1. **Experiment Name:** Write a MATLAB program to evaluate performance of a 1/2-rated convolutionally encoded DS CDMA system in AWGN channel.

Theory:

Direct Sequence Code Division Multiple Access (DS-CDMA) is a multiple-access technique used in wireless communication systems, allowing multiple users to share the same frequency band by assigning unique spreading codes to each user. The system spreads the signal across a wider bandwidth using a pseudo-random noise (PN) sequence, which enhances resistance to interference and improves signal detection in noisy environments. In this experiment, we evaluate the performance of a DS-CDMA system that employs a 1/2-rate convolutional code for error correction and Binary Phase Shift Keying (BPSK) modulation over an Additive White Gaussian Noise (AWGN) channel. The performance metric is the Bit Error Rate (BER), which measures the probability of bit errors at the receiver as a function of the signal-to-noise ratio (SNR).

Convolutional Coding: Convolutional coding is a forward error correction technique that adds redundancy to the transmitted data to improve reliability. A 1/2-rate convolutional encoder produces two output bits for each input bit, effectively doubling the data length. The encoder uses a trellis structure defined by generator polynomials (in this case, 6 (binary 110) and 7 (binary 111)) with a memory order of 3. The encoded bits are decoded at the receiver using the Viterbi algorithm, which finds the most likely sequence of transmitted bits given the received noisy signal. The traceback length in the Viterbi decoder is set to 3, allowing the decoder to consider past states for accurate decoding.

BPSK Modulation: The encoded binary sequence (0s and 1s) is mapped to a bipolar Non-Return-to-Zero (NRZ) format (-1 for 0, $+1$ for 1) and modulated using BPSK. In BPSK, the carrier signal's phase is shifted by 0° for a $+1$ and 180° for a -1. The modulated signal for the i-th bit is expressed as:

$$s(t) = \sqrt{2E_b} \cdot d_i \cdot \cos(2\pi f_c t)$$

where E_b is the energy per bit (set to 0.5), $d_i \in \{+1, -1\}$ is the bipolar data, and f_c is the carrier frequency (5000 Hz).

DS-CDMA Spreading: In DS-CDMA, each bit is spread using a PN sequence, which is a pseudo-random binary sequence generated by a shift register with a specific initial seed (in this case, [1 -1 1 -1]). The chip rate (10,000 Hz) is 10 times the bit rate (1000 Hz), so each bit is represented by 10 chips, increasing the signal's bandwidth. The spreading process involves multiplying the BPSK-modulated signal by the PN sequence, resulting in a spread-spectrum signal:

$$s_{\text{tx}}(t) = s(t) \cdot p(t)$$

where $p(t) \in \{+1, -1\}$ is the PN sequence.

AWGN Channel: The transmitted signal is corrupted by AWGN, which models thermal noise in the channel. The noise is Gaussian with zero mean and variance determined by the SNR, defined as:

$$\mathrm{SNR} = \frac{E_b}{N_0}$$

where N_0 is the noise power spectral density. The SNR is varied from 0 to 10 dB to evaluate the system's performance under different noise conditions.

Receiver Processing: At the receiver, the signal is despread by multiplying it with the same PN sequence used at the transmitter, which correlates the signal and suppresses interference. The despread signal is then demodulated by multiplying it with the carrier signal $\sqrt{2E_b}\cos(2\pi f_c t)$. The demodulated signal is integrated over each bit period (1 ms) to produce a decision statistic. A threshold detector (positive for 1, negative for 0) recovers the binary sequence, which is then decoded using the Viterbi algorithm to correct errors introduced by the channel.

BER Calculation: The BER is computed by comparing the decoded bits with the original message bits, accounting for a decoding delay due to the traceback length. The BER is plotted against SNR to assess the system's performance. Theoretically, convolutional coding reduces the BER compared to an uncoded system, as the coding gain improves error correction at the cost of increased

MATLAB Source Code:

```
clear all;
close all;
msg = round(rand(1,1000)); % Random bit sequence of length 1000
% 1/2 rated convolutional Encoder
trellis = poly2trellis(3,[6 7]); % Trellis structure with generator polynomials 6 and
7
user = convenc(msg,trellis); % Encodes the message
% Convolutionally encoded data (0,1) mapped to +1/-1
length_user = length(user);
for i = 1:length_user
   if user(i) == 0
      user(i) = -1;
   end
end
fc = 5000; % Carrier frequency (KHz)
eb = 0.5; % Energy per bit for BPSK
bitrate = 1000; % 1KHz
tb = 1/bitrate; % Time per bit (1 ms)
chiprate = 10000; % Chip rate (10 chips per bit)
tc = 1/chiprate; % Time per chip
% CDMA transmitter for a single user
t = tc:tc:tb*length_user;
% Plotting baseband signal
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')
% BPSK Modulation
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);
```

```
% Upsample PN sequence
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
% AWGN Channel
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');
   % Demodulation for user 1
   rx = composite_signal.*pnupsampled;
   % BPSK demodulation
   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];
```

```
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
% PN generator for a single user
seed = [1 -1 1 -1]; % Initial seed
spreadspectrum = [];
pn = [];
for i = 1:length_user
    for j = 1:10 % 10 chips per bit
        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
```

```
        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 channel')
grid on
```

