EXPERIMENT NO. 07

TITLE: Viterbi Decoding for Noisy Binary Channels

Digital Communication Lab
Department of Electronics & Electrical Communication Engineering
IIT Kharagpur

PROBLEM STATEMENT: In this software experiment, you need to devise an error correction code for communication over noisy binary channels. In particular, construct a convolutional code, viz., encoder and decoder using Viterbi's algorithm.

KEY COMPONENTS: This is a software experiment. You can use standard functions in MATLAB.

BRIEF THEORY:

Convolutional codes are a class of error-correcting codes used in digital communication systems to detect and correct errors that occur during data transmission. They are called "convolutional" because they are based on the mathematical operation of convolution.

Convolutional codes work by adding redundancy to the transmitted data in a way that allows the receiver to detect and correct errors caused by noise or other forms of interference. This is done by encoding the data using a shift register and a set of predefined generator polynomials. The encoder takes in a stream of input bits and produces a corresponding stream of output bits, which are sent over the communication channel.

At the receiver, a similar shift register and set of generator polynomials are used to decode the received data. To minimize the probability of error, the optimal decoding is maximum likelihood decoding. Hence, the decoder uses a maximum likelihood algorithm to find the most likely original sequence of bits that could have generated the received data. For the convolutional code, Viterbi's decoding is an instance of maximum likelihood decoding (as explained later). If errors are detected, the decoder corrects them by applying the inverse of the convolutional code to the received data.

Convolutional codes are widely used in modern digital communication systems such as satellite communication, cellular networks, and digital television broadcasting. They are efficient, robust, and can achieve high levels of error correction with relatively low overhead.

Encoding and Decoding for Convolutional Codes:

Here is a simplified view of the encoding and decoding process for a convolutional code.

Encoding:

1. The input data stream is divided into fixed-length blocks of k bits.
2. A shift register with N stages is initialized with N - 1 zeros.
3. The input data is fed into the shift register one bit at a time.
4. At each clock cycle, the contents of the shift register are multiplied by a set of generator polynomials to produce a set of m output bits, where m is the number of generator polynomials used.
5. The m output bits are transmitted (some terminating bits are added to force the shift register to a known state, often the initial state of the shift register). Here, we can

observe that for one input stream of bits, the encoder gives m stream of bits which is equivalent to say that the rate of encoding is 1/m.

Decoding:

1. The received data stream is fed into a shift register with N stages.
2. The shift register is initialized with the received data.
3. At each clock cycle, the contents of the shift register are multiplied by the same set of generator polynomials used in the encoding process to produce a set of m output bits.
4. The m output bits are compared with the received output bits to detect errors.
5. If errors are detected, the decoder uses a maximum likelihood algorithm to find the most likely sequence of input bits that could have generated the received data.
6. The decoded data is then outputted.

Viterbi's Decoding Algorithm: In essence, in a convolutional code, a sequence of input bits is encoded into a longer sequence of output bits by convolving the input sequence with a set of predetermined "encoder polynomials". The resulting sequence of output bits is transmitted over a noisy channel, e.g., a binary symmetric channel (BSC), which may introduce errors into the transmitted sequence.

To decode the transmitted sequence, the receiver uses a popular technique called *Viterbi's algorithm* to determine the *most likely sequence of input bits* that could have generated the observed output sequence (recall that noise flips bits, thereby causing the received sequence to look different from the transmitted one), given the encoder polynomials and the properties of the noisy channel.

The key idea behind Viterbi's algorithm for decoding convolutional codes is to use dynamic programming to efficiently compute the most likely sequence of states in a bottom-up fashion, and then to perform a traceback operation to determine the most likely sequence of input bits that could have generated the observed output sequence.

FORMAL DESCRIPTION OF THE EXPERIMENT:

1. Generate a random bit string of size N=256 bits. Let bit string, say **d** is your data or message string.
2. Generate a rate R=⅓ convolutional code with generator polynomials
   a. g_1= [1 1 0]
   b. g_2= [1 1 1]
   c. g_3= [1 0 1]
3. Perform encoding as per standard operations for the convolutional code to generate the codeword **c.** What is the length of the codeword?
4. Now design a Viterbi decoding algorithm. Refer the notes in class or any standard reference to proceed with the Trellis design followed by the path optimization for Viterbi decoding.

5. Now construct a binary symmetric channel (BSC) with a crossover probability p to perturb **c**. Essentially, generate a *noise vector* **z** (length equals that of **c)** where each component in the vector is generated randomly, independently and with an identical distribution of Bernoulli (p). Now generate **y=z+c**, where addition is modulo-2. Here **y** represents the output received over a BSC(p). Consider at least 5 different values of p (low to high, between 0 to 1/2) in this experiment and determine the recovered or estimated message.
6. *Bit error rate:* Repeat the experiment and plot the bit error rate (fraction of bits is error in the data block) as a function of channel crossover probability *p*.
7. *There will be a second part to this experiment which will be discussed in class.*

OUTCOMES AND RESULTS

1. Include step-by-step results and plot as necessary.