ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

[AN AUTONOMOUS INSTITUTION]

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ME-COMMUNICATION SYSTEMS

LAB MANUAL

SUBJECT CODE	:	**24CM231**
SUBJECT NAME	:	**WIRELESS COMMUNICATION LABORATORY**
YEAR/SEM	:	**01 / 02**
REGULATION	:	**RCET-2024**
COURSE COORDINATOR	:	
COURSE FACULTY	:	

24CM231 WIRELESS COMMUNICATION LABORATORY L T P

0 0 3

COURSE OBJECTIVES:

- To enable the student to verify the basic principles of random signal processing, spectral estimation methods, wireless and AWGN channel characterization, application of adaptive filter algorithms for communication system design, coding and modulation design, synchronization aspects and the overall baseband system design.
- To design and conduct experiments, as well as to analyze and interpret data to produce meaningful conclusions and match with theoretical concepts.
- To enable the student to appreciate the practical aspects of baseband system design and understand the associated challenges.

TEACHING-LEARNING PROCESS:

Suggested strategies that teachers may use to effectively achieve the course outcomes:

1. PowerPoint presentation
2. Interactive Simulations
3. Lab experiment videos
4. Blended Mode of Learning
5. Project based Learning
6. Experiential Learning
7. NPTEL and Other Videos
8. Smart Class Room

COURSE OUTCOMES: On completion of the course, the student will have the ability to:

COs	Course Outcome	Cognitive domain
CO1	Develop the physical models of wireless channels.	K3
CO2	Analyze the digital modulation techniques	K4
CO3	Measure capacity of AWGN channel, LTI Gaussian channels and various fading channels	K3
CO4	Illustrate the uplink and downlink model of AWGN channel, fading channels and multiuser diversity	K2

LIST OF EXPERIMENT: **45 PERIODS**

1. SPECTRAL CHARACTERISATION OF COMMUNICATION SIGNALS (USING SPECTRUM ANALYZER)
2. DESIGN AND ANALYSIS OF DIGITAL MODULATION TECHNIQUES ON AN SDR PLATFORM
3. CARRIER AND SYMBOL TIMING SYNCHRONIZATION USING SDR PLATFORM
4. CDMA SIGNAL GENERATION AND RAKE RECEIVER DESIGN USING DSP/MATLAB/ SIMULINK
5. DESIGN AND PERFORMANCE ANALYSIS OF ERROR CONTROL ENCODER AND DECODER (BLOCKAND CONVOLUTIONAL CODES)
6. WIRELESS CHANNEL EQUALIZER DESIGN USING DSP (ZF / LMS / RLS)
7. WIRELESS CHANNEL ESTIMATION AND DIVERSITY COMBINING

Ex. No :	01	SPECTRAL CHARACTERISATION OF COMMUNICATION SIGNALS
Date:		

AIM:

To analyze the spectral characteristics of different communication signals using MATLAB and understand their frequency components.

APPARATUS REQUIRED:

- PC
- MATLAB Software

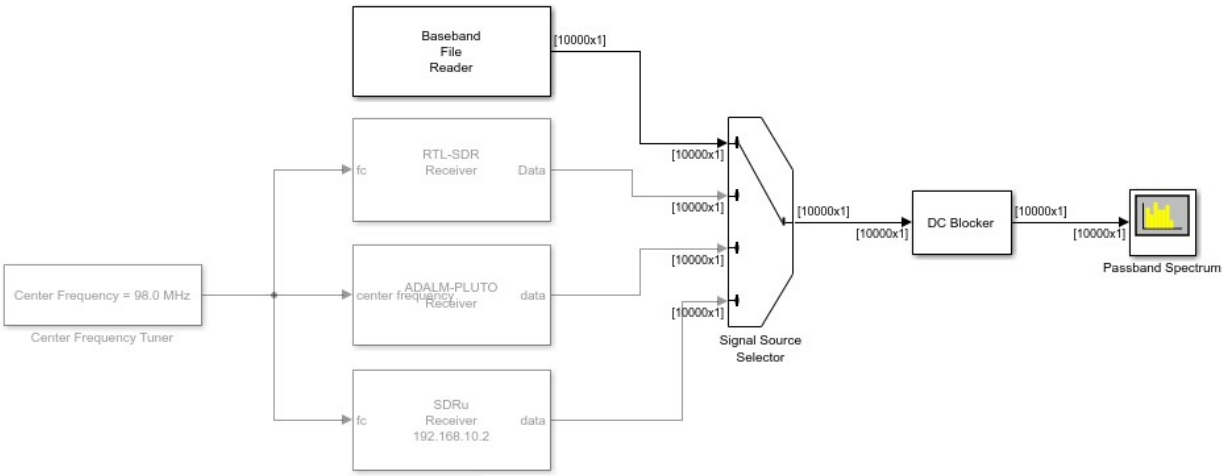
THEORY:

Spectral characterization of communication signals involves analyzing the frequency components of a signal to assess its transmission quality and interference levels. The process typically employs Fast Fourier Transform (FFT) techniques to visualize signal power distribution across the frequency spectrum. Key parameters such as bandwidth, signal-to-noise ratio (SNR), and spur-free dynamic range help in evaluating signal integrity. Spectrum analyzers assist in identifying unwanted distortions, harmonics, and spectral regrowth. This analysis is crucial in optimizing communication system performance, minimizing interference, and ensuring efficient signal transmission.

