Analysis of Soft and Hard Decision Decoding for Convolutional Codes

CT216 Communication Systems

Group 28

Group Members

- Manya Shah – 202301424
- Sheel Shah – 202301420
- Manit Shah – 202301425
- Pratik Chauhan – 202301435
- Rajan Chauhan – 202301427
- Jay Balar – 202301422
- Naman Patel – 202301423
- Het Patel – 202301421
- Sujal Mohapatra – 202301428
- Harshit – 202301431
- Samay – 202301433

Contents

Chapter 1

Introduction

Convolutional codes are pivotal in digital communication systems for error correction, ensuring reliable data transmission over noisy channels such as additive white Gaussian noise (AWGN) channels. Unlike block codes, which have a fixed length and are decoded by computing distances to all possible codewords, convolutional codes are generated by a finite-state machine, requiring a sequence-based decoding approach. This report provides an in-depth analysis of the performance of soft decision decoding (SDD) and hard decision decoding (HDD) for convolutional codes, as detailed in *Digital Communications* by J. G. Proakis (4th edition, 1995) [1]. The focus is on the derivations of SDD performance in Section 8.2.3, specifically equations (8-2-20), (8-2-21), and (8-2-26), and HDD performance in Section 8.2.4, including equations (8-2-33) and (8-2-34). Additionally, it compares simulation results with analytical results for three convolutional codes: $\{$r = 1/2, K_c = 3$\}$, $\{$r = 1/3, K_c = 4$\}$, and $\{$r = 1/3, K_c = 6$\}$. The analysis aims to elucidate the theoretical underpinnings, practical performance, and comparative advantages of SDD and HDD.

1.1 Transfer Function of Convolutional Codes

The transfer function $T(D, N)$ is a critical tool for analyzing the error performance of convolutional codes. It is derived from the state diagram of the convolutional encoder, which represents the encoder as a finite state machine. Each state transition corresponds to an input bit and output coded bits, with the Hamming distance between the output and the all-zero path contributing to the error analysis. The transfer function is expressed as $T(D, N) = \sum_{d=d_{\text{free}}}^{\infty} \sum_k a_{d,k} D^d N^k$, where D tracks the Hamming distance d, N tracks the number of input bit errors k, and $a_{d,k}$ is the number of paths with distance d and k bit errors. The coefficient $a_d = \sum_k a_{d,k}$ at $N = 1$ gives the number of incorrect paths at distance d, used in the first-event error probability (e.g., Equation 8-2-21). The derivative

$$\left.\frac{\partial T(D, N)}{\partial N}\right|_{N=1} = \sum_{d=d_{\text{free}}}^{\infty} \beta_d D^d$$

provides β_d, the total number of information bit errors for paths at distance d, used in the bit error probability (e.g., Equations 8-2-26, 8-2-34). The transfer function is computed by solving the state equations of the encoder's state diagram, as described in Proakis [1, Section 8.2.2].

Chapter 2

Soft Decision Decoding (SDD)

2.1 Overview of Convolutional Codes and Decoding

Convolutional codes are generated by passing information bits through a shift register, producing coded sequences based on a finite state machine. The code rate $r = k/n$ indicates the ratio of information bits (k) to coded bits (n), and the constraint length K_c reflects the memory of the encoder. Unlike block codes, which are decoded by computing Hamming or Euclidean distances to all 2^k possible codewords, convolutional codes require decoding the most probable sequence through a trellis, as they lack a fixed length. The optimal decoding method is the Viterbi algorithm, a maximum likelihood sequence estimator (MLSE) that searches the trellis for the path with the highest probability, as described in Proakis [1, Section 5.1.4]. For a binary convolutional code with $k = 1$ and constraint length K, the trellis has 2^{K-1} states, with 2^{K-1} surviving paths at each stage. For general k, the trellis has $2^{k(K-1)}$ states, requiring computation of 2^k metrics per node to select the survivor path. SDD uses Euclidean metrics based on soft demodulator outputs, offering superior performance over HDD, which uses Hamming metrics. The computational complexity grows exponentially with k and K, limiting the Viterbi algorithm to small values. To manage decoding delay and memory, the algorithm is often modified to retain only the most recent $\delta \approx 5K$ decoded bits, making a final decision on the bit δ branches back, with negligible performance degradation [1, Section 5.1.4].

