Ain Shams University
Faculty of Engineering
Electronics & Comm. Dept.

ECE 351
3rd Year Electrical Engineering
Fall 2023

Project Report

Name	ID	Sec.
Abanoub Emad Hanna Kamel	2002075	2
Yousif Ahmed Ali Mohamed	2001902	2
Mina Wafik Halim Mina	2101716	2
Omar Mahmoud ElSayed Gabr	2000476	4
Aly Ayman Ibrahim	2000921	2
George Nabil Henry Assaad	2000086	3

Team Members Contribution

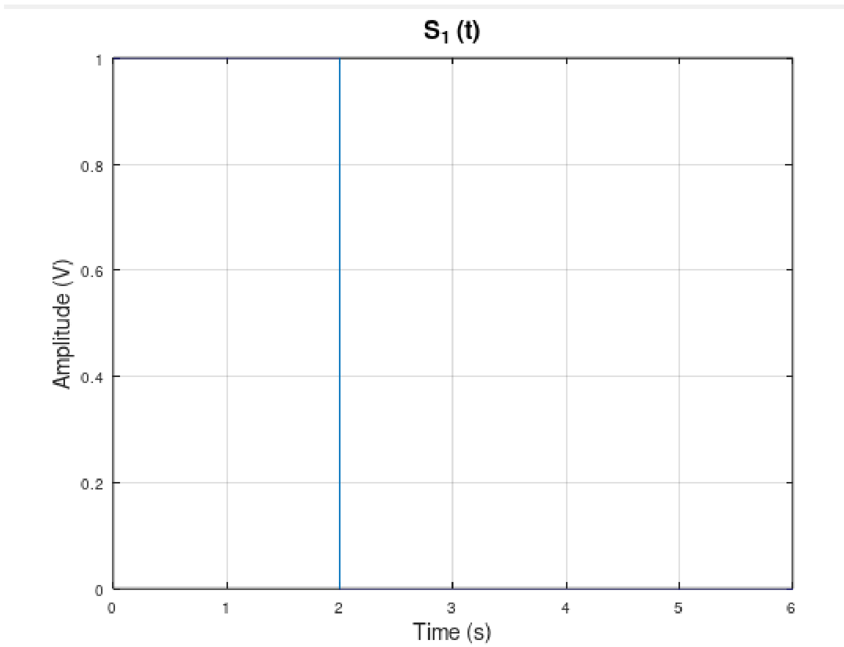
- **Abanoub Emad:** Writing the code for basis function & constellation diagram
- **Yousif Ahmed:** Writing the code for noise sweep function & complex constellation diagram.
- **Omar Gabr:** Writing the code for decision device function & PNRZ encoding.
- **Mina Wafik:** Energy calculation & signal component functions.
- **Aly Ayman:** BER function, signal plots and writing the report.
- **George Nabil:** Theoretical BER (code and report), initialization & sampling.

Part 1

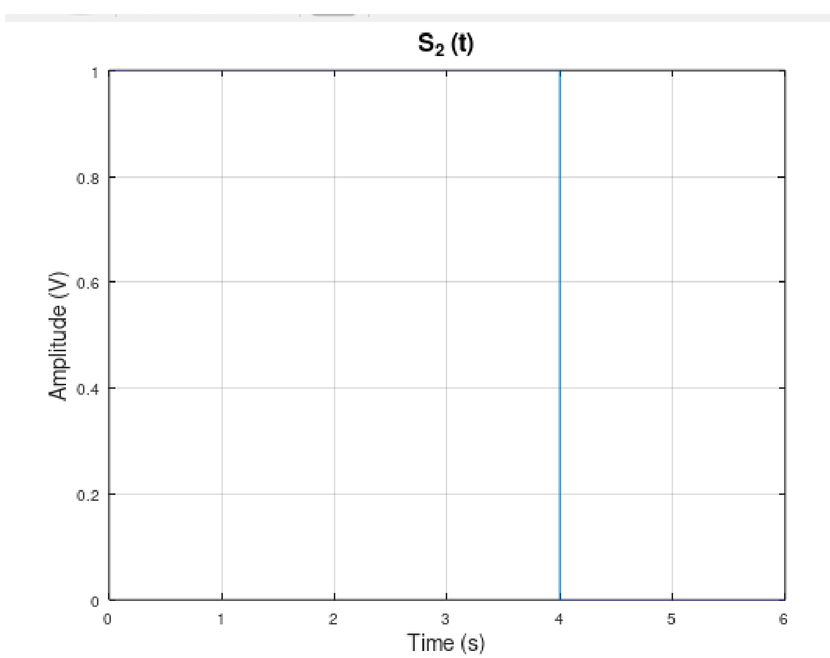
Example 1

Note: Assume T = 6s.

