Digital Communication Laboratory

WEEK - 9 Report

Date: 23rd march 2023 Kadagala Sai Siva Sankar (20EC30022)

Experiment number: 7 Group number: 26(Wednesday batch)

Title: Viterbi Decoding for Noisy Binary Channels

Objective:

- To generate a random signal and encode it using a convolution encoder.
- Generating a received signal after passing the encoded message through binary symmetric and asymmetric channels .
- To decode the received signal using Viterbi decoding algorithm.
- To plot bioterror rate vs crossover probability plots for different probabilities.

Implementation:

- The purpose of the code is to simulate a communication channel and observe the effect of channel noise on the decoded data.
- Generate a random bit string of size N=256 bits using rand functions.
- Pad 2 zeros at the start and end of the data to get full encoded message.
- C is a vector of zeros with a length of 768 (3 times the length of D) that will store the encoded data, and reg is a vector that will store the state of the encoder.
- Write a loop that performs convolutional encoding on the data by iterating over the D vector, updating the encoder's state (stored in the reg vector), and computing the encoder's output for each input bit. The output is computed using the generator polynomials g1, g2, and g3, and is stored in the C vector.
- The section **%% Trellis** sets up the trellis structure for the Viterbi algorithm. It specifies the next state and output for each state and input.
- The section **%% Received Data** simulates the transmission of the encoded data through a noisy channel. It adds noise to the encoded data using a binary symmetric channel (BSC) with different error probabilities. The received data is stored in the variable rsig.
- The section **%%Decoding**: Viterbi algorithm uses the Viterbi algorithm to decode the received data and recover the original bit sequence. The algorithm iterates through the received data and tries to find the most likely sequence of transmitted bits that produced it. The decoded data is stored in the variable decData.
- Generate a plot of bit error rate as a function of crossover probability for different values of probability.
- Generate a plot of the bit error rate as a function of the crossover probability for different values of the constant error probability p0. It uses the **basc** function to add noise to the encoded data with different error probabilities.
- The **basc** function adds noise to the encoded data with different error probabilities depending on the value of the transmitted bit. If the transmitted bit is 0, it adds noise with probability p0, otherwise it adds noise with probability p1.
- The **viterbi** function decodes the received data using the Viterbi algorithm. It takes the received data, trellis structure, and the length of the transmitted data as input, and returns the decoded data.

Outputs:

Generated trellis

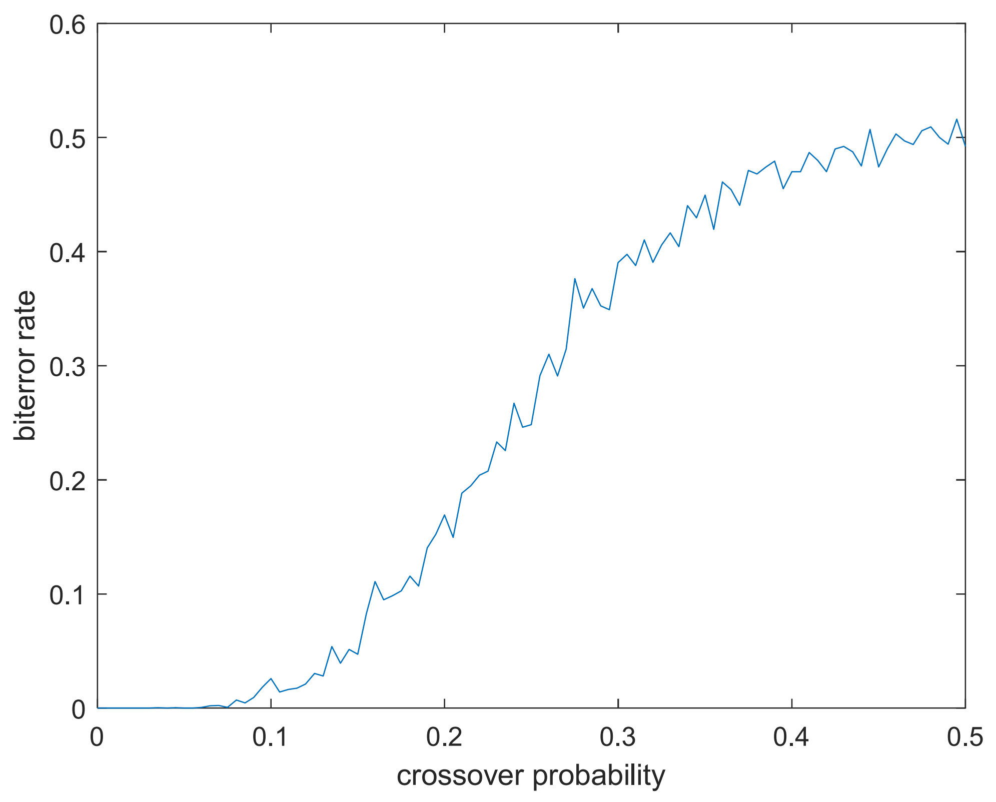
```
nextState    outputs
_________    _______

 0     2      0     3
 0     2      7     4
 1     3      6     5
 1     3      1     2
```

