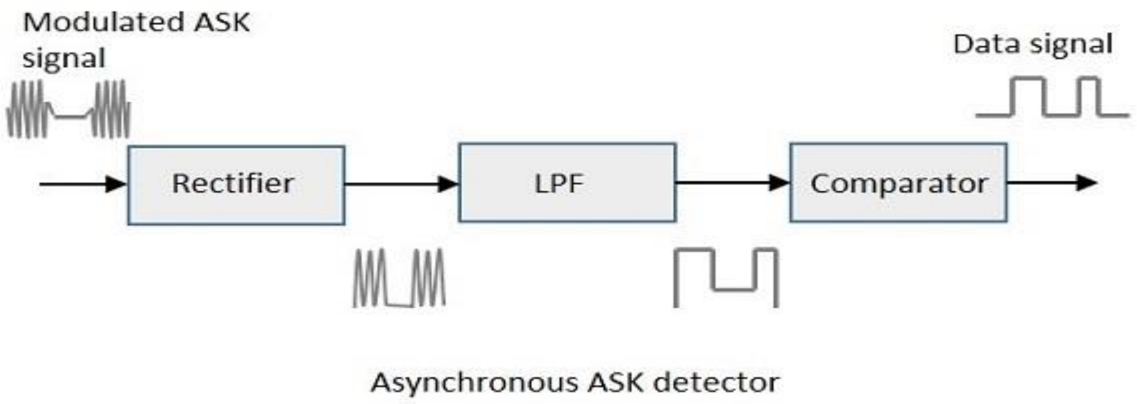


Figure 14.3

The modulated ASK signal is given to the half-wave rectifier, which delivers a positive half output. The low pass filter suppresses the higher frequencies and gives an envelope detected output from which the comparator delivers a digital output.

Synchronous ASK Demodulator

Synchronous ASK detector consists of a Square law detector, low pass filter, a comparator, and a voltage limiter. Following is the block diagram for the same.

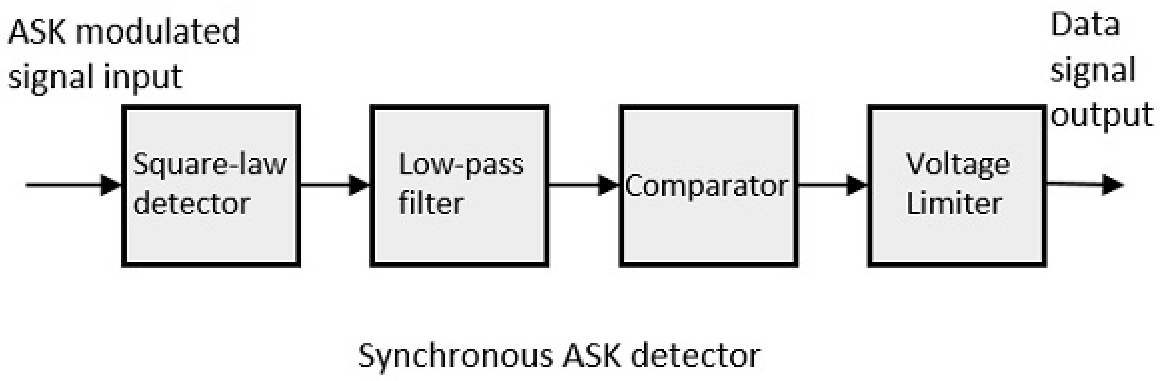


Figure 14.4

The ASK modulated input signal is given to the Square law detector. A square law detector is one whose output voltage is proportional to the square of the amplitude modulated input voltage. The low pass filter minimizes the higher frequencies. The comparator and the voltage limiter help to get a clean digital output.