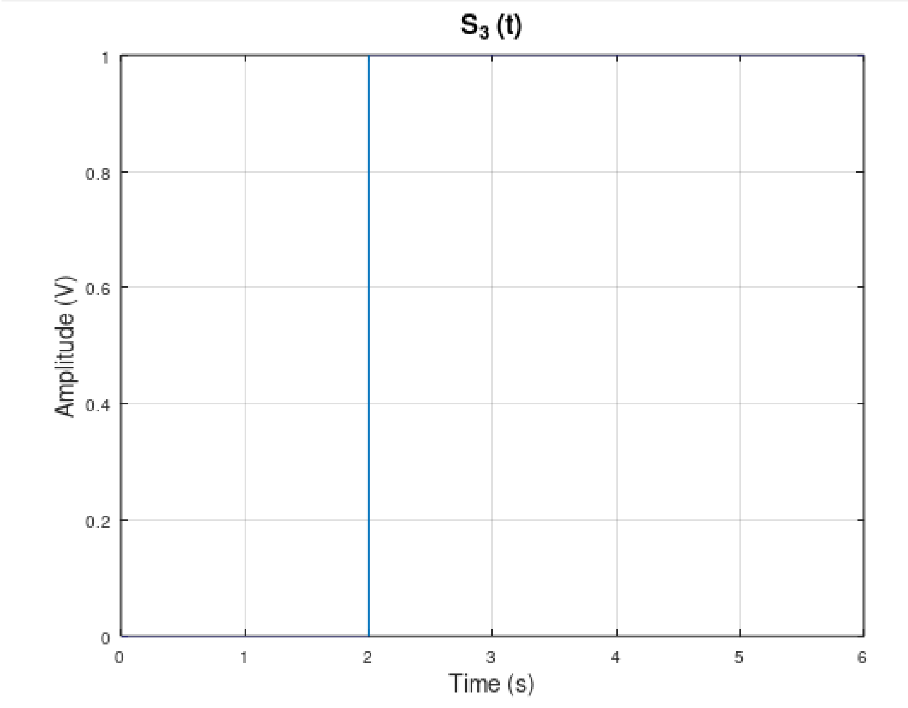
Signal 1



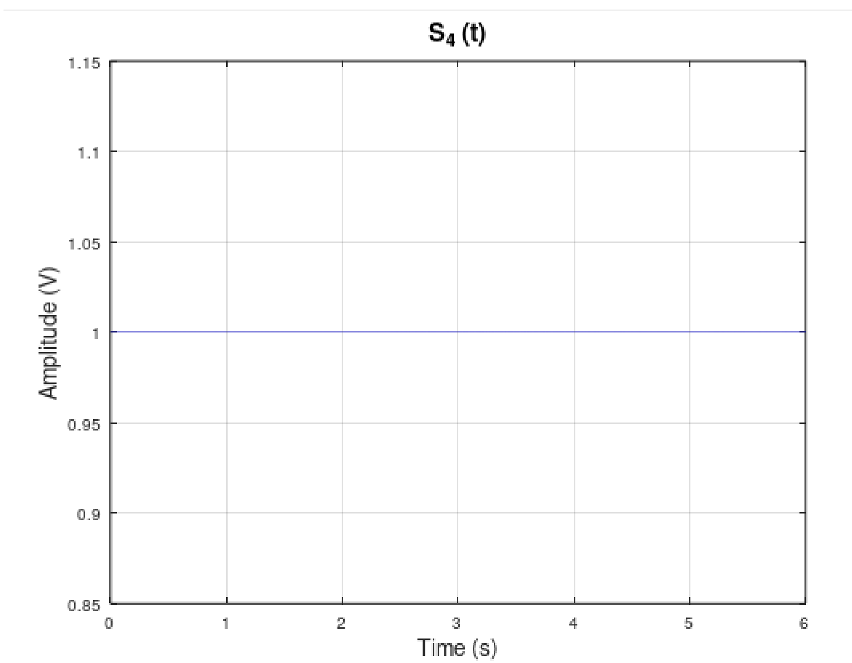
Signal 2



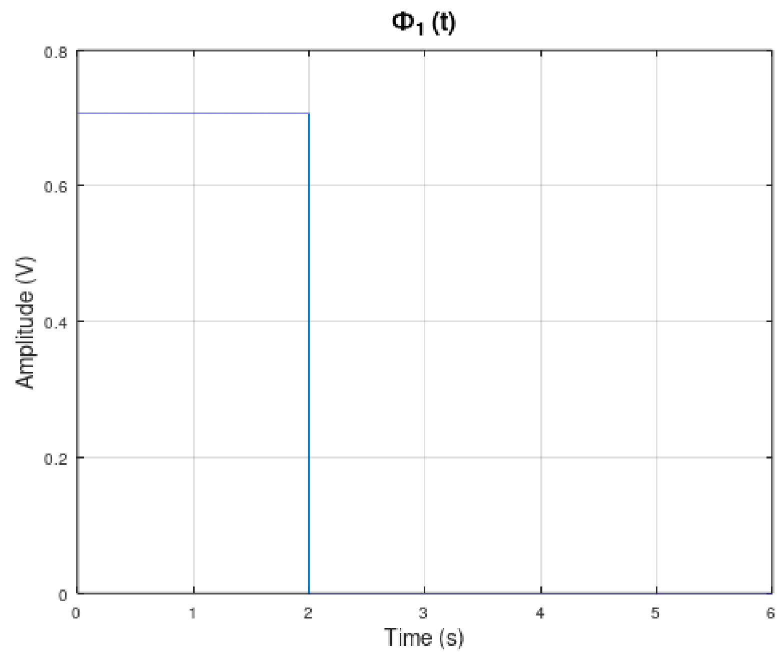
Signal 3



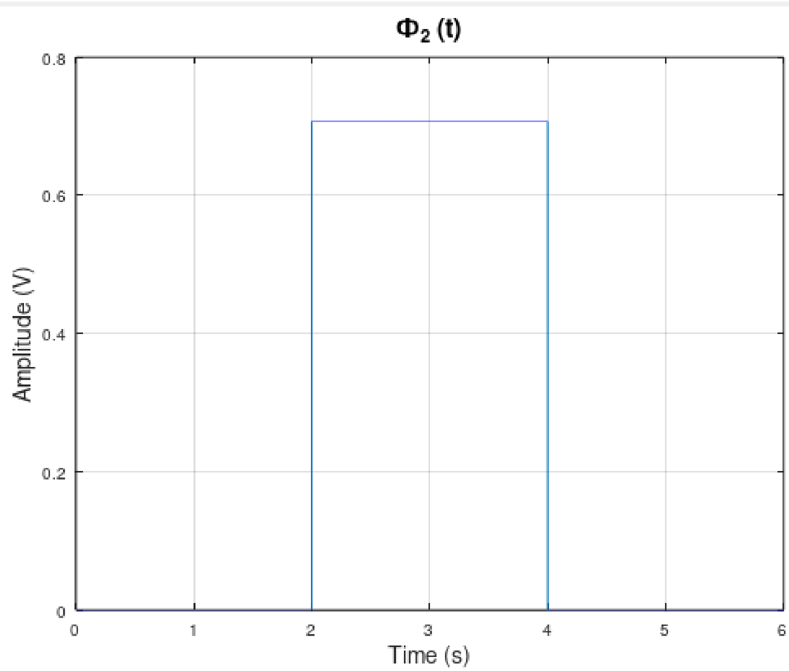
Signal 4



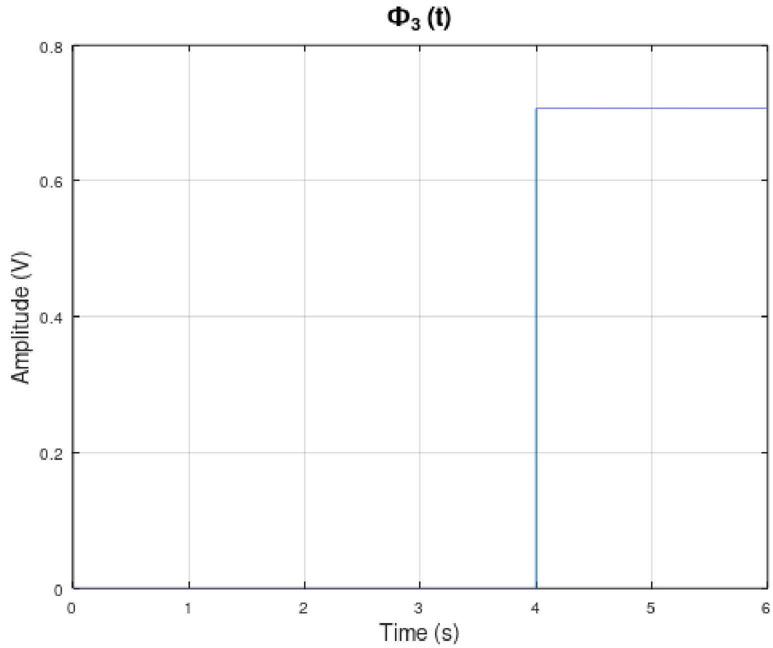
Phi 1


Φ_1 (t)
0.8
0.6
0.4
0.2
0
Amplitude (V)
0
1
2
3
4
5
6
Time (s)

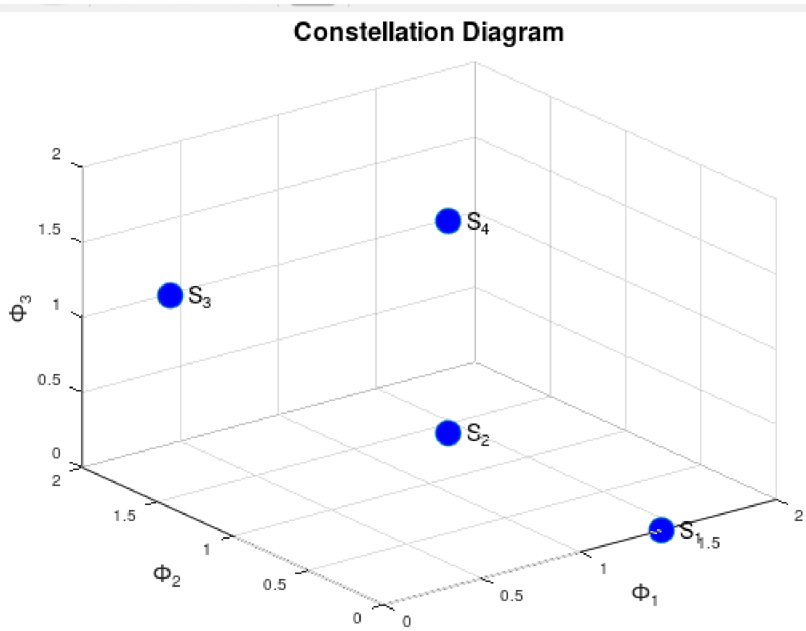
Phi 2


Φ_2 (t)
0.8
0.6
0.4
0.2
0
Amplitude (V)
0
1
2
3
4
5
6
Time (s)

