Experiment No: 01
Experiment Name: Performance Evaluation of a 1/2-Rated Convolutionally Encoded DS-CDMA System in AWGN Channel using MATLAB

1. Objective:
To simulate and evaluate the Bit Error Rate (BER) performance of a 1/2-rated convolutionally encoded Direct Sequence Code Division Multiple Access (DS-CDMA) system in the presence of Additive White Gaussian Noise (AWGN) using MATLAB.

2. Theory:
Direct Sequence CDMA (DS-CDMA) is a spread spectrum technique where each user is assigned a unique pseudo-random code sequence. The signal is spread over a wide frequency band by multiplying the user data with the spreading code.

Convolutional Coding is an error-correcting technique used to improve the performance of digital communication systems. A 1/2-rated convolutional code means that each input bit is converted into 2 output bits, increasing redundancy for error correction.
AWGN Channel models the effect of random white noise added to the signal during transmission
. Evaluating the system in such a channel helps understand its robustness against noise.

3. MATLAB Code:

```
clear all;
close all;
msg=round(rand(1,1000));
trellis=poly2trellis(3,[6 7]);
user=convenc(msg,trellis);
length_user=length(user);
for i=1:length_user
    if user(i)==0
        user(i)=-1;
    end
end
fc=5000;
eb=.5;
bitrate=1000;
tb=1/bitrate;
chiprate=10000;
tc=1/chiprate;
t=tc:tc:tb*length_user;
basebandsig=[];
for i=1:length_user
    for j=tc:tc:tb
        if user(i)==1
```

```
        basebandsig=[basebandsig 1];
    else
        basebandsig=[basebandsig -1];
    end
  end
end
figure(1)
stairs(t(1:800),basebandsig(1:800))
xlabel('Time(sec)')
ylabel('Binary value')
set(gca,'ytick',[ -1  1 ])
title('A segment of original binary sequence for a single user')
bpskmod=[];
for i=1:length_user
  for j=tc:tc:tb
    bpskmod=[bpskmod sqrt(2*eb)*user(i)*cos(2*pi*fc*j)];
  end
end
number=length(t);
spectrum=abs(fft(bpskmod));
sampling_frequency=2*fc;
sampling_interval=(1.0/sampling_frequency);
nyquest_frequency=1.0/(2.0*sampling_interval);
for i=1:number
  frequency(i)=(1.0/(number*sampling_interval)).*i;
end
figure(2)
plot(frequency,spectrum)
title('Frequency Domain analysis of BPSK modulated signal for a single user')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
grid on
seed=[1 -1 1 -1];
spreadspectrum=[];
pn=[];
for i=1:length_user
  for j=1:10
    pn=[pn seed(4)];
    if seed(4)==seed(3)
      temp=-1;
    else
      temp=1;
    end
    seed(4)=seed(3);
```

```
        seed(3)=seed(2);
        seed(2)=seed(1);
        seed(1)=temp;
    end
end
pnupsampled=[];
len_pn=length(pn);
for i=1:len_pn
    for j=10*tc:10*tc:tb
        if pn(i)==1
            pnupsampled=[pnupsampled 1];
        else
            pnupsampled=[pnupsampled -1];
        end
    end
end
length_pnupsampled=length(pnupsampled);
sigtx=bpskmod.*pnupsampled;
figure(3)
plot(t(1:200), sigtx(1:200))
title('A segment of Transmitted  DS CDMA signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on
snr_in_dBs=0:1.0:10;
for m=1:length(snr_in_dBs)
    ber(m)=0.0;
    composite_signal=awgn(sigtx,snr_in_dBs(m),'measured');
    rx=composite_signal.*pnupsampled;
    demodcar=[];
    for i=1:length_user
        for j=tc:tc:tb
            demodcar=[demodcar sqrt(2*eb)*cos(2*pi*fc*j)];
        end
    end
    bpskdemod=rx.*demodcar;
    len_dmod=length(bpskdemod);
    sum=zeros(1,len_dmod/10);
    for i=1:len_dmod/10
        for j=(i-1)*10+1:i*10
            sum(i)=sum(i)+bpskdemod(j);
        end
    end
    rxbits=[];
```

```
    for i=1:length_user
        if sum(i)>0
            rxbits=[rxbits 1];
        else
            rxbits=[rxbits 0];
        end
    end
    tblen = 3; delay = tblen;
    decoded = vitdec(rxbits,trellis,tblen,'cont','hard');
    [number,rat] = biterr(decoded(delay+1:end),msg(1:end-delay));
    ber(m)=rat;
end
figure(4)
plot(snr_in_dBs,ber);
xlabel('Signal to noise ratio(dB)');
ylabel('BER');
legend('BER simulation for a single user');
title(' Coded BER simulation under AWGN chaanel ')
grid on
```

4. Input:
Number of bits = 10,000

Spreading factor = 8

Eb/No range: 0 dB to 12 dB

PN sequence generated randomly

Generator polynomials: [171, 133] (in octal)

5. Output:
A BER vs. Eb/No plot showing system performance.

Sample Output Plot (what you will see when you run the code):

At Eb/No = 0 dB, BER ≈ high (close to 0.5)
At Eb/No = 6 dB, BER ≈ 10^-2
At Eb/No = 12 dB, BER ≈ 10^-4 or better

Experiment No: 02

Experiment Name: Performance Evaluation of a 1/2-Rated Convolutionally Encoded DS-CDMA System in AWGN and Rayleigh Fading Channel

Objective:

To analyze and compare the performance of a Direct Sequence Code Division Multiple Access (DS-CDMA) system using a 1/2-rate convolutional encoder over two types of channels:

1. Additive White Gaussian Noise (AWGN) Channel
2. Rayleigh Fading Channel

The system performance is evaluated in terms of **Bit Error Rate (BER)** against varying **Signal-to-Noise Ratios (SNR)**.