2. **Experiment Name:** Write a MATLAB program to evaluate performance of a 1/2-rated convolutionally encoded DS CDMA system in AWGN and Rayleigh fading channel.

Theory:

Direct Sequence Code Division Multiple Access (DS-CDMA) is a multiple-access technique that enables multiple users to share the same frequency band by assigning each user a unique pseudo-random noise (PN) sequence for spreading the signal across a wider bandwidth. This spreading enhances resistance to interference and improves signal detection in challenging channel conditions. In this experiment, we evaluate the performance of a DS-CDMA system employing a 1/2-rate convolutional code for error correction and Binary Phase Shift Keying (BPSK) modulation. The system operates over a composite channel model that includes both Additive White Gaussian Noise (AWGN) and Rayleigh fading. The performance is measured by the Bit Error Rate (BER) as a function of the signal-to-noise ratio (SNR).

Convolutional Coding: Convolutional coding is a forward error correction technique that introduces redundancy to improve the reliability of data transmission. The 1/2-rate convolutional encoder, defined by a trellis structure with generator polynomials 6 (binary 110) and 7 (binary 111) and a memory order of 3, produces two output bits for each input bit, doubling the data length. At the receiver, the Viterbi algorithm decodes the received sequence by finding the most likely transmitted sequence, using a traceback length of 3 to account for the encoder's memory.

BPSK Modulation: The encoded bits (0s and 1s) map to -1 (0) and $+1$ (1) for BPSK, shifting the carrier phase (0° for $+1$, 180° for -1):

$$s(t) = \sqrt{2E_b} \cdot d_i \cdot \cos(2\pi f_c t)$$

where $E_b = 0.5$, $d_i \in \{+1, -1\}$, and $f_c = 5000\,\text{Hz}$.

DS-CDMA Spreading: Each bit is spread by a PN sequence (chip rate 10,000 Hz, bit rate 1000 Hz, 10 chips/bit), multiplying the BPSK signal:

$$s_{\text{tx}}(t) = s(t) \cdot p(t)$$

where $p(t) \in \{+1, -1\}$.

Channel Model:

- **Rician Fading**: Models a line-of-sight (LOS) path plus multipath, with a Rician factor K (LOS-to-multipath power ratio). The faded signal is: $s_{\text{faded}}(t) = h(t) \cdot s_{\text{tx}}(t)$ where $h(t)$ follows a Rician distribution.
- **AWGN**: Adds Gaussian noise with variance based on SNR: $r(t) = h(t) \cdot s_{\text{tx}}(t) + n(t)$ SNR ranges from 0 to 10 dB.

MATLAB Source Code:

```
clear all;
close all;
msg = round(rand(1,1000));
% 1/2 rated convolutional Encoder
trellis = poly2trellis(3,[6 7]);
user = convenc(msg,trellis);
% Convolutionally encoded data (0,1) mapped to +1/-1
length_user = length(user);
for i = 1:length_user
    if user(i) == 0
        user(i) = -1;
    end
end
fc = 5000; % Carrier frequency (KHz)
eb = 0.5; % Energy per bit
bitrate = 1000; % 1KHz
tb = 1/bitrate; % Time per bit
```

```
        seed(4) = seed(3);
        seed(3) = seed(2);
        seed(2) = seed(1);
        seed(1) = temp;
    end
end
% Upsample PN sequence
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
```

```
chiprate = 10000;
tc = 1/chiprate;
% CDMA transmitter for a single user
t = tc:tc:tb*length_user;
% Plotting baseband signal
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')
% BPSK Modulation
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
% PN generator for a single user
seed = [1 -1 1 -1];
spreadspectrum = [];
pn = [];
for i = 1:length_user
   for j = 1:10 % 10 chips per bit
      pn = [pn seed(4)];
      if seed(4) == seed(3)
         temp = -1;
      else
         temp = 1;
      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
% Adding fading channel effect
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');
   % Demodulation for user 1
   rx = composite_signal.*pnupsampled;
   % BPSK demodulation
   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
```

3. **Experiment Name:** Write a MATLAB program to evaluate performance of a 1/2-rated convolutionally encoded DS CDMA system in AWGN and Rician fading channel.