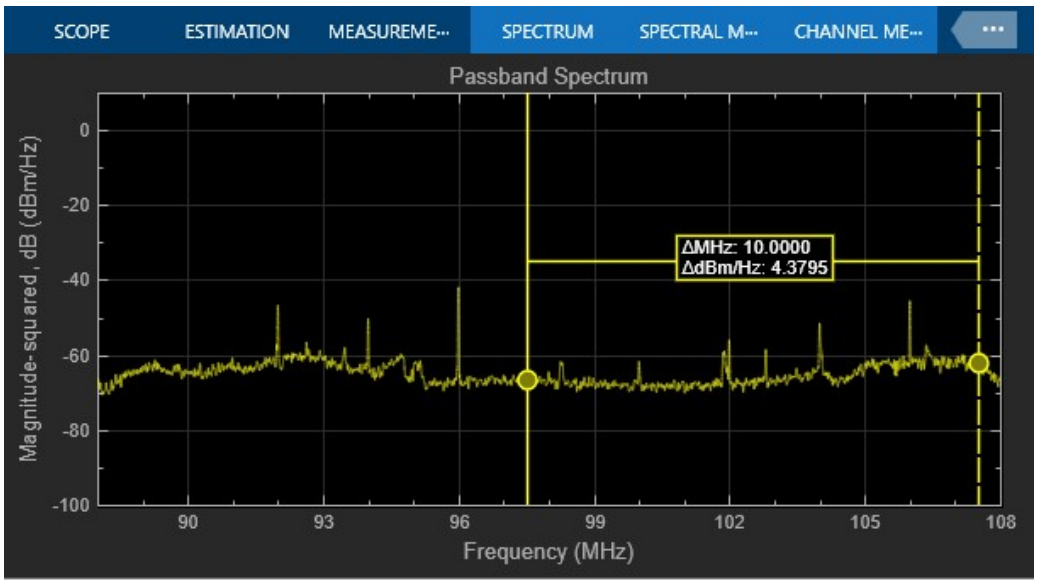
PROCEDURE:

1. Open Matlab version R2020a.
2. Open new Simulink file and design the Block diagram and save it.
3. Add the path to the location of the file in the system.
4. Compile the program and check for any error and debug it.
5. Note down the output.

BLOCK DIAGRAM:



OUTPUT:



Peaks

DC Blocker	P1	P2	P3
Frequency (kHz)			
Power Density (dBm/Hz)			

Stopped ΔT=51.2000 us Samples/Update=1536 VBW=84.5849 Hz RBW=19.5312 kHz Sample Rate=

RESULT :

The spectral analysis of communication signals using MATLAB was successfully implemented and the output was verified.

Ex. No :	02	DESIGN AND ANALYSIS OF DIGITAL MODULATION TECHNIQUES ON AN SDR PLATFORM
Date:		

AIM:

To design and analyze different digital modulation techniques using MATLAB and study their signal characteristics.

APPARATUS REQUIRED:

- PC
- MATLAB Software

THEORY:

Binary Phase Shift Keying (BPSK) modulates the phase of a carrier signal with two distinct phase states, making it robust against noise. Quadrature Phase Shift Keying (QPSK) improves bandwidth efficiency by encoding two bits per symbol using four phase shifts. Quadrature Amplitude Modulation (QAM) combines amplitude and phase variations to transmit multiple bits per symbol, enhancing data rates. MATLAB helps visualize constellation diagrams and analyze performance metrics like Bit Error Rate (BER). These techniques are widely used in wireless communication for efficient data transmission.

PROCEDURE:

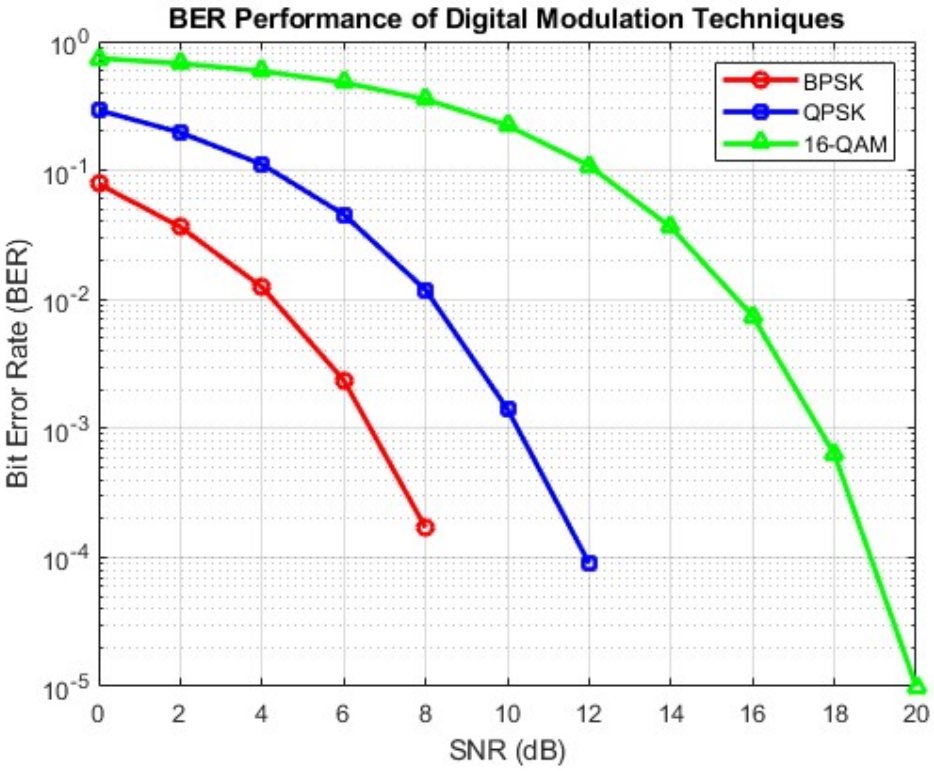
1. Open Matlab version R2020a.
2. Open new file and enter the program and save it.
3. Add the path to the location of the file in the system.
4. Compile the program and check for any error and debug it.
5. Note down the output.

