1.GENERATION OF RANDOM DATA AT A GIVEN DATA RATE

Aim:-

To generate the random data at a given data rate.

Software Required:- Pc with MATLAB 7.0 or above

Theory:-

A maximum length sequence (MLS) is a type of pseudorandom binary sequence.

They are bit sequences generated using maximal linear feedback shift registers and are so called because they are periodic and reproduce every binary sequence that can be reproduced by the shift registers (i.e., for length-*m* registers they produce a sequence of length 2*m* − 1). A MLS is also sometimes called an **n-sequence** or an **m-sequence**. MLSs are spectrally flat, with the exception of a near-zero DC term.

These sequences may be represented as coefficients of irreducible polynomials in a polynomial ring over Z/2Z.

Practical applications for MLS include measuring impulse responses (e.g., of room reverberation). They are also used as a basis for deriving pseudo-random sequences in digital communication systems that employ direct-sequence spread spectrum and frequency-hopping spread spectrum transmission systems, and in the efficient design of some MRI experiments

Generation of maximum length sequences

Figure 1: The next value of register *a*3 in a feedback shift register of length 4 is determined by the modulo-2 sum of *a*0 and *a*1.

MLS are generated using maximal linear feedback shift registers. An MLS-generating system with a shift register of length 4 is shown in Fig. 1. It can be expressed using the following recursive relation:

where *n* is the time index, *k* is the bit register position, and + represents modulo-2 addition.

As MLS are periodic and shift registers cycle through every possible binary value (with the exception of the zero vector), registers can be initialized to any state, with the exception of the zero vector.

A **Gold code**, also known as **Gold sequence**, is a type of binary sequence, used in telecommunication (CDMA) and satellite navigation (GPS). Gold codes are named after Robert Gold. Gold codes have bounded small cross-correlations within a set, which is useful when multiple devices are broadcasting in the same range. A set of Gold code sequences consists of $2n - 1$ sequences each one with a period of $2n - 1$.

A set of Gold codes can be generated with the following steps. Pick two maximum length sequences of the same length $2n - 1$ such that their absolute cross-correlation is less than or equal to $2(n + 2) / 2$, where n is the size of the LFSR used to generate the maximum length sequence (Gold '67). The set of the $2n - 1$ exclusive-ors of the two sequences in their various phases (i.e. translated into all relative positions) is a set of Gold codes. The highest absolute cross-correlation in this set of codes is $2(n + 2) / 2 + 1$ for even n and $2(n + 1) / 2 + 1$ for odd n.

The exclusive or of two Gold codes from the same set is another Gold code in some phase.

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

$G(u,v)= \{u,v,u * v, u * Tv, u *T2\, v, U * T(N-1)\, v\}$

Tk 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.

Matlab Program:-

```
clc;
clear all;
m=input('enter the size of shift registers m =');
N=2^m-1;
tt=-N+1:1:N;
x=ones(1,m);
```

```
x1=x;
taps1=input('enter the taps1 =');
taps2=input('enter the taps2 =');
for i=1:N
   y1(i)=x1(m);
   x2=xor12(x1,taps1);
   x1=[x2 x1(1:m)];
end
%disp(y1);
%%%Test-1
T1=sum(y1);
disp('total number of of 1s in maximalseq1 1s');
disp(T1);
if(T1==2^(m-1))
   disp('maximal sequence is exist between the selected taps1');
else
   disp('maximal sequence is NOT exist between the selected taps1');
end
%%%test2
sy1=2*y1-1;
s1=correlation11(sy1,sy1);
figure(1);
plot(tt,s1);
title('Auto correlation of Maximal sequence1');
xlabel('no of samples---->');
ylabel('magnitude---->');
grid;
%%%2nd maximal sequences
for i=1:N
   y2(i)=x(m);
   x2=xor12(x,taps2);
   x=[x2 x(1:m)];
end
%disp(y2);
%%%Test-1
T2=sum(y2);
disp('total number of of 1s in maximalseq2 1s');
disp(T2);
if(T2==2^(m-1))
   disp('maximal sequence is exist between the selected taps2');
else
   disp('maximal sequence is NOT exist between the selected taps2');
end
%%%test2
sy2=2*y2-1;
s2=correlation11(sy2,sy2);
figure(2);
plot(tt,s2);
title('Auto correlation of Maximal sequence2');
xlabel('no of samples---->');
```

```
ylabel('magnitude---->')
grid;
if(T1==T2)
   %module-2 of 2'maximalseq is given as gold sequences(y)
   y=xor(y1,y2);
   g1=2*y-1;
   %%%to check 3-value cross correlation function
   x=correlation11(sy1,sy2);
   figure(3);
   plot(tt,x);
   title('cross correlation of MLS1&MLS2');
   xlabel('no of samples---->');
   ylabel('magnitude---->')
   grid;
   %%% autocorrelation of gold sequences
   xx=correlation11(g1,g1);
   figure(4);
   plot(tt,xx);
   title('auto correlation of gold sequenceses');
   xlabel('no of samples---->');
   ylabel('magnitude---->')
   grid;
   disp('gold sequences is exist between the selected taps');
else
   disp('gold sequences is NOT exist between the selected taps');
end

%%%gold sequence testing by find corss correlation function
c=input('enter the MLsequence1 or MLSequence2 c=');
if(c==1)
   seq=sy1;
else
   seq=sy2;
end
nn=input('enter the No shift we want nn=');
yz=2*y-1;
yz1=circshift(yz,[0,nn]);
Y=correlation11(seq,yz1);
%disp(Y);
figure(5);
plot(tt ,Y);
title('3-value checking of goldsequence ');
xlabel('No of samples ');
ylabel('magnitude');
grid;
```

xor12 sub function:

```
function z=xor12(s,t)
n=length(t);
```

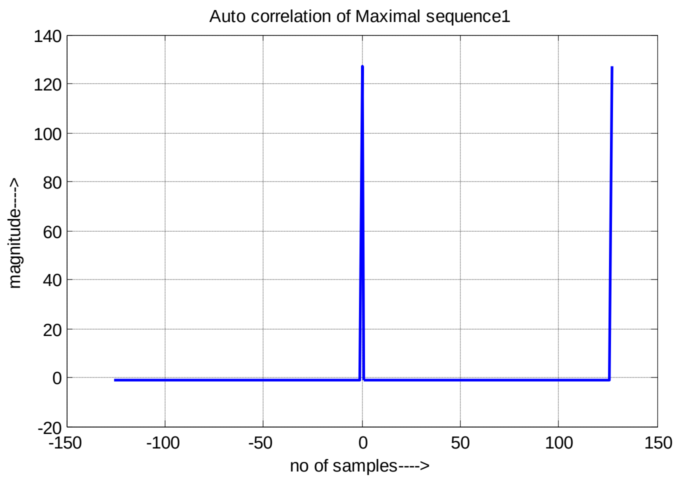
```
c(1)=xor(s(t(1)),s(t(2)));
for i=1:n-2
   c(i+1)=xor(s(t(i+2)),c(i));
end
z=c(length(c));
end
```

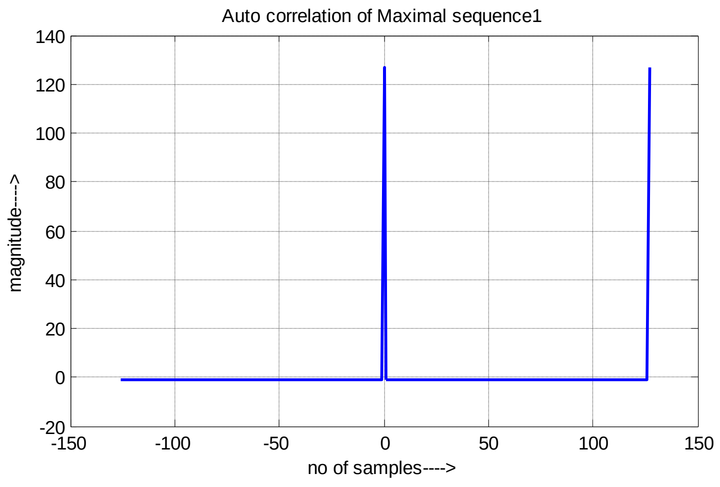
correlation sub function:-

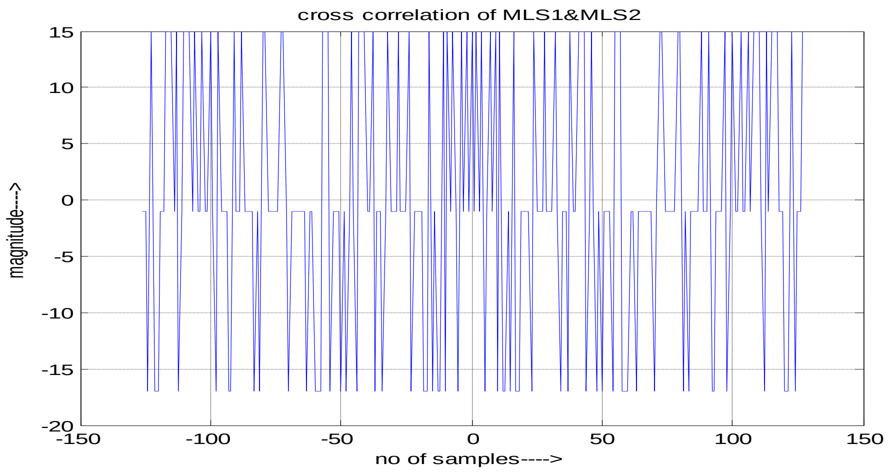
```
function res=correlation11(x,y)
%%%right shift operation
zzz=0;
for i=1:length(x)
   rx=circshift(x,[1,i]);
   z=y.*rx;
   zz=sum(z);
   zzz=[zzz zz];
end
zzz(:,1)=[];

%%%left shift operation
zzz1=0;
for i=1:length(x)
   rx=circshift(x,[1,-i]);
   z=y.*rx;
   zz=sum(z);
   zzz1=[zzz1 zz];
end
zzz1(:,1)=[];
total=[(zzz1) zzz];
res=total;
end
```

