# Scilab Manual for Mobile Communication by Prof Hetal Shah Others Dharmsinh Desai University[1]

Solutions provided by
Prof Hetal Shah
Others
Dharmsinh Desai

February 12, 2026

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

# Contents

List of Scilab Solutions 3

1 Digital Modulation Functions: ASK, FSK, PSK generation. 4

2 Constellation diagram and Error Rate performance of different modulation techniques with AWGN channel. 7

3 Effect of various channel on transmitted data using different modulation techniques. 13

4 Trunking Theory for Probability of blocking(Erlang B) and probability of delay( Erlang C). 23

5 Walsh Code generation 27

6 PN sequence generation. 31

7 Equalization. 35

8 Channel Coding using Linear Block Code 38

9 Transmit and receive diversity 43

10 Speech coding 47

# List of Experiments


| | | |
|---|---|---|
| Solution 1.1 | 1 | 4 |
| Solution 2.1 | Sigal space diagram of different modulation | 7 |
| Solution 2.2 | BER of BPSK and QPSK over AWGN Channel | 9 |
| Solution 3.1 | BER BPSK Rayleigh fading channel | 13 |
| Solution 3.2 | BER QPSK Rayleigh channel | 16 |
| Solution 3.3 | 1 | 20 |
| Solution 4.1 | Traffic calculation inErlang B and Erlang C system | 23 |
| Solution 5.1 | Walsh code generation and spreading and despreading using Walsh code | 27 |
| Solution 6.1 | 3 bit PN sequence generation and spreading and despreading using PN sequence and shifted PN sequence | 31 |
| Solution 7.1 | Adaptive equalization using LMS filter | 35 |
| Solution 8.1 | Linear Block Coding over AWGN channel | 38 |
| Solution 9.1 | Selection Diversity over AWGN channel | 43 |
| Solution 9.2 | Maximal Ratio Combining over AWGN and Rayleigh fading Channel | 44 |
| Solution 10.1 | speech coding and Decoding using LPC | 47 |

# Experiment: 1

# Digital Modulation Functions: ASK, FSK, PSK generation.

**Scilab code Solution 1.1** 1

```
//Amplitude Shift Keying, Frequency Shift Keying And Phase Shift keying waveform generation
clc;
clear;
xdel(winsid());
sym=10;//no. of symbols
g=[1 1 0 1 0 0 1 1 1 0 ]//binary data
f1=1;f2=2;//frequencies of carrier
t=0:2*%pi/99:2*%pi;//range of time
//ASK
cp=[];bit=[];mod_ask=[];mod_fsk=[];mod_psk=[];cp1=[];cp2=[];
for n=1:length(g);//ASK modulation  // Zeros and ones are inserted for proper plot of message signal
    if g(n)==0;
        die=zeros(1,100);
    else g(n)==1;
        die=ones(1,100);
```

```
    end
    c_ask=sin(f1*t);
    cp=[cp die];
    mod_ask=[mod_ask c_ask];
    end
ask=cp.*mod_ask;//ASK modulated signal

//FSK
for n=1:length(g);
    if g(n)==0;
        die=ones(1,100);
        c_fsk=sin(f1*t);
    else g(n)==1;
        die=ones(1,100);
        c_fsk=sin(f2*t);
    end
    cp1=[cp1 die];
    mod_fsk=[mod_fsk c_fsk];
end
fsk=cp1.*mod_fsk;//FSK molated signal

//PSK
for n=1:length(g);
    if g(n)==0;
        die=ones(1,100);
        c_psk=sin(f1*t);
    else g(n)==1;
        die=ones(1,100);
        c_psk=-sin(f1*t);
    end
    cp2=[cp2 die];
    mod_psk=[mod_psk c_psk];
end
psk=cp2.*mod_psk;//PSK modulated signal
subplot(4,1,1);plot(cp,'LineWidth',1.5);//plot
   binary signal
xgrid;
title('Binary Signal');//title
```

```
mtlb_axis([0 100*length(g) -2.5 2.5]); //axis range
subplot(4,1,2);plot(ask,'LineWidth',1.5);//plot of ASK modulated signal
xgrid;
title('ASK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//axis range
subplot(4,1,3);plot(fsk,'LineWidth',1.5);//plot of FSK modulated signal
xgrid;
title('FSK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//axis range
subplot(4,1,4);plot(psk,'LineWidth',1.5);//plot of PSK modulated signal
xgrid;
title('PSK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//range of axis
//Result: This experiment results plots of binary data, ASK modulation,FSK modulation and PSK modulation
```

# Experiment: 2

# Constellation diagram and Error Rate performance of different modulation techniques with AWGN channel.

**Scilab code Solution 2.1** Sigal space diagram of different modulation

```
//Constellation diagram of BPSK and QPSK modulation and BPSK and QPSK modulation over AWGN channel
clc;
clear;
xdel(winsid());
sym=20;//No .of symbols
data1=grand(1,sym,"uin",0,1);//Random symbol generation from 0 to 1 with uniform distribution
snr=10;// Signal to Noise Ratio
qpsk_mod=[];
bpsk_mod=2*data1-1;//BPSK Modulation
for j=1:2:length(data1)// Seperation of I & Q component for QPSK modulation
    i_phase=2*data1(j)-1;//BPSK modulation of I phase component
```

```
    q_phase=2*data1(j+1)-1;//BPSK modulation of Q phase component
    temp=i_phase+%i*q_phase;//Combinibg I phase and Q phase component for QPSK modulation
    qpsk_mod=[qpsk_mod temp];//QPSK modulated signal
  end

   noise=1/sqrt(2)*(10^(-(snr/20)))*(rand(1,length(bpsk_mod),'normal')+%i*(rand(1,length(bpsk_mod),'normal')));//White gaussian noise generation for bpsk
   noise1=1/sqrt(2)*(10^(-(snr/20)))*(rand(1,length(qpsk_mod),'normal')+%i*(rand(1,length(qpsk_mod),'normal')));//White gaussian noise generation for qpsk
   bpsk_awgn=bpsk_mod+noise;//BPSK Modulated signal passed over AWGN channel
  qpsk_awgn=qpsk_mod+noise1;//QPSK Modulated signal passed over AWGN channel

  figure//constellation diagram of ideal BPSK modulated signal and BPSK modulated signal with White Gaussian Noise
a = gca();//to handle various object
a.data_bounds = [ -1 , -1;1 ,1];
a.x_location = "origin";
a.y_location = "origin";
plot2d ( real(bpsk_mod),imag(bpsk_mod),-2);
plot2d ( real(bpsk_awgn),imag(bpsk_awgn),-5);
xlabel( 'In phase' );//X-axis label
ylabel( 'Quadrature phase' );//Y-axis label
title( 'Constellation for BPSK with AWGN' );//title of plot
legend(['Ideal message point';'message point with noise']);//legend
mtlb_axis([-2 2 -2 2]);//range of axis
figure//constellation diagram of ideal QPSK modulated signal and QPSK modulated signal with
```

```
White Gaussian Noise
a = gca();//to handle various object
a.data_bounds = [ -1 , -1;1 ,1];
a.x_location = "origin";
a.y_location = "origin";
plot2d ( real(qpsk_mod),imag(qpsk_mod),-2);
plot2d ( real(qpsk_awgn),imag(qpsk_awgn),-5);
xlabel( 'In phase' );//X-axis label
ylabel( 'Quadrature phase' );//Y-axis label
title( 'Constellation for QPSK with AWGN' );//title of plot
legend(['Ideal message point';'message point with noise']);//legend
mtlb_axis([-2 2 -2 2]);//range of axis
//Result:Generates two plots: BPSK modulated signal with and without noise-figure-0
                        //QPSK modulated signal with and without noise-figure-1
```