PROGRAM :

```
clc; clear; close all;
% Parameters
```

```
SNR_dB = 0:2:20; numSymbols = 1e5;
% BPSK Modulation
M = 2;
dataBits = randi([0 M-1], numSymbols, 1);
modSymbols = pskmod(dataBits, M, pi);
BER_BPSK = zeros(1, length(SNR_dB));
for snrIdx = 1:length(SNR_dB)
    rxSymbols = awgn(modSymbols, SNR_dB(snrIdx), 'measured');
    rxBits = pskdemod(rxSymbols, M, pi);
    BER_BPSK(snrIdx) = sum(dataBits ~= rxBits) / numSymbols;
end

% QPSK Modulation
M = 4;
dataBits = randi([0 M-1], numSymbols, 1);
modSymbols = pskmod(dataBits, M, pi/4);
BER_QPSK = zeros(1, length(SNR_dB));
for snrIdx = 1:length(SNR_dB)
    rxSymbols = awgn(modSymbols, SNR_dB(snrIdx), 'measured');
    rxBits = pskdemod(rxSymbols, M, pi/4);
    BER_QPSK(snrIdx) = sum(dataBits ~= rxBits) / numSymbols;
end

% 16-QAM Modulation
M = 16;
dataBits = randi([0 M-1], numSymbols, 1);
modSymbols = qammod(dataBits, M, 'UnitAveragePower', true);
BER_QAM = zeros(1, length(SNR_dB));
for snrIdx = 1:length(SNR_dB)
    rxSymbols = awgn(modSymbols, SNR_dB(snrIdx), 'measured');
    rxBits = qamdemod(rxSymbols, M, 'UnitAveragePower', true);
    BER_QAM(snrIdx) = sum(dataBits ~= rxBits) / numSymbols;
end

% Plot BER vs. SNR
figure;
semilogy(SNR_dB, BER_BPSK, 'ro-', SNR_dB, BER_QPSK, 'bs-', SNR_dB,
BER_QAM, 'g^-', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)'); ylabel('Bit Error Rate (BER)');
legend('BPSK', 'QPSK', '16-QAM');
title('BER Performance of Digital Modulation Techniques');
```

OUTPUT:



RESULT :

The digital modulation techniques were successfully implemented in MATLAB, and the output was verified.

Ex. No :	03	CARRIER AND SYMBOL TIMING SYNCHRONIZATION USING SDR PLATFORM
Date:		

AIM:

To implement and analyze carrier and symbol timing synchronization using the SDR platform in MATLAB for accurate signal reception.

APPARATUS REQUIRED:

- PC
- MATLAB Software

THEORY:

Carrier and symbol timing synchronization are essential for proper demodulation in wireless communication. Carrier synchronization corrects frequency and phase offsets, ensuring accurate signal decoding. Symbol timing synchronization aligns the received symbols with the correct sampling points to reduce bit errors. MATLAB, along with the SDR platform, helps simulate and correct synchronization issues. Techniques like Phase-Locked Loop (PLL) and correlation methods improve signal alignment. Proper synchronization enhances data accuracy and system performance.

PROCEDURE:

1. Open Matlab version R2020a.
2. Open new file and enter the program and save it.
3. Add the path to the location of the file in the system.
4. Compile the program and check for any error and debug it.
5. Note down the output.

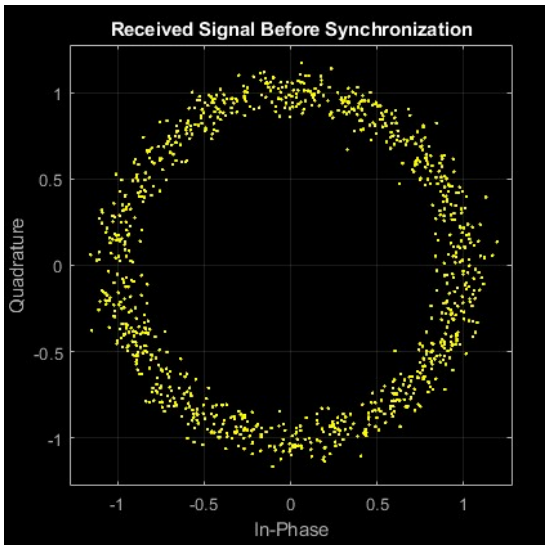
PROGRAM :

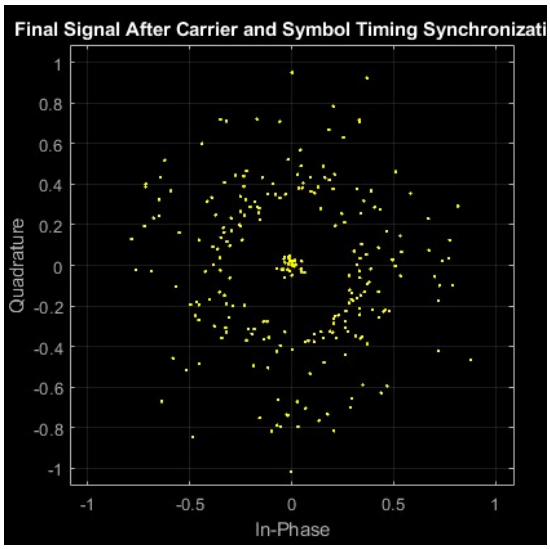
```
% Alternative Carrier and Symbol Timing Synchronization in MATLAB
% Define parameters
```

```
Fs = 1e6; % Sampling Frequency (1 MHz)
Rs = 1e5; % Symbol Rate (100 kHz)
M = 4; % Modulation Order (QPSK)
SNR = 20; % Signal-to-Noise Ratio (dB
% Generate Random Data
numSymbols = 1000;
data = randi([0 M-1], numSymbols, 1);
% QPSK Modulation
modulatedSig = pskmod(data, M, pi/4);
% Simulate Channel Effects (AWGN + Frequency Offset + Timing Offset)
frequencyOffset = 500; % Hz
timingOffset = 0.1; % Fractional delay
timeVector = (0:length(modulatedSig)-1)' / Fs;
freqOffsetSig = modulatedSig .* exp(1j*2*pi*frequencyOffset*timeVector);
rxSig = awgn(freqOffsetSig, SNR, 'measured');
% Plot 1: Received Signal Before Synchronization
figure;
scatterplot(rxSig);
title('Received Signal Before Synchronization');
grid on;
% Carrier Synchronization using Phase-Locked Loop (PLL) Approximation
phzOffset = angle(sum(rxSig .* conj(modulatedSig))) / length(rxSig);
rxCarrierCorrected = rxSig .* exp(-1j * phzOffset);
% Symbol Timing Synchronization using Matched Filtering
matchedFilter = ones(4,1) / 4; % Simple averaging filter
rxFiltered = conv(rxCarrierCorrected, matchedFilter, 'same');
% Downsample to Symbol Rate
```

```
rxAligned = downsample(rxFiltered, 4, round(timingOffset * 4));
% Plot 2: Final Signal After Carrier and Symbol Timing Synchronization
figure;
scatterplot(rxAligned);
title('Final Signal After Carrier and Symbol Timing Synchronization');
grid on;
```

OUTPUT:





RESULT:

Carrier and symbol timing synchronization were successfully implemented using MATLAB and the SDR platform, and the output was verified.