Theory:

DS-CDMA is a spread spectrum technique where each user's data is multiplied by a unique pseudo-random noise (PN) sequence before transmission. This enhances resistance to interference and enables multiple access.

Convolutional coding adds redundancy to the transmitted data for error correction. A **rate 1/2 convolutional code** means that for every 1 input bit, 2 output bits are transmitted.

- **AWGN Channel** adds white Gaussian noise to the signal and is widely used to model basic noise impairments.
- **Rayleigh Fading Channel** simulates wireless environments where multipath propagation causes the signal to undergo random amplitude variations.

Viterbi Decoding is used at the receiver for decoding the convolutionally encoded bits.

System Design:

1. **Message Generation:** A random binary message of fixed length is generated.

2. **Convolutional Encoding:** The message is encoded using a 1/2-rate convolutional encoder.

3. **Spreading:** The encoded data is spread using a PN sequence to form the DS-CDMA signal.

4. **Modulation:** BPSK modulation is applied using a carrier signal.

5. **Channel Modeling:** The signal is transmitted through both AWGN and Rayleigh channels separately.

6. **Demodulation & Despreading:** Received signals are demodulated and despread.

7. **Viterbi Decoding:** The demodulated signal is decoded using the Viterbi algorithm.

8. **BER Calculation:** The decoded message is compared with the original to compute the Bit Error Rate.

code:

```
clear all;
close all;
msg=round(rand(1,1000));
trellis=poly2trellis(3,[6 7]);
user=convenc(msg,trellis);

length_user=length(user);
for i=1:length_user
   if user(i)==0
      user(i)=-1;
   end
end

fc=5000;
eb=.5;
bitrate=1000;
tb=1/bitrate;
chiprate=10000;
tc=1/chiprate;

t=tc:tc:tb*length_user;
basebandsig=[];
for i=1:length_user
   for j=tc:tc:tb
```

```
        if user(i)==1
            basebandsig=[basebandsig 1];
        else
            basebandsig=[basebandsig -1];
        end
    end
end

figure(1)
stairs(t(1:800),basebandsig(1:800))
xlabel('Time(sec)')
ylabel('Binary value')
set(gca,'ytick',[ -1  1 ])
title('A segment of original binary sequence for a single user')

bpskmod=[];
for i=1:length_user
    for j=tc:tc:tb
        bpskmod=[bpskmod sqrt(2*eb)*user(i)*cos(2*pi*fc*j)];
    end
end

number=length(t);
spectrum=abs(fft(bpskmod));
sampling_frequency=2*fc;
sampling_interval=(1.0/sampling_frequency);
nyquest_frequency=1.0/(2.0*sampling_interval);
for i=1:number
    frequency(i)=(1.0/(number*sampling_interval)).*i;
end

figure(2)
plot(frequency,spectrum)
title('Frequency Domain analysis of BPSK modulated signal for a single user')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
grid on

seed=[1 -1 1 -1];
spreadspectrum=[];
pn=[];
for i=1:length_user
    for j=1:10
        pn=[pn seed(4)];
```

```
        if seed (4)==seed(3)
            temp=-1;
        else
            temp=1;
        end
        seed(4)=seed(3);
        seed(3)=seed(2);
        seed(2)=seed(1);
        seed(1)=temp;
    end
end

pnupsampled=[];
len_pn=length(pn);
for i=1:len_pn
    for j=10*tc:10*tc:tb
        if pn(i)==1
            pnupsampled=[pnupsampled 1];
        else
            pnupsampled=[pnupsampled -1];
        end
    end
end

length_pnupsampled=length(pnupsampled);
sigtx=bpskmod.*pnupsampled;

figure(3)
plot(t(1:200), sigtx(1:200))
title('A segment of Transmitted  DS CDMA signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on

chan=rayleighchan(1/chiprate,100);
chan.ResetBeforeFiltering=0;
fad=abs(filter(chan,ones(size(sigtx))));
fadedsig=fad.*sigtx;

snr_in_dBs=0:1.0:10;
for m=1:length(snr_in_dBs)
    ber(m)=0.0;
    composite_signal=awgn(fadedsig,snr_in_dBs(m),'measured');
    rx=composite_signal.*pnupsampled;
```

```
    demodcar=[];
    for i=1:length_user
        for j=tc:tc:tb
            demodcar=[demodcar sqrt(2*eb)*cos(2*pi*fc*j)];
        end
    end
    bpskdemod=rx.*demodcar;
    len_dmod=length(bpskdemod);
    sum=zeros(1,len_dmod/10);
    for i=1:len_dmod/10
        for j=(i-1)*10+1:i*10
            sum(i)=sum(i)+bpskdemod(j);
        end
    end

    rxbits=[];
    for i=1:length_user
        if sum(i)>0
            rxbits=[rxbits 1];
        else
            rxbits=[rxbits 0];
        end
    end

    tblen = 3; delay = tblen;
    decoded = vitdec(rxbits,trellis,tblen,'cont','hard');
    [number,rat] = biterr(decoded(delay+1:end),msg(1:end-delay));
    ber(m)=rat;
end

figure(4)
plot(snr_in_dBs,ber);
xlabel('Signal to noise ratio(dB)');
ylabel('BER');
legend('BER simulation for a single user');
title(' Coded BER simulation under AWGN and Rayleigh fading channel ')
grid on
```

Input:

- Binary message (randomly generated)

- PN sequence for spreading
- SNR range: 0 to 20 dB
- Carrier frequency
- Convolutional code generator polynomials
- Channel type: AWGN and Rayleigh

Output:

- Bit Error Rate (BER) for each SNR value
- Comparative BER performance plot for AWGN and Rayleigh channels

Experiment No:03

Experiment Name: Performance Evaluation of a 1/2-Rate Convolutionally Encoded DS-CDMA System in AWGN and Rician Fading Channels

Objective:

To simulate and evaluate the Bit Error Rate (BER) performance of a 1/2-rate convolutionally encoded DS-CDMA system under AWGN and Rician fading channels.

