<u>ADVANCED DIGITAL COMMUNICATIONS LAB</u>

<u>G622</u>

<u>TERM PROJECT</u>

SEMESTER -II

2020-2021

Aakula Gandharva Krishna 2020H1240082H

Guided By

Dr Prashant K Wali **Mr Sarath Vinnakota**

Digital Communication System

Aim:

To construct a digital transmitter, receiver system which should be able to transmit and receive a series of alphabets , it should be transmitted through a AWGN channel and should follow the conventional flow of a digital communication transmitter and receiver system, perform the following:

i) Construct a source encoder to convert the alphabets to a transmissible form and for compression, calculate the compression ratio
ii) Construct a channel encoder to concatenate redundant bits to remove channel errors
iii) Construct a line encoder to take the bitstream from channel encoder and convert it to signals
iv) Modulate the bits using either BPSK or QPSK or QAM.
v) Demodulate the bits using demodulator and the detector and retrieve the bits and send it to the channel decoder and then to the source decoder and finally find the symbol error probability.

Instructions:

- **Open the zip file to find two matlab files "ADC_Lab_Term_Project.m" and "sample3text.mat".**
- **Add these two files to the Matlab path and Run the code, the code will automatically take the characters in the text file and give the symbol error rate plot.**
- **It would take an approximate 10 to 15 minutes for execution as the plot generated is for an average of 5 simulations.**

Theory:

- **Source Encoder:**

This block plays a role of encoding the received alphabets into codewords of '0s' and '1s' , We can use various encoding schemes in order to encode the alphabets.

I have taken the case of fixed length encoding wherein a 6 bit sequence of binary digits is assigned to each character of 26 alphabets.

A ----------------- 101101

B------------------- 010110

C ------------------ 000110

a ------------------ 010100

z ------------------- 101111

Algorithm:

>Dict of alphabets

>Generate random integers from 1 to 52 length of alphabets (capital and small)

> Use decimal to binary converter and get the sequence

> Compare your sample text with every entry in the dictionary and concatenate the sequence whenever detected.

After the aforesaid is done we get a sequence of 0’s and 1’s of length (no of alpabets x bits assigned for each alphabet). Now this sequence is to be compressed using a lossless compression technique which can achieve a compression not greater than 2:1 , I have taken Run Length Coding to store the repetitions of 0’s and 1’s and then used Huffman encoding to encode the repetitive decimal values using prefix codes, so the take away is that the end product length is always less than the input length and it varies.

00000000000001011100101111001010001010101111000101001111100…..

Sequence after Fixed length coding

13 ’0’ 1 ‘1’ 1 ‘0’ 3 ‘1’ …………………….

Run length coding

Huffman coding

001001100011……….

The above sequence consists of prefix free codes and eventually have a lesser length than the input sequence and hence we take the sequence to be compressed, the values in ‘X’ are stored and are known to the receiver while decoding

$$\text{Compression ratio} = \frac{\text{Uncompressed size after Fixed length Coding}}{\text{Compressed size after Run length and Huffman Coding}}$$

Algorithm:

>Take the Fixed encoded bits and calculate the repetitions of the binary bits and store separately the frequencies and the bit values separately

>Now the frequencies would be decimal numbers

>Taking these decimal numbers, construct a code book of prefix codes and convert the sequence of decimal numbers

>Concatenate the sequence

- **Channel Encoder**

A Channel encoder is used to remove errors caused due to the channel utilized it can be done in various ways like repetition codes, Hamming codes etc.,

A (10,5) Hamming Code with a G generator matrix has been considered.

