Project Component:01

Introduction to Communication Systems

Submission date:
20th October 2020

Submission by
S U Swakath
180020036

Course Instrutor
Prof. Naveen M B

Document Contents:

[This document has bookmarks for easy navigation]

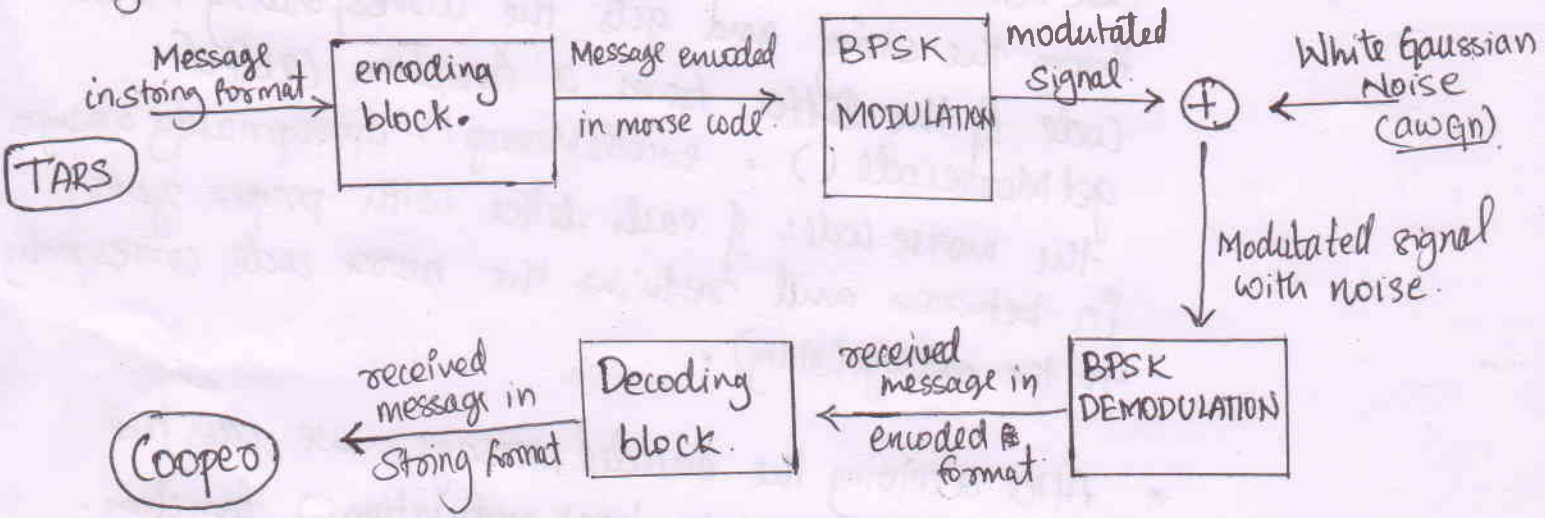
Introduction to Communication Systems.

Project Component : 1

Name: S.V. Swakath.

Roll Number : 180020036.

As per the requirement, I have build a digital communication system for TARS using BPSK modulation and Demodulation and Morse code. The system can be elaborately represented & by the following block diagram.



⇒ Assumptions taken :-

* We have working through out only in the baseband.
* Considering the message signal to be long, both -1 and 1 in bpsk has equal probability of occurance.
* We have 26 english alphabets and 10 digits in total. 36 "characters/letters". While computing the letter error rate, we assume all letters are equally probable to occur in the message. ($1/36$)
* To calculate word error rate. I have take "50" words from english dictionary using "random-word PyPI" in python and later copied the 50 words into matlab in form of a cell array. Even here we consider the occurance of each word is equally probable.

Note:- Since, I have take 50 words and 36 letters to calculate my letter error rate and word error rate respectively. Computation time is very long.

Note :- Reduce the words array size in case of testing the code to run the code in less time.