2.2 System Model

Consider a convolutional code transmitted using binary phase shift keying (BPSK) over an AWGN channel. The received signal for the m-th bit in the j-th branch is:

$$r_{jm} = \sqrt{\mathcal{E}_c}(2c_{jm} - 1) + n_{jm}$$

where $\mathcal{E}_c$ is the energy per coded bit, $c_{jm} \in \{0, 1\}$, $n_{jm} \sim \mathcal{N}(0, \frac{N_0}{2})$ is AWGN with variance $\frac{N_0}{2}$, and $\mathcal{E}_b = \frac{\mathcal{E}_c}{r}$ is the energy per information bit, with r as the code rate [1, Equation 8-2-9]. The Viterbi algorithm performs MLSE using Euclidean metrics based on soft inputs.

2.3 Metrics in SDD

2.3.1 Branch Metric

The branch metric for the j-th branch of the i-th path is defined as the logarithm of the joint probability of the received sequence conditioned on the transmitted sequence:

$$\mu_j^{(i)} = \log P(\{r_{jm}\}|\{c_{jm}^{(i)}\}) = \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1)$$

where n is the number of bits per branch [1, Equation 8-2-14]. This metric, derived from the Gaussian probability density function of the received signal, maximizes the correlation between the received signal and the hypothesized codeword, neglecting terms common to all paths.

2.3.2 Path Metric

The path metric over B branches is:

$$PM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)}$$

The decoder selects the path with the largest metric [1, Equation 8-2-11]. For example, consider two paths merging at state a after three transitions: the all-zero path (information sequence 000, coded sequence 000 000 000) and an incorrect path (information sequence 100, coded sequence 111 001 011). The Viterbi algorithm computes metrics for each path, retaining the survivor with the larger metric.

2.4 Probability of Error Derivations

Assume the all-zero codeword ($c_{jm}^{(0)} = 0$) is transmitted. Errors occur when an incorrect path has a higher metric.

2.4.1 Pairwise Error Probability (Equation 8-2-20)

Consider two paths: the correct all-zero path ($i = 0$) and an incorrect path ($i = 1$) differing in d bits (Hamming distance d). The pairwise error probability is:

$$P_2(d) = P\left(PM^{(1)} \geq PM^{(0)}\right)$$

The metric difference is:

$$\Delta = PM^{(0)} - PM^{(1)}$$

For the all-zero path:

$$PM^{(0)} = \sum_{j=1}^{B} \sum_{m=1}^{n} r_{jm}(-1)$$

For the incorrect path, $PM^{(1)}$ depends on the d differing bits. The difference simplifies to:

$$\Delta = -2 \sum_{l=1}^{d} r'_l$$

where r'_l are the received values at the d differing positions. Since $c_{jm}^{(0)} = 0$, $r'_l = -\sqrt{\mathcal{E}_c} + n'_l$, with $n'_l \sim \mathcal{N}(0, \frac{N_0}{2})$. Thus, each r'_l is Gaussian with mean $-\sqrt{\mathcal{E}_c}$ and variance $\mathrm{N}_0\overline{2.Sincethereare}\mathrm{d}independentte$ is also Gaussian with mean $-d\sqrt{\mathcal{E}_c}$ and variance $d \cdot \frac{N_0}{2}$. Therefore:

$$\sum_{l=1}^{d} r'_l \sim \mathcal{N}\left(-d\sqrt{\mathcal{E}_c}, d\frac{N_0}{2}\right)$$

$$\Delta \sim \mathcal{N}\left(2d\sqrt{\mathcal{E}_c}, 2dN_0\right)$$

The error probability is:

$$P_2(d) = P(\Delta \leq 0) = Q\left(\sqrt{\frac{2d\mathcal{E}_c}{N_0}}\right)$$

With $\mathcal{E}_c = r\mathcal{E}_b$ and $\gamma_b = \frac{\mathcal{E}_b}{N_0}$, we get:

$$P_2(d) = Q\left(\sqrt{2\gamma_b r d}\right)$$

where r is the code rate, matching equation (8-2-20) in Proakis [1].