$$G = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{matrix}$$

Parity matrix

$$H = \begin{matrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{matrix}$$

1	1	0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1

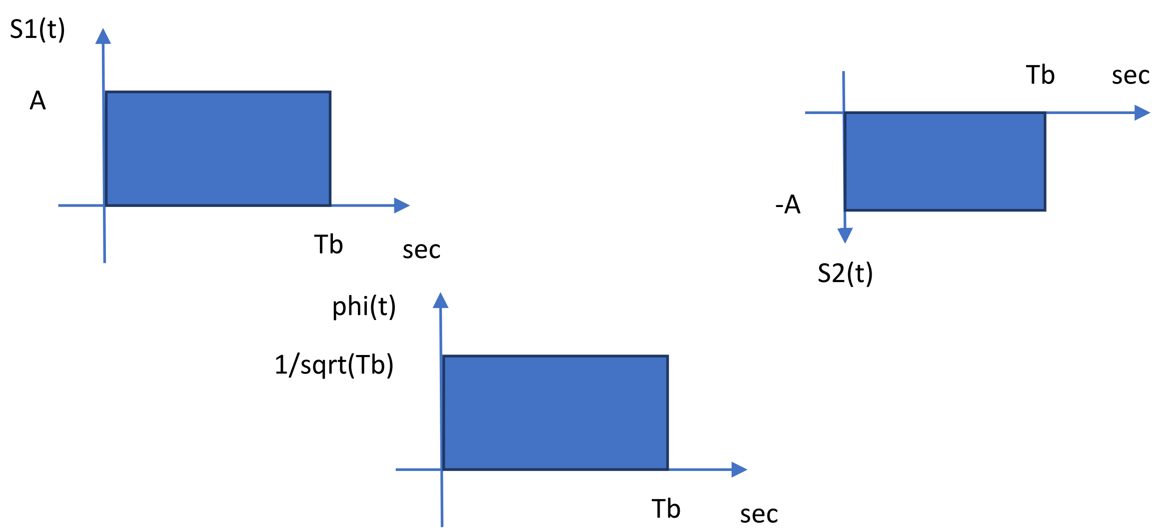
Algorithm:

>So the bits received from the Huffman encoder are taken k at a time and stacked to give a D block any remaining bits are concatenated to the D Block and then we compute the C matrix using C=D*G;

>Now the generated sequence is sent to the line encoder

- **Line Encoder**

Non Return to Zero polar signalling scheme consists of two voltage levels +V and -V which are transmitted for 1's and 0's respectively, In this case of binary transmission we consider them to have an amplitude of +A and -A respectively.



The figures depicted above are the signals representing binary transmission of NRZ bipolar with two levels +A for '1' and -A for '0', the third diagram shows the orthonormal basis function that can be obtained from the Gram Schmidt Orthogonalization procedure.

➢ Demodulator

Matched Filter demodulator uses the Orthonormal basis function as the impulse response to invert and retrieve the transmitted bit sequence. We use the equation

$$y(t) = r(t) * h(t)$$

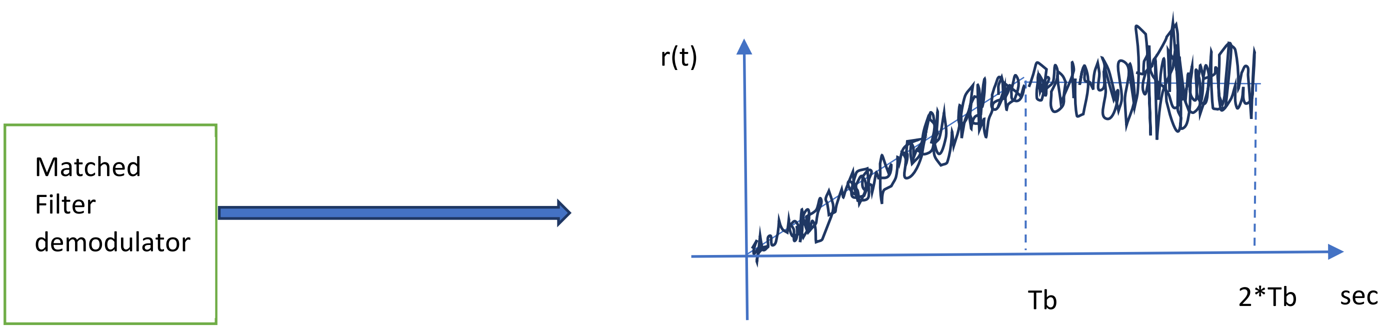
$$y(t) = s(t) * h(t) + w(t) * h(t)$$

$$y(t) = \int_0^{Tb} s(\tau)h(t-\tau)\, d\tau \; + \int_0^{Tb} w(\tau)h(t-\tau)\, d\tau$$

$$[y(t)](t = Tb) = \pm\sqrt{Eb} + n$$

$$h(t) = \emptyset(Tb - t)$$

Therefore the orthonormal basis function is used as the impulse response of the filter, Note that we sample the received convoluted sequence at t=Tb to get the scalar value which is given to the Decoder