**Scilab code Solution 2.2** BER of BPSK and QPSK over AWGN Channel

```
//Performance comparison of Simulated BER and Theoritical BER of BPSK and QPSK modulation over AWGN channel
clc;
clear;
xdel(winsid());
sym=10000;//No .of symbols
M=4;
qpsk_mod=[];i_phase=[];
data1=grand(1,sym,"uin",0,1);//Random Symbol generation from 0 to 1 with uniform distribution
for j=1:2:length(data1)// Seperation of I & Q component
    i_phase=2*data1(j)-1;// BPSK modulation of I
```

```
phase component
    q_phase=2*data1(j+1)-1;//BPSK modulation of Q phase component
    temp=i_phase+%i*q_phase;//combining of I phase and Q phase component for QPSK modulation
    qpsk_mod=[qpsk_mod temp];//QPSK modulated signal
  end
bpsk_mod=2*data1-1;//BPSK Modulated signal

    snr=1:10;//Signal to Noise Ratio
     for k=1:1:length(snr)
         H=1/sqrt(2)*(rand(1,length(qpsk_mod),'normal')+%i*(rand(1,length(qpsk_mod),'normal')));
         noise1=1/sqrt(2)*(10^(-(k/20)))*(rand(1,length(qpsk_mod),'normal')+%i*(rand(1,length(qpsk_mod),'normal')));//White Gaussian Noise generation for QPSK
         noise=1/sqrt(2)*(10^(-(k/20)))*(rand(1,length(bpsk_mod),'normal')+%i*(rand(1,length(bpsk_mod),'normal')));//White Gaussian Noise generation for QPSK
                 rec1_qpsk=qpsk_mod+noise1;//QPSK modulated signal over AWGN channel
                 rec1_bpsk= bpsk_mod+noise;//BPSK modulated signal over AWGN channel

                 rec_data_qpsk=[];rec_data_bpsk=[];
                 rec1_i=real(rec1_qpsk);//Seperation of I phase and Q phase comopnent of received QPSK modulated signal
                 rec1_q=imag(rec1_qpsk);
         //
         for i=1:length(rec1_i)//QPSK Demodulation: BPSK demodulation of I phase and Q phase components
             if rec1_i(i)>=0
                   demod_out_i=1;
```

```
            else rec1_i(i)<0
                demod_out_i=0;
            end
              if rec1_q(i)>=0
                 demod_out_q=1;
            else rec1_q(i)<0
                demod_out_q=0;
            end
           rec_data_qpsk=[rec_data_qpsk demod_out_i demod_out_q];//QPSK Demodulated signal
            end
         for i=1:1:length(data1)//BPSK Demodulation
            if real(rec1_bpsk(i))>=0
                 demod_out_bpsk=1;
            else real(rec1_bpsk(i))<0
                demod_out_bpsk=0;
            end
            rec_data_bpsk=[rec_data_bpsk demod_out_bpsk];//BPSK Demodulated signal
        end

        errA=0;errB=0;
    for i=1:sym
        if rec_data_qpsk(i)==data1(i)
            errA=errA;
        else
            errA=errA+1;
        end
    end
        BER_qpsk(k)=errA/sym;// BER of QPSK

         for i=1:sym
        if rec_data_bpsk(i)==data1(i)
            errB=errB;
        else
            errB=errB+1;
        end
```

```
        BER_bpsk(k)=errB/sym;//BER of BPSK
    end
    theoryBer = 0.5*erfc(sqrt(10.^(snr/10))); //Theoritical BER of BPSK & QPSK
  end

      // end
snr=1:1:10;
plot2d(snr,BER_bpsk,5,logflag="nl");//plot simulated BER of BPSK over AWGN channel
plot2d(snr,BER_qpsk,2,logflag="nl");//plot simulated BER of QPSK over AWGN channel
plot2d(snr,theoryBer,3,logflag="nl");//Plot theoritical BER of QPSK and BPSK over AWGN channel
mtlb_axis([0 20 10^-5 0.5]);//axis
xgrid(10);
xtitle( 'Bit Error Rate plot for BPSK & QPSK Modulation', 'SNR', 'BER') ;//title of plot


legend(['BER_sim_BPSK';'BER_sim_QPSK';'BER_Theory']);//legend
//This experiments results plot of bit error rate(BER) comparison of simulated BPSK over AWGN channel,simulated QPSK over AWGN channel and theoritical BER of BPSK and QPSK
// It will take few minutes to get plots as 100000 bits are applied as an input to get better plots
```