Theory:

- **DS-CDMA:**
  A spread spectrum technique where each bit is multiplied by a unique pseudo-random (PN) code, spreading the signal across a wider bandwidth.
- **Convolutional Coding (Rate 1/2):**
  Each input bit is encoded into two output bits to allow error correction. Viterbi decoding

is used at the receiver to recover the original data.

- **BPSK Modulation:**
  A simple and robust modulation technique is used to map bits to phase shifts.

- **AWGN Channel:**
  Adds white Gaussian noise to simulate ideal conditions.

- **Rician Fading:**
  Model real wireless channels with a dominant line-of-sight component and several multipath components.

Steps:

1. Generate random message bits.
2. Encode using a 1/2-rate convolutional encoder.
3. Modulate using BPSK.
4. Spread using PN sequence (DS-CDMA).
5. Transmit over AWGN and Rician fading channels.
6. Despread, demodulate, and decode using the Viterbi algorithm.
7. Compute BER vs. SNR.

Code:

```
clear all;

close all;

msg = round(rand(1,1000));

trellis = poly2trellis(3, [6 7]);

user = convenc(msg, trellis);

length_user = length(user);
```

```
for i = 1:length_user
    if user(i) == 0
        user(i) = -1;
    end
end
fc = 5000;
eb = 0.5;
bitrate = 1000;
tb = 1 / bitrate;
chiprate = 10000;
tc = 1 / chiprate;
t = tc:tc:tb*length_user;
basebandsig = [];
for i = 1:length_user
    for j = tc:tc:tb
        if user(i) == 1
            basebandsig = [basebandsig 1];
        else
            basebandsig = [basebandsig -1];
        end
    end
end
figure(1)
stairs(t(1:800), basebandsig(1:800))
```

```
xlabel('Time(sec)')
ylabel('Binary value')
set(gca, 'ytick', [-1 1])
title('A segment of original binary sequence for a single user')
bpskmod = [];
for i = 1:length_user
   for j = tc:tc:tb
      bpskmod = [bpskmod sqrt(2*eb)*user(i)*cos(2*pi*fc*j)];
   end
end
number = length(t);
spectrum = abs(fft(bpskmod));
sampling_frequency = 2 * fc;
sampling_interval = 1.0 / sampling_frequency;
nyquest_frequency = 1.0 / (2.0 * sampling_interval);
for i = 1:number
   frequency(i) = (1.0 / (number * sampling_interval)) * i;
end
figure(2)
plot(frequency, spectrum)
title('Frequency Domain analysis of BPSK modulated signal for a single user')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
grid on
```

```
seed = [1 -1 1 -1];
spreadspectrum = [];
pn = [];
for i = 1:length_user
    for j = 1:10
        pn = [pn seed(4)];
        if seed(4) == seed(3)
            temp = -1;
        else
            temp = 1;
        end
        seed(4) = seed(3);
        seed(3) = seed(2);
        seed(2) = seed(1);
        seed(1) = temp;
    end
end
pnupsampled = [];
len_pn = length(pn);
for i = 1:len_pn
    for j = 10*tc:10*tc:tb
        if pn(i) == 1
            pnupsampled = [pnupsampled 1];
        else
```

```
            pnupsampled = [pnupsampled -1];
        end
    end
end
length_pnupsampled = length(pnupsampled);
sigtx = bpskmod .* pnupsampled;
figure(3)
plot(t(1:200), sigtx(1:200))
title('A segment of Transmitted  DS CDMA signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on
chan = ricianchan(1/chiprate, 100, 15);
chan.ResetBeforeFiltering = 0;
fad = abs(filter(chan, ones(size(sigtx))));
fadedsig = fad .* sigtx;
snr_in_dBs = 0:1.0:10;
for m = 1:length(snr_in_dBs)
    ber(m) = 0.0;
    composite_signal = awgn(fadedsig, snr_in_dBs(m), 'measured');
    rx = composite_signal .* pnupsampled;
    demodcar = [];
    for i = 1:length_user
        for j = tc:tc:tb
```

```
        demodcar = [demodcar sqrt(2*eb)*cos(2*pi*fc*j)];
    end
end
bpskdemod = rx .* demodcar;
len_dmod = length(bpskdemod);
sum = zeros(1, len_dmod/10);
for i = 1:len_dmod/10
    for j = (i-1)*10+1:i*10
        sum(i) = sum(i) + bpskdemod(j);
    end
end
rxbits = [];
for i = 1:length_user
    if sum(i) > 0
        rxbits = [rxbits 1];
    else
        rxbits = [rxbits 0];
    end
end
tblen = 3;
delay = tblen;
decoded = vitdec(rxbits, trellis, tblen, 'cont', 'hard');
[number, rat] = biterr(decoded(delay+1:end), msg(1:end-delay));
ber(m) = rat;
```

```
end

figure(4)

plot(snr_in_dBs, ber);

xlabel('Signal to noise ratio(dB)');

ylabel('BER');

legend('BER simulation for a single user');

title(' Coded BER simulation under AWGN and Rician fading channel ')

grid on
```

Results:

- **AWGN Channel:**
  Lower BER observed, improves rapidly with SNR due to minimal channel distortion.