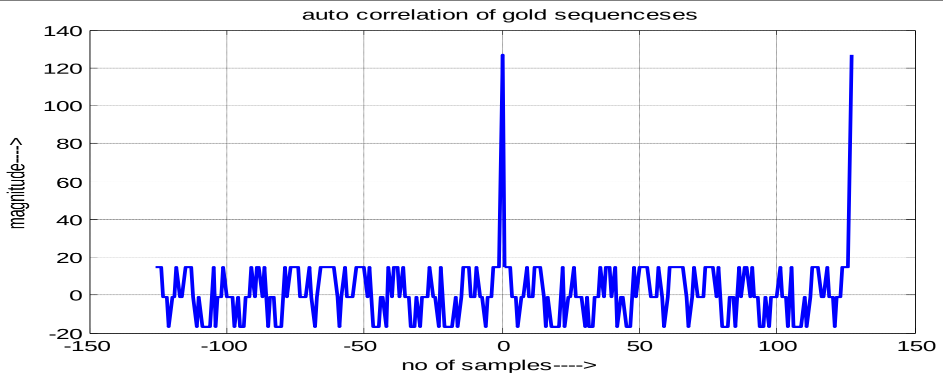
Output Results:-

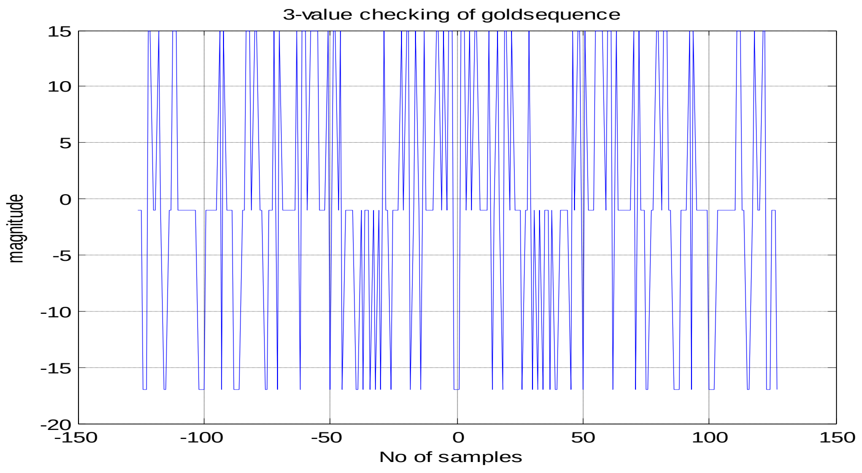
```
enter the size of shift registers m =7
enter the taps1 =[7 1]
enter the taps2 =[7 3]
total number of of 1s in maximalseq1 1s
   64
maximal sequence is exist between the selected taps1
total number of of 1s in maximalseq2 1s
   64
maximal sequence is exist between the selected taps2
gold sequences is exist between the selected taps
enter the MLsequence1 or MLSequence2 c=1
enter the No shift we want nn=23
```


Auto correlation of Maximal sequence1
magnitude---->
no of samples---->
140
120
100
80
60
40
20
0
-20
-150
-100
-50
0
50
100
150


Auto correlation of Maximal sequence1
magnitude---->
no of samples---->
140
120
100
80
60
40
20
0
-20
-150
-100
-50
0
50
100
150


cross correlation of MLS1&MLS2
magnitude---->
no of samples---->
15
10
5
0
-5
-10
-15
-20
-150
-100
-50
0
50
100
150





Result:-

Thus the random data has been generated at a given data rate and the outputs have been verified.

2. Simulation of Rayleigh Fading Channel incorporating speed of the mobile and power delay profile

Aim:- To Simulate the Rayleigh Fading Channel incorporating speed of mobile and power delay profile.

Software Required:- Pc with MATLAB 7.0 or above.

Theory:-

Rayleigh fading is a statistical model for the effect of a propagation environment on a radio signal, such as that used by wireless devices.
Rayleigh fading models assume that the magnitude of a signal that has passed through such a transmission medium (also called a communications channel) will vary randomly, or fade, according to a Rayleigh distribution — the radial component of the sum of two uncorrelated Gaussian random variables.

Rayleigh fading is viewed as a reasonable model for tropospheric and ionospheric signal propagation as well as the effect of heavily built-up urban environments on radio signals. Rayleigh fading is most applicable when there is no dominant propagation along a line of sight between the transmitter and receiver. If there is a dominant line of sight, Rician fading may be more applicable.

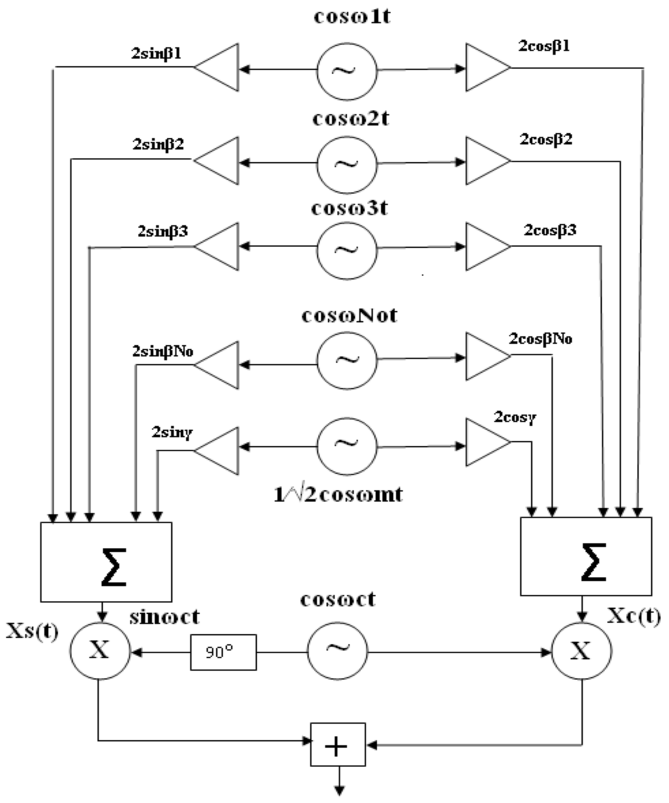
Rayleigh fading is a reasonable model when there are many objects in the environment that scatter the radio signal before it arrives at the receiver. The Central limit Therom holds that, if there is sufficiently much scatter, the channel impulse response will be well-modeled as Gaussian process irrespective of the distribution of the individual components. If there is no dominant component to the scatter, then such a process will have zero mean and phase evenly between 0 and 2π radians. The envelope of the channel response will therefore be Rayleigh.

Calling this random variable *R*, it will have a probability density function.

Where Ω = *E*(*R*2).

Often, the gain and phase elements of a channel's distortion are conveniently represented as a complex number. In this case, Rayleigh fading is exhibited by the assumption that the real and imaginary parts of the response are modelled by independent and identically distributed zero-mean Gaussian processes so that the amplitude of the response is the sum of two such processes.

JAKE'S Model Diagram:-



Jakes' model:-

Jakes popularized a model for Rayleigh fading based on summing sinusoids. Let the scatterers be uniformly distributed around a circle at angles α*n* with *k* rays emerging from each scatterer. The Doppler shift on ray *n* is and, with *M* such scatterers, the Rayleigh fading of the *kth* waveform over time *t* can be modeled as: .

Here, and the and are model parameters with usually set to zero, chosen so that there is no cross-correlation between the real and imaginary parts of *R*(*t*): and used to generate multiple waveforms. If a single-path channel is being modeled, so that there is only one waveform then can be zero. If a multipath, frequency-selective channel is being modeled so that multiple waveforms are needed, Jakes suggests that uncorrelated waveforms are given by: .

In fact, it has been shown that the waveforms are correlated among themselves — they have non-zero cross-correlation — except in special circumstances. The model is also deterministic (it has no random element to it once the parameters are chosen). A modified Jakes' model chooses slightly different spacing for the scatterers and scales their waveforms using Walsh-Hadamard sequences to ensure zero cross-correlation. Setting
And, Results in the following model, usually termed the Dent model or the modified Jakes model:

The weighting functions *Ak*(*n*) are the *k*th Walsh-Hadamard sequence in *n*. Since these have zero cross-correlation by design, this model results in uncorrelated waveforms. The phases can be initialized randomly and have no effect on the correlation properties. The Fast Walsh transform can be used to efficiently generate samples using this model.

The Jakes' model also popularized the Doppler spectrum associated with Rayleigh fading, and, as a result, this Doppler spectrum is often termed Jakes' spectrum.

Program:-

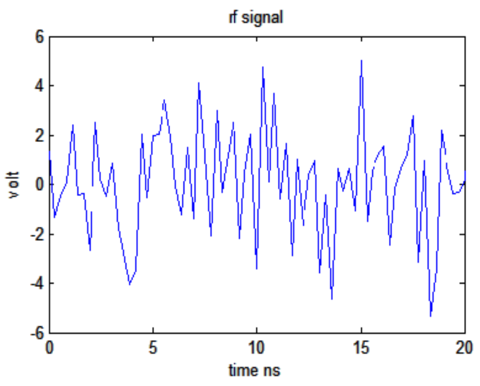
```
% IMPLEMENTATION OF RAYLEIGH FADING CHANNEL
%INCORPORATING SPEED OF MOBILE AND POWER DELAY PROFILE

Clear all;
v=input('MU speed in m/s....5, 10, 15, 20, 25,..>');
numpaths = 10; %number of paths
Fc = 900e6; %carrier frequency
Fs = 4*Fc; %sampling frequency
Ts = 1/Fs; %sampling period
t = [0:Ts:1999*Ts]; %time array
wc = 2*pi*Fc; %radian frequency
%v = 25; %vehicle speed in m/s

   ray = zeros(1,length(t)); %received signal
      a=1;

   for i = 1:numpaths
      wd = 2*pi*v*Fc*cos(unifrnd(0,2*pi))/3e8;
      ray = ray + a.*cos((wc+wd)*t+unifrnd(0,2*pi,1,length(t)));
   end


   [rayi rayq] = demod(ray,Fc,Fs,'qam'); %demodulated signal
```