# Experiment: 3

# Effect of various channel on transmitted data using different modulation techniques.

**Scilab code Solution 3.1** BER BPSK Rayleigh fading channel

```
//Error rate performance of BPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both
clc;
clear;
xdel(winsid());
sym=10000;//No .of symbols
data1=grand(1,sym,"uin",0,1);//Randomly generated Symbolsfrom 0 to 1with uniform distribution

bpsk_mod=2*data1-1;//BPSK Modulation
snr=1:20;//signal to Noise Ratio
    for k=1:1:length(snr)

        H1=1/sqrt(2)*(rand(1,length(bpsk_mod),'normal')+%i*(rand(1,length(bpsk_mod),'normal')));//Rayleigh fading generation
```

```
noise=1/sqrt(2)*(10^(-(k/20)))*(rand(1,length(bpsk_mod),'normal')+%i*(rand(1,length(bpsk_mod),'normal')));// White Gaussian Noise generation

    rec1_bpsk=bpsk_mod+noise;//BPSK modulated signal over AWGN channel
    rec1_bpsk_ray1= H1.*bpsk_mod+noise;//BPSK modulated signal over AWGN channel and Rayleigh Fading channel
   rec1_bpsk_ray=conj(H1).*rec1_bpsk_ray1;//multiplication with conjugate of rayleigh fading to nullify phase because of Rayleigh Fading
  //  rec1_bpsk_ray=rec1_bpsk_ray1./(H1.*conj(H1));

    rec_data_bpsk=[];rec_ray_bpsk=[];

for i=1:1:length(data1)//BPSK Demodulation of received signal over AWGN channel
  if real(rec1_bpsk(i))>=0
       demod_out_bpsk=1;
  else real(rec1_bpsk(i))<0
      demod_out_bpsk=0;
  end
  rec_data_bpsk=[rec_data_bpsk demod_out_bpsk];//Received signal

  if real(rec1_bpsk_ray(i))>=0 //BPSK Demodulation of received signal over AWGN channel and Rayleigh channel
       demod_ray_bpsk=1;
  else real(rec1_bpsk_ray(i))<0
      demod_ray_bpsk=0;
  end
```

```
            rec_ray_bpsk=[rec_ray_bpsk demod_ray_bpsk];////Received signal
        end

        errB=0;errC=0;
    for i=1:sym

        if rec_data_bpsk(i)==data1(i)//Error rate calculation of received signal by considering only AWGN Channel
            errB=errB;
        else
            errB=errB+1;
        end

        BER_bpsk(k)=errB/sym;//BER at receiver by considering only AWGN Channel

         if rec_ray_bpsk(i)==data1(i)//Error rate calculation of received signal by considering AWGN Channel and Rayleigh channel
            errC=errC;
        else
            errC=errC+1;
        end

        BER_bpsk_ray(k)=errC/sym;//BER at receiver by considering AWGN Channel and rayleigh channel
    end  end

       // end
snr=1:1:20;
plot2d(snr,BER_bpsk,5,logflag="nl");
plot2d(snr,BER_bpsk_ray,3,logflag="nl");
mtlb_axis([0 20 10^-5 0.5]);
xgrid(10);
```

```
xtitle( 'Bit Error Rate plot for BPSK modulated signal over AWGN channel and AWGN and Rayleigh channel both', 'SNR', 'BER') ;
legend(['BER_BPSK_AWGN';'BER_BPSK_AWGN & Rayleigh']) ;
//This experiment results plot of error rate performance of BPSK modulated signal over AWGN channe and AWGN and Rayleigh channel both.
//This experiment will take some time to display plot as higher no. of bits entered as an input to get better plots.
```

**Scilab code Solution 3.2** BER QPSK Rayleigh channel

```
//Error rate performance of QPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both
clc;
clear;
xdel(winsid());
sym=10000;//No .of symbols
M=4;
qpsk_mod=[];i_phase=[];
data1=grand(1,sym,"uin",0,1);//Random Symbol generation from 0 to 1 with uniform distribution
for j=1:2:length(data1)// Seperation of I & Q component
    i_phase=2*data1(j)-1;// BPSK modulation of I phase component
    q_phase=2*data1(j+1)-1;//BPSK modulation of Q phase component
    temp=i_phase+%i*q_phase;//combining of I phase and Q phase component for QPSK modulation
    qpsk_mod=[qpsk_mod temp];//QPSK modulated signal
 end
```

```
snr=1:5:41;//Signal to Noise Ratio
 for k=1:length(snr)
     H=1/sqrt(2)*(rand(1,length(qpsk_mod),'normal')+%i*(rand(1,length(qpsk_mod),'normal')));//Rayleigh fading generation

     noise1=1/sqrt(2)*(10^(-(k/20)))*(rand(1,length(qpsk_mod),'normal')+%i*(rand(1,length(qpsk_mod),'normal')));//White Gaussian Noise generation for QPSK

         rec1_qpsk=qpsk_mod+noise1;//QPSK modulated signal over AWGN channel
          rec1_qpsk_ray1= H.*qpsk_mod+noise1;//BPSK modulated signal over AWGN channel and Rayleigh Fading channel
         rec1_qpsk_ray=conj(H).*rec1_qpsk_ray1;//multiplication with conjugate of rayleigh fading to nullify phase because of Rayleigh Fading

         rec_data_qpsk=[];rec_data_qpsk_ray=[];

         rec1_i=real(rec1_qpsk);//Seperation of I phase and Q phase comopnent of received QPSK modulated signal
         rec1_q=imag(rec1_qpsk);

          rec1_i_ray=real(rec1_qpsk_ray);//Seperation of I phase and Q phase comopnent of received QPSK modulated signal
         rec1_q_ray=imag(rec1_qpsk_ray);
     //
     for i=1:length(rec1_i)//QPSK Demodulation:
```

```
            BPSK demodulation of I phase and Q phase components
            if rec1_i(i)>=0
                 demod_out_i=1;
            else rec1_i(i)<0
                demod_out_i=0;
            end
              if rec1_q(i)>=0
                 demod_out_q=1;
            else rec1_q(i)<0
                demod_out_q=0;
             end
                if rec1_i_ray(i)>=0
                 demod_out_i_ray=1;
            else rec1_i(i)<0
                demod_out_i_ray=0;
            end
              if rec1_q_ray(i)>=0
                 demod_out_q_ray=1;
            else rec1_q_ray(i)<0
                demod_out_q_ray=0;
            end
           rec_data_qpsk=[rec_data_qpsk demod_out_i demod_out_q];//QPSK Demodulated signal
           rec_data_qpsk_ray=[rec_data_qpsk_ray demod_out_i_ray demod_out_q_ray];// QPSK Demodulated signal
            end

        errA=0;errB=0;
    for i=1:sym
        if rec_data_qpsk(i)==data1(i)
            errA=errA;
        else
            errA=errA+1;
        end
    end
        BER_qpsk(k)=errA/sym;// BER of QPSK
```

```
            for i=1:sym
          if rec_data_qpsk_ray(i)==data1(i)
              errB=errB;
          else
              errB=errB+1;
          end

          BER_qpsk_ray(k)=errB/sym;//BER of BPSK
      end
      //theoryBer = 0.5*erfc(sqrt(10.^(snr/10))); //
         Theoritical BER of BPSK & QPSK
   end

         // end
snr=1:5:41;
plot2d(snr,BER_qpsk,5,logflag="nl");//plot simulated
    BER of BPSK over AWGN channel
plot2d(snr,BER_qpsk_ray,2,logflag="nl");//plot
   simulated BER of QPSK over AWGN channel
//plot2d(snr,theoryBer,3,logflag="nl");//Plot
   theoritical BER of QPSK and BPSK over AWGN
   channel
mtlb_axis([0 40 10^-5 0.5]);//axis
xgrid(10);
xtitle( 'Bit Error Rate plot for QPSK over AWGN
   channel & AWGN and Rayleigh channel both', 'SNR',
    'BER') ;//title of plot

legend(['BER_QPSK_AWGN';'BER_QPSK_AWGN & Rayleigh'])
   ;//legend
//This experiments results plot of bit error rate(
   BER) comparison of simulated QPSK over AWGN
   channel,simulated QPSK over AWGN channel and
   Rayleigh fading channel.
// It will take few minutes to get plots as 10000
   bits are applied as an input to get better plots
```

