Table of Contents


```
function [Bin_Or_optimum, Bin_Or_theory, Bin_Or_unideal] = Bin_Or(N,
 data, E, M);
```

Transmitter

```
clc;
    data_sq = zeros(1,N) ; %% Creating Binary Sequence
    for i = 1 : N
        if data(i) < 0.5
            data_sq(i) = 0;
        else
            data_sq(i) = 1;
        end
    end
```

```
Not enough input arguments.

Error in Bin_Or (line 4)
    data_sq = zeros(1,N) ; %% Creating Binary Sequence
```

M rect Pulse

```
clc;
   data_seq1 = zeros(1,N * M) ; %Pre allocating for Date Sequence
   data_seq2 = zeros(1,N * M) ;
    counter = 1; %Counter on data seq array
    for bit_counter = 1 : N
```

```
            for sym_counter = 1 : M
                if data_sq(bit_counter) == 1
                    data_seq1(1,counter) = data_sq(bit_counter); ...
                        %Repeat 0 or 1 M times
                    counter = counter + 1;
                else
                    data_seq2(1,counter) = data_sq(bit_counter) + 1; ...
                        %Repeat 0 or 1 M times
                    counter = counter + 1;
                end
            end  %We have to use two orthogonal basis like (1,i)
        end
        data_sqf = data_seq1 + j*data_seq2 ; %Creating Binary Orthogonal
```

Pwelch

```
        figure(1)
        subplot(313)
        pwelch(data_sqf)
        title("Binary Orthogonal Power Specteral")
        grid on;
%       xlabel('Frequency (Hz)')
%       ylabel('Power (dB)')
        legend('Binary Orthogonal PSD')
```

Pwelch

```
clc;
    [pxx,f] = pwelch(data_sqf,[],[],[],1000,'centered','power');
    figure(2)
    subplot(313)
    plot(f,pow2db(pxx))
    title("Binary Orthogonal  Power Specteral")
    grid on;
    xlabel('Frequency (Hz)')
    ylabel('Power (dB)')
    legend('Binary Orthogonal ')
```

Noise add by channel

```
    clc;
    n = randn(1,length(data_sqf))+
 1i*randn(1,length(data_sqf)); %noise
    r = sqrt(E / M) * data_sqf + n ; %received Signal with Noise
    r0 = sqrt(E(120,1) / M) * data_sqf + n;
```

Scatter

```
clc;
    figure(3)
```

```
subplot(313)
scatter(real(r(12,15:250)) , imag(r(12,15:250)),'k');
title("Binary Orthogonal Constellation")
grid on;
legend('Binary Orthogonal Cons')
xlabel('Real Part')
ylabel('Imag Part')
```

Scatter Plot

```
scatterplot(r0);
title("Binary Orthogonal Constellation")
grid on;
legend('Binary Orthogonal Cons')
xlabel('Real Part')
ylabel('Imag Part')
```

Demodulation

```
h = ones(1,M) / M ; % Moving Average
y = zeros(size(E,1), size(r,2) + M - 1); %preallocating
for counter = 1 : size(E,1) %E matrix 1st row
    y(counter, :) = conv(r(counter, :), h) ; %convolution on 130
arrays
end
```

Decision Making (Optimum point Selection)

```
clc;
    temp = zeros(size(E,1) , N ) ; %Preallocating
    for row = 1 : size(E, 1)
        for column = 1 : N
            temp(row, column) = y(row, column * M); %Optimum point
 Selection
        end
    end
```

BER Calculation (Desicion)

```
    br = zeros(size(E,1), N);
    for row_counter = 1 : size(E,1)
        for column_counter = 1 : N
           if real(temp(row_counter,column_counter)) <...
                   imag(temp(row_counter,column_counter))

               br(row_counter,column_counter) = 0 ;
           else
                br(row_counter,column_counter) = 1 ;
           end
        end
    end
```

BER Calculation (Prob of Error)

```
pe = zeros(size(E,1), 1) ; %Preallocating for Pr of error
clc;
for counter = 1 : size(E,1)
    for column_counter = 1 : N
        if br(counter,column_counter) ~= data(column_counter)
               pe(counter,1) = pe(counter,1) + 1 ;
        end
    end
end
```

Return

```
Bin_Or_optimum = pe' / N;
Bin_Or_theory = qfunc(sqrt(E)) ;
```

UnIdeal

```
h = ones(1,M - 1) / M ; % Moving Average with 1 sample delay
y = zeros(size(E,1), size(r,2) + M - 2); %preallocating
for counter = 1 : size(E,1) %E matrix 1st row
    y(counter, :) = conv(r(counter, :), h) ; %convolution on 130
 arrays
   end
```

Decision Making (Optimum point Selection)

```
clc;
    temp = zeros(size(E,1) , N ) ; %Preallocating
    for row = 1 : size(E, 1)
        for column = 1 : N
            temp(row, column) = y(row, column * M - 1); %Optimum point
 Selection
        end
    end
```

BER Calculation (Desicion)

```
br = zeros(size(E,1), N);
for row_counter = 1 : size(E,1)
    for column_counter = 1 : N
       if real(temp(row_counter,column_counter)) <...
               imag(temp(row_counter,column_counter))

           br(row_counter,column_counter) = 0 ;
       else
            br(row_counter,column_counter) = 1 ;
       end
    end
```

```
        end
```

BER Calculation (Prob of Error)

```
        pe = zeros(size(E,1), 1) ; %Preallocating for Pr of error
        clc;
        for counter = 1 : size(E,1)
            for column_counter = 1 : N
                if br(counter,column_counter) ~= data(column_counter)
                       pe(counter,1) = pe(counter,1) + 1 ;
                end
            end
        end
```

Return

```
        Bin_Or_unideal= pe' / N;

    end
```

Published with MATLAB® R2020b