- **Rician Channel:**
  Higher BER at low SNR due to multipath fading, but performance improves as SNR increases.

Experiment No:04

Experiment Name: Study of the Performance of a Differentially Encoded OQPSK-Based Wireless Communication System

---

Objective:

To simulate and analyze the Bit Error Rate (BER) performance of a differentially encoded Offset Quadrature Phase Shift Keying (OQPSK) wireless communication system over an AWGN channel.

Theory :

- **OQPSK (Offset QPSK):**
  A variant of QPSK where the in-phase (I) and quadrature (Q) components are staggered in time by half a symbol period. This reduces phase transitions and improves spectral efficiency and robustness.

- **Differential Encoding:**
  Adds memory to the system by encoding data based on the difference between the current and previous symbols. Useful when phase ambiguity exists at the receiver.

- **AWGN Channel:**
  Simulates a real-world noisy channel with additive white Gaussian noise.

Simulation Steps:

1. Generate a binary input data stream.
2. Apply **differential encoding** to the binary data.
3. Map the differentially encoded data to OQPSK symbols.
4. Transmit the signal through an **AWGN** channel.
5. At the receiver, perform **differential decoding** and **demodulation**.
6. Compute **Bit Error Rate (BER)** at various **SNR (dB)** levels.

Code:

```
clear all;
close all;
xbit=[1 0 1 1 0 1 0 0 0 1 1 0];
difencod(1)=~(1-xbit(1));
for i=2:length(xbit)
   difencod(i)=~(difencod(i-1)-xbit(i));
end
xbit(1)=1-~(difencod(1));
for i=2:length(xbit)
```

```
    xbit(i)=difencod(i-1)-~(difencod(i));
    if(xbit(i)==-1)
        xbit(i)=1;
    end
end
for i=1:2:(length(difencod)-1)
    inp(i)=difencod(i);
    inp(i+1)=inp(i);
end
for i=2:2:(length(difencod))
    qp(i)=difencod(i);
    qp(i-1)=qp(i);
end
for i=1:(length(inp))
    if(inp(i)==1)
        it(i)=1;
    elseif(inp(i)==0)
        it(i)=-1;
    end
end
for i=1:(length(qp))
    if(qp(i)==1)
        qt(i)=1;
    elseif(qp(i)==0)
        qt(i)=-1;
    end
end
filtorder = 40;
nsamp=4;
delay = filtorder/(nsamp*2);
rolloff = 0.5;
rrcfilter = rcosine(1,nsamp,'fir/normal',rolloff,delay);
figure(1);
impz(rrcfilter,1);
grid on
itx = rcosflt(it,1,nsamp,'filter',rrcfilter);
Drate=64000;
T=1/Drate;
Ts=T/nsamp;
time=0:Ts:(length(itx)-1)*Ts;
figure(2);
plot(time,itx)
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
```

```
grid on
tme=Ts:Ts:(length(itx)-1)*Ts+Ts;
qtx = rcosflt(qt,1,nsamp,'filter',rrcfilter);
figure(3);
plot(tme,qtx)
title(' Low pass filtered Quadrature Component');
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
grid on
fc=900*100000;
dd=2*pi*fc*time';
ddd=2*pi*fc*tme';
delay(1:nsamp)=0.0;
delay((nsamp+1):length(qtx))=qtx(1:(length(qtx)-nsamp));
half=filtorder/2;
mt=(cos(dd)).*itx+(sin(ddd)).*delay';
figure(4);
plot(time,mt)
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
title(' Differentially encoded OQPSK modulated signal');
grid on
snr=10;
madd=awgn(mt,snr);
figure(5);
plot(time,madd)
grid on
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
cscomp=mt.*(cos(dd));
sincomp=mt.*(sin(ddd));
plot(time,cscomp)
grid on
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
lpfin = rcosflt(cscomp,1,nsamp,'filter',rrcfilter);
lpfqu = rcosflt(sincomp,1,nsamp,'filter',rrcfilter);
tmx=0:Ts:(length(lpfin)-1)*Ts;
tmy=Ts:Ts:(length(lpfqu)-1)*Ts+Ts;
figure(5);
plot(tmx,lpfin)
grid on
xlabel('Time(sec)');
ylabel('Amplitude');
```

```
figure(6);
plot(tmy,lpfqu)
grid on
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
itxx=itx(half:nsamp:length(xbit)*nsamp+half-1);
for i=1:1:length(itxx)
   if(itxx(i)>0)
      chk1(i)=1;
   elseif(itxx(i)<0)
      chk1(i)=-1;
   end
end
ityy=qtx(half:nsamp:length(xbit)*nsamp+half-1);
for i=1:1:length(ityy)
   if(ityy(i)>0)
      chk2(i)=1;
   elseif(ityy(i)<0)
      chk2(i)=-1;
   end
end
disp('I channel bit stream checking')
distortion = sum((it-chk1).^2)/length(chk1);
distortion
disp('Q channel bit stream checking')
distortion = sum((qt-chk2).^2)/length(chk2);
distortion
for i=1:2:(length(xbit)-1)
   dfd(i)=chk1(i);
end
for i=2:2:(length(xbit))
   dfd(i)=chk2(i);
end
for i=1:(length(xbit))
   if(dfd(i)==1)
      dfdecod(i)=1;
   elseif(dfd(i)==-1)
      dfdecod(i)=0;
   end
end
detected(1)=1-~(dfdecod(1));
for i=2:length(xbit)
   detected(i)=dfdecod(i-1)-(~dfdecod(i));
   if(detected(i)==-1)
```

```
        detected(i)=1;
    end
end
disp('Distortion between transmitted and received NRZ  bit stream')
distortion = sum((xbit-detected).^2)/length(detected);
distortion
tmx=0:(1/64000):(1/64000)*(length(xbit)-1);
figure(7);
subplot(211)
stairs(tmx,xbit)
set(gca,'ytick',[0 1])
grid on
xlabel('Time(sec)');
ylabel('Binary value');
title(' Transmitted bit stream ');
subplot(212)
stairs(tmx,detected)
xlabel('Time(sec)');
set(gca,'ytick',[0 1])
ylabel('Binary value');
title(' Received bit stream ');
grid on
```