**Scilab code Solution 3.3** 1

```
//Error rate performance of BPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both
clc;
clear;
xdel(winsid());
sym=10000;//No .of symbols
data1=grand(1,sym,"uin",0,1);//Randomly generated Symbolsfrom 0 to 1with uniform distribution

bpsk_mod=2*data1-1;//BPSK Modulation
snr=1:20;//signal to Noise Ratio
    for k=1:1:length(snr)

        H1=1/sqrt(2)*(rand(1,length(bpsk_mod),'normal')+%i*(rand(1,length(bpsk_mod),'normal')));//Rayleigh fading generation

        noise=1/sqrt(2)*(10^(-(k/20)))*(rand(1,length(bpsk_mod),'normal')+%i*(rand(1,length(bpsk_mod),'normal')));// White Gaussian Noise generation

            rec1_bpsk=bpsk_mod+noise;//BPSK modulated signal over AWGN channel
            rec1_bpsk_ray1= H1.*bpsk_mod+noise;//BPSK modulated signal over AWGN channel and Rayleigh Fading channel
           rec1_bpsk_ray=conj(H1).*rec1_bpsk_ray1;//multiplication with conjugate of rayleigh fading to nullify phase
```

```
                because of Rayleigh Fading
            //  rec1_bpsk_ray=rec1_bpsk_ray1./(H1.*conj(H1));

              rec_data_bpsk=[];rec_ray_bpsk=[];

        for i=1:1:length(data1)//BPSK Demodulation of received signal over AWGN channel
            if real(rec1_bpsk(i))>=0
                 demod_out_bpsk=1;
            else real(rec1_bpsk(i))<0
                demod_out_bpsk=0;
            end
            rec_data_bpsk=[rec_data_bpsk demod_out_bpsk];//Received signal

            if real(rec1_bpsk_ray(i))>=0 //BPSK Demodulation of received signal over AWGN channel and Rayleigh channel
                 demod_ray_bpsk=1;
            else real(rec1_bpsk_ray(i))<0
                demod_ray_bpsk=0;
            end
            rec_ray_bpsk=[rec_ray_bpsk demod_ray_bpsk];////Received signal
        end

        errB=0;errC=0;
    for i=1:sym

        if rec_data_bpsk(i)==data1(i)//Error rate calculation of received signal by considering only AWGN Channel
            errB=errB;
        else
            errB=errB+1;
        end

```

```
        BER_bpsk(k)=errB/sym;//BER at receiver by considering only AWGN Channel

         if rec_ray_bpsk(i)==data1(i)//Error rate calculation of received signal by considering AWGN Channel and Rayleigh channel
            errC=errC;
        else
            errC=errC+1;
        end

        BER_bpsk_ray(k)=errC/sym;//BER at receiver by considering AWGN Channel and rayleigh channel
    end  end

       // end
snr=1:1:20;
plot2d(snr,BER_bpsk,5,logflag="nl");
plot2d(snr,BER_bpsk_ray,3,logflag="nl");
mtlb_axis([0 20 10^-5 0.5]);
xgrid(10);
xtitle( 'Bit Error Rate plot for BPSK modulated signal over AWGN channel and AWGN and Rayleigh channel both', 'SNR', 'BER') ;
legend(['BER_BPSK_AWGN';'BER_BPSK_AWGN & Rayleigh']);
//This experiment results plot of error rate performance of BPSK modulated signal over AWGN channe and AWGN and Rayleigh channel both.
//This experiment will take some time to display plot as higher no. of bits entered as an input to get better plots.
```

# Experiment: 4

# Trunking Theory for Probability of blocking(Erlang B) and probability of delay( Erlang C).

**Scilab code Solution 4.1** Traffic calculation inErlang B and Erlang C system

```
//Exp-4 Calculates maximum traffic intensity and maximum no. of users accomodated in Erlang B and Erlang C system for given no of channels
clc;
clear;
xdel(winsid());

 function [p1]=erlangB(A1,c1)// calculate blocking probability for Erlang B system
      pr2=0;
        pr1=A1^c1/factorial(c1);
        for k=1:c1
        pr2=pr2+(A1^k/factorial(k));
        end
```

```
      //  A1=A1+1;
        p1=pr1/pr2;
  endfunction

  function [p2]=erlangC(A2,c2)// calculate probability of blocked call delayed in Erlang C system
      temp_1=0;
  for k=0:c2-1
    temp_1=temp_1+A2^k/factorial(k);
end
denominator=A^c2+(factorial(c2)*(1-(A2/c))*temp_1);
p2=A2^c2/denominator;
endfunction

pr_blocking=input('enter probability of blocking'); //enter probability of blocking for perticular system
pr_delay=input('enter probability of block call delay');//enter probability of blocked call delayed for particular system
y=input('enter call rate');// Average no .of calls per minute
H=input('enter the average call duration'); // Average call duration in minute
c=input("enter no.of channels");//Enter no. of channels
disp("no.of channel=");
disp(c);
Au=y*H;//Traffic intensity per user

 p=0;
     for A=1:1:100
         while(p<pr_blocking)//Find maximum traffic intensity for entered blocking probability pr_blocking
        [p]=erlangB(A,c)//calling function erlangB
        A=A+1;
```

```
        end
      disp(pr_blocking,'for blocking probability of');//display blocking probability
      disp(A-1,'Maximum traffic intensity is');//display max. traffic intensity
       u=(A-1)/Au;//no. of users calculation
      disp(u,"no .of users are accomodated");//display maximum no.of users accomodated in Erlang B system
      break;
  end//
  p=0;
      for A=1:1:100
      while(p<pr_delay)//Find maximum traffic intensity for entered blocking probability pr_blocking
        [p]=erlangC(A,c)//calling funtion to calculate erlang C probability
        A=A+1;
    end
      disp(pr_delay,'for block call delay probability of');//display blocking probability
      disp(A-1,'Maximum traffic intensity is');//display max. traffic intensity
      u=(A-1)/Au;
      disp(u,"no.of users are accomodated");//display maximum no.of users accomodated in Erlang C system
      break;
  end
 //Enter blocking probability pr_blocking=0.01
 //Enter probabolity of block call delay pr_delay=0.1
 //Enter call rate= 3/60
 //enter call duration= 2(in minute)
 //Enter no of channels 50

```

```
//Output:
//no.of channel=     50.

//  for blocking probability of  0.01
// Maximum traffic intensity is 38.
// no .of users are accomodated 380.

// for block call delay probability of 0.1
// Maximum traffic intensity is 41.
//no.of users are accomodated   410.
```