Phi 3



Constellation Diagram

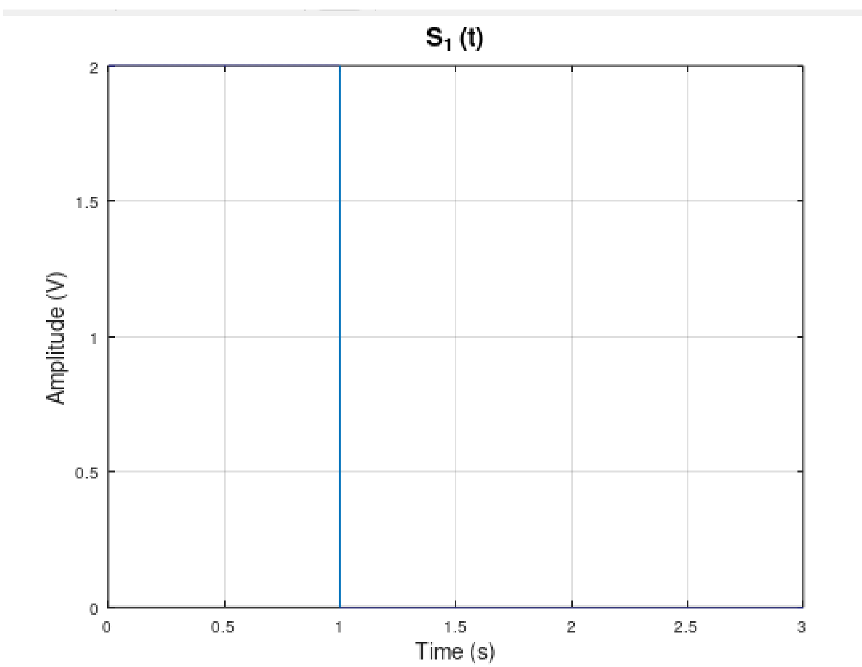


Symbol Energy

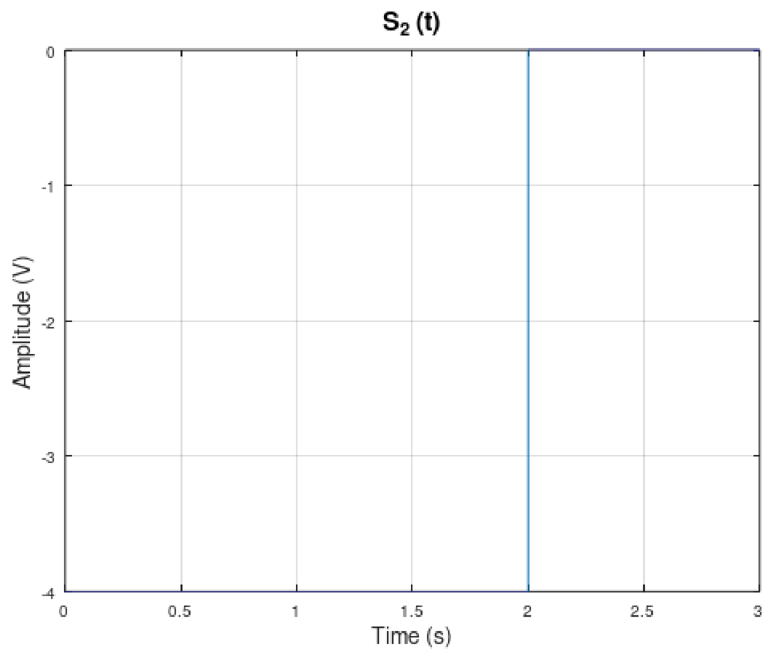
```
Command Window
How many signals do you want to input?
4
Input Signal period
6
Input Signal matrix
[2 6; 1 0]
Input Signal matrix
[4 6; 1 0]
Input Signal matrix
[2 6; 0 1]
Input Signal matrix
[6;1]
symbol_energy =

    2
    4
    4
    6

>> |
```