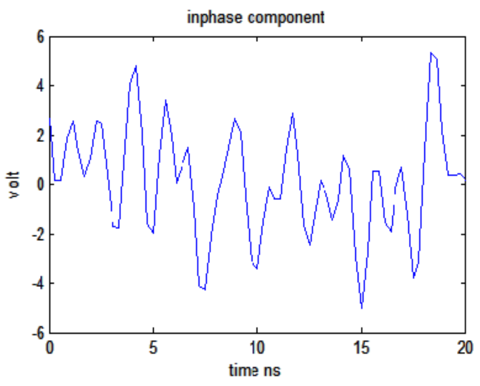
```
    env_ray = sqrt(rayi.^2+rayq.^2); %envelope of received signal
    subplot(2,2,1)
plot(t*1e9,ray)%plots the rf signal
title('rf signal')
xlabel('time ns')
ylabel('volt')
xlim([0 20])
subplot(2,2,2)
plot(t*1e9,rayi)%plots the inphase component
xlabel('time ns')
ylabel('volt')
xlim([0 20])
title('inphase component')
subplot(2,2,3)
plot(t*1e9,rayq)%plots the quadrature component
title('quadrature component')
xlabel('time ns')
ylabel('volt')
xlim([0 20])
subplot(2,2,4)
plot(t*1e9,env_ray)%plots the envelope
title('envelope')
xlabel('time ns')
ylabel('volt')
xlim([0 20])

%computation of the outage probabilities
power_ray=env_ray.^2;
powerdB=10*log10(power_ray);
mean_power=10*log10(mean(env_ray.^2))
MK=length(env_ray)
for k=1:20;
    pow(k)=mean_power-2*k; %threshold power
    kps=pow(k);
    ratio=10^(kps/10)/10^(mean_power/10);
    poutth(k)=1-exp(-ratio);%theoretical outage
    count=0;
    for ku=1:MK;
        power=powerdB(ku);%power in dB
        if power <= kps
            count=count+1;
        else
        end;
    end;
        poutsim(k)=count/MK;%outage probability simulated

end;
figure
plot(pow,poutth,':',pow,poutsim,'--');
xlabel('thrshold power dBm')
```

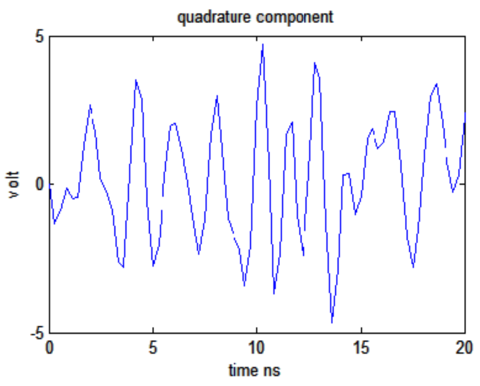
```
ylabel('outage probability')
legend('poutth','poutsim')
```

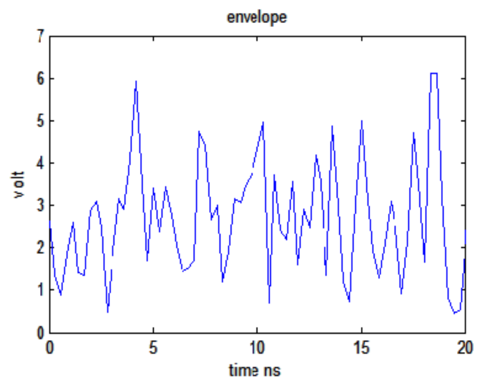
Output results:-

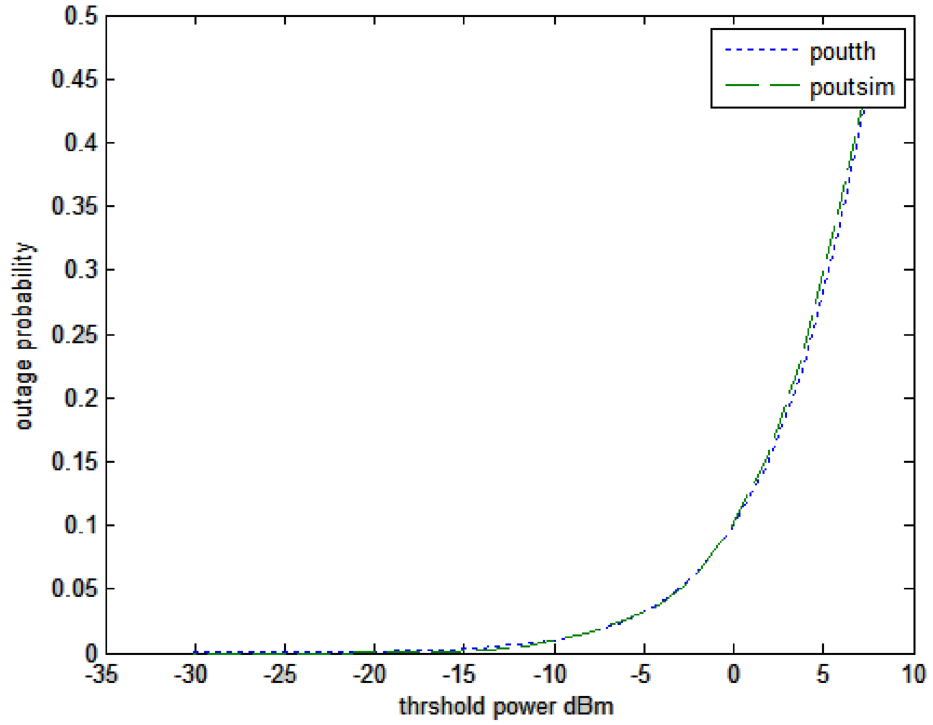
```
MU speed in m/s....5, 10, 15, 20, 25,..>20

mean_power =

            9.8335


    MK =

            2000
```











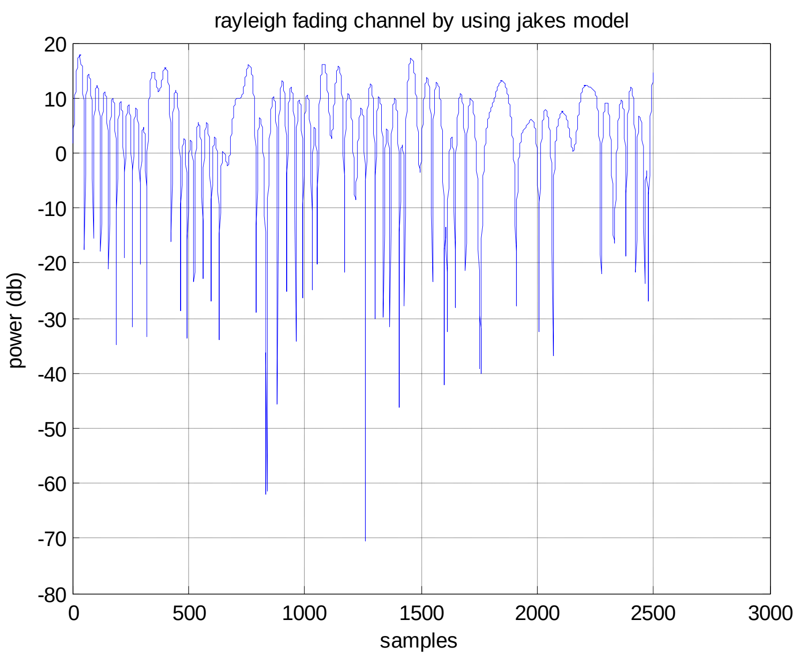
% jakes model %

```
clc;
clear all;
N0=8;
N=2*(2*N0+1);
t=250*10^-3;
t1=0:t/(2500):t;
fm=input('enter the dopplear spread frequency fm=');
wm=2*pi*fm;
gamma=1;
fc=900*10^6;
pw1=0;
for s=0:(t/(2500)):t;
    xca=0;
    xsa=0;
    i=1:N0;
    wn=wm.*cos(2*pi*i/N);
    bt=(pi*i/N0);
        for i=1:N0;
           x=cos(wn(i)*s);
          xca=xca+x.*2*cos(bt(i));
          xsa=xsa+x.*2*sin(bt(i));
        end
    x1=0.707*cos(2*pi*fm*s);
    x1ca=x1*2*cos(gamma);
    x1sa=x1*2*sin(gamma);
    xc=xca+x1ca;
    xs=xsa+x1sa;
```

```
 y=xs.*sin(2*pi*fc*s)+xc.*cos(2*pi*fc*s);
 %disp(y)

 z=y.^2;
 pw=10*log10(z);
 pw1=[pw1 pw];

end

%disp(pw1);
plot(pw1(2:length(pw1)));
title(' rayleigh fading channel by using jakes model');
xlabel('samples');
ylabel('power (db)');
% axis([0 120 -50 20]);
grid;
```

Output results:-

```
 enter the Doppler spread frequency fm=100Hz
```



Result:-

Thus we have simulated the Rayleigh Fading channel incorporating speed of the mobile and power delay profile.

3. SIMULATION OF BPSK SYSTEM OVER AWGN CHANNEL & FINDING ITS PERFORMANCE WITH BER

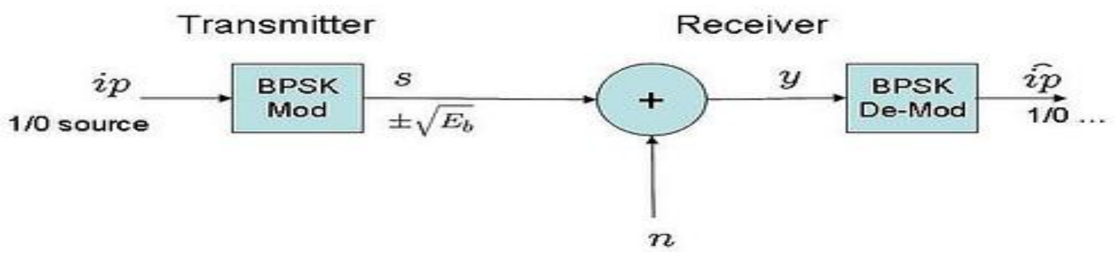
Aim: To implement Simulation of BPSK system over AWGN channel & finding its performance with BER.

Software Required:- Pc with MATLB 7.0 or above

Theory:

BPSK (also sometimes called PRK, phase reversal keying, or 2PSK) is the simplest form of phase shift keying (PSK). It uses two phases which are separated by 180° and so can also be termed 2-PSK. It does not particularly matter exactly where the constellation points are positioned, and in this figure they are shown on the real axis, at 0° and 180°. This modulation is the most robust of all the PSKs since it takes the highest level of noise or distortion to make the demodulator reach an incorrect decision. It is, however, only able to modulate at 1 bit/symbol and so is unsuitable for high data-rate applications.

