Kevin Gilman

Bane Vasić

ECE 435A

23 February 2023

Implementation of Viterbi Detection Algorithm for Convolutional Codes

The Viterbi algorithm is used to decode convolutional codes sent over communication channels that suffer from noise that can corrupt transmitted data. Convolutional codes are used in communication systems to detect errors and correct them. They are often characterized by their code rate, which is a ratio of the number of inputs to outputs for the encoder. The Viterbi algorithm works by creating a trellis structure with nodes that represent the states of the encoder. These nodes have paths to future nodes representing the possible state transitions for the particular convolutional code that is being used. For future nodes, they will have multiple paths connecting to them from the current nodes, so the algorithm uses an accumulation metric to eliminate the worst paths. The accumulation score for a particular path is calculated by summing the accumulation score for the current connecting node with the hamming distance of the path. The hamming distance is calculated by finding the number of differences between the expected convolutional output for that particular state transition and the received output over the channel for that instance of time. The path leading to the smallest accumulation score for a future node is kept and the other paths are eliminated to maximize the likelihood of the algorithm choosing the correct path for decoding.

In my implementation of the Viterbi algorithm, I used two different convolutional codes and simulated transmission of the encoded message over two different types of communication channels. The first type of channel that I used to simulate transmission was the binary symmetric

channel with an adjustable parameter α, representing the probability of a bit being flipped across the channel. I also simulated transmission with an additive white gaussian noise channel with an adjustable parameter σ^2, representing the variance of the noise distribution.

I used python to create the software that was used to simulate the Viterbi algorithm for each convolutional code. The software allows the user to select a channel type and parameter value for simulation, automatically writing data for the chosen parameter and error rate of decoding to the appropriate csv file. The message size that I used for simulation was one hundred thousand bits that were all randomly generated. This message size was chosen to increase the accuracy of my simulated error rate results by decreasing the variability of the channel error rate with smaller values of α and σ^2. To simulate with this message size and provide sufficient time for the Viterbi algorithm to iterate before beginning to produce outputs, I used memory registers of length fifty for each state. The memory registers serve to store the decoded output bits for each state. Every time the worst paths are deleted for the new states that are generated, each new state updates its corresponding register by first using a temporary register to store the values for the state that connects to it, then it appends the transmitted bit for that path. If the temporary register is full, it left shifts all of the bits and stores the shifted out bit in another memory register with a length equal to the number of states. The new state then fills its corresponding register with the values from the temporary register, and then the temporary register is cleared. The need for the extra memory register containing the bits that are shifted out is so the algorithm chooses the value from this register that corresponds to the state with the lowest accumulation score for output. Once a value is chosen, this register clears so that it can be reused for the next generation of states for the trellis structure. The overall process of generating new states, creating connection paths, deleting worst paths, and updating registers continues until all of the received

encoded pairs have been compared. At the end, the state with the lowest accumulation score will append its entire memory register to the output. The software compares this decoded output to the original randomly generated bit stream to calculate an error rate for the chosen channel and parameter.

The first encoder that I used for simulating the Viterbi algorithm decoding was a (2,1,3) convolutional encoder with the two generator polynomials:

$$g^{(1)} = x^3 + x^2 + 1$$

$$g^{(2)} = x^3 + x^2 + x + 1$$

The encoder circuit for these polynomial generators uses three memory blocks and has a code rate of 1/2.

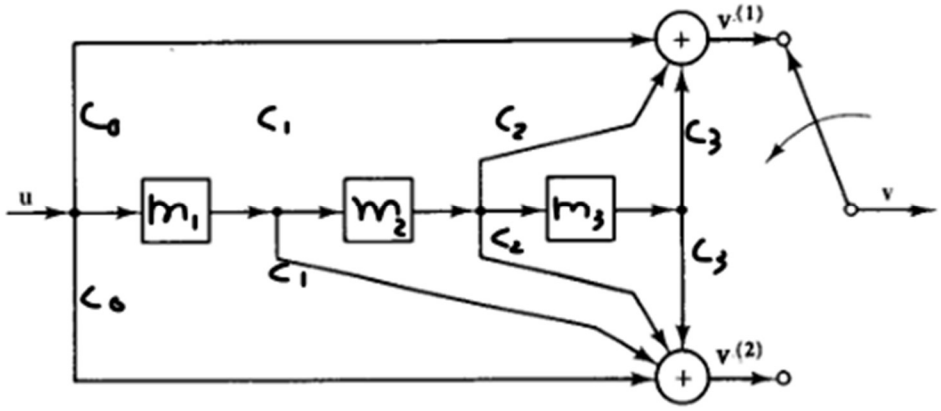

Figure 1: (2,1,3) binary convolutional encoder

To better visualize all possible states, transitions, and outputs of this encoder, a table and trellis structure is provided below.