# Experiment: 5

# Walsh Code generation

**Scilab code Solution 5.1** Walsh code generation and spreading and despreading using Walsh code

```
// Walsh Code generation
//Spreading and despreading of information for three users using Walsh code
clc;
clear;
xdel(winsid());
a=input('enter the number order of 2:');//input required length of Walsh Code which is always order of 2
c1=[1 -1 -1];//information of user 1
c2=[-1 1 -1];//information of user 2
c3=[1 -1 1];//information of user 3
W=[0 0;0 1];// Basic Walsh code Matrix
m=2;
%n=2^m;
for m =2:1:a
for i = 1:1:a//genration of walsh code matrix of entered length
    if i==2^m
    Winv=bitcmp(W,1);
```

```
        W=[W W;W Winv];
        end

end
end
temp=0;
W1=[];
disp(W)
for i=1:1:length(W(1,:))//0 replaced by -1 in walsh code matrix
    for j=1:1:length(W(1,:))
        if W(i,j)==0 then
            W(i,j)=W(i,j)-1;
         else W(i,j)=W(i,j)+0;

     end

end

end
//disp(W)
//spreading using Walsh code
tans_c1=[c1(1,1).*W(1,:) c1(1,2).*W(1,:) c1(1,3).*W(1,:)];//spreading of user 1 information using first row of Walsh Matrix
tans_c2=[c2(1,1).*W(2,:) c2(1,2).*W(2,:) c2(1,3).*W(2,:)];//spreading of user 2 information using second row of Walsh Matrix
tans_c3=[c3(1,1).*W(3,:) c3(1,2).*W(3,:) c3(1,3).*W(3,:)];//spreading of user 3 information using third row of Walsh Matrix
aa1=tans_c1(1,1:a)+tans_c2(1,1:a)+tans_c3(1,1:a);
aa2=tans_c1(1,(a+1):(2*a))+tans_c2(1,(a+1):(2*a))+tans_c3(1,(a+1):(2*a));
aa3=tans_c1(1,((2*a))+1:(3*a))+tans_c2(1,((2*a))+1:(3*a))+tans_c3(1,((2*a))+1:(3*a));
tans_sig=[aa1 aa2 aa3];//transmission of spreaded signal
```

```
det_code1=input('enter detection code');//Enter any integer no. ranging up to no. of rows of walsh matrix

    select det_code1//select case to get information of entered user
    case 1
        det_code=W(1,:);
    case 2
        det_code=W(2,:);
     case 3
        det_code=W(3,:);
    else
       det_code=W(4,:);
       disp('invalid detection code');//display message for input of invalid detection code
        end


rec_sig =[det_code(1,:).*aa1 det_code(1,:).*aa2 det_code(1,:).*aa3];//received signal multiplied with detection code
det_sig=[rec_sig(1,1)+rec_sig(1,2)+rec_sig(1,3)+rec_sig(1,4) rec_sig(1,5)+rec_sig(1,6)+rec_sig(1,7)+rec_sig(1,8) rec_sig(1,9)+rec_sig(1,10)+rec_sig(1,11)+rec_sig(1,12)];//detection of information from received signal
final_sig=(1/4)*det_sig;
disp('transmited information is');
disp(final_sig)//information transmmited using selected valid detection code
//input a=4
//W=[0 0 0 0 ;0 1 0 1;0 0 1 1 ;0 1 1 0]
//detection code=2, output=-1 1-1(information of user 2 spreaded with second row of Walsh Matrix)
//detection code > 3 , results : code not available 0 0 0
```

# Experiment: 6

# PN sequence generation.

**Scilab code Solution 6.1** 3 bit PN sequence generation and spreading and despreading using PN sequence and shifted PN sequence

```
// Spreading  of sequence using PN sequence and
   despreading of sequence using PN sequence and
   shifted PN sequence
clc;
clear;
xdel(winsid());
// Generation of 7 bit PN sequence
// Coefficient of polynomial
a1=1;
a2=1;
a3=1;
// Initial states of flip flop
R(1)=1;
R(2)=0;
R(3)=0;
m=3;
disp('output after every clock pulse');
for i=1:((2^m)-1)//shift of bit in each register for
    every clock pulse
    r1=R(1);
```

```
    r2=R(2);
    r3=R(3);
    PN(i)=R(3);
    //if(a1==0)
R1=bitxor(r2,r3);//input of register is modulo2 addition of R2 and R3
R(3)=R(2);
R(2)=R(1);
R(1)=R1;

disp(R);
end
disp('PN sequence is');
disp(PN);//Display 7 bit PN sequence
c1=[1 -1 -1];//information of user 1
 for j=1:1:length(PN)//0 replaced with -1 in PN sequence
        if PN(j)==0 then
            PN(j)=PN(j)-1;
         else PN(j)=PN(j)+0;
             end

     end
     disp(PN);
spreaded_sig=[c1(1).*PN' c1(2).*PN' c1(3).*PN']//Spreading of data of user 1 using PN sequence
detect_code=[spreaded_sig(1:7).*PN' spreaded_sig(8:14).*PN' spreaded_sig(15:21).*PN'];//at receiver, recieved spreaded signal multiplied with PN sequnce
corr_code=[sum(detect_code(1:7)) sum(detect_code(8:14)) sum(detect_code(15:21))];
rec_sig=(1/7).*corr_code;//get information form received signal
disp('received signal with correct PN sequence is');
disp(rec_sig);//received data of user 1 at receiver :1 -1 -1
//Despreading with shifted PN sequence
```

```
shift_fact=input('enter the shifting factor');
l=1;
    k=shift_fact-1;
for i=1:1:length(PN)    //generation of shifted PN sequence as per entered shifting factor
    if  i<=shift_fact
         shift_seq(i)=PN(length(PN)-k);
         k=k-1;
         else i>shift_fact
         shift_seq(i)=PN(l);
         l=l+1;
       end
      end
      disp('shifted sequence is');
      disp(shift_seq');//display shifted sequence
 //despreading using shifted PN sequence
detect_shift_code=[spreaded_sig(1:7).*shift_seq' spreaded_sig(8:14).*shift_seq' spreaded_sig(15:21).*shift_seq'];
corr_shift_code=[sum(detect_shift_code(1:7)) sum(detect_shift_code(8:14)) sum(detect_shift_code(15:21))];
rec_shift_sig=(1/7).*corr_shift_code;
    disp("recieved signal with shifted PN sequence is ");
    disp(rec_shift_sig);//Invalid data received beacuse signal was despreded with shifted PN sequence
    disp('which is not valid transmitted signal');
    //Result:
    //output of PN sequence generator after each clock pulse
   // PN =0 0 1 0 1 1 1 replace 0 with -1,PN=-1 -1 1 -1 1 1 1
   //entered shifting factor =3, shifted PN sequence= 1 1 1 -1 -1 1 -1
   //Invalid signal is received when despreading is with shifted version of PN
```

```
//rec_shift_sig=- 0.1428571      0.1428571
    0.1428571
```