Theory:

Direct Sequence Code Division Multiple Access (DS-CDMA) is a multiple-access technique widely used in wireless communication systems, enabling multiple users to share the same frequency band by assigning each user a unique pseudo-random noise (PN) sequence. This sequence spreads the signal across a wider bandwidth, enhancing resistance to interference, multipath fading, and noise. The experiment evaluates the performance of a DS-CDMA system that employs a 1/2-rate convolutional code for error correction and Binary Phase Shift Keying (BPSK) modulation, operating over a composite channel model comprising Additive White Gaussian Noise (AWGN) and Rician fading. The performance metric is the Bit Error Rate (BER), which quantifies the probability of bit errors at the receiver as a function of the signal-to-noise ratio (SNR).

Convolutional Coding: Convolutional coding is a forward error correction technique that adds redundancy to the transmitted data to improve reliability in noisy and fading channels. A 1/2-rate convolutional encoder generates two output bits for each input bit, effectively doubling the data length and reducing the effective data rate. The encoder is defined by a trellis structure with generator polynomials (e.g., 6 (binary 110) and 7 (binary 111)) and a memory order of 3, meaning it uses three previous bits to compute each output. At the receiver, the Viterbi algorithm, a maximum-likelihood decoding method, recovers the original message by tracing the most likely path through the trellis, using a traceback length (e.g., 3) to account for the encoder's memory. This coding gain improves BER performance at the cost of increased bandwidth.

BPSK Modulation: The encoded bits (0s and 1s) map to -1 (0) and $+1$ (1) for BPSK, shifting the carrier phase (0° for $+1$, 180° for -1):
$s(t) = \sqrt{2E_b} \cdot d_i \cdot \cos(2\pi f_c t)$
where $E_b = 0.5$, $d_i \in \{+1, -1\}$, and $f_c = 5000\,\mathrm{Hz}$.

DS-CDMA Spreading: Each bit is spread by a PN sequence (chip rate 10,000 Hz, bit rate 1000 Hz, 10 chips/bit), multiplying the BPSK signal:
$s_{\mathrm{tx}}(t) = s(t) \cdot p(t)$
where $p(t) \in \{+1, -1\}$.

Channel Model:

- **Rician Fading**: Models a line-of-sight (LOS) path plus multipath, with a Rician factor K (LOS-to-multipath power ratio). The faded signal is: $s_{\mathrm{faded}}(t) = h(t) \cdot s_{\mathrm{tx}}(t)$ where $h(t)$ follows a Rician distribution.
- **AWGN**: Adds Gaussian noise with variance based on SNR: $r(t) = h(t) \cdot s_{\mathrm{tx}}(t) + n(t)$ SNR ranges from 0 to 10 dB.

Receiver: The signal is despread with the PN sequence, demodulated with the carrier, integrated, and threshold-detected. The Viterbi decoder corrects errors. BER is computed by comparing decoded and original bits.
Performance: Rician fading (with LOS) yields lower BER than Rayleigh fading but higher than AWGN-only. Convolutional coding reduces BER, balancing data rate and reliability.

MATLAB Source Code:

```
clear all;
close all;
msg=round(rand(1,1000));
%1/2 rated convolutional Encoder
trellis=poly2trellis(3,[6 7]);
user=convenc(msg,trellis);
% Convolutionally encoded data(0,1) are  mapping into +1/1

%% To convert the binary sequences to bipolar NRZ format

length_user=length(user);
for i=1:length_user
if user(i)==0
user(i)=-1;
end
end
fc=5000; %%carrier frequency, %KHz
eb=.5;     %% energy per bit
bitrate=1000;% 1KHz
tb=1/bitrate; %% time per bit of message sequence
```

```
seed(4)=seed(3);
seed(3)=seed(2);
seed(2)=seed(1);
seed(1)=temp;
end
end
% each bit has 100 samples. and each pn chip has 10 samples. there r

% 10 chip per bit there fore size of pn samples and original bit is same

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
```

```
chiprate=10000;
tc=1/chiprate;
%%% CDMA transmitter for a single user
t=tc:tc:tb*length_user;
%%plotting base band signal for 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')

%%%% BPSK Modulation
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
%length(bpskmod)
number=length(t); %Total number of time segments

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)
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
%%%%%%%%%%%%%Adding fadig channel effect%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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');  %% SNR of % dbs
%%%%%%%%%%%%%DEMODULATION FOR USER 1%%%%%%%%%%%%%%%%%%%%%
rx=composite_signal.*pnupsampled;
%%%% BPSK demodulation for a single user

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
sum;
rxbits=[];
```

```
title('Frequency Domain analysis of BPSK modulated signal for a single user')

xlabel('Frequency (Hz)')
ylabel('Magnitude')
grid on
%% PN generator for  a single user
%% let initial seed for a single user  is 1000
seed=[1 -1 1 -1];  %convert it into bipolar NRZ format

spreadspectrum=[];
pn=[];
for i=1:length_user
for j=1:10 %chip rate is 10 times the bit rate
pn=[pn seed(4)];
if seed (4)==seed(3) temp=-1;
else temp=1;
end
```

```
for i=1:length_user
if sum(i)>0
rxbits=[rxbits 1];
else
rxbits=[rxbits 0];
end
end
tblen = 3; delay = tblen; % Traceback length
decoded = vitdec(rxbits,trellis,tblen,'cont','hard');

[number,rat] = biterr(decoded(delay+1:end),msg(1:end-delay));

ber(m)=rat;
end % for m
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
```

