Project Title: **Convolutional Encoder and Tail-Biting Viterbi Decoder (AWGN Channel)**

Project Associates:

Venkata Thippanna Rao. P. R.	2005145
Vidyadhar. N	2005147
Vikas. C.	2005148

Project Statement:

To Implement a Convolutional Encoder & a Tail-Biting Viterbi decoder for the AWGN channel. .

Procedure:

The specifications of a Convolutional Encoder are, there are k input bits, which produce n encoded output bits, consequently the code rate is defined as k/n. The constraint length K is given as m+1, where m is the no. of delay elements

In case of normal Viterbi decoder we append zeroes (called as the tail) to the flush the bits in the encoder, which needs to be transmitted. Due to this addition of zeroes, the bandwidth required for transmission increases. In order to reduce the bandwidth, we go for **TAIL-BITING VITERBI DECODER**.

In case of a tail-biting Viterbi decoder we initialize the delay elements with the last K bits from the frame. The first K-k bits of the shift register becomes the starting state or the initial state of the encoder. The total K bits are then bitwise multiplied and bitwise XORed with the n generator polynomials, thereby producing n encoded bits. Now as the next k bits come in, we shift the shift register contents k bits to the right ,now the right most K-k bits give the next state and the bitwise multiplication and bitwise XOR with the generator polynomial gives n bit encoded sequence , this process continues until the end of the frame. We note down the states and the encoded output for every k bit input given to the encoder and construct a state table which gives you the next state and the output when a particular input is given.

Given a uniform random variable U, a Rayleigh random variable R can be obtained by:

$$R = \sqrt{2 \cdot \sigma^2 \cdot \ln(1/(1-U))} = \sigma \cdot \sqrt{2 \cdot \ln(1/(1-U))}$$

Where σ^2 is the variance of the Raleigh random variable, and given R and a second uniform random variable V, two Gaussian random variables G and H can be obtained by

$G = R$ cos V and $H = R$ sin V

In the AWGN channel, the signal is corrupted by additive noise, n(t), which has the power spectrum *No*/2 watts/Hz. The variance σ^2 of this noise is equal to $No/2$. If we set the energy per symbol E_s equal to 1, then $E_s/N_0 = 1/2\sigma^2$. So $\sigma = \sqrt{1/(2 \cdot (E_s/N_0))}$.

For decoding the encoded sequence we first draw a trellis diagram with the help of the state table obtained and at every time instant, we calculate the Euclidean distance corresponding to every path arriving at a particular node.

$$D_{euclidean} = \sqrt{\sum (x_i - y_i)^2}$$

Add this to the error metric and declare the path with the least error metric as the winning path and discard the remaining paths. Perform the above operation till the end of frame is reached. Now trace back via the winning paths declared from the final time instant to the initial time instant. The input bits obtained along the path will result in the decoded sequence.

Implementation Algorithm:

Step 1: Input Rate of Encoder (k, n)

Step 2: Input Constraint Length K

Step 3: Input the Packet size of the Channel

Step 4: Input the Number of Frames to transmit over the Channel (nof)

Step 5: Encode (Encode the input and Transmit the Packet over the channel)

Step 6: Add Noise (Channel adds this)

Step 7: Construct the State Transition Table.

Step 8: Decode (Decode the Received Packet)

Step 9: Repeat Step 5 through 7, nof times

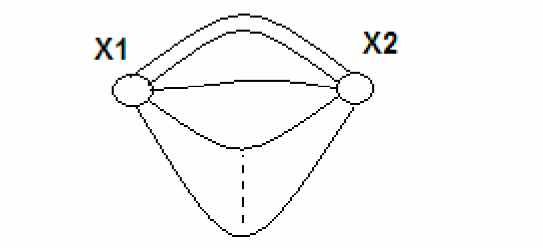
Step 10: Calculate the Bit Error Rate comparing the input and output file.

Step 11: Stop

Calculation of Winning Path and Best Transition:

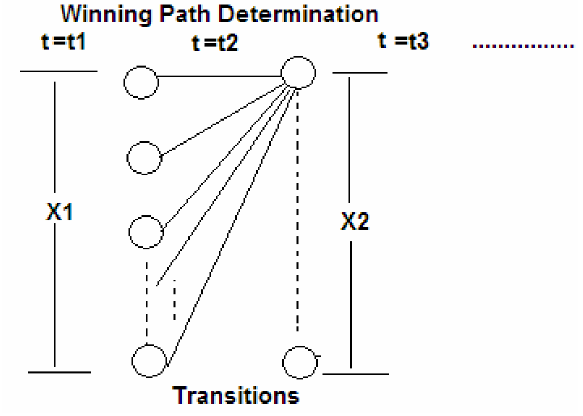
- Time instant t, previous node x1 and present node x2
- Data Structure used to represent the nodes in the Trellis
    - struct staten { int prev; int value; int input; float accum; };
- The case of Multiple transitions from particular x1 to x2 leads to the calculation of Best Transition among the various transitions from particular x1 to x2 using Euclidean distance.

Best Transition Between Two States



- The case of determining the Winning Path at x2 from various x1s to x2 is by considering the minimum Euclidean distance.