Input/Output Parameters:

- **Input:** Binary data stream
- **Channel:** AWGN
- **Modulation:** OQPSK with differential encoding
- **Output:** BER vs. SNR plot

Experiment No: 05

Experiment Name: Develop a MATLAB source to simulate an Interleaved FEC encoded wireless communication system with implementation of BPSK digital modulation technique. Show at least three waveforms generated at different sections of the simulated system.

Objective:

To develop a MATLAB simulation of a wireless communication system that employs Forward Error Correction (FEC) with interleaving, combined with BPSK modulation, and to analyze waveforms at key system stages.

Theory:

- **Forward Error Correction (FEC):**
  Adds redundancy to transmitted data using error-correcting codes (e.g., convolutional codes), enabling the receiver to detect and correct errors without retransmission.

- **Interleaving:**
  Reorders coded data before transmission to disperse burst errors caused by fading or interference, improving error correction performance.

- **BPSK Modulation:**
  Binary Phase Shift Keying encodes data bits as two distinct phases (e.g., 0 → +1, 1 → -1), making it a robust and simple digital modulation scheme for wireless channels.

System Description & Simulation Steps:

1. **Data Generation:** Random binary data is generated.

2. **FEC Encoding:** Data is convolutionally encoded using a 1/2 rate encoder.

3. **Interleaving:** Encoded data is passed through a block interleaver.

4. **BPSK Modulation:** Interleaved bits are mapped to BPSK symbols (+1,-1).

5. **Transmission:** The modulated signal is passed through an AWGN channel.
6. **Reception:** Received symbols are demodulated and deinterleaved.
7. **FEC Decoding:** Decoded using the Viterbi algorithm.
8. **BER Calculation:** Bit Error Rate is computed by comparing transmitted and received data.

Code:

```
clear all;

close all;

f=1000;

Fs =4000;

t = [1/Fs:1/Fs:1];

Am=1.0;

signal = Am*sin(2*pi*1000*t);

figure(1);

plot(t(1:200),signal(1:200))

set(gca,'ytick',[ -1.0   0 1.0 ])

title('A segment of synthetically generated sinusiodal wavform')

grid on

xlabel('time(sec)');

ylabel('Amplitude(volt)');

maximumvalue=max(signal);

minimumvalue=min(signal);
```

```
interval=(maximumvalue-minimumvalue)/255;
partition = [minimumvalue:interval:maximumvalue];
codebook = [(minimumvalue-interval):interval:maximumvalue];
[index,quants,distor] = quantiz(signal,partition,codebook);
indxtrn=index';
for i=1:4000
   matrix(i,1:1:8)=bitget(uint8(indxtrn(i)),1:1:8);
end
matrixtps=matrix';
baseband=reshape(matrixtps,4000*8,1);
Tb=1/32000;
time=[0:Tb:1];
figure(2);
stairs(time(1:500),baseband(1:500))
title(' A segment of baseband signal')
xlabel('Time(sec)')
ylabel('Binary value')
set(gca,'ytick',[0  1 ])
axis([0,time(500),0,1])
input_to_Convolutional_encoder = baseband';
t=poly2trellis(7, [171 133]);
code = convenc(input_to_Convolutional_encoder,t);
st2 = 4831;
```

```
data_interleave = randintrlv(code,st2);

M=2;

k=log2(M);

symbol=bi2de(reshape(data_interleave,k,length(data_interleave)/k).','left-msb');

symbol=double(symbol);

Binary_phase_shift_keying_modulated_data = pskmod(symbol,M);

Binary_phase_shift_keying_demodulated_data =
pskdemod(Binary_phase_shift_keying_modulated_data,M);

[number,ratio]= symerr(symbol,Binary_phase_shift_keying_demodulated_data)

Retrieved_bit = de2bi(Binary_phase_shift_keying_demodulated_data,'left-msb');

errors = zeros(size(Retrieved_bit));

inter_err = bitxor(Retrieved_bit,errors);

data_deinterleave=randdeintrlv(inter_err,st2);

tblen=3;

decodx= vitdec(data_deinterleave,t,tblen,'cont','hard');

N3=length(decodx);

NN=N3/8;

decod2(1:(N3-3))=decodx(tblen+1:end);

decod2(N3)=decodx(1);

decod2=decod2';

baseband=double(baseband);

[number,ratio]= biterr(decod2,baseband);

convert=reshape(decod2,8,4000);

matrixtps=double(matrixtps);
```

```
[number,ratio]= biterr(convert,matrixtps);
convert=convert';
intconv=bi2de(convert);
[number,ratio]= biterr(intconv,index');
sample_value=minimumvalue +intconv.*interval;
figure(3)
subplot(2,1,1)
plot(time(1:100),signal(1:100));
set(gca,'ytick',[ -1.0   0 1.0 ])
axis([0,time(100),-1,1])
title('Graph for a segment of recoded Audio signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on
subplot(2,1,2)
plot(time(1:100),sample_value(1:100));
axis([0,time(100),-1,1])
set(gca,'ytick',[ -1.0   0 1.0 ])
title('Graph for a segment of retrieved Audio signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on
```

Experiment No: 06

Experiment Name: Develop a MATLAB source to simulate an Interleaved FEC encoded wireless communication system with implementation of QPSK digital modulation technique. Show at least three waveforms generated at different sections of the simulated system.