4. **Experiment Name:** Write a MATLAB program to study the performance of a differentially encoded OQPSK based wireless communication system.

Theory:
Offset Quadrature Phase Shift Keying (OQPSK) is a digital modulation scheme used in wireless communication systems to transmit data efficiently over bandlimited channels. This experiment studies the performance of a differentially encoded OQPSK system, evaluating its Bit Error Rate (BER) under a channel model, typically Additive White Gaussian Noise (AWGN).

Differential Encoding: Differential encoding encodes data as phase differences between consecutive symbols, mitigating phase ambiguity at the receiver without requiring absolute phase reference. For OQPSK, input bits are differentially encoded to map phase transitions, simplifying non-coherent detection.

OQPSK Modulation: OQPSK transmits two bits per symbol using four phase states (0°, 90°, 180°, 270°). Unlike QPSK, the in-phase (I) and quadrature (Q) components are offset by half a symbol period, reducing amplitude fluctuations and improving power efficiency. The transmitted signal is:

$$s(t) = \sqrt{E_s} \cdot [d_I(t)\cos(2\pi f_c t) + d_Q(t - T_s/2)\sin(2\pi f_c t)]$$

where E_s is symbol energy, $d_I, d_Q \in \{+1, -1\}$ are data, f_c is carrier frequency, and T_s is symbol duration.

OQPSK Modulation: OQPSK transmits two bits per symbol using four phase states (0°, 90°, 180°, 270°). Unlike QPSK, the in-phase (I) and quadrature (Q) components are offset by half a symbol period, reducing amplitude fluctuations and improving power efficiency. The transmitted signal is:

$s(t) = \sqrt{E_s} \cdot [d_I(t)\cos(2\pi f_c t) + d_Q(t - T_s/2)\sin(2\pi f_c t)]$

where E_s is symbol energy, $d_I, d_Q \in \{+1, -1\}$ are data, f_c is carrier frequency, and T_s is symbol duration.

Channel Model: The signal is typically corrupted by AWGN, with noise variance based on SNR (E_s/N_0). The received signal is:

$r(t) = s(t) + n(t)$

Receiver: The receiver performs differential decoding, correlating the received signal with delayed versions to detect phase differences, followed by decision rules to recover bits. BER is plotted against SNR to assess performance.

Performance: Differential OQPSK offers robust performance in fading channels, with lower BER than coherent QPSK due to non-coherent detection, though it may have slightly higher error rates than coherent OQPSK in AWGN.

MATLAB Source Code:

```
clear all;
close all;
xbit=[1 0 1 1 0 1 0 0 0 1 1 0];

% Initial reference bit is assumed to be 1

% Binary bit strream is in 0 and 1 : 12 bits

% NOT of Exclusive OR operation

difencod(1)=~(1-xbit(1));

for i=2:length(xbit)
difencod(i)=~(difencod(i-1)-xbit(i));

end
% Differential Encoded binary bit stream

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
%Inphase unipolar bit stream

%from differentially encoded baseband

for i=1:2:(length(difencod)-1)

inp(i)=difencod(i);
inp(i+1)=inp(i);
end
%Quadrature unipolar bit stream

%from differentially encoded baseband

for i=2:2:(length(difencod))

qp(i)=difencod(i);
qp(i-1)=qp(i);
end
%Inphase bipolar NRZ bit stream
```

```
title(' Differentially encoded OQPSK modulated signal');

grid on
snr=10;
%Signal-to-noise ratio per sample is assumed to be 10

madd=awgn(mt,snr);
figure(5);
plot(time,madd)
grid on
xlabel( 'Time(sec)');
ylabel( 'Amplitude(volt)');

%title(' Differentially encoded OQPSK modulated signal with added white noise');
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)');

% Initial checking for I and Q channel bit stream

itxx=itx(half:nsamp:length(xbit)*nsamp+half-1);
```

```
for i=1:(length(inp))
if(inp(i)== 1)
it(i)=1;
elseif(inp(i)==0)
it(i)=-1;
end
end
%Quadrature bipolar NRZ bit stream
for i=1:(length(qp))
if(qp(i)== 1)
qt(i)=1;
elseif(qp(i)==0)
qt(i)=-1;
end
end
% Raised Cosine Filter used

filtorder = 40; % Filter order

nsamp=4;
delay = filtorder/(nsamp*2);

rolloff = 0.5; % Rolloff factor of filter

rrcfilter = rcosine(1,nsamp,'fir/normal',rolloff,delay);

% Plot impulse response.

figure(1);
impz(rrcfilter,1);
grid on
%title(' Impulse response of Raised Cosine Filter');

%% Transmitted Signal

% Upsample and apply  raised cosine filter.

itx = rcosflt(it,1,nsamp,'filter',rrcfilter);

Drate=64000;%Bit rate

T=1/Drate;
Ts=T/nsamp;
time=0:Ts:(length(itx)-1)*Ts;

figure(2);
plot(time,itx)
%title(' Low pass filtered InPhase Component');

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;% 900MHz Carrier frequency chosen

dd=2*pi*fc*time';
ddd=2*pi*fc*tme';
% One bit or 1/2 of symbol delay consideration in OQPSK

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)');
```

```
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); % Mean square error

distortion
disp('Q channel bit stream checking')


distortion = sum((qt-chk2).^2)/length(chk2); % Mean square error

distortion
% Differentially decoded bit stream from I and Q channels

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); % Mean square error

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
```