u	S_n (m_1, m_2, m_3)	S_{n+1} (u, m_1, m_2)	V_1 $(u + m_2 + m_3)$	V_2 $(u + m_1 + m_2 + m_3)$
0	(0, 0, 0)	(0, 0, 0)	0	0
1	(0, 0, 0)	(1, 0, 0)	1	1
0	(0, 0, 1)	(0, 0, 0)	1	1
1	(0, 0, 1)	(1, 0, 0)	0	0
0	(0, 1, 0)	(0, 0, 1)	1	1
1	(0, 1, 0)	(1, 0, 1)	0	0
0	(0, 1, 1)	(0, 0, 1)	0	0
1	(0, 1, 1)	(1, 0, 1)	1	1
0	(1, 0, 0)	(0, 1, 0)	0	1
1	(1, 0, 0)	(1, 1, 0)	1	0
0	(1, 0, 1)	(0, 1, 0)	1	0
1	(1, 0, 1)	(1, 1, 0)	0	1
0	(1, 1, 0)	(0, 1, 1)	1	0
1	(1, 1, 0)	(1, 1, 1)	0	1
0	(1, 1, 1)	(0, 1, 1)	0	1
1	(1, 1, 1)	(1, 1, 1)	1	0

Table 1: (2,1,3) convolutional code state machine table

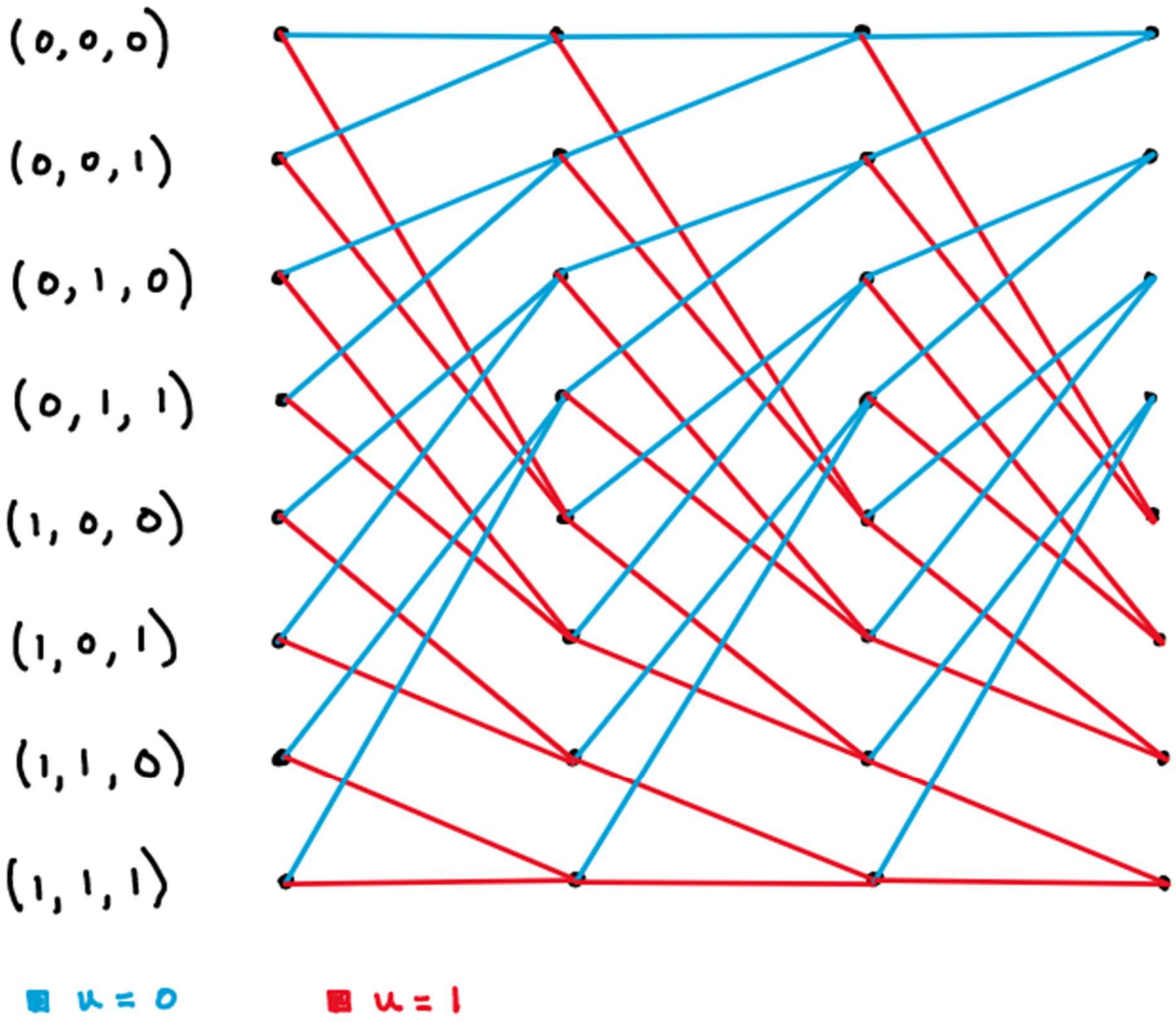

Figure 2: (2,1,3) convolutional code trellis structure

The simulation results of the Viterbi detection algorithm for the binary symmetric channel illustrated a high efficiency for error correcting at lower values of alpha. As the value of alpha increased, the performance of the algorithm began to deteriorate at a significant rate. An interesting observation that can be made when analyzing the data for bit rate error is that for alpha values greater than 0.12, the error rate of the algorithm becomes higher than the error rate of the channel. After extensive research, I discovered that the Viterbi algorithm can suffer at

larger values of α from a phenomenon known as error bursts. In the case of long error bursts from the channel, the Viterbi algorithm can select an incorrect path on the trellis with high confidence (Neal). This deviation from the correct trellis path has cascading error effects for decoding because the Viterbi algorithm is not designed to handle long error bursts. There are various techniques for handling error bursts such as interleaving. According to Neal, this method is highly effective for burst noise correction. He notes that "it involves reordering the data to be transmitted, mixing symbols from different messages together into new messages of the same size, sending them through the channel, and then de-interleaving them back into the original messages at the receiver. This results in spreading out the bursty bits among multiple messages, increasing the likelihood that the number of errors will be less than the error-correcting capability of the ECC". As α decreases, error bursts become less likely, attributing to the "waterfall region" where the slope of error rate becomes very steep. Vasić explains that "at small SNR, the error probability decreases rapidly with SNR, with the curve looking like a waterfall". To illustrate the Viterbi algorithm performance for decoding a transmitted message over a binary symmetric channel the following graphs show the simulation results for different values of α.