# Experiment: 7

# Equalization.

**Scilab code Solution 7.1** Adaptive equalization using LMS filter

```
// Least Mean Square adaptive equalizer
clc;
clear all;
xdel(winsid());
numPoints = 500;
numTaps = 1;          //channel order
Mu = 0.01;            //iteration step size

// input is guassian
x = rand(numPoints,1,'normal') + %i*rand(numPoints
    ,1,'normal');
//choose channel to be random uniform
h = rand(numTaps,1) + %i*rand(numTaps, 1);

h = h/max(abs(h));    //normalize channel
// convolve channel with the input
d = filter(h, 1, x);

//initialize variables
w = [];
y = [];
```

```
in = [];
e = []; // error, final result to be computed

w = zeros(numTaps+1,1) + %i*zeros(numTaps+1,1);
kk=1;
aa(kk,:)=w';
//LMS Adaptation
for n  = numTaps+1 : numPoints

    // select part of training input
    in = x(n : -1 : n-numTaps) ;
    y(n) =w'* in;

    // compute error
    e(n) = d(n)-y(n);

    // update taps

w = w+ Mu*( real(e(n)*conj(in)) - %i*imag(e(n)*conj(
   in)) );

kk=kk+1;
aa(kk,:)=w';
end

// Plot results
figure;
iter=1:500
plot2d(iter,abs(e),5,logflag="nn");
title(['LMS Adaptation Learning Curve Using Mu =
   0.01']);
xlabel('Iteration Number');
ylabel('Output Estimation Error in dB');
figure;
plot3d(abs(aa(:,1)),abs(aa(:,2)),abs(e));
title('LMS adaption curve with weight factors');
xlabel('adaptive weight factor1');
ylabel('adaptive weight factor2');
```

```
zlabel('mean square error');
// Output shows plot of MSE with no. of iterations
   in figure 1 and 3D plot of MSE with weight
   factors
```

# Experiment: 8

# Channel Coding using Linear Block Code

**Scilab code Solution 8.1** Linear Block Coding over AWGN channel

```


//this is a linear block coding and decoding over awgn channel
// 4 bits input signal is coded with linear block code (4,7), 7 bit coded signal is transmitted over awgn channel and at receiver side signal is decoded. If there is error in one bit, li//near block code correct that error and original transmitter code is receved.
//If error is in more than one bit, code is not corrected so wrong code is recieved
clc;
clear all;
xdel(winsid());
global P n k;

n=7;//length of coded input
k=4;//length of input
```

```
P=[1 1 0; 0 1 1; 1 0 1;1 1 1]; //parity  matrix of size k*(n-k) to be
//                                selected so that the systematic generator
//                                matrix is linearly independent or full rank
//                                matrix

//(n,k) linear block code where k - no. of input data bits and n-no. of o/p
//data bits. code rate=k/n
// x is an input vector containing k bits

//This is an linear block encoding function
function y1=linblkcode(x);
global P n k;
n=7;
k=4;
P=[1 1 0; 0 1 1; 1 0 1;1 1 1];//parity matrix
//x=[0 1 1 0];

//G=[ ];     //      % Generator matrix k*n
G=[eye(k,k) P];

y1=zeros(1,n);
for i=1:k//linear block coding
    var(i,:)=x(1,i) & G(i,:);
    var(i,:)=bool2s(var(i,:));
    y1(1,:)=bitxor(var(i,:),y1(1,:));//coded signal
end

endfunction


//%This is a linear block syndrome decoding function file%

function x1=linblkdecoder(y)
```

```
//% here y is recieved vector 7 bits long

//% (7,4) linear block code
global P n k;


//H=[ ]; //% PARITY CHECK MATRIX

H=[P' eye((n-k),(n-k))];
Ht=H'; //%transpose of H

S=zeros(1,n-k); //%syndrome of recieved vector x
for i=1:n-k// decoding of linear block code
    S(i)=y(1) & Ht(1,i);
    S(i)=bool2s(S(i));
    for j=2:n

        S(i)=bitxor(S(i), bool2s((y(j) & Ht(j,i))));
           //decoded signal
    end
end



//%%****SYNDROME LOOK UP TABLE************

//%%*************************************
//%
if S==[0 0 0]
   e=[0 0 0 0 0 0 0];
    z=bitxor(y,e);
end

if S==[0 0 1]
   e=[0 0 0 0 0 0 1];
    z=bitxor(y,e);
end
if S==[0 1 0]
```

```
    e=[0 0 0 0 0 1 0];
     z=bitxor(y,e);
end
if S==[1 0 0]
    e=[0 0 0 0 1 0 0];
     z=bitxor(y,e);
end
if S==[1 1 1]
    e=[0 0 0 1 0 0 0];
     z=bitxor(y,e);
end
if S==[1 0 1]
    e=[0 0 1 0 0 0 0];
     z=bitxor(y,e);
end
if S==[0 1 1]
    e=[0 1 0 0 0 0 0];
     z=bitxor(y,e);
end
if S==[1 1 0]
    e=[1 0 0 0 0 0 0];
     z=bitxor(y,e);
end
//disp('error');
//disp(e);
x1=z(1,1:k);
endfunction
snr_dB=2;

x=[1 0 0 1];    //                input bits to the encoder of size 1* k
y1=linblkcode(x);//            //   y1 is the output of linear block encoder
n1 = 1/sqrt(2)*[rand(1,length(y1),'normal') + %i*rand(1,length(y1),'normal')];//white gaussian noise generation
r=y1+ 10^(-snr_dB/20)*n1;//received signal over awgn channel
```

```
//r1=real(r)
rec=real(r)>=0.5;//detection of bit 1 and 0 in received signal
rec_fin=bool2s(rec);//convert boolean matrix to zero one matrix
//rec_err=rec_fin==y1;
//no_err=bool2s(rec_err);
disp('The information signal=')//display input
disp(x)
disp('The transmitted encoded signal=')//display coded signal
disp(y1)
disp('The recieved signal=')//display received signal
disp(rec_fin);
x1=linblkdecoder(rec_fin); //          % x1 is the output of the linear block decoder
disp('The decoded signal=')//display decoded signal
disp(x1);
if x1==x then disp('one or less than one error so correct code is received');
else
    disp('more than one error so wrong code detected');
end
//Output: The information signal is : 1001
//transmitted code is : 1001001
//1. received signal is :1011001(e.g)(error in only one bit)
//decoded signal: 1001
//one or less than one error so correct code is received
//2. received signal is:1011011(e.g)(error in more than one bits)
//decoded signal:1010
//more than one error so wrong code is received
```

