Facility of Engineering – Cairo University

Communication Project - Single Carrier System

Submitted to:

Dr. Mohamed M. Khairy
Eng. Mohamed Wael

Submitted by:

Name	Section	BN
Abdulrahman Elsayed	3	05

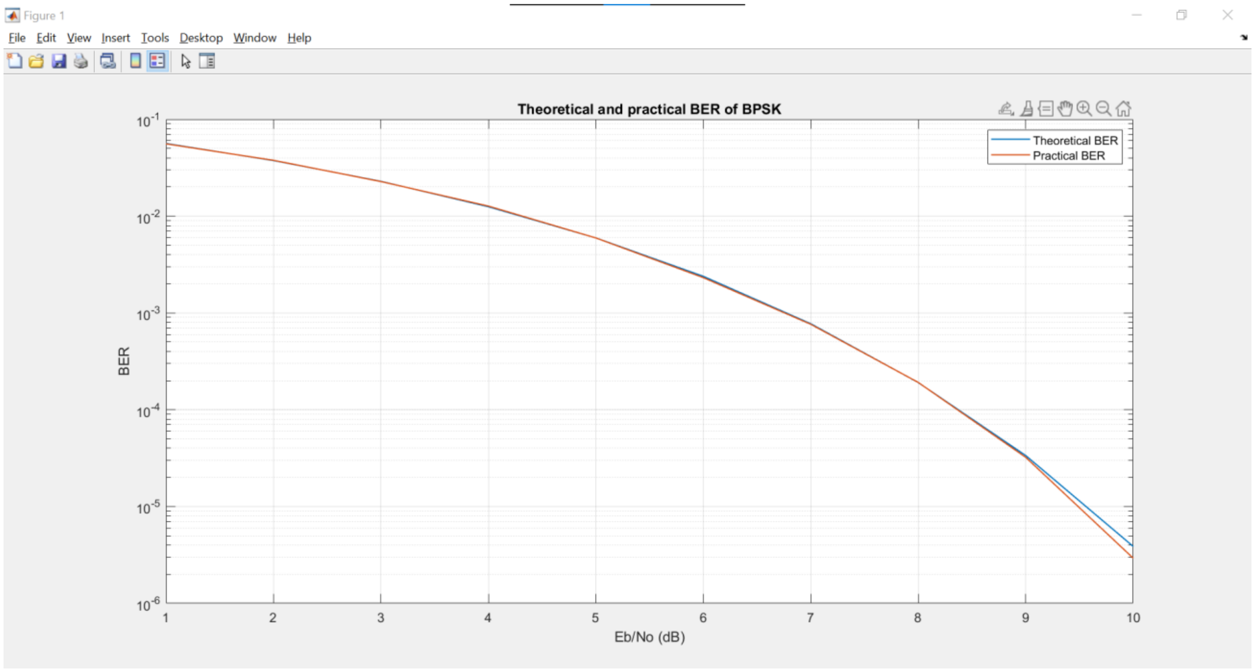
Contents

BPSK

Code

```
% Transmitter
data_length = 1024000;
data_Tx = randi([0, 1], 1, data_length);
BPSK_Tx = data_Tx * 2 - 1; % mapping

% Channel and Receiver
Eb_BPSK = ((1 + 1) / 2) / 1; % Eb = avg symbol energy / # symbols
quotient_range = 10; % range of Eb/No trials
theoritical_BER = zeros(1, quotient_range);
practical_BER = zeros(1, quotient_range);

for quotient = 1: quotient_range
    % Channel
    No = Eb_BPSK / (10^(quotient / 10));  % dB to linear units conversion
    AWGN = sqrt(No / 2) * randn(1, length(BPSK_Tx));

    % Receiver
    BPSK_Rx = BPSK_Tx + AWGN;
    data_Rx = zeros(1, data_length);

    % demapping
    for i = 1: length(BPSK_Rx)
        if (BPSK_Rx(i) <= 0) % decision boundary
            data_Rx(i) = 0;
        else
            data_Rx(i) = 1;
        end
    end

    % BER
    theoritical_BER(quotient) = 0.5 * erfc(sqrt(1 / No));
    [number, ratio] = symerr(data_Tx, data_Rx);
    practical_BER(quotient) = ratio;
end

% plotting
x_axis = 1: quotient_range;
semilogy(x_axis, theoritical_BER, x_axis, practical_BER, 'LineWidth', 1)
grid on
legend('Theoretical BER', 'Practical BER')
xlabel('Eb/No (dB)');
ylabel('BER');
title('Theoretical and practical BER of BPSK')
```