Explaining implementation of each block :-

- Encoding block :-
  * We initially take a message in string format and pass the string to function called encodeMessage().
  * encodeMessage() function converts the message string to all caps letters (as morse code is not case-sensitive), then it reads letter by letter from the string and gets the corresponding Morse code of the letter from a function called getMorseCode(). encodeMessage() appropriately arranges the morse codes of each letter with proper zeros in between and returns the morse code corresponding to the message(string).
  * After receiving the encoded message, we pass the encoded message into bpsk-modulation() function. bpsk-modulation() takes the encoded message with an SNR value. It first converts bits 0 ⇒ 1 and bit 1 ⇒ -1 and adds an awgn noise to the modulated signal with passed SNR value. The function finally returns a bpsk modulated with added white gaussian noise.
  * The modulated signal is then passed to a bpsk_demodulation() function. The function does a ML demodulation with boundary point as zero. That is if we receive a positive number it is mapped to bit 0 and if we receive a negative number we map to bit 1. The function after computation returns a demodulated signal/ received encoded message.

* Later the received encoded message is passed to decodeMessage() function. Where the function find three consecutive zero and seperates the morse code. and find the corresponding letter to the code using getMorseChar() function. In case if the code does not match with any morse code getMorseChar() function will return a "*" (indicating error), by this process the encoded received message is decoded and returned as a string.

List of Functions

* getMorseCode(val) ⇒ take a value and returns the morse code corresponding to the value.

* getMorseChars(code) ⇒ Take a code and returns the corresponding character to the code or returns '*' if no character is found.

* encodeMessage(message) ⇒ Takes a string and returns the corresponding morse code of the string.

* decodeMessage(encodedmessage) ⇒ Takes a morse encoded message and returns the corresponding string.

* bpsk-modulation(signal, SNR) ⇒ performs bpsk modulation on received signal and adds gaussian noise according to the SNR received returns the final modulated + noise signal.

* bpsk-demodulation(signal) ⇒ Take a signal, performs bpsk demodulation and returns the demodulated signal. (threshold / boundary is

Computation of letter error rate vs SNR graph.

Notations

$P_{e,snr}$ → Letter error for specific snr value.

P_{ch} → poopability of character 'ch' occuring in the message.

$P_{e,ch,snr}$ → poopability of error of ch given specific snr.

$P_{e,snr} = \sum_{\text{all ch}} P_{ch} \cdot P_{e,ch,snr}$; from assumptions we know all letters are equally poobable to occurs hence

$P_{ch} = 1/36$.

$$P_{e,snr} = 1/36 \sum_{\text{all,ch}} P_{e,ch,snr}$$

- We do the same computation of snr from 1 to 25 dBW in steps of 0.1. To calculate the ~~probability~~ error for each character of one specific snr, I have taken 1000 samples.
- ~~This~~ latter a graph is plotted between SNR vs Letter error rate.

Computation of Word error rate :-

- To compute the word error rate, we use the similar method that we used to compute letter error rate.
- Instead of letters in here we transmit 50 words for each SNR and compute the word error rate

$P_{e,word,SNR}$ → error of a given word for a given SNR

$$P_{e,words} = \sum_{word=allwords} P_{word} \cdot P_{e,words,SNR}$$

$$= \frac{1}{50} \sum_{word=allwords} P_{e,word,SNR}.$$

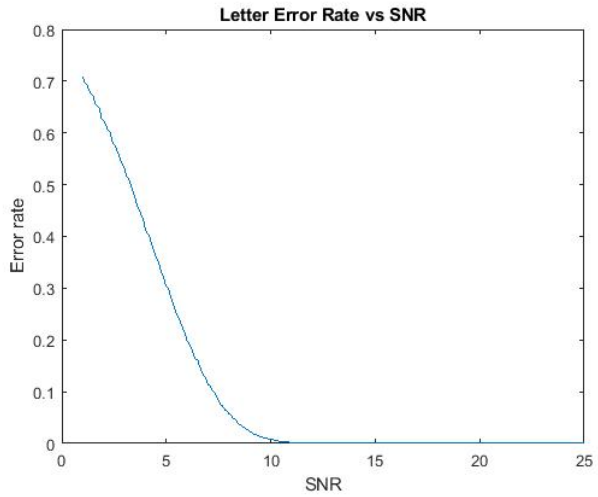
- The graphs are attached at the end of this document. both the graphs depect the trend as expected in theory. The error rates decreases and and saturates close to zero, when SNR is increased.