Example 2

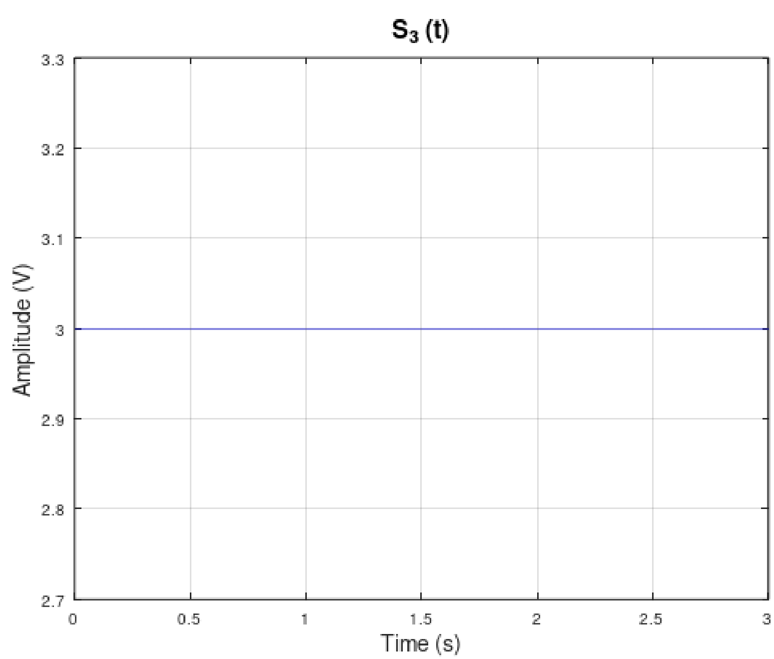
Signal 1



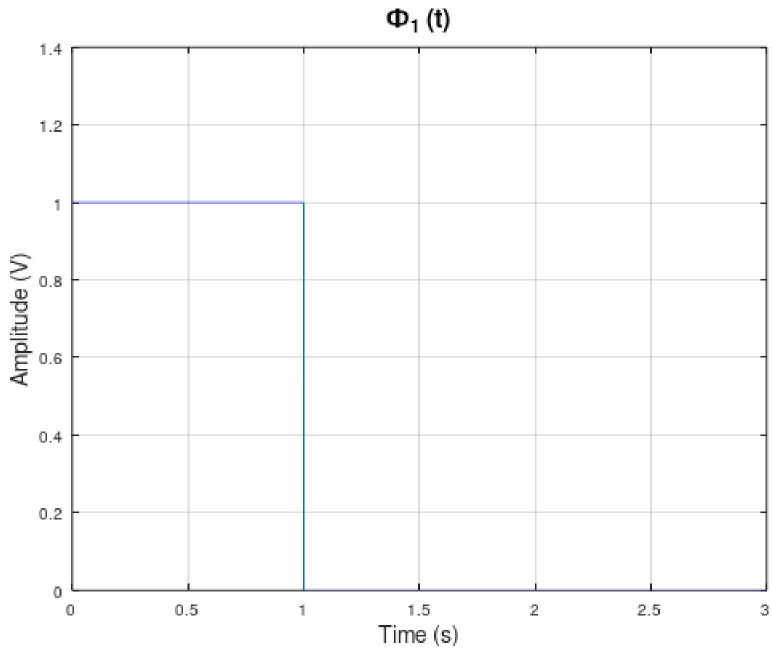
Signal 2



Signal 3

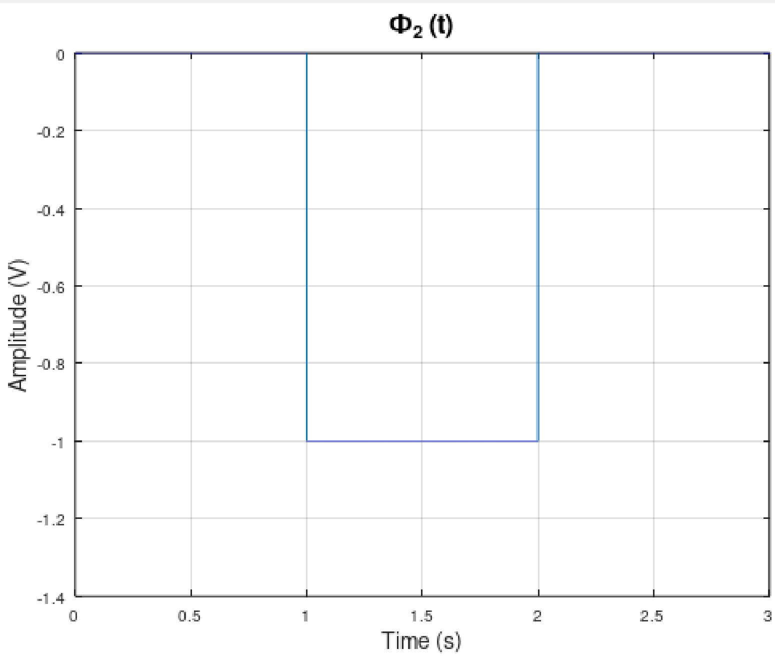


Phi 1



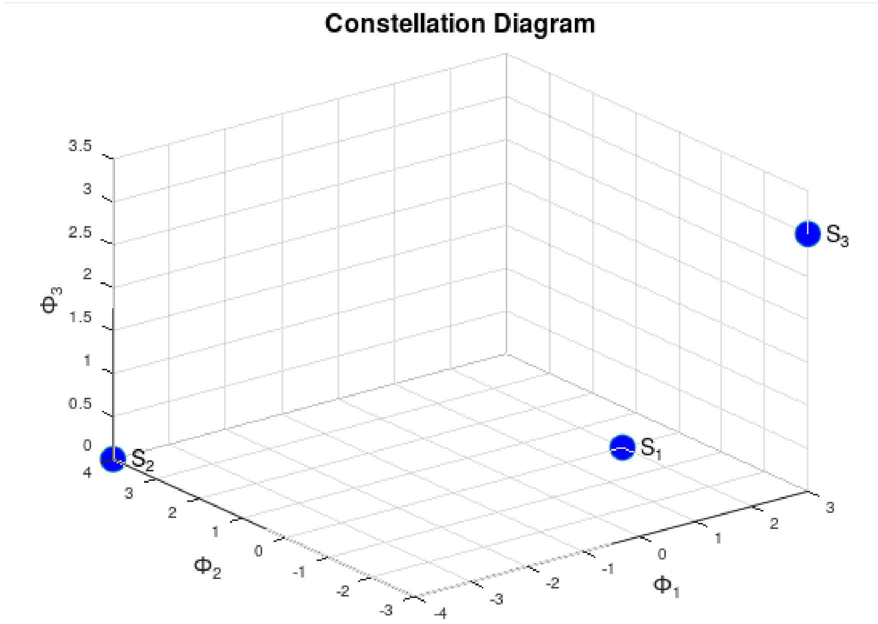
Φ1 (t)
Amplitude (V)
Time (s)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3

Phi 2



Φ2 (t)
Amplitude (V)
Time (s)
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
0
0.5
1
1.5
2
2.5
3

Constellation Diagram



Symbol Energy

```
Command Window
How many signals do you want to input?
3
Input Signal period
3
Input Signal matrix
[1 3; 2 0]
Input Signal matrix
[2 3; -4 0]
Input Signal matrix
[3;3]
symbol_energy =

    4
   32
   27
```

Note: Both codes for part 1 & part 2 are Octave codes. For MATLAB code (click here).

Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialization %%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear
clc
pkg load communications

%% Sampling frequency
global fs = 20000;             % 20KHz
global ts = 1/fs;              % sampling

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

%%%%%%%%%%%%%%%%%%%%%%%%%% Sampling Function %%%%%%%%%%%%%%%%%%%%%%%%%

function sampled_vector = sample(input_vector)
    global ts;
    y = [];                         % vector of amplitudes at any t
    n = size(input_vector, 2);      % n represents the number of
    t_start = 0;                    % different amplitudes in the pulse
    k = 1;                          % loop iterator
    for i=1:n
        t_end = input_vector(1,i);
        for j=t_start: ts : t_end - ts
            y(k) = input_vector(2,i);
            k = k + 1;
        endfor
        t_start = t_end;
    endfor
    sampled_vector = y;
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Energy Calc. %%%%%%%%%%%%%%%%%%%%%%%%%%%%

function energy = CalcEnergy(x, T)
    n = length(x);
    delta = 1/n;
    eg = 0;
    for i = 1:n
          eg = eg + x(i)^2 * delta;
    endfor
    energy = eg * T;
end

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

```
%%%%%%%%%%%%%%%%%%%%%%%%%% Signal Component %%%%%%%%%%%%%%%%%%%%%%%%%%

function coefficient = CalcCoefficient(s, phi, T)
    n = length(s);
    delta = 1/n;
    coef = 0;
    for i=1:n
        coef = coef + s(i) * phi(i) * delta;
    endfor
    coefficient = coef * T;
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%% Basis Functions %%%%%%%%%%%%%%%%%%%%%%%%%%%

function sig_comp = basis(input_signals, T)
    global fs;
    global ts;
    len = T * fs;
    t = 0 : ts : (T-ts);    % time vector
    M = length(input_signals);
    basis_vectors = {};      % cell array of basis functions
    phi = [];      % Initialization of phi amplitude vector
    c = [];          % Initialization of coefficients vector

    %% loop through signals
    for num=1:M
        flag = 0;                 % flag to check wether g = 0 or not
        N = length(basis_vectors);
        y = input_signals{num};  % amplitude vector of signal

        %% calculating basis function
        if num == 1                  % num represents the signal number
            energy = CalcEnergy(input_signals{1}, T);
            for i = 1:len
                phi(i) = y(i) / sqrt(energy);
            endfor
            basis_vectors{1} = phi;
            c(1,1) = CalcCoefficient(input_signals{1}, basis_vectors{1}, T);
            %% Plot basis functions
            figure;
            plot(t, phi);
            grid on;
            title(['Φ_1 (t)'], 'FontSize', 16);
```

```
            xlabel('Time (s)', 'FontSize', 14);
            ylabel('Amplitude (V)', 'FontSize', 14);
        else
            sig = input_signals{num};
            g = sig;
            for i=1:N
                phi = basis_vectors{i};
                comp = CalcCoefficient(sig, phi, T);
                if abs(comp) < 1e-10
                    comp = 0;
                end
                c(num, i) = comp;
                g = g - comp * phi;
            endfor

            for i=1:len
                if abs(g(i)) < 1e-10    % elimnate any errors caused
                    g(i) = 0;          % from the subtraction
                endif                   % of floating numbers
                if g(i) != 0
                    flag = 1;
                endif
            endfor

            if flag == 1                % if g != 0
                energy = CalcEnergy(g, T);
                for i = 1:len
                    phi(i) = g(i) / sqrt(energy);
                endfor
                basis_vectors{N+1} = phi;
                c(num,num) = sqrt(energy);
                %% Plot basis functions
                figure;
                plot(t, phi);
                grid on;
                title(['Φ_' num2str(N+1) ' (t)'], 'FontSize', 16);
                xlabel('Time (s)', 'FontSize', 14);
                ylabel('Amplitude (V)', 'FontSize', 14);
            endif
        end
    endfor
    sig_comp = c;
end

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

```
%%%%%%%%%%%%%%%%%%%%%%% Constellation Diagram %%%%%%%%%%%%%%%%%%%%%%%%

function symbol_energy = constellation_diagram(matrix)

    % Extract coordinates
    [r,c] = size(matrix);
    x = matrix(:, 1);

    % Check if 3d or 2d or 1d
    if c == 3
        y = matrix(:, 2);
        z = matrix(:, 3);
        energy = x.^2 + y.^2 + z.^2;

        figure;
        plot3(x, y, z, 'o', 'MarkerSize', 8, 'MarkerFaceColor', 'b');
        grid on;
        title('Constellation Diagram', 'FontSize', 16);
        xlabel('Φ_1', 'FontSize', 14);
        ylabel('Φ_2', 'FontSize', 14);
        zlabel('Φ_3', 'FontSize', 14);

        % add labels for each point
        for i = 1:r
            text(x(i), y(i), z(i), sprintf('   S_%d', i),
                 'FontSize', 14);
        endfor

    elseif c == 2
        y = matrix(:, 2);
        energy = x.^2 + y.^2;

        % plot
        figure;
        plot(x, y, 'o', 'MarkerSize', 8, 'MarkerFaceColor', 'b');
        grid on;
        title('Constellation Diagram', 'FontSize', 16);
        xlabel('Φ_1', 'FontSize', 14);
        ylabel('Φ_2', 'FontSize', 14);