Part 1: ASK Modulation and Demodulation

PROGRAM:

```
%>>>>>>>>> MATLAB code for binary ASK modulation and de-modulation
>>>>>>>%

clc;
clear all;
close all;
```

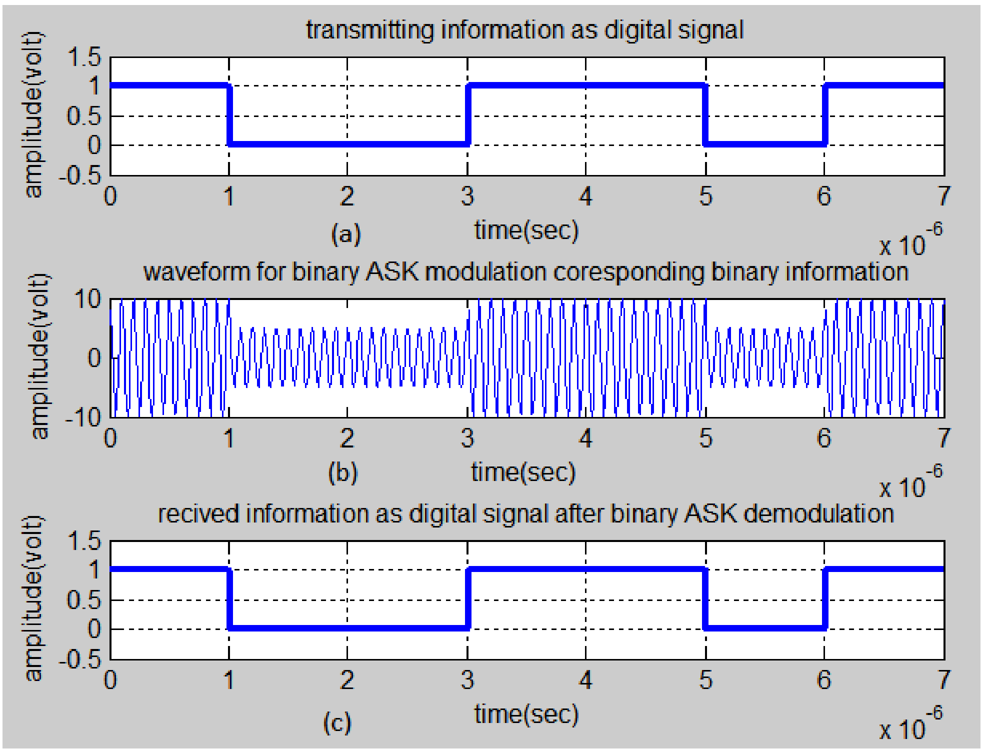
```
x=[ 1 0 0 1 1 0 1];                              % Binary Information
bp=.000001;                                           % bit period
disp(' Binary information at Trans mitter :');
disp(x);
%XX representation of transmitting binary information as digital signal XXX
bit=[];
for n=1:1:length(x)
    if x(n)==1;
       se=ones(1,100);
    else x(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
subplot(3,1,1);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('transmitting information as digital signal');
%XXXXXXXXXXXXXXXXXXXXXXX Binary-ASK modulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%
A1=10;                    % Amplitude of carrier signal for information 1
A2=5;                     % Amplitude of carrier signal for information 0
br=1/bp;                                            % bit rate
f=br*10;                                    % carrier frequency
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for (i=1:1:length(x))
    if (x(i)==1)
```

```
        y=A1*cos(2*pi*f*t2);
    else
        y=A2*cos(2*pi*f*t2);
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(3,1,2);
plot(t3,m);
xlabel('time(sec)');
ylabel('amplitude(volt)');
title('waveform for binary ASK modulation coresponding binary information');
%XXXXXXXXXXXXXXXXXXXXX Binary ASK demodulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