5. **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.

Theory:

This experiment simulates a wireless communication system using BPSK modulation, convolutional coding for Forward Error Correction (FEC), and interleaving to enhance error resilience. **Convolutional Coding** employs a 1/2-rate encoder to add redundancy, improving error correction in noisy channels. **Interleaving** rearranges encoded bits to disperse burst errors, making them appear as random errors that the FEC can correct more effectively. **BPSK Modulation** maps bits to $+1$ (0°) or -1 (180°) phases, offering robustness in AWGN channels. The signal is transmitted through an AWGN channel, modeled as:

$$r(t) = s(t) + n(t)$$

where $s(t)$ is the modulated signal and $n(t)$ is Gaussian noise. At the receiver, the signal is demodulated, deinterleaved, and decoded using the Viterbi algorithm. The **Bit Error Rate (BER)** is computed across a range of Signal-to-Noise Ratios (SNR) to evaluate performance. Waveforms at key stages (baseband, modulated, and received signals) illustrate the system's operation.

Word count: ~150 words, aligned with previous brevity request.

MATLAB Source Code:

```
clear all;
close all;
% Test with synthetically generated sinusoidal wave

f=1000;% Frequency of the audio signal

Fs =4000; % Sampling rate is 4000 samples per second.

t = [1/Fs:1/Fs:1];% total time for simulation=0.05 second.

% Number of samples=4000
Am=1.0;
signal = Am*sin(2*pi*1000*t); % Original signal

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);
```

```
symbol=double(symbol);
Binary_phase_shift_keying_modulated_data = pskmod(symbol,M);

%demodulation of  Binary phase shift keying data
Binary_phase_shift_keying_demodulated_data = pskdemod(Binary_phase_shift_keying_modulated_data,M);

[number,ratio]= symerr(symbol,Binary_phase_shift_keying_demodulated_data) % symbol error
%symbol to  bit mapping
%1-bit symbol to Binary bit mapping
Retrieved_bit = de2bi(Binary_phase_shift_keying_demodulated_data,'left-msb');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Deinterleaving
errors = zeros(size(Retrieved_bit));
inter_err = bitxor(Retrieved_bit,errors); % Include burst error.

data_deinterleave=randdeintrlv(inter_err,st2);
%Convolutional Decoding
tblen=3;
decodx= vitdec(data_deinterleave,t,tblen,'cont','hard'); %

N3=length(decodx);
```

```
interval=(maximumvalue-minimumvalue)/255; % interval:

partition = [minimumvalue:interval:maximumvalue]; % -1:0.0078:1

codebook = [(minimumvalue-interval):interval:maximumvalue]; % -1.0078:0.0078:1

[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,

% matrix is of  4000 rows X 8 columns

% matrixtps is a matrix of 8 rows X4000 columns

matrixtps=matrix';

% Baseband is produced, it has 32000 bits

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

Tb=1/32000;

% bit rate 32 kbps

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'; % 1 X  32000

%Now, the binary converted data is sent to th Convolutional encode

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

%Channel coding

code = convenc(input_to_Convolutional_encoder,t); % 1 x 64000

st2 = 4831;

data_interleave = randintrlv(code,st2); % Interleave, 1 row x 64000 columns

M=2;

k=log2(M);

% bit to symbol mapping

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

NN=N3/8;

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

decod2(N3)=decodx(1);

decod2=decod2' ; % 32000 X 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

baseband=double(baseband);

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

convert=reshape(decod2,8,4000); % First reshaping and then transposing

matrixtps=double(matrixtps);

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

convert=convert' ; % 4000 rows X 8 columns

%binary to decimally converted value

intconv=bi2de(convert); % converted into interger values(0-255) of 4000 samples

% intconv is 4000 rows X 1 column

[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
```

6. **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.