        % add labels for each point
        for i = 1:r
            text(x(i), y(i), sprintf('   S_%d', i), 'FontSize', 14);
        endfor
```

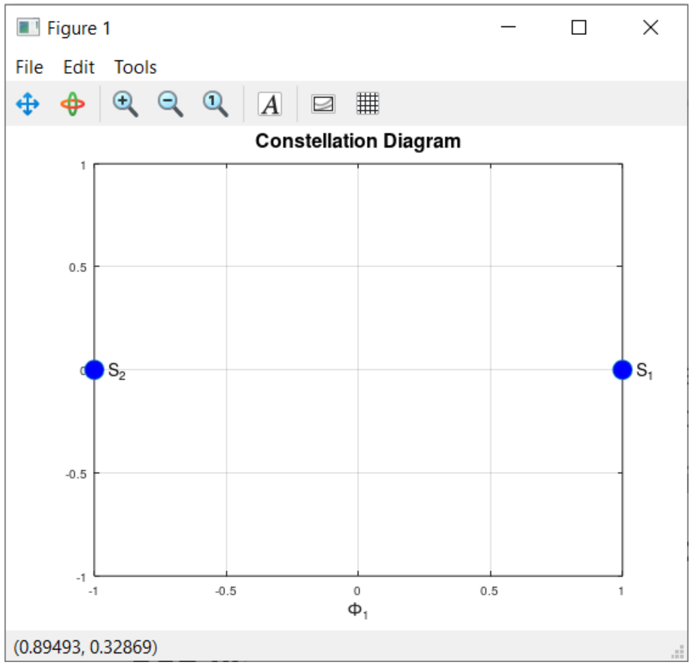
```
    else
        energy = x.^2;
        figure;
        plot(x, zeros(size(x)), 'o', 'MarkerSize', 8,
            'MarkerFaceColor', 'b');
        grid on;
        title('Constellation Diagram', 'FontSize', 16);
        xlabel('Φ_1', 'FontSize', 14);

        % add labels for each point
        for i = 1:r
            text(x(i), 0, sprintf('   S_%d', i), 'FontSize', 14);
        endfor
    end

    % Return the vector containing the energy of each symbol
    symbol_energy = energy;
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%% User Prompts %%%%%%%%%%%%%%%%%%%%%%%%%%%%

M = input("How many signals do you want to input?\n");
T = input("Input Signal period\n");
t = 0 : ts : (T-ts);    % time vector
len = length(t);
signals_vectors = {};    % cell array of user inputs

% input format is a 2xP vector, where P is number of different regions
% in the signal. row 1 represent time & row 2 represent amplitude
% To clarify, signal 1 in example 1 will be: [T/3 T; 1 0]
for i=1:M
    signals_vectors{i} = sample(input("Input Signal matrix\n"));
endfor