Outputs



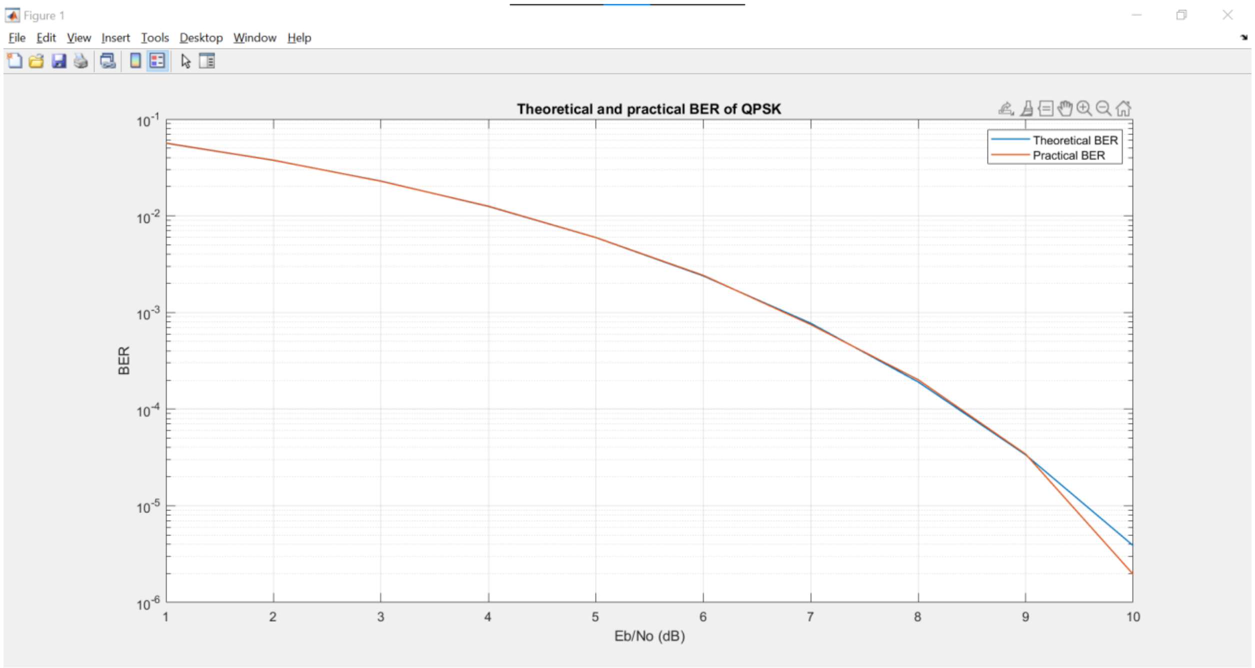
Theoretical and practical BER of BPSK

QPSK

Code

```
% Transmitter
data_length = 1024000;
data_Tx = randi([0, 1], 1, data_length);
PSK4_Tx = zeros(1, floor(data_length / 2));

% mapping
for i = 1: 2: data_length - mod(data_length, 2)
    if (data_Tx(i: i + 1) == [0, 0])
        PSK4_Tx(floor(i / 2) + 1) = -1 - 1i;
    elseif (data_Tx(i: i + 1) == [0, 1])
        PSK4_Tx(floor(i / 2) + 1) = -1 + 1i;
    elseif (data_Tx(i: i + 1) == [1, 0])
        PSK4_Tx(floor(i / 2) + 1) = 1 - 1i;
    elseif (data_Tx(i: i + 1) == [1, 1])
        PSK4_Tx(floor(i / 2) + 1) = 1 + 1i;
    end
end

% Channel and Receiver
Eb_PSK4 = ((4 * 2) / 4) / 2; % Eb = avg symbol energy / # symbols
quotient_range = 10; % range of Eb/No trials
theoritical_BER = zeros(1, quotient_range);
practical_BER = zeros(1, quotient_range);
for quotient = 1: quotient_range
    % Channel
    No = Eb_PSK4 / (10^(quotient / 10)); % dB to linear units conversion;
    AWGN = sqrt(No / 2) * randn(1, length(PSK4_Tx)) + 1i * sqrt(No / 2) ...
        * randn(1, length(PSK4_Tx));

    % Receiver
    PSK4_Rx = PSK4_Tx + AWGN;
    data_Rx = zeros(1, data_length);
    j = 1;

    % demapping
    for i = 1: length(PSK4_Rx)
        if (real(PSK4_Rx(i)) <= 0) % Q decision boundary
            if (imag(PSK4_Rx(i)) <= 0) % I decision boundary
                data_Rx(j: j + 1) = [0, 0];
            elseif (imag(PSK4_Rx(i)) > 0)
                data_Rx(j: j + 1) = [0, 1];
            end
        elseif (real(PSK4_Rx(i)) > 0)
            if (imag(PSK4_Rx(i)) <= 0)
                data_Rx(j: j + 1) = [1, 0];
            elseif (imag(PSK4_Rx(i)) > 0)
                data_Rx(j: j + 1) = [1, 1];
            end
            end
            j = j + 2;
        end

        % BER
        theoritical_BER(quotient) = erfc(sqrt(Eb_PSK4 / No)) / 2;
        [number, ratio] = symerr(data_Tx, data_Rx);
        practical_BER(quotient) = ratio;
end

% plotting
x_axis = 1: quotient_range;
semilogy(x_axis, theoritical_BER, x_axis, practical_BER, 'LineWidth', 1)
grid on
legend('Theoretical BER', 'Practical BER')
xlabel('Eb/No (dB)');
ylabel('BER');
title('Theoretical and practical BER of QPSK')
```

