PROGRAMMER'S MANUAL

This manual gives a brief description of the code which was developed to perform Convolutionally Encoding & Viterbi Decoding for AWGN channel.

The functions used in the program and their functionalities are listed below

int cti(char c)

----- This function is used to convert a character to integer in this code. It is mainly used to convert the file contents(which we usually read in terms of character) and convert it to integer.

void genranddata (int framesize)

----- This function is used to generate the randon data input which is fed to the convolutional encoder ,basing upon the frame size.

void shiftnencode(int k, int K,int n)

----- This function is called from the encode() function. This function takes in k bit blocks of input and encodes them, the resulting bits are first stored in a array and then stored in a file.

void encode()

----- This function marks the beginning of the encoding process, the final few bits of the input sequence is first initialized and then the states are calculated. We then call the shiftnencode () function for every k bit block

float gauss(float mean, float sigma)

----- This function is used to calculate the Gaussian output when the mean and sigma values are passed from the addnoise function. This function returns the Gaussian output to the addnoise function.

void addnoise()

----- This function is used to add the Gaussian noise (obtained from gauss() function) to the encoded bits after converting the encoded bits into +1 and -1 from 1 and 0 respectively.

int btd(int *b,int size)

----- This function is used for conversion of binary into decimal

void dtb(int d,int x,int y,int *b)

----- This function is used for conversion of decimal to binary.

void extract(int x,int y,int *source,int *destination)

----- This function is used for extraction of particular bits from source and place it in the destination. This function is called in the constructstatetable() function and is used to determine the Next State matrix

void constructstatetable()

----- This function is used to construct the State Input Matrix, State Output matrix, next State Matrix which are required in the decoding process.

void decode

----- This function is used for the decoding process. We first open the "gaussop.txt" and read in 'n' bits at a time.Then starting from the tail, we proceed forward along the trellis by calculating the winning path at each node. At each node we store the value of the node from which we obtain the winning path and store it in the structured array as stprev[x1][t].prev ,the accumulated error metric as stprev[x1][t].accum, the best transition input as stprev[x1][t].input, where x1 is the set of current nodes at time instant t. If there are no transitions to the current node, then for that particular node we make stprev[x1][t].value = -1.

float euclidean(int op,float *l)

----- This function is used to calculate the Euclidean(free) distance between the given outputs.

int statetrans(int i,int j,int *bt)

----- This function is used to calculate the minimum distance input between the given two states

void calculatebiterrorrate()

----- This function is used to calculate the Bit error rate (BER) ,which is calculated from the inputfile("input.txt")and the decoded output file("decoded.txt") and store it in a log file.

void main()

---- Input the values of k, n, K, polynomials, Packet Length & No.of Frames (Basing upon the constraints specified in the user manual and perform the following, till all the frames are completed

- Call the genranddata() function to generate random numbers
- Encode the input by calling the encode() function
- Add Noise to the encoded sequence by calling the addnoise() function.
- Construct the State table by calling the constructstatetable() function
- Decode the noise added sequence by calling the decode() function
- Calculate the Bit error rate by calling the calculatebiterrorrate () function.

This is the data structure used for representing a node at a time instant 't' during decoding.

```
struct staten
    {
        int prev;
        int value;
        int input;
        float accum;
    };
```

GLOBAL VARIABLES

int **sim	----	State Input matrix
int **som	----	State Output matrix
int **nsm	----	Next State matrix
k	----	Input bits
n	----	Output bits
pow1	-----	2^(K-k) i.e. No. Of states
pow2	-----	2^(k) i.e. No. Of Inputs
limit	-----	Total no. Of transitions
dbs	-----	Input File Size
ofs	-----	Output File Size
arr[]	-----	Array to store the k input bits
a[][]	-----	For Polynomials Representation
tail[]	-----	The final K-k bits of the input sequence
shiftreg[]	-----	Array to store the shift register contents
str[]	-----	Array to store the floating point Gaussian Output's
h		