%% Ploting of signals
for i=1:M
    figure;
    plot(t, signals_vectors{i});
    grid on;
    title(['S_' num2str(i) ' (t)'], 'FontSize', 16);
    xlabel('Time (s)', 'FontSize', 14);
    ylabel('Amplitude (V)', 'FontSize', 14);
endfor
```

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

%%%%%%%%%%%%%%%%%%%%%%% Gram-Schmidt procedure %%%%%%%%%%%%%%%%%%%%%%%

c = basis(signals_vectors, T);
symbol_energy = constellation_diagram(c)

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

Part 2

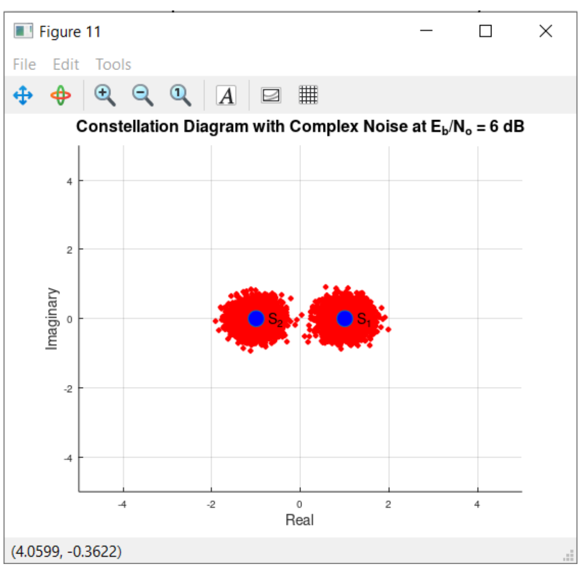
Polar NRZ

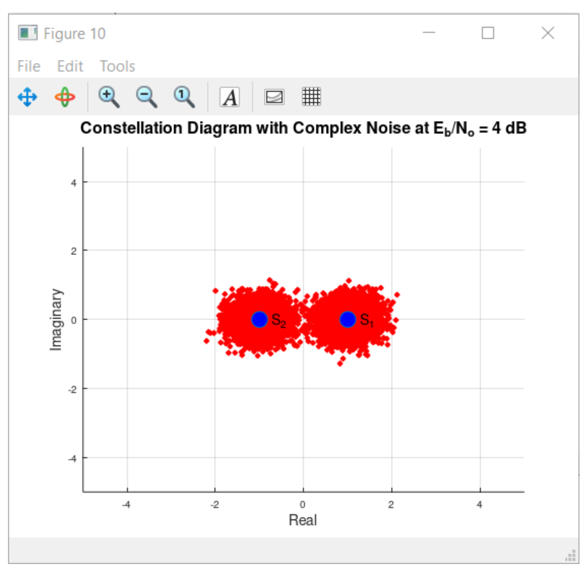
Constellation Diagram Without Noise



Note: In the following diagram, bit rate = 1 bit/s just to clarify the concept. However, in the next part bit rate = 1000 bits/s, and it could be varied from the code if necessary.

Constellation Diagram With Complex Noise





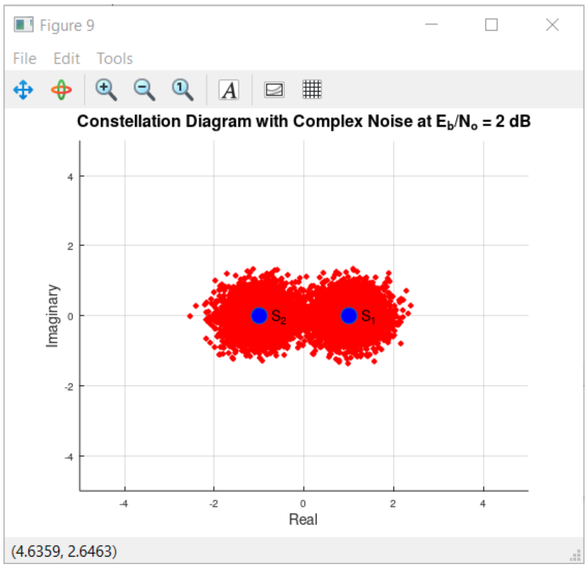


Figure 9
File Edit Tools
Constellation Diagram with Complex Noise at E_b/N_o = 2 dB
Imaginary
Real
S_2
S_1
(4.6359, 2.6463)

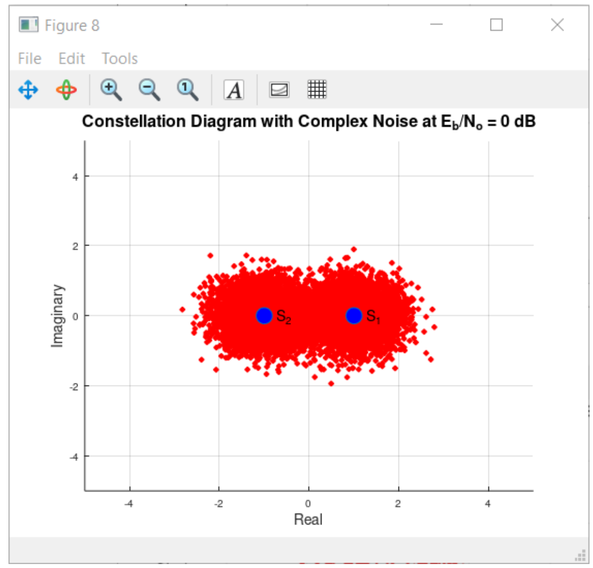


Figure 8
File Edit Tools
Constellation Diagram with Complex Noise at E_b/N_o = 0 dB
Imaginary
Real
S_2
S_1

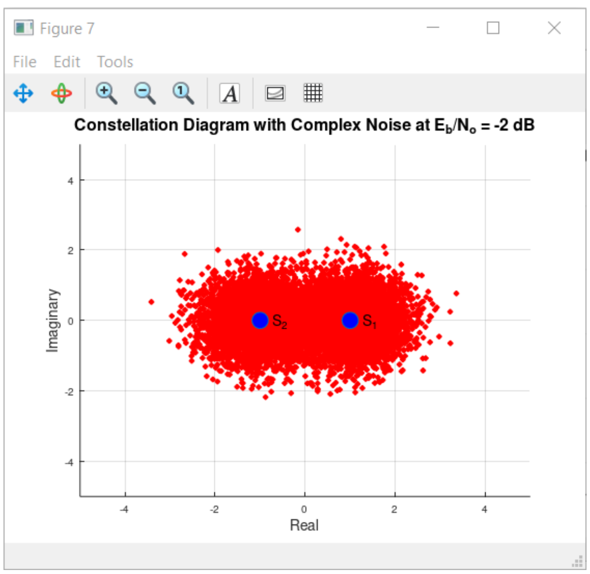


Figure 7
File Edit Tools
Constellation Diagram with Complex Noise at E_b/N_o = -2 dB
Imaginary
Real
S_2
S_1

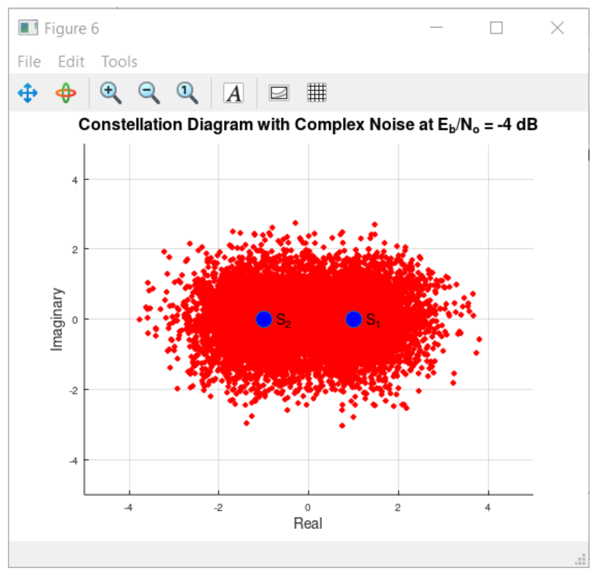


Figure 6
File Edit Tools
Constellation Diagram with Complex Noise at E_b/N_o = -4 dB
Imaginary
Real
S_2
S_1

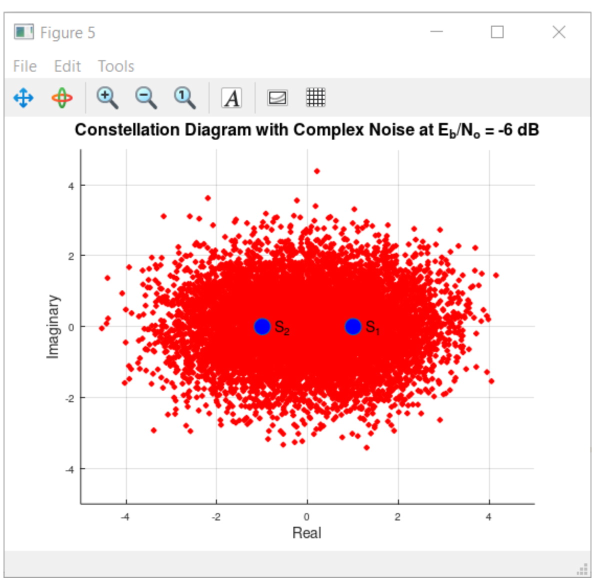


Figure 5
File Edit Tools
Constellation Diagram with Complex Noise at E_b/N_o = -6 dB
Imaginary
Real
S_2
S_1

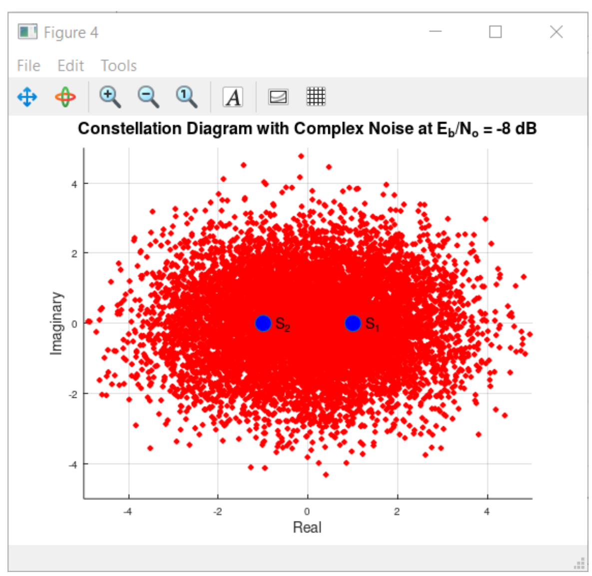
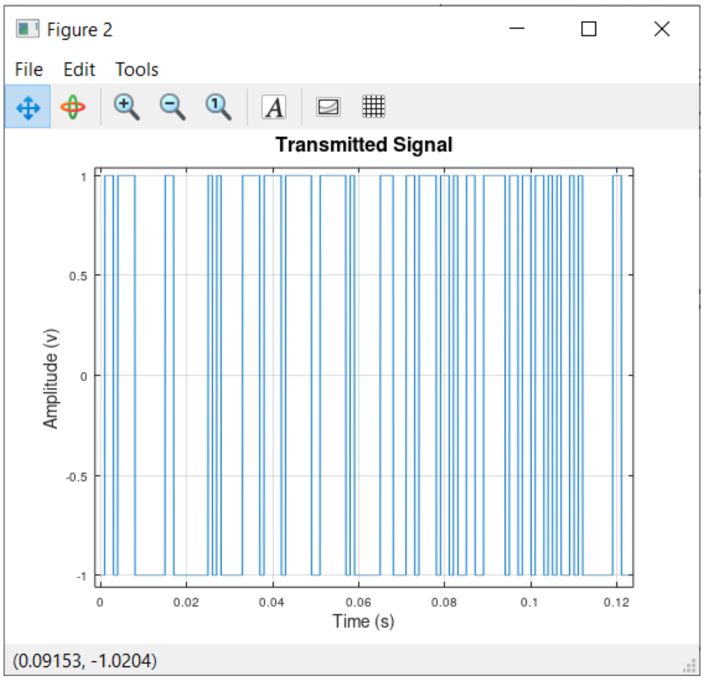


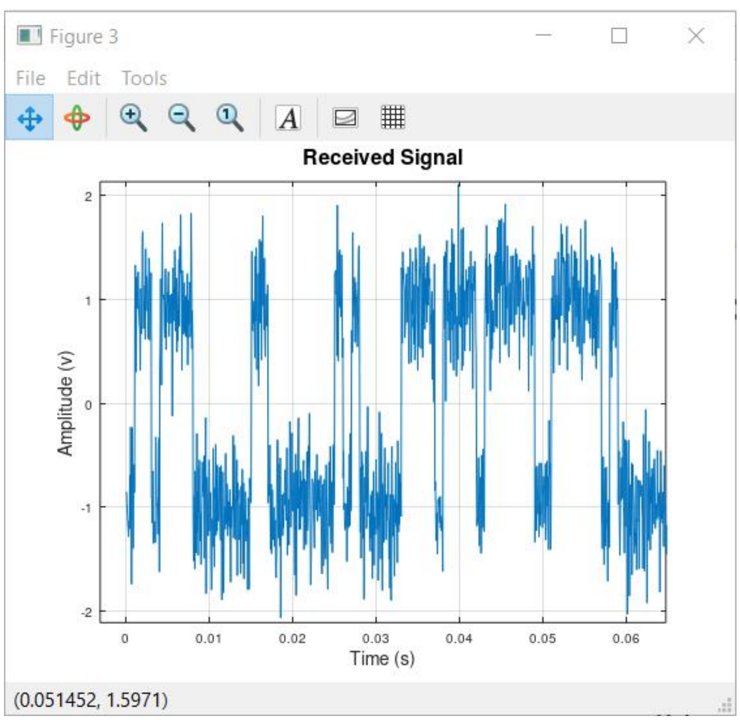
Figure 4
File Edit Tools
Constellation Diagram with Complex Noise at E_b/N_o = -8 dB
Imaginary
Real
S_2
S_1

Transmitter Encoded Signal (Time Domain)

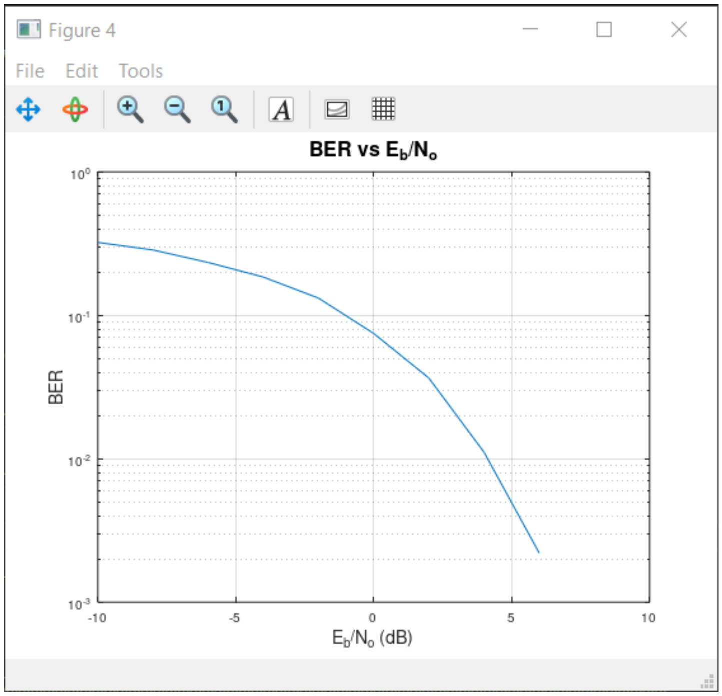


Received Noisy Signal (Time Domain)

Note: Eb/No = 6 dB for this graph.



BER vs $\frac{E_b}{N_o}$



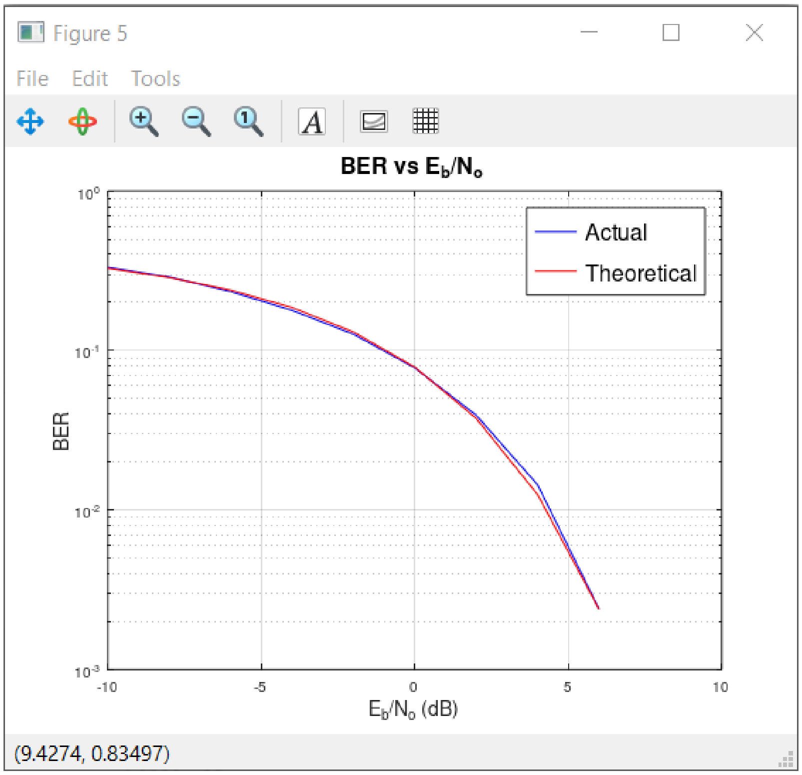
Theoretical Results

Theoretical Analysis

$$P_e(symbol) = Q\left(\frac{S_{21} - S_{11}}{\sqrt{2No}}\right)$$

- In BPSK each bit is represented in one symbol, so BER = $P_e(symbol)$
- From the constellation diagram shown at the beginning of Part 2:
- $S_{21} - S_{11}$ = $2\sqrt{Eb}$
- Therefore, BER = $Q\left(\sqrt{\frac{2 \times Eb}{No}}\right)$

BER vs $\frac{E_b}{N_o}$



BER Values