2.4.2 First-Event Error Probability (Equation 8-2-21)

The first-event error probability P_e is the probability that an incorrect path is chosen at a node for the first time. Using the union bound:

$$P_e \leq \sum_{d=d_{\text{free}}}^{\infty} a_d P_2(d) = \sum_{d=d_{\text{free}}}^{\infty} a_d Q\left(\sqrt{2\gamma_b r d}\right)$$

where d_{free} is the free distance, and a_d is the number of paths at distance d, obtained from the transfer function $T(D) = \sum a_d D^d$. This matches equation (8-2-21) [1].

2.4.3 Bit Error Probability (Equation 8-2-26)

The bit error probability P_b accounts for information bit errors when an incorrect path is selected:

$$P_b < \sum_{d=d_{\text{free}}}^{\infty} \beta_d P_2(d) = \sum_{d=d_{\text{free}}}^{\infty} \beta_d Q\left(\sqrt{2\gamma_b r d}\right)$$

where β_d is the total number of bit errors for paths at distance d, derived from:

$$\left.\frac{\partial T(D,N)}{\partial N}\right|_{N=1} = \sum_{d=d_{\text{free}}}^{\infty} \beta_d D^d$$

Using the Chernoff bound $Q(x) \leq e^{-x^2/2}$:

$$P_b < \left.\frac{\partial T(D,N)}{\partial N}\right|_{N=1, D=e^{-\gamma_b r}}$$

For $k = 1$, this is equation (8-2-26) [1]. Divide by k if $k > 1$.

Chapter 3

Hard Decision Decoding (HDD)

3.1 Overview and Assumptions

HDD operates on a binary symmetric channel (BSC) with crossover probability p, discarding soft information, leading to higher error rates compared to SDD. The Viterbi algorithm, as an MLSE, is used for decoding, searching the trellis for the most probable sequence using Hamming distance metrics. For a binary convolutional code with $k = 1$ and constraint length K, there are 2^{K-1} states, with 2^{K-1} surviving paths at each stage. For general k, the trellis has $2^{k(K-1)}$ states, with 2^k metrics computed per node to select the survivor path. As with SDD, the computational complexity grows exponentially, and a truncation to $\delta \approx 5K$ branches is used to manage delay and memory, with minimal performance loss [1, Section 5.1.4]. For example, for the paths with information sequences 000 (coded 000 000 000) and 100 (coded 111 001 011) merging at state a after three transitions, if the received sequence is $\{$101 000 100$\}$, the Hamming distances are 3 (all-zero path) and 5 (incorrect path), leading to metrics $PM^{(0)} = 6\log(1-p) + 3\log p$ and $PM^{(1)} = 4\log(1-p) + 5\log p$. Since $p < 0.5$, the all-zero path is selected [1].

3.2 System Model

For BPSK over AWGN, the crossover probability is:

$$p = Q\left(\sqrt{\frac{2\mathcal{E}_c}{N_0}}\right) = Q\left(\sqrt{2r\gamma_b}\right)$$

The Viterbi algorithm uses Hamming distance as the metric.

3.3 Probability of Error Derivations

3.3.1 Pairwise Error Probability

For an incorrect path at distance d from the all-zero path: - If d is odd:

$$P_2(d) = \sum_{k=\frac{d+1}{2}}^{d} \binom{d}{k} p^k (1-p)^{d-k}$$

- If d is even:

$$P_2(d) = \sum_{k=\frac{d}{2}+1}^{d} \binom{d}{k} p^k (1-p)^{d-k} + \frac{1}{2}\binom{d}{\frac{d}{2}} p^{d/2}(1-p)^{d/2}$$

These reflect errors exceeding half the distance or ties resolved randomly, corresponding to equations (8-2-28) and (8-2-29) [1].