➢ Detector

Decision making rule:

$$\alpha^* = \; No/(4 * \sqrt{Eb}) * \ln(\frac{1-p}{p})$$

$r > \alpha^*$ - Decode S1(t) and estimated bit is ‘1’

$$r < \alpha^* \text{ - Decode S2(t) and estimated bit is '0'}$$

This decision making rule is based on Maximum Aposteriori Probability if $\alpha^* = 0$ for p=0.5 then it is called Maximum Likelihood Detection

- **Channel Decoder**

These bits are again converted into a received matrix R of dimension equal to Codeword matrix, then calculate the syndrome vector using

S=R*H.';

And then using the S vector value and finding the vector in the parity check transpose matrix and finally correcting the bit at the error position indicated by the vector

Example:

R=0010110101 H=[1011001000

0100101011

0011100110

0101011001

1011100110]

S=[00101] if we assume we obtain this syndrome then take the transpose of H matrix and find the above vector and correct the bit at the position indicated by the vector

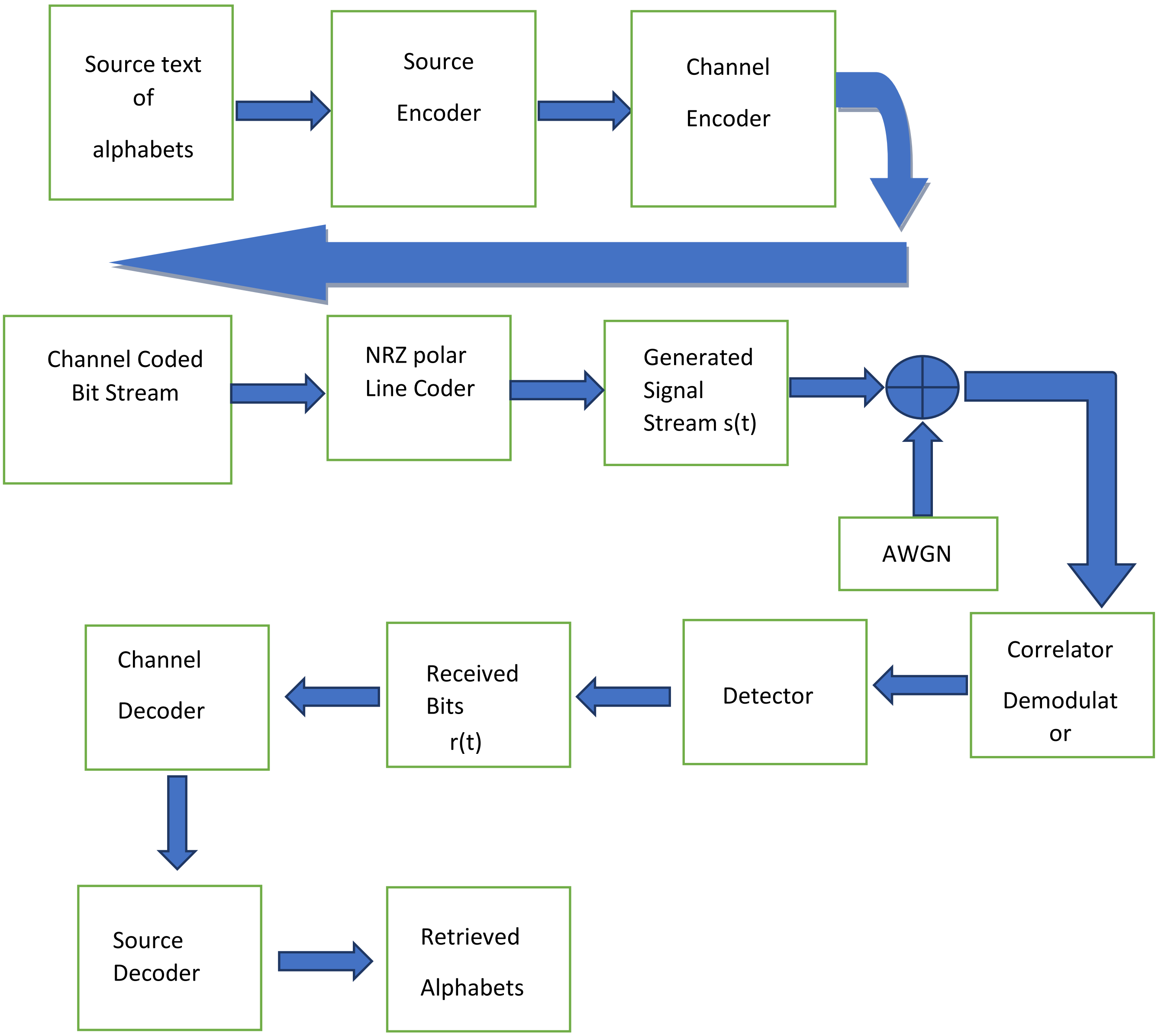
- **Source Decoder**

Now the bits at the output of the channel decoder is sent to the Huffman decoder to retrieve the frequencies of the decimal numbers obtained at the time of run length encoding

Now based on the sequence stored at the transmitter while performing run length encoding we obtain the decoded sequence which is equal to the uncompressed sequence obtained after fixed length coding

- **Retrieve the estimated alphabets**

By performing fixed length decoding retrieve the alphabets and compare with the input sample text to finally obtain the symbol error rate



Block Diagram of the Digital communication transmitter receiver system

Algorithm:

1.) Construct a dictionary of alphabets containing all the 52(small and capital alphabets) characters and calculate the frequency of occurrences of each alphabet in the text file.
2.) Perform fixed length encoding taking the number of bits to be 6 bits as we can generate 64 combinations for 52 characters .
3.) Calculate the frequencies of repetitions of each bit and store the count as well as the corresponding bits this comes under the run length encoding scheme.