Theory:

This experiment simulates a wireless communication system utilizing Quadrature Phase Shift Keying (QPSK) modulation, 1/2-rate convolutional coding for Forward Error Correction (FEC), and interleaving to enhance robustness against errors. **Convolutional Coding** employs a 1/2-rate encoder, defined by generator polynomials (e.g., 6 (binary 110) and 7 (binary 111)) with a memory order of 3, generating two output bits per input bit. This redundancy enables error correction, with decoding performed using the Viterbi algorithm. **Interleaving** reorders encoded bits to spread burst errors across the sequence, transforming them into random errors that FEC can handle more effectively. **QPSK Modulation** maps two bits per symbol to four phase states (45°, 135°, 225°, 315°), improving spectral efficiency over BPSK. The modulated signal is expressed as:

$$s(t) = \sqrt{E_s} \cdot [d_I(t)\cos(2\pi f_c t) - d_Q(t)\sin(2\pi f_c t)]$$

where E_s is symbol energy, $d_I, d_Q \in \{+1, -1\}$ represent in-phase and quadrature components, and f_c is the carrier frequency. The signal is transmitted through an Additive White Gaussian Noise (AWGN) channel:

$$r(t) = s(t) + n(t)$$

where $n(t)$ is Gaussian noise. The receiver demodulates, deinterleaves, and decodes the signal, evaluating **Bit Error Rate (BER)** across various Signal-to-Noise Ratios (SNR) to assess system performance.

MATLAB Source Code:

```
clear all;

close all;

% Test with synthetically generated sinusoidal wave

f=1000;% Frequency of the audio signal

Fs =4000; % Sampling rate is 4000 samples per second.

t = [1/Fs:1/Fs:1];% total time for simulation=0.05 second.

% Number of samples=4000

Am=1.0;

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

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')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Quadrature phase shift keying modulation

M=4;

k=log2(M);

baseband=double(baseband);

% bit to symbol mapping

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

Quadrature_phase_shift_keying_modulated_data = pskmod(symbol,M);

% demodulation of Quadrature phase shift keying data

Quadrature_phase_shift_keying_demodulated_data = pskdemod(Quadrature_phase_shift_keying_modulated_data,M);

[number,ratio]= symerr(symbol,Quadrature_phase_shift_keying_demodulated_data)   % symbol error
```

```
grid on

xlabel( 'time(sec)');

ylabel( 'Amplitude(volt)');

maximumvalue=max(signal);

minimumvalue=min(signal);

interval=(maximumvalue-minimumvalue)/255; % interval:

partition = [minimumvalue:interval:maximumvalue]; % -1:0.0078:1

codebook = [(minimumvalue-interval):interval:maximumvalue]; % -1.0078:0.0078:1

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

% Convertion of deci  into binary from least  to most significant

indxtrn=index';

for i=1:4000

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

end,

% matrix is of  4000 rows X 8 columns

% matrixtps is a matrix of 8 rows X4000 columns

matrixtps=matrix';

% Baseband is produced, it has 32000 bits

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

Tb=1/32000;

% bit rate 32 kbps

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')
```

```
% symbol to  bit mapping

% 2-bit symbol to Binary bit mapping

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

Retrieved_bit=Retrieved_bit';

Retrieved_bit=reshape(Retrieved_bit, 64000,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Deinterleaving

errors = zeros(size(Retrieved_bit));

inter_err = bitxor(Retrieved_bit,errors); % Include burst error.

data_deinterleave=randdeintrlv(inter_err,st2);

%Convolutional Decoding

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' ; % 32000 X 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

baseband=double(baseband);

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

convert=reshape(decod2,8,4000); % First reshaping and then transposing

matrixtps=double(matrixtps);

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

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

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Serial the data for the next step.

input_to_Convolutional_encoder = baseband'; % 1 X  32000

%Now, the binary converted data is sent to th Convolutional encoder.

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

%Channel coding

code = convenc(input_to_Convolutional_encoder,t); % 1 x 64000

%Interleaving

st2 = 4831;

data_interleave = randintrlv(code,st2); % Interleave, 1 row x 64000 columns
```

```
convert=convert' ; % 4000 rows X 8 columns

% binary to decimally converted value

intconv=bi2de(convert); % converted into interger values(0-255) of 4000 samples

% intconv is 4000 rows X 1 column

[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
```

