Project Component:02

Introduction to Communication Systems

Submission date:

24th October 2020

Submission by

S U Swakath

180020036

Course Instrutor

Prof. Naveen M B

Document Contents:

[This document has bookmarks for easy navigation]

Introduction to Communication Systems.

Project Components : 2.

Name: S.U.Swakath. Roll no: 180020036.

2. (a) Given x is the transmitted signal and y is the received signal ; $y = h*x + n$, where h is gaussian random variable with zero mean and unit variance and n is gaussian randomvariable with zero mean and variance changing according to the Signal to Noise Ratio. (for calculation pourpous we will take it as σ^2).

Let us take $x \in \{S_1, S_2\}$ where $S_1 = [a, 0]$ $S_2 = [0, a]$

where $\underline{a > 0}$, $a \in \mathbb{R}^+$

S_1 and S_2 are 2 different symbols with 2 Dimension. Let S_1 represent bit 1 and S_2 represent bit 0.

Since the symbols are 2 dimension we can modell our h (the channel) as 2 dimension.

$h = [h_1, h_2]$ where h_1 and h_2 are gaussian random variables with zero mean and variance '1/2'.

Lets take the noise to be h_n, as expressed before the noise is also a RV with mean 0 and variance σ^2.

$$y = h * x + n \; ; \; x \in \{S_1, S_2\}. \quad n = [h_{n_1}, h_{n_2}]$$

h_{n_1} and h_{n_2} are mean 0 and variance $\underline{\sigma^2/2}$

Y Given S_1 was transmitted is.

$$Y_{S_1} = [ah_1 + h_{n_1}, h_{n_2}]$$

Similarly Y Given S_2 was transmitted is.

$$Y_{S_2} = [h_{n_1}, ah_2 + h_{n_2}].$$

Since we have a two dimension symbol.

$$P(Y/s_1) = P(Y/s_1 \text{ in dimension } 1) * P(Y/s_1 \text{ in dimension } 2)$$

$$= P(ah_1 + hn_1) * P(hn_2).$$

$$P(Y/s_2) = P(hn_1) * P(ah_2 + hn_2).$$

as stastically (h_1, h_2) and (hn_1, hn_2) pairs. are same that is with mean 0 and variance 1/2

$P(Y/s_2) = P(Y/s_1)$ hence

$$\log\left(\frac{P(Y/s_1)}{P(Y/s_2)}\right) = 0.$$

So we cannot demodulate using conventional ML.

We can observe this in another method also.

- Y when s_1 is passed is

  $[ah_1 + hn_1, hn_2]$

  the mean of this random variable is $[0, 0]$

  and variance is. $\left[\frac{a^2}{2} + \frac{\sigma^2}{2}, \frac{\sigma^2}{2}\right]$

  - this is because $ah_1 + hn_1$ and hn_1 are random variables by it self and h_1 has mean $= 0$, variance $= 1$ and hn_1 and hn_2 are noise with mean $= 0$ and variance $= \sigma^2$

- Similarly Y when s_2 is passed is

  $[hn_1, ah_2 + hn_2]$

  the mean of this RV is $[0, 0]$ and variance is

  $\left[\frac{\sigma^2}{2}, \frac{a^2}{2} + \frac{\sigma^2}{2}\right]$

from above we can conclude Y when s_1 or s_2 is passed the received RV is from a gaussian distribution of mean $[0, 0]$ in both case. ML rule can be implemented if and only if s_1 and s_2 received RV are from gaussian distribution with 2 different means.

Hence, we now need to some other method to demodulate the received signal.

$\Rightarrow$ if x is a RV with mean μ_x and variance σ_x^2

the expected power / average power $= E(x \cdot x^*)$

let us take x is always real

so expected power $= E(x^2)$.

$$\sigma_x^2 = E((x-\mu_x)^2) = E(x^2 + \mu_x^2 - 2\mu_x x)$$
$$= E(x^2) + \mu_x^2 - 2\mu_x^2$$

$$E(x^2) = \sigma_x^2 + \mu_x^2$$

by this we can conclude the expected power of a RV x is $\sigma_x^2 + \mu_x^2$.

Applying this observation regarding power on the received random variable (Y).

Y Given s_1 : $[ah_1 + h_{n_1}, h_{n_2}]$ $\xRightarrow{\text{expected power}}$ $\left[\frac{a^2}{2} + \frac{\sigma^2}{2}, \frac{\sigma^2}{2}\right]$.

Y Given s_2 : $[h_{n_1}, ah_2 + h_{n_2}]$ $\xRightarrow{\text{expected power}}$ $\left[\frac{\sigma^2}{2}, \frac{a^2}{2} + \frac{\sigma^2}{2}\right]$.