Ex. No :	04	CDMA SIGNAL GENERATION AND RAKE RECEIVER DESIGN USING DSP/MATLAB/ SIMULINK
Date:		

AIM:

To generate CDMA signals and design a RAKE receiver using MATLAB SIMULINK for improved signal reception.

APPARATUS REQUIRED:

- PC
- MATLAB Software

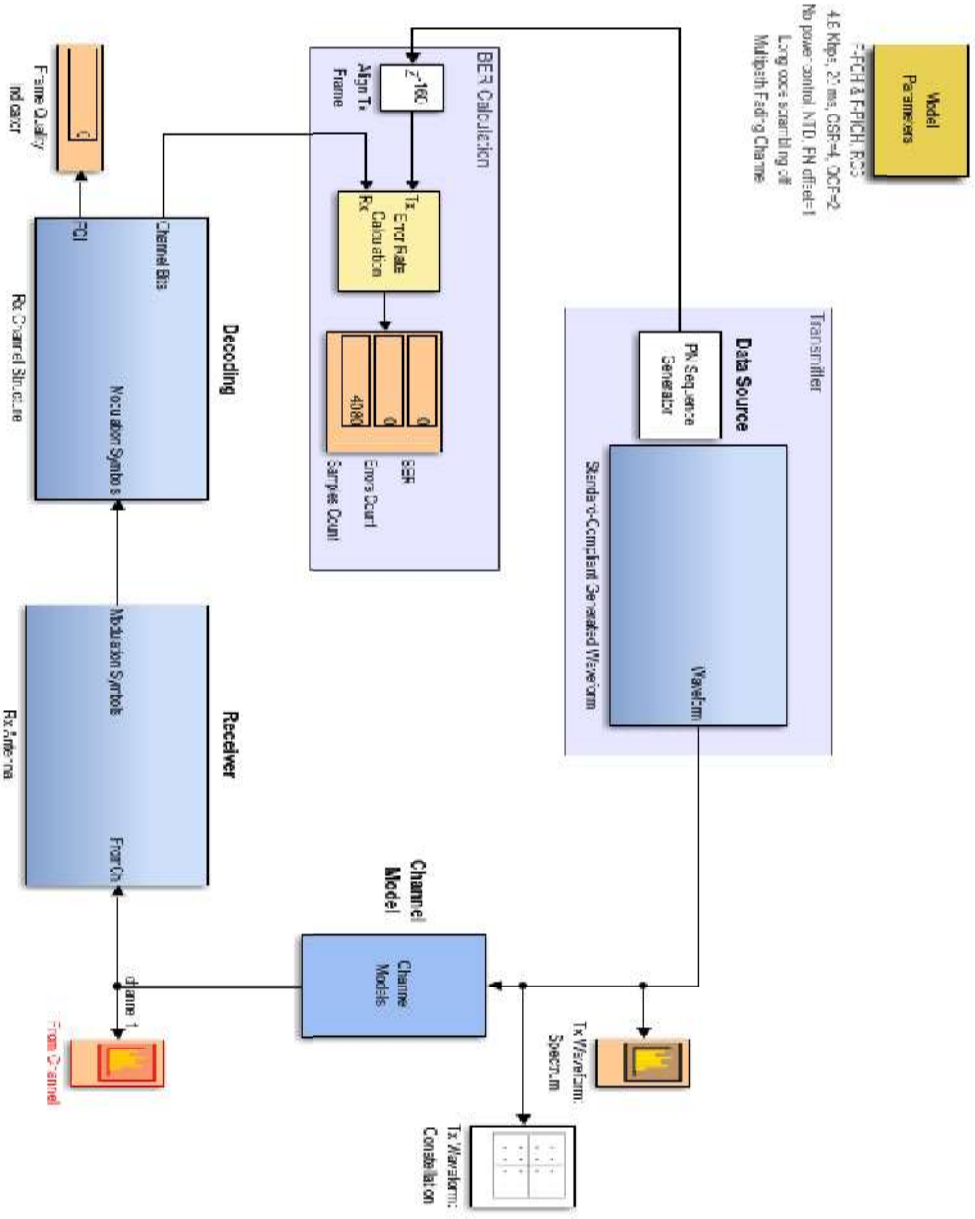
THEORY:

CDMA (Code Division Multiple Access) is a spread spectrum communication technique where multiple users share the same frequency band by using unique spreading codesThe RAKE receiver is designed to mitigate these effects by utilizing multiple correlators to capture and combine delayed versions (multipath components) of the received signal. The RAKE receiver consists of a channel estimator that determines the best weight for each path, enhancing performance in fading environments. This diversity combining technique improves signal reception and minimizes bit errors. The overall system enhances robustness against interference and fading, making CDMA an efficient choice for wireless communication. The combination of CDMA and the RAKE receiver enables reliable and high-capacity transmission in mobile networks.

PROCEDURE:

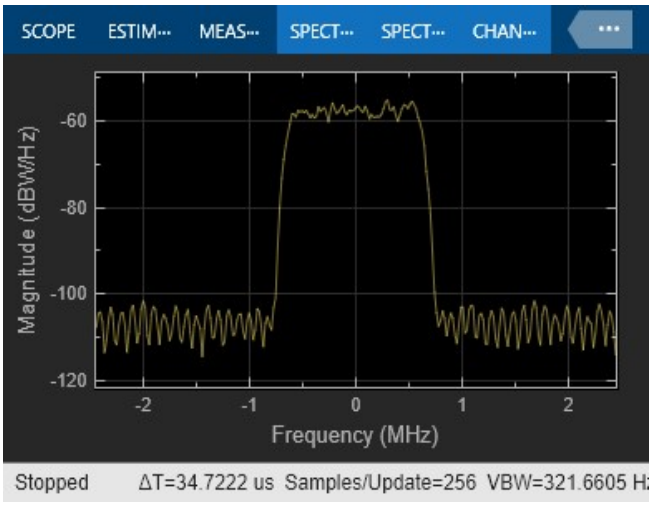
1. Open Matlab version R2020a.
2. Open new simulink file and design the block diagram and save it.
3. Add the path to the location of the file in the system.
4. Compile the program and check for any error and debug it.
5. Note down the output.