Output



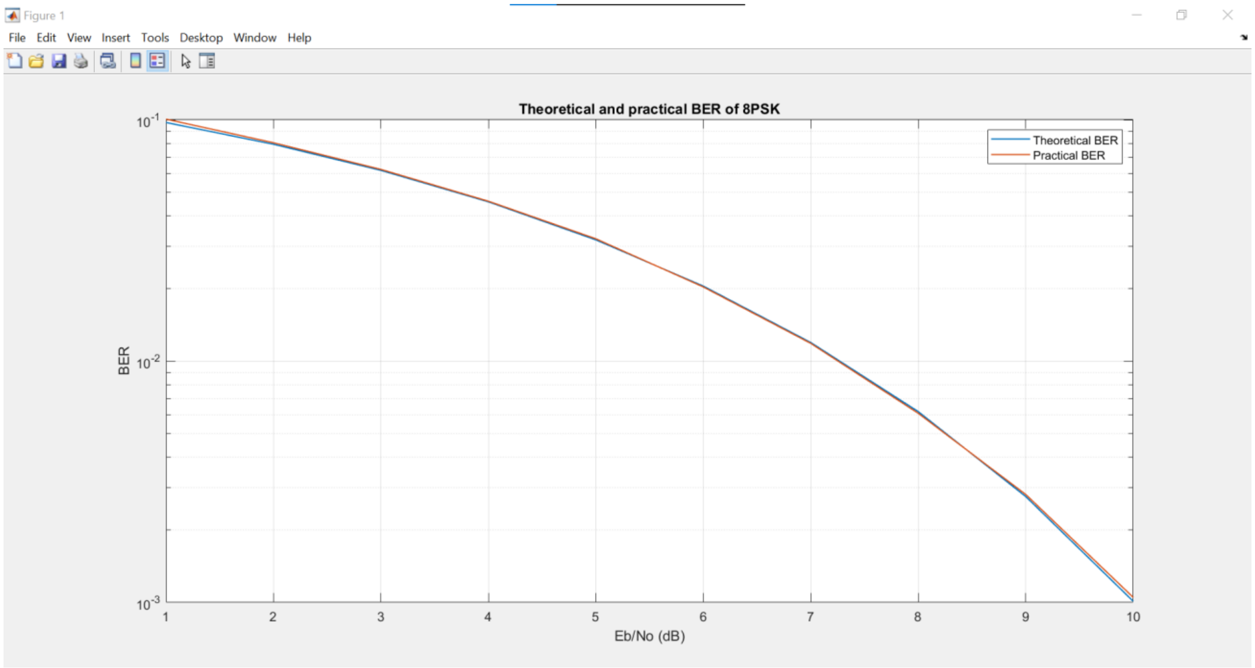
Theoretical and practical BER of QPSK

8PSK

Code

```
% Transmitter
data_length = 1024000;
data_Tx = randi([0, 1], 1, data_length);
PSK8_Tx = zeros(1, floor(data_length / 3));

%mapping
for i = 1: 3: data_length - mod(data_length, 3)
    if (data_Tx(i: i + 2) == [0, 0, 0])
        PSK8_Tx(floor(i / 3) + 1) = cos(0) + 1i * sin(0);
    elseif (data_Tx(i: i + 2) == [0, 0, 1])
        PSK8_Tx(floor(i / 3) + 1) = cos(pi / 4) + 1i * sin(pi / 4);
    elseif (data_Tx(i: i + 2) == [0, 1, 0])
        PSK8_Tx(floor(i / 3) + 1) = cos(3 * pi / 4) + 1i * sin(3 * pi / 4);
    elseif (data_Tx(i: i + 2) == [0, 1, 1])
        PSK8_Tx(floor(i / 3) + 1) = cos(pi / 2) + 1i * sin(pi / 2);
    elseif (data_Tx(i: i + 2) == [1, 0, 0])
        PSK8_Tx(floor(i / 3) + 1) = cos(7 * pi / 4) + 1i * sin(7 * pi / 4);
    elseif (data_Tx(i: i + 2) == [1, 0, 1])
        PSK8_Tx(floor(i / 3) + 1) = cos(3 * pi / 2) + 1i * sin(3 * pi / 2);
    elseif (data_Tx(i: i + 2) == [1, 1, 0])
        PSK8_Tx(floor(i / 3) + 1) = cos(pi) + 1i * sin(pi);
    elseif (data_Tx(i: i + 2) == [1, 1, 1])
        PSK8_Tx(floor(i / 3) + 1) = cos(5 * pi / 4) + 1i * sin(5 * pi / 4);
    end
end

% Channel and Receiver
Eb_PSK8 = 1 / 3; % Eb = avg symbol energy / # symbols
quotient_range = 10; % range of Eb/No trials
theoritical_BER = zeros(1, quotient_range);
practical_BER = zeros(1, quotient_range);

for quotient = 1: quotient_range
    % Channel
    No = Eb_PSK8 / (10^(quotient / 10)); % dB to linear units conversion;
    AWGN = sqrt(No / 2) * randn(1, length(PSK8_Tx)) + 1i * sqrt(No / 2) ...
        * randn(1, length(PSK8_Tx));

    % Receiver
    PSK8_Rx = PSK8_Tx + AWGN;
    data_Rx = zeros(1, data_length);
    j = 1;

    % demapping