```
Command Window
>> BER_s
BER_s =

   3.3190e-01   2.8910e-01   2.3420e-01   1.7850e-01   1.2650e-01   7.7800e-02   3.9300e-02   1.4400e-02   2.4000e-03

>> BER_t
BER_t =

   3.2736e-01   2.8671e-01   2.3923e-01   1.8611e-01   1.3064e-01   7.8650e-02   3.7506e-02   1.2501e-02   2.3883e-03
```

Comments

- The BER decreases with increasing Eb/No, as predicted by the theoretical expression.
- The actual BER curve deviates slightly from the theoretical curve.
- Overall, the results demonstrate that polar NRZ can achieve good BER performance in AWGN channels.

Code

```
%%%%%%%%%%%%%%%%%%%%%%%% Code from Part 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialization %%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
pkg load communications

%% Sampling frequency
global fs = 20000;             % 20KHz
global ts = 1/fs;              % sampling

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

%%%%%%%%%%%%%%%%%%%%%%%%%% Sampling Function %%%%%%%%%%%%%%%%%%%%%%%%%

function sampled_vector = sample(input_vector)
    global ts;
    y = [];                         % vector of amplitudes at any t
    n = size(input_vector, 2);      % n represents the number of
    t_start = 0;                    % different amplitudes in the pulse
    k = 1;                          % loop iterator
    for i=1:n
        t_end = input_vector(1,i);
        for j=t_start: ts : t_end - ts
            y(k) = input_vector(2,i);
            k = k + 1;
        endfor
        t_start = t_end;
    endfor
    sampled_vector = y;
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Energy Calc. %%%%%%%%%%%%%%%%%%%%%%%%%%%%

function energy = CalcEnergy(x, T)
    n = length(x);
    delta = 1/n;
    eg = 0;
    for i = 1:n
          eg = eg + x(i)^2 * delta;
```

```
    endfor
    energy = eg * T;
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%% Signal Component %%%%%%%%%%%%%%%%%%%%%%%%%%

function coefficient = CalcCoefficient(s, phi, T)
    n = length(s);
    delta = 1/n;
    coef = 0;
    for i=1:n
        coef = coef + s(i) * phi(i) * delta;
    endfor
    coefficient = coef * T;
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%% Basis Functions %%%%%%%%%%%%%%%%%%%%%%%%%%%

function sig_comp = basis(input_signals, T)
    global fs;
    global ts;
    len = T * fs;
    t = 0 : ts : (T-ts);    % time vector
    M = length(input_signals);
    basis_vectors = {};      % cell array of basis functions
    phi = [];      % Initialization of phi amplitude vector
    c = [];          % Initialization of coefficients vector

    %% loop through signals
    for num=1:M
        flag = 0;                  % flag to check wether g = 0 or not
        N = length(basis_vectors);
        y = input_signals{num};  % amplitude vector of signal

        %% calculating basis function
        if num == 1                  % num represents the signal number
            energy = CalcEnergy(input_signals{1}, T);
            for i = 1:len
                phi(i) = y(i) / sqrt(energy);
            endfor
            basis_vectors{1} = phi;
```