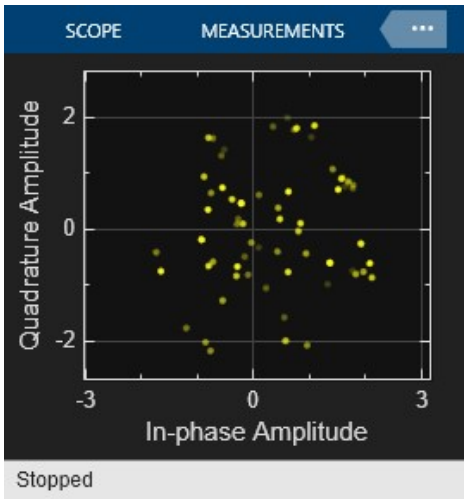
BLOCK DIAGRAM :



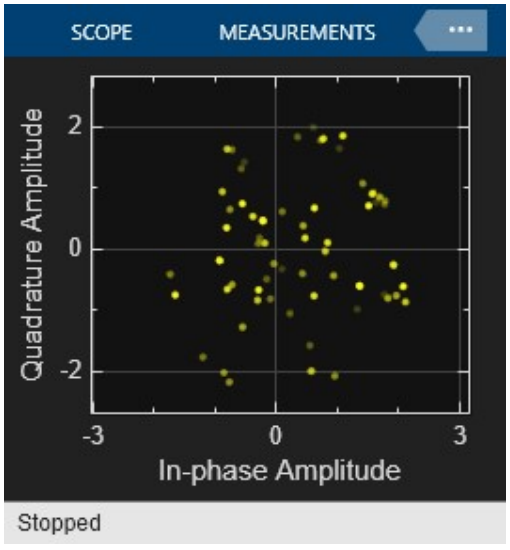
OUTPUT:

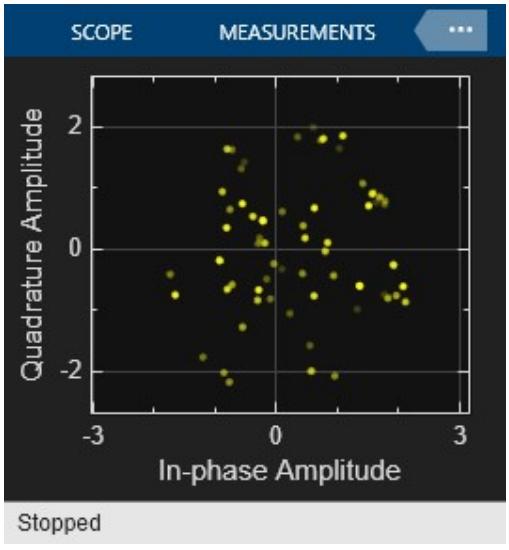
TX WAVE FORM:



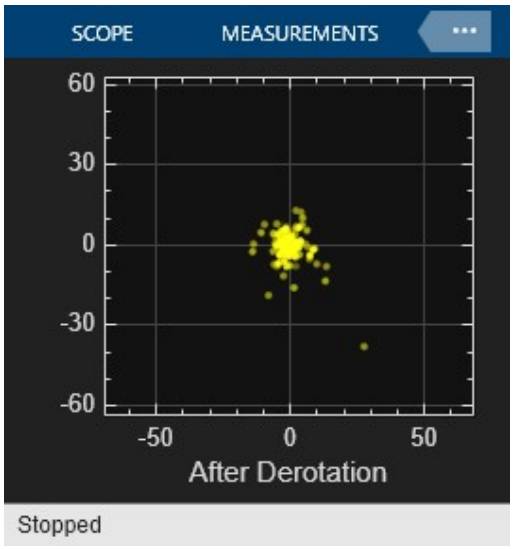


FROM CHANNEL:

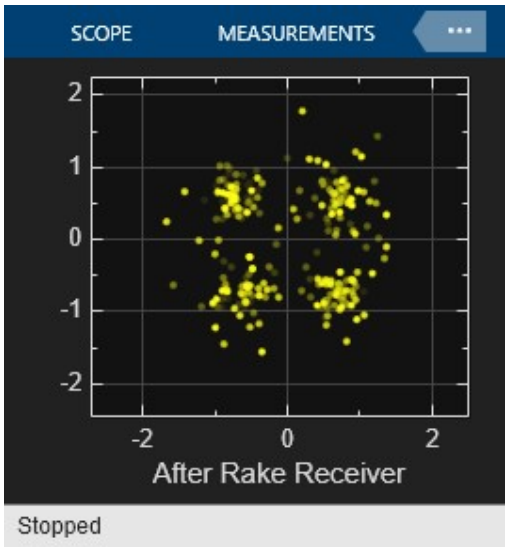




AFTER DETECTION:



RAKE RECIVER :



RESULT:

CDMA signal generation and RAKE receiver design were successfully implemented in MATLAB SIMULINK, and the output was verified.

Ex. No :	05	DESIGN AND PERFORMANCE ANALYSIS OF ERROR CONTROL ENCODER AND DECODER (BLOCKAND CONVOLUTIONAL CODES)
Date:		

AIM:

To design and analyze the performance of error control encoders and decoders using Block and Convolutional Codes in MATLAB.

APPARATUS REQUIRED:

- PC
- MATLAB Software

THEORY:

Error control coding improves data transmission reliability by detecting and correcting errors. Block codes, like Hamming and BCH codes, divide data into fixed-size blocks and add redundancy for error correction. Convolutional codes encode data using memory-based sequences and are decoded using the Viterbi algorithm. These techniques are widely used in wireless communication and digital storage. MATLAB helps simulate encoding, transmission with noise, and decoding performance. Proper error correction improves data integrity and reduces transmission errors.