Scope of Improvement

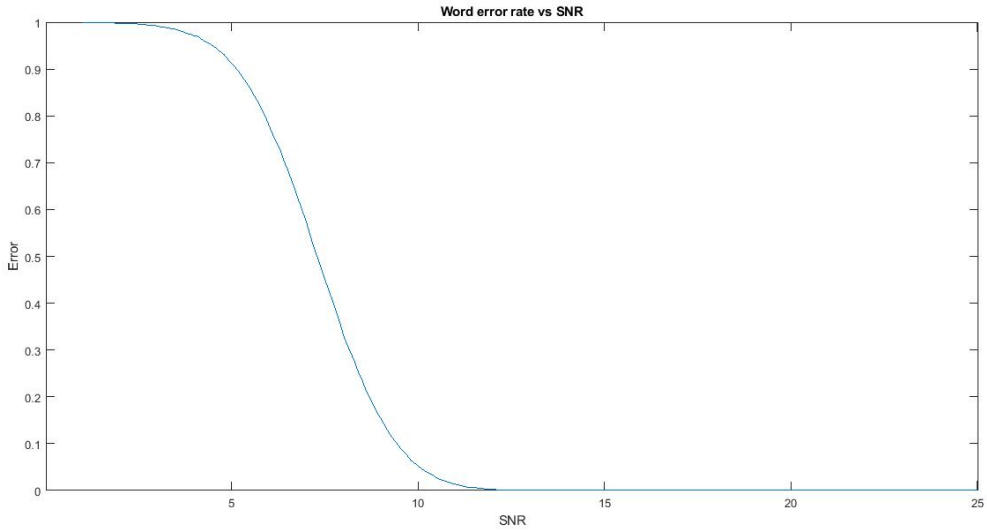
- In this communication, all the process is done in base band. To model the real life scenario we can include pass band modules.
- In this model bits are not repeated to reduce the error rate. To improve efficiency we can transmit one bit multiple times. That is if tx = [10111] we actualy transmit [111 000 111 111 111] in this case 3 time.