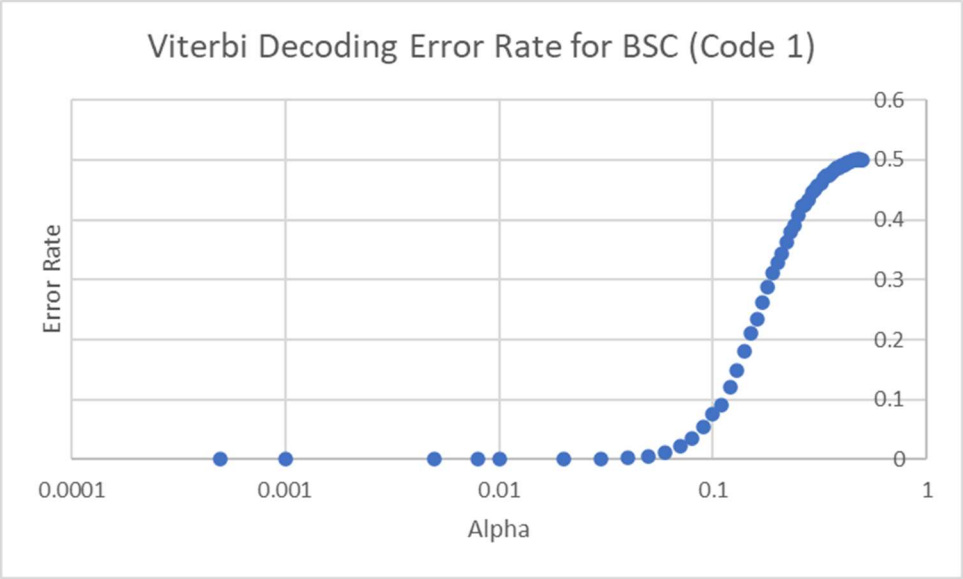

Figure 3: BSC Viterbi decoding performance on (2,1,3) code

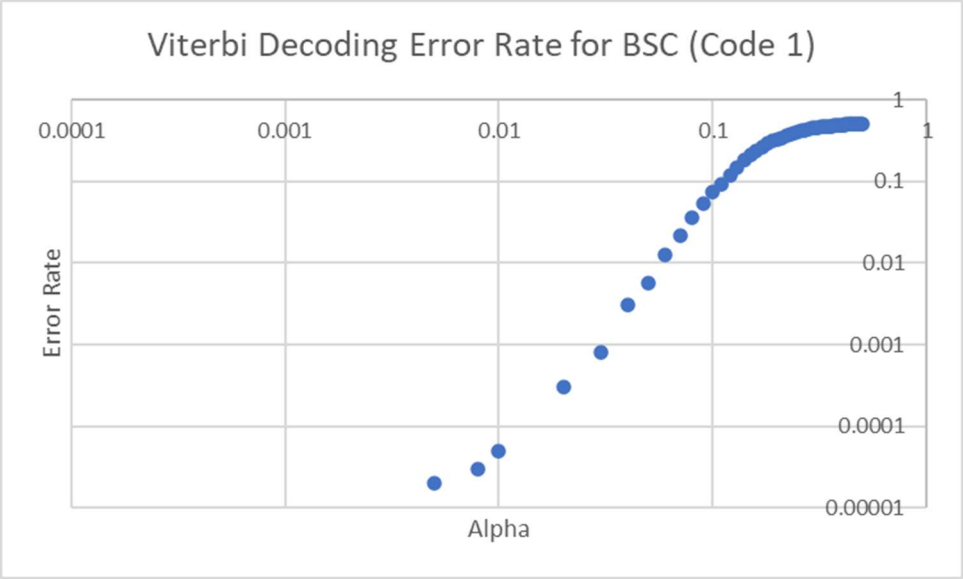

Figure 4: BSC Viterbi decoding performance on (2,1,3) code [log-log]

The simulation results of the Viterbi detection algorithm for the additive white gaussian noise channel expressed a high efficiency for error correction at larger values of SNR. The deterioration of performance at higher values of variance for the noise of the channel exhibited a

similar shape when compared to the results of the decoding for BSC. As mentioned previously, the significant decrease in efficiency for the Viterbi decoding algorithm can be attributed to the higher likelihood of error bursts leading to deviation from the correct path on the trellis. Graphs for the simulation results on the AWGN channel are provided below to show the performance of the Viterbi algorithm.