    for i = 1: length(PSK8_Rx)
        if ((angle(PSK8_Rx(i)) >= -pi / 8) && (angle(PSK8_Rx(i)) < pi / 8))
            data_Rx(j: j + 2) = [0, 0, 0];
        elseif ((angle(PSK8_Rx(i)) >= pi / 8) && ...
                (angle(PSK8_Rx(i)) < 3 * pi / 8))
            data_Rx(j: j + 2) = [0, 0, 1];
        elseif ((angle(PSK8_Rx(i)) >= 3 * pi / 8) ...
                && (angle(PSK8_Rx(i)) < 5 * pi / 8))
            data_Rx(j: j + 2) = [0, 1, 1];
        elseif ((angle(PSK8_Rx(i)) >= 5 * pi / 8) ...
                && (angle(PSK8_Rx(i)) < 7 * pi / 8))
            data_Rx(j: j + 2) = [0, 1, 0];
        elseif (((angle(PSK8_Rx(i)) >= 7 * pi / 8) ...
                && (angle(PSK8_Rx(i)) < pi)) ...
                || ((angle(PSK8_Rx(i)) >= - pi) ...
                && (angle(PSK8_Rx(i)) < -7 * pi / 8)))
            data_Rx(j: j + 2) = [1, 1, 0];
        elseif ((angle(PSK8_Rx(i)) >= -7 * pi / 8) ...
                && (angle(PSK8_Rx(i)) < -5 * pi / 8))
            data_Rx(j: j + 2) = [1, 1, 1];
        elseif ((angle(PSK8_Rx(i)) >= -5 * pi / 8) ...
                && (angle(PSK8_Rx(i)) < -3 * pi / 8))
            data_Rx(j: j + 2) = [1, 0, 1];
        elseif ((angle(PSK8_Rx(i)) >= -3 * pi / 8) ...
                && (angle(PSK8_Rx(i)) < -pi / 8))
            data_Rx(j: j + 2) = [1, 0, 0];
        end
        j = j + 3;
    end

    % BER
    theoritical_BER(quotient) = erfc(sqrt(1 / No) * sin(pi / 8)) / 3;
    [number, ratio] = symerr(data_Tx, data_Rx);
    practical_BER(quotient) = ratio;
end

% plotting
x_axis = 1: quotient_range;
semilogy(x_axis, theoritical_BER, x_axis, practical_BER, 'LineWidth', 1)
grid on
legend('Theoretical BER', 'Practical BER')
xlabel('Eb/No (dB)');
ylabel('BER');
title('Theoretical and practical BER of 8PSK')
```