The basic block diagram of BPSK system is shown in below



The general form for BPSK follows the equation:

$$s_n(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t + \pi(1-n)), n = 0, 1.$$

This yields two phases, 0 and π. In the specific form, binary data is often conveyed with the following signals:

$$s_0(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t + \pi) = -\sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t)$$ for binary "0"

$$s_1(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t)$$ for binary "1"

where f_c is the frequency of the carrier-wave. Hence, the signal-space can be represented by the single basis function.

$$\phi(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_c t)$$

where 1 is represented by $\sqrt{E_b}\phi(t)$ and 0 is represented by $-\sqrt{E_b}\phi(t)$.

The bit error rate (BER) of BPSK in AWGN can be calculated as:[5]

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \text{ or } P_b = \frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right)$$

In this program first we generate some random message signal m and then we convert that message in to binary bits. Based on that binary values we generate s1,s2 signals.

In the next step we generate some additive white Gaussian noise. When the modulated signal passes through the channel the noise will be added to the signal.

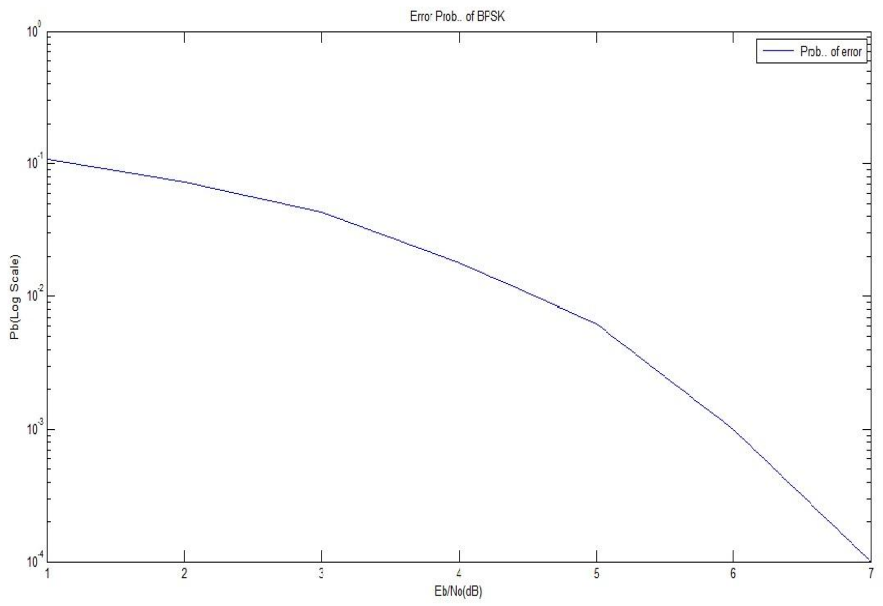
The combination of message signal and noise are received at the receiver. From these corrupted signals we extract the message signal using bpsk demodulation. Then we calculate the error probability which is nothing but the ratio of error bits to the total number of bits . Then we calculate the error complementary function Q and we plot the graphs between probability of error , Q function and energy of bits to the error probability.

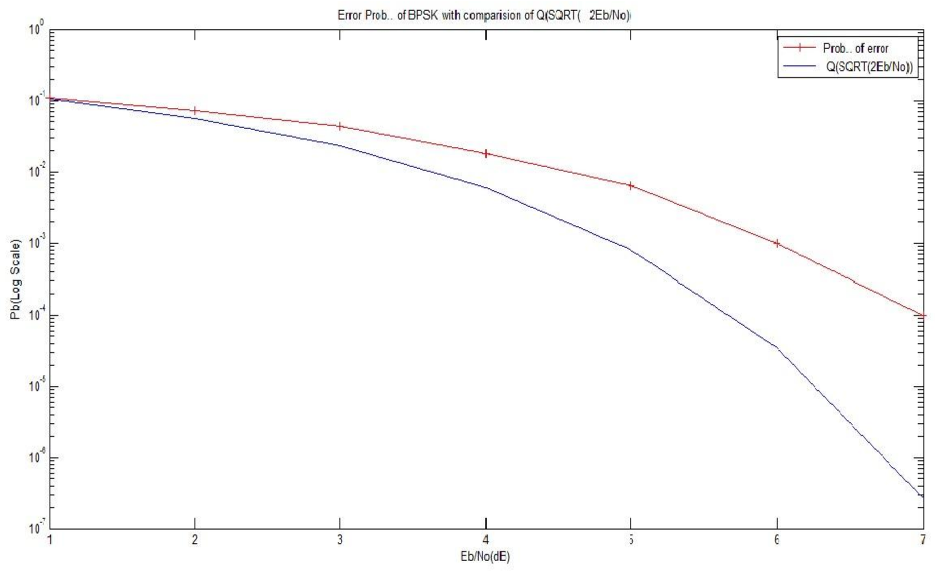
Matlab Program:

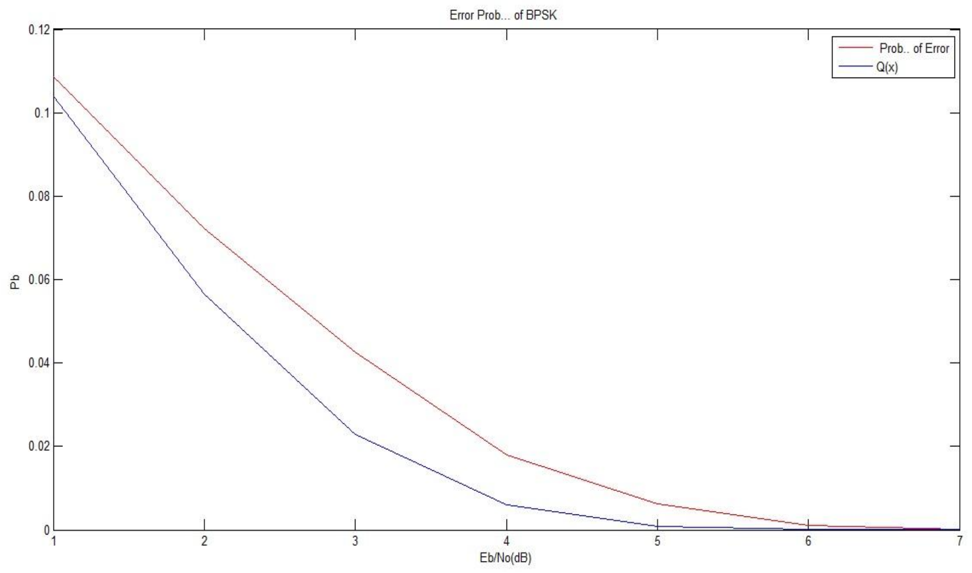
```
clear all;

Eb_No = [1,2,3,4,5,6,7];

No = 1;

Eb = Eb_No * No ;

N = 4 ;

symbol_number = 10000 ;

m = rand(1,symbol_number) ;

[i] = find(m>0.5) ;
```

```
[j] = find(m<=0.5) ;
m(i) = 1 ;
m(j) = 0 ;
for k = 1:length(Eb)
    E = 10^(Eb(k)/10) ;
    E = sqrt(2*E/N) ;
    s1 = ones(1,N)*E ;
    s2 = ones(1,N)*(-E) ;
    for i = 1:symbol_number
        if m(i) == 0
            message((i-1)*(N) + 1 : (i-1)*(N) + N) = s2 ;
        else
            message((i-1)*(N) + 1 : (i-1)*(N) + N) = s1 ;
        end
    end
    received_signal = awgn(message,Eb(k)) ;
    for j = 1:N
        c1(j) = s1(5-j) ;
        c2(j) = s2(5-j) ;
    end
    Z1 = conv(received_signal, c1) ;
    Z2 = conv(received_signal, c2) ;
    m_hat = zeros(1,symbol_number); % added by me
    for i = 1:symbol_number
        m_hat(i) = Z1(i*N - 1) > Z2(i*N - 1) ;
    end
    error = xor(m_hat,m) ;
    error_number = length(find(error>0))

    Pb(k) = error_number/symbol_number ;

end

figure

semilogy(Eb_No,Pb);

title(' Error Prob.. of BPSK ');

xlabel('  Eb/No(dB)  ')

ylabel('  Pb(Log Scale)  ');

legend(' Prob.. of error   ');

Qx = (1/2)*erfc((10.^(Eb/10))/sqrt(2)) ;

figure

semilogy(Eb_No, Pb, 'r+-', Eb_No, Qx) ;

title('  Error Prob.. of BPSK with comparision of Q(SQRT(   2Eb/No))  ');

xlabel('  Eb/No(dB)  ')

ylabel('  Pb(Log Scale)  ');

legend(' Prob.. of error ', '  Q(SQRT(2Eb/No)) ')

figure

plot(Eb_No,Pb,'r',Eb_No,Qx);

title(' Error Prob... of BPSK  ')

xlabel('  Eb/No(dB)  ')

ylabel(' Pb  ');

legend(' Prob.. of Error  ', 'Q(x) ');
```

Output Results:







Result:

Thus Simulation of BPSK system over AWGN channel & finding its performance with BER

4.IMPLEMENT EQUALIZATION AT THE RECEIVER TO REMOVE ISI CAUSED DUE TO LOW CHANNEL BANDWIDTH

Aim:- To implement equalization at the receiver to remove ISI caused due to low channel bandwidth.

Software Required:- Pc with MATLAB 7.0 or above

Theory:

A channel equalizer is a device that reduces the ISI and thus reduces the error rate in the demodulated sequence. Channel distortion is caused due to low channel bandwidth results in Inter symbol Interference(ISI). Many communication channels including telephone channels, some radio channels, may generally characterized as band limited linear filters. As a result of distortion caused by non-ideal channel frequency response characteristic a succession of pulses transmitted through the channel at rates comparable to bandwidth are smearged to a point that they are no longer distinguishable as well defined pulses at receiving terminal instead they overlap causing Inter Symbol Interference.

Linear Equalizers:

The most common type of equalizers used to reduce ISI are adaptive equalizers that update their parameters on periodic basis during the transmission of data so they are capable of tracking slowly time varying channel response.

Adaptive MSE Equalizer:

A communication system transmits binary symbols $\{a_n = 1\}$ through an ISI channel whose characteristic is given by vector x, as number of taps 2K+1.

