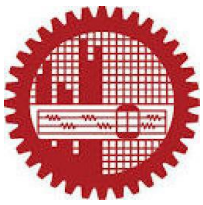


Bangladesh University of Engineering and Technology

Course Number: EEE-437

Course Name: Digital Communication

Name of the Assignment: Building a virtual but complete Digital Communication System using MATLAB

Date of Submission:

02/06/2017

Name: Ehsan Ahmed Dhrubo

Student Id: 1206100

Department: EEE

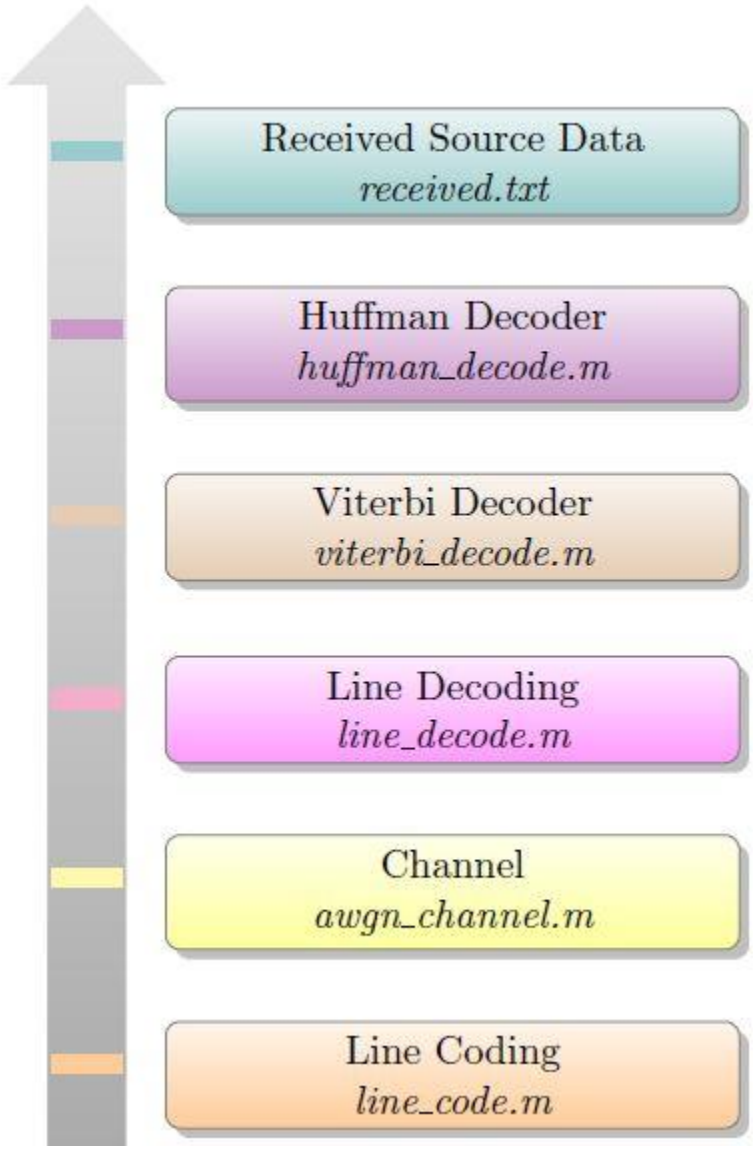
Level: 4 Term: 2

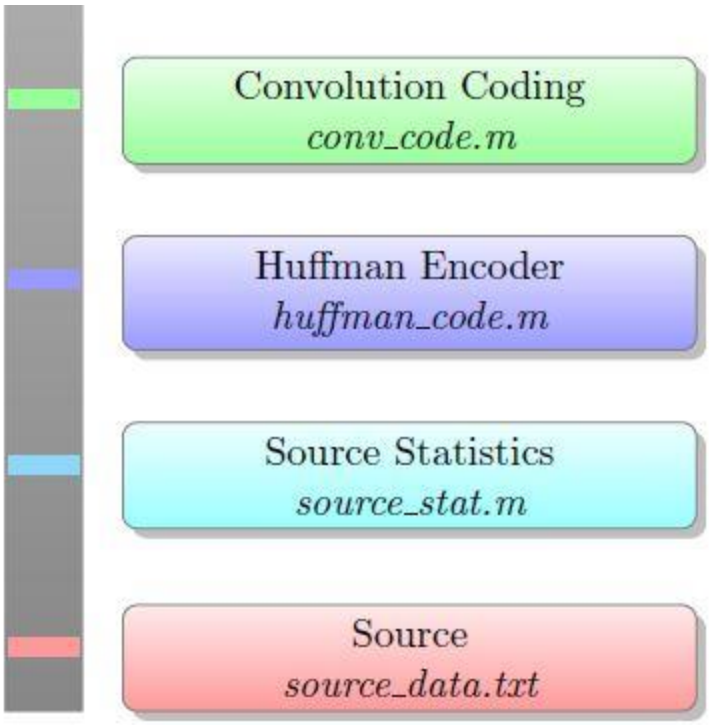
Objective:

The aim of this assignment is to build a virtual, but complete digital communication system using MATLAB.

Theoretical Description of Implemented Blocks:

The flow chart of the digital communication system is presented below with a bottom-up flow chart.





The succinct theoretical description of the blocks and functions that are implemented in my MATLAB code is given below-

1. Master Script (main_file.m):

This is my master script which calls different functions to do certain tasks. First it calls the source_stat.m function to provide "source_data.txt" file in it and eventually it calls huffman_code.m, conv_code.m, line_code.m, awgn_channel.m, line_decode.m, conv_decode.m and huffman_decode.m. After calling huffman_decode.m function, received.txt file is achieved and we can design the whole digital communication system virtually.

2. Source Statistics (source_stat.m):

This function takes "source_data.txt" in its input and returns the symbols of the entire text file and its corresponding probabilities. This function first reads the given file by 'fopen' function. Then it counts the number of occurrences of symbols and finds their corresponding probabilities by dividing with the total occurrences.

3. Huffman Encoder (huffman_code.m):

This function takes the symbols and their corresponding probabilities calculated earlier as inputs and returns their corresponding huffman coded bit and the entire bit stream of the given text files. It first does huffman coding operation with the symbols and probabilities. Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters; lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code. The variable-length codes assigned to input characters are Prefix Codes means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not prefixed of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bit stream. Then it finds out the Shannon Entropy, Average code length and efficiency of the code. Then it converts individual coded bit into an entire bit stream of the text file so that the entire bit stream represents the given text file.