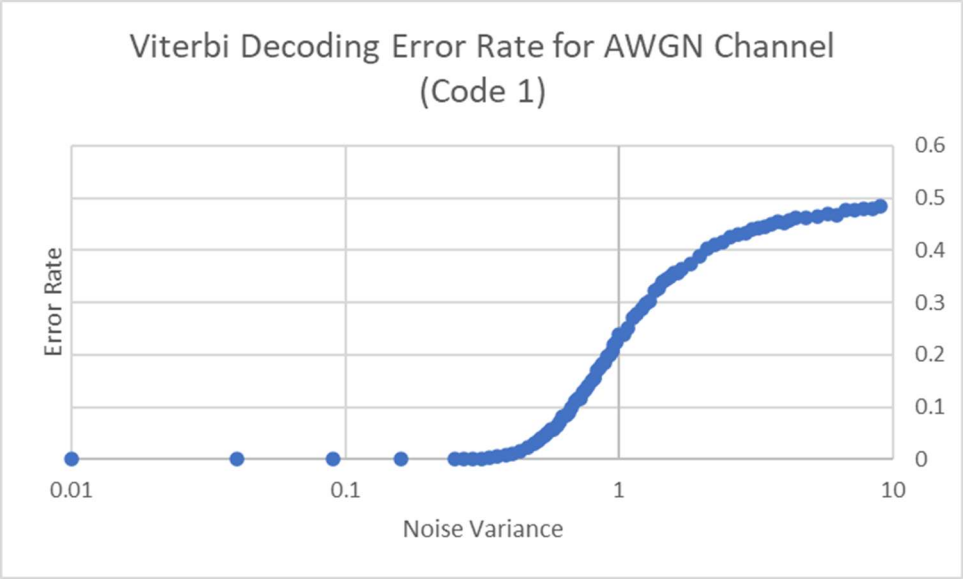

Figure 5: AWGN Viterbi decoding performance on (2,1,3) code

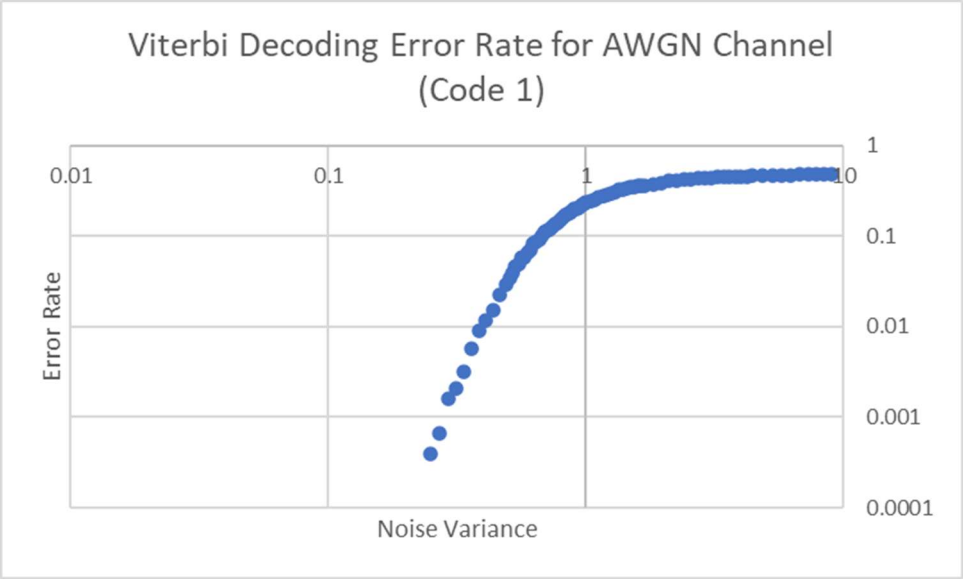

Figure 6: AWGN Viterbi decoding performance on (2,1,3) code [log-log]