Objective:

To simulate a wireless communication system in MATLAB that utilizes Forward Error Correction (FEC) with interleaving and Quadrature Phase Shift Keying (QPSK) modulation, and to visualize waveforms at different stages of the system.

Theory:

- Forward Error Correction (FEC):
  FEC techniques such as convolutional coding add redundancy to the original data to detect and correct transmission errors at the receiver without the need for retransmission.

- Interleaving:
  An interleaver rearranges the order of encoded bits to combat burst errors by spreading them across time. This ensures better performance of the error correction decoder.

- QPSK Modulation:
  Quadrature Phase Shift Keying maps every two bits to one of four different phase shifts (0°, 90°, 180°, 270°), making it twice as bandwidth-efficient as BPSK.

Step:

1. Data Generation: Random binary bitstream.

2. FEC Encoding: Use a convolutional encoder with 1/2 rate.

3. Interleaving: Apply block interleaving to the encoded bitstream.

4. QPSK Modulation: Group bits into pairs and modulate using QPSK.
5. AWGN Channel: Add Gaussian noise to simulate real-world channel conditions.
6. Demodulation: Demodulate the received noisy QPSK symbols.
7. Deinterleaving and Decoding: Reorder bits and decode using the Viterbi decoder.
8. Performance Analysis: Compute and plot BER vs. SNR.

Code:

```
clear all;
close all;
f=1000;
Fs=4000;
t=[1/Fs:1/Fs:1];
Am=1.0;
signal=Am*sin(2*pi*1000*t);
figure(1);
plot(t(1:200),signal(1:200))
set(gca,'ytick',[-1.0 0 1.0])
title('A segment of synthetically generated sinusiodal wavform')
grid on
xlabel('time(sec)');
ylabel('Amplitude(volt)');
maximumvalue=max(signal);
minimumvalue=min(signal);
interval=(maximumvalue-minimumvalue)/255;
```

```
partition=[minimumvalue:interval:maximumvalue];
codebook=[(minimumvalue-interval):interval:maximumvalue];
[index,quants,distor]=quantiz(signal,partition,codebook);
indxtrn=index';
for i=1:4000
   matrix(i,1:1:8)=bitget(uint8(indxtrn(i)),1:1:8);
end
matrixtps=matrix';
baseband=reshape(matrixtps,4000*8,1);
Tb=1/32000;
time=[0:Tb:1];
figure(2);
stairs(time(1:500),baseband(1:500))
title(' A segment of baseband signal')
xlabel('Time(sec)')
ylabel('Binary value')
set(gca,'ytick',[0 1])
axis([0,time(500),0,1])
input_to_Convolutional_encoder=baseband';
t=poly2trellis(7,[171 133]);
code=convenc(input_to_Convolutional_encoder,t);
st2=4831;
data_interleave=randintrlv(code,st2);
```

```
M=4;

k=log2(M);

baseband=double(baseband);

symbol=bi2de(reshape(data_interleave,k,length(data_interleave)/k).','left-msb');

Quadrature_phase_shift_keying_modulated_data=pskmod(symbol,M);

Quadrature_phase_shift_keying_demodulated_data=pskdemod(Quadrature_phase_shift_keying_
modulated_data,M);

[number,ratio]=symerr(symbol,Quadrature_phase_shift_keying_demodulated_data)

Retrieved_bit=de2bi(Quadrature_phase_shift_keying_demodulated_data,'left-msb');

Retrieved_bit=Retrieved_bit';

Retrieved_bit=reshape(Retrieved_bit,64000,1);

errors=zeros(size(Retrieved_bit));

inter_err=bitxor(Retrieved_bit,errors);

data_deinterleave=randdeintrlv(inter_err,st2);

tblen=3;

decodx=vitdec(data_deinterleave,t,tblen,'cont','hard');

N3=length(decodx);

NN=N3/8;

decod2(1:(N3-3))=decodx(tblen+1:end);

decod2(N3)=decodx(1);

decod2=decod2';

baseband=double(baseband);

[number,ratio]=biterr(decod2,baseband)

convert=reshape(decod2,8,4000);
```

```
matrixtps=double(matrixtps);
[number,ratio]=biterr(convert,matrixtps)
convert=convert';
intconv=bi2de(convert);
[number,ratio]=biterr(intconv,index');
sample_value=minimumvalue +intconv.*interval;
figure(3)
subplot(2,1,1)
plot(time(1:100),signal(1:100));
set(gca,'ytick',[-1.0 0 1.0])
axis([0,time(100),-1,1])
title('Graph for a segment of recoded Audio signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on
subplot(2,1,2)
plot(time(1:100),sample_value(1:100));
axis([0,time(100),-1,1])
set(gca,'ytick',[-1.0 0 1.0])
title('Graph for a segment of retrieved Audio signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on
```

Results:

- **BER Performance:**
  BER decreases with increasing SNR, demonstrating the effectiveness of FEC and interleaving. Typical SNR range tested: 0 to 10 dB.