4.) Now perform Huffman encoding by generating the codebook and then encoding based on the codebook.
5.) Compute the compression ratio and compressed space percentage
6.) Send the bits through the Channel encoder to concatenate the check bits using the parity matrix.
7.) Transmit the bits through the line encoder.
8.) Take A=10 and Tb=1 and t=(0,Tb)with and interval of 0.1 and T=(0,N*Tb), so that we get a Tb pulse for each bit and we have N bits
9.) Using two for loops construct the signals S1(t) and S2(t).
10.) Then based on the bit sequence concatenate S1(t) for '1' and s2(t) for '0'.
11.) Plot the variable 's' to get plot of signals corresponding to bitstream.
12.) Take a variable aphi and divide it by sqrt(Eb) to get the Orthonormal Bias.
13.) Now add the WGN noise using the awgn command.
14.) Take snr values from snr=[-15,10] with an interval of 1dB.
15.) Calculate the variance No/2 value for each value of snr.
16.) **Matched Filter demodulator:** In this case we take the inverting filter h(t) to be reversed and time shifted version of the orthonormal basis function the received symbol is convolved with the above filter and the received output is sampled at t=Tb and the scalar value is given to the decoder for decision making.
17.) We calculate the threshold taking alpha=(No/4*sqrt(Eb))*ln(1-p/p)
18.) Then we take an if elseif condition that if r>alpha then estimated bit is '1' and if r<alpha estimated bit is '0'.
19.) Now the estimated bits are then converted to a received matrix and calculate the syndrome vector and correct the bits.

20.) These corrected bits are now sent through the Huffman decoder and then through the run length decoding.

21.) And finally fixed length decoding is done and the characters are retrieved and compared with the input sample text file

22.) And the Symbol error rate is calculated and plotted

Note: The Energy Eb in all the above cases is to be normalized by dividing it by the length of the interval Tb because MatLab stores discrete values.