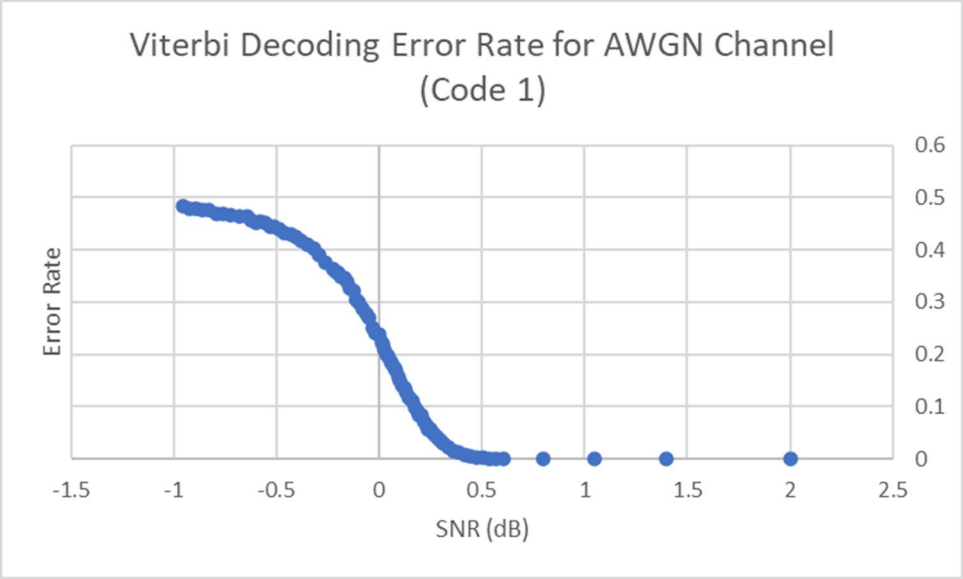

Figure 7: AWGN Viterbi decoding performance on (2,1,3) code

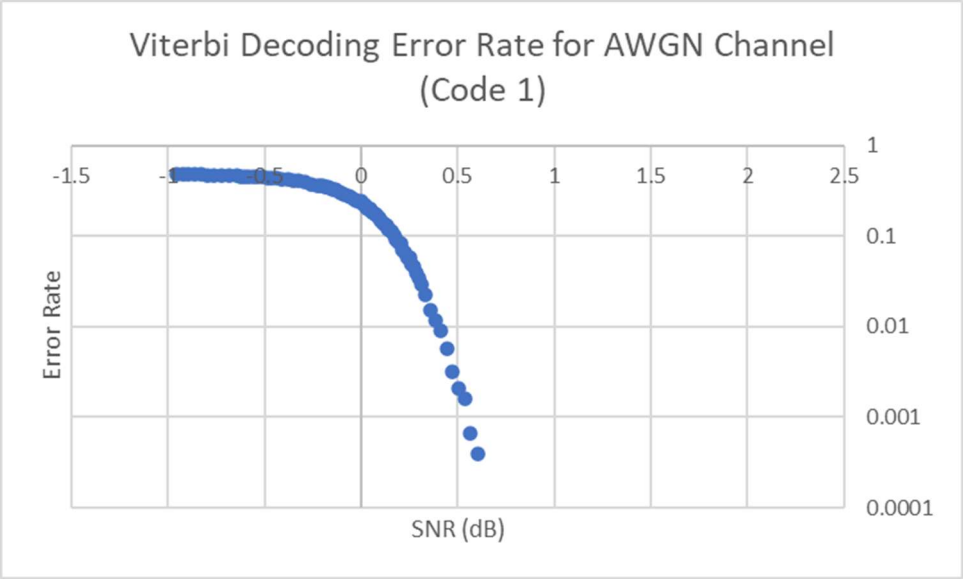

Figure 8: AWGN Viterbi decoding performance on (2,1,3) code [log-log]

The second encoder that I used for simulating the Viterbi algorithm decoding was a (2,1,4) convolutional encoder with the two generator polynomials:

$$g^{(1)} = g^{(2)} = x + 1$$

The encoder circuit for these polynomial generators uses one memory block and has a code rate of 1/2.

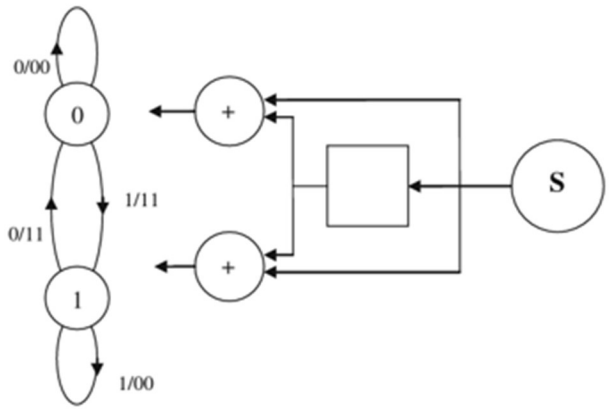

Figure 9: (2,1,4) binary convolutional encoder

To give visual representation of all possible states, transitions, and outputs of this encoder, a table and trellis structure is provided below.

u	S_n (m_1)	S_{n+1} (u)	V_1 $(u+m_1)$	V_2 $(u+m_1)$