mn=[];
for n=ss:ss:length(m)
  t=bp/99:bp/99:bp;
  y=cos(2*pi*f*t);                              % carrier siignal
  mm=y.*m((n-(ss-1)):n);
  t4=bp/99:bp/99:bp;
  z=trapz(t4,mm)                                  % intregation
  zz=round((2*z/bp))
  if(zz>7.5)                          % logic level = (A1+A2)/2=7.5
    a=1;
  else
    a=0;
  end
  mn=[mn a];


end
disp(' Binary information at Reciver :');
disp(mn);
%XXXXX Representation of binary information as digital signal which achived
%after ASK demodulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X
bit=[];
for n=1:length(mn);
    if mn(n)==1;
       se=ones(1,100);
    else mn(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t4=bp/100:bp/100:100*length(mn)*(bp/100);
subplot(3,1,3)
plot(t4,bit,'LineWidth',2.5);grid on;
axis([ 0 bp*length(mn) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('recived information as digital signal after binary ASK demodulation');

%>>>>>>>>>>>>>>>>>>>>>>>>>> end of program
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

OUTPUT:



Part 2: average probability of symbol error as a function of SNR Eb/No

PROGRAM:

RESULT:

DISCUSSION:

Q1. What is the use of Amplitude Shift Keying?

Q2. Why amplitude shift keying is called on off keying?

Q3.What are advantaged of ASK

Q4. What are disadvantages of ASK?

Q5. What are applications of ASK?

EXPERIMENT NO.15

