Table of Contents

 1
Transmitter 1
Pwelch 1
Noise add by Channel 2
Scatter 2
Scatter Plot 2
Demodulation 2
Decision Making (Optimum point Selection) 2
BER Calculation (Desicion) 3
BER Calculation (Prob of Error) 3
Return 3
Unideal 3
Decision Making (Optimum point Selection) 3
BER Calculation (Desicion) 4
BER Calculation (Prob of Error) 4
Return 4

```
function [ook_optimum, ook_theory, ook_unideal] = OOK(N, data, E, M)
```

Transmitter

```
    data_seq1 = zeros(1,N * M) ; %Pre allocating for Date Sequence
    counter = 1; %Counter on data seq array
    for bit_counter = 1 : N
        for sym_counter = 1 : M
            data_seq1(1,counter) = data(bit_counter); %Repeat 0 or 1 M
 times
            counter = counter + 1;
        end
    end
    figure(1)
    subplot(312)
    pwelch(data_seq1)
    title("OOK Power Specteral")
    grid on;
%     xlabel('Frequency (Hz)')
%     ylabel('Power (dB)')
    legend('OOK PSD')
```

```
Not enough input arguments.

Error in OOK (line 3)
    data_seq1 = zeros(1,N * M) ; %Pre allocating for Date Sequence
```

Pwelch

```
clc;
    [pxx,f] = pwelch(data_seq1,[],[],[],1000,'centered','power');
    figure(2)
    subplot(312)
    plot(f,pow2db(pxx))
    title("OOK Power Specteral")
    grid on;
    xlabel('Frequency (Hz)')
    ylabel('Power (dB)')
    legend('PSD of OOK')
```

Noise add by Channel

```
clc;
    n = randn(1,length(data_seq1))+
 1i*randn(1,length(data_seq1)); %noise
    r = sqrt(2* E / M) * data_seq1 + n ; %received Signal with Noise
    r0 = sqrt(2* E(120,1) / M) * data_seq1 + n;
```

Scatter

```
clc;
    figure(3)
    subplot(312)
    scatter(real(r(130,25:2500)) , imag(r(130,25:2500)),'y');
    title("OOK Constellation")
    grid on;
    legend('OOK Cons')
    xlabel('Real Part')
    ylabel('Imag Part')
```

Scatter Plot

```
    scatterplot(r0);
    title("OOK Constellation")
    grid on;
    legend('OOK 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)

```
    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)) <
(0.5.*sqrt(2.*E...
                  (row_counter,1) /M)/2)
              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

```
   ook_optimum = pe' / N;
   ook_theory = qfunc(sqrt(E/4)) ;
```

Unideal

```
   h = ones(1,M - 1) / M  ; % Moving Average with 1 sample delay
0.1Ts
   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)

```
   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)) <
 (0.5.*sqrt(2.*E...
                   (row_counter,1) /M)/2)
               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

```
    ook_unideal = pe' / N;

end
```

Published with MATLAB® R2020b