0	0	0	0	0
1	0	1	1	1
0	1	0	1	1
1	1	1	0	0

Table 2: (2,1,4) convolutional code state machine table

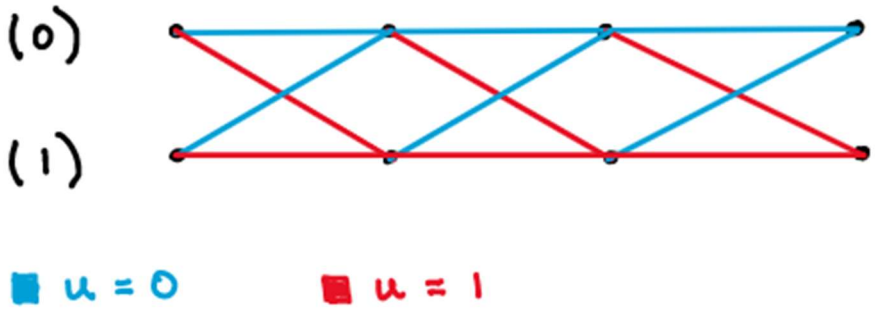

Figure 10: (2,1,4) convolutional code trellis structure

The simulation results of the Viterbi detection algorithm for the binary symmetric channel proved to be highly inefficient for all selected values of alpha. The cause for this becomes more evident if you analyze the structure of the code. The possible encoded outputs are 00 if two consecutive bits share the same value, and 11 if two consecutive bits have different values. An encoded output of 00 means that the state does not change, and 11 indicates that the state does change. This means that if there is a single bit error in a received encoded pair, the