AIM: Generate modulation and demodulation of FSK. Plot the average probability of symbol error as a function of SNR Eb/No, where Eb is the transmitted energy per bit and No/2 is the double sided power spectral density of additive white Gaussian noise (AWGN) with zero mean.

SOFTWARE USED: MATLAB 7.1

THEORY

Frequency Shift Keying FSKFSK is the digital modulation technique in which the frequency of the carrier signal varies according to the digital signal changes. FSK is a scheme of frequency modulation.

The output of a FSK modulated wave is high in frequency for a binary High input and is low in frequency for a binary Low input. The binary **1s** and **0s** are called Mark and Space frequencies.

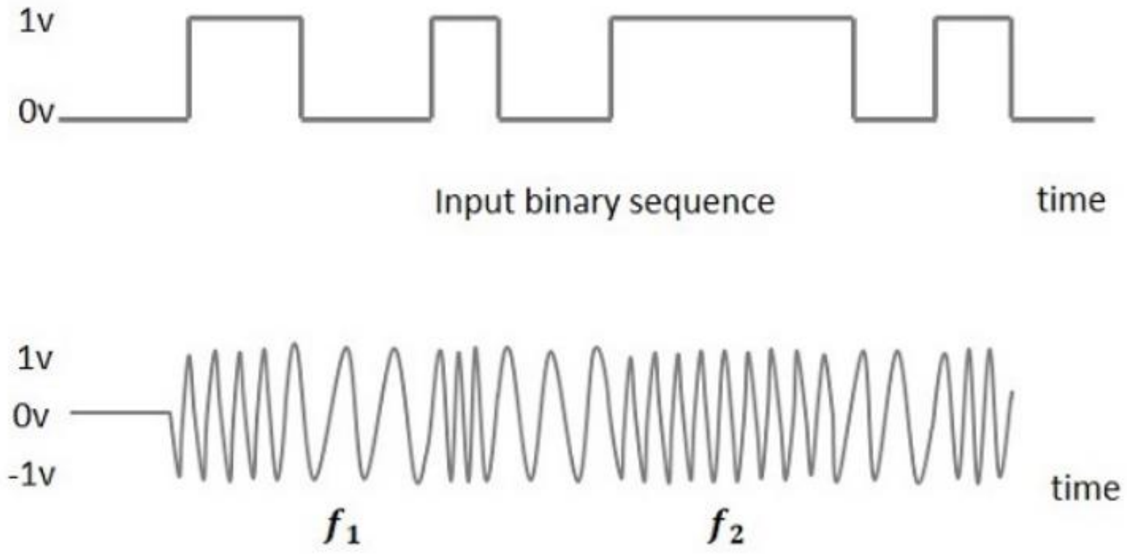


Figure 15.1

FSK Modulator

The FSK modulator block diagram comprises of two oscillators with a clock and the input binary sequence. Following is its block diagram.

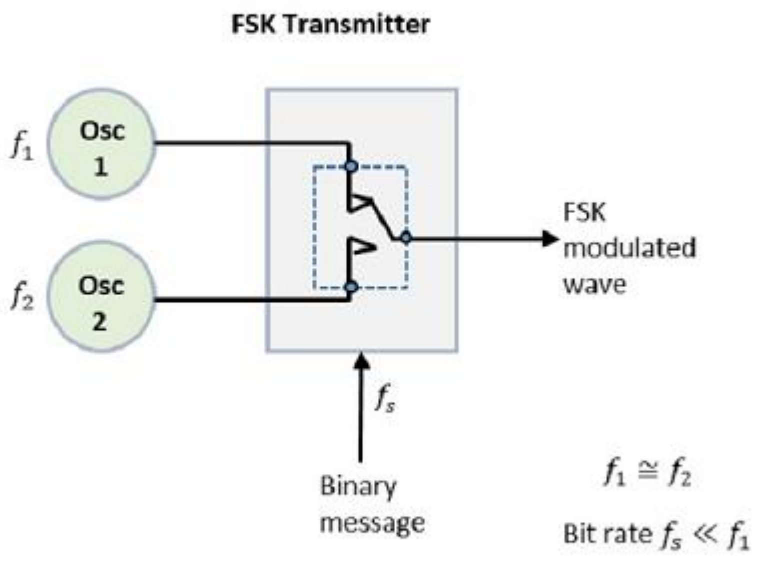


Figure 15.2

The two oscillators, producing a higher and a lower frequency signals, are connected to a switch along with an internal clock. To avoid the abrupt phase discontinuities of the output waveform during the transmission of the message, a clock is applied to both the oscillators, internally. The binary input sequence is applied to the transmitter so as to choose the frequencies according to the binary input.

FSK Demodulator

There are different methods for demodulating a FSK wave. The main methods of FSK detection are **asynchronous detector** and **synchronous detector**. The synchronous detector is a coherent one, while asynchronous detector is a non-coherent one.