Plots and Inferences

Letter Error Rate vs SN



Word Error Rate vs SN



for the above graphs we can conclude that the error rate decreases as the signal to noise ratio increases. This trend is as expected , that is when the power of the signal is larger than that of noise the probability that a change of bit occurs will reduce.

```
%This function takes a character as input and finds the corresponding morse
%code from a dictionary and returns the morse code.
% Coded by S U Swakath. Last Updated: 19th Oct 2020

function [code] = getMorseCode(val)

%Morse code dictionary
    charMorseCode = {[1 0 1 1 1];[1 1 1 0 1 0 1 0 1];[1 1 1 0 1 0 1 1 1 0 1];[1 1 1 0 1 0 1];[1];[1 0 1 0 1 1 1 0 1];[1 1 1 0 1 1 1 0 1];[1 0 1 0 1 0 1];[1 0 1];[1 0 1 1 1 0 1 0 1];[1 1 1 0 1 1 1 0 1];[1 0 1 1 1 0 1 0 1];[1 1 1 0 1 1 1];[1 1 1 0 1];
        [1 1 1 0 1 1 1 0 1 1 1];[1 0 1 1 1 0 1 1 1 0 1];[1 1 1 0 1 1 1 0 1 0 1 1];[1 0 1 1 1 0 1];[1 0 1 0 1];[1 1 1];[1 0 1 0 1 1 1];[1 0 1 0 1 0 1 1 1];[1 0 1 1 1 0 1 1 1];[1 1 1 0 1 0 1 0 1 1 1];[1 1 1 0 1 0 1 1 1 0 1 1 1];[1 1 1 0 1 1 1 0 1 0 1];
        [1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1];[1 0 1 0 1 1 1 0 1 1 1 0 1 1 1];[1 0 1 0 1 1 1 0 1 1 1];[1 0 1 0 1 0 1 1 1];[1 0 1 0 1 0 1 0 1];[1 1 1 0 1 0 1 0 1 0 1];[1 1 1 0 1 1 1 0 1 0 1 0 1];[1 1 1 0 1 1 1 0 1 1 1 0 1 0 1];
        [1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1];[1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1];[0 0 0 0 0 0 0]};

    chars = ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","1","2","3","4","5","6","7","8","9","0"," "];
    charsLen = size(chars,2);

%Finding the code from the dictionary
    for i = 1:charsLen
        if isequal(val,chars(i))
             code = charMorseCode{i};
             break;
        end
    end

end
```

```
 1 %This function takes a morse code as input and finds the corresponding
 2 %character associated to the code from a dictionary and returns
 3 %the character. If a character is no found in the dictionary it returns '*'
 4 % Coded by S U Swakath. Last Updated: 18th Oct 2020
 5
 6 function [char] = getMorseChar(code)
 7     %Morse code dictionary
 8     charMorseCode = {[1 0 1 1 1];[1 1 1 0 1 0 1 0 1];[1 1 1 0 1 0 1 1 1 0 1];[1 1 1
0 1 0 1];[1];[1 0 1 0 1 1 1 0 1];[1 1 1 0 1 1 1 0 1];[1 0 1 0 1 0 1];[1 0 1];[1 0 1 1
1 0 1 1 1 0 1 1 1];[1 1 1 0 1 0 1 1 1];[1 0 1 1 1 0 1 0 1];[1 1 1 0 1 1 1];[1 1 1 0
1];
 9         [1 1 1 0 1 1 1 0 1 1 1];[1 0 1 1 1 0 1 1 1 0 1];[1 1 1 0 1 1 1 0 1 0 1 1
1];[1 0 1 1 1 0 1];[1 0 1 0 1];[1 1 1];[1 0 1 0 1 1 1];[1 0 1 0 1 0 1 1 1];[1 0 1 1 1
0 1 1 1];[1 1 1 0 1 0 1 0 1 1 1];[1 1 1 0 1 0 1 1 1 0 1 1 1];[1 1 1 0 1 1 1 0 1 0 1];
10         [1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1];[1 0 1 0 1 1 1 0 1 1 1 0 1 1 1];[1 0 1
0 1 0 1 1 1 0 1 1 1];[1 0 1 0 1 0 1 0 1 1 1];[1 0 1 0 1 0 1 0 1];[1 1 1 0 1 0 1 0 1 0
1];[1 1 1 0 1 1 1 0 1 0 1 0 1];[1 1 1 0 1 1 1 0 1 1 1 0 1 0 1];
11         [1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1];[1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
1];[0 0 0 0 0 0 0]};
12
13     chars = ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","
R","S","T","U","V","W","X","Y","Z","1","2","3","4","5","6","7","8","9","0"," "];
14     charsLen = size(chars,2);
15
16     noCharFound = true; %to identify if a character is found for the code
17
18     %Finding the charater from the dictionary
19     for i = 1:charsLen
20          if isequal(charMorseCode{i},code)
21              char = chars(i);
22              noCharFound = false; %as the character is found.
23          end
24     end
25
26     %If character not found then returning '*'
27     if noCharFound
28          char  = "*";
29     end
30 end
```

```
%This function take a message in form of a string and returns the
%corresponding Morse code.
% Coded by S U Swakath. Last Update: 18th Oct 2020