7. **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**.**

Theory:

This experiment simulates a wireless communication system employing 4-Quadrature Amplitude Modulation (4-QAM), 1/2-rate convolutional coding for Forward Error Correction (FEC), and interleaving to enhance error resilience. **Convolutional Coding** uses a 1/2-rate encoder with generator polynomials (e.g., 6 (binary 110), 7 (binary 111)) and memory order 3, producing two output bits per input bit to enable error correction. The Viterbi algorithm decodes the received sequence, leveraging redundancy to mitigate errors. **Interleaving** reorders encoded bits to disperse burst errors, converting them into random errors that FEC can correct more effectively. **4-QAM Modulation**, equivalent to QPSK, maps two bits per symbol to four constellation points (e.g., (±1, ±1)), achieving higher spectral efficiency than BPSK. The modulated signal is:

$$s(t) = \sqrt{E_s} \cdot [d_I(t)\cos(2\pi f_c t) - d_Q(t)\sin(2\pi f_c t)]$$

where E_s is symbol energy, $d_I, d_Q \in \{+1, -1\}$ are in-phase and quadrature components, and f_c is the carrier frequency. The signal is transmitted through an Additive White Gaussian Noise (AWGN) channel:

$$r(t) = s(t) + n(t)$$

where $n(t)$ is Gaussian noise with variance based on Signal-to-Noise Ratio (SNR). The receiver demodulates the signal by correlating with carrier signals, deinterleaves the bits, and applies Viterbi decoding. **Bit Error Rate (BER)** is evaluated across SNR values to assess performance. Waveforms at key stages (baseband, modulated, received) illustrate signal transformations, highlighting the impact of modulation, noise, and error correction in the system.

MATLAB Source Code:

```
clear all;

close all;

% Test with synthetically generated sinusoidal wave

f=1000;% Frequency of the audio signal

Fs =4000; % Sampling rate is 4000 samples per second.

t = [1/Fs:1/Fs:1];% total time for simulation=0.05 second.

% Number of samples=4000

Am=1.0;

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

figure(1);

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

```
Quadrature_amplitude_modulated_data = qammod(symbol,M);

% demodulation of Quadrature amplitude data

Quadrature_amplitude_demodulated_data = qamdemod(Quadrature_amplitude_modulated_data,M);

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

% symbol to  bit mapping

% 2-bit symbol to Binary bit mapping

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

Retrieved_bit=Retrieved_bit';

Retrieved_bit=reshape(Retrieved_bit, 64000,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
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; % interval:

partition = [minimumvalue:interval:maximumvalue]; % -1:0.0078:1

codebook = [(minimumvalue-interval):interval:maximumvalue]; % -1.0078:0.0078:1

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

% Convertion of deci  into binary from least  to most significant

indxtrn=index';

for i=1:4000

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

end,

% matrix is of  4000 rows X 8 columns

% matrixtps is a matrix of 8 rows X4000 columns

matrixtps=matrix';

% Baseband is produced, it has 32000 bits

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

Tb=1/32000;

% bit rate 32 kbps

time=[0:Tb:1];

figure(2);

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

title(' A segment of baseband signal')

% Deinterleaving

errors = zeros(size(Retrieved_bit));

inter_err = bitxor(Retrieved_bit,errors); % Include burst error.

data_deinterleave=randdeintrlv(inter_err,st2);

%Convolutional Decoding

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' ; % 32000 X 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

baseband=double(baseband);

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

convert=reshape(decod2,8,4000); % First reshaping and then transposing

matrixtps=double(matrixtps);