PROCEDURE:

1. Open Matlab version R2020a.
2. Open new file and enter the program and save it.
3. Add the path to the location of the file in the system.
4. Compile the program and check for any error and debug it.
5. Note down the output.

PROGRAM :

LINEAR BLOCK CODE :

```
clc; clear all;
```

```
disp('Generator Matrix (G):');
G = [1 1 0 1 0 0 0;
     0 1 1 0 1 0 0;
     1 1 1 0 0 1 0;
     1 0 1 0 0 0 1];
disp(G);

[k, n] = size(G);
disp(['Codeword length (n): ', num2str(n), ', Message length (k): ', num2str(k)]);

% Generate all possible messages
M = de2bi(0:2^k-1, k, 'left-msb');
disp('Message Bits:'); disp(M);

% Compute codewords
C = mod(M * G, 2);
disp('Codewords:'); disp(C);

% Compute minimum Hamming weight (excluding all-zero codeword)
disp(['Minimum Hamming Weight: ', num2str(min(sum(C(2:end, :), 2)))]);

% Parity-Check Matrix (H)
H = [eye(n-k), G(:, 1:n-k)'];
disp('Parity-Check Matrix (H):'); disp(H);

% Error Detection & Correction
R = [0 0 1 1 1 0 1];
```

```
S = mod(R * H', 2);
disp('Syndrome:'); disp(S);

for i = 1:size(H, 1)
    if isequal(H(i, :), S), R(i) = 1 - R(i); break; end
end
disp('Corrected Codeword:'); disp(R);
```

CONVOLUTIONAL CODE:

```
clc; clear; close all;

% Convolutional Encoder & Viterbi Decoder setup
trellis = poly2trellis(7, [171 133]);
puncturePattern = [1;1;0;1;1;0];
hConvEnc = comm.ConvolutionalEncoder(trellis, 'PuncturePatternSource', 'Property', 'PuncturePattern', puncturePattern);
hMod = comm.BPSKModulator;
hChan = comm.AWGNChannel('NoiseMethod', 'Signal to noise ratio (Eb/No)', 'SignalPower', 1);
hVitDec = comm.ViterbiDecoder(trellis, 'InputFormat', 'Unquantized', 'PuncturePatternSource', 'Property', 'PuncturePattern', puncturePattern, 'TracebackDepth', 96);
hErrorCalc = comm.ErrorRate('ReceiveDelay', hVitDec.TracebackDepth);

% Eb/N0 values
EbNoEncoderInput = 2:0.5:5;
EbNoEncoderOutput = EbNoEncoderInput + 10*log10(3/4);
frameLength = 3000; targetErrors = 300; maxNumTransmissions = 5e6;
```

```
BERVec = zeros(3, length(EbNoEncoderOutput));

% BER Simulation
for n = 1:length(EbNoEncoderOutput)
    reset(hErrorCalc); reset(hConvEnc); reset(hVitDec);
    hChan.EbNo = EbNoEncoderOutput(n);
    while BERVec(2,n) < targetErrors && BERVec(3,n) < maxNumTransmissions
        data = randi([0 1], frameLength, 1);
        decData = step(hVitDec, real(step(hChan, step(hMod, step(hConvEnc, data)))));
        BERVec(:,n) = step(hErrorCalc, data, decData);
    end
end

% Theoretical BER
dist = 5:11;
nerr = [42 201 1492 10469 62935 379644 2253373];
bound = nerr*(1/6)*erfc(sqrt((3/4) * (10.^((2:.02:5)/10))' * dist))';

% Plot BER results
berfit(EbNoEncoderInput, BERVec(1,:)); hold on;
semilogy(2:.02:5, bound, 'g'); grid on;
legend('Empirical BER', 'Fit for simulated BER', 'Theoretical bound on BER');
xlabel('Eb/N0 (dB)'); ylabel('BER'); title('BER Performance of Punctured Convolutional Codes');
axis([1 6 10^-5 1]);
```

OUTPUT:

LINEAR BLOCK CODE :

Generator Matrix (G):

1 1 0 1 0 0 0

0 1 1 0 1 0 0

1 1 1 0 0 1 0

1 0 1 0 0 0 1

Codeword length (n): 7, Message length (k): 4

Message Bits:

0 0 0 0

0 0 0 1

0 0 1 0

0 0 1 1

0 1 0 0

0 1 0 1

0 1 1 0

0 1 1 1

1 0 0 0

1 0 0 1

1 0 1 0

1 0 1 1

1 1 0 0

1 1 0 1

1 1 1 0

1 1 1 1

Codewords:

0	0	0	0	0	0	0
1	0	1	0	0	0	1
1	1	1	0	0	1	0
0	1	0	0	0	1	1
0	1	1	0	1	0	0
1	1	0	0	1	0	1
1	0	0	0	1	1	0
0	0	1	0	1	1	1
1	1	0	1	0	0	0
0	1	1	1	0	0	1
0	0	1	1	0	1	0
1	0	0	1	0	1	1
1	0	1	1	1	0	0
0	0	0	1	1	0	1
0	1	0	1	1	1	0
1	1	1	1	1	1	1

Minimum Hamming Weight: 3

Parity-Check Matrix (H):