K=[0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0.0126 0.038 0.088]

Channel output is the sequence

$$y_n = \sum_{k=0}^{10} x_k a_{n-k} + \omega_n$$

where $a_n = 0$, n<0

$\{\omega_n\}$ is white Gaussian noise sequence with variance.

The equalizer is linear adaptive equalizer based on the MSE criterion. Its output sequence is

$$z_n = \sum_{k=-K}^{K} C_k y_{n-k}$$

where $\{C_k\}$ are equalizer coefficients.

To determine the error probability for the equalizer as a function of SNR use the first 200 transmitted symbols to the equalizer with use of LMS algorithm and a step size $\Delta = 0.0045$. Then performing the simulation for 10,000 transmitted symbols for each choice of variance.

For Instance, The gradient vector denoted as g_k for the MSE criterion found by taking the derivatives of MSE w.r.t each of 2K+1 coefficients is

$$g_k = Bc_k - d$$

Adaptive channel equalizers are required for channels whose characteristic change with time. In such a case the ISI varies with time which is tracked in channel response and update its coefficients to reduce the ISI.

The algorithm for adjusting the equalizer tap coefficients may be expressed as

$$\widehat{C}_{k+1} = \widehat{C}_k - \Delta \widehat{g}_k$$

Where $\widehat{g}_k$= estimate of gradient vector

$\widehat{C}_k$= estimate of tap coefficient vector

In case of MSE criterion the gradient vector g_k is also expressed as

$$g_k = -E(e_k y^*_k)$$

An estimate $\hat{g}_k$ of gradient vector at the kth iteration

$$\hat{g}_k = -(e_k y^*_k)$$

Where ; e_k = difference between desired output from equalizer at the kth time instant and the actual output z(kT) and y_k denote the column vector of 2K+1 received signal values contained in equalizer at time instant K.

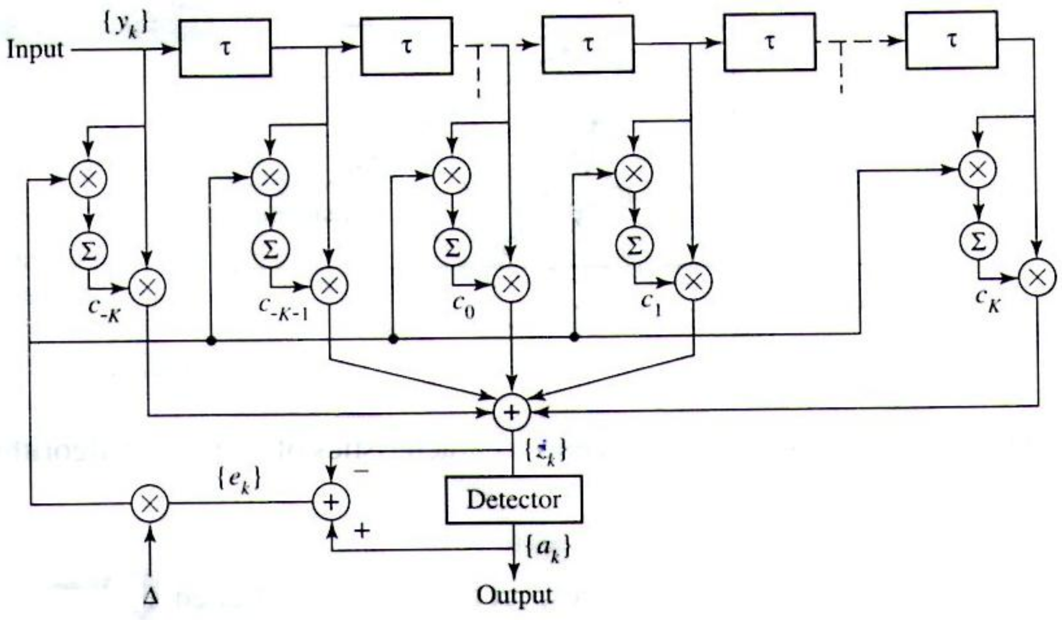
Error signal $e_k = a_k - z_k$

Where

$$z_k = \sum_{n=-K}^{K} C_n y(mT - n\tau)$$

Adaptive algorithm for optimizing the tap coefficient based on MSE criterion has the block diagram that adapts its tap coefficients according to

$$\hat{C}_{k+1} = \hat{C}_k + \Delta e_k y^*_k$$



Linear adaptive equalizer based on the MSE criterion

The difference between desired output and actual output z_k from equalizer used to form the error signal e_k.The error is scaled by step size parameter Δ and scaled error signal multiplies the received signal values $\{y(kT - n\tau)\}$ at (2K+1) taps. The products $\Delta e_k y * (kT - n\tau)$ at the (2K+1) taps are then added to previous values of taps coefficients to obtain the updated tap coefficients. The computation is repeated at each new symbol is received. Thus the equalizer coefficients are updated at the symbol rate.

Initially the adaptive equalizer is trained by the transmission if known pseudo-random sequence $\{a_m\}$ over the channel. At the demodulator the equalizer employs the known signal to adjust its coefficients upon initial adjustment the adaptive equalizers switches from a training mode to a decision directed mode in which case the decisions at the output of the detector are sufficiently reliable so that the error signal formed by computing the difference between the detector output and the equalizer output.

$$e_k = \hat{a}_k - z_k$$

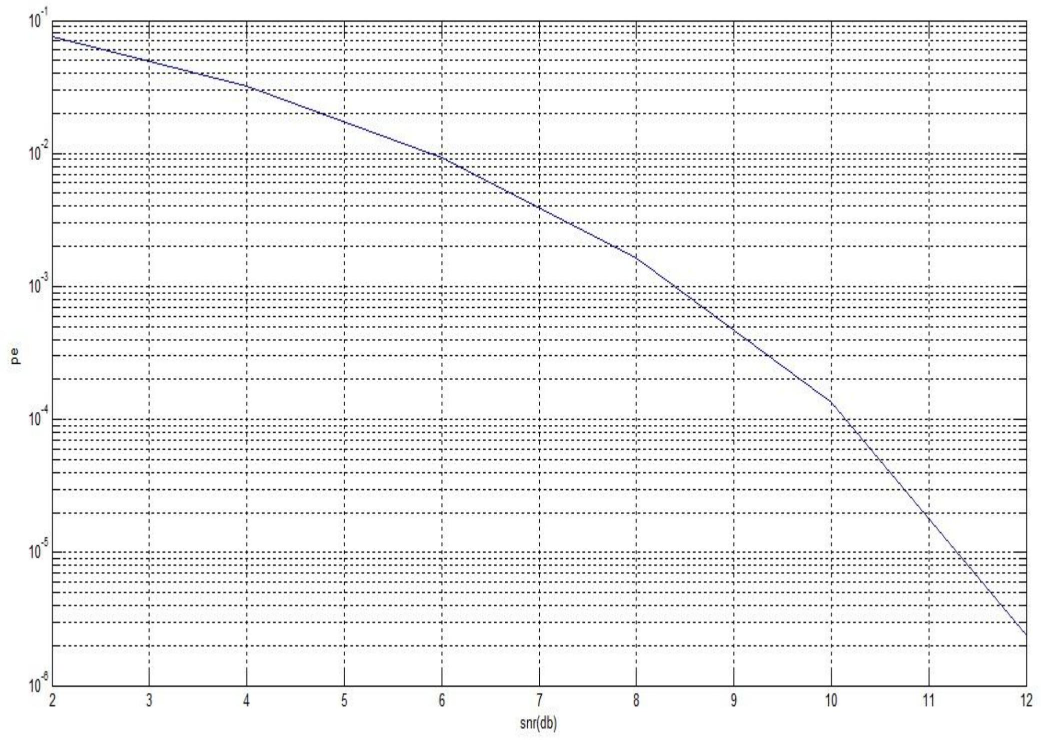
Where $\hat{a}_k$ = Output of detector.

Matlab Program:

```
%%% echo on
clear all;
close all;
N = 20000;
Nt = [1000000 1000000 1000000 1000000 1000000 5000000 10000000];
delta = 0.0045;
K = 5;
actual_isi = [0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038
0.088];
snr_db = 2:2:14;
snr = 10.^(snr_db/10);
l_snr = size(snr,2);
var = 1/2./snr;
sigma = sqrt(var);
pe = zeros(1,l_snr);
for idx = 1:l_snr
   % [A] the training sequence
   training_s = ones(1,N);
   for i = 1:N
      if(rand<0.5)
         training_s(i) = -1 ;
      end
```

```
        end
        % The channel output
        y = filter(actual_isi,1,training_s);
        noise = zeros(1,N);
        for i = 1:2:N
            noise(i) = random('Normal',0,sigma(idx));
            noise(i+1) = noise(i);
        end
        y = y +noise;
        % The equalization part follows
        estimated_c = [0 0 0 0 0 1 0 0 0 0 0]; % initial estimate of isi
        for k = 1:N-2*K
            y_k = y(k:k+2*K);
            z_k = estimated_c*y_k';
            e_k = training_s(k) - z_k;
            estimated_c = estimated_c + delta*e_k*y_k;
        end
        % (B) the transmitted information sequence
        info = ones(1,Nt(idx));
        for i = 1:Nt(idx)
            if(rand<0.5)
                info(i) = -1;
            end
        end
        % The channeled output
        y = filter(actual_isi,1,info);
        noise = sigma(idx)*randn(1,Nt(idx));
        y = y + noise;
        % The equalization part
        count = 0;
        err_count = 0;
        z_k_vec = ones(1,Nt(idx) - 2*K);
        for k = 1:Nt(idx)-2*K;
            y_k = y(k:k+2*K);
            z_k = estimated_c*y_k';
            if z_k<0
                z_k_vec(k) = -1 ;
            end
            err_count = err_count + 0.5*abs(info(k)-z_k_vec(k));
        end
        pe(idx) = err_count/length(z_k_vec);
        clear y; clear noise
    end
    % Plot the results
    semilogy(snr_db,pe)
    grid
    ylabel('  pe   ')
    xlabel('  snr(db)  ')
```

Output Results:



Result:

Implementing equalization at the receiver to remove ISI caused due to low channel bandwidth is simulated and results are verified.