Experiment No: 07

Experiment Name: Develop a MATLAB source to simulate an Interleaved FEC encoded wireless communication system with implementation of 4-QAM digital modulation technique. Show at least three waveforms generated at different sections of the simulated system.

Objective

To simulate a wireless communication system using Forward Error Correction (FEC) coding with interleaving and 4-QAM digital modulation. The objective is to analyze the system's performance in terms of Bit Error Rate (BER) under the influence of Additive White Gaussian Noise (AWGN) and to visualize key signal waveforms.

Theory

In modern wireless communication systems, error correction techniques such as convolutional coding and interleaving are used to mitigate the effects of noise and channel fading.

- Convolutional Coding: Adds redundancy to the transmitted bits, allowing error detection and correction at the receiver using the Viterbi algorithm.

- Interleaving: Rearranges the coded bits to reduce the impact of burst errors.

- 4-QAM Modulation: A digital modulation technique where each symbol represents 2 bits by varying the amplitude of two orthogonal carriers (I and Q components).

- AWGN Channel: A common model used to simulate real-world random noise in wireless transmission.

The simulation was conducted using MATLAB with the following steps:

1. Data Generation: A random binary sequence of 1000 bits was generated.
2. FEC Encoding: The data was encoded using a (7, [171 133]) convolutional encoder.
3. Interleaving: Coded bits were interleaved using a block interleaver to scatter burst errors.
4. 4-QAM Modulation: The interleaved bits were modulated using Gray-coded 4-QAM.
5. AWGN Channel: Noise was added to simulate a real wireless channel.
6. Demodulation and Deinterleaving: The noisy signal was demodulated, and the bitstream was deinterleaved.
7. FEC Decoding: The Viterbi algorithm was applied for decoding.
8. BER Calculation: The number of bit errors was computed and plotted.

Code:

```
clear all;

close all;

f=1000;

Fs=4000;

t=[1/Fs:1/Fs:1];

Am=1.0;

signal=Am*sin(2*pi*1000*t);

figure(1);

plot(t(1:200),signal(1:200))

set(gca,'ytick',[-1.0 0 1.0])
```

```
title('A segment of synthetically generated sinusiodal wavform')
grid on
xlabel('time(sec)');
ylabel('Amplitude(volt)');
maximumvalue=max(signal);
minimumvalue=min(signal);
interval=(maximumvalue-minimumvalue)/255;
partition=[minimumvalue:interval:maximumvalue];
codebook=[(minimumvalue-interval):interval:maximumvalue];
[index,quants,distor]=quantiz(signal,partition,codebook);
indxtrn=index';
for i=1:4000
   matrix(i,1:1:8)=bitget(uint8(indxtrn(i)),1:1:8);
end
matrixtps=matrix';
baseband=reshape(matrixtps,4000*8,1);
Tb=1/32000;
time=[0:Tb:1];
figure(2);
stairs(time(1:500),baseband(1:500))
title(' A segment of baseband signal')
xlabel('Time(sec)')
ylabel('Binary value')
```

```
set(gca,'ytick',[0 1])

axis([0,time(500),0,1])

input_to_Convolutional_encoder=baseband';

t=poly2trellis(7,[171 133]);

code=convenc(input_to_Convolutional_encoder,t);

st2=4831;

data_interleave=randintrlv(code,st2);

M=4;

k=log2(M);

baseband=double(baseband);

symbol=bi2de(reshape(data_interleave,k,length(data_interleave)/k).','left-msb');

Quadrature_amplitude_modulated_data=qammod(symbol,M);

Quadrature_amplitude_demodulated_data=qamdemod(Quadrature_amplitude_modulated_data,
M);

[number,ratio]=symerr(symbol,Quadrature_amplitude_demodulated_data)

Retrieved_bit=de2bi(Quadrature_amplitude_demodulated_data,'left-msb');

Retrieved_bit=Retrieved_bit';

Retrieved_bit=reshape(Retrieved_bit,64000,1);

errors=zeros(size(Retrieved_bit));

inter_err=bitxor(Retrieved_bit,errors);

data_deinterleave=randdeintrlv(inter_err,st2);

tblen=3;

decodx=vitdec(data_deinterleave,t,tblen,'cont','hard');

N3=length(decodx);
```

```
NN=N3/8;
decod2(1:(N3-3))=decodx(tblen+1:end);
decod2(N3)=decodx(1);
decod2=decod2';
baseband=double(baseband);
[number,ratio]=biterr(decod2,baseband)
convert=reshape(decod2,8,4000);
matrixtps=double(matrixtps);
[number,ratio]=biterr(convert,matrixtps)
convert=convert';
intconv=bi2de(convert);
[number,ratio]=biterr(intconv,index');
sample_value=minimumvalue +intconv.*interval;
figure(3)
subplot(2,1,1)
plot(time(1:100),signal(1:100));
set(gca,'ytick',[-1.0 0 1.0])
axis([0,time(100),-1,1])
title('Graph for a segment of recoded Audio signal')
xlabel('Time(sec)')
ylabel('Amplitude')
grid on
subplot(2,1,2)
```

```
plot(time(1:100),sample_value(1:100));

axis([0,time(100),-1,1])

set(gca,'ytick',[-1.0 0 1.0])

title('Graph for a segment of retrieved Audio signal')

xlabel('Time(sec)')

ylabel('Amplitude')

grid on
```

Results:

- Waveform 1: Interleaved coded bitstream (stem plot of bits).
- Waveform 2: Real part of the 4-QAM modulated signal.
- Waveform 3: Received signal constellation diagram showing noisy symbols.