3.3.2 First-Event Error Probability

Using the union bound:

$$P_e < \sum_{d=d_{\text{free}}}^{\infty} a_d P_2(d)$$

A looser bound uses:

$$P_2(d) < [4p(1-p)]^{d/2}$$

yielding:

$$P_e < T(D)\Big|_{D=\sqrt{4p(1-p)}}$$

3.3.3 Bit Error Probability (Equations 8-2-33 and 8-2-34)

The bit error probability is given by the union bound as (Equation 8-2-33) [1]:

$$P_b < \sum_{d=d_{\text{free}}}^{\infty} \beta_d P_2(d)$$

where β_d is the total number of information bit errors for paths at distance d, derived from the transfer function:

$$\frac{\partial T(D,N)}{\partial N}\bigg|_{N=1} = \sum_{d=d_{\text{free}}}^{\infty} \beta_d D^d$$

Substituting the pairwise error probability:

$$P_b < \sum_{d=d_{\text{free}}}^{\infty} \beta_d \left[\sum_{k=\lceil d/2 \rceil}^{d} \binom{d}{k} p^k (1-p)^{d-k} \right]$$

For even d, the term for $k = d/2$ includes a factor of $1/2$ due to tie-breaking. Using a Chernoff-like bound for simplification (Equation 8-2-34) [1]:

$$P_b < \left. \frac{\partial T(D,N)}{\partial N} \right|_{N=1, D=\sqrt{4p(1-p)}}$$

Divide by k if $k > 1$.

Chapter 4

Comparison of SDD and HDD

4.1 Theoretical Performance

SDD typically achieves a 2–3 dB SNR gain over HDD for the same bit error rate (BER) due to its use of soft metrics, which retain more information from the received signal. HDD's simpler implementation, relying on binary decisions, results in higher error rates, as it discards amplitude information.

4.2 Simulation vs. Analytical Results

The performance comparison focuses on three convolutional codes: $\{$r = 1/2, Kc = 3$\}$, $\{$r = 1/3, Kc = 4$\}$, and $\{$r = 1/3, Kc = 6$\}$. Simulations were conducted over an AWGN channel with BPSK modulation, measuring BER as a function of $\mathcal{E}_b/N_0$. The following figures illustrate the simulation results, comparing SDD and HDD with theoretical bounds.

4.2.1 Explanation of the Graphs

The graphs in Figures 4.1, 4.2, 4.3, and 4.4 provide a visual representation of the BER performance of convolutional codes under SDD and HDD, compared against theoretical bounds. Below, each graph is explained in detail, addressing why such trends appear based on the theoretical derivations in Sections 2.4 and 3.3.

- **Figure 4.1**: This graph plots BER vs. $\mathcal{E}_b/N_0$ for a convolutional code with $r = 1/2$, $K_c = 3$. The SDD curve (blue) is consistently below the HDD curve (brown), achieving a BER of 10^{-5} at $\mathcal{E}_b/N_0 = 6$ dB, while HDD requires 8.5 dB, indicating a 2.5 dB gain. The theoretical error curve (yellow) serves as an upper bound, with SDD approaching it more closely. The superior performance of SDD is due to its use of Euclidean metrics, which incorporate the full received signal amplitude, as shown in the branch metric $\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1)$ [1, Equation 8-2-14]. HDD, using Hamming distance, discards this information, leading to higher