Output



Theoretical and practical BER of 8PSK

16QAM

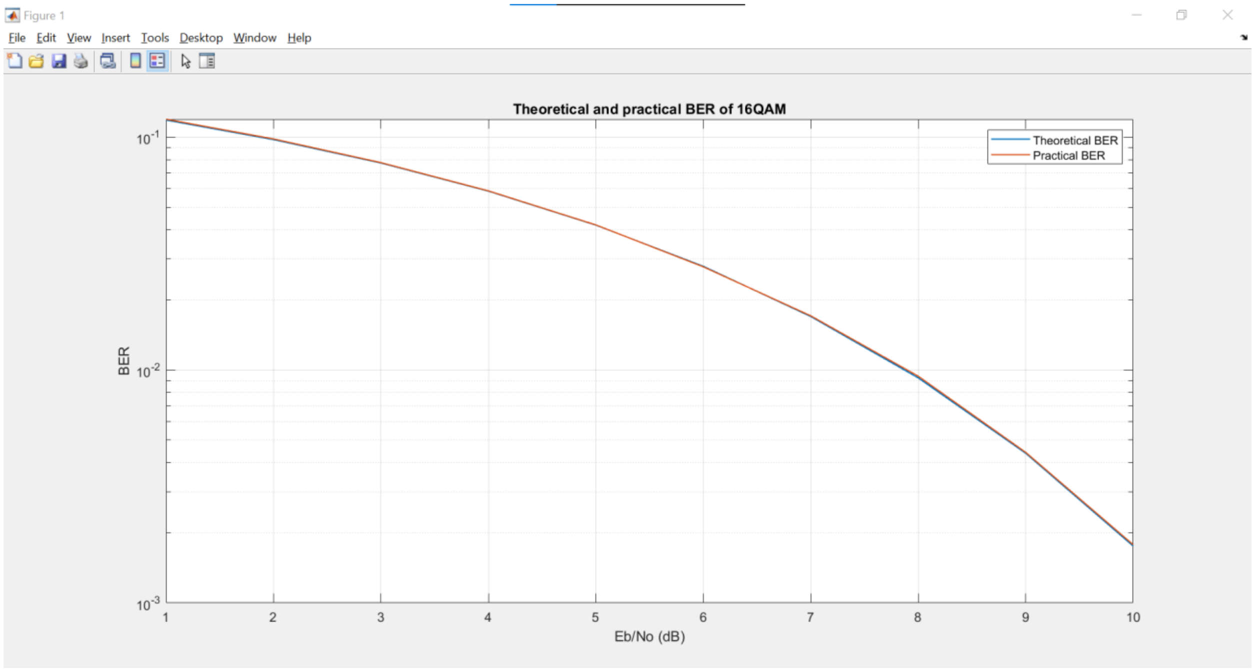
Code

```
% Transmitter
data_length = 1024000;
data_Tx = randi([0, 1], 1, data_length);
QAM16_Tx = zeros(1, floor(data_length / 4));

% mapping
for i = 1: 4: data_length - mod(data_length, 4)
    if (data_Tx(i: i + 3) == [0, 0, 0, 0])
        QAM16_Tx(floor(i / 4) + 1) = -3 - 3i;
    elseif (data_Tx(i: i + 3) == [0, 0, 0, 1])
        QAM16_Tx(floor(i / 4) + 1) = -3 - 1i;
    elseif (data_Tx(i: i + 3) == [0, 0, 1, 0])
        QAM16_Tx(floor(i / 4) + 1) = -3 + 3i;
    elseif (data_Tx(i: i + 3) == [0, 0, 1, 1])
        QAM16_Tx(floor(i / 4) + 1) = -3 + 1i;

    elseif (data_Tx(i: i + 3) == [0, 1, 0, 0])
        QAM16_Tx(floor(i / 4) + 1) = -1 - 3i;
    elseif (data_Tx(i: i + 3) == [0, 1, 0, 1])
        QAM16_Tx(floor(i / 4) + 1) = -1 - 1i;
    elseif (data_Tx(i: i + 3) == [0, 1, 1, 0])
        QAM16_Tx(floor(i / 4) + 1) = -1 + 3i;
    elseif (data_Tx(i: i + 3) == [0, 1, 1, 1])
        QAM16_Tx(floor(i / 4) + 1) = -1 + 1i;

    elseif (data_Tx(i: i + 3) == [1, 0, 0, 0])
        QAM16_Tx(floor(i / 4) + 1) = 3 - 3i;
    elseif (data_Tx(i: i + 3) == [1, 0, 0, 1])
        QAM16_Tx(floor(i / 4) + 1) = 3 - 1i;
    elseif (data_Tx(i: i + 3) == [1, 0, 1, 0])
        QAM16_Tx(floor(i / 4) + 1) = 3 + 3i;
    elseif (data_Tx(i: i + 3) == [1, 0, 1, 1])
        QAM16_Tx(floor(i / 4) + 1) = 3 + 1i;

    elseif (data_Tx(i: i + 3) == [1, 1, 0, 0])
        QAM16_Tx(floor(i / 4) + 1) = 1 - 3i;
    elseif (data_Tx(i: i + 3) == [1, 1, 0, 1])
        QAM16_Tx(floor(i / 4) + 1) = 1 - 1i;
    elseif (data_Tx(i: i + 3) == [1, 1, 1, 0])