function [encodedMessage] = encodeMessage(message)
    encodedMessage = []; %final encodedMessage
    charCode = []; %stores the morse code of each character
    letterGap = [0 0 0]; %Morse code for gap between each letter
    capsMessage = upper(message); %converting message to uppercase.
    messageLen = strlength(capsMessage);
    prevSpace = false; %to identify was the previous character a space.

    for i = 1:messageLen
        %Extracting each character and finding it's morse code
        char = extractBetween(capsMessage,i,i);
        charCode = getMorseCode(char);

        %Adding the character code into final encodedMessage
        if((char ~= " ")&&(i~=1)&&(~prevSpace))
            encodedMessage = [encodedMessage letterGap];
        end
        encodedMessage = [encodedMessage charCode];

        if (char == " ")
            prevSpace = true;
        else
            prevSpace = false;
        end
        charCode = [];
    end
end
```

```
%This function take a morse code encoded message and returns the decoded
%message in the form of a string.
% Coded by S U Swakath. Last Updated: 19th Oct 2020

function [decodedMessage] = decodeMessage(encodedMessage)
encodedMessageLen = size(encodedMessage,2);
decodedMessage = "";
j = 1;
stack = [];
prevSpace = false;

while(j<=encodedMessageLen)
    stack = [stack encodedMessage(j)];
    stackLen = size(stack,2);
    if stackLen > 3
        if isequal(stack(stackLen-2:stackLen),[0 0 0])
            char = getMorseChar(stack(1:stackLen-3));
            decodedMessage = append(decodedMessage,char);
            stack = [];
            prevSpace = false;
        end
    elseif stackLen == 3
            if isequal(stack,[0 0 0])
                char = " ";
                decodedMessage = append(decodedMessage,char);
                stack = [];
                j = j+1;
                if(prevSpace)
                    j = j+3;
                end
                prevSpace = true;
            end
    end

    if j == encodedMessageLen
        if (~isequal(stack,[]))
            char = getMorseChar(stack);
            decodedMessage = append(decodedMessage,char);
            stack = [];
        end
    end
    j = j+1;
end
end
```

```
1 %This function take a binary signal and Signal to Noise Ratio(snr) as input
2 %It maps bit 1=> -1 and bit 0=>1 and adds awgn to the modulates signal and
3 %returns the final modulated signal+noise.
4 % Coded by S U Swakath. Last Updated: 18th Oct 2020
5
6 function [modulatedSignal] = bpsk_modulation(signal,snr)
7     modulatedSiganl = -2*signal + 1; %mapping bits 1 => -1 and bits 0 => 1
8     modulatedSignal = awgn(modulatedSiganl,snr,'measured'); % adding noise
9 end
```

```
% This function is used to demodulate a bpsk signal. The boundare condition
% is taken to be 0. The function takes a signal and maps all the positive
% values to bit 0 and all the negative values to bit 1.
% The final demodulated signal is returned.
% Coded by S U Swakath. Last Updated: 18th Oct 2020

function [demodulatedSignal] = bpsk_demodulation(signal)
    signalLen = size(signal,2);
    for i = 1:signalLen
        if(signal(i)>0)
            signal(i) = 0;
        else
            signal(i) = 1;
        end
    end
    demodulatedSignal = signal;
end
```

```m
%This program finds the letter error rate for each value of snr and plots a
%graphs between letter error rate and snr.
% Coded by S U Swakath. Last Updated: 19th Oct 2020

%chars array (below) contain all english letters and numbers from 0 to 9
chars = ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","1","2","3","4","5","6","7","8","9","0"];

charsLen = size(chars,2); % stores the length of the chars array

snr = 1:0.1:25; %different values of snr for plotting the graphs
snrLen = size(snr,2); %length of snr array