From this we can conclude that if s_1 was transmitted my received signal's power in dimension 1 will be most likely greater than power in dimension 2. Similarly when s_2 is transmitted then received signal's power in dimension 2 is most likely to be greater than power in dimension 1..

from this we can write the following expression.

if received signal $Y = [\alpha, \beta]$

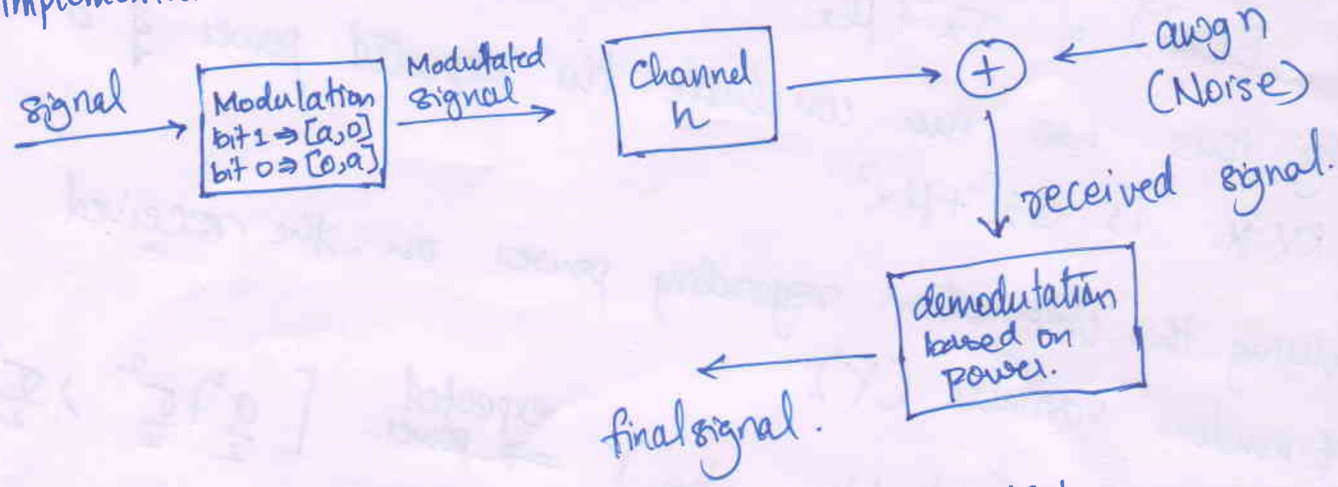
$$\alpha^2 \underset{S_2}{\overset{S_1}{\gtrless}} \beta^2 \Rightarrow \frac{\alpha^2}{\beta^2} \underset{S_2}{\overset{S_1}{\gtrless}} 1$$

from computation simplification

$$\frac{|\alpha|}{|\beta|} \underset{S_2}{\overset{S_1}{\gtrless}} 1$$

The above discussed. demodulation is implemented in the second part of the question.

(b) Implementation.



We are implementing the above block diagram in Matlab.

The following functions ~~are used~~. coded and used :-

- channel.m
- demodulation.m
- modulation.m
- Bit_Error_Rate.m.

* modutation.m
  - This function takes a signal and a scalar value 'a' as input and maps bit 1 ⇒ [a, 0] and bit 0 ⇒ [0, a] and returns the modulated signal.

* Channel.m.
  - This function takes modutated signal (x) and SNR as input and computes the receive signal (Y) using
  $Y = h * x + n$, where $h = [h\_c1 , h\_c2]$
  h_c1 and h_c2 are gaussian random variable with zero mean and variance 1. n is noise , the values are determined based on SNR. and returns Y.

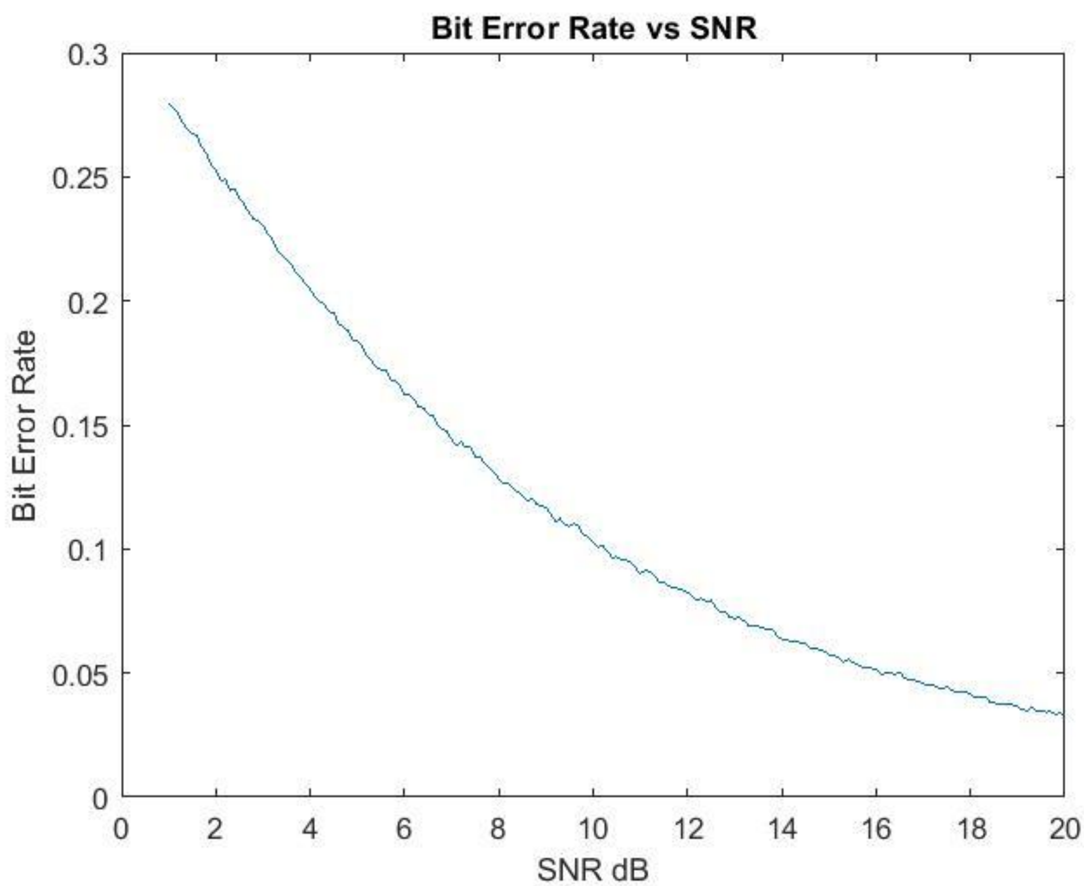
* ~~Demo~~ demodulation.m
  - This function takes the received signal (Y) and maps [α β] to bit 1 if |α| > |β| else it maps to bit 0.
  - the function returns the final signal.