1	0	0	1	0	1	1
0	1	0	1	1	1	0
0	0	1	0	1	1	1

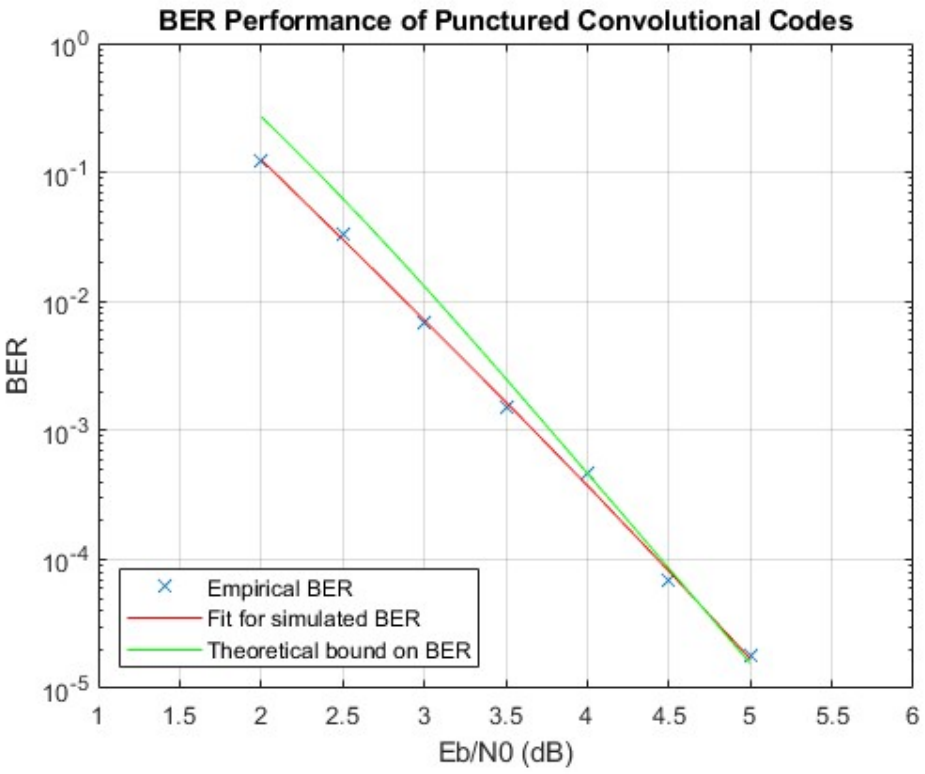
Syndrome:

0	0	1

Corrected Codeword:

0 0 1 1 1 0 1

CONVOLUTIONAL CODE:



RESULT :

The Block and Convolutional error control codes were successfully implemented in MATLAB, and the output was verified.

Ex. No :	06	WIRELESS CHANNEL EQUALIZER DESIGN USING DSP (ZF / LMS / RLS)
Date:		

AIM:

To design and analyze wireless channel equalizers using ZF, LMS, and RLS algorithms in MATLAB for improved signal recovery.

APPARATUS REQUIRED:

- PC
- MATLAB Software

THEORY:

Wireless communication signals suffer from distortion due to multipath fading and interference. Equalizers help in reducing these distortions and improving signal quality. The Zero Forcing (ZF) equalizer removes inter-symbol interference by inverting the channel effect. The Least Mean Squares (LMS) algorithm adapts to changing channel conditions by minimizing the error iteratively. The Recursive Least Squares (RLS) algorithm offers faster convergence and better performance in dynamic environments. MATLAB helps simulate and compare these equalization techniques.

PROCEDURE:

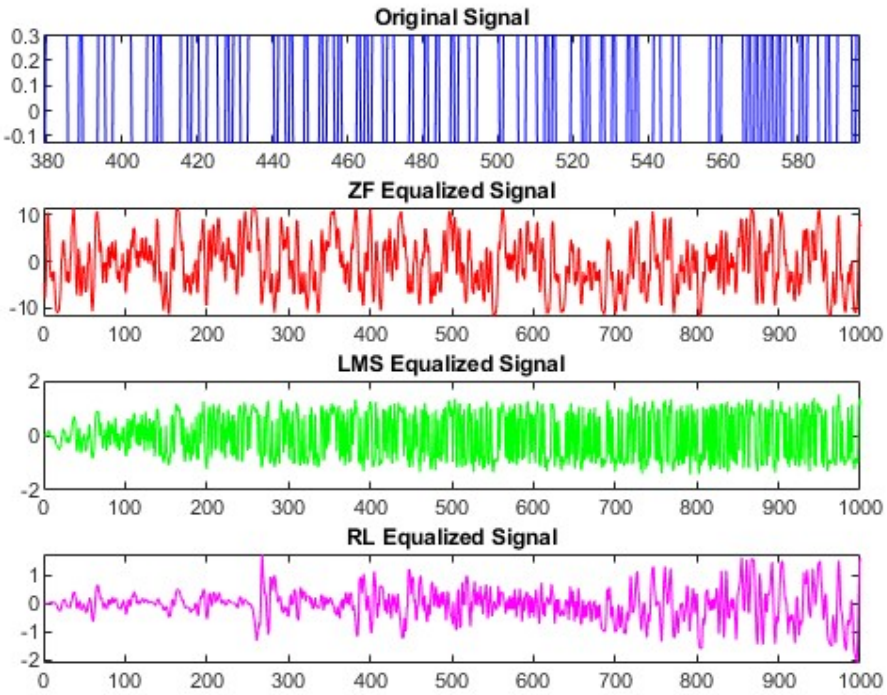
1. Open Matlab version R2020a.
2. Open new file and enter the program and save it.
3. Add the path to the location of the file in the system.
4. Compile the program and check for any error and debug it.
5. Note down the output.

PROGRAM :

```
% Wireless Channel Equalizer Design (ZF / LMS / RL)
% Parameters
N = 1000; SNR = 20; mu = 0.01; % Samples, Noise Ratio, LMS Step size
% Transmit Signal
x = 2 * randi([0 1], 1, N) - 1; % BPSK Signal
% Channel Model with Noise
h = [0.9, 0.5, 0.3];
y = conv(x, h, 'same');
y = y + sqrt(var(y) / (10^(SNR/10))) * randn(size(y));
% Zero-Forcing (ZF) Equalizer
x_zf = conv(y, 1 ./ h, 'same');
% LMS Equalizer
w = zeros(1, length(h)); x_lms = zeros(1, N);
for n = length(h):N
    y_n = y(n:-1:n-length(h)+1);
    e = x(n) - (w * y_n.');
    w = w + mu * e * y_n; x_lms(n) = w * y_n.';
end
% Reinforcement Learning (RL) Equalizer
gamma = 0.99; % Discount factor
w_rl = zeros(1, length(h)); x_rl = zeros(1, N);
for n = length(h):N
    y_n = y(n:-1:n-length(h)+1);
    x_hat = w_rl * y_n.';
    reward = -(x(n) - x_hat)^2; % Reward based on error
```

```
    w_rl = w_rl + mu * reward * y_n * gamma; % Update weights
    x_rl(n) = x_hat;
end
% Plotting Results
subplot(4, 1, 1); plot(x, 'b'); title('Original Signal');
subplot(4, 1, 2); plot(real(x_zf), 'r'); title('ZF Equalized Signal');
subplot(4, 1, 3); plot(real(x_lms), 'g'); title('LMS Equalized Signal');
subplot(4, 1, 4); plot(real(x_rl), 'm'); title('RL Equalized Signal');
```

OUTPUT:



RESULT:

Wireless Channel (ZF / LMS / RLS) equalizers were successfully implemented in MATLAB, and the output was verified.