5. IMPLEMENTATION OF RAKE RECEIVER AND FINDING ITS PERFORMANCE THROUGH BER CURVE.

Aim:- To implement Rake Receiver and to find its performance through BER curve.

Software Required:- Pc with MATLAB 7.0 or above.

Theory:-

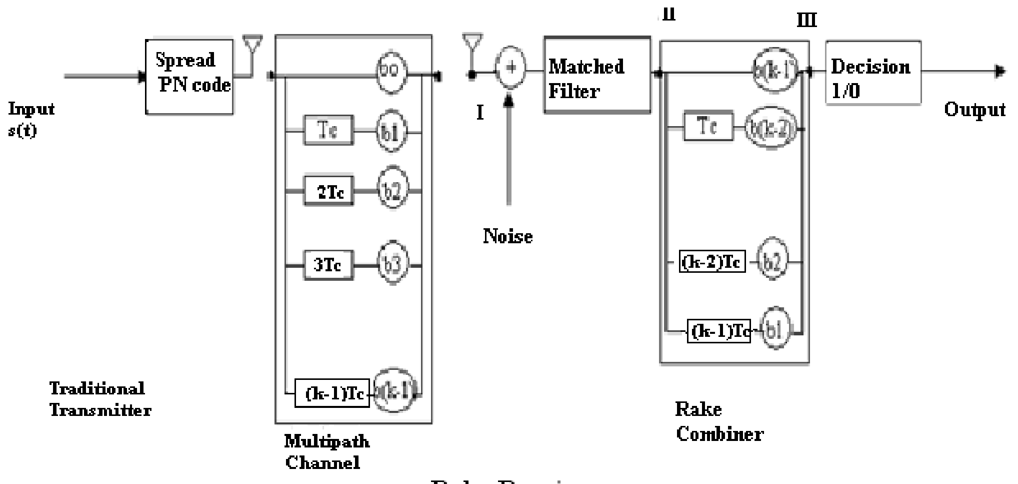
A rake receiver is a radio receiver designed to counter the effects of multipath fading. It does this by using several "sub-receivers" called *fingers*, that is, several correlators each assigned to a different multipath component. Each finger independently decodes a single multipath component; at a later stage the contribution of all fingers are combined in order to make the most use of the different transmission characteristics of each transmission path. This could very well result in higher signal-to-noise ratio (or Eb/N0) in a multipath environment than in a "clean" environment.

The multipath channel through which a radio wave transmits can be viewed as transmitting the original (line of sight) wave plus a number of multipath components. Multipath components are delayed copies of the original transmitted wave traveling through a different echo path, each with a different magnitude and time-of-arrival at the receiver. Since each component contains the original information, if the magnitude and time-of-arrival (phase) of each component is computed at the receiver (through a process called channel estimation), then all the components can be added coherently to improve the information reliability.

The rake receiver is so named because it reminds the function of a garden rake, each finger collecting symbol energy similarly to how tines on a rake collect leaves.

Rake receivers are common in a wide variety of CDMA and W-CDMA radio devices such as mobile phones and wireless LAN equipment.

Rake Receiver Diagram:-



Rake Receiver

Rake receiver:-

If, in a mobile radio channel reflected waves arrive with small relative time delays, self interference occurs. Direct Sequence (DS) Spread Spectrum is often claimed to have particular properties that makes it less vulnerable to multipath reception. In particular, the rake receiver architecture allows an optimal combining of energy received over paths with different. It avoids wave cancellation (fades) if delayed paths arrive with phase differences and appropriately weighs signals coming in with different signal-to-noise ratios.

The rake receiver consists of multiple correlators, in which the receive signal is multiplied by time-shifted versions of a locally generated code sequence. The intention is to separate signals such that each finger only sees signals coming in over a single (resolvable) path. The spreading code is chosen to have a very small autocorrelation value for any nonzero time offset. This avoids crosstalk between fingers. In practice, the situation is less ideal. It is not the full periodic autocorrelation that determines the crosstalk between signals in different fingers, but rather two *partial* correlations, with contributions from two consecutive bits or symbols. It has been attempted to find sequences that have satisfactory partial correlation values, but the crosstalk due to partial (non-periodic) correlations remains substantially more difficult to reduce than the effects of periodic correlations.

The rake receiver is designed to optimally detect a DS-CDMA signal transmitted over a dispersive multipath channel. It is an extension of the concept of the matched filter.

In a multipath channel, delayed reflections interfere with the direct signal. However, a DS-CDMA signal suffering from multipath dispersion can be detected by a rake receiver. This receiver optimally combines signals received over multiple paths.

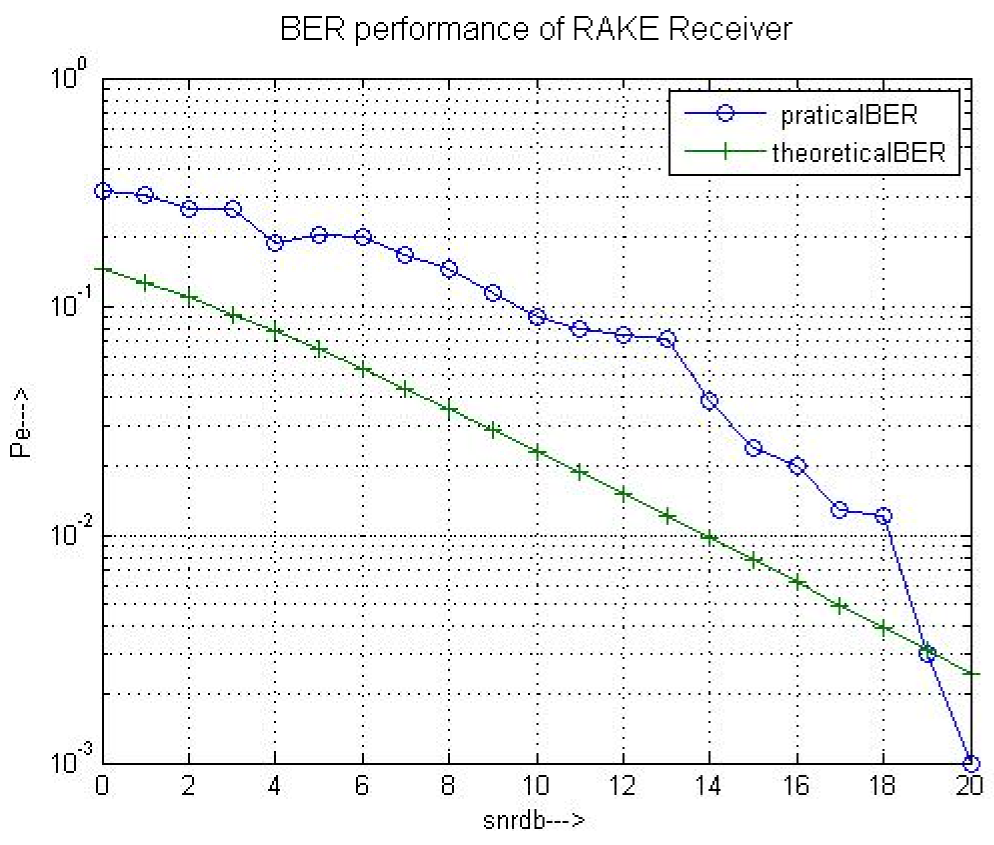
Like a garden rake, the rake receiver gathers the energy received over the various delayed propagation paths. According to the maximum ratio combining principle, the SNR at the output is the sum of the SNRs in the individual branches, provided that

- we assume that only AWGN is present (no interference)
- codes with a time offset are truly orthogonal

Matlab Program:

```
clc;
clear all;
e=1;
snrdb=0:1:20;
n=1000;
L=length(snrdb);N=7;code=[1 1 1 -1 -1 1 -1];G=7;chipsperslot=7;
bfad=randn(1,10);%[0.1 -0.2 0.3];
bfadt=conj(fliplr(bfad));%[0.3 -0.2 0.1];

t=0:ts:((n*tb/2)-ts);
sp=10;
for ii=1:L
    pe1=0;
    snr=10^((ii-1)/10);
    sigmao=sqrt(sum(bfad.*bfadt)*G/snr);
        %%%data generater
      d=rand(1,n)>.5;
      d1=2*d-1;
      %%%%SPREADING THE DATA
     % out=conv(d1,code);
       out=[];
      for i=1:n
        out=[out d1(i)*code];
      end


        K=length(bfad);
        mfdata=conv(out,bfad);
%        %% AWGN CHANNEL
        w=sigmao.*randn(1,length(mfdata));
        rx=w+mfdata;
%        %%AT RECEIVER SIDE
%%%matched filter
mfdata1=conv(rx,conj(fliplr(bfad)));
%delay=length(code)+length(bfad);
getdata=mfdata1(3:length(mfdata1)-2);
%getdata=mfdata1(delay:delay+chipsperslot);
%getdata=getdata(K:length(code):length(getdata));
 rdata=getdata;
for i=1:n
      rdata(i)= getdata(((K)+(i-1)*N));
end
rd=rdata>0;
        for i=1:n
            if(rd(i)==d(i))
            else
                pe1=pe1+1;
            end
        end

    pe2(ii)=pe1;
    %%probability error in simulated system
    BER(ii)=pe2(ii)/(n);
    %%probability error in theoretical function
    geta=G*(sum(bfad.*bfadt)/sigmao^2);
    qf4(ii)=0.5*(1-sqrt(geta/(1+geta)));
end
disp(pe2);
disp(qf4);
disp('bit error rate is');
disp(BER);
semilogy(snrdb,BER,'-o',snrdb,qf4,'-+');
hold all;
%axis([0 14 10e-7 10e-1]);
legend(' praticalBER','theoreticalBER');
title('BER performance of RAKE Receiver')
xlabel('snrdb--->');
ylabel('Pe--->');
grid;
```

Output Result:-



Result:-

Thus we have implemented the Rake Receiver and its performance has been calculated through BER curve.