        QAM16_Tx(floor(i / 4) + 1) = 1 + 3i;
    elseif (data_Tx(i: i + 3) == [1, 1, 1, 1])
        QAM16_Tx(floor(i / 4) + 1) = 1 + 1i;
    end
end

% Channel and Receiver
Eb_QAM16 = ((4 * 2 + 4 * 18 + 8 * 10) / 16) / 4; % Eb = avg symbol energy / # symbols
```

```
quotient_range = 10; % range of Eb/No trials
theoritical_BER = zeros(1, quotient_range);
practical_BER = zeros(1, quotient_range);

for quotient = 1: quotient_range
    % Channel
    No = Eb_QAM16 / (10^(quotient / 10)); % dB to linear units conversion;
    AWGN = sqrt(No / 2) * randn(1, length(QAM16_Tx)) + 1i * sqrt(No / 2) *
randn(1, length(QAM16_Tx));

    % Receiver
    QAM16_Rx = QAM16_Tx + AWGN;
    data_Rx = zeros(1, data_length);
    j = 1;

    % demapping
    for i = 1: length(QAM16_Rx)
        if (real(QAM16_Rx(i)) <= -2)
            if (imag(QAM16_Rx(i)) >= 2)
                data_Rx(j: j + 3) = [0, 0, 1, 0];
            elseif ((imag(QAM16_Rx(i)) >= 0) && (imag(QAM16_Rx(i)) < 2))
                data_Rx(j: j + 3) = [0, 0, 1, 1];
            elseif ((imag(QAM16_Rx(i)) >= -2) && (imag(QAM16_Rx(i)) < 0))
                data_Rx(j: j + 3) = [0, 0, 0, 1];
            elseif (imag(QAM16_Rx(i)) < -2)
                data_Rx(j: j + 3) = [0, 0, 0, 0];
            end
        elseif ((real(QAM16_Rx(i)) > -2) && (real(QAM16_Rx(i)) <= 0))
            if (imag(QAM16_Rx(i)) >= 2)
                data_Rx(j: j + 3) = [0, 1, 1, 0];
            elseif ((imag(QAM16_Rx(i)) >= 0) && (imag(QAM16_Rx(i)) < 2))
                data_Rx(j: j + 3) = [0, 1, 1, 1];
            elseif ((imag(QAM16_Rx(i)) >= -2) && (imag(QAM16_Rx(i)) < 0))
                data_Rx(j: j + 3) = [0, 1, 0, 1];
