DEPARTMENT OF EE - IIT MADRAS

Assignment 4
Viterbi Decoding of Convolutional Codes

Surajkumar Harikumar (EE11B075)

November 30, 2014

1 PROBLEM STATEMENT

Implement the soft-decision Viterbi decoder for a convolutional code of rate 1/2 and memory order 2. Plot the BER-SNR curve for the decoder, and compare with uncoded transmission.

2 ENCODING

The convolutional code used here has the base generator matrix

$$G = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \;\; ; \;\; G(D) = \begin{bmatrix} D \\ 1 + D + D^2 \end{bmatrix} \tag{2.1}$$

The feedforward encoder diagram is shown in Figure (2.1) and the state transition diagram is shown in Figure (2.2).

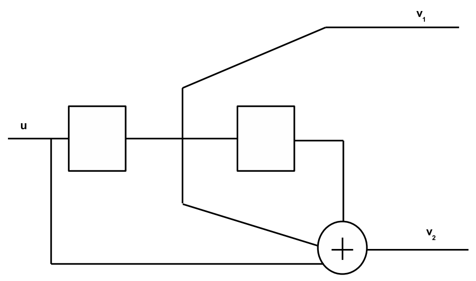

Figure 2.1: Feedforward encoder for the specified convolutional code of memory order 2

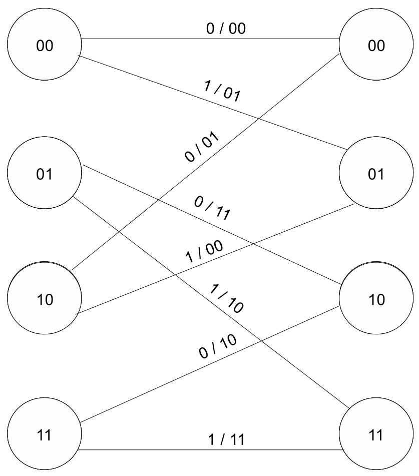

Figure 2.2: State Diagram for the convolutional code

For the decoding, we assume the all-zero input message. We pass it through the trellis encoder to obtain the output. Note that for an input message of length 10^4, the codeword length is $N = 2(\frac{k}{R} + mem) = 2 * 10^4 + 4$.

For code simplicity, we store 2 arrays, one for the state-nextstate-output relations for input 0, and one for input 1. We take the Generator matrix as input, and create the trellis diagram

We then use BPSK modulation to obtain the codeword sent through the channel. We use the code in [1] to simulate the AWGN channel for a given SNR.

3 Viterbi Decoding

We use the soft decision Viterbi-algorithm for decoding. We repeat the state-diagram and trace a path through the trellis corresponding to the shortest distance codeword. Rather than making hard-bit-decisions in each stage, we use a branch-metric, computed by the closeness of the received vector to input $\{0,1\}$. It can be shown that over the AWGN channel, the metric $\frac{2r}{\sigma^2}$ corresponds to the ML decision. So, we just use the metric as r, the distance from the outputs corresponding to input $\{0,1\}$, that is we use Euclidian distance as our metric.

The script used to do this is **viterbi.m**. We store the path metrics for each state in *SM*. We compute for each state, the branch metrics *BM0,BM1* corresponding to the 2 inputs. We then compute the next state path metric as the smallest metric of incoming branches, stored in *SM_next*. We check for the path with the smallest metric, and use that to get the closest codeword. It performs encoding decoding, and generates the BER-Curve. We sent a

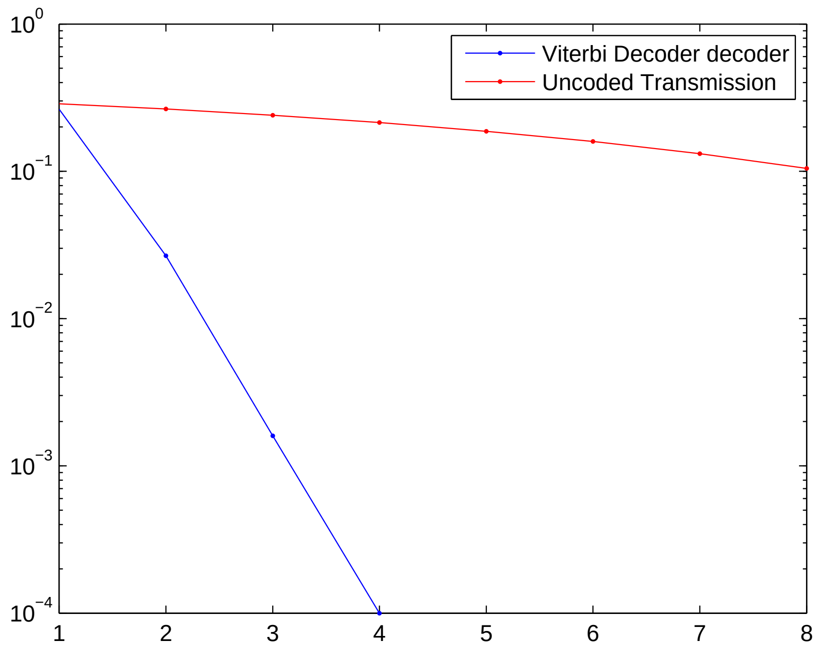

Figure 3.1: Comparison of BER-SNR curves for the Viterbi Decoder, and Uncoded BPSK transmission

message of length 10^4. Figure (3.1) shows the BER-SNR curves. .

REFERENCES

[1] AWGN Generation in MATLAB *addAWGN.m* - http://stackoverflow.com/questions/23690766/proper-way-to-add-noise-to-signal