Viterbi algorithm cannot determine if the state needs to be changed or not. This leads to a 50% probability of choosing an incorrect path on the trellis. If an incorrect path is chosen, the structure of this code will keep the Viterbi algorithm on the incorrect path until another bit error occurs in an encoded pair, and once again there will be a 50% probability of choosing an incorrect path. If we consider an encoded pair where both bits were flipped, this guarantees that the algorithm will choose the incorrect path. This helps to understand the poor performance of the Viterbi algorithm for this convolutional code. Below is a graph that exhibits the performance for the binary symmetric channel.

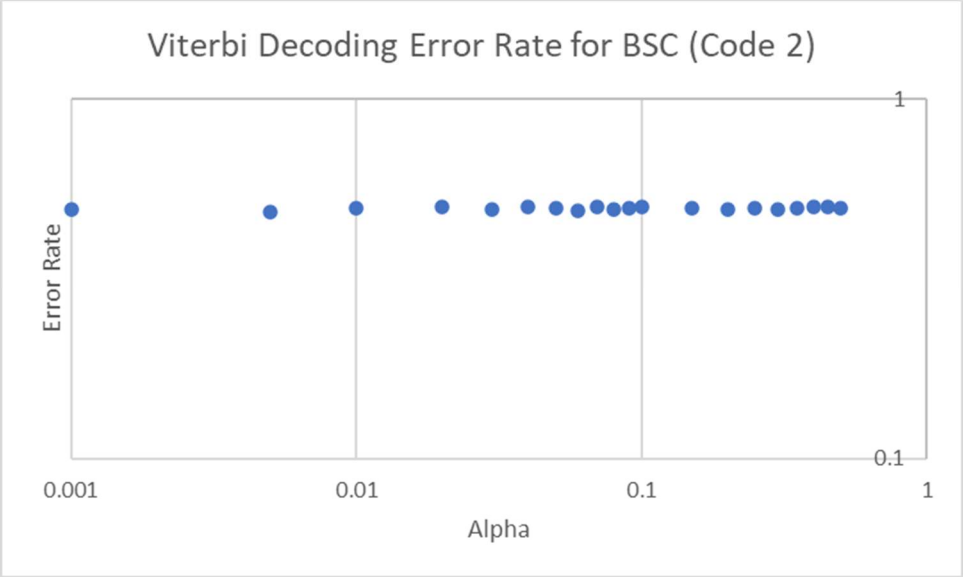

Figure 11: BSC Viterbi decoding performance on (2,1,4) code [log-log]

The simulation results of the Viterbi detection algorithm for the additive white gaussian noise channel also demonstrated an unsatisfactory performance. As stated before, this is a direct consequence of the convolutional code that was used in this simulation. Below are plots for the Viterbi decoding error rate for the AWGN channel.