```
            c(1,1) = CalcCoefficient(input_signals{1}, basis_vectors{1}, T);
        else
            sig = input_signals{num};
            g = sig;
            for i=1:N
                phi = basis_vectors{i};
                comp = CalcCoefficient(sig, phi, T);
                if abs(comp) < 1e-10
                    comp = 0;
                end
                c(num, i) = comp;
                g = g - comp * phi;
            endfor

            for i=1:len
                if abs(g(i)) < 1e-10    % elimnate any errors caused
                    g(i) = 0;         % from the subtraction
                endif                   % of floating numbers
                if g(i) != 0
                    flag = 1;
                endif
            endfor

            if flag == 1                % if g != 0
                energy = CalcEnergy(g, T);
                for i = 1:len
                    phi(i) = g(i) / sqrt(energy);
                endfor
                basis_vectors{N+1} = phi;
                c(num,num) = sqrt(energy);
            endif
        end
    endfor
    sig_comp = c;
end

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

%%%%%%%%%%%%%%%%%%%%%%% Constellation Diagram %%%%%%%%%%%%%%%%%%%%%%%%

function symbol_energy = constellation_diagram(matrix)

    % Extract coordinates
    [r,c] = size(matrix);
    x = matrix(:, 1);
```

```
% Check if 3d or 2d or 1d
if c == 3
    y = matrix(:, 2);
    z = matrix(:, 3);
    energy = x.^2 + y.^2 + z.^2;

    figure;
    plot3(x, y, z, 'o', 'MarkerSize', 8, 'MarkerFaceColor', 'b');
    grid on;
    title('Constellation Diagram', 'FontSize', 16);
    xlabel('Φ_1', 'FontSize', 14);
    ylabel('Φ_2', 'FontSize', 14);
    zlabel('Φ_3', 'FontSize', 14);

    % add labels for each point
    for i = 1:r
        text(x(i), y(i), z(i), sprintf('   S_%d', i),
             'FontSize', 14);
    endfor

elseif c == 2
    y = matrix(:, 2);
    energy = x.^2 + y.^2;

    % plot
    figure;
    plot(x, y, 'o', 'MarkerSize', 8, 'MarkerFaceColor', 'b');
    grid on;
    title('Constellation Diagram', 'FontSize', 16);
    xlabel('Φ_1', 'FontSize', 14);
    ylabel('Φ_2', 'FontSize', 14);

    % add labels for each point
    for i = 1:r
        text(x(i), y(i), sprintf('   S_%d', i), 'FontSize', 14);
    endfor
else
    energy = x.^2;
    figure;
    plot(x, zeros(size(x)), 'o', 'MarkerSize', 8,
        'MarkerFaceColor', 'b');
    grid on;
    title('Constellation Diagram', 'FontSize', 16);
    xlabel('Φ_1', 'FontSize', 14);
```

```
        % add labels for each point
        for i = 1:r
            text(x(i), 0, sprintf('   S_%d', i), 'FontSize', 14);
        endfor
    end

    % Return the vector containing the energy of each symbol
    symbol_energy = energy;
end

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


%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Polar NRZ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assume input that the Amplitude of the signal is:
% 1 --> 1 & -1 --> 0, and period T = 1s

PNRZ_sig = {sample([1;1]), sample([1;-1])};
T = PNRZ_sig{1}(1);
c = basis(PNRZ_sig, T);
symbol_energy = constellation_diagram(c);

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%% BER Function %%%%%%%%%%%%%%%%%%%%%%%%%%%%

function val = calculate_BER(n,txb,rxb)
  error_count = 0;
  for i = 1:n
    if rxb(i) != txb(i)
      error_count = error_count + 1;
    endif
  endfor
  val = error_count / n;
endfunction

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


%%%%%%%%%%%%%%%%%%%%%%%%%%% Decision Device %%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function rxb = decision_device(n, samples_per_bit, amplitude_vector)
    k = samples_per_bit;
    for i = 1:n
        v = amplitude_vector(k);
        k = k + samples_per_bit; % sample every Tb
        if real(v) < 0
          rxb(i) = 0;
        else
          rxb(i) = 1;
        end
    endfor
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Noise Sweep %%%%%%%%%%%%%%%%%%%%%%%%%%%%

function BER_s = noise_sweep(time_vector, a_vector, n,
                            samples_per_bit, txb, Eb_over_No)
    len = length(Eb_over_No);
    snr = Eb_over_No + 10 * log10(n);
    c_v = a_vector + 0.0000000000000000000000000000000000000000000001i;
    pn = zeros(2,1);
    pn(1,1) = 1;
    pn(2,1) = -1;

    for i = 1:len
        %% Signal Power = (N * amplitude ^ 2 * Tb) / Tb = N
        s = awgn(c_v, snr(i), 10 * log10(n));
        rxb = decision_device(n, samples_per_bit, s);
        BER_s(i) = calculate_BER(n, txb, rxb);

%%%%%%%%%%%%%%%%%% Complex Constellation Diagram %%%%%%%%%%%%%%%%%%%%%

        swcn = s(samples_per_bit : samples_per_bit : n*samples_per_bit);
        [r,c] = size(pn);
        x = pn(:, 1);
        energy = x.^2;
        figure;
        scatter(real(swcn), imag(swcn),7,
            'd','MarkerFaceColor','r','MarkerEdgeColor','r');
        xlim([-5 5]);
        ylim([-5 5]);
        hold on;
        plot(x, zeros(size(x)), 'o', 'MarkerSize', 8,
```

```
            'MarkerFaceColor', 'b');
        grid on;
        title(['Constellation Diagram with Complex Noise at E_b/N_o = ' ...
        num2str(Eb_over_No(i)) ' dB'], 'FontSize', 16);
        xlabel('Real', 'FontSize', 14);
        ylabel('Imaginary', 'FontSize', 14);
        % add labels for each point
        for ord = 1:r
            text(x(ord), 0, sprintf('   S_%d', ord), 'FontSize', 14);
        endfor

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    endfor


    % Plot noisy signal
    figure;
    plot(time_vector, s);
    grid on;
    title('Received Signal', 'FontSize', 16);
    xlabel('Time (s)', 'FontSize', 14);
    ylabel('Amplitude (v)', 'FontSize', 14);

    % Plot BER vs Eb/No for simple decision device
    figure;
    semilogy(Eb_over_No, BER_s);
    grid on;
    title('BER vs E_b/N_o', 'FontSize', 16);
    xlabel('E_b/N_o (dB)', 'FontSize', 14);
    ylabel('BER', 'FontSize', 14);

endfunction

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


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Tx %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Generate stream of random bits (10,000 bits)
N = 10000;                  % number of bits
bits = randi([0 1], 1, N);  % random binary vector

%% Choose a bit rate (bit/s)
Rb = 1000;
Tb = 1/Rb;
```

```
%% Sampling frequency
fs = 20000;             % 20KHz
ts = 1/fs;              % sampling period
ds = fs/Rb;             % samples per bit

%% time domain
T = N * Tb;             % total time
t = 0 : ts : (T-ts);    % time vector

y = [];     % amplitude vector

%% Encode the bits in PNRZ
k = 1;
for i = 1:N
    if bits(i) == 1
        for j = 1:ds
            y(k) = 1;
            k = k + 1;
        endfor
    elseif bits(i) == 0
        for j = 1:ds
            y(k) = -1;
            k = k + 1;
        endfor
    end
end

figure;
plot(t, y);
grid on;
title('Transmitted Signal', 'FontSize', 16);
xlabel('Time (s)', 'FontSize', 14);
ylabel('Amplitude (v)', 'FontSize', 14);

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Rx %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Eb_over_No_dB = [-10 -8 -6 -4 -2 0 2 4 6];
BER_s = noise_sweep(t, y, N, ds, bits, Eb_over_No_dB);

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

%%%%%%%%%%%%%%%%%%%%%%%%%% Theoretical BER %%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Eb_over_No = [];
BER_t = [];
for i=1:length(Eb_over_No_dB)
    Eb_over_No(i) = 10^(Eb_over_No_dB(i)/10);
    BER_t(i) = qfunc(sqrt(2*Eb_over_No(i)));
end

% Plot BER vs Eb/No for actual & theoretical
figure;
semilogy(Eb_over_No_dB, BER_s, 'b-');
hold on;
semilogy(Eb_over_No_dB, BER_t, 'r-');
grid on;
legend('Actual', 'Theoretical', 'FontSize', 16);
title('BER vs E_b/N_o ', 'FontSize', 16);
xlabel('E_b/N_o (dB)', 'FontSize', 14);
ylabel('BER', 'FontSize', 14);

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