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

convert=convert' ; % 4000 rows X 8 columns

% binary to decimally converted value

intconv=bi2de(convert); % converted into interger values(0-255) of 4000 samples

% intconv is 4000 rows X 1 column

[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])
```

xlabel('Time(sec)')
 	
ylabel('Binary value')
 	
set(gca,'ytick',[0 1])
 	
axis([0,time(500),0,1])
 	
%%
 	
% Serial the data for the next step.
 	
input_to_Convolutional_encoder = baseband'; % 1 X 32000
 	
%Now, the binary converted data is sent to th Convolutional encoder.
 	
t=poly2trellis(7, [171 133]);
 	
%Channel coding
 	
code = convenc(input_to_Convolutional_encoder,t); % 1 x 64000
 	
%Interleaving
 	
st2 = 4831;
 	
data_interleave = randintrlv(code,st2); % Interleave, 1 row x 64000 columns
 	
%Quadrature amplitude modulation
 	
M=4;
 	
k=log2(M);
 	
baseband=double(baseband);
 	
% bit to symbol mapping
 	
symbol=bi2de(reshape(data_interleave,k,length(data_interleave)/k).','left-msb');
	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

8. **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.

Theory:

This experiment simulates a wireless communication system using 16-Quadrature Amplitude Modulation (16-QAM), 1/2-rate convolutional coding for Forward Error Correction (FEC), and interleaving to enhance error resilience. **Convolutional Coding** employs a 1/2-rate encoder with generator polynomials (e.g., 6 (binary 110), 7 (binary 111)) and memory order 3, generating two output bits per input bit to improve error correction. The Viterbi algorithm decodes the received sequence, leveraging redundancy to mitigate errors. **Interleaving** reorders encoded bits to disperse burst errors, making them appear random for effective FEC decoding. **16-QAM Modulation** maps four bits per symbol to 16 constellation points (e.g., (±1, ±1), (±1, ±3), (±3, ±1), (±3, ±3)), achieving higher spectral efficiency than QPSK. The modulated signal is:
s(t)>&= \sqrt{E_s} \cdot [d_I(t) \cos(2\pi f_c t) - d_Q(t) \sin(2\pi f_c t)]
where E_s is symbol energy, $d_I, d_Q \in \{\pm1, \pm3\}$ are in-phase and quadrature amplitudes, and f_c is the carrier frequency. The signal is transmitted through an AWGN channel:
r(t)>&= s(t) + n(t)
where $n(t)$ is Gaussian noise. The receiver demodulates, deinterleaves, and decodes using Viterbi decoding. **Bit Error Rate (BER)** is evaluated across Signal-to-Noise Ratios (SNR) to assess performance. Waveforms (baseband, modulated, received) illustrate signal transformations, highlighting modulation, noise, and error correction effects.

MATLAB Source Code:

```
clear all;
close all;
% Test with synthetically generated sinusoidal wave

f=1000;% Frequency of the audio signal

Fs =4000; % Sampling rate is 4000 samples per second.

t = [1/Fs:1/Fs:1];% total time for simulation=0.05 second.

% Number of samples=4000
Am=1.0;
signal = Am*sin(2*pi*1000*t); % Original signal

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)');
```

```
Retrieved_bit=Retrieved_bit';

Retrieved_bit=reshape(Retrieved_bit, 64000,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

% Deinterleaving

errors = zeros(size(Retrieved_bit));

inter_err = bitxor(Retrieved_bit,errors); % Include burst error.

data_deinterleave=randdeintrlv(inter_err,st2);

%Convolutional Decoding

tblen=3;
decodx= vitdec(data_deinterleave,t,tblen,'cont','hard'); %

N3=length(decodx);

NN=N3/8;
```

```
ylabel( 'Amplitude(volt)');

maximumvalue=max(signal);

minimumvalue=min(signal);

interval=(maximumvalue-minimumvalue)/255; % interval:

partition = [minimumvalue:interval:maximumvalue]; % -1:0.0078:1

codebook = [(minimumvalue-interval):interval:maximumvalue]; % -
1.0078:0.0078:1

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

% Convertion of deci  into binary from least  to most significant

indxtrn=index';

for i=1:4000

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

end,

% matrix is of  4000 rows X 8 columns

% matrixtps is a matrix of 8 rows X4000 columns

matrixtps=matrix';

% Baseband is produced, it has 32000 bits

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

Tb=1/32000;

% bit rate 32 kbps

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])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

% Serial the data for the next step.

input_to_Convolutional_encoder = baseband'; % 1 X  32000

%Now, the binary converted data is sent to th Convolutional encoder.

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

%Channel coding
```

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

decod2(N3)=decodx(1);

decod2=decod2' ; % 32000 X 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

baseband=double(baseband);

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

convert=reshape(decod2,8,4000); % First reshaping and then
transposing

matrixtps=double(matrixtps);

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

convert=convert' ; % 4000 rows X 8 columns

% binary to decimally converted value

intconv=bi2de(convert); % converted into interger values(0-255) of
4000 samples

% intconv is 4000 rows X 1 column

[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])
```

```
code = convenc(input_to_Convolutional_encoder,t); % 1 x 64000

%Interleaving

st2 = 4831;

data_interleave = randintrlv(code,st2); % Interleave, 1 row x 64000 columns

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%Quadrature amplitude modulation

M=16;

k=log2(M);

baseband=double(baseband);

% bit to symbol mapping

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

Quadrature_amplitude_modulated_data = qammod(symbol,M);

% demodulation of Quadrature amplitude data

Quadrature_amplitude_demodulated_data =
qamdemod(Quadrature_amplitude_modulated_data,M);

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

% symbol to  bit mapping

% 2-bit symbol to Binary bit mapping

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

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

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

xlabel('Time(sec)')

ylabel('Amplitude')

grid on
```