# Experiment: 9

# Transmit and receive diversity

**Scilab code Solution 9.1** Selection Diversity over AWGN channel

```
//ber performance with 1, 2 and 3 receiver antennas over awgn channel using selection diversity
clc;
clear;
xdel(winsid());
sym=10000;//no. of symbols
data1=grand(1,sym,"uin",0,1);//randomly generated input
s = 2*data1-1; // BPSK modulation 0 -> -1; 1 -> 1
nRx = [1 2 3];//no .of receiving antennas
snr_dB =   [1:10]; // signal to noise ratio
for j = 1:length(nRx)
    for i = 1:length(snr_dB)
        n = 1/sqrt(2)*[rand(nRx(j),sym,'normal') + %i*rand(nRx(j),sym,'normal')]; //white gaussian noise

        y = ones(nRx(j),1)*s + 10^(-snr_dB(i)/20)*n; //received signal over awgn channel
         [yHat1 ind] = mtlb_max(y,[],1);//find strongest received signal from all
```

```
            antennas

        ipHat1 = real(yHat1)>0;
        ipHat = bool2s(ipHat1);//boolean to zero one matrix conversion
        // effective SNR
         nErr(j,i) = size(find([data1- ipHat]),2);// no. of error calculation
            end
end
simBer = nErr/sym; //BER calculation
// plot of ber comparison plot for 1,2 and 3 receiving antennas
snr_dB=1:10
plot2d(snr_dB,simBer(1,:),5,logflag="nl");
plot2d(snr_dB,simBer(2,:),2,logflag="nl");
plot2d(snr_dB,simBer(3,:),12,logflag="nl");
xgrid
legend( ['1X1';'1X2';'1x3']);
xlabel('Number of receive antenna');
ylabel('effective SNR, dB');
title('SNR improvement with Selection Combining');
//output presents BER performance comparison plots with 1,2 and 3 receiving antennas over awgn channels
```

**Scilab code Solution 9.2** Maximal Ratio Combining over AWGN and Rayleigh fading Channel

```
// BER Performance coamparison with one receivivng atenna and two receiving antennas with Maximal ratio Combining diversity technique over awgn channe and rayleigh fading channel
clc;
clear;
```

```
xdel(winsid());
sym=100000;// no. of symbols
M=2;
data1=grand(1,sym,"uin",0,1);// input signal is randomly generated
//N = 10; % number of bits or symbols
//ip = rand(1,N)>0.5; % generating 0,1 with equal probability
s = 2*data1-1; // BPSK modulation 0 -> -1; 1 -> 1
nRx = [1 2];//no of receivers
snr_dB =   [1:20]; // signal to noise ration in dB
for jj = 1:length(nRx)
    for ii = 1:length(snr_dB)
        n = 1/sqrt(2)*[rand(nRx(jj),sym,'normal') + %i*rand(nRx(jj),sym,'normal')]; //white gaussian noise,
        h = 1/sqrt(2)*[rand(nRx(jj),sym,'normal') + %i*rand(nRx(jj),sym,'normal')]; // Rayleigh fading channel
        // Channel and noise Noise addition
        sD = kron(ones(nRx(jj),1),s);
        y = h.*sD + 10^(-snr_dB(ii)/20)*n;// received signal over awgn channel and ayleigh fading channel
        // finding the power of the channel on all rx chain
         yHat =  sum(conj(h).*y,1)./sum(h.*conj(h),1); // maximal ratio combining
       // hPower1 = h.*conj(h);

        ipHat = real(yHat)>0;
        // effective SNR
                nErr(jj,ii) = size(find([data1-ipHat]),2);//calculate error
            end
end
simBer = nErr/sym; //bit error rate calculation
// plot
```

```
snr_dB=1:20
plot2d(snr_dB,simBer(1,:),5,logflag="nl");//snr- ber plot with one receiving antenna
plot2d(snr_dB,simBer(2,:),2,logflag="nl");//snr- ber plot with two receiving antennas
//plot(nRx,10*log10(EbN0EffSim),'bp-','LineWidth',2);
//mtlb_axis([1 20 0 6])
xgrid
legend(['1X1';'1X2']);
xlabel('Number of receive antenna');
ylabel('effective SNR, dB');
title('SNR improvement with Maximal ratio Combining');
```