Asynchronous FSK Detector

The block diagram of Asynchronous FSK detector consists of two band pass filters, two envelope detectors, and a decision circuit. Following is the diagrammatic representation.

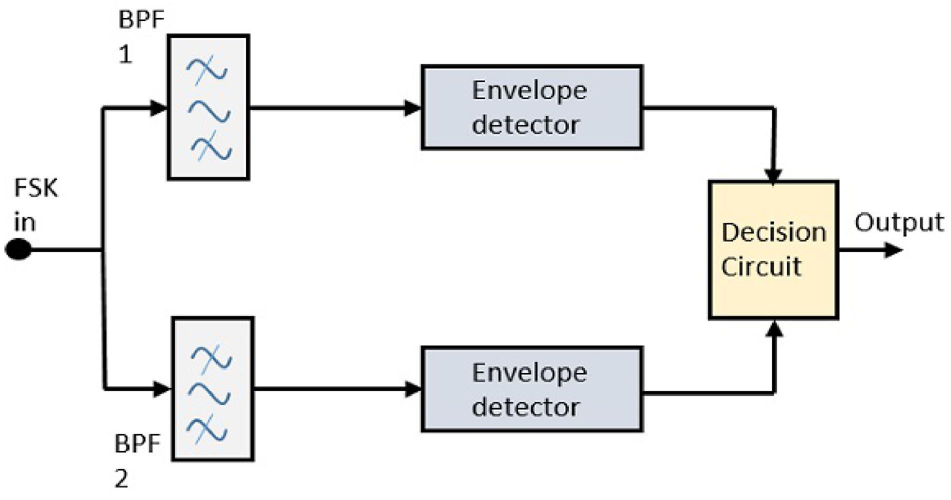


Figure 15.3

The FSK signal is passed through the two Band Pass Filters BPFsBPFs, tuned to **Space** and **Mark** frequencies. The output from these two BPFs look like ASK signal, which is given to the envelope detector. The signal in each envelope detector is modulated asynchronously.

The decision circuit chooses which output is more likely and selects it from any one of the envelope detectors. It also re-shapes the waveform to a rectangular one.

Synchronous FSK Detector

The block diagram of Synchronous FSK detector consists of two mixers with local oscillator circuits, two band pass filters and a decision circuit. Following is the diagrammatic representation.

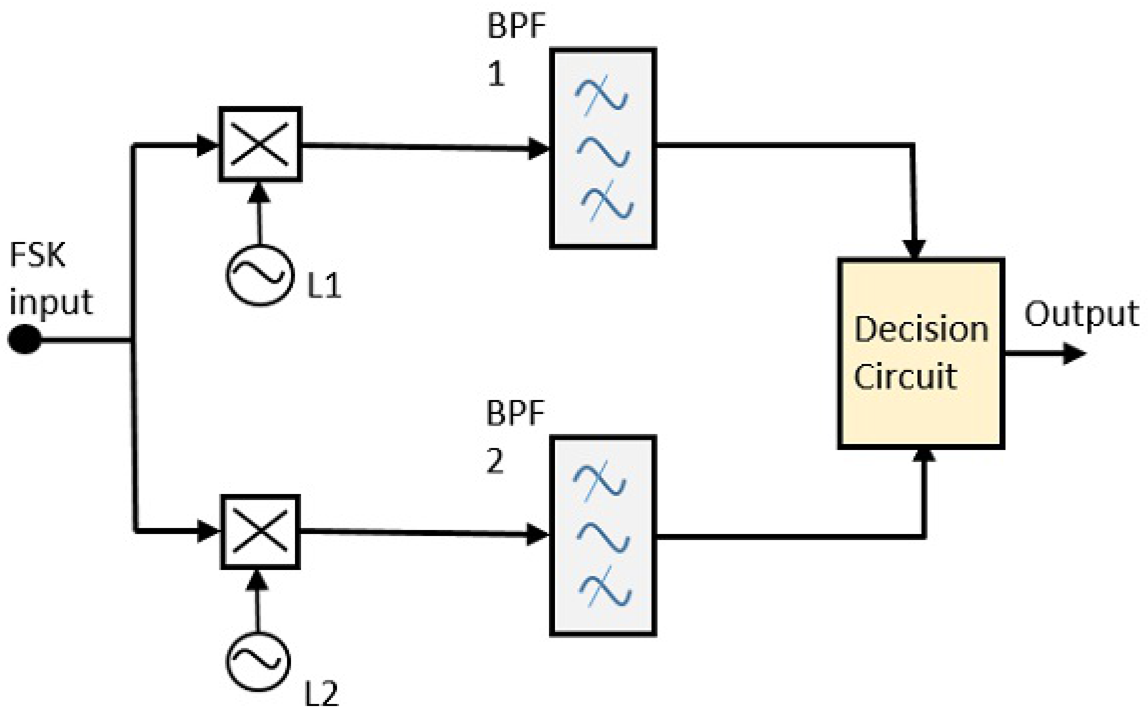


Figure 15.4

The FSK signal input is given to the two mixers with local oscillator circuits. These two are connected to two band pass filters. These combinations act as demodulators and the decision circuit chooses which output is more likely and selects it from any one of the detectors. The two signals have a minimum frequency separation.

For both of the demodulators, the bandwidth of each of them depends on their bit rate. This synchronous demodulator is a bit complex than asynchronous type demodulators.

Part 1: FSK Modulation and Demodulation

PROGRAM