Ex. No :	07	WIRELESS CHANNEL ESTIMATION AND DIVERSITY COMBINING
Date:		

AIM:

To implement and analyze wireless channel estimation and diversity combining techniques using MATLAB for improved signal reception.

APPARATUS REQUIRED:

- PC
- MATLAB Software

THEORY:

Wireless communication signals suffer from multipath fading, causing interference and signal degradation. Channel estimation helps predict channel conditions for better signal recovery. Maximal Ratio Combining (MRC) is a diversity technique that enhances signal strength by weighting and combining received signals from multiple antennas. It maximizes the Signal-to-Noise Ratio (SNR) by using channel estimates to optimize reception. In this experiment, BPSK-modulated symbols are transmitted through a Rayleigh fading channel, and MRC is applied to improve signal quality. The performance is evaluated using Bit Error Rate (BER) analysis in MATLAB.

PROCEDURE:

1. Open Matlab version R2020a.
2. Open new file and enter the program and save it.
3. Add the path to the location of the file in the system.
4. Compile the program and check for any error and debug it.
5. Note down the output.

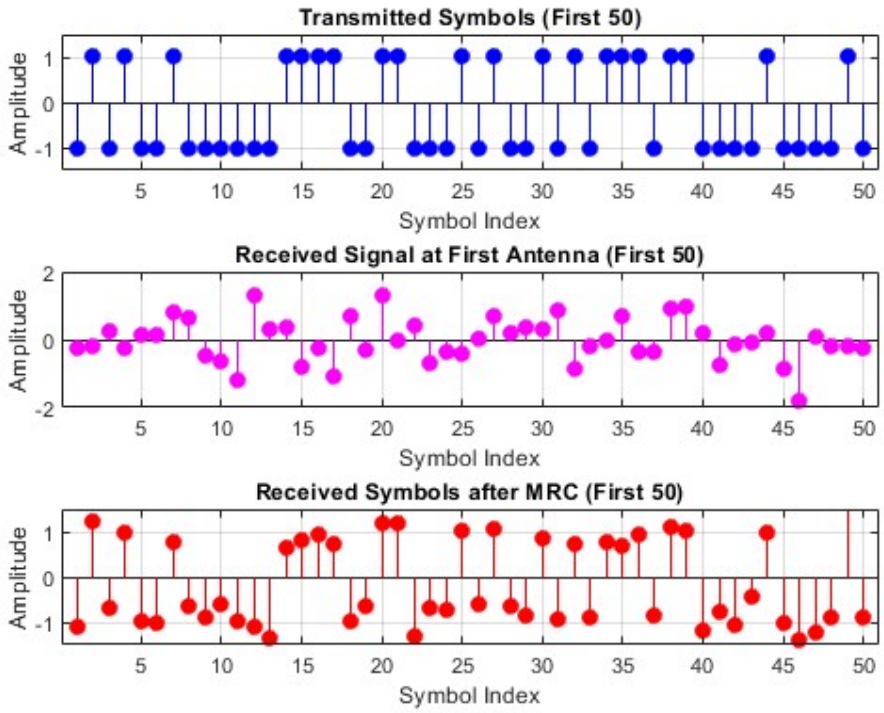
PROGRAM :

```
% Wireless Channel Estimation and Diversity Combining using MRC
clc; clear; close all;

% Parameters
N = 1000; % Number of symbols
SNR_dB = 10; % Signal-to-Noise Ratio in dB
SNR = 10^(SNR_dB/10); % Linear SNR
M = 2; % Number of receive antennas

% Generate BPSK symbols
symbols = 2 * randi([0, 1], 1, N) - 1;

% Create a Rayleigh fading channel with M receive antennas
h = (randn(M, N) + 1j * randn(M, N)) / sqrt(2);

% Pass symbols through the fading channel with noise
noise = (1/sqrt(SNR)) * (randn(M, N) + 1j * randn(M, N));
received = h .* symbols + noise;

% Channel Estimation (Assuming Perfect Knowledge)
h_est = h;

% Apply Maximal Ratio Combining (MRC)
combined_signal = sum(conj(h_est) .* received, 1) ./ sum(abs(h_est).^2, 1);
```

```
% Decision: BPSK Demodulation
decoded_symbols = real(combined_signal) > 0;

% Calculate Bit Error Rate (BER)
BER = sum(decoded_symbols ~= (symbols > 0)) / N;
disp(['Bit Error Rate (BER): ', num2str(BER)]);

% Plot transmitted vs received signal
figure;
subplot(3,1,1);
stem(symbols(1:50), 'bo', 'filled'); grid on;
title('Transmitted Symbols (First 50)'); xlabel('Symbol Index'); ylabel('Amplitude');
ylim([-1.5, 1.5]);

subplot(3,1,2);
stem(real(received(1,1:50)), 'mo', 'filled'); grid on;
title('Received Signal at First Antenna (First 50)'); xlabel('Symbol Index'); ylabel('Amplitude');
ylim([-2, 2]);

subplot(3,1,3);
stem(real(combined_signal(1:50)), 'ro', 'filled'); grid on;
title('Received Symbols after MRC (First 50)'); xlabel('Symbol Index'); ylabel('Amplitude');
ylim([-1.5, 1.5]);
```

OUTPUT:

Bit Error Rate (BER): 0.006



RESULT:

Wireless channel estimation and diversity combining were successfully implemented in MATLAB, and the output was verified.