6. IMPLEMENTATION OF LMS ALGORITHM TO ESTIMATE THE ORIGINAL DATA WHEN IT IS CORRUPTED BY NOISE AND CHANNEL

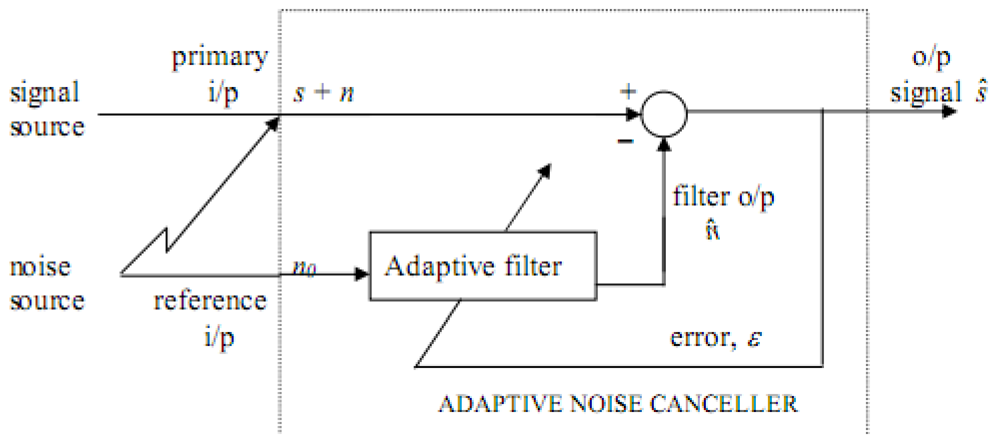
Aim:- To implement the LMS algorithm to estimate the original data when it is corrupted by noise and channel.

Software Required:- Pc with MATLAB 7.0 or above.

Theory:-

Adaptive Noise Cancellation (ANC) and its Applications. Adaptive noise Cancellation is an alternative technique of estimating signals corrupted by additive noise or interference. Its advantage lies in that, with no apriori estimates of signal or noise, levels of noise rejection are attainable that would be difficult or impossible to achieve by other signal processing methods of removing noise.

Its cost, inevitably, is that it needs two inputs - a primary input containing the corrupted signal and a reference input containing noise correlated in some unknown way with the primary noise. The reference input is adaptively filtered and subtracted from the primary input to obtain the signal estimate. Adaptive filtering before subtraction allows the treatment of inputs that are deterministic or stochastic, stationary or time-variable.



For the application of ANC, it is desired to choose an algorithm which is computationally very fast. Taking this into consideration, LMS algorithm became an obvious choice over RLS.

For the LMS algorithm, the update equation of the filter weights becomes

$$\widehat{\mathbf{w}}(n+1) = \widehat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n)$$

where $e^*(n) = d^*(n) - \mathbf{u}^H(n)\widehat{\mathbf{w}}(n)$.

Compare with the corresponding equation for the SD:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu E\{\mathbf{u}(n)e^*(n)\}$$

where $e^*(n) = d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n)$.

The difference is thus that $E\{\mathbf{u}(n)e^*(n)\}$ is estimated as $\mathbf{u}(n)e^*(n)$. This leads to *gradient noise*.

In the given program we designed the LMS algorithm with order K = 5 so we require total 11 tap weights and actual_isi, sigma and delta are taken as fixed. With these specifications first we generate the information signal info then with the help of gngauss function some random noise will be generated and we add that noise to the info then we get corrupted signal which is called as primary signal.

Matlab Program:

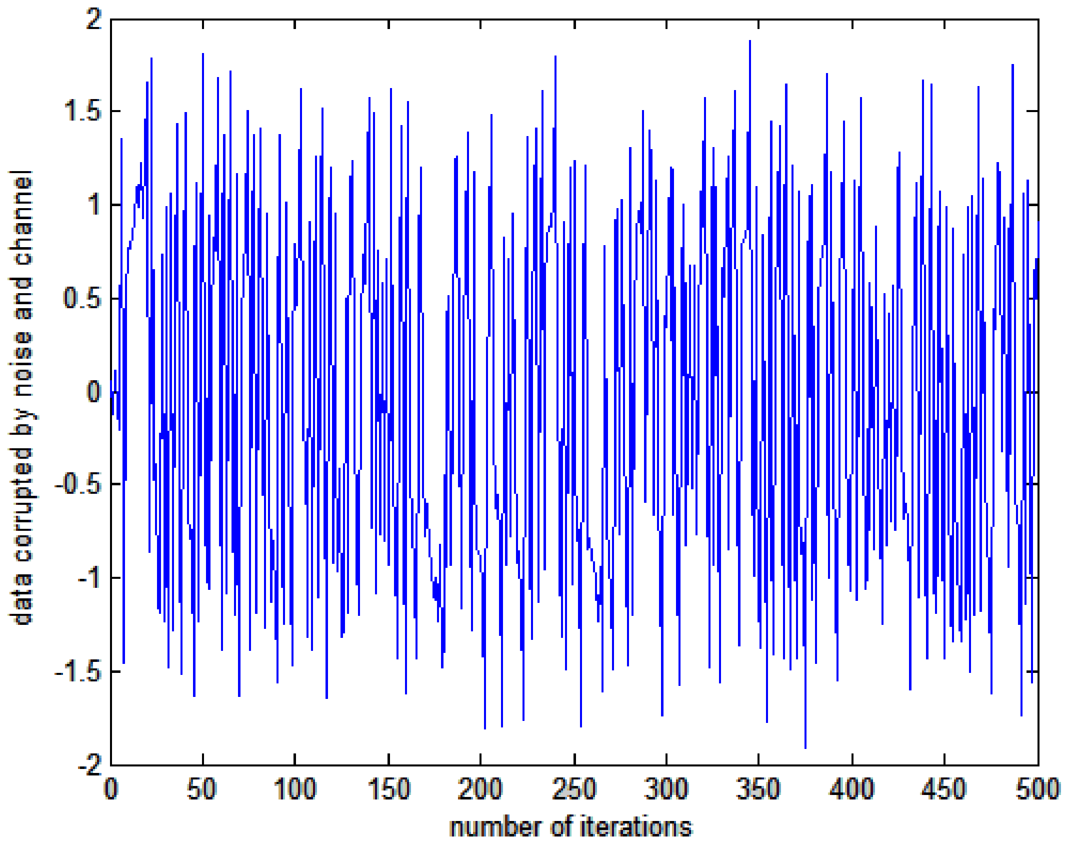
```
echo on

N = 500 ;
K = 5 ;
actual_isi = [0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088] ;
sigma = 0.01 ;
delta = 0.115 ;
Num_of_realizations = 1000 ;
mse_av = zeros(1,N-2*K) ;
for j = 1:Num_of_realizations
   % the information sequence
   for i = 1:N
      if(rand<0.5)
         info(i) = -1 ;
      else
         info(i) = 1 ;
      end
      echo off ;
```

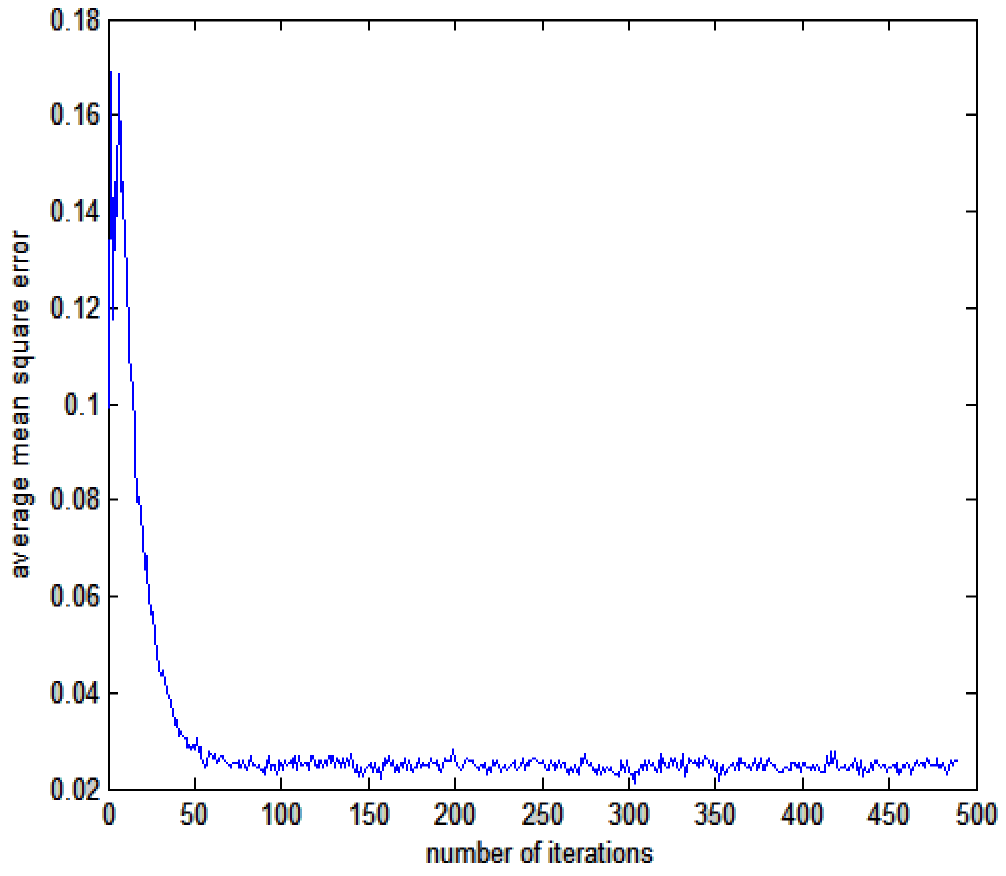
```
    end
    if(j == 1) ;
       echo on ;
    end
    % the channel output
    y = filter(actual_isi,1,info);
    for i = 1:2:N, [noise(i) noise(i+1)] = gngauss(sigma) ; end;
    y = y+noise ;
    % Now the equalization part follows
    estimated_c = [ 0 0 0 0 0 1 0 0 0 0 0] ;  % initial estimate of ISI
    for k = 1:N-2*K,
       y_k = y(k:k+2*K) ;
       z_k = estimated_c*y_k.' ;
       e_k = info(k) - z_k ;
       estimated_c = estimated_c + delta*e_k*y_k ;
       mse(k) = e_k^2 ;
       echo off ;
    end ;
    if(j == 1) ; echo on ; end
    mse_av = mse_av + mse ;
    echo off ;
end ;
echo on ;
mse_av = mse_av/Num_of_realizations ;   % mean-squared error versus
iteraions
% plotting commands follow.
figure
plot(y)
ylabel(' data corrupted by noise and channel  ')
xlabel('  number of iterations  ')
figure
plot(mse_av)
ylabel('  average mean square error  ')
xlabel('  number of iterations  ')
```

gngauss sub function:

```
function [gsrv1, gsrv2] = gngauss(m,sgma)

% [gsrv1, gsrv2] = gngauss(m,sgma)
% [gsrv1, gsrv2] = gngauss(sgma)
% [gsrv1, gsrv2] = gngauss

%  GNGAUSS  generates two independent Gaussian random variables with
%          mean m and standard deviation sgma. If one of the input
%          arguments is missing, it takes the mean as 0.
%          If neither the mean nor the variance is given, it generates two
%          standard Gaussian random variables.

if nargin == 0,
    m = 0 ; sgma = 1 ;
elseif nargin ==1,
    sgma = m; m =0 ;
end ;
u = rand ;    % a uniform random variable in (0,1)
z = sgma*(sqrt(2*log(1/(1-u)))) ;  % a Rayleigh distributed random variable
u = rand ;    %   another uniform random variable in (0,1)
gsrv1 = m + z*cos(2*pi*u) ;
gsrv2 = m + z*sin(2*pi*u) ;
```

Output Result:-





Result:-

Thus we have implemented the LMS algorithm to estimate the original data when it is corrupted by noise and channel.