4. Convolution Coding (conv_code.m):

This function takes previous entire bit stream as input and returns encoded bit stream based on convolution coding. Convolutional codes offer an approach to error control coding substantially different from that of block codes. A convolutional encoder encodes the entire data stream into a single code word, maps information to code bits sequentially by convolving a sequence of information bits with "generator" sequences and does not need to segment the data stream into blocks of fixed size (Convolutional codes are often forced to block structure by periodic truncation). Convolution coding is a machine with memory and is based on construction techniques. This convolutional encoder consists of a shift register with kL stages where L is called the *constraint length* of the code. At each instant of time, k-information bits enter the shift register and the contents of the last k stages of the shift register are dropped. After the k bits have entered the shift register, n-linear combinations of the contents of the shift register are computed and used to generate the encoded bits. From the above coding procedure, it is obvious that the n-encoder outputs not only depend on the most recent k bits that have entered the encoder but also on the $(L-1)k$ contents of the first $(L-1)k$ stages of the shift register before the k bits arrived. Therefore, the shift register is a finite state machine with $2^{(L-1)k}$ states. Because for each k-input bit, we have n-

output bits, the rate of this code is simply $R_C = \frac{k}{n}$. For our code the code words or generator sequences are fixed and they are g1=[1 0 0], g2=[1 0 1], g3=[1 1 1]. So, in our code, n=3, k=1 and L=3. Thus, $R_C = \frac{1}{3}$. There is function named out_func.m in our code in order to create generator sequences g1, g2 and g3 by the operation of modulo 2 addition. In our code, bit rate, Rb is 500 bit per sec and samples per bit, k is 20.