```
%>>>>>>>>> MATLAB code for binary FSK modulation and de-modulation
>>>>>>>%
clc;

clear all;

close all;

x=[ 1 0 0 1 1 0 1];                         % Binary Information
```

```
bp=.000001;                                          % bit period
disp(' Binary information at Trans mitter :');
disp(x);
%XX representation of transmitting binary information as digital signal XXX
bit=[];
for n=1:1:length(x)
    if x(n)==1;
       se=ones(1,100);
    else x(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
subplot(3,1,1);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('transmitting information as digital signal');
%XXXXXXXXXXXXXXXXXXXXXXXX Binary-FSK modulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXX%
A=5;                                    % Amplitude of carrier signal
br=1/bp;                                               % bit rate
f1=br*8;                       % carrier frequency for information as 1
f2=br*2;                       % carrier frequency for information as 0
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for (i=1:1:length(x))
```

```
  if (x(i)==1)
   y=A*cos(2*pi*f1*t2);
    else
        y=A*cos(2*pi*f2*t2);
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(3,1,2);
plot(t3,m);
xlabel('time(sec)');
ylabel('amplitude(volt)');
title('waveform for binary FSK modulation coresponding binary information');
%XXXXXXXXXXXXXXXXXXXX Binary FSK demodulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
mn=[];
for n=ss:ss:length(m)
  t=bp/99:bp/99:bp;
  y1=cos(2*pi*f1*t);                % carrier siignal for information 1
  y2=cos(2*pi*f2*t);                % carrier siignal for information 0
  mm=y1.*m((n-(ss-1)):n);
  mmm=y2.*m((n-(ss-1)):n);
  t4=bp/99:bp/99:bp;
  z1=trapz(t4,mm)                                   % intregation
  z2=trapz(t4,mmm)                                   % intregation
  zz1=round(2*z1/bp)
  zz2= round(2*z2/bp)
  if(zz1>A/2)      % logic lavel= (0+A)/2 or (A+0)/2 or 2.5 ( in this case)
    a=1;
  else(zz2>A/2)
a=0;
  end
  mn=[mn a];
end
disp(' Binary information at Reciver :');
disp(mn);
%XXXXX Representation of binary information as digital signal which achived
%after demodulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXX
bit=[];
for n=1:length(mn);
    if mn(n)==1;
       se=ones(1,100);
    else mn(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t4=bp/100:bp/100:100*length(mn)*(bp/100);
subplot(3,1,3)
plot(t4,bit,'LineWidth',2.5);grid on;
axis([ 0 bp*length(mn) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('recived information as digital signal after binary FSK demodulation');
```

OUTPUT:

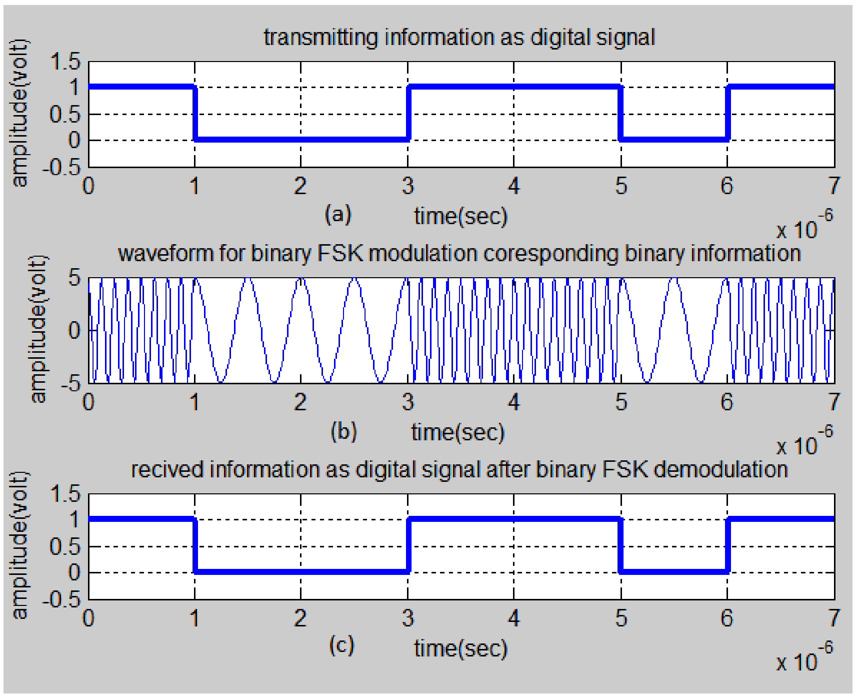


Figure 15.5

Part 2: average probability of symbol error as a function of SNR Eb/No

PROGRAM

```
%This program simulates BER of BFSK in AWGN channel%
 clear all; close all; clc;
 num_bit=10000;                     %Signal length
 max_run=20;                        %Maximum number of iterations for a single SNR
 Eb=1;                              %Bit energy
 SNRdB=0:1:10;                      %Signal to Noise Ratio (in dB)
 SNR=10.^(SNRdB/10);
 hand=waitbar(0,'Please Wait....');
 for count=1:length(SNR)             %Beginning of loop for different SNR
     avgError=0;


No=Eb/SNR(count);                    %Calculate noise power from SNR
     for run_time=1:max_run           %Beginning of loop for different runs
```

```
        waitbar((((count-1)*max_run)+run_time-1)/(length(SNRdB)*max_run));
Error=0;
        data=randint(1,num_bit);        %Generate binary data source
        s=data+j*(~data);               %Baseband BFSK modulation