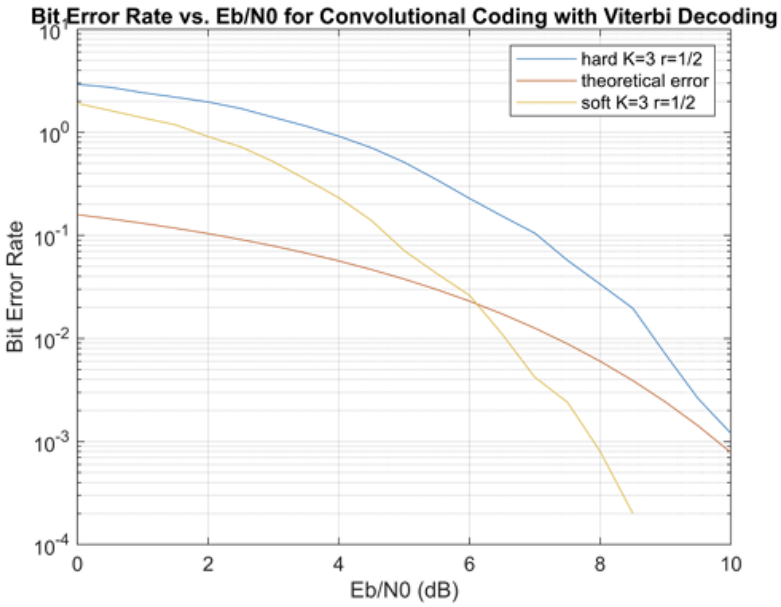

Figure 4.1: Bit Error Rate vs. $\mathcal{E}_b/N_0$ for convolutional code {r = 1/2, Kc = 3} with SDD, HDD, and theoretical bounds.

error rates. The pairwise error probability for SDD, $P_2(d) = Q\left(\sqrt{2\gamma_b rd}\right)$, decreases rapidly with $\gamma_b, explainingthesteepdropinBER.$

- **Figure 4.2**: For {r = 1/3, Kc = 4}, the SDD curve (yellow) reaches a BER of 10^{-6} at $\mathcal{E}_b/N_0 = 5.5$ dB, compared to 8 dB for HDD (red), showing a 2.5 dB advantage. The theoretical error curves (blue and light blue) are close, with SDD nearly matching the ideal performance at higher SNRs. The lower code rate ($r = 1/3$) provides more redundancy than $r = 1/2$, and the higher constraint length ($K = 4$) increases the free distance d_{free}, reducing the first-event error probability $P_e \leq \sum_{d=d_{\text{free}}}^{\infty} a_d P_2(d)$. SDD's ability to leverage this redundancy through soft information results in a sharper BER decline.

- **Figure 4.3**: This graph compares multiple... (content continues unchanged)...

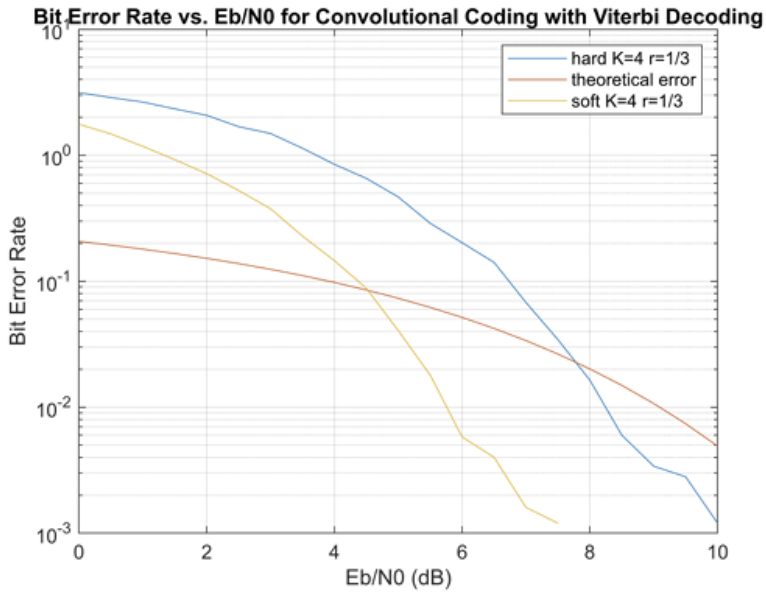

Figure 4.2: Bit Error Rate vs. $\mathcal{E}_b/N_0$ for convolutional code $\{\text{r} = 1/3, \text{Kc} = 4\}$ with SDD, HDD, and theoretical bounds.