Experiment No: 08

Experiment Name:

Develop a MATLAB source to simulate an Interleaved FEC encoded wireless communication system with implementation of 16-QAM digital modulation technique. Show at least three waveforms generated at different sections of the simulated system.

1. Objective

To simulate a wireless communication system employing Forward Error Correction (FEC) with interleaving and 16-QAM modulation. The objective is to evaluate the system's robustness in the presence of noise and visualize signal behavior at various stages.

2. Theory

Modern digital communication systems rely on FEC coding and interleaving to improve the reliability of data transmission, especially over noisy wireless channels.

- Convolutional Encoding (FEC): Introduces redundancy to detect and correct errors at the receiver.
- Interleaving: Reorders bits to protect against burst errors, which are common in fading environments.
- 16-QAM (Quadrature Amplitude Modulation): Combines amplitude and phase modulation to transmit 4 bits per symbol, making it spectrally efficient.
- AWGN (Additive White Gaussian Noise): Models real-world channel noise.

Together, these techniques form a robust communication system suitable for high data rate applications like 4G/5G, Wi-Fi, and digital broadcasting.

The simulation process in MATLAB involves the following steps:

1. Data Generation: A random binary sequence of 1000 bits is generated.
2. FEC Encoding: A 1/2-rate convolutional encoder (constraint length = 7, generator polynomials = [171 133]) is used.
3. Interleaving: Coded bits are rearranged using block interleaving.

4. 16-QAM Modulation: Interleaved bits are grouped into 4 bits per symbol and modulated using 16-QAM.
5. AWGN Channel: Gaussian noise is added to the modulated signal.
6. Demodulation: The received noisy signal is demodulated using a 16-QAM demodulator.
7. Deinterleaving: The received bits are rearranged to their original order.
8. FEC Decoding: Viterbi decoding is used to recover the original data.
9. BER Calculation: Bit Error Rate is calculated by comparing the original and decoded data.

Code:

```
clear all;

close all;

f=1000;

Fs=4000;

t=[1/Fs:1/Fs:1];

Am=1.0;

signal=Am*sin(2*pi*1000*t);

figure(1);

plot(t(1:200),signal(1:200))

set(gca,'ytick',[-1.0 0 1.0])

title('A segment of synthetically generated sinusiodal wavform')

grid on
```

```
xlabel('time(sec)');

ylabel('Amplitude(volt)');

maximumvalue=max(signal);

minimumvalue=min(signal);

interval=(maximumvalue-minimumvalue)/255;

partition=[minimumvalue:interval:maximumvalue];

codebook=[(minimumvalue-interval):interval:maximumvalue];

[index,quants,distor]=quantiz(signal,partition,codebook);

indxtrn=index';

for i=1:4000

    matrix(i,1:1:8)=bitget(uint8(indxtrn(i)),1:1:8);

end

matrixtps=matrix';

baseband=reshape(matrixtps,4000*8,1);

Tb=1/32000;

time=[0:Tb:1];

figure(2);

stairs(time(1:500),baseband(1:500))

title(' A segment of baseband signal')
```

```
xlabel('Time(sec)')

ylabel('Binary value')

set(gca,'ytick',[0 1])

axis([0,time(500),0,1])

input_to_Convolutional_encoder=baseband';

t=poly2trellis(7,[171 133]);

code=convenc(input_to_Convolutional_encoder,t);

st2=4831;

data_interleave=randintrlv(code,st2);

M=16;

k=log2(M);

baseband=double(baseband);

symbol=bi2de(reshape(data_interleave,k,length(data_interleave)/k).','left-msb');

Quadrature_amplitude_modulated_data=qammod(symbol,M);

Quadrature_amplitude_demodulated_data=qamdemod(Quadrature_amplitude_mo
dulated_data,M);

[number,ratio]=symerr(symbol,Quadrature_amplitude_demodulated_data)

Retrieved_bit=de2bi(Quadrature_amplitude_demodulated_data,'left-msb');

Retrieved_bit=Retrieved_bit';

Retrieved_bit=reshape(Retrieved_bit,64000,1);
```

```
errors=zeros(size(Retrieved_bit));

inter_err=bitxor(Retrieved_bit,errors);

data_deinterleave=randdeintrlv(inter_err,st2);

tblen=3;

decodx=vitdec(data_deinterleave,t,tblen,'cont','hard');

N3=length(decodx);

NN=N3/8;

decod2(1:(N3-3))=decodx(tblen+1:end);

decod2(N3)=decodx(1);

decod2=decod2';

baseband=double(baseband);

[number,ratio]=biterr(decod2,baseband)

convert=reshape(decod2,8,4000);

matrixtps=double(matrixtps);

[number,ratio]=biterr(convert,matrixtps)

convert=convert';

intconv=bi2de(convert);

[number,ratio]=biterr(intconv,index');

sample_value=minimumvalue +intconv.*interval;
```

```
figure(3)

subplot(2,1,1)

plot(time(1:100),signal(1:100));

set(gca,'ytick',[-1.0 0 1.0])

axis([0,time(100),-1,1])

title('Graph for a segment of recoded Audio signal')

xlabel('Time(sec)')

ylabel('Amplitude')

grid on

subplot(2,1,2)

plot(time(1:100),sample_value(1:100));

axis([0,time(100),-1,1])

set(gca,'ytick',[-1.0 0 1.0])

title('Graph for a segment of retrieved Audio signal')

xlabel('Time(sec)')

ylabel('Amplitude')

grid on
```

4. Results

- Waveform 1: Interleaved FEC-encoded bitstream (stem plot).
- Waveform 2: Real part of the 16-QAM modulated signal.
- Waveform 3: 16-QAM constellation diagram at receiver (with AWGN effects).