* Bit_Error_Rate.m
  - the code ~~also~~ creates a signal of length N (= 1000), modulates the signal with (a=2). Then computes the Bit Error rate for each value of SNR, where SNR is varying from 1 to 20 in steps of 0.1. To ~~calcu~~ calculate the BER we take 1000 samples for each SNR value.

The ~~ob~~ plot obtained and matlab code is attached below.

Plots and Inferences

Bit Error Rate vs SNR



from the above graph we can conclude that the Bit Error Rate (BER) decreases as the Signal to Noise Ratio (SNR) increases. This trend is as expected , that is when the power of the signal is larger than that of the noise, the probability that a change of bit occurs will reduces.

```
%This function takes a signal and value "a" as inputs and maps
%bit 0 => [0 a] and bit 1 => [a 0] and returns the modulatedSignal
function [modulatedSignal] = modulation(signal,a)
    signalLen = size(signal,2);
    modulatedSignal = [];
    for i = 1:signalLen
        if(signal(i) == 0)
            modulatedSignal = [modulatedSignal ; [0 a]];
        else
            modulatedSignal = [modulatedSignal ; [a 0]];
        end
    end
end
```

```
%This fuction takes a modulatedSignal (represented x) and returns received
%signal (represented y), where y = h*x + n , h is gaussian random variable
%with zero mean and unit variance, n is gaussian noise according to the
%given signal to noise ratio.
function [receivedSignal] = channel(modulatedSignal,snr)
    receivedSignal = [];
    modulatedSignalLen = size(modulatedSignal,1);

    h_c1 = rand(modulatedSignalLen,1);
    h_c2 = rand(modulatedSignalLen,1);

    %finding (h*x)
    receivedSignal = modulatedSignal;
    receivedSignal(:,1) = h_c1.*receivedSignal(:,1);
    receivedSignal(:,2) = h_c2.*receivedSignal(:,2);

    %adding awgn
    receivedSignal = awgn(receivedSignal,snr,'measured');

end
```

```matlab
%This function takes a receivedSignal and according to the energy of the
%received symbol it maps it to bit1 or bit0 and returns the final signal

function [finalSignal] = demodulation(receivedSignal)
    finalSignal = [];
    receivedSignalLen = size(receivedSignal,1);
    absValues = abs(receivedSignal);

    for i = 1:receivedSignalLen
        temp = absValues(i,:);
        if(temp(1)>temp(2))
            finalSignal = [finalSignal 1];
        else
            finalSignal = [finalSignal 0];
        end
    end
end
```

```
%Code to find Bit Error Rate vs SNR graph
clear;clc;
%Creating a signal of length equal to N
a = 2;  % a value for modulation
N = 1000;   % length of the signal
signal = randi([0 1],1,N); %creating random signal containing bit1 and bit0

modulatedSignal = modulation(signal,a); %finding the modulated signal
SNR = 1:0.1:20; %different values of SNR for the plot
errorSNR = []; %array containing BER corresponding to each SNR
sampleSize = 1000; %Number of sample taken to calculate the error

%Finding the BER for each SNR value
for i = 1:0.1:20
    fprintf("SNR %0.2f ....\n",i);
    error = 0;
    for j = 1:sampleSize
        receivedSignal = channel(modulatedSignal,i);
        finalSignal = demodulation(receivedSignal);
        allError = finalSignal~=signal;
        error = error + sum(allError)/size(allError,2);
    end
    error = error/sampleSize;
    errorSNR = [errorSNR error];
end

%Plotting the BER  vs SNR graph
plot(SNR,errorSNR);
xlabel("SNR dB");
ylabel("Bit Error Rate")
title("Bit Error Rate vs SNR");
```