5. Line Coding (line_code.m) :

This function takes entire bit stream and encoded bit stream of convolution code as input and returns modulating time domain signal as output. This function also gives options to choose between different method of line coding as BPSK, BFSK,BASK and GMSK. For each method of line coding the encoded bit stream is first sampled and modulated to corresponding line coding. Then a sinusoidal signal is multiplied with it and the line coded time domain signal is transmitted.

6. Channel (awgn_channel.m):

This function takes the modulating time domain line coded signal and SNR (signal to noise ratio) as input and returns noisy signal as output. It mixes additive white Gaussian noise with the modulating signal through the virtual channel and transmits this noisy signal to the receiver side. The amount of noise is varied with SNR.

7. Line Decoding (line_decode.m):

This function takes the noisy received line coded time domain signal, entire bit stream and the method of line coding as input and returns a demodulated bit sequences as output. It demodulates the noisy signal and builds bit sequences of corresponding method of line coding.

8. Convolution Decoding (conv_decode.m):

This function takes previous demodulated bit sequences as input and returns decoded bit sequences based on Viterbi algorithm as output. It converts the bit stream into decoded bit sequences using Viterbi algorithm. The decoding

algorithm uses two metrics: the branch metric (BM) and the path metric (PM). The branch metric is a measure of the "distance" between what was transmitted and what was received, and is defined for each arc in the trellis. In hard decision decoding, where we are given a sequence of digitized parity bits, the branch metric is the Hamming distance between the expected parity bits and the received ones. The path metric is a value associated with a state in the trellis (i.e., a value associated with each node). For hard decision decoding, it corresponds to the Hamming distance over the most likely path from the initial state to the current state in the trellis. By "most likely", we mean the path with smallest Hamming distance between the initial state and the current state, measured over all possible paths between the two states. The path with the smallest Hamming distance minimizes the total number of bit errors, and is most likely when the BER is low. The minimum Hamming distance is found when the summation of path metric and corresponding branch metric is minimum. In our code, we have used Path_Metric.m function to calculate path metric, branch metric, sum them and find out the minimum Hamming distances. Then we have formed trellis diagram to calculate the shortest path for convolution decoding. Thus we can get the decoded bit sequences from Viterbi algorithm.

9. Huffman Decoder (huffman_decode.m):

This function takes the decoded bit sequences, symbol and code as input and returns decoded message. It decodes the code obtained from Viterbi algorithm and returns the decoded message as received text file. It first converts the decoded bit sequences into a long character string. Then it uses huffman decoding process to extract the original message and return it into a text file. Thus a source text file is transmitted into a virtual channel and is received in the form of another text file through huffman coding, convolution coding, line coding, noise mixing, line decoding, convolution decoding and huffman decoding processes.

1. In this code, for high SNR (SNR>0db) the BER(Bit Error Rate) is almost zero. But for low SNR(SNR<-30db) BER is very high and almost equal to the input bit stream. In our code, for almost -10db SNR BER is nearly zero. If we decrease SNR any further, BER will be higher. For about SNR= -20db BER is for which at least 40% of the symbols in the source and received text does not match.

2. The presence of channel coding (convolution coding) have an important effect on the error rate of the system. For high level SNR, convolution coding works tremendously well and for very low level SNR, it does not perform so well comparing to without convolution coding.

3. The error rate changes for same level of SNR but different methods of Line coding. M-ary PSK has the lowest error rate compared to any other Line coding. Then M-ary FSK gives lower error rate, then M-ary ASK gives lower error rate and GMSK gives high error rate compared to other Line coding.

4. Huffman coding helps to decrease the error rate. If we do not use huffman coding, error rate will be higher. If huffman coding is used, error rate will be comparatively much lower as it compresses data bit and transmits the signal at a lower data bit. Thus noise and interference do not affect the coded signal much.

5. As the generator sequence of my code is fixed, effect of the error rate for changing the generating sequence of the convolution code cannot be determined.