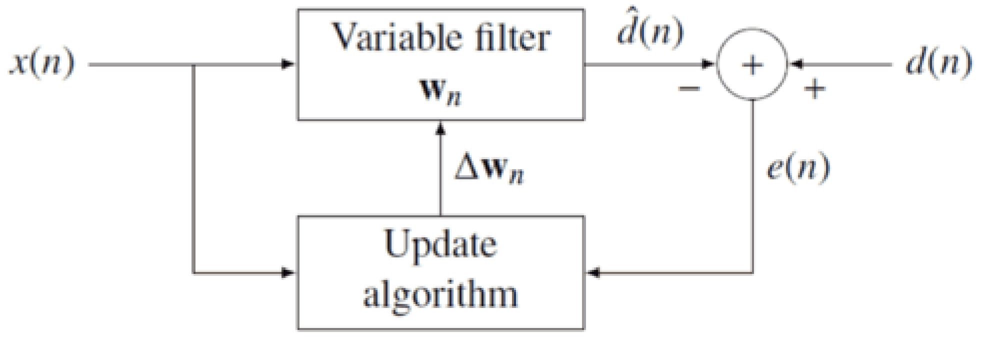
7. IMPLEMENTATION OF R.L.S ALGORITHM TO ESTIMATE THE ORIGINAL DATA WHEN IT IS CORRUPTED BY NOISE AND CHANNEL

Aim: To implement R.L.S algorithm to estimate the original data when it is corrupted by noise & channel.

Software Required: Pc with Matlab 7.0 or above

THEORY: Another popular noise cancellation adaptive filter involves using the recursive least squares method (RLS). Unlike the LMS method, the RLS method finds the filter coefficients that minimize a weighted linear least squares function relating to the input signals instead of trying to reduce the mean square error. It is deterministic (i.e. no randomness is involved of future states of the system).The RLS method is faster than the LMS method at the expense of more complex computations

The idea behind RLS filters is to minimize a cost function C by appropriately selecting the filter coefficients $\mathbf{w}_n$, updating the filter as new data arrives. The error signal $e(n)$ and desired signal $d(n)$ are defined in the negative feedback diagram below:



parameters:

p = filter order

$\lambda =$ forgetting factor

$\delta =$ value to initialize p(0)

n = length of the desired signal

Initialization:

w = [w(0) = 0 ,w(1) = 0 ,.......,w(p) = 0]

computations:

$For\ n = 0, 1, 2, \ldots$

$\boldsymbol{x}(n) = [x(n), x(n-1), \ldots, x(n-p+1)]^T$

$\tilde{s}(n) = \boldsymbol{x}(n) \cdot \boldsymbol{w}(n-1)$

$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \tilde{s}(n)$

$\boldsymbol{g}(n) = \boldsymbol{P}(n-1) \cdot \boldsymbol{x}(n)\{\lambda + \boldsymbol{x}^T(n) \cdot \boldsymbol{P}(n-1) \cdot \boldsymbol{x}(n)\}^{-1}$

$\boldsymbol{P}(n) = \lambda^{-1} \cdot \boldsymbol{P}(n-1) - \boldsymbol{g}(n) \cdot \boldsymbol{x}^T(n) \cdot \lambda^{-1} \cdot \boldsymbol{P}(n-1)$

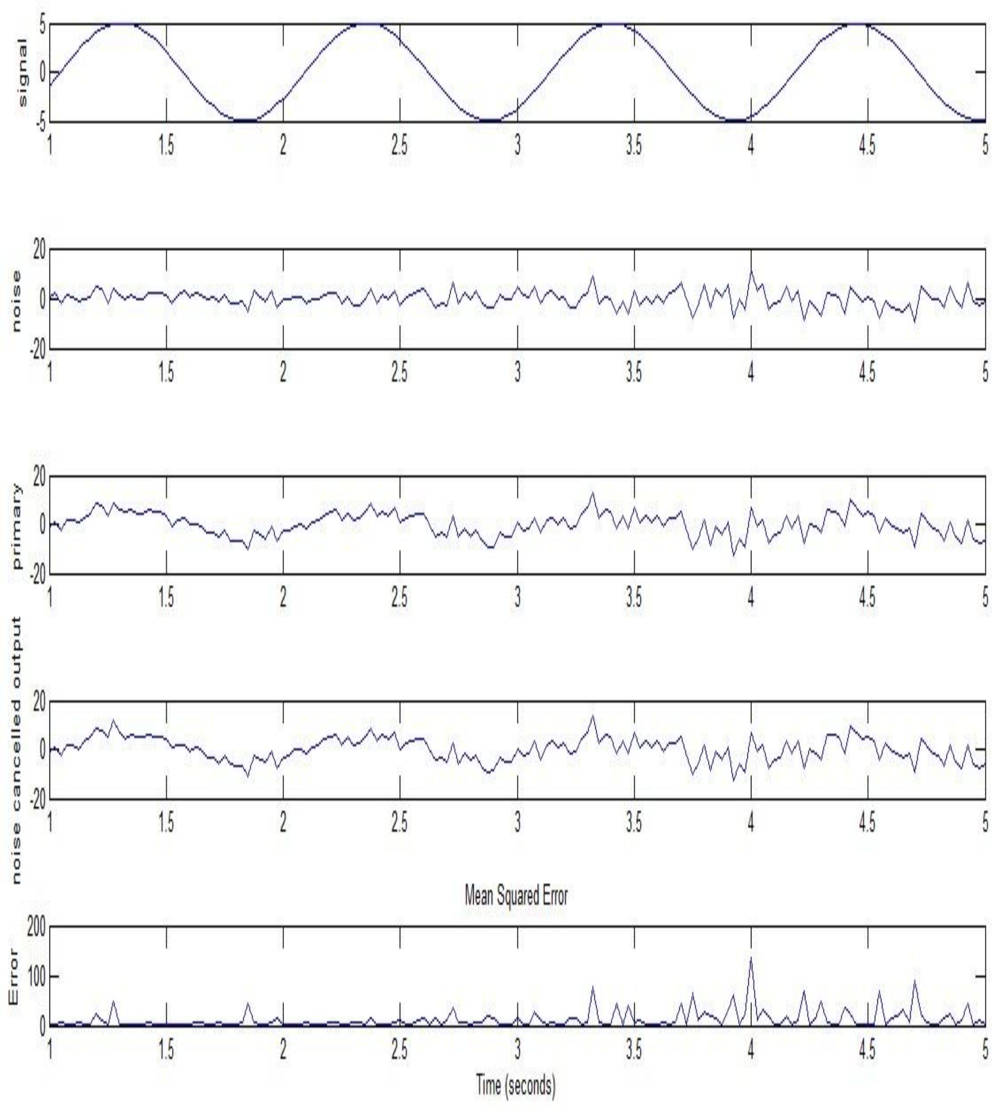
$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{e}(n) \cdot \boldsymbol{g}(n)$

Matlab Program:

```
% Recursive Least Squares
% Importing Data
% Vectors must have same length and sampling frequency
% signal = source input
% noise = noise to add to source input
% ref = reference signal (e.g. delayed noise)
% lambda = forgetting factor,
% M = filter order
% delta = initial value for P; use a value in the 10^-3 range

% Output arguments:
% xi = output
% w = final filter coefficients

% Initialization
clear all;
close all;
clc;
lambda = 1;
M =5;
delta = .0005;
w=zeros(M,1); k=zeros(M,1);
t=1:.025:5;
signal=5*sin(2*3.*t);
ref=5*sin(2*50*3.*t+3/20);
noise=random('normal',0,t);
primary = signal + noise;
subplot(5,1,1);
plot(t,signal);
ylabel('signal');
subplot(5,1,2);
```

```
plot(t,noise);
ylabel('noise');

subplot(5,1,3);
plot(t,primary);
ylabel('primary');
% eye(M) gives an MxM identity matrix
% dividing by delta replaces the 1's in the identity matrix with delta^-1
P=eye(M)/delta;

% Input Signal Length
N=length(ref);
% Set output as desired signal
xi=primary;

% Loop, RLS
for n=M:N
uvec=ref(n:-1:n-M+1);
k=lambda^(-1)*P*uvec'/(1+lambda^(-1)*uvec*P*uvec');
% xi is output
xi(n)=primary(n)-w'*uvec';
% Recursive equation to minimize cost function (difference between the
% desired and input signals)
w=w+k*conj(xi(n));
P=lambda^(-1)*P-lambda^(-1)*k*uvec*P;

end

% Calculate Mean Squared Error
MSE = xi - signal(1:length(xi));
MSE = MSE.^2;
subplot(5,1,4);
plot(t,xi);
ylabel('noise cancelled output');

% Time to use for plotting in time domain
% 0-5 seconds with 1/fs increments

subplot(5,1,5);
plot(t(1:length(MSE)), MSE)
xlabel('Time (seconds)')
ylabel('Error')
title('Mean Squared Error')
```

OUTPUT RESULTS:



Result:

Thus R.L.S algorithm to estimate the original data when it is corrupted by noise & channel is implemented by using matlab software.