Observations:

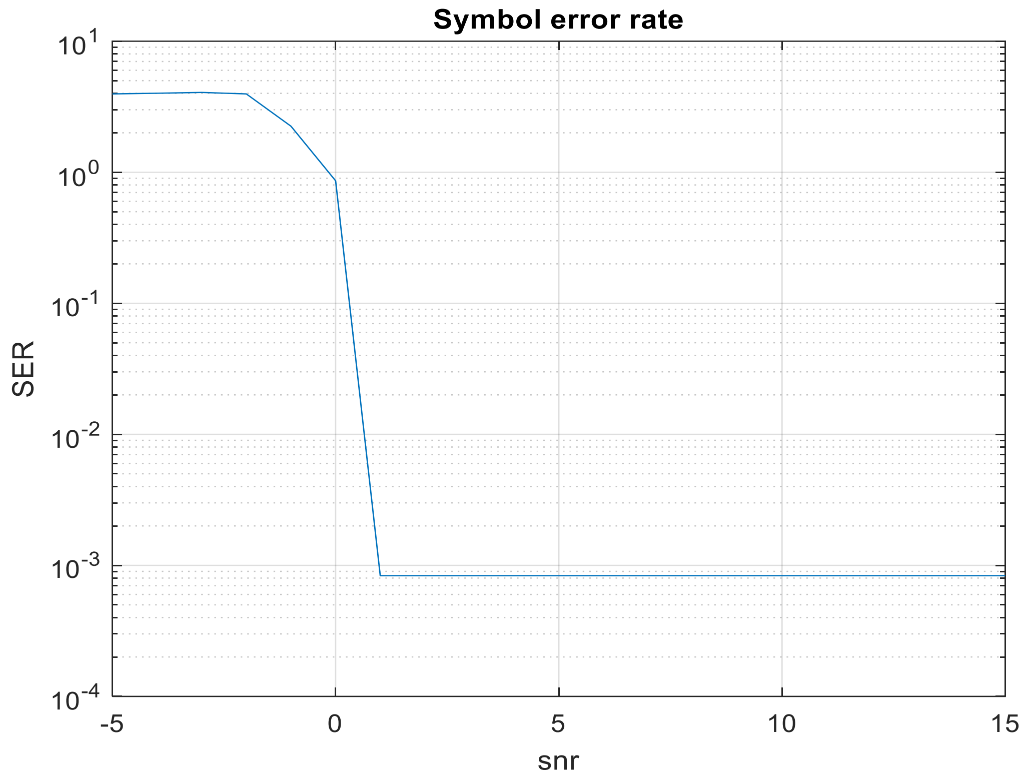
- The compression ratio determines the reduction in Bits and we can observe it to be in the range 1<x<2.
- The plot of the SER is generally high for negative snr and as the value of SNR increases there is a gradual fall in the SER.
- The SER depends on a lot of sub blocks , such as the detector, Channel Encoder and Decoder and source encoder and decoder
- The usage of a (10,5) Hamming Code and the corresponding parity matrix H gives a d* = 3,
  which results in error detectability = d*-1=2,
  and error correctability = (d*-1)*0.5=1
- The resulting errors for the cases of negative SNR changes the length of the Huffman decoded sequence and for the sake of decoding we truncate values or do 1’s padding to make it equal to the length of initial sequence
- The plot is generally sharp and has edges when plotted and quite random in nature but the SER decreases gradually , to ensure that the randomness is reduced we take the average of a set of simulatuions.

Results:

SER / SNR	SER_1	SER_2	SER_3	SER_4	SER_5	SER_6	SER_7	SER_8	SER_9	SER_10	SER_Avg
-5dB	0.7692	0.85	0.803	0.7722	0.7793	0.7793	0.821	0.8262	0.8277	0.831	0.8059
0dB	0.000417	0	0	0	0.00041	0	0.00041	0.138	0	0	0.0139
5dB	0.000417	0.000417	0	0.0004	0	0.0004	0.0004	0.0004	0.0004	0.00041	0.00033
10dB	0	0.000417	0	0	0	0.0004	0.0004	0.0004	0.0004	0	0.0002085
15dB	0.000417	0	0.0004	0.0004	0.0004	0	0	0	0	0	0.00016

Compression ratio achieved for 5 simulations:

1.0006, 1.0002, 1.0039,1.0011,1.0017



Conclusions:

Hence to conclude with the digital communication system with all the sub blocks were understood and implemented using MATLAB. The Symbol error rate reduces as the SNR increases form -5 to 15 dB.