            elseif (imag(QAM16_Rx(i)) < -2)
                data_Rx(j: j + 3) = [0, 1, 0, 0];
            end
        elseif ((real(QAM16_Rx(i)) > -0) && (real(QAM16_Rx(i)) <= 2))
            if (imag(QAM16_Rx(i)) >= 2)
                data_Rx(j: j + 3) = [1, 1, 1, 0];
            elseif ((imag(QAM16_Rx(i)) >= 0) && (imag(QAM16_Rx(i)) < 2))
                data_Rx(j: j + 3) = [1, 1, 1, 1];
            elseif ((imag(QAM16_Rx(i)) >= -2) && (imag(QAM16_Rx(i)) < 0))
                data_Rx(j: j + 3) = [1, 1, 0, 1];
            elseif (imag(QAM16_Rx(i)) < -2)
                data_Rx(j: j + 3) = [1, 1, 0, 0];
            end
        elseif (real(QAM16_Rx(i)) > 2)
            if (imag(QAM16_Rx(i)) >= 2)
                data_Rx(j: j + 3) = [1, 0, 1, 0];
            elseif ((imag(QAM16_Rx(i)) >= 0) && (imag(QAM16_Rx(i)) < 2))
                data_Rx(j: j + 3) = [1, 0, 1, 1];
            elseif ((imag(QAM16_Rx(i)) >= -2) && (imag(QAM16_Rx(i)) < 0))
```

```
                data_Rx(j: j + 3) = [1, 0, 0, 1];
            elseif (imag(QAM16_Rx(i)) < -2)
                data_Rx(j: j + 3) = [1, 0, 0, 0];
            end
        end
        j = j + 4;
    end

    % BER
    theoritical_BER(quotient) = 1.5 * erfc(sqrt(Eb_QAM16 / (2.5 * No))) / 4;
    [number, ratio] = symerr(data_Tx, data_Rx);
    practical_BER(quotient) = ratio;
end

% plotting
x_axis = 1: quotient_range;
semilogy(x_axis, theoritical_BER, x_axis, practical_BER, 'LineWidth', 1)
grid on
legend('Theoretical BER', 'Practical BER')
xlabel('Eb/No (dB)');
ylabel('BER');
title('Theoretical and practical BER of 16QAM')
```

Output



Theoretical and practical BER of 16QAM