        NI=sqrt(No/2)*randn(1,num_bit);
        NQ=sqrt(No/2)*randn(1,num_bit);
        N=NI+j*NQ;                      %Generate complex AWGN

        Y=s+N;                          %Received Signal

        for k=1:num_bit                 %Decision device taking hard decision and deciding
error
            Z(k)=real(Y(k))-imag(Y(k));
            if ((Z(k)>0 && data(k)==0)||(Z(k)<0 && data(k)==1))
                Error=Error+1;
            end
        end
        Error=Error/num_bit;            %Calculate error/bit
        avgError=avgError+Error;        %Calculate error/bit for different runs
    end                                 %Termination of loop for different runs
    BER_sim(count)=avgError/max_run;    %Calculate BER for a particular SNR
end                                     %Termination of loop for different SNR
BER_th=(1/2)*erfc(sqrt(SNR/2));         %Calculate analytical BER
close(hand);
semilogy(SNRdB,BER_th,'k');             %Plot BER
hold on
semilogy(SNRdB,BER_sim,'k*');
legend('Theoretical','Simulation',3);
axis([min(SNRdB) max(SNRdB) 10^(-5) 1]);
hold off
```

OUTPUT

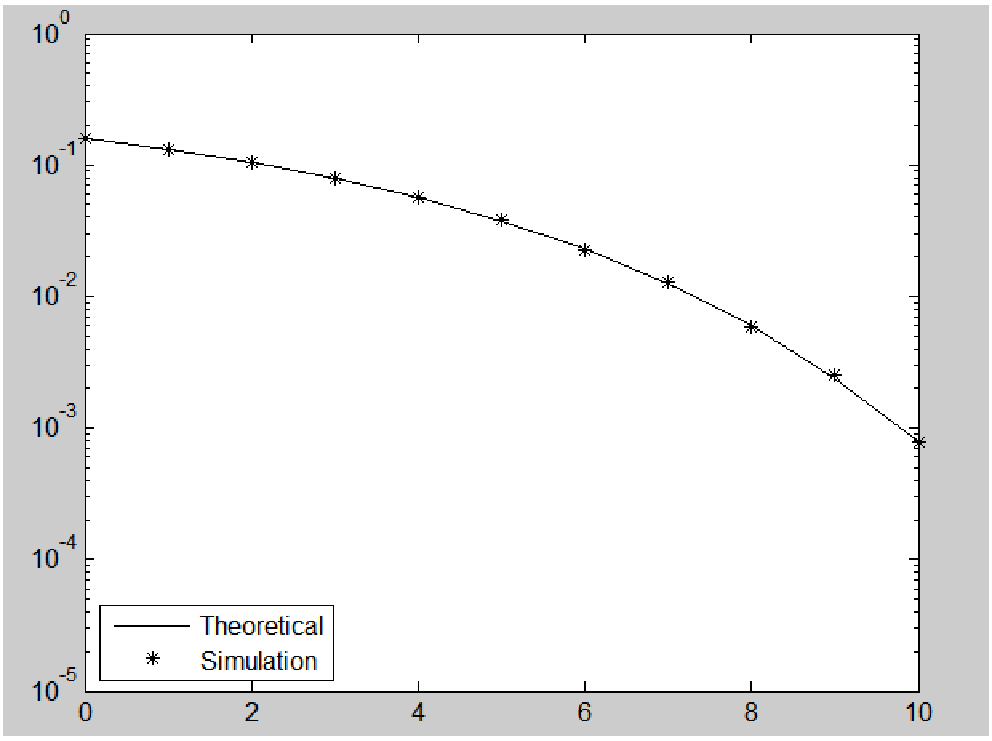


Figure 15.6 average probability of symbol error as a function of SNR Eb/No

RESULT: FSK modulation and demodulation is generated as shown in Figure 11.5. The average probability of symbol error as a function of SNR Eb/No is generated as shown in Figure 11.6.

DISCUSSION:

Q1. Where is FSK used?

Q2. What are the advantages of FSK?

Q3. What is the bandwidth of FSK?

Q4. How is an FSK signal generated?

Q5. Which is better FSK or PSK?

REFERENCES

- https://in.mathworks.com/matlabcentral/fileexchange/