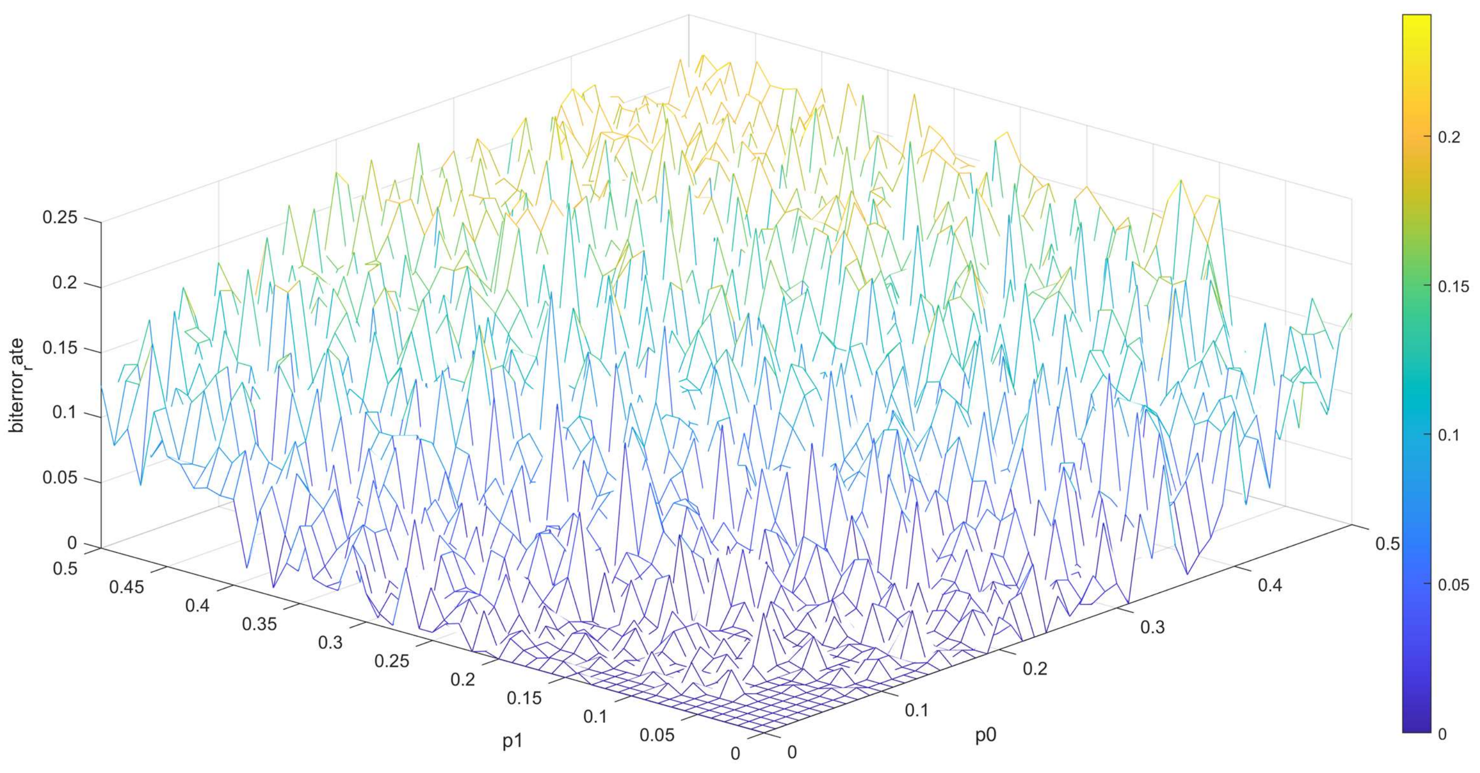
(1,1) element of nextstate matrix represents the next state of the register after input bit 0 has been shifted into the register while (1,1) element of outputs matrix represent the corresponding output convolution coder gives.

Parallelly, (1,2) element of nextstate matrix represents the next state of the register after input bit 1 has been shifted into the register while (1,1) element of outputs matrix represent the corresponding output convolution coder gives.

Crossover probability vs bit error rate (runs =10) and N =256(bsc)



Crossover probability vs bit error rate (runs =4) and N =256(basc)



Discussion:

- To make the curve smooth, bit error rate must be calculated for a greater number of runs and can be averaged.
- As we can observe from the bit error rate vs crossover probability plot for bsc, we see that it is a increasing function of probability. Also when the probability is more than 0.5 we must use a inverse decoder since probability of 1 flipping to 0 is greater when crossover probability is greater than 0.5. So, ideally the plot should have a maximum at 0.5 and be symmetric about x = 0.5.
- Parallelly, for the basc case, as p0 and p1 increase the bit error rate increases as expected.

- Binary asymmetric channel can be realized by using 2 binary symmetric channels and appending output of one channel (for data =1 case) and appending output of other channel (for data =0 case).
- Since for the same p, Z = bsc(d,p) gives different outputs , directly calculating the bioterror rate isn’t a good idea(as it is random and we wont get a smooth curve). So, a greater number of runs and averaging the bit error rate gives us expected value of the error rate.