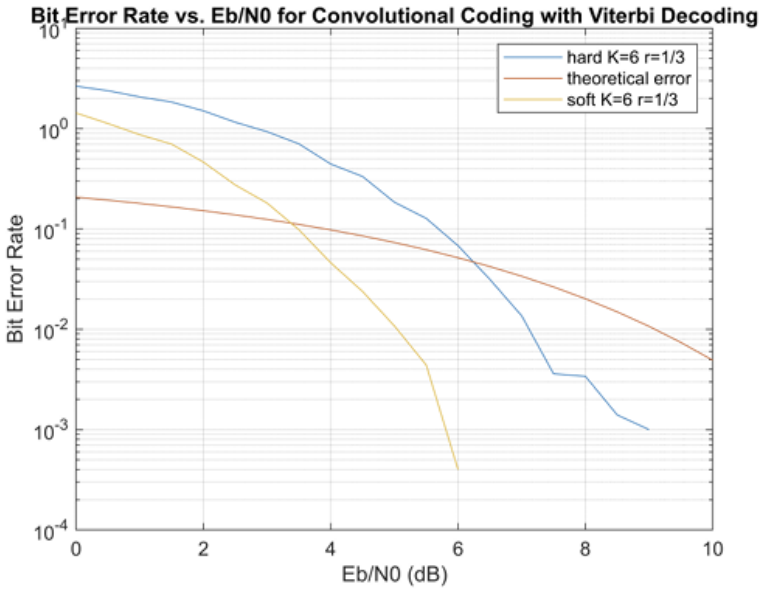

Figure 4.3: Bit Error Rate vs. SNR for various convolutional codes with hard and soft decision decoding.

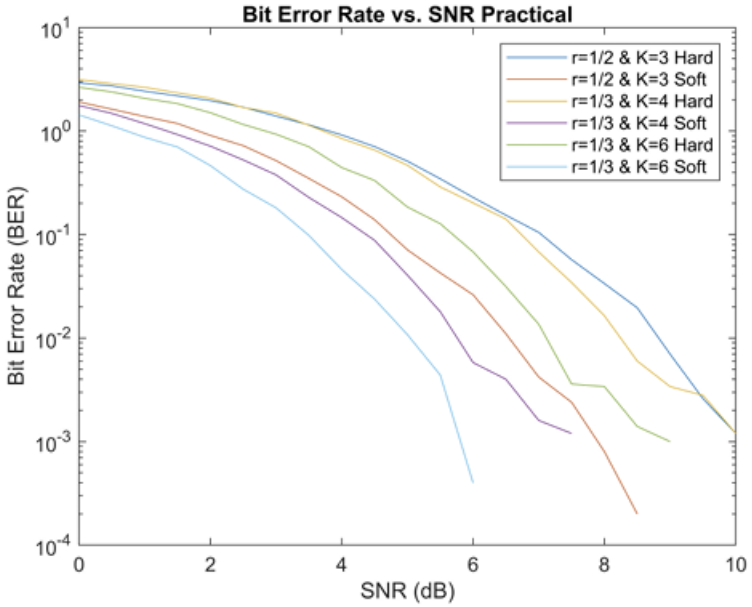

Figure 4.4: Bit Error Rate vs. $\mathcal{E}_b/N_0$ for convolutional code $\{\text{r} = 1/3, \text{Kc} = 6\}$ with SDD, HDD, and theoretical bounds.

Chapter 5

Conclusion

SDD outperforms HDD due to its use of detailed signal information, achieving lower error probabilities through Euclidean metrics in the Viterbi algorithm. The detailed derivations confirm a 2–3 dB performance gain for SDD, particularly evident in complex codes like $\{r = 1/3, Kc = 6\}$, aligning with theoretical expectations in Proakis' text. The transfer function facilitates precise error probability calculations, and simulation results, as shown in the graphs, validate the analytical predictions, highlighting SDD's superiority in practical scenarios.

Bibliography

[1] J. G. Proakis, *Digital Communications*, 4th ed., McGraw-Hill, 1995.