finalerr = zeros(1,snrLen);%final letter error rate for each snr
charerr =[]; %to store error rate of each character for different snr

%Finding the letter error
for k = 1:charsLen
    char = chars(k);
    fprintf("Working on character %s ...\n",char);
    encodedMessage = encodeMessage(char);
    for i= 1:0.1:25
        leterr = 0;

        for j = 1:1000  %1000 samples are take to determine the error
            modulatedSignal = bpsk_modulation(encodedMessage,i);
            receievedCodedMessage = bpsk_demodulation(modulatedSignal);
            receievedMessage = decodeMessage(receievedCodedMessage);
            if(receievedMessage~=char)
                leterr = leterr + 1;
            end
        end
        leterr = leterr/1000;
        charerr = [charerr leterr];
    end
    finalerr = finalerr + charerr;
    charerr = [];
end
finalerr = finalerr/charsLen;
disp("Computation done now plotting");

%Plotting the graph between computed letter error rates vs snr
plot(snr,finalerr);
title("Letter Error Rate vs SNR");
xlabel("SNR");
ylabel("Error rate")
```

```
 1 %This program finds the word error rate for each value of snr and plots a
 2 %graphs between word error rate and snr.
 3 % Coded by S U Swakath. Last Updated: 19th Oct 2020
 4
 5 %words array (below) containa 50 words
 6 %Reduce the words array length for faster computation
 7 %50 words will take 20+min for computation.
 8 words = {'symmachy', 'sargus', 'tanzim', 'flymaker', 'testones', 'wrey',↙
'subsenses', 'apomorphy', 'fulvalene', 'beardmoss', 'commutant', 'ipus', 'astony',↙
'horkey', 'amarelle', 'erectility', 'chideress', 'dynode', 'stapedius', 'Mawworms',↙
'renitence', 'pallasites', 'trindle', 'crevisse', 'pentagraph', 'peekapoo', 'creyme',↙
'whangee', 'chondr', 'poison', 'borism', 'teascrub', 'fighting', 'sixteenmo',↙
'lightrays', 'finific', 'plugrod', 'atoner', 'dactylics', 'HGHarrison', 'zonohedron',↙
'firecrests', 'chiromys', 'wolving', 'smicker', 'minth', 'gymgoer', 'eschel',↙
'horseload', 'oides'};
 9 wordsLen = size(words,2);
10
11 snr = 1:0.1:25; %values of snr for the plot
12 snrLen = size(snr,2); %lenght of snr vector
13 finalerr = zeros(1,snrLen); %final word error rate for each snr
14 worderr = []; % to save error rate for each word from the words array
15 curerr = 0; %to store the number of non matching signals
16
17 %Finding the word error rate for each snr
18 for k =1:wordsLen
19     curWord = words{k};
20     upCurWord = upper(curWord);
21     fprintf("Working on %d %s..\n",k,curWord);
22     encodedMessage = encodeMessage(curWord);
23     for i= 1:0.1:25
24         curerr = 0;
25         for j = 1:1000 %1000 samples are taken to determine the error
26             modulatedSignal = bpsk_modulation(encodedMessage,i);
27             receievedCodedMessage = bpsk_demodulation(modulatedSignal);
28             receievedMessage = decodeMessage(receievedCodedMessage);
29             if(receievedMessage~=upCurWord)
30                 curerr = curerr + 1;
31             end
32         end
33         curerr = curerr/1000;
34         worderr = [worderr curerr];
35     end
36     finalerr = finalerr + worderr;
37     worderr = [];
38 end
39
```

```
40 finalerr = finalerr/wordsLen;
41
42 %plotting the graph between snr and computed word error rates
43 plot(snr,finalerr)
44 title("Word error rate vs SNR")
45 xlabel("SNR");
46 ylabel("Error")
```

```
%Program used to debug and experiment

message = "I Love India "; %input message/ transmitted message

codedMessage = encodeMessage(message);%encoding the message to morse code
modulatedSignal = bpsk_modulation(codedMessage,20); %bpsk modulation
demodulatedSignal = bpsk_demodulation(modulatedSignal); %bpsk demodulation
receivedMessage = decodeMessage(demodulatedSignal);%decoding received code

%printing the received message
fprintf("The receieved signal:%s\n",receivedMessage);
```