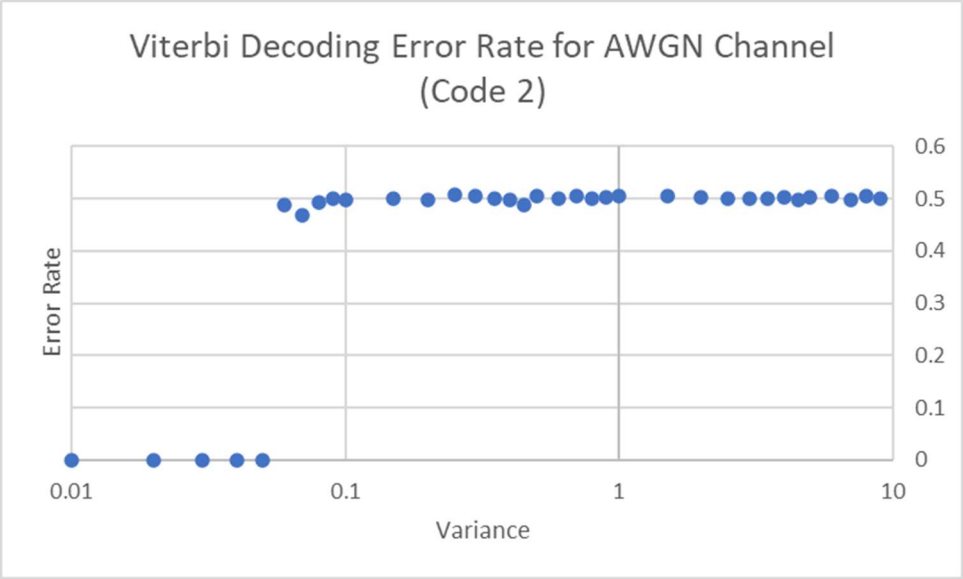

Figure 12: AWGN Viterbi decoding performance on (2,1,4) code

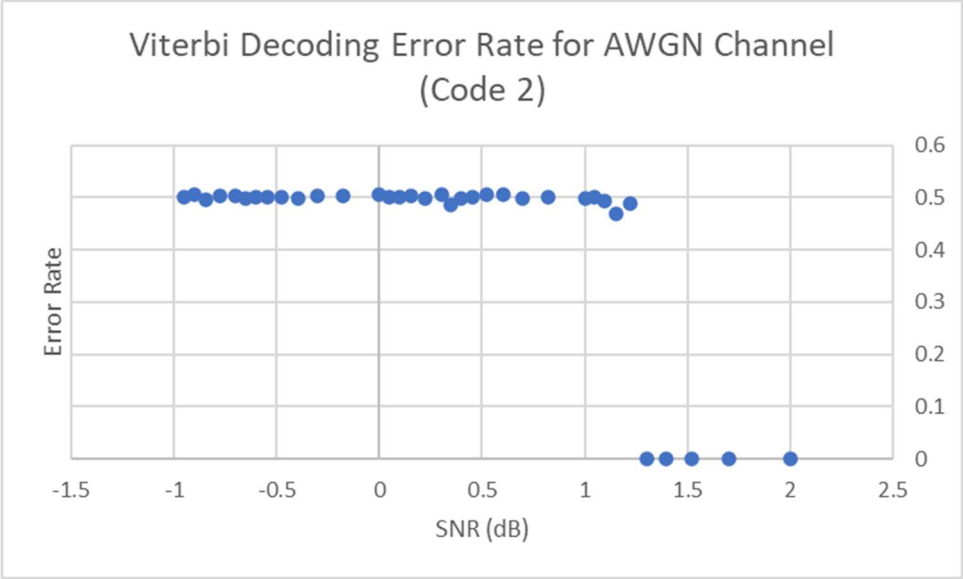

Figure 13: AWGN Viterbi decoding performance on (2,1,4) code

After performing simulations with both convolutional codes, it is extremely clear that they do not have the same performance. The structure of the second convolutional code leads to an approximate error rate of 50% for all values of α and σ^2. In the plots for the second code, the

data points that have a bit error rate of 0% are a result of no bit errors over the channel. The simulation procedure used had a message size of one hundred thousand bits meaning that two hundred thousand encoded bits were sent across the channel. As the message size increases, the bit error rate for those data points would approach 50%.

Simulation plays a crucial role in evaluating the performance of a communication system. After constructing the trellis diagrams for the given convolutional codes, and creating a software implementation of the Viterbi algorithm for error correction, it is apparent that the characteristics of the communication channel are important in determining the performance of the decoding algorithm. The first convolutional code proved to be very efficient for smaller values of α and σ^2. The performance of error correction degraded significantly as SNR decreased, hence, careful consideration must be taken in choosing an appropriate decoding technique based on the channel parameters. The second convolutional code demonstrated the consequence of an inefficient code structure. The structure of this code led to a high bit error rate and significantly degraded the quality of the received message. This demonstrates the importance of the structure for a convolutional code in determining the performance of the decoding algorithm.

Works Cited

Ivkovic, Milos, et al. “Eliminating Trapping Sets in Low-Density Parity-Check Codes by Using Tanner Graph Covers.” *IEEE Transactions on Information Theory*, vol. 54, no. 8, 2008, pp. 3763–3768., https://doi.org/10.1109/tit.2008.926319.

Mrutu, Salehe, et al. “Trellis Analysis of Transmission Burst Errors in Viterbi Decoding.” *International Journal of Computer Science and Information Security,* vol. 12, no. 8, 2014, pp. 46–53.

Neal, David A. “Utilizing Correct Prior Probability Calculation to Improve Performance of Low-Density Parity-Check Codes in the Presence of Burst Noise.” 2012.

Vasić, Bane, et al. “Failures and Error Floors of Iterative Decoders.” *Academic Press Library in Mobile and Wireless Communications*, 2014, pp. 299–341., https://doi.org/10.1016/b978-0-12-396499-1.00006-6.