# Experiment: 10

# Speech coding

**Scilab code Solution 10.1** speech coding and Decoding using LPC

```
//Exp-10 Speech coding using Long Term Predictive coder
//This code read wav file and play original signal and compressed signal
// It also plots original signal as well as compressed signal

function [aCoeff, tcount_of_aCoeff, e] = func_lev_durb(y, M);
//M=order and y is array of the data point  of the current frame
sk=0;        //initializing summartion term "sk"
a=[zeros(M+1);zeros(M+1)]; //defining a matrix of zeros for "a" for init.
//MAIN BODY OF THIS PROGRAM STARTS FROM HERE >>>>>>>>>>>>>>
z=xcorr(y);

//finding array of R[l]
R=z( ( (length(z)+1) ./2 ) : length(z)); //R=array of "R[l]", where l=0,1,...(b+N)-1  %R(1)=R[lag
```

```
=0], R(2)=R[lag=1], %R(3)=R[lag=2]... etc

//GETTING OTHER PARAMETERS OF PREDICTOR OF ORDER "0":
s=1;          //s=step no.
J(1)=R(1);            //J=array of "Jl", where l=0,1,2...(b+N)-1,    J(1)=J0, J(2)=J1, J(3)=J2 etc

//GETTING OTHER PARAMETERS OF PREDICTOR OF ORDER "(s-1)"
for s=2:M+1,
    sk=0;                    //clearing "sk" for each iteration
    for i=2:(s-1),
        sk=sk + a(i,(s-1)).*R(s-i+1);
    end                      //now we know value of "sk", the summation term
                             //of formula of calculating "k(l)"
    k(s)=(R(s) + sk)./J(s-1);
    J(s)=J(s-1).*(1-(k(s)).^2);

    a(s,s)= -k(s);
    a(1,s)=1;
    for i=2:(s-1),
        a(i,s)=a(i,(s-1)) - k(s).*a((s-i+1),(s-1));
    end
    end
//increment "b" and do same for next frame until end of frame when
//combining this code with other parts of LPC algo

//PREDICTION ERROR; FOR TESTING THE ABOVE PREDICTOR
aCoeff=a((1:s),s)';        //array of "a(i,s)", where, s=M+1
tcount_of_aCoeff = length(aCoeff);

 y_padded_for_delay_r = [y'; zeros(1,1)]; //it is
```

```
padded with zeros to remove the effect of delay in filter
 est_y_with_dummy_pad = filter([0 -aCoeff(2:9)],1, y_padded_for_delay_r);    // = s^(n) with a cap on page 92 of the book
est_y = est_y_with_dummy_pad(2:321);
e = y' - est_y;      //supposed to be a white noise
endfunction

function [aCoeff, b_LTopt, Topt, e_prime] = f_ENCODER_relp(x, fs)
M = 8; //prediction order for LP analysis
//INITIALIZATION;
b=1;         //index no. of starting data point of current frame
fsize = 20e-3;    //frame size (in milisec)
frame_length = round(fs .* fsize);   //=number data points in each framesize of "x"
N= frame_length - 1; //N+1 = frame length = number of data points in each framesize

y_proc = filter([1 -1], [1 -0.999], x);  //pre-processing
//FRAME SEGMENTATION
for b=1 : frame_length : (length(x) - N)

y_f = y_proc(b:b+N);     //"b+N" denotes the end point of current frame. "y" denotes an array of the data points of the current frame
 //LP ANALYSIS [lev-durb] & PREDICTION ERROR (short-term) FILTER;
    [a, tcount_of_aCoeff, e_s] = func_lev_durb (y_f, M); //e=error signal from lev-durb proc
    aCoeff(b: (b + tcount_of_aCoeff - 1)) = a;  //aCoeff is array of "a" for whole "x"
    //LONG-TERM LP ANALYSIS, FILTERING, AND CODING analysis:
    T_min = round (fs .* 5e-3);  //=total data
```

```
points in 5ms of "x"
T_max = round (fs .* 15e-3);
c1 = 1;
for bs = b : 40 : b+length(y_f)-40  //subframing bs = 1281;
    if bs < T_max
        break;
    end

    Jmin(bs) = 10^9;

    for T = T_min : T_max      //within 1 (current) frame  T = 40;
            for c = 1:40   //data points of current subframe c=1; temporary
                sm1(c) = ( y_proc(bs+(c-1)) .* y_proc(bs-T+(c-1))); //es(n)
                sm2(c) = y_proc(bs-T+(c-1)); //=es(n-T)
                sm22(c) = sm2(c).^2;
            end
            q1 = sum(sm1);
            q2 = sum(sm22);
            b_LT(T) = -(q1./q2);
        //J loop:
            for c = 1:40    //data points of current subframe c=1; temporary
                smJ1(c) = y_proc(bs+(c-1));
                smJ2(c) = b_LT(T) .* y_proc(bs-T+(c-1));
            end
            smJ = smJ1 + smJ2;
            qJ = smJ.^2;
            J(T) = sum(qJ);

            if J(T) < Jmin(bs),
                Jmin(bs) = J(T);
                Topt(bs) = T;
```

```
                    if b_LT(T)>=1,
                        b_LTopt(bs) = 0.9999; //trancation
                    else
                        b_LTopt(bs) = b_LT(T);
                    end
                else
               end
        end     //T loop ends
        //predictor:
        LT_gain = [zeros(1, Topt(bs)-1), b_LTopt(bs)];     //as it says z^-T in page 121
         e_s_padded_for_delay_r = [e_s(c1:c1+39); zeros(Topt(bs), 1)]; //it is padded with zeros to remove the effect of delay in filter. %Topt(bs) no. of 'z's and one '1' results in total 'Topt(bs)' amount of delay
        e_with_dummy_pad = filter([1 LT_gain], 1, e_s_padded_for_delay_r);   // = 1 + 0*z^-1 + 0*z^-2 + ... + b*z^-T
         e_LT(bs:bs+39,1) = e_with_dummy_pad(Topt(bs)+1 : Topt(bs)+1+39);   //LT predicted "e"
        e(bs:bs+39, 1) = e_s(c1 : c1+39) - e_LT(bs : bs+39);

        //WEIGHTING FILTER:
        w = [-0.0004; -0.0156;-0.0677;0.0545;0.6069;1.0000;0.6069;0.0545;-0.0677 //11 point flattop window is temporarily chosen
        wndd = conv(w, e(bs:bs+39));     //outputs total 50 samples
        x_n(bs:bs+39) = wndd(6:45);  //middle 40 samples are taken

        //POSITION SELECTION & EXCITATION GENERATOR:
```

```
        for i1 = 0:3
            for i = i1+bs : 3 : bs+i1+38;
                x_m(i1+1,i) = x_n(i);
            end

            E_m(i1+1,1) = sum((x_m(i1+1, bs:3)).^2);
        end
        [E_m_max, index_max] = gsort(E_m);
        e_prime(bs : bs+39) = x_m(index_max(4), bs:
           bs+39);
        c1 = c1 + 40;
    end
end
endfunction

//RELP DECODER portion:
function [synth_speech, synth_speech1, LT_gain,
   e_prime_pad_for_d_r, e_prime_op_dummy_pad,
   e_prime_op, e_prime_op_pad_delay_r,
   synth_speech_dummy_pad] = f_DECODER_relp(aCoeff,
   b_LTopt, Topt, e_prime)
//re-calculating frame_length for this decoder
frame_length=9; //initial value for calculation
for i=10:length(aCoeff)
    if aCoeff(i) == 0
        frame_length = frame_length + 1;
    else break;
    end
end
e_prime = e_prime';     //making it a column matrix
   for convenience

for b=1 : frame_length : length(aCoeff)    //length(
   aCoeff) should be very close (i.e less than a
   frame_length error) to length(x)
    for bs = b : 40 : b+frame_length-40  //
       subframing

```

```
        //EXCITATION GENERATOR:  not done yet. because e_prime has been sent to this decoder directly. without quantization.
        //PITCH SYNTHESIS FILTER:    %has to be done per subframe
        LT_gain = [zeros(1, Topt(bs)-1), b_LTopt(bs)];     //as it says z^-T
         e_prime_pad_for_d_r = [e_prime(bs:bs+39); zeros(Topt(bs), 1)]; //it is padded with zeros to remove the effect of delay in filter. %Topt(bs) no. of 'z's and one '1' results in total 'Topt(bs)' amount of delay
        e_prime_op_dummy_pad = filter(1, [1 LT_gain], e_prime_pad_for_d_r);  //= 1 / (1 + 0*z^-1 + 0*z^-2 + ... + b*z^-T)
        e_prime_op(bs:bs+39,1) = e_prime_op_dummy_pad(Topt(bs)+1 : Topt(bs)+1+39);   //pitch-synthesis filter output
    end        //FORMANT SYNTHESIS FILTER:
       e_prime_op_pad_delay_r= [e_prime_op(b : b+159); zeros(1,1)]; //it is padded with zeros to remove the effect of delay in filter
        synth_speech_dummy_pad = filter(1, [1 aCoeff(b+1 : b+8)], e_prime_op_pad_delay_r);
        synth_speech1(b : b+159) = synth_speech_dummy_pad (2:161);   //DE-EMPHASIS (de-proprocessing):
        synth_speech(b : b+159) = filter([1 -0.999], [1 -1], synth_speech1(b : b+159));  //De-processing
end
endfunction

clc;
clear all;
```

```
xdel(winsid());
inpfilenm = "SCI/modules/sound/demos/s1ofwb.wav";
[x,fs,bits] =wavread(inpfilenm);

t=length(x)./fs;// total time t seconds
//COMPRESSION STARTS HERE,
disp('original signal');
sound(x, fs);
[aCoeff, b_LTopt, Topt, e_prime] = f_ENCODER_relp(x, fs);

        // e_prime is instead of position, peak_magitude_index and sample_amplitude_index. (temporarily)
//halt()
//halt('Press a key to play the original sound!')

[synth_speech] = f_DECODER_relp(aCoeff, b_LTopt, Topt, e_prime);

//RESULTS,


disp('compressed signal');
sound(synth_speech, fs);

figure;
subplot(211),
plot(x); title(['Original signal = "', inpfilenm, '"']);
subplot(212), plot(synth_speech); title('RELP compressed output');
//Output plays original signal and after approximately 5 minutes it plays compressed sound and plot the original signal and compressed signal.
```