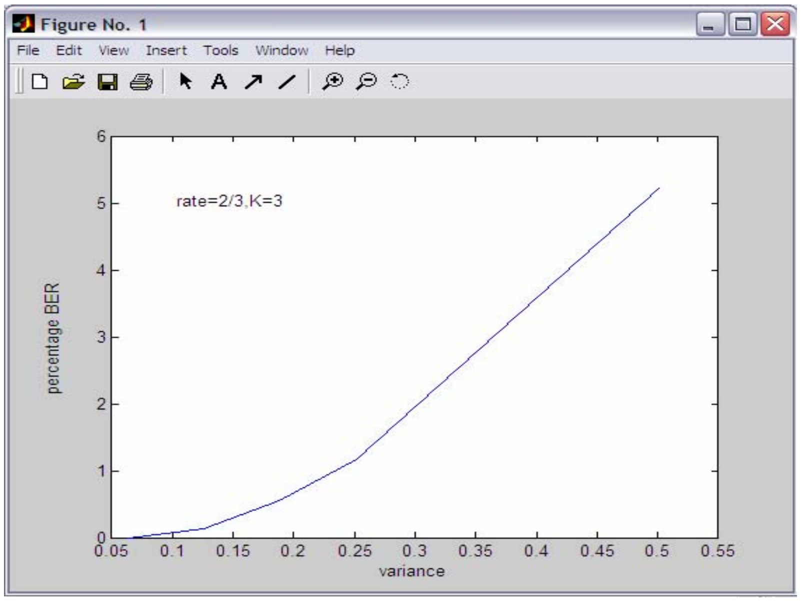
Winning Path Determination



- Using the Data Structure we back track from the final state (tail) to the starting state (tail).

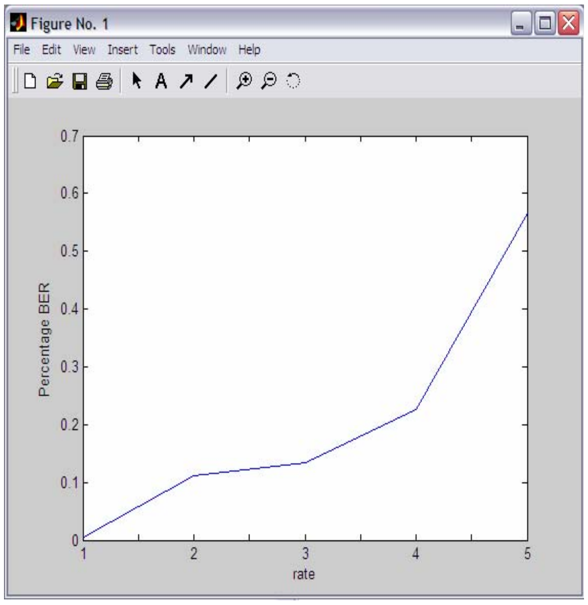
Observation:

1. Variance Vs Percentage Bit Error Rate Plot:



Polynomials considered are**:** (1 0 0, 0 1 1, 1 0 1)

2. Rate Vs Bit error Rate Plot:



On X- axis

1 => k=1, n=2, K=5
(10000, 01000)
Packet Length=10000*10
2 => k=2, n=3, K=5
(10000, 01100, 10100)
Packet Length=9999*10
3 => k=3, n=4, K=5
(10000, 01000, 00100, 00010)
Packet Length=10000*10
4 => k=4, n=5, K=5
(10000, 11100, 10101
01010, 00011)
Packet Length=10000*10
5 => k=5, n=6, K=7
(1000000, 0110011, 1100110,
0011001, 1010101, 0101010)
Packet Length=12000*10

3. When a 1Mb file is given as input:

k=1,n=2,K=3
Polynomials: 1 0 0
1 0 1
Packet Length: 2000000
Number of packets Sent: 01
Bit Error Rate: 0.00000% (for a given Noise Level i.e.., Energy Per Symbol=1 and Variance=0.1255)

4. Tracing out the Reasons for Errors in Decoding (Inference):

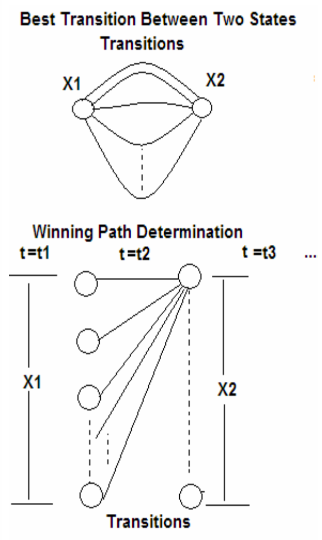
Case 1: Best Transition (x1 to x2) Selection Ambiguity.

K-k is small- Number of states between which transition takes place is small. 2 raised to k inputs-which leads to multiple transitions between any two states. So, Increase n constraint length increases K-k keeping k constant. Less will be the probability of error due to multiple transitions (polynomial factor).

k=2, n=3, K=3 (100,101,110) BER=0.150038%.
k=2, n=4, K=3 (1000,1011,0100) BER=0.0%
k=2, n=3, K=4 (100,101,110,001) BER=0.09%
k=3, n=4, K=4 (1000,1010,0101,1011) BER=0.33%
… … … … … … … …

Case 2: Winning Path Selection Ambiguity. (Due to Noise Addition)

Case 3: Role of Polynomials.
Case … … .


Best Transition Between Two States
Transitions
X1
X2
Winning Path Determination
t=t1
t=t2
t =t3
...
X1
X2
Transitions

5. Another Example:

K=10, n=15, K=12
Polynomials:
1 0 0 1 0 1 0 1 1 1 0 1
1 0 1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1 0 1
1 0 1 1 1 1 1 0 0 0 0 1
1 0 0 0 0 1 1 1 1 1 0 0
1 0 1 0 1 0 1 1 1 1 1 0
0 0 0 1 1 1 1 1 1 0 0 1
1 0 1 0 1 0 1 1 1 1 1 1
0 1 1 1 0 1 0 1 1 0 1 0
1 0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 0 0 0 0 0 0
1 0 1 0 1 0 0 0 0 1 1 0
1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1

Packet Size: 15,000
Number of Packets Sent: 10
Bit Error Rate: 0.04600%
Real: 33m35.108s
User: 2 m48.833s
Sys: 0m49.448s